Skriptsyntax och diagramfunktioner

Qlik Sense®
May 2026
Copyright © 1993-åååå QlikTech International AB. Alla rättigheter förbehållna.

© åååå QlikTech International AB. All rights reserved. All company and/or product names may be trade names, trademarks and/or registered trademarks of the respective owners with which they are associated.

Contents

1 Vad är Qlik Sense? 17
1.1 Vad kan du göra i Qlik Sense? 17
1.2 Hur fungerar Qlik Sense? 17
Appmodellen 17
Den associativa analysen 17
Samarbete och mobilitet 17
1.3 Hur distribuerasQlik Sense? 17
Qlik Sense Desktop 18
Qlik Sense Enterprise 18
1.4 Så här administrerar och hanterar du en Qlik Sense-plats 18
1.5 Utöka Qlik Sense och anpassa det efter dina syften 18
Bygga komplement och kombinationsprogram 18
Bygga klienter 18
Bygga serververktyg 18
Koppla till andra datakällor 18
2 Översikt över skriptsyntax 19
2.1 Introduktion till skriptsyntax 19
2.2 Vad är Backus-Naur-formalism? 19
3 Skriptsatser och nyckelord 21
3.1 Kontrollsatser i skriptet 21
Översikt av kontrollsatser i skriptet 21
Call 23
Do-loop 24
End 25
Exit 25
Exit script 25
For..next 26
For each..next 28
If..then..elseif..else..end if 30
Next 31
Sub..end sub 32
Switch..case..default..end switch 33
To 34
3.2 Skriptprefix 34
Skriptprefix – en översikt 34
Add 38
Buffer 40
Concatenate 41
Crosstable 47
First 56
Generic 58
Hierarchy 64
HierarchyBelongsTo 66
Inner 68
IntervalMatch 69
Join 73
Keep 94

Left ... 95
Mappning ... 97
Slå samman ... 99
NoConcatenate ... 103
Only ... 112
Outer ... 112
Delvis laddning ... 113
Replace ... 116
Right ... 118
Sample ... 119
Semantic ... 122
Unless ... 126
When ... 132
3.3 Vanliga skriptsatser ... 138
Översikt av vanliga skriptsatser ... 138
Alias ... 145
AutoNumber ... 146
Binary ... 149
Comment field ... 150
Comment table ... 151
Connect ... 152
Constrain ... 153
Declare ... 163
Derive ... 165
Direct Query ... 166
Directory ... 171
Disconnect ... 172
Drop ... 172
Drop table ... 176
Execute ... 178
Field/Fields ... 179
FlushLog ... 179
Force ... 179
From ... 181
Load ... 182
Let ... 201
Loosen Table ... 201
Map ... 202
NullAsNull ... 203
NullAsValue ... 203
Qualify ... 204
Rem ... 205
Rename ... 206
Search ... 208
Section ... 209
Select ... 209
Set ... 211
Sleep ... 212

SQL 212
SQLColumns 213
SQLTables 214
SQLTypes 215
Star 216
Store 218
Table/Tables 224
Tag 224
Trace 225
Unmap 226
Unqualify 226
Untag 227
Variable/Variables 228
3.4 Arbetskatalog 228
Qlik Sense Desktop-arbetskatalog 228
Qlik Sense-arbetskatalog 228
4 Arbeta med variabler i Skriptredigeraren 229
4.1 Översikt 229
4.2 Definiera en variabel 229
Namnge dina variabler 230
4.3 Ytterligare alternativ vid definition av variabler 230
Begränsa variabelvärden 230
Definiera skriptspecifika variabler 230
Släppa variabler 230
4.4 Ta bort en variabel 230
4.5 Ladda ett variabelvärde som ett fältvärde 231
4.6 Beräkning med variabler 231
4.7 Systemvariabler 232
Översikt av systemvariabler 232
CreateSearchIndexOnReload 235
HidePrefix 235
HideSuffix 236
Include 236
ScriptOnlyVariables 237
OpenUrlTimeout 238
StripComments 239
Verbatim 239
4.8 Variabler för värdehantering 239
Översikt av variabler för värdehantering 239
NullDisplay 240
NullInterpret 240
NullValue 241
OtherSymbol 241
4.9 Variabler för tolkning av tal 241
Valutaformat 242
Talformat 242
Tidsformat 243

BrokenWeeks 244
DateFormat 245
DayNames 251
DecimalSep 255
FirstWeekDay 257
LongDayNames 262
LongMonthNames 265
MoneyDecimalSep 269
MoneyFormat 273
MoneyThousandSep 277
MonthNames 281
NumericalAbbreviation 286
ReferenceDay 287
ThousandSep 291
TimeFormat 297
TimestampFormat 298
4.10 Variabler i Direkt upptäckt 301
Systemvariabler i Direkt upptäckt 301
Variabler för Teradata-querybanding 302
Direkt upptäckt-teckenvariabler 303
Direkt upptäckt-variabler för tolkning av tal 304
4.11 Felvariabler 305
Felvariabler – en översikt 305
ErrorMode 305
ScriptError 306
ScriptErrorCount 307
ScriptErrorList 307
5 Skriptuttryck 308
6 Diagramuttryck 309
6.1 Definiera aggregeringens omfattning 309
Exempel: TOTAL-kvalificeraren 310
Exempel: set-analys 311
Exempel: kvalificeraren TOTAL och set-analys ({1}-identifierare) 311
Exempel: TOTAL kvalificerare och set-analys (jämförelse av data mot ett referensvärde) 312
Exempel: TOTAL-kvalificerare och set-analys 313
Data som används i exempel på den här sidan 314
6.2 Set-analys 314
Set-uttryck 314
Exempel 316
Naturliga uppsättningar 316
Set-identifierare 318
Set-operatorer 319
Set-modifierare 320
Flagga för att bevara tom mängd 342
Inre och yttre uppsättningsuttryck 343
Yttre set-uttryck med dollarteckenexpansioner 348
Introduktion – Skapa ett set-uttryck 350
Syntax för set-uttryck 360

6.3 Allmän syntax för diagramuttryck 360
6.4 Allmän syntax för aggregeringar 361
7 Operatorer 362
7.1 Bit-operatorer 362
7.2 Logiska operatorer 363
7.3 Numeriska operatorer 363
7.4 Relationsoperatorer 364
7.5 Strängoperatorer 365
& 366
like 366
8 Skript- och diagramfunktioner 367
8.1 Analytiska kopplingar för komplement på serversidan (SSE) 367
8.2 Aggregeringsfunktioner 367
Använda aggregeringsfunktioner i ett dataladdningsskript 368
Använda aggregeringsfunktioner i diagramuttryck 368
Så beräknas aggregeringar 368
Aggregering av nyckelfält 368
Grundläggande aggregeringsfunktioner 369
Räkneaggregeringsfunktioner 406
Finansiella aggregeringsfunktioner 433
Statistiska aggregeringsfunktioner 462
Statistiska testfunktioner 532
Strängaggregeringsfunktioner 596
Syntetiska dimensionsfunktioner 609
Nästlade aggregeringar 612
8.3 Aggr - diagramfunktion 612
Exempel: diagramuttryck som använder Aggr 615
8.4 Färgfunktioner 619
Fördefinierade färgfunktioner 621
ARGB 622
RGB 627
HSL 633
8.5 Villkorsfunktioner 638
Villkorsfunktioner – en översikt 638
alt 639
class 646
coalesce 651
if 655
match 662
mixmatch 669
pick 677
wildmatch 682
8.6 Räknefunktioner 690
Räknefunktioner - en översikt 690
autonumber 691
autonumberhash128 693
autonumberhash256 695

IterNo 698
RecNo 698
RowNo 700
RowNo - diagramfunktion 701
8.7 Datum- och tidsfunktioner 704
Datum- och tidsfunktioner – en översikt 704
addmonths 713
addyears 723
age 730
converttolocaltime 732
day 736
dayend 742
daylightsaving 750
dayname 750
daynumberofquarter 753
daynumberofyear 759
daystart 765
firstworkdate 773
GMT 774
hour 778
inday 782
indaytotime 790
inlunarweek 800
inlunarweektodate 812
inmonth 824
inmonths 831
inmonthstodate 845
inmonthtodate 858
inquarter 868
inquartertodate 881
inweek 894
inweektodate 910
inyear 924
inyeartodate 936
lastworkdate 949
localtime 958
lunarweekend 961
lunarweekname 973
lunarweekstart 986
makedate 998
maketime 1004
makeweekdate 1011
minute 1019
month 1025
monthend 1031
monthname 1041
monthsend 1048
monthsname 1061

monthsstart 1074
monthstart 1087
networkdays 1097
now 1107
quarter 1114
quarterend 1119
quartername 1133
quarterstart 1144
second 1156
setdateyear 1161
setdateyearmonth 1163
timezone 1165
today 1166
UTC 1171
week 1172
weekday 1188
weekend 1196
weekname 1208
weekstart 1223
weekyear 1235
year 1244
yearend 1251
yearname 1263
yearstart 1275
yeartodate 1287
8.8 Exponentiella och logaritmiska funktioner 1303
8.9 Fältfunktioner 1304
Räknefunktioner 1304
Fält- och urvalsfunktioner 1305
GetAlternativeCount - diagramfunktion 1305
GetCurrentSelections - diagramfunktion 1311
GetExcludedCount - diagramfunktion 1316
GetFieldSelections - diagramfunktion 1320
GetObjectDimension - diagramfunktion 1325
GetObjectField - diagramfunktion 1328
GetObjectMeasure - diagramfunktion 1332
GetPossibleCount - diagramfunktion 1334
GetSelectedCount - diagramfunktion 1338
GetStateCounts - diagramfunktion 1342
8.10 Filfunktioner 1346
Filfunktioner – en översikt 1346
Attribute 1348
ConnectString 1356
FileBaseName 1356
FileDir 1356
FileExtension 1357
FileName 1357
FilePath 1357

FileSize 1358
FileTime 1359
GetFolderPath 1359
QvdCreateTime 1360
QvdFieldName 1361
QvdNoOfFields 1362
QvdNoOfRecords 1363
QvdTableName 1364
8.11 Finansiella funktioner 1365
Finansiella funktioner – en översikt 1365
BlackAndSchole 1366
FV 1367
nPer 1368
Pmt 1368
PV 1369
Rate 1370
8.12 Formateringsfunktioner 1371
Formateringsfunktioner - en översikt 1371
ApplyCodepage 1373
Date 1374
Dual 1377
Interval 1382
Money 1386
Num 1391
Time 1396
Timestamp 1400
8.13 Allmänna numeriska funktioner 1405
Allmänna numeriska funktioner – en översikt 1406
Kombinations- och permutationsfunktioner 1406
Modulofunktioner 1407
Paritetsfunktioner 1407
Avrundningsfunktioner 1407
BitCount 1408
Ceil 1408
Combin 1409
Div 1410
Even 1410
Fabs 1411
Fact 1411
Floor 1412
Fmod 1413
Frac 1414
Mod 1414
Odd 1415
Permut 1415
Round 1416
Sign 1417
8.14 Geospatiala funktioner 1418

Översikt över geospatiala funktioner 1418
GeoAggrGeometry 1420
GeoBoundingBox 1422
GeoCountVertex 1422
GeoGetBoundingBox 1423
GeoGetPolygonCenter 1423
GeoInvProjectGeometry 1424
GeoMakePoint 1424
GeoProject 1425
GeoProjectGeometry 1426
GeoReduceGeometry 1426
MakeGeoLine 1427
MakeGeoPoint 1428
MakeGeoPolygon 1428
MakeGeoRegion 1429
8.15 Tolkningsfunktioner 1430
Tolkningsfunktioner - en översikt 1430
Date# 1431
Interval# 1434
Money# 1438
Num# 1442
Text 1447
Time# 1450
Timestamp# 1453
8.16 Postöverskridande funktioner 1456
Radfunktioner 1456
Kolumnfunktioner 1457
Fältfunktioner 1458
Pivottabellfunktioner 1458
Postöverskridande funktioner i dataladdningsskriptet 1459
Above - diagramfunktion 1459
Below - diagramfunktion 1473
Bottom - diagramfunktion 1483
Column - diagramfunktion 1490
Dimensionality - diagramfunktion 1495
Exists 1499
FieldIndex 1503
FieldValue 1505
FieldValueCount 1506
LookUp 1508
NoOfRows 1510
Peek 1512
Previous 1519
Top - diagramfunktion 1521
SecondaryDimensionality - diagramfunktion 1528
After - diagramfunktion 1531
Before - diagramfunktion 1536
First - diagramfunktion 1540

Last - diagramfunktion 1546
ColumnNo - diagramfunktion 1551
NoOfColumns - diagramfunktion 1555
8.17 Logiska funktioner 1558
8.18 Mappningsfunktioner 1559
Mappningsfunktioner – en översikt 1559
ApplyMap 1560
MapSubstring 1561
8.19 Matematiska funktioner 1563
8.20 NULL-funktioner 1564
NULL-funktioner – en översikt 1564
EmptyIsNull 1564
IsNull 1569
NULL 1573
8.21 Intervallfunktioner 1576
Grundläggande intervallfunktioner 1577
Räkneintervallfunktioner 1578
Statistiska intervallfunktioner 1578
Finansiella intervallfunktioner 1579
RangeAvg 1579
RangeCorrel 1582
RangeCount 1584
RangeFractile 1587
RangeIRR 1589
RangeKurtosis 1591
RangeMax 1591
RangeMaxString 1594
RangeMin 1595
RangeMinString 1598
RangeMissingCount 1599
RangeMode 1601
RangeNPV 1604
RangeNullCount 1605
RangeNumericCount 1607
RangeOnly 1608
RangeSkew 1609
RangeStdev 1611
RangeSum 1612
RangeTextCount 1614
RangeXIRR 1616
RangeXNPV 1617
8.22 Relaterade funktioner 1620
Rangordningsfunktioner 1620
Klustringsfunktioner 1620
Funktioner för uppdelning av tidsserier 1621
Rank - diagramfunktion 1622
HRank - diagramfunktion 1627
Optimering med k-medelvärde: ett exempel från verkligheten 1629

KMeans2D - diagramfunktion ... 1638
KMeansND - diagramfunktion ... 1653
KMeansCentroid2D - diagramfunktion ... 1668
KMeansCentroidND - diagramfunktion ... 1669
STL_Trend - diagramfunktion ... 1670
STL_Seasonal - diagramfunktion ... 1672
STL_Residual - diagramfunktion ... 1674
Introduktionskurs – upplösning av tidsserie i Qlik Sense ... 1676
8.23 Statistiska fördelningsfunktioner ... 1680
Översikt av statistiska fördelningsfunktioner ... 1681
BetaDensity ... 1683
BetaDist ... 1683
BetaInv ... 1684
BinomDist ... 1684
BinomFrequency ... 1685
BinomInv ... 1685
ChiDensity ... 1686
ChiDist ... 1686
ChiInv ... 1687
FDensity ... 1688
FDist ... 1688
FInv ... 1689
GammaDensity ... 1689
GammaDist ... 1690
GammaInv ... 1690
NormDist ... 1691
NormInv ... 1692
PoissonDist ... 1692
PoissonFrequency ... 1693
PoissonInv ... 1693
TDensity ... 1694
TDist ... 1694
TInv ... 1695
8.24 Strängfunktioner ... 1695
Strängfunktioner - översikt ... 1695
Arbeta med reguljära uttryck i laddningsskript och diagramuttryck ... 1700
Capitalize ... 1702
Chr ... 1705
CountRegEx ... 1707
Evaluate ... 1713
ExtractRegEx ... 1717
ExtractRegExGroup ... 1722
FindOneOf ... 1732
Hash128 ... 1736
Hash160 ... 1740
Hash256 ... 1745
Index ... 1750
IndexRegEx ... 1754

IndexRegExGroup 1757
IsJson 1762
IsRegEx 1765
JsonArray 1769
JsonGet 1771
JsonObject 1773
JsonSet 1774
JsonSetEx 1776
KeepChar 1777
Left 1780
Len 1784
LevenshteinDist 1789
Lower 1793
LTrim 1795
MatchRegEx 1800
Mid 1804
Ord 1807
PurgeChar 1810
Repeat 1813
Replace 1817
ReplaceRegEx 1820
ReplaceRegExGroup 1826
Right 1832
RTrim 1835
SubField 1839
SubFieldRegEx 1844
SubStringCount 1848
TextBetween 1854
Trim 1857
Upper 1861
8.25 Systemfunktioner 1865
Översikt av systemfunktioner 1865
CalcDim - diagramfunktion 1868
EngineVersion 1874
GetSysAttr 1874
HämtaAnvändarAttribut 1875
InObject - diagramfunktion 1875
IsPartialReload 1880
ObjectId - diagramfunktion 1880
ProductVersion 1883
StateName - diagramfunktion 1883
8.26 Tabellfunktioner 1884
Tabellfunktioner - översikt 1884
FieldName 1886
FieldNumber 1886
NoOfFields 1887
NoOfRows 1887
8.27 Trigonometriska och hyperboliska funktioner 1888

8.28 Fönsterfunktioner ... 1890
Window ... 1890
WRank ... 1900
9 Behörighetskontroll för filsystem ... 1907
9.1 Säkerhetsaspekter vid anslutning till filbaserade ODBC- och OLE DB-dataanslutningar 1907
9.2 Begränsningar i standardläge ... 1907
Systemvariabler ... 1907
Vanliga skriptsatser ... 1909
Kontrollsatser i skriptet ... 1910
Filfunktioner ... 1910
Systemfunktioner ... 1913
9.3 Inaktivera standardläge ... 1913
Qlik Sense ... 1913
Qlik Sense Desktop ... 1913
10 Skript på diagramnivå ... 1914
10.1 Kontrollsatser ... 1914
Översikt över kontrollsatser för diagrammodifierare ... 1914
Call ... 1916
Do-loop ... 1917
End ... 1918
Exit ... 1918
Exit script ... 1918
For..next ... 1919
For each..next ... 1920
If..then..elseif..else..end if ... 1923
Next ... 1924
Sub..end sub ... 1924
Switch..case..default..end switch ... 1925
To ... 1926
10.2 Prefix ... 1926
Översikt över prefix för diagrammodifierare ... 1927
Add ... 1927
Replace ... 1928
10.3 Vanliga satser ... 1928
Översikt över regelbundna satser för diagrammodifierare ... 1928
Load ... 1929
Let ... 1932
Set ... 1933
Put ... 1933
HCValue ... 1934
11 QlikView-funktioner och -satser som inte stöds i Qlik Sense ... 1935
11.1 Skriptsatser som inte stöds i Qlik Sense ... 1935
11.2 Funktioner som inte stöds i Qlik Sense ... 1935
11.3 Prefix som inte stöds i Qlik Sense ... 1935
12 Funktioner och satser som inte rekommenderas i Qlik Sense ... 1937
12.1 Skriptsatser som inte rekommenderas i Qlik Sense ... 1937
12.2 Skriptsatsparametrar som inte rekommenderas i Qlik Sense ... 1937

12.3 Funktioner som inte rekommenderas i Qlik Sense 1939
Kvalificeraren ALL 1939

1 Vad är Qlik Sense?

Qlik Sense är en plattform för dataanalys. Med Qlik Sense kan du analysera data och göra upptäckter i dessa data på egen hand. Du kan dela kunskaper och analysera data i grupper och inom organisationer. Qlik Sense låter dig ställa och besvara dina egna frågor och följa dina egna vägar till insikt. Qlik Sense gör det möjligt för dig och dina kollegor att fatta beslut tillsammans.

1.1 Vad kan du göra i Qlik Sense?

De flesta Business Intelligence (BI)-produkter kan hjälpa dig få svar på frågor som har förberetts. Men hur gör du med följdfrågor? Sådana frågor som kommer när någon läser din rapport eller ser din datavisualisering? Med Qlik Senses associativa funktioner kan du svara på fråga efter fråga efter fråga och följa din egen väg mot insikt. Med Qlik Sense kan du utforska dina data fritt, genom att klicka. Du lär dig något hela tiden och funderat ut nästa steg utifrån det du har hittat.

1.2 Hur fungerar Qlik Sense?

Qlik Sense genererar dynamiska vyer av information åt dig. Qlik Sense kräver inga fördefinierade och statiska rapporter. Du är inte heller beroende av andra användare. Det enda du behöver göra är att klicka och lära dig längs vägen. Varje gång du klickar svarar Qlik Sense omedelbart och uppdaterar varje Qlik Sense-visualisering och vy i applikationen med en nyberäknad uppsättning data och visualiseringar som är specifika för dina urval.

Appmodellen

I stället för att driftsätta och underhålla gigantiska verksamhetsapplikationer kan du skapa dina egna Qlik Sense applikationer som du kan återanvända, förändra och dela med andra. Med applikationsmodellen kan du gå vidare och ställa nästa fråga på egen hand, utan att behöva kontakta en expert för att få nya rapporter eller visualiseringar.

Den associativa analysen

Qlik Sense hanterar alla relationer i datamängden automatiskt och presenterar informationen färgkodad i **green/white/gray**. Urval markeras i grönt, associerade data visas i vitt och uteslutna (icke associerade) data visas i grått. När du får denna omedelbara återkoppling kan du enkelt komma på nästa fråga och fortsätta utforska och upptäcka.

Samarbete och mobilitet

Med Qlik Sense kan du dessutom samarbeta med kollegor oberoende av var de befinner sig. Alla Qlik Sense-funktioner, inklusive den associativa upplevelsen och samarbete, finns tillgängliga på mobila enheter. Med Qlik Sense kan du ställa frågor och få svar på dina frågor och följdfrågor, tillsammans med dina kollegor, var du än befinner dig.

1.3 Hur distribuerasQlik Sense?

Det finns två versioner av Qlik Sense att distribuera, Qlik Sense Desktop och Qlik Sense Enterprise.

Qlik Sense Desktop

Det här är en lättinstallerad version för en användare som oftast installeras på en lokal dator.

Qlik Sense Enterprise

Den här versionen används för att distribuera Qlik Sense-platser. En plats är en eller flera servrar som är kopplade till ett gemensamt logiskt lager eller en central nod.

1.4 Så här administrerar och hanterar du en Qlik Sense-plats

Med Qlik Hanteringskonsol kan du konfigurera, hantera och övervaka Qlik Sense-platser på ett enkelt och intuitivt sätt. Du kan hantera licenser, åtkomst och säkerhetsregler, konfigurera noder och kopplingar för datakällor samt synkronisera innehåll och användare bland många andra uppgifter och resurser.

1.5 Utöka Qlik Sense och anpassa det efter dina syften

Qlik Sense ger dig flexibla API:er och SDK:er så att du kan utveckla egna komplement och anpassa och integrera Qlik Sense för olika syften, exempelvis:

Bygga komplement och kombinationsprogram

Här kan webbutveckla med JavaScript för att bygga tillägg som är anpassade visualiseringar i Qlik Sense-appar, eller använda API:er för kombinationsprogram för att bygga webbplatser med Qlik Sense-innehåll.

Bygga klienter

Du kan bygga klienter i .NET och bädda in Qlik Sense-objekt i dina egna applikationer. Du kan även bygga programegna klienter med valfritt programmeringsspråk, som kan hantera WebSocket-kommunikation med hjälp av Qlik Sense-klientprotokollet.

Bygga serververktyg

Med tjänst- och användarkatalog-API:er kan du bygga ett eget verktyg för att administrera och hantera Qlik Sense-platser.

Koppla till andra datakällor

Skapa Qlik Sense-kopplingar för att hämta data från egna datakällor.

2 Översikt över skriptsyntax

2.1 Introduktion till skriptsyntax

I ett skript definieras namnet på den datakälla, de tabeller och de fält som ska användas i logiken. Här anges även vilka fält som ska ingå i behörighetsdefinitionen. Ett skript består av ett antal satser som exekveras i följd.

Kommandoradssyntaxen och skriptsyntaxen för Qlik Sense beskrivs under Backus-Naur-formalism, eller BNF-kod.

De första kodraderna skapas automatiskt redan när en ny Qlik Sense-fil skapas. Standardvärdena för dessa variabler för taltolkning hämtas från operativsystemets nationella inställningar.

Skriptet består av ett antal skriptsatser och nyckelord som exekveras i följd. Alla skriptsatser måste avslutas med ett semikolon, ”**;**”.

Du kan använda uttryck och funktioner i **LOAD**-satser för att omvandla data som har laddats.

För tabellfiler som har komman, tabbtecken eller semikolon som avgränsare kan **LOAD**-satsen användas. Standardinställningen för **LOAD**-satsen är att den laddar alla fält från filen.

Det går att komma åt allmänna databaser via ODBC- eller OLE DB-databaskopplingar. Här används SQL-standardsatser. SQL-syntaxen som kan användas skiljer sig åt mellan olika ODBC-drivrutiner.

Du kan dessutom komma åt andra datakällor med hjälp av anpassade kopplingar.

2.2 Vad är Backus-Naur-formalism?

Kommandoradssyntaxen och skriptsyntaxen för Qlik Sense beskrivs under Backus-Naur-formalism, eller BNF-kod.

I nedanstående tabell finns en lista med symboler som används i BNF-kod, men en beskrivning av hur de tolkas.

Symboler

Symbol	Beskrivning
|	Logiskt OR: symbolen på valfri sida kan användas.
( )	Parentes som definierar prioritet: används för att strukturera BNF-syntaxen.
[ ]	Hakparentes: Symbolerna innanför parentesen är valbara.
{ }	Klammer: Symbolerna innanför klammern får upprepas noll eller fler gånger.
Symbol	En icke-terminal syntaktisk kategori som kan delas upp ytterligare i andra symboler. Exempelvis sammanslagningar av ovanstående, andra icke-terminala symboler, textsträngar och så vidare.

Symbol	Beskrivning
::=	Markerar början av ett block som beskriver en symbol.
LADDA	Exempel på en terminal symbol som består av en textsträng. Ska skrivas som den är i skriptet.

Alla terminala symboler skrivs med **bold face**. Exempelvis ska ”(” tolkas som en parentes som definierar prioritet, medan "**(**" ska tolkas som ett tecken som ska skrivas i skriptet.

Exempel:

Alias-satsen kan beskrivas på följande sätt:

```
alias fieldname as aliasname { , fieldname as aliasname}
```

Detta tolkas som textsträngen "alias", följd av ett godtyckligt fältnamn, följt av textsträngen "as", följd av ett annat godtyckligt aliasnamn. Valfritt antal kombinationer av fieldname as alias" kan anges, avgränsade av kommatecken.

Följande satser är korrekta:

```
alias a as first;

alias a as first, b as second;

alias a as first, b as second, c as third;
```

Följande satser är inte korrekta:

```
alias a as first b as second;

alias a as first { , b as second };
```

3 Skriptsatser och nyckelord

Qlik Sense-skriptet består av ett antal satser. En sats kan vara antingen en vanlig skriptsats eller en kontrollsats. Vissa satser kan föregås av prefix.

Vanliga satser används normalt för att modifiera data på ett eller annat sätt. Dessa satser kan skrivas på ett valfritt antal rader i skriptet och måste alltid avslutas med ett semikolon: ";".

Kontrollsatser används normalt för att kontrollera skriptexekveringens flöde. Varje tillägg i en kontrollsats måste hållas inom en och samma rad i skriptet och avslutas med antingen semikolon eller radslut.

Prefix kan sättas framför vissa vanliga satser, men aldrig framför kontrollsatser. Prefixen **when** och **unless** kan emellertid användas som suffix i ett fåtal tillägg i kontrollsatser.

I nästa avsnitt finns en alfabetisk uppställning över alla satser, kontrollsatser och prefix som kan användas i skriptet.

Alla nyckelord i skriptet kan skrivas med antingen versaler eller gemener. Fält- och variabelnamn är dock skiftlägeskänsliga.

3.1 Kontrollsatser i skriptet

Qlik Sense-skriptet består av ett antal satser. En sats kan vara antingen en vanlig skriptsats eller en kontrollsats.

Kontrollsatser används normalt för att kontrollera skriptexekveringens flöde. Varje tillägg i en kontrollsats måste hållas inom en och samma rad i skriptet och avslutas med antingen semikolon eller radslut.

Prefix sätts aldrig framför kontrollsatser, med undantag för prefixen **when** och **unless** som kan användas med några få specifika kontrollsatser.

Alla nyckelord i skriptet kan skrivas med antingen versaler eller gemener.

Översikt av kontrollsatser i skriptet

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Call

Kontrollsatsen **call** anropar en subrutin som måste vara definierad av en **sub**-sats på en tidigare plats i skriptet.

```
Call name ( [ paramlist ])
```

Do..loop

Kontrollsatsen **do..loop** är en konstruktion för skriptiteration som exekverar en eller flera satser tills ett logiskt villkor uppfylls.

```
Do-loop [ ( while | until ) condition ] [statements]
[exit do [ ( when | unless ) condition ] [statements]
loop [ ( while | until ) condition ]
```

Exit script

Denna kontrollsats avbryter skriptexekveringen. Satsen får förekomma var som helst i skriptet.

```
Exit script[ (when | unless) condition ]
```

For each ..next

Kontrollsatsen **for each..next** är en konstruktion för skriptiteration som exekverar en eller flera satser för varje värde i en kommaavgränsad lista. Satserna inom slingan som innesluts av **for** och **next** exekveras för varje värde i listan.

```
For each..next var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

For..next

Kontrollsatsen **for..next** är en konstruktion för skriptiteration med en räknevariabel. Satserna inom slingan som innesluts av **for** och **next** exekveras för varje värde i räknevariabeln mellan angivna övre och undre gränser.

```
For..next counter = expr1 to expr2 [ stepexpr3 ]

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

If..then

Kontrollsatsen **if..then** är en skriptvalskonstruktion som gör att skriptexekveringen slår in på olika vägar beroende på ett eller flera logiska villkor.

> *Eftersom **if..then**-satsen är en kontrollsats, och som sådan slutar med ett semikolon eller radslut, måste var och en av dess fyra möjliga satser (**if..then**, **elseif..then**, **else** och **end if**) rymmas på en egen rad i skriptet.*

```
If..then..elseif..else..end if condition then
```

```
  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

Sub

Kontrollsatsen **sub..end sub** definierar en subrutin som kan anropas från en **call**-sats.

```
Sub..end sub name [ ( paramlist )] statements end sub
```

Switch

Satsen **switch** är en konstruktion för att göra val i skriptet som tvingar skriptexekveringen att följa olika vägar beroende på värdet hos ett uttryck.

```
Switch..case..default..end switch expression {case valuelist [ statements ]}
[default statements] end switch
```

Call

Kontrollsatsen **call** anropar en subrutin som måste vara definierad av en **sub**-sats på en tidigare plats i skriptet.

Syntax:

```
Call name ( [ paramlist ])
```

Argument:

Argument

Argument	Beskrivning
name	Subrutinens namn.
paramlist	En kommaavgränsad lista över de faktiska parametrar som ska skickas till den underordnade rutinen. Varje parameter i listan kan vara ett fältnamn, ett variabelnamn eller ett godtyckligt uttryck.

Den underordnade rutinen som anropas med hjälp av en **call**-sats måste vara definierad i en **sub**-sats som påträffats tidigare under skriptexekveringen.

Parametrarna kopieras in i subrutinen. Om parametern i **call**-satsen är ett variabelnamn, inte ett uttryck, kopieras dess värde dessutom tillbaka ut igen när subrutinen avslutas.

Begränsningar:

- Eftersom **call**-satsen är en kontrollsats och som sådan slutar med ett semikolon eller ett radslut, får den inte korsa en linjegräns.
- När du definierar en underordnad rutin med `Sub..end sub` inuti en kontrollsats, till exempel `if..then`, kan du bara anropa den underordnade rutinen inifrån samma kontrollsats.

Exempel:

I det här exemplet visas alla Qlik-relaterade filer i en mapp och dess undermappar, och filinformationen lagras i en tabell. Vi utgår från att du har skapat en dataanslutning med namnet Applications till mappen.

Den underordnade rutinen DoDir anropas med referensen till mappen 'lib://Applications' som parameter. Inuti den underordnade rutinen finns det ett rekursivt anrop, `Call DoDir (Dir)`, som får funktionen att leta efter filer rekursivt i undermappar.

```
sub DoDir (Root)
   For Each Ext in 'qvw', 'qvo', 'qvs', 'qvt', 'qvd', 'qvc', 'qvf'
      For Each File in filelist (Root&'\*.' &Ext)
          LOAD
             '$(File)' as Name,
             FileSize( '$(File)' ) as Size,
             FileTime( '$(File)' ) as FileTime
          autogenerate 1;
      Next File
   Next Ext
   For Each Dir in dirlist (Root&'\*' )
      Call DoDir (Dir)
   Next Dir
End Sub

Call DoDir ('lib://Applications')
```

Do-loop

Kontrollsatsen **do..loop** är en konstruktion för skriptiteration som exekverar en eller flera satser tills ett logiskt villkor uppfylls.

Syntax:

```
Do [ ( while | until ) condition ] [statements]
[exit do [ ( when | unless ) condition ] [statements]
loop[ ( while | until ) condition ]
```

*Eftersom **do..loop**-satsen är en kontrollsats, och som sådan slutar med ett semikolon eller radslut, måste var och en av dess tre möjliga komponenter (**do**, **exit do** och **loop**) rymmas på en egen rad i skriptet.*

Argument:

Argument

Argument	Beskrivning
condition	Ett logiskt uttryck som utvärderas till True eller False.
statements	Valfri grupp av en eller flera Qlik Sense-skriptsatser.
while / until	Villkorssatsen **while** eller **until** får bara förekomma en gång i en **do..loop**-sats, dvs. antingen efter **do** eller efter **loop**. Varje villkor tolkas bara första gången det påträffas, men utvärderas för varje gång det förekommer i slingan.
exit do	Om en **exit do**-sats påträffas i slingan flyttas skriptexekveringen till den första satsen efter **loop**-satsen som markerar slutet på slingan. En **exit do**-sats kan göras villkorad genom det valbara användandet av ett **when**- eller **unless**-suffix.

Exempel:

```
// LOAD files file1.csv..file9.csv

Set a=1;

Do while a<10

LOAD * from file$(a).csv;

Let a=a+1;

Loop
```

End

Skriptnyckelordet **End** används för att stänga **If**-, **Sub**- och **Switch**-satser.

Exit

Skriptnyckelordet **Exit** är en del av **Exit Script**-satsen, men kan även användas för att lämna**Do**-, **For**- eller **Sub**-tillägg.

Exit script

Denna kontrollsats avbryter skriptexekveringen. Satsen får förekomma var som helst i skriptet.

Syntax:

```
Exit Script [ (when | unless) condition ]
```

Eftersom **exit script**-satsen är en kontrollsats och som sådan slutar med ett semikolon eller ett radslut, får den inte korsa en linjegräns.

Argument:

Argument

Argument	Beskrivning
condition	Ett logiskt uttryck som utvärderas till True eller False.
when / unless	En **exit script**-sats kan göras villkorad genom det valbara användandet av en **when**- eller **unless**-sats.

Exempel:

```
//Exit script

Exit Script;



//Exit script when a condition is fulfilled

Exit Script when a=1
```

For..next

Kontrollsatsen **for..next** är en konstruktion för skriptiteration med en räknevariabel. Satserna inom slingan som innesluts av **for** och **next** exekveras för varje värde i räknevariabeln mellan angivna övre och undre gränser.

Syntax:

```
For counter = expr1 to expr2 [ step expr3 ]

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

Uttrycken *expr1*, *expr2* och *expr3* utvärderas bara första gången slingan påbörjas. Värdet på räknevariabeln kan ändras av satser inom slingan, men detta brukar inte göras inom programmering.

Om en **exit for**-sats påträffas i slingan flyttas skriptexekveringen till den första satsen efter **next**-satsen som markerar slutet på slingan. En **exit for**-sats kan göras villkorad genom det valbara användandet av ett **when**- eller **unless**-suffix.

*Eftersom **for..next**-satsen är en kontrollsats, och som sådan slutar med ett semikolon eller radslut, måste var och en av dess tre möjliga komponenter (**for..to..step**, **exit for** och **next**) rymmas på en egen rad i skriptet.*

Argument:

Argument

Argument	Beskrivning
counter	Ett variabelnamn. Om *counter* anges efter **next**, måste variabelnamnet överensstämma med det variabelnamn som angivits efter motsvarande **for**.
expr1	Ett uttryck som anger det första värdet i *counter* för vilket slingan ska exekveras.
expr2	Ett uttryck som anger det sista värdet i *counter* för vilket slingan ska exekveras.
expr3	Ett uttryck som anger hur många steg värdet i *counter* ska öka varje gång slingan exekverats.
condition	Ett logiskt uttryck som utvärderas till True eller False.
statements	Valfri grupp av en eller flera Qlik Sense-skriptsatser.

Example 1: Läsa in ett antal filer i följd

```
// LOAD files file1.csv..file9.csv

for a=1 to 9
      LOAD * from file$(a).csv;
next
```

Example 2: Läsa in valfritt antal filer i

I detta exempel utgår vi från datafilerna *x1.csv, x3.csv, x5.csv, x7.csv* och *x9.csv*. Inläsningen stoppas vid valfri punkt med hjälp av `if rand( )<0.5 then` villkoret.

```
for counter=1 to 9 step 2
      set filename=x$(counter).csv;

      if rand( )<0.5 then
            exit for unless counter=1
      end if

      LOAD a,b from $(filename);
next
```

For each..next

Kontrollsatsen **for each..next** är en konstruktion för skriptiteration som exekverar en eller flera satser för varje värde i en kommaavgränsad lista. Satserna inom slingan som innesluts av **for** och **next** exekveras för varje värde i listan.

En speciell syntax gör det möjligt att generera listor med namn på filer och mappar i den aktuella mappen.

```
for each var in list
[statements]
[exit for [ ( when | unless ) condition ]
[statements]
next [var]
```

Argument

Argument	Beskrivning
var	Namnet på en skriptvariabel som får ett nytt värde från lista för varje exekvering av slingan. Om **var** anges efter **next**, måste variabelnamnet överensstämma med det variabelnamn som angivits efter motsvarande **for each**.

Värdet på variabeln **var** kan ändras av satser inom slingan, men detta brukar inte göras inom programmering.

Om en **exit for**-sats påträffas i slingan flyttas skriptexekveringen till den första satsen efter **next**-satsen som markerar slutet på slingan. En **exit for**-sats kan göras villkorad genom det valbara användandet av ett **when**- eller **unless**-suffix.

> *Eftersom **for each..next**-satsen är en kontrollsats och som sådan slutar med ett semikolon eller radslut, måste var och en av dess tre möjliga satser (**for each**, **exit for** och **next**) rymmas på en egen rad i skriptet.*

```
list := item { , item }
item := constant | (expression) | filelist mask | dirlist mask |
fieldvaluelist mask
```

Argument

Argument	Beskrivning
constant	Valfritt tal eller valfri sträng Observera att en sträng som skrivs direkt i skriptet måste omslutas av enkla citattecken. En sträng utan enkla citattecken tolkas som en variabel och då används värdet för variabeln. Tal behöver inte omslutas av enkla citattecken.
expression	Ett godtyckligt uttryck.
mask	En fil- eller katalognamnsmask som får innehålla valfria giltiga filnamnstecken samt standardjokertecknen, ***** och **?**. Du kan använda absoluta sökvägar eller sökvägar till lib://.

Argument	Beskrivning
condition	Ett logiskt uttryck som utvärderas till True eller False.
statements	Valfri grupp av en eller flera Qlik Sense-skriptsatser.
filelist mask	Denna syntax skapar en kommaavgränsad lista över alla filer i den aktuella mappen som matchar filnamnsmasken. *Argumentet har enbart stöd för bibliotekskopplingar i standardläget.*
dirlist mask	Denna syntax skapar en kommaavgränsad lista över alla kataloger i den aktuella katalog som matchar mappnamnsmasken. *Argumentet har enbart stöd för bibliotekskopplingar i standardläget.*
fieldvaluelist mask	Syntaxen itererar genom värdena i ett fält som redan har laddats in i Qlik Sense.

*Qlik Kopplingar till webblagringsleverantörer och andra DataFiles-kopplingar stöder inte filtermasker som använder jokertecken (***** och **?**).*

Example 1: Ladda en fillista

```
// LOAD the files 1.csv, 3.csv, 7.csv and xyz.csv
for each a in 1,3,7,'xyz'
   LOAD * from file$(a).csv;
next
```

Example 2: Skapa en fillista på disk

Detta exempel laddar en lista över alla Qlik Sense-relaterade filer i en mapp.

```
sub DoDir (Root)
   for each Ext in 'qvw', 'qva', 'qvo', 'qvs', 'qvc', 'qvf', 'qvd'

      for each File in filelist (Root&'/*.' &Ext)

         LOAD
            '$(File)' as Name,
            FileSize( '$(File)' ) as Size,
            FileTime( '$(File)' ) as FileTime
         autogenerate 1;

      next File

   next Ext
   for each Dir in dirlist (Root&'/*' )
```

```
      call DoDir (Dir)

   next Dir

end sub

call DoDir ('lib://DataFiles')
```

Example 3: Itererar genom värdena i ett fält

Det här exemplet itererar genom listan med laddade värden för FIELD och genererar ett nytt fält, NEWFIELD. För varje värde i FIELD skapas två NEWFIELD-poster.

```
load * inline [
FIELD
one
two
three
];

FOR Each a in FieldValueList('FIELD')
LOAD '$(a)' &'-'&RecNo() as NEWFIELD AutoGenerate 2;
NEXT a
```

Den resulterande tabellen ser ut så här:

Example table

NEWFIELD
one-1
one-2
two-1
two-2
three-1
three-2

If..then..elseif..else..end if

Kontrollsatsen **if..then** är en skriptvalskonstruktion som gör att skriptexekveringen slår in på olika vägar beroende på ett eller flera logiska villkor.

Kontrollsatser används normalt för att kontrollera skriptexekveringens flöde. I ett diagramuttryck, använd den **if** villkorsfunktionen istället.

Syntax:

```
If  condition then

  [ statements ]

{ elseif condition then
```

```
  [ statements ] }

[ else

  [ statements ] ]

end if
```

Eftersom **if..then**-satsen är en kontrollsats, och som sådan slutar med ett semikolon eller radslut, måste var och en av dess fyra möjliga satser (**if..then**, **elseif..then**, **else** och **end if**) rymmas på en egen rad i skriptet.

Argument:

Argument

Argument	Beskrivning
condition	Ett logiskt uttryck som kan utvärderas som True eller False.
statements	Valfri grupp av en eller flera Qlik Sense-skriptsatser.

Example 1:

```
if a=1 then
      LOAD * from abc.csv;

      SQL SELECT e, f, g from tab1;
end if
```

Example 2:

```
if a=1 then; drop table xyz; end if;
```

Example 3:

```
if x>0 then
      LOAD * from pos.csv;
elseif x<0 then
      LOAD * from neg.csv;
else
      LOAD * from zero.txt;
end if
```

Next

Skriptnyckelordet **Next** används för att stänga **For**-slingor.

Sub..end sub

Kontrollsatsen **sub..end sub** definierar en subrutin som kan anropas från en **call**-sats.

Syntax:

```
Sub name [ ( paramlist )] statements end sub
```

Argument kopieras in i subrutinen. Om motsvarande faktiska parameter i **call**-satsen är ett variabelnamn, kopieras de dessutom tillbaka ut igen när subrutinen avslutas.

Om en underordnad rutin har fler formella parametrar än antalet faktiska parametrar som skickas från en **call**-sats, initialiseras de extra parametrarna till NULL och kan användas som lokala variabler inom den underordnade rutinen.

Argument:

Argument

Argument	Beskrivning
name	Subrutinens namn.
paramlist	En kommaavgränsad lista över variabelnamn för den underordnade rutinens formella parametrar. Dessa kan användas som valfri variabel inuti subrutinen.
statements	Valfri grupp av en eller flera Qlik Sense-skriptsatser.

Begränsningar:

- Eftersom **sub**-satsen är en kontrollsats, och som sådan slutar med ett semikolon eller radslut, måste var och en av dess två möjliga satser (**sub** och **end sub**) rymmas på en egen rad i skriptet.
- När du definierar en underordnad rutin med `Sub..end sub` inuti en kontrollsats, till exempel `if..then`, kan du bara anropa den underordnade rutinen inifrån samma kontrollsats.

Example 1:

```
Sub INCR (I,J)

I = I + 1

Exit Sub when I < 10

J = J + 1

End Sub

Call INCR (X,Y)
```

Example 2: - parameteröverföring

```
Sub ParTrans (A,B,C)

A=A+1

B=B+1

C=C+1

End Sub

A=1

X=1

C=1

Call ParTrans (A, (X+1)*2)
```

Resultatet av exemplet ovan blir följande (lokalt, inuti subrutinen): A initialiseras till 1, B initialiseras till 4 och C initialiseras till NULL.

När man lämnar subrutinen får den globala variabeln A värdet 2 (kopieras tillbaka från subrutinen). Den andra faktiska parametern “(X+1)*2” kopieras inte tillbaka eftersom den inte är någon variabel. Slutligen påverkas den globala variabeln C inte av subrutinens anrop.

Switch..case..default..end switch

Satsen **switch** är en konstruktion för att göra val i skriptet som tvingar skriptexekveringen att följa olika vägar beroende på värdet hos ett uttryck.

Syntax:

```
Switch expression {case valuelist [ statements ]} [default statements] end switch
```

> *Eftersom **switch**-satsen är en kontrollsats, och som sådan slutar med ett semikolon eller radslut, måste var och en av dess fyra möjliga satser (**switch**, **case**, **default** och **end switch**) rymmas på en egen rad i skriptet.*

Argument:

Argument

Argument	Beskrivning
expression	Ett godtyckligt uttryck.

Argument	Beskrivning
valuelist	En kommaavgränsad lista av värden med vilka uttryckets värde jämförs. Skriptexekveringen fortsätter med satserna efter den första gruppen där värdet i valuelist motsvarar uttryckets värde. Varje värde i valuelist kan vara ett godtyckligt uttryck. Om det inte finns någon motsvarighet efter något **case**, körs satserna efter **default**, om sådant finns.
statements	Valfri grupp av en eller flera Qlik Sense-skriptsatser.

Exempel:

```
Switch I

Case 1

LOAD '$(I): CASE 1' as case autogenerate 1;

Case 2

LOAD '$(I): CASE 2' as case autogenerate 1;

Default

LOAD '$(I): DEFAULT' as case autogenerate 1;

End Switch
```

To

Skriptnyckelordet **To** används i flera skriptsatser.

3.2 Skriptprefix

Prefix kan sättas framför vissa vanliga satser, men aldrig framför kontrollsatser. Prefixen **when** och **unless** kan emellertid användas som suffix i ett fåtal tillägg i kontrollsatser.

Alla nyckelord i skriptet kan skrivas med antingen versaler eller gemener. Fält- och variabelnamn är dock skiftlägeskänsliga.

Skriptprefix – en översikt

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Add

Add-prefixet kan adderas till valfri **LOAD**- eller **SELECT**-sats i skriptet för att specificera att det ska lägga till poster i en annan tabell. Det specificerar även att satsen ska köras i en delvis laddning. **Add**-prefixet kan även användas i en **Map**-sats.

```
Add [only] [Concatenate[(tablename )]] (loadstatement | selectstatement)
```

```
Add [ Only ] mapstatement
```

Buffer

QVD-filer kan skapas och underhållas automatiskt via prefixet **buffer**. Detta prefix kan användas på de flesta **LOAD**- och **SELECT**-satser i skript. Det anger att en QVD-fil används för att cacha/buffra satsens resultat.

```
Buffer[(option [ , option])] ( loadstatement | selectstatement )
```

```
option::= incremental | stale [after] amount [(days | hours)]
```

Concatenate

Om man vill konkatenera två tabeller som inte har samma fältuppsättning, kan man utföra en tvingad konkatenering med hjälp av **Concatenate**-prefixet.

```
Concatenate[ (tablename ) ] ( loadstatement | selectstatement )
```

Crosstable

crosstable-laddningsprefixet används för att transponera strukturerad data med ”korstabell” eller ”pivottabell”. Data som är strukturerad på detta sätt påträffas ofta när vid arbete med kalkylarkskällor. Resultatet och syftet med **crosstable**-laddningsprefixet är att omvandla sådana strukturer till en vanlig kolumnorienterad tabellekvivalent, eftersom denna struktur i allmänhet är bättre lämpad för analys i Qlik Sense.

```
Crosstable (attribute field name, data field name [ , n ] ) ( loadstatement |
selectstatement )
```

First

Prefixet **First** till en **LOAD**- eller **SELECT (SQL)**-sats används för att ladda ett angivet antal poster från en datakälla i tabellformat.

```
First n( loadstatement | selectstatement )
```

Generic

Generic-laddningsprefixet tillåter konvertering av data modellerade på element, attribut och värde (EAV) till en traditionell, normaliserad relationstabellstruktur. EAV-modellering kallas även för ”generisk datamodellering” eller ”öppet schema”.

```
Generic ( loadstatement | selectstatement )
```

Hierarchy

Prefixet **hierarchy** används för att omvandla en hierarkisk tabell till en tabell som fungerar i en Qlik Sense-datamodell. Det kan sättas framför en **LOAD**- eller **SELECT**-sats och det använder resultatet från den laddade satsen som indata för en tabellomvandling.

```
Hierarchy (NodeID, ParentID, NodeName, [ParentName], [PathSource],
[PathName], [PathDelimiter], [Depth])(loadstatement | selectstatement)
```

HierarchBelongsTo

Detta prefix används för att omvandla en överordnad-underordnad hierarkisk tabell till en tabell som fungerar i en Qlik Sense-datamodell. Det kan sättas framför en **LOAD**- eller **SELECT**-sats och det använder resultatet från den laddade satsen som indata för en tabellomvandling.

```
HierarchyBelongsTo (NodeID, ParentID, NodeName, AncestorID, AncestorName,
[DepthDiff])(loadstatement | selectstatement)
```

Inner

Prefixen **join** och **keep** kan inledas med prefixet **inner**.

Om det används före **join** anger det att inner join ska användas. Den resulterande tabellen kommer således endast att innehålla kombinationer av fältvärden från rådatatabeller där de länkande fältvärdena visas i en eller båda tabellerna. Om det används framför **keep**, anger det att båda rådatatabellerna ska reduceras till det gemensamma snittet av deras datamängder innan de lagras i Qlik Sense.
.

```
Inner ( Join | Keep) [ (tablename) ](loadstatement |selectstatement )
```

IntervalMatch

Det utökade **IntervalMatch**-prefixet används för att skapa en tabell där diskreta numeriska värden matchas till ett eller flera numeriska intervall. Det kan även matchas med värdena för en eller flera nycklar.

```
IntervalMatch (matchfield)(loadstatement | selectstatement )
```

```
IntervalMatch (matchfield,keyfield1 [ , keyfield2, ... keyfield5 ] )
(loadstatement | selectstatement )
```

Join

join-prefixet länkar den inlästa tabellen till en existerande namngiven tabell eller den senast skapade datatabellen.

```
[Inner | Outer | Left | Right ] Join [ (tablename ) ]( loadstatement |
selectstatement )
```

Keep

Prefixet **keep** liknar prefixet **join**. Precis som prefixet **join** jämför det här prefixet den inlästa tabellen med en befintlig namngiven tabell eller den senaste tidigare skapade datatabellen, men i stället för att koppla ihop den inlästa tabellen med en befintlig tabell gör den så att den ena eller båda tabellerna minskas innan de sparas i Qlik Sense, baserat på intersektionen av tabelldata. Jämförelsen som görs motsvarar en naturlig länkning (join) som görs över alla gemensamma fält. Den görs med andra ord likadant som en motsvarande join. Skillnaden är dock att de två tabellerna inte länkas utan lagras i Qlik Sense som två tabeller med olika namn.

```
(Inner | Left | Right) Keep [(tablename ) ]( loadstatement | selectstatement
)
```

Left

Prefixen **Join** och **Keep** kan inledas med prefixet **left**.

Om det används före **join** anger det att left join ska användas. Den resulterande tabellen kommer således att innehålla endast kombinationer av fältvärden från rådatatabeller där de länkande fältvärdena visas i den första tabellen. Om det används framför **keep**, anger det att den andra rådatatabellen ska reduceras till det gemensamma snittet med den första tabellen innan den lagras i Qlik Sense.

```
Left ( Join | Keep) [ (tablename) ](loadstatement |selectstatement )
```

Mapping

mapping-prefixet används för att skapa en mappningstabell som exempelvis kan användas för att ersätta fältvärden och fältnamn under skriptexekveringen.

```
Mappning  ( loadstatement | selectstatement )
```

Merge

Merge-prefixet kan adderas till valfri **LOAD**- eller **SELECT**-sats i skriptet för att specificera att den laddade tabellen ska slås samman med en annan tabell. Det specificerar även att satsen ska köras i en delvis laddning.

```
Slå samman [only] [(SequenceNoField [, SequenceNoVar])] On ListOfKeys
[Concatenate [(TableName)]] (loadstatement | selectstatement)
```

NoConcatenate

Med **NoConcatenate**-prefixet tvingas skriptet att behandla två inlästa tabeller med identiska fält som två separata interna tabeller. Annars skulle tabellerna konkateneras automatiskt.

```
NoConcatenate( loadstatement | selectstatement )
```

Outer

Det explicita prefixet **Join** går att fördefiniera med prefixet **Outer** för att ange en outer join. I en outer join genereras alla kombinationer mellan de två tabellerna. Den resulterande tabellen kommer således att innehålla kombinationer av fältvärden från rådatatabeller där de länkande fältvärdena visas i en eller båda tabellerna. Nyckelordet **Outer** är valfritt och är den join-typ som är standard när ett join-prefix inte anges.

```
Outer Join [ (tablename) ](loadstatement |selectstatement )
```

Partial reload

En fullständig laddning inleds alltid med att alla tabeller i den befintliga datamodellen tas bort, och sedan körs laddningsskriptet.

En *Delvis laddning (page 113)* kommer inte att göra detta. Då behålls istället alla tabeller i datamodellen och sedan exekveras bara **Load**- och **Select**-satser med ett inledande **Add**-, **Merge**- eller **Replace**-prefix. Andra datatabeller påverkas inte av kommandot. **Only**-argumentet anger att satsen bara ska exekveras vid delvisa laddningar och ignoreras vid fullständiga laddningar. Följande tabell sammanfattar programutförandet för partiella och fullständiga ominläsningar.

Replace

Prefixet **Replace** kan adderas till valfri **LOAD**- eller **SELECT**-sats i skriptet för att specificera att den laddade tabellen ska ersätta en annan tabell. Det specificerar även att satsen ska köras i en delvis laddning. **Replace**-prefixet kan även användas i en **Map**-sats.

```
Replace [only] [Concatenate[(tablename) ]] (loadstatement | selectstatement)
```

```
Replace [only] mapstatement
```

Right

Prefixen **Join** och **Keep** kan inledas med prefixet **right**.

Om det används före **join** anger det att right join ska användas. Den resulterande tabellen kommer endast att innehålla kombinationer av fältvärden från rådatatabeller där de länkande fältvärdena visas i den andra tabellen. Om det används framför **keep**, anger det att den första rådatatabellen ska reduceras till det gemensamma snittet med den andra tabellen innan den lagras i Qlik Sense.

```
Right (Join | Keep) [(tablename)](loadstatement |selectstatement )
```

Sample

Prefixet **sample** till en **LOAD**- eller **SELECT**-sats används för att ladda ett slumpmässigt urval av poster från datakällan.

```
Sample p ( loadstatement | selectstatement )
```

Semantic

Tabeller som innehåller relationer mellan poster kan laddas med hjälp av ett **semantic**-prefix. Det kan exempelvis handla om referenser inom en tabell där en post pekar på en annan, såsom förfader, tillhör eller föregångare.

```
Semantic ( loadstatement | selectstatement)
```

Unless

Prefixet eller suffixet **unless** används för att skapa en villkorssats, som bestämmer om en sats eller ett exit-tillägg ska utvärderas eller ej. Det kan ses som ett kompakt alternativ till en fullständig **if..end if**-sats.

```
(Unless condition statement | exitstatement Unless condition )
```

When

Prefixet eller suffixet **when** används för att skapa en villkorssats, som bestämmer om en sats eller ett exit-tillägg ska exekveras eller ej. Det kan ses som ett kompakt alternativ till en fullständig **if..end if**-sats.

```
( When condition statement | exitstatement when condition )
```

Add

Add-prefixet kan adderas till valfri **LOAD**- eller **SELECT**-sats i skriptet för att specificera att det ska lägga till poster i en annan tabell. Det specificerar även att satsen ska köras i en delvis laddning. **Add**-prefixet kan även användas i en **Map**-sats.

> *För att delvis laddning ska fungera som avsett måste applikationen öppnas med data innan en delvis laddning utlöses.*

Gör en delvis laddning med knappen **Ladda**. Du kan även använda Qlik Engine JSON API.

Syntax:

```
Add [only] [Concatenate[(tablename)]] (loadstatement | selectstatement)
```

```
Add [only] mapstatement
```

Under en vanlig laddning (ej delvis) fungerar konstruktionen **Add LOAD** som en vanlig **LOAD**-sats. Poster genereras och lagras i en tabell.

Om **Concatenate**-prefixet används, eller om det finns en tabell med samma uppsättning fält, kommer posterna att läggas till efter den relevanta befintliga tabellen. Annars skapar konstruktionen **Add LOAD** en ny tabell.

En delvis laddning gör samma sak. Den enda skillnaden är att konstruktionen **Add LOAD** aldrig skapar en ny tabell. Det finns alltid en relevant tabell från föregående skriptexekvering där posterna ska läggas till.

Ingen kontroll av dubbletter kommer att genomföras. En sats som föregås av **Add**-prefixet innehåller därför ofta en distinct-kvalificerare eller en where-sats som hindrar förekomsten av dubbletter.

Om **Add Map...Using**-satsen används utförs mappningen även vid delvis skriptexekvering.

Argument:

Argument

Argument	Beskrivning
only	En valfri kvalificerare som innebär att satsen bara ska exekveras vid delvisa laddningar. Den ska ignoreras vid vanliga (ej delvisa) laddningar.

Exempel och resultat:

Exempel	Resultat

`Tab1:`
 	
`LOAD Name, Number FROM Persons.csv;`
 	
`Add LOAD Name, Number FROM newPersons.csv;`
	Vid vanlig laddning laddas data från *Persons.csv* och lagras i Qlik Sense-tabellen Tab1. Därefter konkateneras data från *NewPersons.csv* med samma Qlik Sense-tabell.
	
	Vid delvis laddning laddas data från *NewPersons.csv* och läggs till i Qlik Sense-tabellen Tab1. Ingen kontroll av dubbletter genomförs.

`Tab1:`
 	
`SQL SELECT Name, Number FROM Persons.csv;`
 	
`Add LOAD Name, Number FROM NewPersons.csv where not exists(Name);`
	Genom att kontrollera om Name existerar i tidigare inlästa tabelldata genomförs en kontroll av dubbletter.
	
	Vid vanlig laddning laddas data från *Persons.csv* och lagras i Qlik Sense-tabellen Tab1. Därefter konkateneras data från *NewPersons.csv* med samma Qlik Sense-tabell.
	
	Vid partiell laddning läses data från *NewPersons.csv* och läggs till i Qlik Sense-tabellen Tab1. Genom att kontrollera om Name existerar i tidigare inlästa tabelldata genomförs en kontroll av dubbletter.

Exempel	Resultat
Tab1: LOAD Name, Number FROM Persons.csv; Add Only LOAD Name, Number FROM NewPersons.csv where not exists(Name);	Vid vanlig laddning laddas data från *Persons.csv* och lagras i Qlik Sense-tabellen Tab1. Satsen som läser *NewPersons.csv* beaktas ej. Vid partiell laddning läses data från *NewPersons.csv* och läggs till i Qlik Sense-tabellen Tab1. Genom att kontrollera om Name existerar i tidigare inlästa tabelldata genomförs en kontroll av dubbletter.

Buffer

QVD-filer kan skapas och underhållas automatiskt via prefixet **buffer**. Detta prefix kan användas på de flesta **LOAD**- och **SELECT**-satser i skript. Det anger att en QVD-fil används för att cacha/buffra satsens resultat.

```
Buffer [(option [ , option])] ( loadstatement | selectstatement )
option::= incremental | stale [after] amount [(days | hours)]
```

Om inget alternativ används så används den QVD-buffert som skapades när skriptet kördes första gången på obegränsad tid.

Buffertfilen sparas i delmappen *Buffers*, normalt *C:\ProgramData\Qlik\Sense\Engine\Buffers* (serverinstallation) eller *C:\Users\{user}\Documents\Qlik\Sense\Buffers* (Qlik Sense Desktop).

QVD-filens namn är ett uträknat namn (ett 160-bitars hexadecimalt hash-värde av hela den efterföljande **LOAD**- eller **SELECT**-satsen samt annan särskiljande information). Detta innebär att QVD-bufferten blir ogiltig om ändringar görs i den efterföljande **LOAD**- eller **SELECT**-satsen.

Normalt avlägsnas QVD-buffertar när de inte längre blir refererade någonstans under en fullständig skriptexekvering i den applikation som skapade dem. De avlägsnas även när den applikation som skapade dem inte längre finns.

Argument

Argument	Beskrivning
inkrementell	Med alternativet incremental öppnas möjligheten att enbart läsa in delar av den underliggande filen. Filens tidigare storlek lagras i XML-huvudet i QVD-filen. Detta är särskilt användbart för loggfiler. Alla tidigare inlästa poster läses in från QVD-filen medan nyare poster läses in från originalkällan och en uppdaterad QVD-fil skapas. Alternativet incremental kan bara användas med **LOAD**-satser och textfiler. Inkrementell laddning kan inte användas där gamla data ändras eller raderas.

Argument	Beskrivning
gammal [efter] belopp [(dagar | timmar)]	amount är ett tal som anger tidsperioden. Decimals kan användas. Om ingen enhet anges förutsätts dagar. Alternativet stale after används oftast med databaskällor vars originaldata är svåra att förse med en tidsmarkör. Istället anger man hur gammal en QVD-ögonblicksbild kan vara för att få användas. En stale after-sats anger helt enkelt en tidsperiod efter att QVD-bufferten skapades, efter vilken den inte längre kommer att anses vara giltig. Dessförinnan används QVD-bufferten som källa för data och därefter används den ursprungliga datakällan. Därefter uppdateras QVD-buffertfilen automatiskt och en ny period påbörjas.

Det finns flera begränsningar, den mest anmärkningsvärda är att det måste finnas antingen en fil-**LOAD**- eller en **SELECT**-sats i basen för alla komplexa satser.

```
Buffer SELECT * from MyTable;
Buffer (stale after 7 days) SELECT * from MyTable;
Buffer (incremental) LOAD * from MyLog.log;
```

Concatenate

`Concatenate` är ett skriptladdningsprefix som gör det möjligt att lägga till en datauppsättning i en befintlig tabell i minnet. Den används ofta för att lägga till olika uppsättningar med transaktionsdata i en enda central faktatabell, eller för att bygga upp gemensamma referensdatauppsättningar av en specifik typ som härstammar från olika källor. Den har samma funktion som en SQL UNION-operator.

Den resulterande tabellen från en `concatenate`-operation kommer att innehålla den ursprungliga datauppsättningen med de nya raderna med data tillagda längst ner i tabellen. Käll- och måltabellerna kan ha olika fält. Om fälten skiljer sig åt kommer den resulterande tabellen att vidgas så att det kombinerade resultatet av alla befintliga fält i både källtabellen och måltabellen kan representeras.

Syntax:

```
Concatenate[ (tablename ) ] ( loadstatement | selectstatement )
```

Argument

Argument	Beskrivning
`tablename`	Namnet på en befintlig tabell. Den namngivna tabellen kommer att vara målet för `Concatenate`-operationen och alla poster med data som laddas kommer att läggas till i den tabellen. Om parametern `tablename` inte används kommer måltabellen att vara den sist laddade tabellen före den här satsen.
`loadstatement/selectstatement`	`loadstatement/selectstatement`-argumentet som följer efter `tablename`-argumentet kommer att konkateneras till den angivna tabellen.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Funktionsexempel

Exempel	Resultat
`Concatenate (Transactions) Load …. ;`	De data som laddades i laddningsskriptet nedanför prefixet `Concatenate` kommer att läggas till i den befintliga tabellen i minnet som heter `Transactions` (förutsatt att en tabell som heter `Transactions` har laddats tidigare i laddningsskriptet).

Exempel 1 – lägga till flera uppsättningar med data i en måltabell med konkatenerat laddningsprefix

Laddningsskript och resultat

Översikt

I det här exemplet kommer du att ladda två skript i ordningsföljd.

- Det första laddningsskriptet innehåller en initial datauppsättning med datum och belopp som skickas till en tabell som heter `Transactions`.
- Det andra laddningsskriptet innehåller:
  - En andra datauppsättning som läggs till den initiala datauppsättningen genom att använda prefixet `Concatenate`. Den här datauppsättningen har ytterligare ett fält, `type`, som inte ingår i den initiala datauppsättningen.
  - Prefixet `Concatenate`.

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskript 1

```
Transactions:
Load * Inline [

id, date, amount
3750, 08/30/2018, 23.56
3751, 09/07/2018, 556.31
3752, 09/16/2018, 5.75
```

```
3753, 09/22/2018, 125.00
3754, 09/22/2018, 484.21
3756, 09/22/2018, 59.18
3757, 09/23/2018, 177.42
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `amount`

Resultattabell för det första laddningsskriptet

id	datum	belopp
3750	08/30/2018	23.56
3751	09/07/2018	556,31
3752	09/16/2018	5,75
3753	22/09/2018	125,00
3754	22/09/2018	484,21
3756	22/09/2018	59,18
3757	23/09/2018	177,42

Tabellen visar den initiala datauppsättningen.

Laddningsskript 2

Öppna skriptredigeraren och lägg till laddningsskriptet nedan

```
Concatenate(Transactions)
Load * Inline [
id, date, amount, type
3758, 10/01/2018, 164.27, Internal
3759, 10/03/2018, 384.00, External
3760, 10/06/2018, 25.82, Internal
3761, 10/09/2018, 312.00, Internal
3762, 10/15/2018, 4.56, Internal
3763, 10/16/2018, 90.24, Internal
3764, 10/18/2018, 19.32, External
];
```

Resultat

Ladda data och gå till arket. Skapa det här fältet som en dimension.

- type

Resultattabell för det andra laddningsskriptet

id	datum	belopp	typ av
3750	08/30/2018	23.56	-
3751	09/07/2018	556,31	-
3752	09/16/2018	5,75	-
3753	22/09/2018	125,00	-
3754	22/09/2018	484,21	-
3756	22/09/2018	59,18	-
3757	23/09/2018	177,42	-
3758	10/01/2018	164.27	Intern
3759	10/03/2018	384.00	Extern
3760	10/06/2018	25.82	Intern
3761	10/09/2018	312.00	Intern
3762	10/15/2018	4.56	Intern
3763	10/16/2018	90.24	Intern
3764	10/18/2018	19.32	Extern

Observera nullvärdena i type-fältet för de första sju laddade posterna där type inte hade definierats.

Exempel 2 – lägga till flera uppsättningar med data i en måltabell med implicitet konkatenering

Laddningsskript och resultat

Översikt

Ett typiskt användarfall för att lägga till data implicit är du när du laddar flera filer med identiskt strukturerade data och vill lägga till dem alla i en måltabell.

Till exempel genom att använda wildcards i filnamn med syntax som till exempel:

```
myTable:
Load * from [myFile_*.qvd] (qvd);
```

eller i slingor med hjälp av konstruktioner som exempelvis:

```
for each file in filelist('myFile_*.qvd')

myTable:
Load * from [$(file)] (qvd);
```

```
next file
```

> *Implicit konkatenering kommer att ske mellan alla tabeller som är laddade med fält med identiska namn, även om de inte har definierats efter varandra i skriptet. Detta kan leda till att data oavsiktligt läggs till i tabeller. Om du inte vill att en sekundär tabell med identiska fält ska läggas till på det här sättet använder du laddningsprefixet* `NoConcatenate`*. Det räcker inte att byta namn på tabellen med en alternativ namntagg för att förhindra attimplicita konkateneringar inträffar. Mer information finns i* *NoConcatenate (page 103)**.*

I det här exemplet kommer du att ladda två skript i ordningsföljd.

- Det första laddningsskriptet innehåller en initial datauppsättning med fyra fält som skickas till en tabell som heter `Transactions`.
- Det andra laddningsskriptet innehåller en datauppsättning med samma fält som den första datauppsättningen.

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskript 1

```
Transactions:
Load * Inline [
id, date, amount, type
3758, 10/01/2018, 164.27, Internal
3759, 10/03/2018, 384.00, External
3760, 10/06/2018, 25.82, Internal
3761, 10/09/2018, 312.00, Internal
3762, 10/15/2018, 4.56, Internal
3763, 10/16/2018, 90.24, Internal
3764, 10/18/2018, 19.32, External
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `amount`
- `type`

Resultattabell för det första laddningsskriptet

id	datum	typ av	belopp
3758	10/01/2018	Intern	164.27
3759	10/03/2018	Extern	384.00

id	datum	typ av	belopp
3760	10/06/2018	Intern	25.82
3761	10/09/2018	Intern	312.00
3762	10/15/2018	Intern	4.56
3763	10/16/2018	Intern	90.24
3764	10/18/2018	Extern	19.32

Tabellen visar den initiala datauppsättningen.

Laddningsskript 2

Öppna skriptredigeraren och lägg till laddningsskriptet nedan

```
Load * Inline [
id, date, amount, type
3765, 11/03/2018, 129.40, Internal
3766, 11/05/2018, 638.50, External
];
```

Resultat

Ladda data och gå till arket.

Resultattabell för det andra laddningsskriptet

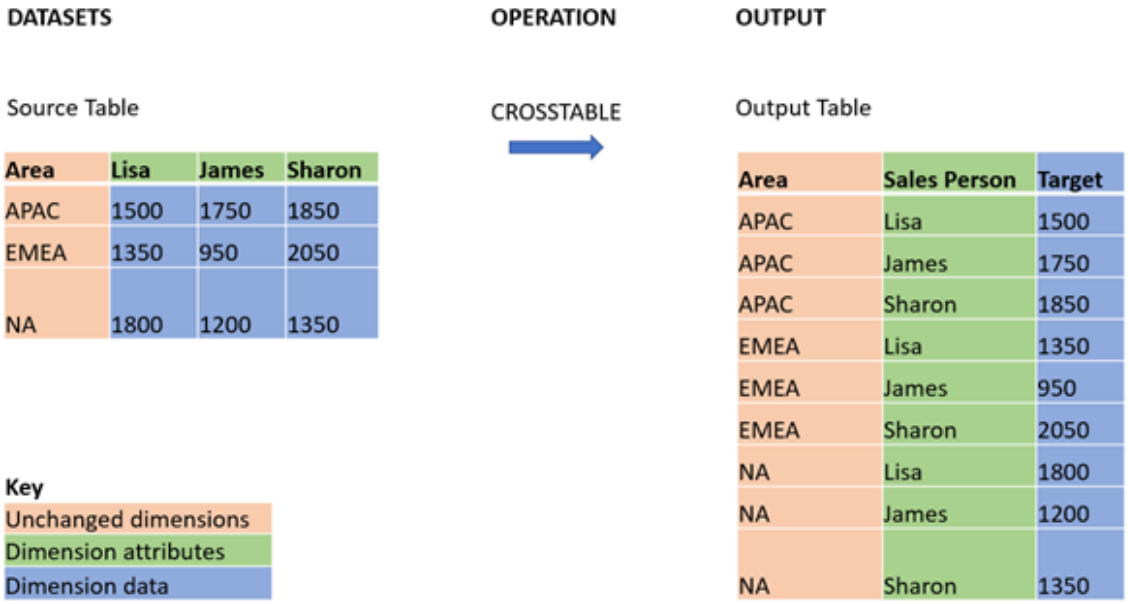
id	datum	typ av	belopp
3758	10/01/2018	Intern	164.27
3759	10/03/2018	Extern	384.00
3760	10/06/2018	Intern	25.82
3761	10/09/2018	Intern	312.00
3762	10/15/2018	Intern	4.56
3763	10/16/2018	Intern	90.24
3764	10/18/2018	Extern	19.32
3765	11/03/2018	Intern	129.40
3766	11/05/2018	Extern	638.50

Det andra laddningsskriptet konkatenerades implicit till den initiala datauppsättningen eftersom de hade identiska fält.

Crosstable

crosstable-laddningsprefixet används för att transponera strukturerad data med ”korstabell” eller ”pivottabell”. Data som är strukturerad på detta sätt påträffas ofta när vid arbete med kalkylarkskällor. Resultatet och syftet med **crosstable**-laddningsprefixet är att omvandla sådana strukturer till en vanlig kolumnorienterad tabellekvivalent, eftersom denna struktur i allmänhet är bättre lämpad för analys i Qlik Sense.

Exempel på data strukturerad som en korstabell och dess ekvivalenta struktur efter en korstabellomvandling



Syntax:

```
crosstable (attribute field name, data field name [ , n ] ) ( loadstatement | selectstatement )
```

Argument

Argument	Beskrivning
attribute field name	Det önskade utdatafältets namn som beskriver den horisontellt orienterade dimensionen som ska transponeras (rubrikraden).
data field name	Det önskade utdatafältets namn som beskriver den horisontellt orienterade dimensionen som ska transponeras (matris av datavärden under rubrikraden).
n	Antalet kvalificerarfält eller oförändrade dimensioner som föregår den tabell som ska omvandlas till generisk form. Standardvärdet är 1.

Skriptingfunktionen är relaterad till följande funktioner:

Relaterade funktioner

Funktion	Interaktion
Generic (page 58)	Ett laddningsprefix för omvandling som tar en strukturerad datamängd av entitetsattributvärde och omvandlar den till en vanlig relationstabellstruktur, som separerar varje attribut som påträffas i ett nytt fält eller kolumn med data.

Exempel 1 – Omvandla pivoterade försäljningsdata (enkla)

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Det första laddningsskriptet innehåller en datauppsättning på vilken skriptprefixet `crosstable` kommer att tillämpas senare, med avsnittet som tillämpar `crosstable` kommenterat. Detta betyder att kommentarsyntax användes för att inaktivera det här avsnittet i laddningsskriptet.

Det andra laddningsskriptet är detsamma som det första, men med tillämpning av `crosstable` okommenterat (aktiveras genom att ta bort kommentarsyntaxen). Skripten visas på detta sätt för att markera värdet av denna skriptfunktion vid omvandling av data.

Första laddningsskriptet (funktionen tillämpas inte)

```
tmpData:
//Crosstable (MonthText, Sales)
Load * inline [
Product, Jan 2021, Feb 2021, Mar 2021, Apr 2021, May 2021, Jun 2021
A, 100, 98, 103, 63, 108, 82
B, 284, 279, 297, 305, 294, 292
C, 50, 53, 50, 54, 49, 51];

//Final:
//Load Product,
//Date(Date#(MonthText,'MMM YYYY'),'MMM YYYY') as Month,
//Sales

//Resident tmpData;

//Drop Table tmpData;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Product`
- `Jan 2021`
- `Feb 2021`

- Mar 2021
- Apr 2021
- May 2021
- Jun 2021

Resultattabell

Produkt	Jan 2021	Feb 2021	Mar 2021	Apr 2021	Maj 2021	Jun 2021
A	100	98	103	63	108	82
B	284	279	297	305	294	292
C	50	53	50	54	49	51

Det här skriptet gör det möjligt att skapa en korstabell med en kolumn för varje månad och en rad per produkt. I det nuvarande formatet är det inte lätt att analysera informationen. Det skulle vara mycket bättre att ha alla siffror i ett fält och alla månader i ett annat, det vill säga i en tabell med tre kolumner. Nästa avsnitt förklarar hur du gör denna omvandling till korstabell.

Andra laddningsskriptet (funktionen tillämpas)

Avkommentera skriptet genom att ta bort `//`. Det laddade skriptet bör nu se ut så här:

```
tmpData:
Crosstable (MonthText, Sales)
Load * inline [
Product, Jan 2021, Feb 2021, Mar 2021, Apr 2021, May 2021, Jun 2021
A, 100, 98, 103, 63, 108, 82
B, 284, 279, 297, 305, 294, 292
C, 50, 53, 50, 54, 49, 51];

Final:
Load Product,
Date(Date#(MonthText,'MMM YYYY'),'MMM YYYY') as Month,
Sales

Resident tmpData;

Drop Table tmpData;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- Product
- Month
- Sales

Resultattabell

Produkt	Månad	Försäljning
A	Jan 2021	100
A	Feb 2021	98
A	Mar 2021	103
A	Apr 2021	63
A	May 2021	108
A	Jun 2021	82
B	Jan 2021	284
B	Feb 2021	279
B	Mar 2021	297
B	Apr 2021	305
B	May 2021	294
B	Jun 2021	292
C	Jan 2021	50
C	Feb 2021	53
C	Mar 2021	50
C	Apr 2021	54
C	May 2021	49
C	Jun 2021	51

När skriptprefixet har tillämpats så omvandlas korstabellen till en rak tabell med en kolumn för `Month` och en annan för `Sales`. Detta förbättrar läsbarheten av data.

Exempel 2 – Omvandling av pivoterade försäljningsmåldata till en vertikal tabellstruktur (mellanliggande)

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning läses in i en tabell som heter Mål.
- `crosstable`-laddningsprefixet, som överför de pivoterade säljarnamnen till ett eget fält, märkt `Sales`

`Person`.

- Den associerade försäljningsmåldata, som är strukturerade i ett fält som heter `Target`.

Laddningsskript

```
SalesTargets:
CROSSTABLE([Sales Person],Target,1)
LOAD
*
INLINE [
Area, Lisa, James, Sharon
APAC, 1500, 1750, 1850
EMEA, 1350, 950, 2050
NA, 1800, 1200, 1350
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Area`
- `Sales Person`

Lägg till måttet:

```
=Sum(Target)
```

Resultattabell

Yta	Säljare	=Sum(Target)
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
Ej tillämpligt	James	1200
Ej tillämpligt	Lisa	1800
Ej tillämpligt	Sharon	1350

Om du vill replikera visningen av data som en pivoterad inmatningstabell kan du skapa ett motsvarande pivottabell i ett ark.

Gör följande:

1. Kopiera och klistra in tabellen du just har skapat i arket.
2. Dra diagramobjektet för **pivottabellen** upp på den nyligen skapade tabellkopian. Välj **Konvertera**.
3. Klicka på ✔ **Redigering klar**.
4. Dra `Sales Person`-fältet från den vertikala kolumnhyllan till den horisontella kolumnhyllan.

Följande tabell visar data i sin ursprungliga tabellform, som den visas i Qlik Sense:

Originalresultattabell, som visas i Qlik Sense

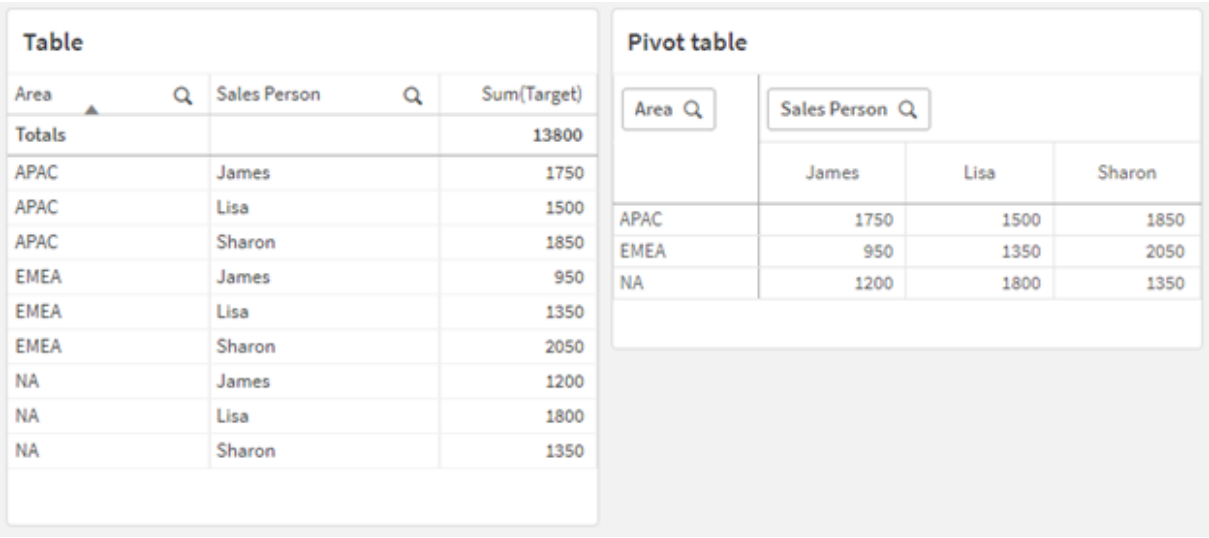
Yta	Säljare	=Sum(Target)
Totalvärden	-	13800
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
Ej tillämpligt	James	1200
Ej tillämpligt	Lisa	1800
Ej tillämpligt	Sharon	1350

Den motsvarande pivottabellen liknar följande, där kolumnen för varje säljares namn finns i den större raden för `Sales Person`:

Motsvarande pivottabell med `Sales Person`-fältet pivoterat horisontellt

Yta	James	Lisa	Sharon
APAC	1750	1500	1850
EMEA	950	1350	2050
Ej tillämpligt	1350	1350	1350

Exempel på data som visas som en tabell och en motsvarande pivottabell med `Sales Person`-fältet pivoterat horisontellt



Table

Area	Sales Person	Sum(Target)
Totals		13800
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
NA	James	1200
NA	Lisa	1800
NA	Sharon	1350

Pivot table

Area	James	Lisa	Sharon
APAC	1750	1500	1850
EMEA	950	1350	2050
NA	1200	1800	1350

Exempel 3 – Omvandling av pivoterade försäljningsmåldata till en vertikal tabellstruktur (mellanliggande)

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som representerar försäljnings- och måldata, organiserad efter område och månad på året. Detta läses in i en tabell som heter `SalesAndTargets`.
- `crosstable`-laddningsprefixet. Detta används för att avpivotera `Month Year`-dimensionen till ett dedikerat fält, samt för att överföra matrisen av försäljnings- och målbelopp till ett dedikerat fält som kallas `Amount`.
- En konvertering av `Month Year`-fältet från text till ett korrekt datum, med hjälp av text-till-datum-konverteringsfunktionen `date#`. Detta datumkonverterade `Month Year`-fält kopplas tillbaka till `SalesAndTarget`-tabellen via ett `Join`-laddningsprefix.

Laddningsskript

```
SalesAndTargets:
CROSSTABLE(MonthYearAsText,Amount,2)
LOAD
      *
INLINE [
Area   Type    Jan-22  Feb-22  Mar-22  Apr-22  May-22  Jun-22  Jul-22  Aug-22  Sep-22  Oct-22  Nov-22  Dec-22
APAC   Target  425     425     425     425     425     425     425     425     425     425     425     425
APAC   Actual  435     434     397     404     458     447     413     458     385     421     448     397
```

```
EMEA    Target  362.5   362.5   362.5   362.5   362.5   362.5   362.5   362.5   362.5   362.5   362.5   362.5
EMEA    Actual  363.5   359.5   337.5   361.5   341.5   337.5   379.5   352.5   327.5   337.5   360.5   334.5
NA      Target  375     375     375     375     375     375     375     375     375     375     375     375
NA      Actual  378     415     363     356     403     343     401     365     393     340     360     405
] (delimiter is '\t');

tmp:
LOAD DISTINCT MonthYearAsText,date#(MonthYearAsText,'MMM-YY') AS [Month Year]
RESIDENT SalesAndTargets;

JOIN (SalesAndTargets)
LOAD * RESIDENT tmp;

DROP TABLE tmp;
DROP FIELD MonthYearAsText;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Area`
- `Month Year`

Skapa följande mått med `Actual`-etiketten:

```
=Sum({<Type={'Actual'}>} Amount)
```

Skapa även måttet med `Target`-etiketten:

```
=Sum({<Type={'Target'}>} Amount)
```

Resultattabell (beskuren)

Yta	Månad år	Faktisk	Mål
APAC	Jan-22	435	425
APAC	Feb-22	434	425
APAC	Mar-22	397	425
APAC	Apr-22	404	425
APAC	Maj-22	458	425
APAC	Jun-22	447	425
APAC	Jul-22	413	425
APAC	Aug-22	458	425
APAC	Sep-22	385	425
APAC	Okt-22	421	425
APAC	Nov-22	448	425

Yta	Månad år	Faktisk	Mål
APAC	Dec-22	397	425
EMEA	Jan-22	363.5	362.5
EMEA	Feb-22	359.5	362.5

Om du vill replikera visningen av data som en pivoterad inmatningstabell kan du skapa ett motsvarande pivottabell i ett ark.

Gör följande:

1. Kopiera och klistra in tabellen du just har skapat i arket.
2. Dra diagramobjektet för **pivottabellen** upp på den nyligen skapade tabellkopian. Välj **Konvertera**.
3. Klicka på ✔ **Redigering klar**.
4. Dra `Month Year`-fältet från den vertikala kolumnhyllan till den horisontella kolumnhyllan.
5. Dra `Values`-elementet från den horisontella kolumnhyllan till den vertikala kolumnhyllan.

Följande tabell visar data i sin ursprungliga tabellform, som den visas i Qlik Sense:

Originalresultattabell, som visas i Qlik Sense

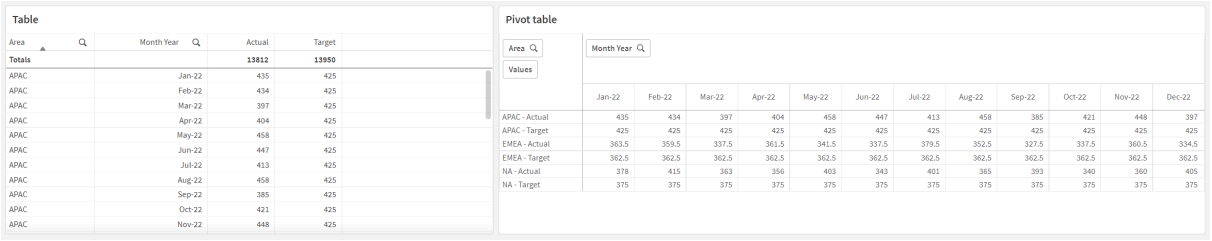
Yta	Månad år	Faktisk	Mål
Totalvärden	-	13812	13950
APAC	Jan-22	435	425
APAC	Feb-22	434	425
APAC	Mar-22	397	425
APAC	Apr-22	404	425
APAC	Maj-22	458	425
APAC	Jun-22	447	425
APAC	Jul-22	413	425
APAC	Aug-22	458	425
APAC	Sep-22	385	425
APAC	Okt-22	421	425
APAC	Nov-22	448	425
APAC	Dec-22	397	425
EMEA	Jan-22	363.5	362.5
EMEA	Feb-22	359.5	362.5

Den motsvarande pivottabellen liknar följande, där kolumnen för varje enskild månad för året finns i den större raden för `Month Year`:

Motsvarande pivottabell (beskuren) med `Month Year`-fältet pivoterat horisontellt

Område (värden)	Jan-22	Feb-22	Mar-22	Apr-22	Maj-22	Jun-22	Jul-22	Aug-22	Sep-22	Okt-22	Nov-22	Dec-22
APAC - faktisk	435	434	397	404	458	447	413	458	385	421	448	397
APAC - mål	425	425	425	425	425	425	425	425	425	425	425	425
EMEA - faktisk	363.5	359.5	337.5	361.5	341.5	337.5	379.5	352.5	327.5	337.5	360.5	334.5
EMEA - mål	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5
EMEA - faktisk	378	415	363	356	403	343	401	365	393	340	360	405
EMEA - mål	375	375	375	375	375	375	375	375	375	375	375	375

Exempel på data som visas som en tabell och en motsvarande pivottabell med `Month Year`-fältet pivoterat horisontellt



First

`First`-prefixet för en `LOAD`- eller `SELECT`-sats (SQL) används för att ladda ett angivet antal poster från en datakälla i tabellformat. Ett typiskt användningsfall för att använda `First`-prefixet är när du vill hämta en liten delmängd av poster från ett stort och/eller långsamt dataladdningssteg. Så snart det definierade `n`-antalet poster har laddats, avslutas laddningssteget i förtid och resten av skriptexekveringen fortsätter som vanligt.

Syntax:

```
First n ( loadstatement | selectstatement )
```

Argument

Argument	Beskrivning
`n`	Ett godtyckligt uttryck, vars resultat är ett heltal som anger det maximala antalet poster som ska laddas. `n`kan också inneslutas inom parentes: `(n)`.
`loadstatement | selectstatement`	`load statement/select statement` som följer efter argumentet n kommer att definiera den angivna tabellen som måste laddas med det inställda maximala antalet poster.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`FIRST 10 LOAD * from abc.csv;`	Detta exempel kommer att hämta de första tio raderna från en excel-fil.
`FIRST (1) SQL SELECT * from Orders;`	Det här exemplet kommer att hämta den första markerade raden från datauppsättningen `Orders`.

Exempel – Ladda de första fem raderna

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum från de första två veckorna 2020.
- `First`-variabeln som instruerar programmet att endast ladda de första fem posterna.

Laddningsskript

```
Sales:
FIRST 5
LOAD
```

```
*
Inline [
date,sales
01/01/2020,6000
01/02/2020,3000
01/03/2020,6000
01/04/2020,8000
01/05/2020,5000
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
01/11/2020,7000
01/12/2020,7000
01/13/2020,7000
01/14/2020,7000
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabelln och lägg till `Date` som ett fält och `sum(sales)` som mått.

Resultattabell

Datum	sum(sales)
01/01/2020	6000
01/02/2020	3000
01/03/2020	6000
01/04/2020	8000
01/05/2020	5000

Skriptet laddar bara de första fem posterna i `Sales`-tabellen.

Generic

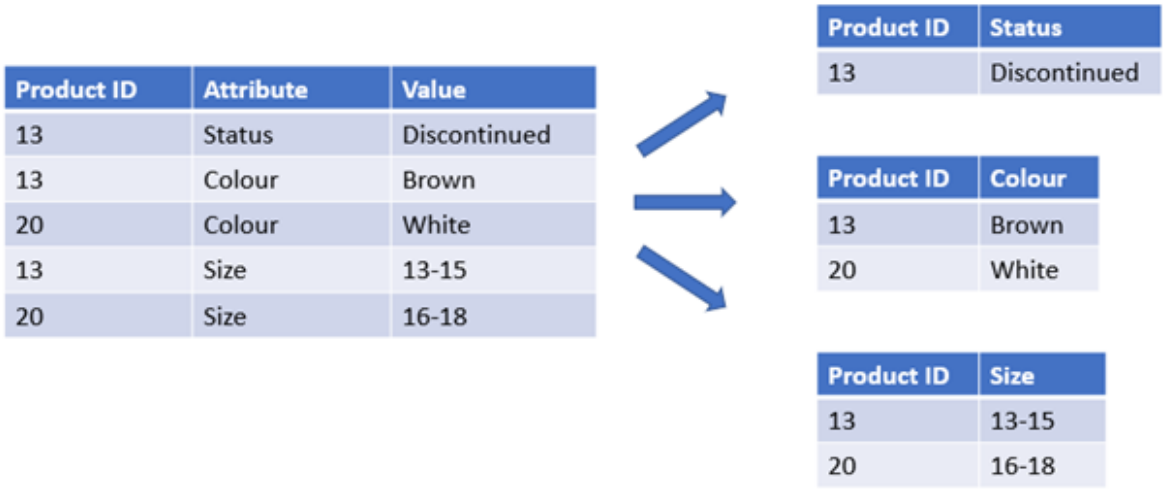
Generic-laddningsprefixet tillåter konvertering av data modellerade på element, attribut och värde (EAV) till en traditionell, normaliserad relationstabellstruktur. EAV-modellering kallas även för ”generisk datamodellering” eller ”öppet schema”.

Exempel på EAV-modellerad data och en motsvarande denormaliserad relationstabell

Product ID	Attribute	Value
13	Status	Discontinued
13	Colour	Brown
20	Colour	White
13	Size	13-15
20	Size	16-18

Product ID	Status	Colour	Size
13	Discontinued	Brown	13-15
20		White	16-18

Exempel på EAV-modellerade data och en motsvarande upsättning av normaliserade relationstabeller



Product ID	Attribute	Value
13	Status	Discontinued
13	Colour	Brown
20	Colour	White
13	Size	13-15
20	Size	16-18

Product ID	Status
13	Discontinued

Product ID	Colour
13	Brown
20	White

Product ID	Size
13	13-15
20	16-18

Även om det är tekniskt möjligt att ladda och analysera EAV-modellerad data i Qlik, är det ofta lättare att arbeta med en likvärdig traditionell relationsdatastruktur.

Syntax:

```
Generic( loadstatement | selectstatement )
```

Dessa ämnen kan hjälpa dig att arbeta med den här funktionen:

Relaterade ämnen

Avsnitt	Beskrivning
Crosstable (page 47)	`Crosstable`-laddningsprefixet omvandlar data som är horisontellt orienterade till vertikalt orienterade data. Ur ett rent funktionellt perspektiv utförs här den motsatta omvandlingen till `Generic`-laddningsprefixet, även om prefixen vanligtvis är avsett för helt andra användningsfall.
Generiska databaser i *Hantera data*	EAV-strukturerade datamodeller beskrivs ytterligare här.

Exempel 1 – Transformering av EAV-strukturerad data med det generiska laddningsprefixet

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller en datauppsättning som laddas in i en tabell med namnet `Transactions`. Datauppsättningen inkluderar ett datumfält. Standarddefinitionen för `MonthNames` används.

Laddningsskript

```
Products:
Generic
Load * inline [
Product ID, Attribute, Value
13, Status, Discontinued
13, Color, Brown
20, Color, White
13, Size, 13-15
20, Size, 16-18
2, Status, Discontinued
5, Color, Brown
2, Color, White
44, Color, Brown
45, Size, 16-18
45, Color, Brown
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `Color`.

Lägg till måttet:

```
=Count([Product ID])
```

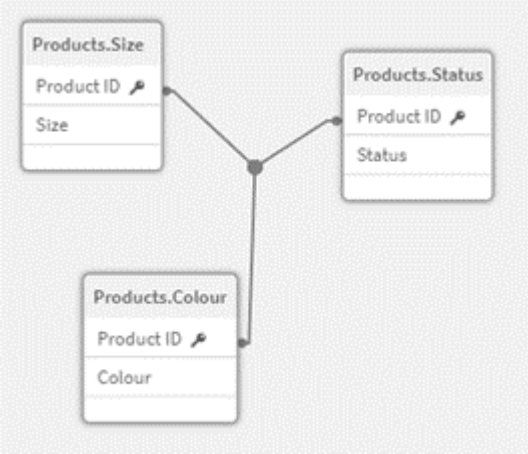
Nu kan du inspektera antalet produkter efter färg.

Resultattabell

Färg	=Antal([Produkt-ID])
Brun	4
Vit	2

Notera formen på datamodellen, där varje attribut har brutits ut i en separat tabell med namnet enligt den ursprungliga måltabelltaggen `Product`. Varje tabell har attributet som ett suffix. Ett exempel på detta är `Product.Color`. De resulterande produktattribututdataposterna associeras med `Product ID`.

Datamodellvyns representation av resultaten



Resulterande tabell med poster: Products.Status

Produkt-ID	Status
13	Avvecklad
2	Avvecklad

Resulterande tabell med poster: Products.Size

Produkt-ID	Storlek
13	13-15
20	16-18
45	16-18

Resulterande tabell med poster: Products.Color

Produkt-ID	Färg
13	Brun
5	Brun
44	Brun
45	Brun
20	Vit
2	Vit

Exempel 2 – Analysera EAV-strukturerad data med det generiska laddningsprefixet

Laddningsskript och diagramuttryck

Översikt

Det här exemplet visar hur du analyserar EAV-strukturerad data i sin ursprungliga form.

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller en datauppsättning som laddas in i en tabell med namnet `Products` i en EAV-struktur.

I det här exemplet räknar vi fortfarande produkter efter färgattribut. För att analysera data som är strukturerad på detta sätt måste du använda uttrycksnivåfiltrering av produkter med attributvärdet `Color`.

Dessutom är individuella attribut inte tillgängliga att välja som dimensioner eller fält, vilket gör det svårare att avgöra hur du skapar effektiva visualiseringar.

Laddningsskript

```
Products:
Load * Inline
[
Product ID, Attribute, Value
13, Status, Discontinued
13, Color, Brown
20, Color, White
13, Size, 13-15
20, Size, 16-18
2, Status, Discontinued
5, Color, Brown
2, Color, White
44, Color, Brown
45, Size, 16-18
45, Color, Brown
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `Value`.

Skapa följande mått:

```
=Count({<Attribute={'Color'}>} [Product ID])
```

Nu kan du inspektera antalet produkter efter färg.

Resulterande tabell med poster: Products.Status

Värde	=Antal({<Attribut={'Färg'}>} [Produkt-ID])
Brun	4
Vit	2

Exempel 3 – Denormalisering av de resulterande utdatatabellerna från en generisk laddning (avancerat)

Laddningsskript och diagramuttryck

Översikt

I det här exemplet visar vi hur den normaliserade datastrukturen som produceras av `Generic`-laddningsprefixet kan denormaliseras tillbaka till en konsoliderad `Product`-dimensionstabell. Detta är en avancerad modelleringsteknik som kan användas som en del av prestandajustering av datamodeller.

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskript

```
Products:

Generic
Load * inline [
Product ID, Attribute, Value
13, Status, Discontinued
13, Color, Brown
20, Color, White
13, Size, 13-15
20, Size, 16-18
2, Status, Discontinued
5, Color, Brown
2, Color, White
44, Color, Brown
45, Size, 16-18
45, Color, Brown
];

RENAME TABLE Products.Color TO Products;

OUTER JOIN (Products)
LOAD * RESIDENT Products.Size;

OUTER JOIN (Products)
LOAD * RESIDENT Products.Status;
DROP TABLES Products.Size,Products.Status;
```

Resultat

Öppna datamodellvyn och notera formen på den resulterande datamodellen. Endast en denormaliserad tabell finns. Det är en kombination av de tre mellanliggande utdatatabellerna: `Products.Size`, `Products.Status`och `Products.Color`.

Resulterande intern datamodell

Produkter
Produkt-ID
Status
Färg
Storlek

Resulterande tabell med poster: Products

Produkt-ID	**Status**	**Färg**	**Storlek**
13	Avvecklad	Brun	13-15
20	-	Vit	16-18
2	Avvecklad	Vit	-
5	-	Brun	-
44	-	Brun	-
45	-	Brun	16-18

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `Color`.

Lägg till måttet:

```
=Count([Product ID])
```

Resultattabell

Färg	**=Antal([Produkt-ID])**
Brun	4
Vit	2

Hierarchy

Prefixet **hierarchy** används för att omvandla en hierarkisk tabell till en tabell som fungerar i en Qlik Sense-datamodell. Det kan sättas framför en **LOAD**- eller **SELECT**-sats och det använder resultatet från den laddade satsen som indata för en tabellomvandling.

Prefixet skapar en expanderad nodtabell som normalt har samma antal poster som indatatabellen, men dessutom lagras varje nivå i hierarkin i ett separat fält. Sökvägsfältet kan användas i en trädstruktur.

Syntax:

```
Hierarchy (NodeID, ParentID, NodeName, [ParentName, [PathSource, [PathName,
[PathDelimiter, Depth]]]])(loadstatement | selectstatement)
```

Indatatabellen måste vara en ”adjacent nodes”-tabell. ”Adjacent nodes”-tabeller är tabeller där varje post motsvarar en nod och har ett fält som innehåller en referens till modernoden. I en sådan tabell lagras noden i en post men kan ha valfritt antal döttrar. Tabellen kan självklart innehålla ytterligare fält som beskriver nodernas attribut.

Prefixet skapar en expanderad nodtabell som normalt har samma antal poster som indatatabellen, men dessutom lagras varje nivå i hierarkin i ett separat fält. Sökvägsfältet kan användas i en trädstruktur.

Indatatabellen har vanligtvis exakt en post per nod. Om så är fallet innehåller utdatatabellen samma antal poster. Ibland kan det finnas noder med flera mödrar, d.v.s. varje nod motsvaras av flera poster i indatatabellen. Om så är fallet kan utdatatabellen innehålla fler poster än indatatabellen.

Alla noder med ett överordnat ID som inte hittas i nod-ID-kolumnen (inklusive de som saknar överordnat ID) behandlas som rötter. Dessutom kommer endast noder med en koppling till en rotnod – direkt eller indirekt – att laddas för att undvika cirkelreferenser.

Man kan skapa ytterligare fält som innehåller modernodens namn, sökvägen till noden och nodens djup.

Argument:

Argument

Argument	Beskrivning
NodeID	Namnet på det fält som innehåller nodens ID. Detta fält måste förekomma i indatatabellen.
ParentID	Namnet på det fält som innehåller modernodens nod-ID. Detta fält måste förekomma i indatatabellen.
NodeName	Namnet på det fält som innehåller nodens namn. Detta fält måste förekomma i indatatabellen.
ParentName	En sträng som används för att namnge det nya **ParentName**-fältet. Om den utelämnas skapas ej detta fält.
ParentSource	Namnet på det fält som innehåller namnet på den nod som används för att bygga nodens sökväg. Valfri parameter. Om den utelämnas används **NodeName**.
PathName	En sträng som används för att namnge det nya fältet **Path** som innehåller sökvägen från rot till nod. Valfri parameter. Om den utelämnas skapas ej detta fält.
PathDelimiter	En sträng som används som avgränsare i det nya **Path**-fältet. Valfri parameter. Om den utelämnas används '/'.

Argument	Beskrivning
Depth	En sträng som används för att namnge det nya fältet **Depth** som innehåller djupet på noden i hierarkin. Valfri parameter. Om den utelämnas skapas ej detta fält.

Exempel:

```
Hierarchy(NodeID, ParentID, NodeName, ParentName, NodeName, PathName, '\', Depth) LOAD *
inline [

NodeID, ParentID, NodeName

1, 4, London

2, 3, Munich

3, 5, Germany

4, 5, UK

5, , Europe

];
```

NodeID	ParentID	NodeName	NodeName1	NodeName2	NodeName3	ParentName	PathName	Depth
1	4	London	Europe	UK	London	UK	Europe\UK\London	3
2	3	Munich	Europe	Germany	Munich	Germany	Europe\Germany\Munich	3
3	5	Germany	Europe	Germany	-	Europe	Europe\Germany	2
4	5	UK	Europe	UK	-	Europe	Europe\UK	2
5		Europe	Europe	-	-	-	Europe	1

HierarchyBelongsTo

Detta prefix används för att omvandla en överordnad-underordnad hierarkisk tabell till en tabell som fungerar i en Qlik Sense-datamodell. Det kan sättas framför en **LOAD**- eller **SELECT**-sats och det använder resultatet från den laddade satsen som indata för en tabellomvandling.

Prefixet genererar en tabell som visar alla relationer mellan moder och dotter i hierarkin. Moderfälten kan sedan användas för att välja hela träd i hierarkin. Utdatatabellen innehåller i de flesta fall flera poster per nod.

Syntax:

```
HierarchyBelongsTo (NodeID, ParentID, NodeName, AncestorID, AncestorName,
[DepthDiff])(loadstatement | selectstatement)
```

Indatatabellen måste vara en ”adjacent nodes”-tabell. ”Adjacent nodes”-tabeller är tabeller där varje post motsvarar en nod och har ett fält som innehåller en referens till modernoden. I en sådan tabell lagras noden i en post men kan ha valfritt antal döttrar. Tabellen kan självklart innehålla ytterligare fält som beskriver nodernas attribut.

Prefixet genererar en tabell som visar alla relationer mellan moder och dotter i hierarkin. Moderfälten kan sedan användas för att välja hela träd i hierarkin. Utdatatabellen innehåller i de flesta fall flera poster per nod.

Ett ytterligare fält som innehåller djupskillnaden mellan noderna kan skapas.

Argument:

Argument

Argument	Beskrivning
NodeID	Namnet på det fält som innehåller nodens ID. Detta fält måste förekomma i indatatabellen.
ParentID	Namnet på det fält som innehåller modernodens nod-ID. Detta fält måste förekomma i indatatabellen.
NodeName	Namnet på det fält som innehåller nodens namn. Detta fält måste förekomma i indatatabellen.
AncestorID	En sträng som används för att namnge det nya fältet för överordnat ID som innehåller den överordnade nodens ID.
AncestorName	En sträng som används för att namnge det nya fältet Förfaders-ID som innehåller modernodens namn.
DepthDiff	En sträng som används för att namnge det nya **DepthDiff**-fältet som innehåller djupet på noden i hierarkin i förhållande till modernoden. Valfri parameter. Om den utelämnas skapas ej detta fält.

Exempel:

```
HierarchyBelongsTo (NodeID, AncestorID, NodeName, AncestorID, AncestorName, DepthDiff) LOAD *
inline [

NodeID, AncestorID, NodeName

1, 4, London
```

```
2, 3, Munich

3, 5, Germany

4, 5, UK

5, , Europe

];
```

Results

NodeID	AncestorID	NodeName	AncestorName	DepthDiff
1	1	London	London	0
1	4	London	UK	1
1	5	London	Europe	2
2	2	Munich	Munich	0
2	3	Munich	Germany	1
2	5	Munich	Europe	2
3	3	Germany	Germany	0
3	5	Germany	Europe	1
4	4	UK	UK	0
4	5	UK	Europe	1
5	5	Europe	Europe	0

Inner

Prefixen **join** och **keep** kan inledas med prefixet **inner**. Om det används före **join** anger det att inner join ska användas. Den resulterande tabellen kommer således endast att innehålla kombinationer av fältvärden från rådatatabeller där de länkande fältvärdena visas i en eller båda tabellerna. Om det används framför **keep**, anger det att båda rådatatabellerna ska reduceras till det gemensamma snittet av deras datamängder innan de lagras i Qlik Sense.

Syntax:

```
Inner ( Join | Keep) [ (tablename) ](loadstatement |selectstatement )
```

Argument:

Argument

Argument	Beskrivning
tablename	Namnet på den tabell som ska jämföras med den inlästa tabellen.
loadstatementeller selectstatement	**LOAD**- eller **SELECT**-satsen för den laddade tabellen.

Exempel

Laddningsskript

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
Table1:
Load * inline [
Column1, Column2
A, B
1, aa
2, cc
3, ee ];

Table2:
Inner Join Load * inline [
Column1, Column3
A, C
1, xx
4, yy ];
```

Resultat

Resultattabell

Column1	Column2	Column3
A	B	C
1	aa	xx

Förklaring

Det här exemplet demonstrerar Inner Join-utdata där endast värden som finns i den första (vänstra) och den andra (högra) tabellen länkas.

IntervalMatch

Det utökade **IntervalMatch**-prefixet används för att skapa en tabell där diskreta numeriska värden matchas till ett eller flera numeriska intervall. Det kan även matchas med värdena för en eller flera nycklar.

Syntax:

```
IntervalMatch (matchfield)(loadstatement | selectstatement )
```

```
IntervalMatch (matchfield,keyfield1 [ , keyfield2, ... keyfield5 ] )
(loadstatement | selectstatement )
```

Prefixet **IntervalMatch** måste placeras framför en **LOAD**- eller **SELECT**-sats som läser in intervallen. Fältet som innehåller de diskreta datapunkterna (Time i exemplet nedan) och ytterligare nycklar måste redan ha laddats i Qlik Sense före satsen med prefixet **IntervalMatch**. Prefixet kan inte av sig själv läsa in detta fält från databasens tabell. Prefixet omvandlar den inlästa tabellen med intervall och nycklar till en tabell som innehåller ytterligare en kolumn: de diskreta numeriska datapunkterna. Det utvidgar dessutom antalet poster så att den nya tabellen får en post per möjlig kombination av diskret datapunkt, intervall och värdet på nyckelfältet (nyckelfälten).

Intervallen kan överlappa varandra. De diskreta värdena länkas då till alla passande intervall.

När prefixet IntervalMatch utvidgas med nyckelfält används det för att skapa en tabell där diskreta numeriska värden matchas till ett eller flera numeriska intervall, samt med värdena för en eller flera kompletterande nycklar.

För att undvika att odefinierade intervallgränser ignoreras kan NULL-värden behöva mappas till andra fält som utgör intervallets undre och övre gränser. Detta kan hanteras av **NullAsValue**-satsen eller av ett explicit test som ersätter NULL-värden med ett numeriskt värde före eller efter någon av de diskreta numeriska datapunkterna.

Argument:

Argument

Argument	Beskrivning
matchfield	Det fält som innehåller de diskreta numeriska värden som ska länkas till intervallen.
keyfield	Fält som innehåller attribut som ska matchas i omvandlingen.
loadstatementor selectstatement	Måste resultera i en tabell där det första fältet innehåller den undre gränsen för varje intervall, det andra fältet innehåller den övre gränsen för varje intervall och i fallet då nyckelmatchning används innehåller det tredje och eventuella efterföljande fält det eller de nyckelfält som finns i **IntervalMatch**-satsen. Intervallen är alltid slutna, vilket innebär att start- och slutpunkterna alltid är inkluderade i intervallet. Om icke-numeriska gränser används, ignoreras intervallet (odefinerat).

Example 1:

I de två tabellerna nedan listar den första ett antal diskreta händelser och i den andra definieras start- och sluttiderna för produktionen av olika order. Med **IntervalMatch**-prefixet kan de två tabellerna kopplas logiskt för att exempelvis ta reda på vilka order som påverkades av driftstörningar och vilka order som behandlades i vilka skift.

```
EventLog:
LOAD * Inline [
Time, Event, Comment
```

```
00:00, 0, Start of shift 1
01:18, 1, Line stop
02:23, 2, Line restart 50%
04:15, 3, Line speed 100%
08:00, 4, Start of shift 2
11:43, 5, End of production
];

OrderLog:
LOAD * INLINE [
Start, End, Order
01:00, 03:35, A
02:30, 07:58, B
03:04, 10:27, C
07:23, 11:43, D
];

//Link the field Time to the time intervals defined by the fields Start and End.
Inner Join IntervalMatch ( Time )
LOAD Start, End
Resident OrderLog;
```

Tabellen **OrderLog** innehåller nu ytterligare en kolumn: *Time*. Antalet poster hat också utökats.

Table with additional column

Time	Start	End	Order
00:00	-	-	-
01:18	01:00	03:35	A
02:23	01:00	03:35	A
04:15	02:30	07:58	B
04:15	03:04	10:27	C
08:00	03:04	10:27	C
08:00	07:23	11:43	D
11:43	07:23	11:43	D

Example 2: (med keyfield)

Samma exempel som ovan, där man lägger till *ProductionLine* som ett nyckelfält.

```
EventLog:

LOAD * Inline [

Time, Event, Comment, ProductionLine

00:00, 0, Start of shift 1, P1

01:00, 0, Start of shift 1, P2
```

```
01:18, 1, Line stop, P1

02:23, 2, Line restart 50%, P1

04:15, 3, Line speed 100%, P1

08:00, 4, Start of shift 2, P1

09:00, 4, Start of shift 2, P2

11:43, 5, End of production, P1

11:43, 5, End of production, P2

];


OrderLog:

LOAD * INLINE [

Start, End, Order, ProductionLine

01:00, 03:35, A, P1

02:30, 07:58, B, P1

03:04, 10:27, C, P1

07:23, 11:43, D, P2

];


//Link the field Time to the time intervals defined by the fields Start and End and match the
values

// to the key ProductionLine.

Inner Join

IntervalMatch ( Time, ProductionLine )

LOAD Start, End, ProductionLine

Resident OrderLog;
```

En tabellbox kan du skapas enligt nedan:

Tablebox example

ProductionLine	Time	Event	Comment	Order	Start	End
P1	00:00	0	Start of shift 1	-	-	-
P2	01:00	0	Start of shift 1	-	-	-
P1	01:18	1	Line stop	A	01:00	03:35
P1	02:23	2	Line restart 50%	A	01:00	03:35
P1	04:15	3	Line speed 100%	B	02:30	07:58
P1	04:15	3	Line speed 100%	C	03:04	10:27
P1	08:00	4	Start of shift 2	C	03:04	10:27
P2	09:00	4	Start of shift 2	D	07:23	11:43
P1	11:43	5	End of production	-	-	-
P2	11:43	5	End of production	D	07:23	11:43

Join

join-prefixet länkar den inlästa tabellen till en existerande namngiven tabell eller den senast skapade datatabellen.

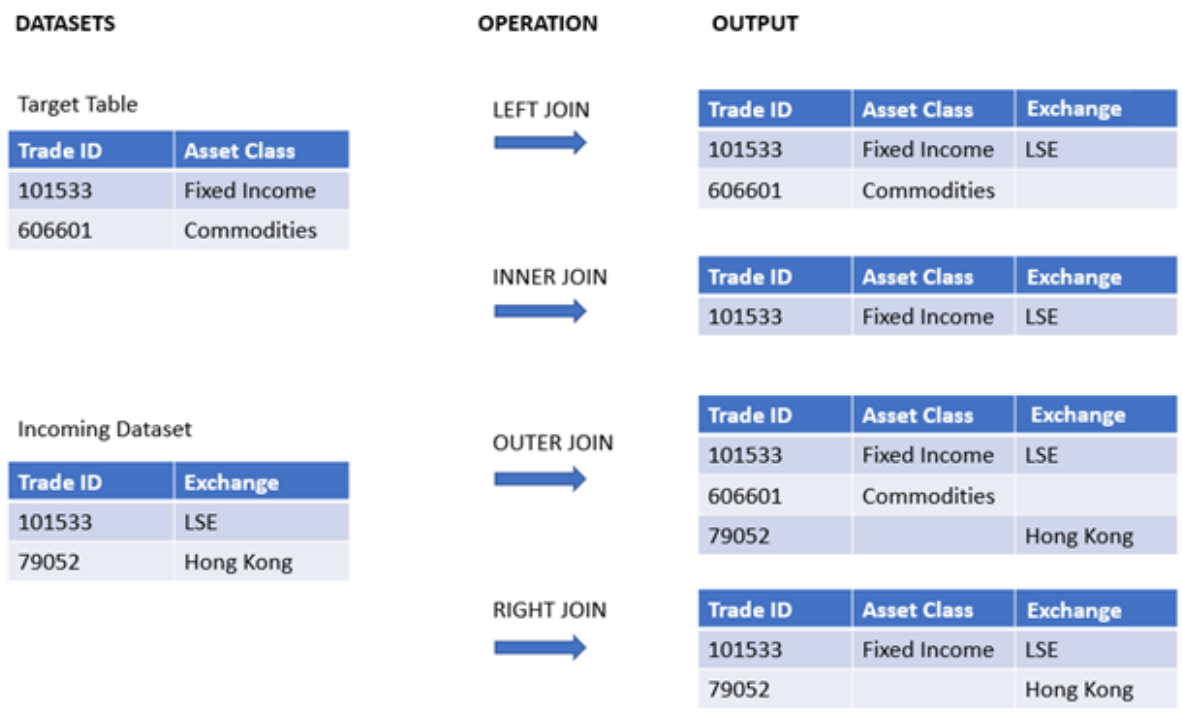
Effekten av att slå samman (join) data är att måltabellen utökas med ytterligare en uppsättning fält eller attribut, nämligen sådana som inte redan finns i måltabellen. Eventuella gemensamma fältnamn mellan källdatauppsättningen och måltabellen används för att räkna ut hur de nya inkommande posterna ska associeras. Detta kallas ofta för en "naturlig koppling" (natural join). En Qlik-kopplingsoperation kan leda till att den resulterande måltabellen får fler eller färre poster än den hade från början, beroende på hur unik kopplingsassociationen är och vilken typ av koppling som används.

Typer av kopplingar

Det finns fyra typer av kopplingar:

- Left join
- Inner join
- Outer join
- Right join

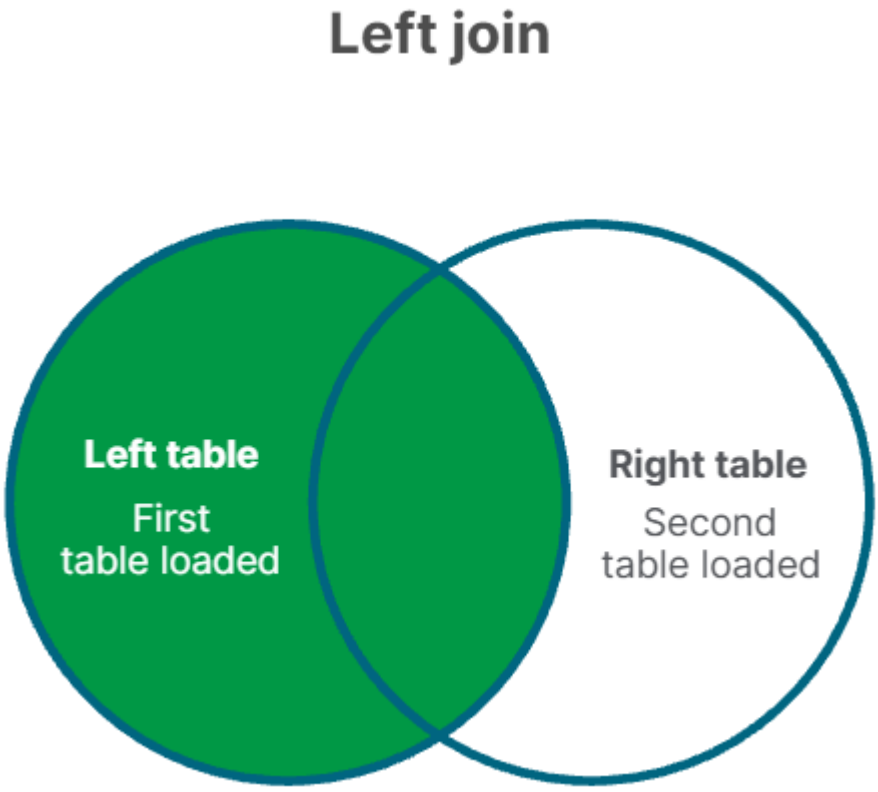
Exempel på resultatuppsättningar från olika typer av kopplingsoperationer



Left join

Left join är den vanligaste kopplingstypen. Om du till exempel har en transaktionsdatauppsättning och vill kombinera den med en referensdatauppsättning, skulle du normalt använda en `Left Join`. Du skulle ladda transaktionstabellen först och sedan ladda referensdatauppsättningen samtidigt som du kopplar den via ett `Left Join`-prefix till den redan laddade transaktionstabellen. En `Left Join` behåller alla transaktioner som de är och lägger till de kompletterande referensdatafälten där en matchning hittas.

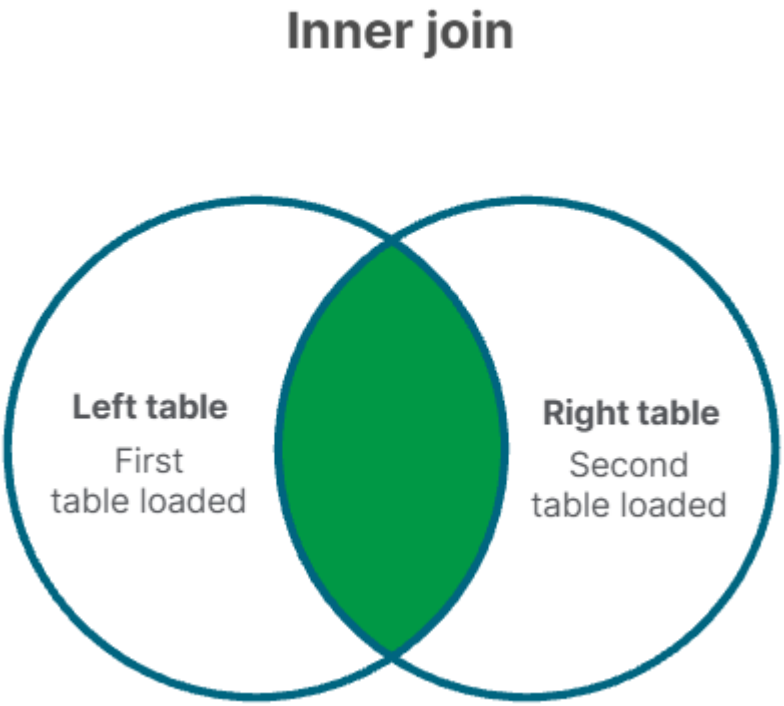
Left join



Inner join

När du har två datauppsättningar där du bara bryr dig om resultat där det finns en matchande association, bör du överväga att använda en `Inner Join`. Detta eliminerar alla poster från både de laddade källdata och måltabellen om ingen matchning hittas. Som ett resultat kan detta lämna din måltabell med färre poster än innan kopplingsoperationen ägde rum.

Inner join

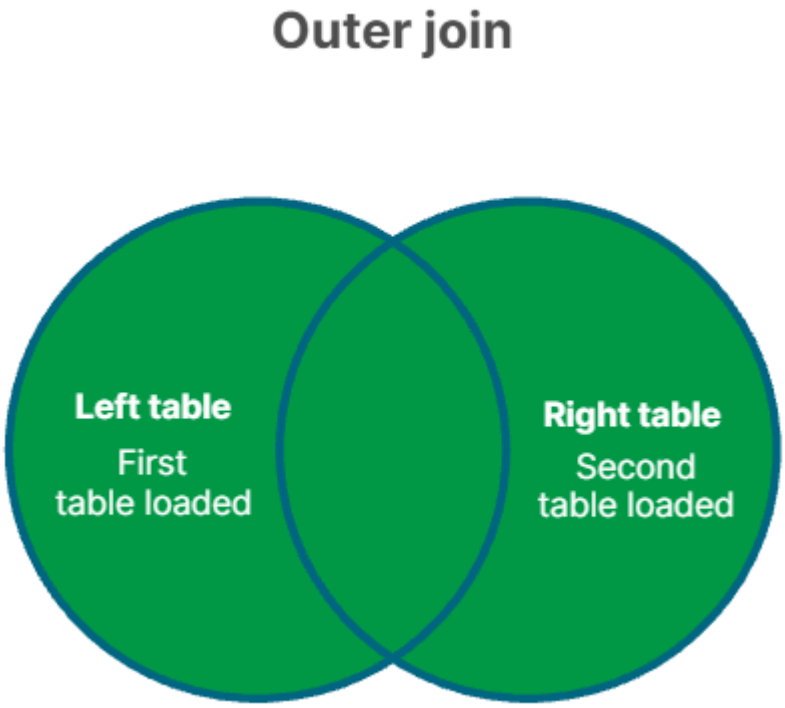


Outer join (standard)

När du behöver behålla både målposterna och alla inkommande poster använder du en `Outer Join`. Där ingen matchning hittas behålls fortfarande varje uppsättning poster medan fälten från den motsatta sidan av kopplingen förblir ofyllda (null).

Om typnyckelordet utelämnas är standardkopplingstypen en outer join.

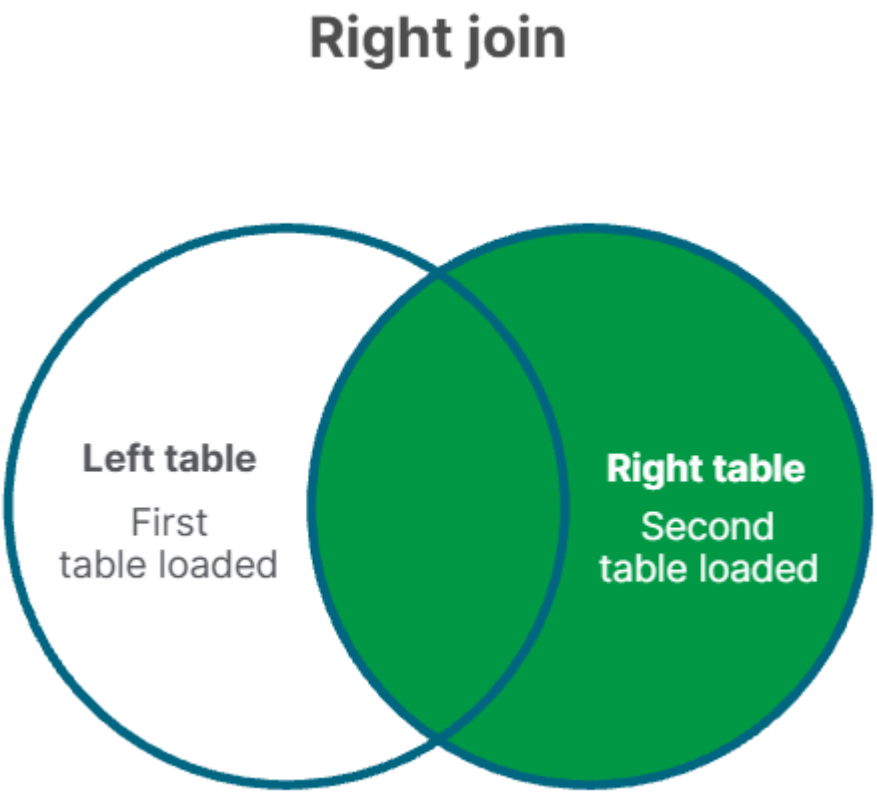
Outer join



Right join

Denna kopplingstyp behåller alla poster som ska laddas, samtidigt som den reducerar posterna i den tabell som är målet för kopplingen till endast de poster där det finns en associationsmatchning i de inkommande posterna. Detta är en nischad kopplingstyp som ibland används som ett sätt att trimma ner en redan förladdad tabell med poster till en nödvändig delmängd.

Right join



Cross join

Om det inte finns några gemensamma fältnamn mellan källan och målet för en kopplingsoperation, kommer kopplingen att resultera i en kartesisk produkt av alla rader – detta kallas en "cross join".

Exempel på resultatuppsättning från en "cross join"-operation

Syntax och argument

```
[inner | outer | left | right ]Join [ (tablename ) ]( loadstatement | selectstatement )
```

Argument

Argument	Beskrivning
tablename	Namnet på den tabell som ska jämföras med den inlästa tabellen.
loadstatementeller selectstatement	**LOAD**- eller **SELECT**-satsen för den laddade tabellen.

Relaterade ämnen

Dessa ämnen kan hjälpa dig att arbeta med den här funktionen:

Relaterade ämnen

Ämne	Beskrivning
Kombinera tabeller med Join och Keep i *Hantera data*	Det här ämnet ger ytterligare förklaring av begreppen att "koppla" (join) och "behålla" (keep) datauppsättningar.
Keep (page 94)	Laddningsprefixet `Keep` liknar prefixet `Join`, men det kombinerar inte käll- och måldatauppsättningarna. Istället trimmar det varje datauppsättning enligt den typ av operation som antagits (inner, outer, left eller right).

Exempel på laddningsskript - Left join

En left join behåller alla poster från din huvudtabell (måltabell) och lägger till matchande data från en andra tabell (källtabell). Om det inte finns någon matchning i källan visas huvudtabellens post fortfarande, men med tomma fält från källtabellen.

Left join: grundläggande exempel

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

Laddningsskript

```
Table1:
Load * inline [
Column1, Column2
A, B
1, aa
2, cc
3, ee ];

Table2:
Left Join Load * inline [
Column1, Column3
A, C
1, xx
4, yy ];
```

Resultat

Resulterande tabell

Column1	Column2	Column3
A	B	C
1	aa	xx
2	cc	-
3	ee	-

Det här exemplet demonstrerar utdata för **Left Join** där endast värden som finns i den första (vänstra) tabellen kopplas.

Exempel på left join: Berika beställningar med kundinformation

Översikt

Anta att du har en lista med beställningar, men att inte alla beställningar har en matchande kundpost.

Laddningsskript

```
Orders:
Load * inline [
OrderID, CustomerID, Amount
1, 101, 200
2, 102, 150
3, 104, 400
];
```

```
Customers:
Left Join (Orders)
Load * inline [
CustomerID, Name
101, Alice
102, Bob
103, Carol
];
```

Resultat

Om du expanderar förhandsgranskningsfönstret i datamodellvyn ser du en del av hela resultatuppsättningen organiserad i en tabell:

Förhandsgranskning av tabell i datamodellvyn

OrderID	CustomerID	Amount	Name
1	101	200	Alice
2	102	150	Bob
3	104	400	-

OrderID 1 och 2 har matchande kunder. *OrderID* 3 har ingen matchande kund, så *Name* är tomt.

Exempel på left join: Berika en måltabell med en referensdatauppsättning

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som representerar ändringsposter, som laddas in i en tabell med namnet `Changes`. Den inkluderar ett Status ID-nyckelfält.
- En andra datauppsättning som representerar ändringsstatusar, som laddas och kombineras med de ursprungliga ändringsposterna genom att koppla den med ett vänster `Join`-laddningsprefix.

Denna left join säkerställer att ändringsposterna förblir intakta samtidigt som statusattribut läggs till där en matchning i de inkommande statusposterna hittas baserat på ett gemensamt Status ID.

Laddningsskript

```
Changes:
Load * inline [
Change ID      Status ID       Scheduled Start Date    Scheduled End Date      Business Impact
10030  4       19/01/2022      23/02/2022      None
10015  3       04/01/2022      15/02/2022      Low
10103  1       02/04/2022      29/05/2022      Medium
10185  2       23/06/2022      08/09/2022      None
10323  1       08/11/2022      26/11/2022      High
10326  2       11/11/2022      05/12/2022      None
10138  2       07/05/2022      03/08/2022      None
```

```
10031  3       20/01/2022      25/03/2022      Low
10040  1       29/01/2022      22/04/2022      None
10134  1       03/05/2022      08/07/2022      Low
10334  2       19/11/2022      06/02/2023      Low
10220  2       28/07/2022      06/09/2022      None
10264  1       10/09/2022      17/10/2022      Medium
10116  1       15/04/2022      24/04/2022      None
10187  2       25/06/2022      24/08/2022      Low
] (delimiter is '\t');

Status:
Left Join (Changes)
Load * inline [
Status ID      Status  Sub Status
1      Open    Not Started
2      Open    Started
3      Closed  Completed
4      Closed  Cancelled
5      Closed  Obsolete
] (delimiter is '\t');
```

Resultat

Öppna datamodellvyn och notera datamodellens form. Endast en avnormaliserad tabell finns. Det är en kombination av alla ursprungliga ändringsposter, med de matchande statusattributen kopplade till varje ändringspost.

Resulterande intern datamodell

Changes
Change ID
Status ID
Scheduled Start Date
Scheduled End Date
Business Impact
Status
Sub Status

Om du expanderar förhandsgranskningsfönstret i datamodellvyn ser du en del av hela resultatuppsättningen organiserad i en tabell:

Förhandsgranskning av tabellen Changes i datamodellvyn

Change ID	Status ID	Scheduled Start Date	Scheduled End Date	Business Impact	Status	Sub Status
10030	4	19/01/2022	23/02/2022	None	Closed	Cancelled
10031	3	20/01/2022	25/03/2022	Low	Closed	Completed
10015	3	04/01/2022	15/02/2022	Low	Closed	Completed
10103	1	02/04/2022	29/05/2022	Medium	Open	Not Started
10116	1	15/04/2022	24/04/2022	None	Open	Not Started
10134	1	03/05/2022	08/07/2022	Low	Open	Not Started
10264	1	10/09/2022	17/10/2022	Medium	Open	Not Started
10040	1	29/01/2022	22/04/2022	None	Open	Not Started
10323	1	08/11/2022	26/11/2022	High	Open	Not Started
10187	2	25/06/2022	24/08/2022	Low	Open	Started
10185	2	23/06/2022	08/09/2022	None	Open	Started
10220	2	28/07/2022	06/09/2022	None	Open	Started
10326	2	11/11/2022	05/12/2022	None	Open	Started
10138	2	07/05/2022	03/08/2022	None	Open	Started
10334	2	19/11/2022	06/02/2023	Low	Open	Started

Eftersom den femte raden i tabellen Status (Status ID: '5', Status: 'Closed', Sub Status: 'Obsolete') inte motsvarar någon av posterna i tabellen Changes, visas inte informationen på den här raden i resultatuppsättningen ovan.

Gå tillbaka till Dataladdningsredigeraren. Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `Status`.

Lägg till detta mått:

```
=Count([Change ID])
```

Nu kan du inspektera antalet Changes per Status.

Resultattabell

Status	=Count([Change ID])
Open	12
Closed	3

Exempel på laddningsskript - Inner join

En inner join behåller endast poster som finns i båda tabellerna (baserat på matchande fältvärden).

Inner join: grundläggande exempel

Laddningsskript

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
Table1:
Load * inline [
Column1, Column2
A, B
1, aa
2, cc
3, ee ];

Table2:
Inner Join Load * inline [
Column1, Column3
A, C
1, xx
4, yy ];
```

Resultat

Resulterande tabell

Column1	Column2	Column3
A	B	C
1	aa	xx

Det här exemplet demonstrerar utdata för **Inner Join** där endast värden som finns i både den första (vänstra) och den andra (högra) tabellen kopplas.

Exempel på inner join: Reducera beställningar till endast skickade

Översikt

Du har beställningar och en lista över skickade beställningar. Du vill bara ha de beställningar som har skickats.

Laddningsskript

```
Orders:
Load * inline [
OrderID, Amount
1, 200
2, 150
3, 400
];

Shipped:
```

```
Inner Join (Orders)
Load * inline [
OrderID, ShipDate
2, 2022-05-01
3, 2022-05-02
4, 2022-05-03
];
```

Resultat

Om du expanderar förhandsgranskningsfönstret i datamodellvyn ser du en del av hela resultatuppsättningen organiserad i en tabell:

Förhandsgranskning av tabell i datamodellvyn

OrderID	Amount	ShipDate
2	150	2022-05-01
3	400	2022-05-02

Endast *OrderID* 2 och 3 visas, eftersom de finns i båda tabellerna.

Exempel på inner join: Kombinera endast matchande poster

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som representerar ändringsposter, som laddas in i en tabell med namnet `Changes`.
- En andra datauppsättning som representerar ändringsposter som härrör från källsystemet `JIRA`. Denna laddas och kombineras med de ursprungliga posterna genom att koppla den med ett `Inner Join`-laddningsprefix.

Denna `Inner Join` säkerställer att endast de fem ändringsposter som finns i båda datauppsättningarna behålls.

Laddningsskript

```
Changes:
Load * inline [
Change ID      Status ID       Scheduled Start Date    Scheduled End Date      Business Impact
10030  4       19/01/2022      23/02/2022      None
10015  3       04/01/2022      15/02/2022      Low
10103  1       02/04/2022      29/05/2022      Medium
10185  2       23/06/2022      08/09/2022      None
10323  1       08/11/2022      26/11/2022      High
10326  2       11/11/2022      05/12/2022      None
10138  2       07/05/2022      03/08/2022      None
```

```
10031  3       20/01/2022      25/03/2022      Low
10040  1       29/01/2022      22/04/2022      None
10134  1       03/05/2022      08/07/2022      Low
10334  2       19/11/2022      06/02/2023      Low
10220  2       28/07/2022      06/09/2022      None
10264  1       10/09/2022      17/10/2022      Medium
10116  1       15/04/2022      24/04/2022      None
10187  2       25/06/2022      24/08/2022      Low
] (delimiter is '\t');

JIRA_changes:
Inner Join (Changes)
Load
  [Ticket ID] AS [Change ID],
  [Source System]
inline
[
Ticket ID      Source System
10000  JIRA
10030  JIRA
10323  JIRA
10134  JIRA
10334  JIRA
10220  JIRA
20000  TFS
] (delimiter is '\t');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Source System`
- `Change ID`
- `Business Impact`

Nu kan du inspektera de fem resulterande posterna. Den resulterande tabellen från en `Inner Join` kommer endast att inkludera poster med matchande information i båda datauppsättningarna.

Resultattabell

Source System	Change ID	Business Impact
JIRA	10030	None
JIRA	10134	Low
JIRA	10220	None
JIRA	10323	High
JIRA	10334	Low

Exempel på laddningsskript - Outer join

En outer join behåller alla poster från båda tabellerna. Där det inte finns någon matchning lämnas fälten tomma.

Outer join: Grundläggande exempel

Laddningsskript

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
Table1:
Load * inline [
Column1, Column2
A, B
1, aa
2, cc
3, ee ];

Table2:
Outer Join Load * inline [
Column1, Column3
A, C
1, xx
4, yy ];
```

Resultat

Resulterande tabell

Column1	Column2	Column3
A	B	C
1	aa	xx
2	cc	-
3	ee	-
4	-	yy

I det här exemplet slås de två tabellerna, *Table1* och *Table2*, samman till en enda tabell med namnet *Table1*. I fall som detta används ofta prefixet **outer** för att koppla flera tabeller till en enda tabell för att utföra aggregeringar över värdena i en enda tabell.

Outer join: Skapa en fullständig lista över alla kunder och beställningar

Översikt

Du vill ha en komplett lista över alla kunder och beställningar, även om vissa kunder inte har några beställningar och vissa beställningar inte har någon kundpost.

Laddningsskript

```
Orders:
Load * inline [
OrderID, CustomerID
1, 101
2, 102
3, 104
];

Customers:
Outer Join (Orders)
Load * inline [
CustomerID, Name
101, Alice
102, Bob
103, Carol
];
```

Resultat

Om du expanderar förhandsgranskningsfönstret i datamodellvyn ser du en del av hela resultatuppsättningen organiserad i en tabell:

Förhandsgranskning av tabell i datamodellvyn

OrderID	CustomerID	Name
1	101	Alice
2	102	Bob
3	104	-
-	103	Carol

OrderID 3 har ingen matchande kund (*Name* är tomt).

CustomerID 103 (*Carol*) har ingen matchande beställning (*OrderID* är tomt).

Outer join: Kombinera överlappande postuppsättningar

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som representerar ändringsposter, som laddas in i en tabell med namnet `Changes`.

- En andra datauppsättning som representerar ändringsposter som härrör från källsystemet `JIRA`, som laddas och kombineras med de ursprungliga posterna genom att koppla den med ett `Outer Join`-laddningsprefix.

Detta säkerställer att alla överlappande ändringsposter från båda datauppsättningarna behålls.

Laddningsskript

```
// 8 Change records

Changes:
Load * inline [
Change ID      Status ID       Scheduled Start Date    Scheduled End Date      Business Impact
10030  4       19/01/2022      23/02/2022      None
10015  3       04/01/2022      15/02/2022      Low
10138  2       07/05/2022      03/08/2022      None
10031  3       20/01/2022      25/03/2022      Low
10040  1       29/01/2022      22/04/2022      None
10134  1       03/05/2022      08/07/2022      Low
10334  2       19/11/2022      06/02/2023      Low
10220  2       28/07/2022      06/09/2022      None
] (delimiter is '\t');

// 6 Change records

JIRA_changes:
Outer Join (Changes)
Load
  [Ticket ID] AS [Change ID],
  [Source System]
inline
[
Ticket ID      Source System
10030  JIRA
10323  JIRA
10134  JIRA
10334  JIRA
10220  JIRA
10597  JIRA
] (delimiter is '\t');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Source System`
- `Change ID`
- `Business Impact`

Nu kan du inspektera de 10 resulterande posterna.

Resultattabell

Source System	Change ID	Business Impact
JIRA	10030	None
JIRA	10134	Low
JIRA	10220	None
JIRA	10323	-
JIRA	10334	Low
JIRA	10597	-
-	10015	Low
-	10031	Low
-	10040	None
-	10138	None

Exempel på laddningsskript - Right join

En right join behåller alla poster från källtabellen och lägger till matchande information från måltabellen. Om måltabellen inte har någon matchning är fälten från den tomma.

Right join: grundläggande exempel

Laddningsskript

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
Table1:
Load * inline [
Column1, Column2
A, B
1, aa
2, cc
3, ee ];

Table2:
Right Join Load * inline [
Column1, Column3
A, C
1, xx
4, yy ];
```

Resultat

Resulterande tabell

Column1	Column2	Column3
A	B	C
1	aa	xx
4	-	yy

Det här exemplet demonstrerar utdata för **Right Join** där endast värden som finns i den andra (högra) tabellen kopplas.

Exempel på right join: kombinera alla kunder med matchande beställningar

Översikt

Din källtabell är en lista över kunder; du vill ha alla kunder, plus eventuella matchande beställningar.

Laddningsskript

```
Orders:
Load * inline [
OrderID, CustomerID
1, 101
2, 102
3, 104
];

Customers:
Right Join (Orders)
Load * inline [
CustomerID, Name
101, Alice
102, Bob
103, Carol
];
```

Resultat

Om du expanderar förhandsgranskningsfönstret i datamodellvyn ser du en del av hela resultatuppsättningen organiserad i en tabell:

Förhandsgranskning av tabell i datamodellvyn

CustomerID	Name	OrderID
101	Alice	1

CustomerID	Name	OrderID
102	Bob	2
103	Carol	-

Alla kunder visas. Carol har inga beställningar (OrderID är tomt).

Exempel på right join: trimma ner en måltabell med en sekundär huvuddatauppsättning

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som representerar ändringsposter, som laddas in i en tabell med namnet `Changes`.
- En andra datauppsättning som representerar ändringsposter som härrör från källsystemet `Teamwork`. Denna laddas och kombineras med de ursprungliga posterna genom att koppla den med ett `Right Join`-laddningsprefix.

Detta säkerställer att endast `Teamwork`-ändringsposter behålls, samtidigt som inga `Teamwork`-poster går förlorade om måltabellen inte har ett matchande `Change ID`.

Laddningsskript

```
Changes:

Load * inline [
Change ID      Status ID       Scheduled Start Date    Scheduled End Date      Business Impact
10030  4       19/01/2022      23/02/2022      None
10015  3       04/01/2022      15/02/2022      Low
10103  1       02/04/2022      29/05/2022      Medium
10185  2       23/06/2022      08/09/2022      None
10323  1       08/11/2022      26/11/2022      High
10326  2       11/11/2022      05/12/2022      None
10138  2       07/05/2022      03/08/2022      None
10031  3       20/01/2022      25/03/2022      Low
10040  1       29/01/2022      22/04/2022      None
10134  1       03/05/2022      08/07/2022      Low
10334  2       19/11/2022      06/02/2023      Low
10220  2       28/07/2022      06/09/2022      None
10264  1       10/09/2022      17/10/2022      Medium
10116  1       15/04/2022      24/04/2022      None
10187  2       25/06/2022      24/08/2022      Low
] (delimiter is '\t');

Teamwork_changes:
Right Join (Changes)
Load
  [Ticket ID] AS [Change ID],
  [Source System]
```

```
inline
[
Ticket ID      Source System
10040  Teamwork
10015  Teamwork
10103  Teamwork
10031  Teamwork
50231  Teamwork
] (delimiter is '\t');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Source System`
- `Change ID`
- `Business Impact`

Nu kan du inspektera de fem resulterande posterna.

Resultattabell

Source System	Change ID	Business Impact
Teamwork	10015	Low
Teamwork	10031	Low
Teamwork	10040	None
Teamwork	10103	Medium
Teamwork	50231	-

Felsökning - Cross join

Exempel på felsökning av laddningsskript: inga gemensamma fältnamn

Översikt

Om du kopplar två tabeller utan matchande fältnamn kommer varje rad från den första tabellen att kombineras med varje rad från den andra – vilket potentiellt producerar ett enormt antal poster (en kartesisk produkt) och leder till oväntade dataassociationer.

Laddningsskript

```
TableA:
Load * inline [
A_ID, ValueA
1, foo
2, bar
];

TableB:
```

```
Join (TableA)
Load * inline [
B_ID, ValueB
10, baz
11, qux
];
```

Resultat

Om du expanderar förhandsgranskningsfönstret i datamodellvyn ser du en del av hela resultatuppsättningen organiserad i en tabell:

Förhandsgranskning av tabell i datamodellvyn

A_ID	ValueA	B_ID	ValueB
1	foo	10	baz
1	foo	11	qux
2	bar	10	baz
2	bar	11	qux

Varje *A_ID* paras ihop med varje *B_ID* (totalt 4 rader).

Kontrollera alltid att dina fältnamn matchar som avsett innan du kör en koppling.

Keep

Prefixet **keep** liknar prefixet **join**. Precis som prefixet **join** jämför det här prefixet den inlästa tabellen med en befintlig namngiven tabell eller den senaste tidigare skapade datatabellen, men i stället för att koppla ihop den inlästa tabellen med en befintlig tabell gör den så att den ena eller båda tabellerna minskas innan de sparas i Qlik Sense, baserat på intersektionen av tabelldata. Jämförelsen som görs motsvarar en naturlig länkning (join) som görs över alla gemensamma fält. Den görs med andra ord likadant som en motsvarande join. Skillnaden är dock att de två tabellerna inte länkas utan lagras i Qlik Sense som två tabeller med olika namn.

Syntax:

```
(inner | left | right) keep [(tablename ) ]( loadstatement | selectstatement
)
```

Prefixet **keep** måste inledas med något av följande prefix: **inner**, **left** eller **right**.

Det explicita **join** prefixet i Qlik Sense-skriptspråket skapar en fullständig länkning (full join) mellan de båda tabellerna. Resultatet är en tabell. I många fall resulterar sådana länkningar i mycket stora tabeller. En av finesserna med Qlik Sense är just att programmet gör associationer mellan flera tabeller istället för att

skapa länkningar, vilket minskar minnesanvändningen avsevärt, ökar hastigheten och ger större flexibilitet. Explicita länkningar bör därför normalt undvikas i Qlik Sense-skript. Keep-funktionaliteten har utvecklats för att minska antalet fall där man måste använda länkning.

Argument:

Argument

Argument	Beskrivning
tablename	Namnet på den tabell som ska jämföras med den inlästa tabellen.
loadstatementeller selectstatement	**LOAD**- eller **SELECT**-satsen för den laddade tabellen.

Exempel:

```
Inner Keep LOAD * from abc.csv;

Left Keep SELECT * from table1;

tab1:

LOAD * from file1.csv;

tab2:

LOAD * from file2.csv;

.. .. ..

Left Keep (tab1) LOAD * from file3.csv;
```

Left

Prefixen **Join** och **Keep** kan inledas med prefixet **left**.

Om det används före **join** anger det att left join ska användas. Den resulterande tabellen kommer således att innehålla endast kombinationer av fältvärden från rådatatabeller där de länkande fältvärdena visas i den första tabellen. Om det används framför **keep**, anger det att den andra rådatatabellen ska reduceras till det gemensamma snittet med den första tabellen innan den lagras i Qlik Sense.

Letade du efter strängfunktionen med samma namn? Se: Left (page 1780)

Syntax:

```
Left ( Join | Keep) [ (tablename) ](loadstatement | selectstatement)
```

Argument:

Argument

Argument	Beskrivning
tablename	Namnet på den tabell som ska jämföras med den inlästa tabellen.
loadstatementeller selectstatement	**LOAD**- eller **SELECT**-satsen för den laddade tabellen.

Exempel

Laddningsskript

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
Table1:
Load * inline [
Column1, Column2
A, B
1, aa
2, cc
3, ee ];

Table2:
Left Join Load * inline [
Column1, Column3
A, C
1, xx
4, yy ];
```

Resultat

Resultattabell

Column1	Column2	Column3
A	B	C
1	aa	xx
2	cc	-
3	ee	-

Förklaring

Det här exemplet demonstrerar Left Join-utdata där endast värden i den första (vänstra) tabellen kopplas.

Exempel

Laddningsskript

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
Table1:
Load * inline [
Column1, Column2
A, B
1, aa
2, cc
3, ee ];

Table2:
Left Join Load * inline [
Column1, Column3
A, C
1, xx
4, yy ];
```

Resultat

Resultattabell

Column1	Column2	Column3
A	B	C
1	aa	xx
2	cc	-
3	ee	-

Förklaring

Det här exemplet demonstrerar Left Join-utdata där endast värden i den första (vänstra) tabellen kopplas.

Mappning

mapping-prefixet används för att skapa en mappningstabell som exempelvis kan användas för att ersätta fältvärden och fältnamn under skriptexekveringen.

Syntax:

```
Mapping( loadstatement | selectstatement )
```

Prefixet **mapping** kan sättas framför en **LOAD**- eller **SELECT**-sats. Det sparar resultatet från laddningssatsen i form av en mappningstabell. Mappning är ett effektivt sätt att byta ut fältvärden under skriptexekvering, t.ex. byta ut US, U.S. eller Amerika mot USA. En mappningstabell måste bestå av två kolumner, den första innehållande jämförelsevärden och den andra innehållande önskade mappningsvärden. Mappningstabeller lagras tillfälligt i minnet och avlägsnas automatiskt efter skriptexekveringen. Alternativt kan du släppa mappningstabeller tidigare i skriptexekveringen med hjälp av **Drop mapping table**-varianten av *Drop table (page 176)*-satsen.

Mappningstabellens innehåll kan exempelvis nås via **Map … Using**- eller **Rename Field**-satsen, samt via **Applymap()**- eller**Mapsubstring()**-funktionen.

Exempel:

I det här exemplet laddar vi en lista med säljare med en landskod som står för det land där de är bosatta. Vi använder en tabell som mappar en landskod till ett land för att ersätta landskoden med landets namn. Enbart tre länder är definierade i mappningstabellen, övriga landskoder mappas till 'Rest of the world'.

```
// Load mapping table of country codes:
map1:
mapping LOAD *
Inline [
CCode, Country
Sw, Sweden
Dk, Denmark
No, Norway
] ;
// Load list of salesmen, mapping country code to country
// If the country code is not in the mapping table, put Rest of the world
Salespersons:
LOAD *,
ApplyMap('map1', CCode,'Rest of the world') As Country
Inline [
CCode, Salesperson
Sw, John
Sw, Mary
Sw, Per
Dk, Preben
Dk, Olle
No, Ole
Sf, Risttu] ;
// We don't need the CCode anymore
Drop Field 'CCode';
```

Den resulterande tabellen ser ut så här:

Mapping table

Salesperson	Country
John	Sweden
Mary	Sweden
Per	Sweden
Preben	Denmark
Olle	Denmark
Ole	Norway
Risttu	Rest of the world

Slå samman

Merge-prefixet kan adderas till valfri **LOAD**- eller **SELECT**-sats i skriptet för att specificera att den laddade tabellen ska slås samman med en annan tabell. Det specificerar även att satsen ska köras i en delvis laddning.

Används ofta när du laddar en ändringslogg och vill använda denna för att tillämpa inserts, updates och deletes på en befintlig tabell.

> *För att delvis laddning ska fungera som avsett måste applikationen öppnas med data innan en delvis laddning utlöses.*

Gör en delvis laddning med knappen **Ladda**. Du kan även använda Qlik Engine JSON API.

Syntax:

```
Merge [only] [(SequenceNoField [, SequenceNoVar])] On ListOfKeys [Concatenate [(TableName)]] (loadstatement | selectstatement)
```

Argument:

Argument

Argument	Beskrivning
only	En valfri kvalificerare som innebär att satsen bara ska exekveras vid delvisa laddningar. Satsen ska ignoreras vid vanliga (ej delvisa) laddningar.
SequenceNoField	Namnet på det fält som innehåller en tidsmarkör eller ett ordningstal som definierar ordningen för operationerna.
SequenceNoVar	Namnet på den variabel som blir tilldelad det högsta värdet för SequenceNoField i tabellen som slås ihop.
ListOfKeys	En kommaavgränsad lista med fältnamn som specificerar den primära nyckeln.
Operation	Det första fältet i LOAD-satsen måste innehålla operationen som textsträng: "Insert", "Update"' eller "Delete". "i", "u" och "d" accepteras också.

Allmän funktionalitet

Vid en vanlig (ej delvis) laddning fungerar konstruktionen **Merge LOAD** som en vanlig **LOAD**-sats, med tillägget att äldre, obsoleta poster tas bort, liksom poster som markerats för borttagning. Det första fältet i **LOAD**-satsen måste innehålla information om operationen: Insert, Update eller Delete.

För varje post som laddas jämförs postens identifierare med poster som laddats tidigare, och bara den senaste posten (enligt ordningstalet) sparas. Om den senaste posten har en Delete-markering sparas inte någon.

Måltabell

Vilken tabell som ska ändras bestäms av uppsättningen fält. Om en tabell med samma uppsättning fält (förutom det första fältet; operationen) redan finns, kommer detta att vara den relevanta tabellen att ändra. Alternativt kan prefixet **Concatenate** användas för att specificera tabellen. Om måltabellen inte är fastställd, lagras resultatet av **Merge LOAD**-konstruktionen i en ny tabell.

Om prefixet Concatenate används, har den resulterande tabellen en uppsättning fält som motsvarar föreningen av den befintliga tabellen och indata till sammanslagningen. Därför kan måltabellen få fler fält än ändringsloggen som används som indata för sammanslagningen.

En delvis laddning gör samma sak som en fullständig laddning. En skillnad är att en partiell omladdning sällan skapar en ny tabell. Om du inte har använts satsen **Only**, finns alltid en måltabell med samma uppsättning fält från föregående skriptkörning.

Sekvensnummer

Om den laddade ändringsloggen är en ackumulerad logg, det vill säga innehåller ändringar som redan har laddats, kan parametern SequenceNoVar användas i en **Where**-sats för att begränsa mängden indata. Då kan **Merge LOAD** formuleras så att bara poster där fältet SequenceNoField är större än SequenceNoVar laddas. När exekveringen är klar tilldelar **Merge LOAD** ett nytt värde till SequenceNoVar med det största värde som finns i fältet SequenceNoField.

Drift

Merge LOAD kan ha färre fält än måltabellen. De olika operationerna behandlar saknade fält olika:

Insert: fält som saknas i **Merge LOAD**, men som finns i måltabellen, får NULL i måltabellen.

Delete: saknade fält påverkar inte resultatet. Relevanta poster raderas ändå.

Update: fält som listas i **Merge LOAD** uppdateras i måltabellen. Saknade fält ändras inte. Det betyder att de två följande påståendena inte är identiska:

- Sammanfoga på nyckel Läs in och konkatenera 'U' som Åtgärd, Nyckel, F1, Null() som F2 Från ...;
- Sammanfoga på nyckel Läs in och konkatenera 'U' som Åtgärd, Nyckel, F1 Från ...;

Den första satsen uppdaterar de listade posterna och ändrar F2 to NULL. Den andra ändrar inte F2, utan lämnar i stället värdena i måltabellen.

Exempel

Exempel 1: enkel sammanslagning med specificerad tabell

I det här exemplet laddas en inline-tabell med namnet Persons med tre rader. Sedan ändrar **Merge** tabellen på följande sätt:

- Lägger till raden *Mary, 4*
- Tar bort raden *Steven, 3*
- Tilldelar talet *5* till *Jake*

Variabeln *LastChangeDate* anges som det högsta värdet i kolumnen *ChangeDate* efter att **Merge** har exekverats.

Laddningsskript

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
Set DateFormat='D/M/YYYY';
Persons:
load * inline [
Name, Number
Jake, 3
Jill, 2
Steven, 3
];

Merge (ChangeDate, LastChangeDate) on Name Concatenate(Persons)
LOAD * inline [
Operation, ChangeDate,   Name,     Number
Insert,    1/1/2021,     Mary,     4
Delete,    1/1/2021,     Steven,
Update,    2/1/2021,     Jake,     5
];
```

Resultat

Före **Merge Load**ser tabellen ut så här:

Resulting table

Name	Number
Jake	3
Jill	2
Steven	3

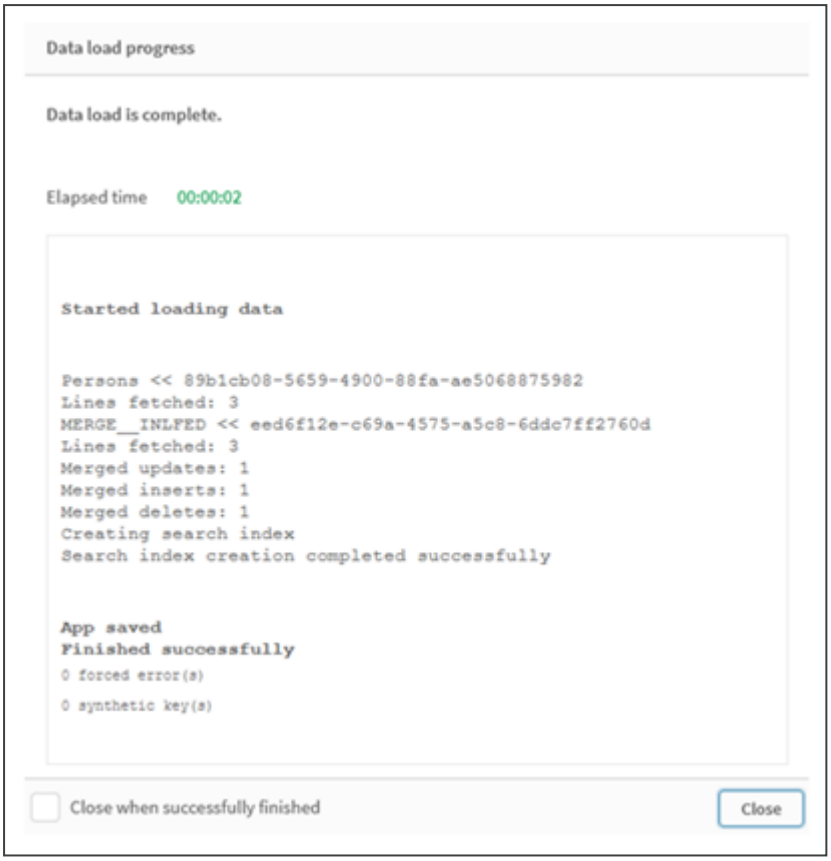
Efter **Merge Load** ser tabellen ut så här:

Resulting table

ChangeDate	Name	Number
2/1/2021	Jake	5
-	Jill	2
1/1/2021	Mary	4

När data laddas visas operationerna som utförs i dialogrutan **Dataladdning**:

Dataladdningsdialogrutan



Exempel 2: dataladdningsskript med saknade fält

I detta exempel läses samma data som ovan in, men nu med ett id för varje person.

Merge ändrar tabellen på följande sätt:

- Lägger till raden *Mary, 4*
- Tar bort raden *Steven, 3*
- Tilldelar talet *5* till *Jake*
- Tilldelar talet *6* till *Jill*

Laddningsskript

Här använder vi två **Merge Load**-satser. En för "Infoga" och "Radera", och en andra för "Uppdatera".

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
Set DateFormat='D/M/YYYY';
Persons:
Load * Inline [
PersonID, Name, Number
1, Jake, 3
2, Jill, 2
3, Steven, 3
];
```

```
Merge (ChangeDate, LastChangeDate) on PersonID Concatenate(Persons)
Load * Inline [
Operation, ChangeDate,   PersonID, Name,     Number
Insert,    1/1/2021,     4,        Mary,     4
Delete,    1/1/2021,     3,        Steven,
];

Merge (ChangeDate, LastChangeDate) on PersonID Concatenate(Persons)
Load * Inline [
Operation, ChangeDate,   PersonID, Number
Update,    2/1/2021,     1,        5
Update,    3/1/2021,     2,        6
];
```

Resultat

Efter **Merge Load**-satserna ser tabellen ut så här:

Resulting table

PersonID	ChangeDate	Name	Number
1	2/1/2021	Jake	5
2	3/1/2021	Jill	6
4	1/1/2021	Mary	4

Observera att den andra **Merge**-satsen inte inkluderar fältet **Name**, och därför har namnen inte ändrats.

Exempel 3: dataladdningsskript – delvis laddning med hjälp av en Where-sats med ChangeDate

I följande exempel specificerar **Only**-argumentet att **Merge**-kommandot endast exekveras vid en delvis laddning. Uppdateringar filtreras utifrån det tidigare beskrivna LastChangeDate. När **Merge** är klart tilldelas LastChangeDate-variabeln det största värdet i kolumnen ChangeDate, som bearbetats under sammanslagningen.

Laddningsskript

```
Merge Only (ChangeDate, LastChangeDate) on Name Concatenate(Persons)
LOAD Operation, ChangeDate,  Name, Number
from [lib://ChangeFilesFolder/BulkChangesInPersonsTable.csv] (txt)
where ChangeDate >='$(LastChangeDate)';
```

NoConcatenate

Med **NoConcatenate**-prefixet tvingas skriptet att behandla två inlästa tabeller med identiska fält som två separata interna tabeller. Annars skulle tabellerna konkateneras automatiskt.

Syntax:

```
NoConcatenate( loadstatement | selectstatement )
```

Om en tabell är laddad som innehåller ett identiskt antal fält och matchande fältnamn som en tabell som laddats tidigare i skriptet kommer Qlik Sense som standard att autokonkatenera de här två tabellerna. Detta kommer att ske även om den andra tabellen har ett annat namn.

Men om skriptprefixet NoConcatenate anges före load-satsen eller select-satsen i den andra tabellen kommer dessa två tabeller att laddas separat.

Ett typiskt användarfall för NoConcatenate är när du behöver skapa en tillfällig kopia av en tabell för att utföra någon tillfällig transformering på den kopian, samtidigt som du behåller eh en kopia av ursprungliga data. NoConcatenate säkerställer att du kan göra den kopian utan att implicit lägga till den tillbaka på källtabellen.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i SET DateFormat-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Funktionsexempel

Exempel	Resultat
`Source: LOAD A,B from file1.csv;` `CopyOfSource: NoConcatenate LOAD A,B resident Source;`	En tabell med A och B som mått laddas. En andra tabell med samma fält laddas separat genom att använda NoConcatenate-variabeln.

Exempel 1 – implicit konkatenering

Laddningsskript och resultat

Översikt

I det här exemplet kommer du att lägga till två laddningsskript i ordningsföljd.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En initial datauppsättning med datum och belopp som skickas till en tabell som heter Transactions.

Laddningsskript 1

```
Transactions:
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
```

```
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `amount`

Resultattabell 1

id	datum	belopp
1	08/30/2018	23.56
2	09/07/2018	556,31
3	09/16/2018	5,75
4	2018-09-22	125,00
5	22/09/2018	484,21
6	22/09/2018	59,18
7	23/09/2018	177,42

Laddningsskript 2

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En andra datauppsättning med identiska fält skickas till en tabell som heter `Sales`.

```
Sales:
LOAD
*
Inline [
id, date, amount
8, 10/01/2018, 164.27
9, 10/03/2018, 384.00
10, 10/06/2018, 25.82
11, 10/09/2018, 312.00
12, 10/15/2018, 4.56
13, 10/16/2018, 90.24
14, 10/18/2018, 19.32
];
```

Resultat

Ladda data och gå till tabellen.

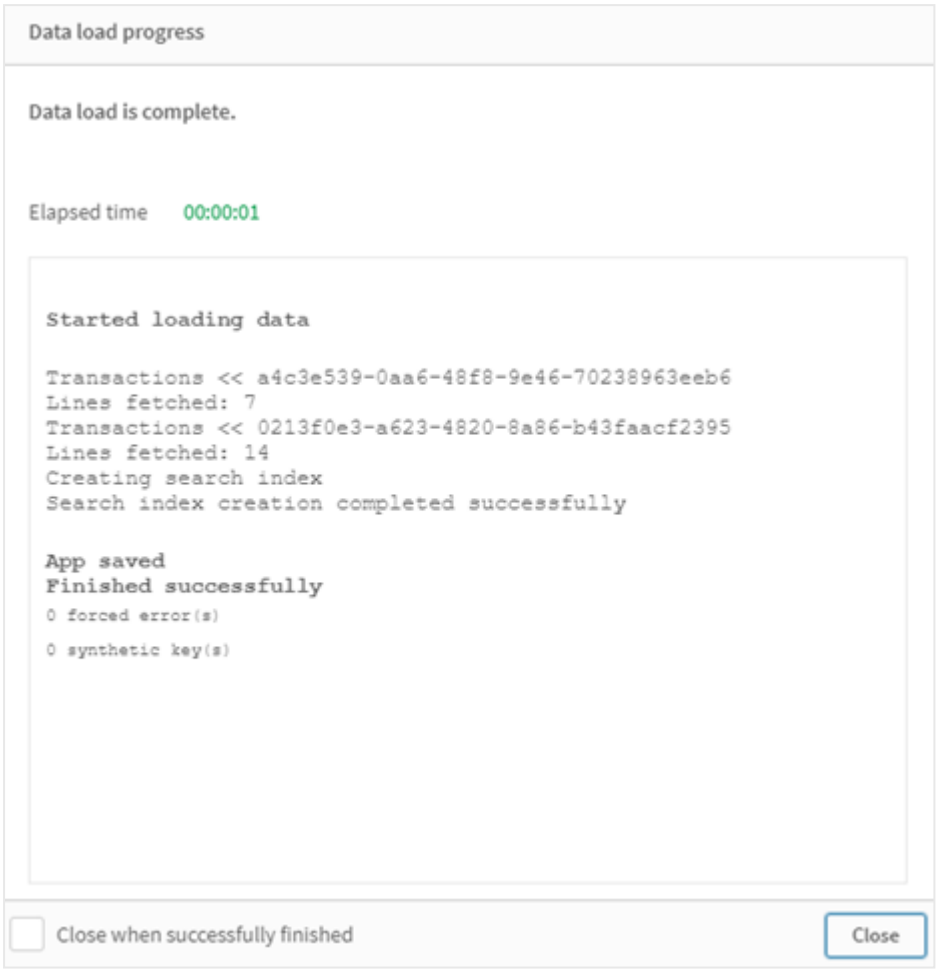
Resultattabell 2

id	datum	belopp
1	08/30/2018	23.56
2	09/07/2018	556,31
3	09/16/2018	5,75
4	2018-09-22	125,00
5	22/09/2018	484,21
6	22/09/2018	59,18
7	23/09/2018	177,42
8	10/01/2018	164.27
9	10/03/2018	384.00
10	10/06/2018	25.82
11	10/09/2018	312.00
12	10/15/2018	4.56
13	10/16/2018	90.24
14	10/18/2018	19.32

När skriptet körs konkateneras tabellen `Sales` implicit till den befintliga `Transactions`-tabellen på grund av att de två datauppsättningarna delar ett identiskt antal fält, med identiska fältnamn. Detta sker trots att den andra tabellens namntagg försöker ge den resulterande uppsättningen namnet `'Sales'`.

Du kan se att datauppsättningen över försäljningen konkateneras implicit genom att titta på loggen över **dataladdningsförloppet** .

Logg över dataladdningsförlopp som visar transaktionsdata som konkateneras implicit.



Exempel 2 – scenario med användarfall

Laddningsskript och resultat

Översikt

I det här användarfallet har du:

- Eatauppsättning med transaktioner med:
  - id
  - datum
  - belopp (i GBP)
- En valutatabell med:
  - Valutakurser för USD till GBP
- Ytterligare en datauppsättning med transaktioner med:
  - id

- datum
- belopp (i USD)

Du kommer att ladda fem skript i ordningsföljd.

- Det första laddningsskriptet innehåller en initial datauppsättning med datum och belopp i GBP som skickas till en tabell som heter `Transactions`.
- Det andra laddningsskriptet innehåller:
  - Ytterligare en datauppsättning med datum och belopp i USD som skickas till en tabell som heter `Transactions_in_USD`.
  - Prefixet `noconcatenate` som är placerat före load-satsen i datauppsättningen `Transactions_in_USD` så att implicit konkatenering förhindras.
- Det tredje laddningsskriptet innehåller prefixet `join` som kommer att användas för att skapa en valutakurs mellan GBP och USD i tabellen `Transactions_in_USD`.
- Det fjärde laddningsskriptet innehåller prefixet `concatenate` som kommer att lägga till `Transactions_in_USD` i den initiala tabellen `Transactions`.
- Det femte laddningsskriptet innehåller prefixet `drop table` som kommer att ta bort `Transactions_in_USD` i den initiala tabellen `Transactions`.

Laddningsskript 1

```
Transactions:

Load * Inline [
id, date, amount
1, 12/30/2018, 23.56
2, 12/07/2018, 556.31
3, 12/16/2018, 5.75
4, 12/22/2018, 125.00
5, 12/22/2018, 484.21
6, 12/22/2018, 59.18
7, 12/23/2018, 177.42
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `amount`

Resultat från laddningsskript 2

id	datum	belopp
1	12/30/2018	23.56
2	12/07/2018	556,31

id	datum	belopp
3	12/16/2018	5,75
4	12/22/2018	125,00
5	2018-12-22	484,21
6	2018-12-22	59,18
7	12/23/2018	177,42

Tabellen visar den initiala datauppsättningen med belopp i GBP.

Laddningsskript 2

```
Transactions_in_USD:
NoConcatenate
Load * Inline [
id, date, amount
8, 01/01/2019, 164.27
9, 01/03/2019, 384.00
10, 01/06/2019, 25.82
11, 01/09/2019, 312.00
12, 01/15/2019, 4.56
13, 01/16/2019, 90.24
14, 01/18/2019, 19.32
];
```

Resultat

Ladda data och gå till tabellen.

Resultat för det andra laddningsskriptet

id	datum	belopp
1	2018-12-30	23.56
2	2018-12-07	556,31
3	2018-12-16	5,75
4	22/12/2018	125,00
5	22/12/2018	484,21
6	22/12/2018	59,18
7	23/12/2018	177,42
8	01/01/2019	164.27
9	01/03/2019	384.00
10	01/06/2019	25.82

id	datum	belopp
11	01/09/2019	312.00
12	01/15/2019	4.56
13	01/16/2019	90.24
14	01/18/2019	19.32

Du kommer att se att den andra datauppsättningen från `Transactions_in_USD`-tabellen har lagts till.

Laddningsskript 3

Det här laddningsskriptet kopplar en valutakurs från USD till GBP till `Transactions_in_USD`-tabellen.

```
Join (Transactions_in_USD)
Load * Inline [
rate
0.7
];
```

Resultat

Ladda data och gå till datamodellvyn. Välj `Transactions_in_USD`-tabellen så kommer du att se att alla befintliga poster har ett "kurs"-värde på 0,7.

Laddningsskript 4

Genom att använda resident load kommer det här laddningsskriptet att konkatenera `Transactions_in_USD`-tabellen till `Transactions`-tabellen efter att beloppen har konverterats till USD.

```
Concatenate (Transactions)
LOAD
id,
date,
amount * rate as amount
Resident Transactions_in_USD;
```

Resultat

Ladda data och gå till tabellen. Du kommer att se nya poster med belopp i GBP från raderna åtta till fjorton.

Resultat från laddningsskript 4

id	datum	belopp
1	30/12/2018	23.56
2	2018-12-07	556,31
3	2018-12-16	5,75
4	2018-12-22	125,00

id	datum	belopp
5	2018-12-22	484,21
6	2018-12-22	59,18
7	2018-12-23	177,42
8	01/01/2019	114.989
8	01/01/2019	164.27
9	2019-01-03	268.80
9	2019-01-03	384.00
10	2019-01-06	18.074
10	2019-01-06	25.82
11	09/01/2019	218.40
11	09/01/2019	312.00
12	15/01/2019	3.192
12	15/01/2019	4.56
13	16/01/2019	63.168
13	2019-01-16	90.24
14	2019-01-18	13.524
14	2019-01-18	19.32

Laddningsskript 5

Det här laddningsskriptet kommer att släppa dubblerade poster från det fjärde laddningsskriptets resultattabell. På så sätt kommer bara poster med belopp i GBP att återstå.

```
drop tables Transactions_in_USD;
```

Resultat

Ladda data och gå till tabellen.

Resultat från laddningsskript 5

id	datum	belopp
1	2018-12-30	23.56
2	2018-12-07	556,31
3	16/12/2018	5,75
4	22/12/2018	125,00
5	22/12/2018	484,21

id	datum	belopp
6	22/12/2018	59,18
7	23/12/2018	177,42
8	01/01/2019	114,989
9	03/01/2019	268,80
10	06/01/2019	18,074
11	2019-01-09	218,40
12	2019-01-15	3,192
13	2019-01-16	63,168
14	18/01/2019	13,524

När det femte laddningsskriptet har laddats visar resultattabellen alla fjorton transaktioner som fanns i båda datauppsättningarna med transaktioner, men beloppen i transaktionerna 8-14 har konverterats till GBP.

Om vi tar bort prefixet `NoConcatenate` som användes före `Transactions_in_USD` i det andra laddningsskriptet kommer skriptet att misslyckas och felet “Tabell "`Transactions_in_USD`" visas. Detta beror på att `Transactions_in_USD`-tabellen skulle ha autokonkatenerats till den ursprungliga `Transactions`-tabellen.

Only

Skriptnyckelordet **Only** används som en aggregeringsfunktion, eller som en del av syntaxen i prefixen för partiella laddningar **Add**, **Replace** och **Merge**.

Outer

Det explicita prefixet **Join** går att fördefiniera med prefixet **Outer** för att ange en outer join. I en outer join genereras alla kombinationer mellan de två tabellerna. Den resulterande tabellen kommer således att innehålla kombinationer av fältvärden från rådatatabeller där de länkande fältvärdena visas i en eller båda tabellerna. Nyckelordet **Outer** är valfritt och är den join-typ som är standard när ett join-prefix inte anges.

Syntax:

```
Outer Join [ (tablename) ](loadstatement |selectstatement )
```

Argument:

Argument

Argument	Beskrivning
tablename	Namnet på den tabell som ska jämföras med den inlästa tabellen.
loadstatementeller selectstatement	**LOAD**- eller **SELECT**-satsen för den laddade tabellen.

Exempel

Laddningsskript

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
Table1:
Load * inline [
Column1, Column2
A, B
1, aa
2, cc
3, ee ];

Table2:
Outer Join Load * inline [
Column1, Column3
A, C
1, xx
4, yy ];
```

Resultattabell

Kolumn1	Kolumn2	Kolumn3
A	B	C
1	aa	xx
2	cc	-
3	ee	-
4	-	yy

Förklaring

I det här exemplet slås de två tabellerna Table1 och Table2 samman till en enda tabell som är märkt Table1. I fall som detta används prefixet **outer** ofta till att koppla flera tabeller till en enda tabell för att utföra aggregeringar över värdena till en enda tabell.

Delvis laddning

En fullständig laddning inleds alltid med att alla tabeller i den befintliga datamodellen tas bort, och sedan körs laddningsskriptet.

Detta sker inte under en delvis laddning. Då behålls istället alla tabeller i datamodellen och sedan exekveras bara **Load**- och **Select**-satser med ett inledande **Add**-, **Merge**- eller **Replace**-prefix. Andra datatabeller påverkas inte av kommandot. **Only**-argumentet anger att satsen bara ska exekveras vid delvisa laddningar och ignoreras vid fullständiga laddningar. Följande tabell sammanfattar programutförandet för partiella och fullständiga ominläsningar.

Sats	Fullständig omladdning	Delvis laddning
Load ...	Sats körs	Sats körs inte
Lägg till/ersätt/sammanfoga laddning ...	Sats körs	Sats körs
Lägg till/ersätt/sammanfoga endast laddning ...	Sats körs inte	Sats körs

Delvisa laddningar har flera fördelar jämfört med fullständiga laddningar:

- Det går fortare eftersom det bara är data som ändrats nyligen som måste laddas. Skillnaden är avsevärd för stora datauppsättningar.
- Det går åt mindre minne eftersom färre data laddas.
- Mer tillförlitligt eftersom frågor till källdata körs snabbare, vilket minskar risken för nätverksrelaterade problem.

För att delvis laddning ska fungera som avsett måste applikationen öppnas med data innan en delvis laddning utlöses.

Gör en delvis laddning med knappen **Ladda**. Du kan även använda Qlik Engine JSON API.

Begränsningar

En delvis laddning misslyckas om det finns kommandon med referenser till tabeller som fanns före den fullständiga laddningen, men inte under den delvisa laddningen.

Exempel

Kommandoexempel

```
LEFT JOIN(<Table_removed_after_full_reload>)
CONCATENATE(<Table_removed_after_full_reload>)
```

Där <Table_removed_after_full_reload> är en tabell som fanns före den fullständiga laddningen, men inte i den delvisa laddningen.

Kringgå detta

För att kringgå detta kan du omge kommandot med följande if-kommando:

```
IF NOT IsPartialReload() THEN ... ENDIF.
```

En delvis laddning kan ta bort värden ur datan. Det här kommer dock inte att reflekteras i listan över distinkta värden, som är en tabell som underhålls internt. Så efter en delvis laddning kommer listan att innehålla alla distinkta värden som har funnits i fältet sedan den senaste fullständiga laddningen, vilket kan vara mer än vad som för närvarande existerar efter en delvis laddning. Det här påverkar utdatan från funktionerna FieldValueCount() och FieldValue(). FieldValueCount() kan eventuellt returnera ett större antal än det nuvarande antalet fältvärden.

Exempel

Exempel 1

Laddningsskript

Lägg till exempelskriptet i din applikation och gör en delvis omladdning. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
T1:
Add only Load distinct recno()+10 as Num autogenerate 10;
```

Resultat

Resulting table

Num	Count(Num)
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	1

Förklaring

Satsen exekveras endast under en delvis omladdning. Om prefixet "distinct" utesluts ökar antalet **Num**-fält med varje konsekutiv delvis omladdning.

Exempel 2

Laddningsskript

Lägg till exempelskriptet i din applikation. Gör en fullständig omladdning och se resultatet. Gör sedan en delvis omladdning och se resultatet. När du vill se resultaten, lägger du till de fält som anges i resultatkolumnen i ett ark i din applikation.

```
T1:
Load recno() as ID, recno() as Value autogenerate 10;

T1:
Replace only Load recno() as ID, repeat(recno(),3) as Value autogenerate 10;
```

Resultat

Output table after full reload

ID	Value
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

Output table after partial reload

ID	Value
1	111
2	222
3	333
4	444
5	555
6	666
7	777
8	888
9	999
10	101010

Förklaring

Den första tabellen laddas under en fullständig omladdning och den andra tabellen ersätter helt enkelt den första tabellen under en delvis omladdning.

Replace

Skriptnyckelordet **Replace** används som en strängfunktion, eller som ett prefix i en partiell laddning.

Replace

Prefixet **Replace** kan adderas till valfri **LOAD**- eller **SELECT**-sats i skriptet för att specificera att den laddade tabellen ska ersätta en annan tabell. Det specificerar även att satsen ska köras i en delvis laddning. **Replace**-prefixet kan även användas i en **Map**-sats.

> *För att delvis laddning ska fungera som avsett måste applikationen öppnas med data innan en delvis laddning utlöses.*

Gör en delvis laddning med knappen **Ladda**. Du kan även använda Qlik Engine JSON API.

Syntax:

```
Replace [only] [Concatenate[(tablename)]] (loadstatement | selectstatement)

Replace [only] mapstatement
```

Under en vanlig laddning (ej delvis) fungerar konstruktionen **Replace LOAD** som en vanlig **LOAD**-sats, men den föregås av **Drop Table**. Först utgår den gamla tabellen, och sedan genereras poster som lagras som en ny tabell.

Om **Concatenate**-prefixet används, eller om det finns en tabell med samma uppsättning fält, blir detta den tabell som utgår. Annars finns det ingen tabell som ska utgå och konstruktionen **Replace LOAD** är likadan som en vanlig **LOAD**-sats.

En delvis laddning gör samma sak. Den enda skillnaden är att det alltid finns en tabell från den tidigare skriptexekveringen att låta utgå. Med konstruktionen **Replace LOAD** utgår alltid den gamla tabellen först, och sedan skapas en ny.

Om **Replace Map...Using**-satsen används utförs mappningen även vid delvis skriptexekvering.

Argument:

Argument

Argument	Beskrivning
only	En valfri kvalificerare som innebär att satsen bara ska exekveras vid delvisa laddningar. Den ska ignoreras vid vanliga (ej delvisa) laddningar.

Exempel och resultat:

Exempel	Resultat

`Tab1:`
`Replace LOAD *`
`from File1.csv;`
	Under både normal och partiell laddning utgår Qlik Sense-tabellen Tab1 inledningsvis. Sedan läses nya data in från File1.csv och lagras i Tab1.

Exempel	Resultat

`Tab1:`
`Replace only`
`LOAD * from`
`File1.csv;`
	Vid vanlig laddning beaktas inte denna sats.
	
	Vid partiell laddning utgår alla Qlik Sense-tabeller som tidigare hette Tab1 inledningsvis. Sedan läses nya data in från File1.csv och lagras i Tab1.

`Tab1:`
`LOAD a,b,c from`
`File1.csv;`
`Replace LOAD`
`a,b,c from`
`File2.csv;`
	Vid vanlig laddning läses filen File1.csv först in i Qlik Sense-tabellen Tab1, men utgår sedan omedelbart och ersätts av nya data från File2.csv. Alla data från File1.csv går förlorade.
	
	Under delvis laddning utgår hela Qlik Sense-tabellen Tab1 inledningsvis. Den ersätts därefter av nya data från File2.csv.

`Tab1:`
`LOAD a,b,c from`
`File1.csv;`
`Replace only`
`LOAD a,b,c from`
`File2.csv;`
	Vid normal laddning läses data från File1.csv och lagras i Qlik Sense-tabellen Tab1. File2.csv beaktas ej.
	
	Under delvis laddning utgår hela Qlik Sense-tabellen Tab1 inledningsvis. Den ersätts därefter av nya data från File2.csv. Alla data från File1.csv går förlorade.

Right

Prefixen **Join** och **Keep** kan inledas med prefixet **right**.

Om det används före **join** anger det att right join ska användas. Den resulterande tabellen kommer endast att innehålla kombinationer av fältvärden från rådatatabeller där de länkande fältvärdena visas i den andra tabellen. Om det används framför **keep**, anger det att den första rådatatabellen ska reduceras till det gemensamma snittet med den andra tabellen innan den lagras i Qlik Sense.

Letade du efter strängfunktionen med samma namn? Se: Right (page 1832)

Syntax:

```
Right (Join | Keep) [(tablename)](loadstatement |selectstatement )
```

Argument:

Argument

Argument	Beskrivning
tablename	Namnet på den tabell som ska jämföras med den inlästa tabellen.
loadstatementeller selectstatement	**LOAD**- eller **SELECT**-satsen för den laddade tabellen.

Exempel

Laddningsskript

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
Table1:
Load * inline [
Column1, Column2
A, B
1, aa
2, cc
3, ee ];

Table2:
Right Join Load * inline [
Column1, Column3
A, C
1, xx
4, yy ];
```

Resultat

Resultattabell

Kolumn1	Kolumn2	Kolumn3
A	B	C
1	aa	xx
4	-	yy

Förklaring

Det här exemplet demonstrerar Right Join-utdata där endast värden i den andra (högra) tabellen kopplas.

Sample

Prefixet **sample** till en **LOAD**- eller **SELECT**-sats används för att ladda ett slumpmässigt urval av poster från datakällan.

Syntax:

```
Sample  p ( loadstatement | selectstatement )
```

Uttrycket som utvärderas definierar inte procentandelen poster i datauppsättningen som kommer att laddas i Qlik Sense-programmet, utan sannolikheten för att varje post som läses in laddas i programmet. Att specificera ett värde `p = 0.5` innebär med andra ord inte att 50 % av det totala antalet poster kommer att läsas in, utan att det finns 50 % chans att den laddas till Qlik Sense-programmet.

Argument

Argument	Beskrivning
p	Ett godtyckligt uttryck som resulterar i ett tal större än 0 och mindre än eller lika med 1. Talet anger sannolikheten för att en viss post ska läsas. Alla poster läses, men bara vissa av dem läses in i Qlik Sense.

Användning

Stickprov är användbart när du vill ta stickprov på data som kommer från en stor tabell, för att förstå egenskaperna hos data, fördelningen eller fältinnehållet. Eftersom det hämtar en delmängd av data laddas data snabbare så att skript kan testas snabbare. Till skillnad från `First` hämtar `Sample`-funktionen data från hela tabellen, till skillnad från att begränsa den till de första raderna. Detta kan ge en mer rättvisande representation av data i vissa fall.

I följande exempel visas möjliga användningar av skriptprefixet `Sample`:

```
Sample 0.15 SQL SELECT * from Longtable;

Sample(0.15) LOAD * from Longtab.csv;
```

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – stickprov från en inline-tabell

Laddningsskript och resultat

Översikt

I det här exemplet laddar skriptet en sticksprovsuppsättning med data från en datauppsättning som innehåller sju poster till en tabell som heter `Transactions` från en inline-tabell.

Laddningsskript

```
Transactions:
SAMPLE 0.3
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
```

```
7, 09/23/2018, 177.42
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `amount`

Lägg till följande mått:

```
=sum(amount)8
```

Resultattabell

id	datum	=Sum(amount)
2	09/07/2018	556.31
4	09/22/2018	125
1	08/30/2018	23.56
3	09/16/2018	5.75

I itereringen av laddningen som används i det här exemplet lästes alla sju posterna, men bara fyra poster laddades till datatabellen. Om körningen upprepas kan det leda till en annan siffra, och att ett annat antal poster laddas till programmet.

Exempel 2 – stickprov från en autogenererad tabell

Laddningsskript och resultat

Översikt

I det här exemplet har `Autogenerate` använts för att skapa en datauppsättning med 100 poster med fälten `date`, `id` och `amount`. Men prefixet `Sample` används, med ett värde på 0,1.

Laddningsskript

```
SampleData:
Sample 0.1
LOAD
RecNo() AS id,
MakeDate(2013, Ceil(Rand() * 12), Ceil(Rand() * 29)) as date,
Rand() * 1000 AS amount

Autogenerate(100);
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- id
- amount

Lägg till följande mått:

Resultattabell

id	datum	=Sum(amount)
48	9/28/2013	763
20	5/15/2013	752
19	11/8/2013	657
25	3/24/2013	522
27	8/23/2013	389
81	6/1/2013	53
100	8/15/2013	17

I itereringen av laddningen som används i det här exemplet laddades sju poster från den skapade datauppsättningen. Om körningen upprepas kan det som sagt leda till en annan siffra, och att ett annat antal poster laddas till programmet.

Semantic

`semantic`-laddningsprefixet skapar en speciell typ av fält som kan användas i Qlik Sense för att ansluta och hantera relationsdata, såsom trädstrukturer, självrefererande strukturerade överordnade och underordnade data och/eller data som kan beskrivas som ett diagram.

Observera att `semantic`-laddningen kan fungera på samma sätt som *Hierarchy (page 64)*- och *HierarchyBelongsTo (page 66)*-prefixen. Alla tre prefixen kan användas som byggstenar i effektiva front-end-lösningar för att korsa relationsdata.

Syntax:

```
Semantic( loadstatement | selectstatement)
```

En semantisk belastning förväntar sig indata som är exakt tre eller fyra fält bred med en strikt definition av vad varje ordnat fält representerar, som visas i tabellen nedan:

Semantiska laddningsfält

Fältnamn	Fältbeskrivning
Första fältet:	Denna tagg är en representation av det första av två objekt mellan vilka det finns en relation.
Andra fältet:	Denna tagg kommer att användas för att beskriva ett framåtförhållande mellan det första och andra objektet. Om det första objektet är underordnad och det andra är överordnat kan du skapa en relationsflik som anger "överordnad" eller "överordnad till" som om du följde en relation mellan barn och förälder.

Fältnamn	Fältbeskrivning
Tredje fältet:	Denna tagg är en representation av det första av två objekt mellan vilka det finns en relation.
Fjärde fältet:	Detta fält är valfritt. Denna tagg kommer beskriver en bakåtriktad eller inverterad relation mellan det första och andra objektet. Om det första objektet är underordnad och det andra är överordnat kan du skapa en relationsflik som anger "underordnad" eller "underordnad till" som om du följde en relation mellan barn och förälder. Om du inte lägger till ett fjärde fält kommer den andra fälttaggen att användas för att beskriva förhållandet i endera riktningen. I så fall läggs en pilsymbol automatiskt till som en del av taggen.

Följande kod är ett exempel på `semantic`-prefixet.

```
Semantic
Load
Object,
‘Parent’ AS Relationship,
NeighbouringObject AS Object,
‘Child’ AS Relationship
from graphdata.csv;
```

Det är tillåtet och vanlig praxis att märka det tredje fältet på samma sätt som det första fältet. Detta skapar en självrefererande sökmetod, så att du kan följa objektet/objekten till det/de relaterade objekten ett relationssteg i taget. Om det tredje fältet inte bär samma namn, kommer slutresultatet att bli en enkel sökning från ett eller flera objekt till dess direkta relationsgrannar ett steg bort, vilket är ett resultat som har mycket liten praktisk användning.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Relaterade funktioner

Funktioner	Interaktion
Hierarchy (page 64)	Hierarkiladdningsprefixet används för att dela och organisera noder i överordnade-underordnade och andra grafliknande datastrukturer och omvandla dem till tabeller.

Funktioner	Interaktion
HierarchyBelongsTo (page 66)	HierarchyBelongsTo-prefixet används för att hitta och organisera föregångare till överordnade-underordnade och andra grafliknande datastrukturer och omvandla dem till tabeller.

Exempel – Skapa ett specialfält för att koppla relationer med det semantiska prefixet

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som representerar geografirelationsposter, som läses in i en tabell med namnet `GeographyTree`.
    - Varje post har ett ID i början av raden och ett ParentID i slutet av raden.
- `semantic`-prefixet som kommer att lägga till ett speciellt beteendefält märkt, `Relation`.

Laddningsskript

```
GeographyTree:
LOAD
    ID,
    Geography,
    if(ParentID='',null(),ParentID) AS ParentID

INLINE [
ID,Geography,ParentID
1,World
2,Europe,1
3,Asia,1
4,North America,1
5,South America,1
6,UK,2
7,Germany,2
8,Sweden,2
9,South Korea,3
10,North Korea,3
11,China,3
12,London,6
13,Birmingham,6
];

SemanticTable:
Semantic Load
       ID as ID,
       'Parent' as Relation,
       ParentID as ID,
```

```
      'Child' as Relation
resident GeographyTree;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner.

- `Id`
- `Geography`

Skapa sedan en filterruta med `Relation` som dimension. Klicka på **Redigering klar**.

Resultattabell

Id	Geografi
1	Värld
2	Europa
3	Asien
4	Nordamerika
5	Sydafrika
6	Storbritannien
7	Tyskland
8	Sverige
9	Sydkorea
10	Nordkorea
11	Kina
12	London
13	Birmingham

Filterruta

Relationer
Underordnad
Överordnad

Klicka på **Europa** från `Geography`-dimensionen i tabellen och klicka på **Barn** från `Relation`-dimensionen i filterrutan. Notera det förväntade resultatet i tabellen:

Resultattabell som visar underordnade i Europa

Id	Geografi
6	Storbritannien
7	Tyskland
8	Sverige

Om du klickar på **Underordnad** igen visas platser som är "Underordnade" i Storbritannien, ett steg längre ner.

Resultattabell som visar underordnade i Storbritannien

Id	Geografi
12	London
13	Birmingham

Unless

Prefixet eller suffixet **unless** används för att skapa en villkorssats, som bestämmer om en sats eller ett exit-tillägg ska utvärderas eller ej. Det kan ses som ett kompakt alternativ till en fullständig **if..end if**-sats.

Syntax:

```
(Unless condition statement | exitstatement Unless condition )
```

statement eller **exitstatement** exekveras bara om **condition** utvärderas till False.

Prefixet **unless** får användas även om satsen redan har en eller flera andra satser, inklusive ytterligare **when**- eller **unless**-prefix.

Argument

Argument	Beskrivning
condition	Ett logiskt uttryck som utvärderas till True eller False.
statement	Alla Qlik Sense-skriptsatser utom kontrollsatser.
exitstatement	Ett **exit for**-, **exit do**- eller **exit sub**-tillägg eller en **exit script**-sats.

Användning

Satsen `Unless` returnerar ett booleskt resultat. Den här typen av funktion används typiskt som ett villkor när användaren vill ladda eller utesluta delar av skriptet beroende på vissa villkor.

I följande rader visas tre exempel på hur `Unless`-funktionen kan användas:

```
exit script unless A=1;
```

```
unless A=1 LOAD * from myfile.csv;

unless A=1 when B=2 drop table Tab1;
```

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – unless-prefix

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En variabel A skapas, som får värdet 1.
- En datauppsättning som laddas till en tabell som heter transaktioner, såvida inte variabel A = 2.

Laddningsskript

```
LET A = 1;

UNLESS A = 2

Transactions:
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `amount`

Resultattabell

id	datum	belopp
1	08/30/2018	23.56
2	09/07/2018	556,31
3	09/16/2018	5,75
4	2018-09-22	125,00
5	22/09/2018	484,21
6	22/09/2018	59,18
7	23/09/2018	177,42

Eftersom variabel A tilldelas värdet 1 när skriptet startar kommer villkoret som följer efter prefix `Unless` att utvärderas, vilket returnerar resultatet`FALSE`. Som resultat fortsätter skriptet att köra `Load`-satsen. I den här resultattabellen visas alla poster från `Transactions`-tabellen.

Om den här variabelns värde har satts till 2 kommer inga data att laddas till datamodellen.

Exempel 2 – unless-prefix

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet startas genom att en initial datauppsättning laddas till en tabell som heter `Transactions`. Skriptet avslutas därefter såvida det inte finns färre än 10 poster i `Transactions`-tabellen.

Om det här villkoret inte resulterar till att skriptet avslutas konkateneras ytterligare en uppsättning transaktioner till `Transactions`-tabellen och den här processen upprepas.

Laddningsskript

```
Transactions:
LOAD
*
Inline [
id, date, amount
```

```
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];


exit script unless NoOfRows('Transactions') < 10 ;

Concatenate
LOAD
*
Inline [
id, date, amount
8, 10/01/2018, 164.27
9, 10/03/2018, 384.00
10, 10/06/2018, 25.82
11, 10/09/2018, 312.00
12, 10/15/2018, 4.56
13, 10/16/2018, 90.24
14, 10/18/2018, 19.32
];

exit script unless NoOfRows('Transactions') < 10 ;

Concatenate
LOAD
*
Inline [
id, date, amount
15, 10/01/2018, 164.27
16, 10/03/2018, 384.00
17, 10/06/2018, 25.82
18, 10/09/2018, 312.00
19, 10/15/2018, 4.56
20, 10/16/2018, 90.24
21, 10/18/2018, 19.32
];

exit script unless NoOfRows('Transactions') < 10 ;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- id
- date
- amount

Resultattabell

id	datum	belopp
1	08/30/2018	23.56
2	09/07/2018	556,31
3	09/16/2018	5,75
4	2018-09-22	125,00
5	22/09/2018	484,21
6	22/09/2018	59,18
7	23/09/2018	177,42
8	10/01/2018	164.27
9	10/03/2018	384.00
10	10/06/2018	25.82
11	10/09/2018	312.00
12	10/15/2018	4.56
13	10/16/2018	90.24
14	10/18/2018	19.32

Det finns sju poster i vart och ett av de tre datauppsättningarna i laddningsskriptet.

Den första datauppsättningen (med `id`-transaktion 1-7) laddas till applikationen. `Unless`-villkoret utvärderar om det finns färre än 10 rader i `Transactions`-tabellen. Detta utvärderas till `TRUE` och därför laddas den andra uppsättningen (med `id`-transaktion 8-14) laddas till programmet. Det andra `Unless`-villkoret utvärderar om det finns färre än 10 poster i `Transactions`-tabellen. Det här utvärderas till `FALSE`. Skriptet avslutas.

Exempel 3 – flera unless-prefix

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

I det här exemplet skapas en datauppsättning som innehåller en transaktion som en tabell som heter `Transactions`. Därefter utlöses en "for"-slinga, där två nästlade unless-satser utvärderas:

1. Såvida inte det finns mer än 100 poster i `Transactions`-tabellen
2. Såvida inte antalet poster i `Transactions`-tabellen är en multipel av 6

Om de här villkoren är FALSE genereras ytterligare sju poster som konkateneras till den befintliga Transactions-tabellen. Den här processen upprepas tills en eller två transaktioner returnerar ett värde på TRUE.

Laddningsskript

```
Transactions:
Load
        0 as id
Autogenerate 1;

For i = 1 to 100
        unless NoOfRows('Transactions') > 100 unless mod(NoOfRows('Transactions'),6) = 0
                Concatenate
      Load
if(isnull(Peek(id)),1,peek(id)+1) as id
               Autogenerate 7;
next i
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:id.

Resultattabell

id
0
1
2
3
4
5
ytterligare 30+ rader

De nästlade unless-satserna som finns i "for"-slingan utvärderas till följande:

1. Finns det fler än 100 rader i Transactions-tabellen?
2. Är det totala antalet poster i Transactions-tabellen en multipel av 6?

När båda unless-satserna returnerar värdet FALSE genereras ytterligare sju poster som konkateneras till den befintliga Transactions-tabellen.

De här satserna returnerar ett värde på FALSE fem gånger. Det finns då totalt 36 rader med data i Transactions-tabellen.

Därefter returnerar den andra Unless-satsen värdet TRUE, och därför kommer den efterföljande load-satsen inte längre att exekveras.

When

Prefixet eller suffixet **when** används för att skapa en villkorssats, som bestämmer om en sats eller ett exit-tillägg ska exekveras eller ej. Det kan ses som ett kompakt alternativ till en fullständig **if..end if**-sats.

Syntax:

```
(when condition statement | exitstatement when condition )
```

Returnerad datatyp: Booleansk

I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.

statement eller **exitstatement** exekveras bara om villkoret utvärderas till TRUE.

Prefixet `When` får användas även om satsen redan har en eller flera andra satser, inklusive ytterligare `When`- eller `Unless`-prefix.

Användning

Satsen `When` returnerar ett booleskt resultat. Den här typen av funktion används typiskt som ett villkor när användaren vill ladda eller utesluta delar av skriptet.

Argument

Argument	Beskrivning
condition	Ett logiskt uttryck som utvärderas till TRUE eller FALSE
statement	Alla Qlik Sense-skriptsatser utom kontrollsatser.
exitstatement	Ett **exit for**-, **exit do**- eller **exit sub**-tillägg eller en **exit script**-sats.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`exit script when A=1;`	När satsen `A=1` utvärderas till TRUE stannar skriptet.
`when A=1 LOAD * from myfile.csv;`	När satsen `A=1` utvärderas till TRUE körs `myfile.csv`.
`when A=1 unless B=2 drop table Tab1;`	När satsen `A=1` utvärderas till TRUE och om `B=2` utvärderas till FALSE kommer `Tab1`-tabellen att släppas.

Exempel 1 – when-prefix

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum och belopp som skickas till en tabell som heter `Transactions`.
- `Let`-satsen som anger att variabeln `A` skapas och har värdet `1`.
- `When`-villkoret som tillhandahåller villkoret att om `A` är lika med `1` kommer skriptet att fortsätta att laddas.

Laddningsskript

```
LET A = 1;

WHEN A = 1

Transactions:
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- id
- date
- amount

Resultattabell

id	datum	belopp
1	08/30/2018	23.56
2	09/07/2018	556,31
3	09/16/2018	5,75
4	2018-09-22	125,00
5	22/09/2018	484,21
6	22/09/2018	59,18
7	23/09/2018	177,42

Eftersom variabel A tilldelas värdet 1 när skriptet startar kommer villkoret som följer efter prefix When utvärderas, vilket returnerar resultatet TRUE. Eftersom den returnerar resultatet TRUE kommer skriptet att fortsätta att köra laddningssatsen. Alla poster från resultattabellen visas.

Om den här variabelns värde har satts till något värde som inte är 1 kommer inga data att laddas till datamodellen.

Exempel 2 – when-prefix

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Tre datauppsättningar med datum och belopp som skickas till en tabell som heter Transactions.
  - Den första datauppsättningen innehåller transaktionerna 1-7.
  - Den första datauppsättningen innehåller transaktionerna 8-14.
  - Den första datauppsättningen innehåller transaktionerna 15-21.
- Ett When-villkor som avgör om Transactions-tabellen innehåller fler än tio rader. Om någon av When-satserna utvärderas till TRUE stannar laddningsskriptet. Villkoret placeras på slutet av var och en av de tre datauppsättningarna.

Laddningsskript

```
Transactions:
LOAD
*
Inline [
```

```
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];

exit script when NoOfRows('Transactions') > 10 ;

Concatenate
LOAD
*
Inline [
id, date, amount
8, 10/01/2018, 164.27
9, 10/03/2018, 384.00
10, 10/06/2018, 25.82
11, 10/09/2018, 312.00
12, 10/15/2018, 4.56
13, 10/16/2018, 90.24
14, 10/18/2018, 19.32
];

exit script when NoOfRows('Transactions') > 10 ;

Concatenate
LOAD
*
Inline [
id, date, amount
15, 10/01/2018, 164.27
16, 10/03/2018, 384.00
17, 10/06/2018, 25.82
18, 10/09/2018, 312.00
19, 10/15/2018, 4.56
20, 10/16/2018, 90.24
21, 10/18/2018, 19.32
];

exit script when NoOfRows('Transactions') > 10 ;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- id
- date
- amount

Resultattabell

id	datum	belopp
1	08/30/2018	23.56
2	09/07/2018	556,31
3	09/16/2018	5,75
4	2018-09-22	125,00
5	22/09/2018	484,21
6	22/09/2018	59,18
7	23/09/2018	177,42
8	10/01/2018	164.27
9	10/03/2018	384.00
10	10/06/2018	25.82
11	10/09/2018	312.00
12	10/15/2018	4.56
13	10/16/2018	90.24
14	10/18/2018	19.32

Det finns sju transaktioner i var och en av de tre datauppsättningarna. Den första datauppsättningen innehåller transaktion 1-7 och laddas till applikationen. `When`-villkoret som följer efter denna laddningssats utvärderas som FALSE eftersom det finns färre än tio rader i `Transactions`-tabellen. Laddningsskriptet fortsätter till nästa datauppsättning.

Den andra datauppsättningen innehåller transaktion 8-14 och laddas till applikationen. Det andra `When`-villkoret utvärderas som TRUE eftersom det finns fler än tio rader i `Transactions`-tabellen. Därför avslutas skriptet.

Exempel 3 – flera when-prefix

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en enda transaktion skaps som en tabell som heter `Transactions`.
- En `For`-slinga som utlöses innehåller två nästlade `When`-villkor som utvärderar huruvida:

1. Det finns färre än `100` poster i `Transactions`-tabellen.
2. Antalet poster i `Transactions`-tabellen är inte en multipel av `6`.

Laddningsskript

```
RowsCheck = NoOfRows('Transactions') < 100 or mod(NoOfRows('Transactions'),6) <> 0;
Transactions:
Load
       0 as id
Autogenerate 1;
For i = 1 to 100
       when(RowsCheck)
              Concatenate
       Load
              if(isnull(Peek(id)),1,peek(id)+1) as id
       Autogenerate 7;
next i
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `id`

I resultattabellen visas bara de fem första transaktions-ID, men laddningsskriptet skapar 36 rader och avslutas sedan när `When`-villkoret är uppfyllt.

Resultattabell

id
0
1
2
3
4
5
ytterligare 30+ rader

De nästlade `When`-villkoren i `For`-loopen utvärderar följande frågor:

- Finns det färre än 100 rader i `Transactions`-tabellen?
- Är det totala antalet poster i `Transactions`-tabellen inte en multipel av sex?

När båda `When`-villkoren returnerar värdet TRUE genereras ytterligare sju poster som konkateneras till den befintliga `Transactions`-tabellen.

`When`-villkoren returnerar ett TRUE-värde fem gånger. Det finns nu 36 rader med data i `Transactions`-tabellen.

När 36 rader med data har skapats i `Transactions`-tabellen returnerar den andra `When`-satsen värdet FALSE, och därför kommer den efterföljande load-satsen inte längre att exekveras.

3.3 Vanliga skriptsatser

Vanliga satser används normalt för att modifiera data på ett eller annat sätt. Dessa satser kan skrivas på ett valfritt antal rader i skriptet och måste alltid avslutas med ett semikolon: ";".

Alla nyckelord i skriptet kan skrivas med antingen versaler eller gemener. Fält- och variabelnamn är dock skiftlägeskänsliga.

Översikt av vanliga skriptsatser

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Alias

Satsen **alias** används för att ställa in ett alias enligt vilket ett fält ska döpas om när det förekommer i skriptet som följer.

```
Alias fieldname as aliasname {,fieldname as aliasname}
```

Autonummer

Denna sats returnerar ett unikt heltal för varje distinkt utvärderat värde i ett fält som påträffas under skriptexekveringen.

```
AutoNumber fields [Using namespace] ]
```

Binary

binary-satsen används för att ladda data från ett annat QlikView-dokument, inklusive Section Access-data.

```
Binary [path] filename
```

kommentar

Gör det möjligt att visa fältkommentarerna (metadata) från databaser eller kalkylblad. Fältnamn som inte finns i applikationen ignoreras. Om flera förekomster av ett fältnamn hittas används det senaste värdet.

```
Comment field *fieldlist using mapname
Comment field fieldname with comment
```

kommentarstabell

Gör det möjligt att visa tabellkommentarerna (metadata) från databaser eller kalkylblad.

```
Comment table tablelist using mapname
Comment table tablename with comment
```

Anslut

Den här funktionen är inte tillgänglig i Qlik Sense SaaS.

CONNECT-satsen används för att ange Qlik Sense-åtkomst till en allmän databas via OLE DB/ODBC-gränssnittet. För ODBC måste datakällan först anges med hjälp av ODBC-administratören.

```
ODBC Connect TO connect-string [ ( access_info ) ]
OLEDB CONNECT TO connect-string [ ( access_info ) ]
CUSTOM CONNECT TO connect-string [ ( access_info ) ]
LIB CONNECT TO connection
```

Begränsa

Satsen **Constrain** kan användas i kombination med satserna **Let** eller **Set** för att definiera variabler i skriptet. Med hjälp av **Constrain**-satsen kan du definiera begränsningar för möjliga värden för dessa variabler. Om en definition av en variabel bryter mot begränsningarna misslyckas inläsningen. Med begränsningar kan du kräva att värden för variabler ska matcha specifika typer, ligga inom specifika numeriska intervall och matcha acceptabla värden som du definierar.

```
Constrain variablename = json
```

Deklarera

Declare-satsen används för att skapa fältdefinitioner där du kan definiera relationer mellan fält eller funktioner. En uppsättning fältdefinitioner kan användas för att automatiskt generera härledda fält, vilka kan användas som dimensioner. Du kan till exempel skapa en kalenderdefinition och använda den för att generera relaterade dimensioner, som år, månad, vecka och dag, från ett datumfält.

```
definition_name:
Declare [Field[s]] Definition [Tagged tag_list ]
[Parameters parameter_list ]
Fields field_list
[Groups group_list ]
<definition name>:
Declare [Field][s] Definition
Using <existing_definition>
[With <parameter_assignment> ]
```

Härled

Derive-satsen används för att generera härledda fält baserat på en fältdefinition som skapats med en **Declare**-sats. Du kan antingen specifikt ange vilka datafält du vill härleda fält från, eller härleda dem explicit eller implicit baserat på fälttaggar.

```
Derive [Field[s]] From [Field[s]] field_list Using definition
Derive [Field[s]] From Explicit [Tag[s]] (tag_list) Using definition
Derive [Field[s]] From Implicit [Tag[s]] Using definition
```

Direct Query

Med **DIRECT QUERY**-satsen kan du komma åt tabeller genom enODBC eller OLE DB-koppling med hjälp av funktionen Direkt upptäckt.

```
Direct Query [path]
```

Katalog

Directory-satsen anger i vilken katalog datafilerna ska sökas i följande **LOAD**-satser tills en ny **Directory**-sats anges.

```
Directory [path]
```

Koppla från

Disconnect-satsen avslutar den aktuella ODBC-kopplingen/OLE DB-kopplingen/anpassade kopplingen. Denna sats är valfri.

```
Disconnect
```

släpp fält

Ett eller flera Qlik Sense-fält kan avlägsnas från datamodellen, och därigenom från minnet, när som helst under skriptexekveringen med hjälp av en **drop field**-sats. Egenskapen "distinct" för en tabell tas bort efter en **drop field**-sats.

> *Både **drop field** och **drop fields** är tillåtna och har samma betydelse. Om ingen tabell har angetts avlägsnas fältet från alla tabeller där det förekommer.*

```
Drop field  fieldname [ , fieldname2 ...] [from tablename1 [ , tablename2
...]]
drop fields fieldname [ , fieldname2 ...] [from tablename1 [ , tablename2
...]]
```

ta bort tabell

En eller flera interna Qlik Sense-tabeller kan tas bort från datamodellen och därmed också från minnet, när som helst vid skriptkörning med hjälp av en **drop table**-sats. Annars kan du använda **drop mapping table**-varianten för att släppa mappningstabeller – de som skapats med **Mapping**-prefixet – före slutet av skriptkörningen.

> *Formerrna **drop table** och **drop tables** accepteras båda.*

```
Drop [mapping] table tablename [, tablename2 ...]
drop [mapping] tables[ tablename [, tablename2 ...]
```

Ta bort variabel

Använd **Drop variable**-satsen för att släppa en skriptvariabel under skriptkörning. När en variabel släpps är den inte tillgänglig eller synlig, till exempel i arkvyn.

*Både **drop variable** och **drop variables** är tillåtna och har samma betydelse.*

```
Drop variable variablename [ , variablename2 ...]
Drop variables variablename [ , variablename2 ...]
```

Kör

Execute-satsen använda för att köra andra program medan Qlik Sense laddar data, exempelvis för att göra de konverteringar som krävs.

```
Execute commandline
```

Töm logg

Satsen **FlushLog** tvingar Qlik Sense att skriva innehållet i skriptbufferten till skriptets loggfil.

```
FlushLog
```

Tvinga

force-satsen tvingar Qlik Sense att tolka fältvärden i efterföljande **LOAD**- och **SELECT**-satser som att de är skrivna med enbart versaler, enbart gemener, alltid med inledande versaler eller som de uppträder (blandat). Med hjälp av denna sats är det möjligt att koppla fält från tabeller med olika konventioner.

```
Force ( capitalization | case upper | case lower | case mixed )
```

LADDA

LOAD-satsen laddar fält från en fil, direkt från data i skriptet, från en tidigare inläst tabell, från en webbsida, från resultatet av en efterföljande **SELECT**-sats eller genom att generera data automatiskt. Det går även att ladda data från analyskopplingar.

```
 Load [ distinct ] *fieldlist
[( from file [ format-spec ] |
from_field fieldassource [format-spec]
inline data [ format-spec ] |
resident table-label |
autogenerate size )]
[ where criterion | while criterion ]
[ group_by groupbyfieldlist ]
[order_by orderbyfieldlist ]
[extension pluginname.functionname(tabledescription)]
```

Låt

let-satsen är ett komplement till **set**-satsen som används för att definiera skriptvariabler. I motsats till **set**-satsen utvärderas **let**-satsens uttryck till höger om '=' när skriptet körs, innan det tilldelas variabeln.

```
Let variablename=expression
```

Lossa tabell

En eller flera av Qlik Senses interna datatabeller kan explicit förklaras vara löst kopplade under skriptexekvering med hjälp av satsen **Loosen Table**. När en tabell är löst kopplad tas alla associationer mellan fältvärdena i tabellen bort. Man kan åstadkomma en liknande effekt genom att ladda vart och ett av fälten i den löst kopplade tabellen som fristående, okopplade tabeller. Löst kopplad kan vara användbart under tester för att tillfälligt isolera olika delar av datastrukturen. En löst kopplad tabellen kan identifieras i tabellvyn genom de streckade linjerna. Om en eller flera **Loosen Table**-satser används i skriptet gör detta att Qlik Sense ignorerar alla inställningar för löst kopplade tabeller gjorda innan skriptexekveringen.

```
tablename [ , tablename2 ...]
Loosen Tables tablename [ , tablename2 ...]
```

Map ... using

Satsen **map ... using** används för att mappa ett visst fältvärde eller uttryck till värdena i en specifik mappningstabell. Mappningstabellen skapas via satsen **Mapping**.

```
Map  *fieldlist Using  mapname
```

NullAsNull

Med **NullAsNull**-satsen upphävs konvertering av NULL-värden till strängvärden som dessförinnan har ställts in med **NullAsValue**-satsen.

```
NullAsNull *fieldlist
```

NullAsValue

Satsen **NullAsValue** anger för vilka fält de NULL-värdena ska konverteras till värden.

```
NullAsValue *fieldlist
```

Kvalificera

Satsen **Qualify** används för att aktivera bestämning av fältnamn, där fältnamn får tabellnamnet som prefix.

```
Qualify *fieldlist
```

Påminnelse

rem-satsen används för att infoga kommentarer i skriptet, eller för att tillfälligt avaktivera skriptsatser utan att ta bort dem.

```
Rem string
```

Byt namn på fält

Denna skriptfunktion döper om ett eller flera Qlik Sense-fält efter att de har lästs in.

```
Rename field (using mapname|oldname to newname{ , oldname to newname })
```

```
Rename Fields (using mapname | oldname to newname{ , oldname to newname })
```

Byt namn på tabell

Denna skriptfunktion döper om ett eller flera interna tabeller i Qlik Sense efter att de har lästs in.

```
Rename table  (using mapname | oldname to newname{ , oldname to newname })
```

```
Rename Tables (using mapname | oldname to newname{ , oldname to newname })
```

Avsnitt

Med en **section**-sats är det möjligt att definiera om efterföljande **LOAD**- och **SELECT**-satser ska betraktas som data eller som en definition av behörighet.

```
Section (access | application)
```

Välj

Val av fält från en ODBC-datakälla eller en OLE DB-drivrutin görs via vanliga SQL **SELECT**-satser. Om **SELECT**-satserna accepteras eller ej beror framför allt på den ODBC-drivrutin eller OLE DB-drivrutin som används.

```
Select [all | distinct | distinctrow | top n [percent] ] *fieldlist

From tablelist

[Where criterion ]

[Group by fieldlist [having criterion ] ]

[Order by fieldlist [asc | desc] ]

[ (Inner | Left | Right | Full)Join tablename on fieldref = fieldref ]
```

Uppsättning

Satsen **set** används för att definiera skriptvariabler. Dessa kan användas för att ersätta strängar, sökvägar, drivrutiner med mera.

```
Set variablename=string
```

Sömn

sleep-satsen pausar skriptexekveringen under en angiven tidsperiod.

```
Sleep n
```

SQL

Med **SQL**-satsen kan du skicka ett godtyckligt SQL-kommando via en ODBC- eller OLE DB-koppling.

```
SQL sql_command
```

SQLColumns

Satsen **sqlcolumns** returnerar ett antal fält som beskriver kolumnerna i den ODBC- eller OLE DB-datakälla som man gjort en koppling, **connect**, till.

```
SQLColumns
```

SQLTables

Satsen **sqltables** returnerar ett antal fält som beskriver tabellerna i den ODBC- eller OLE DB-datakälla som man gjort en koppling, **connect**, till.

```
SQLTables
```

SQL-typer

Satsen **sqltypes** returnerar ett antal fält som beskriver typerna i den ODBC- eller OLE DB-datakälla som man gjort en koppling, **connect**, till.

```
SQLTypes
```

Stjärna

Den textsträng som används för att representera den totala uppsättningen av alla värden i ett fält i databasen kan ställas in med hjälp av **star**-satsen. Den påverkar efterföljande **LOAD**- och **SELECT**-satser.

```
Star is [ string ]
```

Butik

Store-satsen skapar en QVD-, Parquet-,CSV- eller TXT-fil.

```
Store [ *fieldlist from] table into filename [ format-spec ];
```

Tagg

Den här skriptsatsen erbjuder ett sätt att tilldela taggar till ett eller flera fält eller tabeller. Om du försöker tagga ett fält eller en tabell som inte finns i applikationen ignoreras taggningen. Om det finns flera förekomster av ett fältnamn eller taggnamn används det senaste värdet.

```
Tag[field|fields] fieldlist with tagname
Tag [field|fields] fieldlist using mapname
Tag table tablelist with tagname
```

Spårning

trace-satsen skriver en sträng till fönstret **Skriptexekvering** och skriptets loggfil när den används. Detta är mycket användbart vid felsökning. Med hjälp av dollarteckenexpansion av variabler som beräknas innan **trace**-satsen används kan man anpassa meddelandet.

```
Trace string
```

Unmap

Satsen **Unmap** avaktiverar fältvärdesmappningar, som har gjorts med hjälp av en tidigare **Map ... Using**-sats för fält som läses in efteråt.

```
Unmap *fieldlist
```

Avkvalificera

Unqualify-satsen upphäver tabellbestämning av fältnamn som tidigare definierats av en **Qualify**-sats.

```
Unqualify *fieldlist
```

Avtagga

Den här skriptsatsen erbjuder ett sätt att ta bort taggar från fält eller tabeller. Om du försöker ta bort taggar från ett fält eller en tabell som inte finns i applikationen, ignoreras försöket.

```
Untag[field|fields] fieldlist with tagname
Tag [field|fields] fieldlist using mapname
Tag table tablelist with tagname
```

Alias

Satsen **alias** används för att ställa in ett alias enligt vilket ett fält ska döpas om när det förekommer i skriptet som följer.

Syntax:

```
alias fieldname as aliasname {,fieldname as aliasname}
```

Argument:

Argument

Argument	Beskrivning
fieldname	Fältets namn i dina källdata
aliasname	Ett alias som du vill använda i stället

Exempel med resultat

För exemplen i tabellen nedan kommer de definierade ändringarna av fältnamn att tillämpas på referenser till det ursprungliga fältnamnet på alla efterföljande **SELECT**- och **LOAD**-satser. Ett nytt alias för ett fältnamn kan definieras med en ny **alias**-sats var som helst i skriptet.

Fältet refereras också till med namnet på aliaset i datamodellen och applikationsinnehållet.

Exempel och resultat

Exempel	Resultat
`Alias ID_N as NameID;`	Fält *ID_N* byter namn till *NameID*.
`Alias A as Name, B as Number, C as Date;`	Fälten *A*, *B* och *C* byter namn till *Name*, *Number* respektive *Date*.

Exempel på långa formulär

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskript

```
alias [A] as [AAA];
T1:
Load 'string 1' as A autogenerate 1;
T2:
Load 'string 2' as B autogenerate 1;
T3:
Load 'string 3' as A autogenerate 1;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- AAA

Resultattabell

AAA
string 1
string 3

AutoNumber

Denna sats returnerar ett unikt heltal för varje distinkt utvärderat värde i ett fält som påträffas under skriptexekveringen.

Du kan också använda *autonumber (page 691)* funktionen i ett **LADDA** påstående men den ar vissa begränsningar när du vill använda en optimerad laddning. Du kan skapa en optimerad laddning genom att ladda data från en **QVD**-fil först och sedan använda **Autonummer** påståendet för att konvertera värden till symbolnycklar.

Syntax:

```
AutoNumber *fieldlist [Using namespace] ]
```

Argument:

Argument

Argument	Beskrivning
*fältlista	En kommaseparerad lista med fält där värdena ska ersättas med ett unikt heltal. Du kan använda jokertecken ? och * i fältnamnen för att inkludera alla fält med matchande namn. Du kan också använda * för att inkludera alla fält. Du måste citera fältnamn när du använder jokertecken.
namnområde	Det är valfritt att använda `namnområdet` **`Using`**. Du kan använda det här alternativet om du vill skapa ett namnområde, där identiska värden i olika fält delar samma nyckel. Om du inte använder det här alternativet, har alla fält separat nyckelindex.

Begränsningar:

Om du har flera **LADDA** meddelanden i skriptet måste du placera **Autonummer**-meddelandet efter det sista **LADDA** meddelandet.

Exempel – skript med AutoNumber

Skriptexempel

I det här exemplet laddas data först utan satsen **AutoNumber**. Sedan läggs **AutoNumber**-satsen till för att visa effekten.

Data som används i exemplet

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa skriptexemplet nedan. Låt **AutoNumber**-satsen vara bortkommenterad så länge.

```
RegionSales:
LOAD *,
Region &'|'& Year &'|'& Month as KeyToOtherTable
INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May,    367
South, 2013,   May,    221
];

Budget:
LOAD Budget,
Region &'|'& Year &'|'& Month as KeyToOtherTable
INLINE
[Region, Year, Month, Budget
North,  2014,   May,    200
North,  2014,   May,    350
North,  2014,   June,   150
South,  2014,   June,   500
South,  2013,   May,    300
South,  2013,   May,    200
];

//AutoNumber KeyToOtherTable;
```

Skapa visualiseringar

Skapa två tabellvisualiseringar i ett Qlik Sense-ark. Lägg till **KeyToOtherTable**, **Region**, **Year**, **Month** och **Sales** som dimensioner i den första tabellen. Lägg till **KeyToOtherTable**, **Region**, **Year**, **Month** och **Budget** som dimensioner i den andra tabellen.

Resultat

Tabellen RegionSales

KeyToOtherTable	Region	Year	Month	Sales
North|2014|June	North	2014	June	127

KeyToOtherTable	Region	Year	Month	Sales
North|2014|May	North	2014	May	245
North|2014|May	North	2014	May	347
South|2013|May	South	2013	May	221
South|2013|May	South	2013	May	367
South|2014|June	South	2014	June	645

Tabellen Budget

KeyToOtherTable	Region	Year	Month	Budget
North|2014|June	North	2014	June	150
North|2014|May	North	2014	May	200
North|2014|May	North	2014	May	350
South|2013|May	South	2013	May	200
South|2013|May	South	2013	May	300
South|2014|June	South	2014	June	500

Förklaring

I exemplet visas det sammansatta fältet **KeyToOtherTable** som länkar ihop de båda tabellerna. **AutoNumber** används inte. Observera längden på **KeyToOtherTable**-värdena.

Lägga till AutoNumber-satsen

Ta bort kommentarsmarkeringen på **AutoNumber**-satsen i laddningsskriptet.

```
AutoNumber KeyToOtherTable;
```

Resultat

Tabellen RegionSales

KeyToOtherTable	Region	Year	Month	Sales
1	North	2014	June	127
1	North	2014	May	245
2	North	2014	May	347
3	South	2013	May	221
4	South	2013	May	367
4	South	2014	June	645

Tabellen Budget

KeyToOtherTable	Region	Year	Month	Budget
1	North	2014	June	150
1	North	2014	May	200
2	North	2014	May	350
3	South	2013	May	200
4	South	2013	May	300
4	South	2014	June	500

Förklaring

Fältvärdena i **KeyToOtherTable** har ersatts med unika heltalsvärden och därmed reduceras även längden på fältvärdena, vilket sparar minne. Nyckelfälten i de båda tabellerna påverkas av **AutoNumber** och tabellerna förblir länkade. Exemplet är kortfattat eftersom det är avsett som en illustration, men blir meningsfullt med en tabell som innehåller ett stort antal rader.

Binary

binary-satsen används för att ladda data från en annan Qlik Sense applikation eller QlikView-dokument, inklusive Section Access-data. Övriga element i applikationen ingår inte, exempelvis ark, berättelser, visualiseringar, originalobjekt eller variabler.

Endast en **binary**-sats tillåts i skriptet. **binary**-satsen måste vara den första satsen i skriptet, till och med före SET-satserna som oftast finns i början av skriptet.

```
binary [path] filename
```

Argument

Argument	Beskrivning
path	Sökvägen till filen, som ska vara en referens till en mappdatakoppling. Detta krävs om filen inte finns i Qlik Sense-arbetskatalogen. **Exempel:** ***'lib://Table Files/'*** I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:**data*\ • relativ till den applikation som innehåller denna skriptrad. **Exempel:** *data*\
filename	Filens namn, inklusive tillägget .qvw eller .qvf.

Du kan inte använda **binary** för att ladda data från en applikation på samma Qlik Sense Enterprise distribution genom att referera till applikationens ID. Du kan bara ladda från en *.qvf*-fil.

Exempel

Sträng	Beskrivning
`Binary lib://DataFolder/customer.qvw;`	I det här exemplet måste filen finnas i **Mapp**-datakopplingen. Det kan till exempel vara en mapp som administratören skapar på Qlik Sense-servern. Klicka på **Skapa ny koppling** i Skriptredigeraren och välj sedan **Mapp** under **Filplatser**.
`Binary customer.qvf;`	I det här exemplet måste filen finnas i Qlik Sense-arbetskatalogen.
`Binary c:\qv\customer.qvw;`	Det här exemplet, som använder en absolut sökväg, fungerar bara i bakåtkompatibelt skriptläge.

Comment field

Gör det möjligt att visa fältkommentarerna (metadata) från databaser eller kalkylblad. Fältnamn som inte finns i applikationen ignoreras. Om flera förekomster av ett fältnamn hittas används det senaste värdet.

Syntax:

```
comment [fields] *fieldlist using mapname
```

```
comment [field] fieldname with comment
```

Mappningstabellen som används bör ha två kolumner: en med fältnamn och en annan med kommentarer.

Argument:

Argument

Argument	Beskrivning
**fieldlist*	En kommaavgränsad lista över fält som ska kommenteras. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken.
mapname	Namnet på en mappningstabell som tidigare har lästs in en **LOAD**- eller **SELECT**-sats för mappning.
fieldname	Namnet på det fält som ska kommenteras.
comment	Den kommentar som ska läggas till i fältet.

Example 1:

```
commentmap:

mapping LOAD * inline [

a,b

Alpha,This field contains text values

Num,This field contains numeric values

];

comment fields using commentmap;
```

Example 2:

```
comment field Alpha with AFieldContainingCharacters;

comment field Num with '*A field containing numbers';

comment Gamma with 'Mickey Mouse field';
```

Comment table

Gör det möjligt att visa tabellkommentarerna (metadata) från databaser eller kalkylblad.

Tabellnamn som inte finns i applikationen ignoreras. Om flera förekomster av ett tabellnamn hittas används det senaste värdet. Nyckelordet kan användas för att läsa kommentarer från en datakälla.

Syntax:

```
comment [tables] tablelist using mapname
```

```
comment [table] tablename with comment
```

Argument:

Argument

Argument	Beskrivning
tablelist	(table{,table})
mapname	Namnet på en mappningstabell som tidigare har lästs i en **LOAD**- eller **SELECT**-sats för mappning.
tablename	Namnet på den tabell som ska kommenteras.
comment	Den kommentar som ska läggas till i tabellen.

Example 1:

```
Commentmap:

mapping LOAD * inline [

a,b

Main,This is the fact table

Currencies, Currency helper table

];

comment tables using Commentmap;
```

Example 2:

```
comment table Main with 'Main fact table';
```

Connect

CONNECT-satsen används för att ange Qlik Sense-åtkomst till en allmän databas via OLE DB/ODBC-gränssnittet. För ODBC måste datakällan först anges med hjälp av ODBC-administratören.

Den här funktionen är inte tillgänglig i Qlik Sense SaaS.

Satsen har enbart stöd för mappdatakopplingar i standardläget.

```
ODBC CONNECT TO connect-string
OLEDB CONNECT TO connect-string
CUSTOM CONNECT TO connect-string
LIB CONNECT TO connection
```

Argument

Argument	Beskrivning

connect-string	`connect-string ::= datasourcename { ; conn-spec-item }`
	Kopplingssträngen består av namnet på datakällan och om du vill en lista över en eller flera kopplingsspecifikatorer. Om namnet på datakällan innehåller blanktecken eller kopplingsspecifikatorer, måste strängen stå inom citationstecken.
	datasourcename måste vara en angiven ODBC-datakälla eller en sträng som anger en OLE DB-leverantör.
	`conn-spec-item ::=`**`DBQ=`**`database_specifier |`**`DriverID=`**`driver_specifier |`**`UID=`**`userid |`**`PWD=`**`password`
	Vilka kopplingsspecifikatorer som är möjliga varierar mellan olika databaser. För vissa databaser är ytterligare specifikatorer tillåtna. För OLE DB är vissa kopplingsrelaterade poster obligatoriska.

connection	Namnet på en datakoppling som finns sparad i Skriptredigeraren.

Om **ODBC** placeras före **CONNECT** kommer gränssnittet ODBC att användas, annars kommer OLE DB att användas.

Med **LIB CONNECT TO** kopplar du upp dig till en databas med hjälp av en lagrad datakoppling som har skapats i Dataladdningsredigeraren.

```
ODBC CONNECT TO 'Sales
DBQ=C:\Program Files\Access\Samples\Sales.mdb';
```

Datakällan som specificeras genom denna sats används av alla följande **Select (SQL)**-satser tills en ny **CONNECT**-sats skapas.

```
LIB CONNECT TO 'DataConnection';
```

Connect32

Den här satsen används likadant som **CONNECT**-satsen, men den tvingar ett 64-bitarssystem att använda en 32-bitars ODBC/OLE DB-provider. Kan ej användas för anpassade uppkopplingar.

Connect64

Den här satsen används likadant som **CONNECT**-satsen, men den kräver en 64-bitars drivrutin. Kan ej användas för anpassade uppkopplingar.

Constrain

Satsen **Constrain** kan användas i kombination med satserna **Let** eller **Set** för att definiera variabler i skriptet. Med hjälp av **Constrain**-satsen kan du definiera begränsningar för möjliga värden för dessa variabler. Om en definition av en variabel bryter mot begränsningarna misslyckas inläsningen. Med begränsningar kan du kräva att värden för variabler ska matcha specifika typer, ligga inom specifika numeriska intervall och matcha acceptabla värden som du definierar.

```
Constrain variablename = json
```

Där:

- `variablename` är en skriptvariabel.
- `json` är ett giltigt JSON-objekt som specificerar begränsningarna. Individuella begränsningar läggs till i detta objekt som nyckel-värde-par.

Begränsningstyper

Begränsningstyp (nyckel)	Betydelse	Begränsande krav (värde)	Exempel
`"type"`	Begränsa variabelvärden till en specifik datatyp.	`"text"` anger en texttyp. `"number"` anger en numerisk typ.	`CONSTRAIN vExampleText = {"type": "text"}` `CONSTRAIN vExampleNumeric = {"type": "number"}`
`"maxnum"`	Ange ett maxvärde för en variabel.	Endast siffror (heltal eller flyttal). Vetenskaplig notation är tillåten.	`CONSTRAIN vExample = {"maxnum": 5000}`
`"minnum"`	Ange ett minvärde för en variabel.	Endast siffror (heltal eller flyttal). Vetenskaplig notation är tillåten.	`CONSTRAIN vExample = {"minnum": 250}`
`"valuesnum"`	Definiera en lista över acceptabla numeriska värden för variabeln.	Kommaseparerad lista med siffror inom hakparenteser. Till exempel: `[1,2,3]`	`CONSTRAIN vExample = {"valuesnum": [1,2,3]}`
`"valuestext"`	Definiera en lista	Kommaseparerad lista	`CONSTRAIN vExample = {"valuestext": ["Department A", "Department B", "Department C"]}`

Begränsningstyp (nyckel)	Betydelse	Begränsande krav (värde)	Exempel
	över acceptabla textvärden för variabeln.	med strängar inom hakparenteser. Till exempel: `["a","b","c"]`	

När du ska använda en **Constrain** sats

Använd **Constrain**-satsen för att förhindra att oönskade variabelvärden laddas in i analysapplikationer.

Överväganden

- Definitionen **Constrain** för variabeln måste vara ett giltigt JSON-objekt.
- Du kan ange mer än en begränsning för en och samma variabel. Alla begränsningar för en variabel måste behållas inom ett enda objekt för begränsningar. Se *Exempel – Flera begränsningar (page 160)* för exempel.
- Om du redan har läst in en skriptvariabel med ett visst namn i applikationen kan framtida försök att definiera begränsningar och värden för den variabeln misslyckas på grund av konflikter mellan de ursprungliga och de omdefinierade variablerna.
  För att lösa dessa konflikter, återställ begränsningarna och värdena för variablerna genom att infoga tomma definitioner. Exempel:
  ```
  CONSTRAIN x;
  LET x;
  CONSTRAIN x;
  SET x;
  ```
  Du kan sedan omdefiniera variabeln och dess begränsningar i efterföljande delavsnitt i laddningsskriptet. Se *Exempel – lösning av konflikter vid definition av variabler (page 161)* för ett utförligt exempel.
- Ordningen på definitionen av variabeln och definitionen av begränsningen spelar ingen roll. Vid ett variabelrelaterat inläsningsfel kommer dock linjen vid vilken brytpunkten inträffar att vara annorlunda.
- **Constrain**-skriptet begränsar endast variabeldefinitioner för skriptvariabler. Du kan också tillämpa begränsningar på skript- och slutanvändarvariabler (de som skapas eller uppdateras i arkvyn eller som sessionsvariabler med hjälp av API:et) genom att använda det publika API:et. Mer information finns i *Alternativ för utvecklare för att definiera begränsningar för variabler (page 156)*.
- Begränsningar valideras för att undvika scenarier som inte kan matcha något värde för variabeln. Exempel på ogiltiga begränsningar:
  - `{"minnum": 5000, "maxnum": 4}`
  - `{"type": "text", "maxnum": 4}`

Alternativ för utvecklare för att definiera begränsningar för variabler

Du kan också använda det publika API:et för att definiera variabelbegränsningar i stället för att använda **Constrain**-satsen i laddningsskriptet. När du definierar begränsningar via API, kommer alla försök att uppdatera variabelvärden – oavsett om det sker direkt i laddningsskriptet eller i arkvyn – att omfattas av begränsningarna.

Om du vill definiera begränsningar för variabler via det offentliga API:et, använder du GenericVariableConstraints egenskapen i API:et QlikEngine JSON: GenericVariableProperties.

Exempel - type

Kunde inte läsa in `"typ"`

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En **Constrain**-sats för att ange att definitionen av variabeln för `a` måste ha typen `number`.
- En **LET**-sats som definierar variabeln `a`.

Laddningsskript

```
CONSTRAIN a = {"type": "text"};
LET a = 1000;
```

Resultat

Ladda data.

Inläsningen misslyckas eftersom begränsningen kräver en texttyp för variabeln `a`. Definitionen av variabeln är ett tal.

Om man däremot anger en `type` av `number` i begränsningen kommer inläsningen att lyckas.

Exempel - maxnum

Kunde inte läsa in med `"maxnum"`

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En **Constrain**-sats för att ange att definitionen av variabeln för `b` kan ha ett maxvärde på 3000.
- En **LET**-sats som definierar variabeln `b`.

```
CONSTRAIN b = {"maxnum": "3000"};
LET b = 3001;
```

Resultat

Ladda data.

Inläsningen misslyckas eftersom begränsningen kräver ett maxvärde på 3000 för variabel `b`. Variabeldefinitionen utvärderas som 3001.

Om man däremot anger `b` som 2999 kommer inläsningen att lyckas.

Exempel - minnum

Kunde inte läsa in `"minnum"`

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En **Constrain**-sats för att ange att variabeldefinitionen för `b` kan ha ett minvärde på 3000.
- En **LET**-sats som definierar variabeln `c`.

```
CONSTRAIN c = {"minnum": "3000"};
LET c = 2999;
```

Resultat

Ladda data.

Inläsningen misslyckas eftersom begränsningen kräver ett minvärde på 3000 för variabel `c`. Variabeldefinitionen utvärderas som 2999.

Om man däremot anger `c` som 3000 kommer inläsningen att lyckas.

Exempel - valuesnum

Misslyckad omladdning med `"valuesnum"`

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En **Constrain**-sats för att ange att variabeldefinitionen för `d` måste vara numerisk och måste vara 2, 3 eller 4.
- En **LET**-sats som definierar variabeln `d`.

```
CONSTRAIN d = {"valuesnum": [2,3,4]};
LET d = 1;
```

Resultat

Ladda data.

Inläsningen misslyckas eftersom variabeln `d` utvärderas till 1, vilket inte finns med i listan över tre möjliga värden.

Om man däremot anger `d` som 2 kommer inläsningen att lyckas.

Exempel - valuestext

Kunde inte läsa in `"värdetext"`

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En **Constrain**-sats för att ange att definitionen av variabeln för `e` måste vara ett av följande textvärden:
    - Department A
    - Department B
    - Department C
- En **SET**-sats som definierar variabeln `e`.

```
CONSTRAIN e = {"valuestext": ["Department A", "Department B", "Department C"]};
SET e = Department D;
```

Resultat

Ladda data.

Inläsningen misslyckas eftersom variabeln `e` är `Department D`, vilket inte finns med i listan över tillåtna värden som definieras av begränsningen.

Om man däremot anger `e` som `Department C` kommer inläsningen att lyckas.

Exempel – SET kontra LET

Dessa exempel visar skillnaderna mellan hur begränsningar tillämpas när **SET** eller **LET** används för att definiera variabler. I alla exempel används begränsningar som tillämpar ett type-krav, men dessa principer gäller generellt för alla typer av begränsningar.

Kunde inte läsa in med `SET`

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En **Constrain**-sats för att ange att definitionen av variabeln för `f` måste ha typen `number`.
- En **SET**-sats som definierar variabeln `f`.

Laddningsskript

```
CONSTRAIN f = {"type": "number"};
SET f = 500*2;
```

Resultat

Ladda data.

Inläsningen misslyckas eftersom begränsningen kräver en numerisk typ för variabeln `f`. Definitionen av variabeln innehåller ett numeriskt uttryck, men eftersom en **SET**-sats används utvärderas definitionen som text.

Kunde inte läsa in `LET`

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En **Constrain**-sats för att ange att definitionen av variabeln för `g` måste ha typen `text`.
- En **LET**-sats som definierar variabeln `g`.

Laddningsskript

```
CONSTRAIN g = {"type": "text"};
LET g = 500*2;
```

Ladda data.

Inläsningen misslyckas eftersom begränsningen kräver en texttyp för variabeln `g`. Definitionen av variabeln innehåller ett numeriskt uttryck och en **LET**-sats används vilket gör att definitionen utvärderas som text.

Lyckad inläsning med `SET`

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En **Constrain**-sats för att ange att definitionen av variabeln för `h` måste ha typen `text`.
- En **SET**-sats som definierar variabeln `h`.

Laddningsskript

```
CONSTRAIN h = {"type": "text"};
SET h = 500*2;
```
Ladda data.

Inläsningen lyckas. Inläsningen misslyckas eftersom begränsningen kräver en texttyp för variabeln `h`. Definitionen av variabeln innehåller ett numeriskt uttryck, men eftersom en **SET**-sats används utvärderas definitionen som text.

Lyckad inläsning med `LET`

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En **Constrain**-sats för att ange att definitionen av variabeln för `i` måste ha typen `number`.
- En **LET**-sats som definierar variabeln `i`.

Laddningsskript

```
CONSTRAIN i = {"type": "number"};
LET i = 500*2;
```
Ladda data.

Inläsningen lyckas. Begränsningen kräver en numerisk typ för variabeln `i`. Definitionen av variabeln innehåller ett numeriskt uttryck och en **LET**-sats används vilket gör att definitionen utvärderas som text.

Exempel – Flera begränsningar

Inställning av min- och maxvärden

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En **Constrain**-sats för att ange att definitionen av variabeln för `j` måste vara mellan 1 och 5.
- En **LET**-sats som definierar variabeln `j`.

Laddningsskript

```
CONSTRAIN j = {"minnum": 1, "maxnum": 5};
LET j = 2;
```
Ladda data.

Inläsningen lyckas. Variabeln `f` utvärderas till 2, vilket ligger inom det acceptabla intervallet.

Accepterar både text och siffror från listor

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En **Constrain**-sats som gör att variabeln kan vara text eller numerisk, så länge det är ett av de acceptabla värdena.
- En **LET**-sats som definierar variabeln `k`.

Laddningsskript

```
CONSTRAIN k = {"valuestext": ["abc","def","ghi"], "valuesnum": [0,1,3,5]};
LET k = 'def';
```
Ladda data.

Inläsningen lyckas. Värdet för variabeln `k` finns i listan över godtagbara värden. På samma sätt skulle en angivelse av `k` som 3 resultera i en lyckad inläsning.

Exempel – lösning av konflikter vid definition av variabler

Laddningsskript och resultat

Översikt

Det här exemplet visar hur variabeldefinitionskonflikter kan uppstå när man lägger till begränsningar för en variabel som redan har definierats i applikationen från en tidigare inläsning.

Laddningsskript 1: ställa in en variabel för skriptet

Öppna Skriptredigeraren. Lägg till följande skript:

```
LET vConflictExample = 'Department C';
```
Ladda data. Inläsningen lyckades. Variabeln `vConflictExample` har skapats i applikationen som en skriptvariabel.

Laddningsskript 2: försöker definiera begränsning på variabel i tidigare laddningsskript

Ta bort laddningsskript 1. Lägg till följande skript:

```
CONSTRAIN vConflictExample = {"type": "number"};
LET vConflictExample = 123;
```
Ladda data.

Inläsningen misslyckas med ett begränsningsfel, trots att den nya begränsningen och värdedefinitionen är kompatibla. Detta beror på att laddningsskript 1 redan har läst in variabeln i applikationen, med ett textvärde, och detta värde finns fortfarande kvar i applikationen. För att lösa det här problemet måste vi återställa variabelns begränsning och definition. Se laddningsskript 3.

Laddningsskript 3: återställning av definitioner av begränsningar och värden

Ta bort laddningsskript 2. Lägg till följande skript:

```
CONSTRAIN vConflictExample;
LET vConflictExample;

CONSTRAIN vConflictExample = {"type": "number"};
LET vConflictExample = 123;

MockData:
LOAD
*
Inline [
id, date, amount
1, 05/30/2025, 23.56
];
```

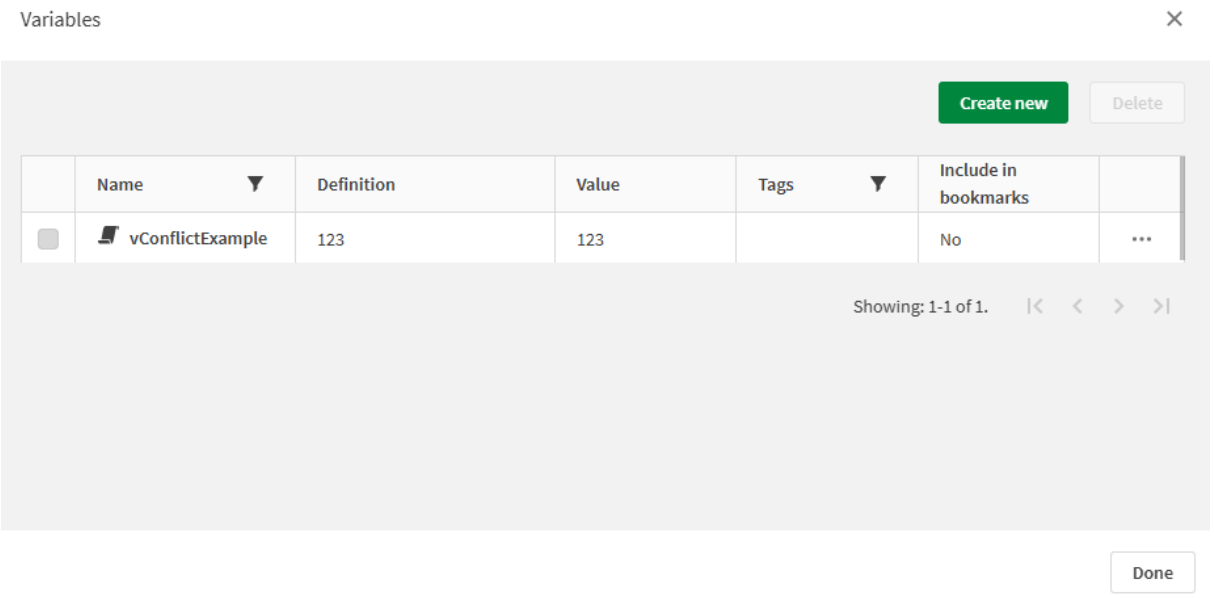
Ladda data.

Inläsningen lyckas. Skriptet ovan kan delas upp i tre delar, var och en åtskild av tomma rader:

- Den första delen återställer begränsnings- och värdedefinitionen för `vConflictExample`. Detta tar bort definitionen som laddades in i applikationen med laddningsskript 1.
- Den andra delen återanvänder de nya begränsnings- och värdedefinitionerna som misslyckades i laddningsskript 2.
- Den tredje delen lägger till testdata så att vi kan öppna arkvyn och granska värdet på variabeln (se nedan).

Öppna arkvyn i redigeringsläge och öppna dialogrutan **Variabler**. Vi kan se att variabeln `vConflictExample` har skapats.

Dialogrutan Variabler i arkvyn visar att konflikten med definitionen av variabeln för `vConflictExample` har lösts.



Declare

Declare-satsen används för att skapa fältdefinitioner där du kan definiera relationer mellan fält eller funktioner. En uppsättning fältdefinitioner kan användas för att automatiskt generera härledda fält, vilka kan användas som dimensioner. Du kan till exempel skapa en kalenderdefinition och använda den för att generera relaterade dimensioner, som år, månad, vecka och dag, från ett datumfält.

Du kan använda **Declare** antingen för att skapa en ny fältdefinition eller en fältdefinition baserad på en befintlig definition.

Skapa en ny fältdefinition

Syntax:

```
definition_name:

Declare [Field[s]] Definition [Tagged tag_list ]

[Parameters parameter_list ]

Fields field_list
```

Argument:

Argument	Beskrivning
definition_name	Fältdefinitionens namn, avslutad med kolon. *Använd inte autoCalendar som namn på fältdefinitioner eftersom detta namn är reserverat för automatiskt genererade kalendermallar.* **Exempel:** `Calendar:`
tag_list	En kommaavgränsad lista över taggar att tillämpa på fält som härletts från fältdefinitionen. Det är valfritt att tillämpa taggar, men om du inte tillämpar taggar för att specificera sorteringsordning, som t.ex. $date, $numeric eller $text, kommer det härledda fältet som standard att sorteras efter laddningsordning. **Exempel:** `'$date'Thank you for bringing this to our attention, and apologies for the inconvenience.`

Argument	Beskrivning
parameter_list	En kommaavgränsad lista över parametrar. En parameter definieras i formen `name=value` och tilldelas ett startvärde, vilket kan åsidosättas när en fältdefinition återanvänds. Valfritt. **Exempel:** `first_month_of_year = 1`
field_list	En kommaavgränsad lista över fält att generera när fältdefinitionen används. Ett fält definieras i formen `<expression> As field_name tagged tag`. Använd `$1` för att referera till datafältet som de härledda fälten ska genereras från. **Exempel:** `Year($1) As Year tagged ('$numeric')`

Exempel:

```
Calendar:
DECLARE FIELD DEFINITION TAGGED '$date'
   Parameters
      first_month_of_year = 1
   Fields

      Year($1) As Year Tagged ('$numeric'),
      Month($1) as Month Tagged ('$numeric'),
      Date($1) as Date Tagged ('$date'),
      Week($1) as Week Tagged ('$numeric'),
      Weekday($1) as Weekday Tagged ('$numeric'),
      DayNumberOfYear($1, first_month_of_year) as DayNumberOfYear Tagged ('$numeric')
;
```

Kalendern är nu definierad och du kan tillämpa den på datumfälten som du har laddat, i det här fallet OrderDate och ShippingDate, med en**Derive**-sats.

Återanvända en befintlig fältdefinition

Syntax:

```
<definition name>:

Declare [Field][s] Definition

Using <existing_definition>

[With <parameter_assignment> ]
```

Argument:

Argument	Beskrivning
definition_ name	Fältdefinitionens namn, avslutad med kolon. **Exempel:** `MyCalendar:`
existing_ definition	Fältdefinitionen som ska återanvändas för att skapa den nya fältdefinitionen. Den nya fältdefinitionen kommer att fungera på samma sätt som definitionen den är baserad på, med undantag för om du använder parameter_assignment för att ändra värdet som används i fältuttrycken. **Exempel:** `Using Calendar`
parameter_ assignment	En kommaavgränsad lista över parametertilldelningar. En parametertilldelning definieras i formen `name=value` och åsidosätter parametervärdet som är angett i basfältdefinitionen. Valfritt. **Exempel:** `first_month_of_year = 4`

Exempel:

I det här exemplet återanvänder vi kalenderdefinitionen som skapades i föregående exempel. I det här fallet vill vi använda ett budgetår som börjar i april. Detta uppnår vi genom att tilldela värdet 4 till first_ month_of_year-parametern, vilket påverkar DayNumberOfYear-fältet som definieras.

I exemplet antas att du använder samma exempeldata och fältdefinition som i föregående exempel.

```
MyCalendar:
DECLARE FIELD DEFINITION USING Calendar WITH first_month_of_year=4;



DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING MyCalendar;
```

När du har laddat dataskriptet blir de genererade fälten tillgängliga i arkredigeraren med namnen OrderDate.MyCalendar.* och ShippingDate.MyCalendar.*.

Derive

Derive-satsen används för att generera härledda fält baserat på en fältdefinition som skapats med en **Declare**-sats. Du kan antingen specifikt ange vilka datafält du vill härleda fält från, eller härleda dem explicit eller implicit baserat på fälttaggar.

Syntax:

```
Derive [fields]] From [Field[s]] field_list Using definition

Derive [Field[s]] From Explicit [Tag[s]] tag_list Using definition

Derive [Field[s]] From Implicit [Tag[s]] Using definition
```

Argument:

Argument

Argument	Beskrivning
definition	Namnet på fältdefinitionen som ska användas för att härleda fält. **Exempel:** `Calendar`
field_list	En kommaavgränsad lista över datafält som de härledda fälten ska genereras från, baserat på fältdefinitionen. Datafälten ska vara fält som du redan har laddat i skriptet. **Exempel:** `OrderDate, ShippingDate`
tag_list	En kommaavgränsad lista över taggar. Härledda fält kommer att genereras för alla datafält med någon av taggarna i listan. Tagglistan ska omges av runda parenteser. **Exempel:** `('$date','$timestamp')`

Exempel:

- Härleda fält för specifika datafält.
  I det här fallet anger vi OrderDate- och ShippingDate-fälten.
  `DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING Calendar;`
- Härleda fält för alla fält med en specifik tagg.
  I det här fallet härleder vi fält baserat på Calendar för alla fält med en `$date`-tagg.
  `DERIVE FIELDS FROM EXPLICIT TAGS ('$date') USING Calendar;`
- Härleda fält för alla fält med fältdefinitionstaggen.
  I det här fallet härleder vi fält för alla datafält med samma tagg som Calendar-fältsdefinitionen, vilken i detta fall är `$date`.
  `DERIVE FIELDS FROM IMPLICIT TAG USING Calendar;`

Direct Query

Med **DIRECT QUERY**-satsen kan du komma åt tabeller genom enODBC eller OLE DB-koppling med hjälp av funktionen Direkt upptäckt.

Syntax:

```
DIRECT QUERY DIMENSION fieldlist [MEASURE fieldlist] [DETAIL fieldlist] FROM
tablelist
[WHERE where_clause]
```

Nyckelorden **DIMENSION**, **MEASURE** och **DETAIL** kan användas i valfri ordning.

Nyckelordssatserna **DIMENSION** och **FROM** krävs i alla **DIRECT QUERY**-satser. Nyckelordet **FROM** måste komma efter nyckelordet **DIMENSION**.

Fälten som anges direkt efter nyckelordet **DIMENSION** laddas i minnet och kan användas för att skapa associationer mellan data i minnet och Direkt upptäckt-data.

> *DIRECT QUERY-satsen får inte innehålla DISTINCT- eller GROUP BY-satser.*

Med hjälp av nyckelordet **MEASURE** kan du definiera fält som Qlik Sense kan tolka på "metanivå". Faktiska data för ett measure-fält finns enbart i databasen under dataladdningsprocessen, och hämtas vid behov utifrån de diagramuttryck som används i en visualisering.

Normalt bör fält med diskreta värden som ska användas som dimensioner laddas in med nyckelordet **DIMENSION**, medan tal som ska användas i aggregeringar enbart bör markeras med nyckelordet **MEASURE**.

DETAIL fälten innehåller information eller detaljer, exempelvis fälten ”Comment”, som en användare kan välja att visa i en detaljerad tabellbox. **DETAIL**-fält kan inte användas i diagramuttryck.

DIRECT QUERY-satsen är utformad för att vara datakällsneutral för datakällor som har stöd för SQL. Tack vare detta kan samma **DIRECT QUERY**-sats användas för olika SQL-databaser utan att behöva ändras. Direkt upptäckt genererar frågor lämpliga för databasen när så krävs.

Programegen syntax för datakällan kan användas när användaren vet vilken databas frågan ska skickas till och vill utnyttja databasspecifika komplement till SQL. Programegen syntax för datakällan stöds:

- Som fältuttryck i **DIMENSION**- och **MEASURE**-satser
- Som innehåll i **WHERE**-satsen

Exempel:

```
DIRECT QUERY

      DIMENSION Dim1, Dim2
      MEASURE

            NATIVE ('X % Y') AS X_MOD_Y

FROM TableName
DIRECT QUERY

      DIMENSION Dim1, Dim2
      MEASURE X, Y
      FROM TableName
      WHERE NATIVE ('EMAIL MATCHES "\*.EDU"')
```

> *Följande termer används som nyckelord och kan inte användas som kolumn- eller fältnamn utan att placeras inom citationstecken: and, as, detach, detail, dimension, distinct, from, in, is, like, measure, native, not, or, where*

Argument:

Argument	Beskrivning
fieldlist	En kommaavgränsad lista med fältspecifikationer, *fieldname {, fieldname}*. En fältspecifikation kan vara ett fältnamn, och i så fall används samma namn som databaskolumnens namn och Qlik Sense-fältnamn. En fältspecifikation kan även vara ett ”fältalias”. Då får ett databasuttryck eller ett kolumnnamn ett Qlik Sense-fältnamn.
tablelist	En lista över namnen på tabeller eller vyer i databasen som data kommer att läsas in från. Normalt är det vyer som innehåller en JOIN som utförs på databasen.

where_ clause	Den fullständiga syntaxen för **WHERE**-databassatser anges inte här, men de flesta ”SQL- relationsuttryck” är tillåtna, inklusive användning av funktionsanrop, **LIKE**-operatorn för strängar, **IS NULL** och **IS NOT NULL** och **IN. BETWEEN** ingår inte.
	
	NOT är en unär operator i motsats till en modifierare för vissa nyckelord.
	
	Exempel:
	
	`WHERE x > 100 AND "Region Code" IN ('south', 'west')`
	`WHERE Code IS NOT NULL and Code LIKE '%prospect'`
	`WHERE NOT X in (1,2,3)`
	Det sista exemplet kan inte skrivas som:
	
	`WHERE X NOT in (1,2,3)`

Exempel:

I det här exemplet används en databastabell som heter TableName och innehåller fälten Dim1, Dim2, Num1, Num2 och Num3.Dim1 och Dim2 laddas in i datauppsättningen Qlik Sense.

```
DIRECT QUERY DIMENSTION Dim1, Dim2 MEASURE Num1, Num2, Num3 FROM TableName ;
```

Dim1 och Dim2 blir tillgängliga att använda som dimensioner. Num1, Num2 och Num3 blir tillgängliga för aggregeringar. Dim1 och Dim2 är också tillgängliga för aggregeringar. Vilken typ av aggregeringar Dim1 och Dim2 kan användas för beror på deras datatyper. I många fall innehåller till exempel **DIMENSION**-fält strängdata som namn eller kontonummer. Dessa fält kan inte läggas ihop, men de kan räknas: `count (Dim1)`.

__DIRECT QUERY__-satser skrivs direkt i Skriptredigeraren. För att underlätta vid konstruktionen av __DIRECT QUERY__-satser kan du generera en __SELECT__-sats från en dataanslutning och sedan redigera det genererade skriptet för att ändra det till en __DIRECT QUERY__-sats.
Till exempel kan __SELECT__-satsen:

```
SQL SELECT
 SalesOrderID,
 RevisionNumber,
 OrderDate,
 SubTotal,
 TaxAmt
FROM MyDB.Sales.SalesOrderHeader;
```

ändras till följande __DIRECT QUERY__-sats:

```
DIRECT QUERY
 DIMENSION
 SalesOrderID,
 RevisionNumber

MEASURE
 SubTotal,
 TaxAmt

DETAIL
OrderDate

FROM MyDB.Sales.SalesOrderHeader;
```

Direkt upptäckt fältlistor

En fältlista är en kommaseparerad lista med fältspecifikationer, *fieldname {, fieldname}*. En fältspecifikation kan vara ett fältnamn, och i så fall används samma namn som databaskolumnens namn och fältnamn. En fältspecifikation kan även vara ett fältalias. Då får ett databasuttryck eller ett kolumnnamn ett Qlik Sense-fältnamn.

Fältnamn kan vara antingen enkla namn eller namn inom citationstecken. Ett enkelt namn börjar med ett alfabetiskt Unicode-tecken och följs av valfri kombination av alfabetiska eller numeriska tecken eller understreck. Namn inom citationstecken börjar med dubbla citationstecken och innehåller en valfri följd av tecken. Om ett namn inom citationstecken innehåller citationstecken, representeras dessa av två citationstecken bredvid varandra.

Qlik Sense-fältnamn är skiftlägeskänsliga. Databasfältnamn kan vara skiftlägeskänsliga eller inte, beroende på databasen. En Direkt upptäckt-fråga bevarar skiftläget för alla fältidentifierare och alias. I följande exempel används aliaset "MyState" internt för att lagra data från databaskolumnen "STATEID".

```
DIRECT QUERY Dimension STATEID as MyState Measure AMOUNT from SALES_TABLE;
```

Detta skiljer sig från resultatet av en **SQL Select**-sats med ett alias. Om inte aliaset explicit är inom citationstecken innehåller resultatet standardskiftläget för kolumnen som returneras av måldatabasen. I följande exempel skapar **SQL Select**-satsen till en Oracle-databas "MYSTATE," med endast stora bokstäver som det interna Qlik Sense-aliaset, trots att aliaset är angivet med både stora och små bokstäver. **SQL Select**-satsen använder kolumnnamnet som returneras av databasen, och för Oracle har det bara stora bokstäver.

```
SQL Select STATEID as MyState, STATENAME from STATE_TABLE;
```

För att undvika detta kan du använda LOAD-satsen för att ange aliaset.

```
Load STATEID as MyState, STATENAME;
SQL Select STATEID, STATEMENT from STATE_TABLE;
```

I det här exemplet lagras "STATEID"-kolumnen internt av Qlik Sense som "MyState".

De flesta skalära databasuttryck är tillåtna som fältspecifikationer. Funktionsanrop kan också användas i fältspecifikationer. Uttryck kan innehålla konstanter som är booleska, numeriska eller strängar inom enkla citationstecken (inbäddade enkla citationstecken representeras av enkla citationstecken bredvid varandra).

Exempel:

```
DIRECT QUERY

   DIMENSION

      SalesOrderID, RevisionNumber

   MEASURE

      SubTotal AS "Sub Total"

FROM AdventureWorks.Sales.SalesOrderHeader;


DIRECT QUERY

   DIMENSION

      "SalesOrderID" AS "Sales Order ID"

   MEASURE

      SubTotal,TaxAmt,(SubTotal-TaxAmt) AS "Net Total"

FROM AdventureWorks.Sales.SalesOrderHeader;


DIRECT QUERY

   DIMENSION
```

```
            (2*Radius*3.14159) AS Circumference,

            Molecules/6.02e23 AS Moles

         MEASURE

            Num1 AS numA

      FROM TableName;


DIRECT QUERY
   DIMENSION
      concat(region, 'code') AS region_code
   MEASURE
      Num1 AS NumA
FROM TableName;
```

Direkt upptäckt stöder inte användning av aggregeringar i **LOAD**-satser. Om aggregeringar används blir resultaten oförutsägbara. En **LOAD**-sats som följande bör inte användas:

```
DIRECT QUERY DIMENSION stateid, SUM(amount*7) AS MultiFirst MEASURE amount FROM sales_table;
```

SUM bör inte finnas i **LOAD**-satsen.

Direkt upptäckt stöder inte heller Qlik Sense-funktioner i **Direct Query**-satser. Till exempel resulterar följande specifikation för ett **DIMENSION**-fält i misslyckande när "Mth"-fältet används som en dimension i en visualisering:

```
month(ModifiedDate) as Mth
```

Directory

Directory-satsen anger i vilken katalog datafilerna ska sökas i följande **LOAD**-satser tills en ny **Directory**-sats anges.

```
Directory[path]
```

Om **Directory**-satsen används utan någon **path** eller utelämnas, kommer Qlik Sense att leta i Qlik Sense-arbetskatalogen.

Argument

Argument	Beskrivning
path	En text som kan tolkas som sökvägen till data-filen. Sökvägen är sökvägen till filen, antingen: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense-applikationens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*

```
DIRECTORY C:\userfiles\data; // OR -> DIRECTORY data\

LOAD * FROM
[data1.csv] // ONLY THE FILE NAME CAN BE SPECIFIED HERE (WITHOUT THE FULL PATH)
(ansi, txt, delimiter is ',', embedded labels);

LOAD * FROM
[data2.txt] // ONLY THE FILE NAME CAN BE SPECIFIED HERE UNTIL A NEW DIRECTORY STATEMENT IS
MADE
(ansi, txt, delimiter is '\t', embedded labels);
```

Disconnect

Disconnect-satsen avslutar den aktuella ODBC-kopplingen/OLE DB-kopplingen/anpassade kopplingen. Denna sats är valfri.

Syntax:

```
Disconnect
```

Kopplingen bryts automatiskt när en ny **connect**-sats exekveras eller när skriptexekveringen är avslutad.

Exempel:

```
Disconnect;
```

Drop

Nyckelordet **Drop** skript kan användas för att släppa innehåll under skriptkörning. Följande **Drop** satser är tillgängliga:

- Med **Drop field** och **Drop table** kan du släppa tabeller eller fält från minnet.
- Med **Drop variable** kan du släppa skriptvariabler.

Drop field

Ett eller flera Qlik Sense-fält kan avlägsnas från datamodellen, och därigenom från minnet, när som helst under skriptexekveringen med hjälp av en **drop field**-sats. Egenskapen "distinct" för en tabell tas bort efter en **drop field**-sats.

> *Både **drop field** och **drop fields** är tillåtna och har samma betydelse. Om ingen tabell har angetts avlägsnas fältet från alla tabeller där det förekommer.*

Syntax:

```
Drop field fieldname { , fieldname2 ...} [from tablename1 { , tablename2
...}]
```

```
Drop fields fieldname { , fieldname2 ...} [from tablename1 { , tablename2
...}]
```

Exempel:

```
Drop field A;

Drop fields A,B;

Drop field A from X;

Drop fields A,B from X,Y;
```

Drop table

En eller flera interna Qlik Sense-tabeller kan tas bort från datamodellen och därmed också från minnet, när som helst vid skriptkörning med hjälp av en **drop table**-sats. Annars kan du använda **drop mapping table**-varianten för att släppa mappningstabeller – de som skapats med **Mapping**-prefixet – före slutet av skriptkörningen.

```
drop [mapping] table  tablename {, tablename2 ...}
drop [mapping] tables tablename {, tablename2 ...}
```

> *Formerrna **drop table** och **drop tables** accepteras båda.*

Följande objekt kommer att gå förlorade som ett resultat av detta:

- Själva tabellen/tabellerna.
- Alla fält som inte ingår i återstående tabeller.
- Fältvärden i återstående fält, som uteslutande kom från den/de borttagna tabellen/tabellerna.

Exempel och resultat:

Exempel	Resultat
`drop table Orders, Salesmen, T456a;`	Den här raden resulterar i att tre tabeller tas bort från minnet.

`Tab1:`
`Load * Inline [`
`Customer, Items, UnitPrice`
`Bob, 5, 1.50`
`];`
 	
`Tab2:`
`LOAD Customer, Sum( Items * UnitPrice ) as Sales`
`resident Tab1`
`group by Customer;`
 	
`drop table Tab1;`
	När tabellen *Tab2* har skapats tas tabellen *Tab1* bort.

Exempel	Resultat

`// Load mapping table of country codes:`
`map1:`
`mapping LOAD *`
`Inline [`
`CCode, Country`
`Sw, Sweden`
`Dk, Denmark`
`No, Norway`
`] ;`
`// Load list of salesmen, mapping country code to country`
`// If the country code is not in the mapping table, put Rest of the world`
`Salespersons:`
`LOAD *,`
`ApplyMap('map1', CCode,'Rest of the world') As Country`
`Inline [`
`CCode, Salesperson`
`Sw, John`
`Sw, Mary`
`Sw, Per`
`Dk, Preben`
`Dk, Olle`
`No, Ole`
`Sf, Risttu] ;`
`// We don't need the CCode anymore`
`Drop Field 'CCode';`
 	
`Drop Mapping Table map1;`
 	
`Metrics:`
 	
`Load * Inline [`
`Transaction ID, Sales Amount, Sales Cost, City, Customer ID`
`1000012, 4509.33, 1234.22, Oslo, 250v9849`
`1000013, 1043.21, 180.23, Gothenburg, 195d5930`
`1000014, 6038.35, 1400.10, Copenhagen, 195d5930`
 	
`];`
	Det här exemplet visar hur varianten **Drop mapping table** kan användas. En mappningstabell *map1* skapas och används sedan i en efterföljande **LOAD**-sats med funktionen **ApplyMap**. När mappningstabellen har använts tas den bort från minnet, vilket optimerar RAM-användningen för de data som ska laddas i efterföljande tabeller under skriptkörningen.
	
	I det här fallet, med endast ett fåtal rader i mappningstabellen, skulle RAM-optimeringen vara minimal. För stora mappningstabeller eller när mappningstabeller skapas i loopar kan dock prestandan förbättras avsevärt.

Drop variable

Använd **Drop variable**-satsen för att släppa en skriptvariabel under skriptkörning. När en variabel släpps är den inte tillgänglig eller synlig, till exempel i arkvyn.

*Både **drop variable** och **drop variables** är tillåtna och har samma betydelse.*

Syntax:

```
Drop variable variablename {, variablename2 ...}
```

```
Drop variables variablename {, variablename2 ...}
```

Exempel:

```
Drop variable MyVarToDrop1;

Drop variables MyVarToDrop2, MyVarToDrop3;
```

När man ska använda en **Drop variable**-sats

Drop variable-satser är användbara för att separera skriptutveckling från arkutveckling. Du kan använda en variabel för att utföra beräkningar eller för att endast ladda specifik data, utan att lägga till oreda i arkbyggnadsupplevelsen. Om du har många variabler och en stor datavolym att läsa in i skriptet, kan det hjälpa till att minska minnesanvändningen och öka prestandan att släppa variabler tidigt i skriptet.

Drop table

En eller flera interna Qlik Sense-tabeller kan tas bort från datamodellen och därmed också från minnet, när som helst vid skriptkörning med hjälp av en **drop table**-sats. Annars kan du använda **drop mapping table**-varianten för att släppa mappningstabeller – de som skapats med **Mapping**-prefixet – före slutet av skriptkörningen.

```
drop [mapping] table  tablename {, tablename2 ...}
drop [mapping] tables tablename {, tablename2 ...}
```

*Formerrna **drop table** och **drop tables** accepteras båda.*

Följande objekt kommer att gå förlorade som ett resultat av detta:

- Själva tabellen/tabellerna.
- Alla fält som inte ingår i återstående tabeller.
- Fältvärden i återstående fält, som uteslutande kom från den/de borttagna tabellen/tabellerna.

Exempel och resultat:

Exempel	Resultat
`drop table Orders, Salesmen, T456a;`	Den här raden resulterar i att tre tabeller tas bort från minnet.

Exempel	Resultat
Tab1: Load * Inline [Customer, Items, UnitPrice Bob, 5, 1.50]; Tab2: LOAD Customer, Sum(Items * UnitPrice) as Sales resident Tab1 group by Customer; drop table Tab1;	När tabellen *Tab2* har skapats tas tabellen *Tab1* bort.

// Load mapping table of country codes:
map1:
mapping LOAD *
Inline [
CCode, Country
Sw, Sweden
Dk, Denmark
No, Norway
] ;
// Load list of salesmen, mapping country code to country
// If the country code is not in the mapping table, put Rest of the world
Salespersons:
LOAD *,
ApplyMap('map1', CCode,'Rest of the world') As Country
Inline [
CCode, Salesperson
Sw, John
Sw, Mary
Sw, Per
Dk, Preben
Dk, Olle
No, Ole
Sf, Risttu] ;
// We don't need the CCode anymore
Drop Field 'CCode';
 	
Drop Mapping Table map1;
 	
Metrics:
 	
Load * Inline [
Transaction ID, Sales Amount, Sales Cost, City, Customer ID
1000012, 4509.33, 1234.22, Oslo, 250v9849
1000013, 1043.21, 180.23, Gothenburg, 195d5930
1000014, 6038.35, 1400.10, Copenhagen, 195d5930
 	
];
	Det här exemplet visar hur varianten **Drop mapping table** kan användas. En mappningstabell *map1* skapas och används sedan i en efterföljande **LOAD**-sats med funktionen **ApplyMap**. När mappningstabellen har använts tas den bort från minnet, vilket optimerar RAM-användningen för de data som ska laddas i efterföljande tabeller under skriptkörningen.
	
	I det här fallet, med endast ett fåtal rader i mappningstabellen, skulle RAM-optimeringen vara minimal. För stora mappningstabeller eller när mappningstabeller skapas i loopar kan dock prestandan förbättras avsevärt.

Execute

Execute-satsen använda för att köra andra program medan Qlik Sense laddar data, exempelvis för att göra de konverteringar som krävs.

Den här funktionen är inte tillgänglig i Qlik Sense SaaS.

Satsen stöds inte i standardläget.

```
execute commandline
```

Argument

Argument	Beskrivning
commandline	En text som kan tolkas som en kommandorad av operativsystemet. Du kan hänvisa till en absolut filsökväg eller en lib:// katalogsökväg.

Om du vill använda **Execute** måste följande villkor vara uppfyllda:

- Du måste köra i bakåtkompatibelt läge (tillämpligt för Qlik Sense och Qlik Sense Desktop).
- Du måste ställa in OverrideScriptSecurity till 1 i *Settings.ini* (gäller för Qlik Sense). *Settings.ini* finns i *C:\ProgramData\Qlik\Sense\Engine* och är vanligtvis en tom fil.

*Om du ställer in OverrideScriptSecurity för att aktivera **Execute**, kan alla användare exekvera filer på servern. En användare kan exempelvis bifoga en exekverbar fil till en applikation, och sedan exekvera filen i dataladdningsskriptet.*

Gör följande:

1. Gör en kopia av *Settings.ini* och öppna den i en textredigerare.
2. Kontrollera att filen innehåller *[Settings 7]* på första raden.
3. Infoga en ny rad och skriv *OverrideScriptSecurity=1*.
4. Lägg till en ny rad i slutet av filen.
5. Spara filen.
6. Byt ut *Settings.ini* mot den redigerade filen.
7. Starta om Qlik Sense Engine Service (QES).

Om Qlik Sense körs som en tjänst kan en del kommandon ha ett annat beteende än väntat.

```
Execute C:\Program Files\Office12\Excel.exe;
Execute lib://win\notepad.exe // win is a folder connection referring to c:\windows
```

Field/Fields

De **Field** och **Fields** skriptnyckelorden används i **Declare**, **Derive**, **Drop**, **Comment**, **Rename** och **Tag/Untag**-satser.

FlushLog

Satsen **FlushLog** tvingar Qlik Sense att skriva innehållet i skriptbufferten till skriptets loggfil.

Syntax:

```
FlushLog
```

Innehållet i bufferten skrivs till loggfilen. Detta kommando kan vara användbart för felsökning eftersom du får data som annars hade kunnat gå förlorade när en skriptexekvering inte lyckas.

Exempel:

```
FlushLog;
```

Force

force-satsen tvingar Qlik Sense att tolka fältvärden i efterföljande **LOAD**- och **SELECT**-satser som att de är skrivna med enbart versaler, enbart gemener, alltid med inledande versaler eller som de uppträder (blandat). Med hjälp av denna sats är det möjligt att koppla fält från tabeller med olika konventioner.

force-satsen kan också ändra fältnamn under en laddning eller ett urval med följande datakällor:

- QVD
- CSV (textfiler)
- XLS
- QVX (filer och ODBC-kopplingar)

force-satsen byter bara fältnamn om data laddas i kompakt läge (laddas med *).

Fältnamnen i följande datakällor påverkas inte av **force**-satser.

- JSON
- Parquet
- XML
- XLSX

Syntax:

```
Force ( capitalization | case upper | case lower | case mixed )
```

Standard är force case mixed. Konventionen som specificeras genom denna sats används fram tills en ny force-sats påträffas.

I behörighetssektionen har **force**-satsen ingen effekt: ingen skillnad görs här mellan versaler och gemener vid laddning av fälten.

Exempel och resultat

Exempel	Resultat

Det här exemplet visar hur du tvingar fram inledande versal.
`FORCE Capitalization;`
`Capitalization:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Tabellen **Capitalization** innehåller följande värden:
	`Ab`
	`Cd`
	`Ef`
	`Gh`
	Alla värden inleds med versal.

Det här exemplet visar hur du tvingar fram versaler.
`FORCE Case Upper;`
`CaseUpper:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	CaseUpper-tabellen innehåller följande värden:
	`AB`
	`CD`
	`EF`
	`GH`
	Alla värden är versala.

Exempel	Resultat

Det här exemplet visar hur du tvingar fram gemener.
`FORCE Case Lower;`
`CaseLower:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	CaseLower-tabellen innehåller följande värden:
	`ab`
	`cd`
	`ef`
	`gh`
	Alla värden är gemena.

Det här exemplet visar hur du tvingar fram både gemener och versaler.
`FORCE Case Mixed;`
`CaseMixed:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	CaseMixed-tabellen innehåller följande värden:
	`ab`
	`Cd`
	`eF`
	`GH`
	Alla värden ser ut som de visas i skriptet.

Se även:

From

Skriptnyckelordet **From** används i **Load**-satserna för att hänvisa till en fil och i **Select**-satserna för att hänvisa till en databastabell eller -vy.

Load

LOAD-satsen laddar fält från en fil, direkt från data i skriptet, från en tidigare inläst tabell, från en webbsida, från resultatet av en efterföljande **SELECT**-sats eller genom att generera data automatiskt. Det går även att ladda data från analytiska kopplingar.

```
LOAD [ distinct ] fieldlist
[( from file [ format-spec ] |
from_field fieldassource [format-spec]|
inline data [ format-spec ] |
resident table-label |
autogenerate size ) |extension pluginname.functionname([script]
tabledescription)]
[ where criterion | while criterion ]
[ group by groupbyfieldlist ]
[order by orderbyfieldlist ]
```

Argument

Argument	Beskrivning
distinct	Du kan använda **distinct** som predikat om du bara vill ladda unika poster. Om det finns dubblettposter laddas den första. Om du använder föregående load måste du placera **distinct** i den första load-satsen, eftersom **distinct** bara inverkar på destinationstabellen.

Argument	Beskrivning
fieldlist	*fieldlist ::= (* | field{, * | field })* En lista på de fält som ska läsas in. Genom att använda * som fältlista anger man alla fält i tabellen. *field ::= (fieldref | expression) [***as** *aliasname]* Fältdefinitionen måste alltid innehålla en litteral, en referens till ett befintligt fält eller ett uttryck. *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [**I** | **U** | **R** | **B** | **T**])* *fieldname* är en text som är identisk med fältnamnet i tabellen. Observera att fältnamnet måste omslutas av raka, dubbla citationstecken eller av hakparenteser om det innehåller exempelvis mellanslag. Ibland är fältnamn inte explicit tillgängliga. Då används en annan metod: *@fieldnumber* representerar fältnumret i en avgränsad tabellfil. Det måste vara ett positivt heltal som föregås av "@". Numreringen går alltid från 1 och upp till antalet fält. *@startpos*:*endpos* representerar första och sista teckenposition för fältet i en fixpostfil med fasta fältpositioner. Positionerna måste vara positiva heltal. De två talen måste föregås av "@" och avgränsas av kolon. Numreringen går alltid från 1 och upp till antalet positioner. I det sista fältet används **n** som slutposition. • Om*@startpos*:*endpos* omedelbart följs av tecknen **I** eller**U**, tolkas inlästa bytes som binärt signerade (**I**) eller osignerade (**U**) heltal (Intel byte order). Antalet positioner som läses måste vara 1, 2 eller 4. • Om *@startpos*:*endpos* omedelbart följs av tecknet **R**, tolkas inlästa bytes som ett binärt reellt tal (IEEE 32-bitars eller 64-bitars floating point). Antalet positioner som läses in måste vara 4 eller 8. • Om *@startpos*:*endpos* omedelbart följs av tecknet **B**, tolkas inlästa bytes som BCD (Binary Coded Decimal)-tal enligt COMP-3-standard. Valfritt antal bytes kan anges. *expression* kan vara en numerisk funktion eller en strängfunktion baserad på ett eller flera andra fält i samma tabell. För ytterligare information, se uttryckens syntax. **as** används för att döpa om fält.

Argument	Beskrivning
from	**from** används om data ska läsas in från en fil med en mapp eller webbfilsdatakoppling *file ::= [path] filename* **Exempel:** ***'lib://Table Files/'*** Om sökvägen utelämnas söker Qlik Sense efter filen i den mapp som specificerats av en **Directory**-sats. Om det inte finns någon **Directory**-sats söker Qlik Sense i arbetskatalogen *C:\Users\{user}\Documents\Qlik\Sense\Apps*. *I en Qlik Sense-serverinstallation anges arbetskatalogen i Qlik Sense Repository Service. Som standard är det C:\ProgramData\Qlik\Sense\Apps.* *filename* kan innehålla standardiserade jokertecken från DOS (* och ?). Detta får alla matchande filer i den angivna katalogen att läsas in. *format-spec ::= (fspec-item { , fspec-item })* Formatspecifikationen består av en lista med flera formatspecifikatorer inom parentes. **Bakåtkompatibel skriptkod** I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** ***c:\data*** • relativ till Qlik Sense-applikationens arbetskatalog. **Exempel:** ***data*** • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** ***http://www.qlik.com***

Argument	Beskrivning
from_field	**from_field** används om data ska läsas in från ett tidigare inläst fält. *fieldassource::=*(*tablename, fieldname)* Fältet är namnet på tidigare inlästa *tablename* och *fieldname*. *format-spec ::= (fspec-item {, fspec-item })* Formatspecifikationen består av en lista med flera formatspecifikatorer inom parentes. Mer information finns i *Formatspecifikatorer (page 191)*. ***from_field** har bara stöd för komman som listavgränsare vid separering av fält i tabeller.*
inline	**inline** används om data ska skrivas i skriptet, och inte läsas in från en fil. *data ::= [text]* Data som läses in med en **inline**-sats måste inledas och avslutas med specifika tecken – hakparenteser, citationstecken eller backticks. Texten mellan dessa tolkas som om den vore skriven i en fil. Precis som du infogar en ny rad i en textfil bör du göra det även i texten i en **inline**-sats. Klicka på vanligt sätt på returtangenten när du skriver skriptet. I en enkel inline-laddning definieras antalet kolumner av den första raden. *format-spec ::= (fspec-item {, fspec-item })* Du kan anpassa inline-laddningen med många av samma formatspecifikationselement som kan användas för andra laddade tabeller. De här elementen listas inom parentes. Mer information finns i *Formatspecifikatorer (page 191)*. För mer information om inline-laddningar, se Använda inline-laddningar för att ladda data.
resident	**resident** används om data ska läsas in från en tidigare inläst tabell. *table label* är en etikett som föregår de **LOAD**- eller **SELECT**-satser som skapade den ursprungliga tabellen. Etiketten ska avslutas med kolon.
autogenerate	**autogenerate** används när data ska genereras automatiskt av Qlik Sense. *size ::= number* *Number* är ett heltal som anger antalet poster som ska genereras. Fältlistan får inte innehålla uttryck som behöver data från en extern datakälla eller från en tidigare laddad tabell, om du inte hänvisar till ett enda fältvärde i en tidigare laddad tabell med hjälp av **Peek**-funktionen.

Argument	Beskrivning
extension	Du kan ladda data från analytiska kopplingar. Du måste använda **extension**-satsen för att anropa en funktion som definierats i komplement på serversidan, (SSE) -instickspprogrammet, eller utvärdera ett skript. Du kan skicka en enstaka tabell till SSE-instickspprogrammet. En datatabell returneras. Fälten kallas Field1, Field2 och så vidare om instickspprogrammet inte specificerar namnen på de fält som returneras. `Extension pluginname.functionname( tabledescription );` • Ladda data med en funktion i ett SSE-instickspprogram *tabledescription ::= (table { ,tablefield})* Om du inte anger tabellfält används fälten i laddningsordning. • Ladda data genom att utvärdera ett skript i ett SSE-instickspprogram *tabledescription ::= (script, table { ,tablefield})* **Hantering av datatyp i tabellfältdefinitionen** Datatyper identifieras automatiskt i analytiska kopplingar. Om data inte har numeriska värden och minst en icke-NULL textsträng betraktas fältet som text. I alla andra fall betraktas det som numeriskt. Du kan tvinga datatypen genom att radbryta ett fältnamn med **String()** eller **Mixed()**. • **String()** tvingar fältet att vara text. Om fältet är numeriskt extraheras textdelen av det duala värdet. Ingen konvertering görs. • **Mixed()** tvingar fältet att vara dualt. **String()** eller **Mixed()** kan inte användas utanför **komplement**-tabellfältdefinitioner och du kan inte använda andra Qlik Sense-funktioner i en tabellfältdefinition. **Mer om analytiska kopplingar** Du måste konfigurera analytiska kopplingar innan du kan använda dem.
where	**where** är ett tillägg som används för att tala om huruvida en post ska inkluderas i valet eller inte. Valet inkluderas om *criterion* är True. *criterion* är ett logiskt uttryck.
while	**while** är en sats som används för att tala om när en post ska läsas in upprepade gånger. Samma post läses in så länge *criterion* är True. För att vara användbar måste en **while**-sats typiskt sett innehålla en **IterNo( )**-funktion. *criterion* är ett logiskt uttryck.

Argument	Beskrivning
group by	**group by** används för att definiera över vilka fält data ska aggregeras (grupperas). Aggregeringsfälten ska på något sätt inlemmas i de uttryck som läses in. Inga andra fält än aggregeringsfälten får användas utanför aggregeringsfunktionerna i inläsningsuttrycken. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	**order by** är en sats som används för att sortera poster i en resident tabell innan de bearbetas av **load**-satsen. Den residenta tabellen kan sorteras efter ett eller flera fält, i stigande eller sjunkande ordning. Sortering görs i första hand efter talvärde, i andra hand efter nationell sorteringsspråkvariant. Satsen är endast användbar när datakällan är en resident tabell. Ordningsföljdsfälten anger efter vilket fält resident-tabellen sorteras. Ange fältets namn eller dess nummer i resident-tabellen (det första fältet får nummer 1). orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } *sortorder* är antingen *asc* för stigande eller *desc* för fallande. Om ingen *sortorder* anges, antas *asc*. *fieldname*, *path*, *filename* och *aliasname* är textsträngar som representerar det namnen antyder. Valfritt fält i källtabellen kan användas som *fieldname*. Däremot hamnar fält som har skapats med hjälp av as-satsen (*aliasname*) utanför och kan inte användas inuti samma **load**-sats.

Om ingen datakälla anges genom en **from**, **inline**, **resident**, **from_field, komplement** eller **autogenerate**-sats kommer data att läsas in från resultatet av närmast efterföljande **SELECT**- eller **LOAD**-sats. Den efterföljande satsen får inte ha något prefix.

Läsa in olika filformat

Ladda en avgränsad data fil utan standardalternativ:

```
LOAD * from data1.csv;
```

Ladda en avgränsad datafil från en bibliotekskoppling (DataFiles):

```
LOAD * from 'lib://DataFiles/data1.csv';
```

Ladda alla avgränsade datafiler från en bibliotekskoppling (DataFiles):

```
LOAD * from 'lib://DataFiles/*.csv';
```

Ladda en avgränsad fil, med komma som angiven avgränsare och inbäddade etiketter:

```
LOAD * from 'c:\userfiles\data1.csv' (ansi, txt, delimiter is ',', embedded labels);
```

Ladda en avgränsad fil, med tabb som angiven avgränsare och inbäddade etiketter:

```
LOAD * from 'c:\userfiles\data2.txt' (ansi, txt, delimiter is '\t', embedded labels);
```

Ladda en dif-fil med inbäddade rubriker:

```
LOAD * from file2.dif (ansi, dif, embedded labels);
```

Ladda tre fält från en fil med fasta poster utan rubriker:

```
LOAD @1:2 as ID, @3:25 as Name, @57:80 as City from data4.fix (ansi, fix, no labels, header is
0, record is 80);
```

Ladda en QVX-fil som specificerar en absolut sökväg:

```
LOAD * from C:\qdssamples\xyz.qvx (qvx);
```

Ladda webbfiler

Ladda från en standard-URL som anges i webbfildatakopplingen:

```
LOAD * from [lib://MyWebFile];
```

Ladda från en specifik URL och åsidosätt URL:en som anges i webbfildatakopplingen:

```
LOAD * from [lib://MyWebFile] (URL is 'http://localhost:8000/foo.bar');
```

Ladda från en specifik URL som anges i en variabel med dollarteckenexpansion:

```
SET dynamicURL = 'http://localhost/foo.bar';
LOAD * from [lib://MyWebFile] (URL is '$(dynamicURL)');
```

Välja vissa fält, döpa om och beräkna fält

Ladda endast tre specifika fält från en avgränsad fil:

```
LOAD FirstName, LastName, Number from data1.csv;
```

Döp om fet första fältet som A och det andra fältet som B när du laddar en fil utan etiketter:

```
LOAD @1 as A, @2 as B from data3.txt (ansi, txt, delimiter is '\t', no labels);
```

Ladda Name som en konkatenering av FirstName, ett blanksteg och LastName:

```
LOAD FirstName&' '&LastName as Name from data1.csv;
```

Ladda Quantity, Price och Value (produkten av Quantity och Price):

```
LOAD Quantity, Price, Quantity*Price as Value from data1.csv;
```

Läsa in vissa poster

Ladda endast unika poster, dubblettposter kommer att uteslutas:

```
LOAD distinct FirstName, LastName, Number from data1.csv;
```

Ladda endast poster där fältet Litres har ett värde över noll:

```
LOAD * from Consumption.csv where Litres>0;
```

Ladda data som inte finns i filer samt automatiskt genererade data

Ladda en tabell med inline-data, två fält som kallas CatID och Category:

```
LOAD * Inline
[CatID, Category
0,Regular
1,Occasional
2,Permanent];
```

Ladda en tabell med inline-data, tre fält som kallas UserID, Password och Access:

```
LOAD * Inline [UserID, Password, Access
A, ABC456, User
B, VIP789, Admin];
```

Ladda en tabell med 10 000 rader. Fält A kommer innehålla talet för posten som ska laddas (1,2,3,4,5...) och fältet B innehåller ett slumpmässigt nummer mellan 0 och 1:

```
LOAD RecNo( ) as A, rand( ) as B autogenerate(10000);
```

Parentesen efter autogenerate är tillåten men inte obligatorisk.

Läsa in data från en tidigare inläst tabell

Först laddar vi en avgränsad tabellfil och kallar dentab1:

```
tab1:
SELECT A,B,C,D from 'lib://DataFiles/data1.csv';
```

Ladda fält från den redan laddade tabellen tab1 som tab2:

```
tab2:
LOAD A,B,month(C),A*B+D as E resident tab1;
```

Ladda fält från den redan laddade tabellen tab1 men bara poster där A är större än B:

```
tab3:
LOAD A,A+B+C resident tab1 where A>B;
```

Ladda fält som den redan laddade tabellentab1 ordnade efter A:

```
LOAD A,B*C as E resident tab1 order by A;
```

Ladda fält från den redan laddade tabellen tab1, ordnade efter det första fältet, sedan efter det andra fältet:

```
LOAD A,B*C as E resident tab1 order by 1,2;
```

Ladda fält från den redan laddade tabellentab1 ordnade efter C i fallande ordning, sedan efter B i stigande ordning och därefter efter det första fältet i fallande ordning:

```
LOAD A,B*C as E resident tab1 order by C desc, B asc, 1 desc;
```

Läsa in data från tidigare inlästa fält

Ladda fältet Types från den tidigare laddade tabellen Characters som A:

```
LOAD A from_field (Characters, Types);
```

Laddar data från en efterföljande tabell (föregående laddning)

Ladda A, B och de beräknade fälten X och Y från Table1 som laddas i efterföljande **SELECT**-sats:

```
LOAD A, B, if(C>0,'positive','negative') as X, weekday(D) as Y;
SELECT A,B,C,D from Table1;
```

Gruppering av data

Ladda fält grupperade (aggregerade) efter ArtNo:

```
LOAD ArtNo, round(Sum(TransAmount),0.05) as ArtNoTotal from table.csv group by ArtNo;
```

Ladda fält grupperade (aggregerade) efter Week och ArtNo:

```
LOAD Week, ArtNo, round(Avg(TransAmount),0.05) as WeekArtNoAverages from table.csv group by
Week, ArtNo;
```

Upprepad inläsning av en post

I detta exempel har vi indatafilen Grades.csv som innehåller betygen för varje student, kondenserat till ett fält:

```
Student,Grades
Mike,5234
```

```
John,3345
Pete,1234
Paul,3352
```

Betygen, i en skala från 1 till 5, motsvarar Math, English, Science ochHistory. Vi kan dela upp betygen i separata värden genom att läsa in varje post flera gånger med en **while**-sats och använda **IterNo( )**-funktionen som räknare. Vid varje inläsning extraheras betyget med funktionen **Mid** och lagras i Grade och ämnet väljs med hjälp av funktionen **pick** och lagras i Subject. Den slutliga **while**-satsen innehåller testet för att kontrollera om alla betyg har lästs (fyra per student i det här fallet), vilket innebär att nästa studentpost ska läsas.

```
MyTab:
LOAD Student,
mid(Grades,IterNo( ),1) as Grade,
pick(IterNo( ), 'Math', 'English', 'Science', 'History') as Subject from Grades.csv
while IsNum(mid(Grades,IterNo(),1));
```

Resultatet blir en tabell med dessa data:

Student	Subject	Grade
John	English	3
John	History	5
John	Math	3
John	Science	4
Mike	English	2
Mike	History	4
Mike	Math	5
Mike	Science	3
Paul	English	3
Paul	History	2
Paul	Math	3
Paul	Science	5
Pete	English	2
Pete	History	4
Pete	Math	1
Pete	Science	3

Ladda från analytiska kopplingar

Följande exempeldata används.

```
Values:
Load
 Rand() as A,
 Rand() as B,
 Rand() as C
AutoGenerate(50);
```

Ladda data med en funktion

I dessa exempel antar vi att vi har ett instickspro gram med analytisk koppling som heter *P* som innehåller den anpassade funktionen *Calculate(Parameter1, Parameter2)*. Funktionen returnerar tabellen *Results* som innehåller fälten *Field1* och*Field2*.

```
Load * Extension P.Calculate( Values{A, C} );
```

Ladda alla fält som returneras när fälten A och C skickas till funktionen.

```
Load Field1 Extension P.Calculate( Values{A, C} );
```
Ladda bara Field1-fältet när fälten A och C skickas till funktionen.

```
Load * Extension P.Calculate( Values );
```
Ladda alla fält som returneras när fälten A och B skickas till funktionen. Eftersom fält inte är angivna används A och B eftersom de står först i tabellordningen.

```
Load * Extension P.Calculate( Values {C, C});
```
Ladda alla fält som returneras när fältet C skickas till funktionens båda parametrar.

```
Load * Extension P.Calculate( Values {String(A), Mixed(B)});
```
Ladda alla fält som returneras när fält A skickas tvingad till sträng och fält B skickas tvingad till numerisk till funktionen.

Ladda data genom att utvärdera ett skript

```
Load A as A_echo, B as B_echo Extension R.ScriptEval( 'q;', Values{A, B} );
```
Ladda tabellen som returneras av skriptet q när värdena av A och B skickas.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{A, B} );
```
Ladda tabellen som returneras av skriptet som lagras i My_R_Script-variabeln när värdena av A och B skickas.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{B as D, *} );
```
Ladda tabellen som returneras av skriptet som lagras i My_R_Script-variabeln när värdena av B som döpts om till D, A och C skickas. Använd * för att skicka återstående orefererade fält.

Filtillägget för DataFiles-kopplingar är skiftlägeskänsligt. Till exempel: .qvd.

Formatspecifikatorer

Varje formatspecifikator definierar en viss egenskap i tabellfilen:

fspec-item ::= [**ansi | oem | mac | UTF-8 | Unicode | txt | fix | dif | biff | ooxml | html | xml | kml | qvd | qvx | parquet | json | delimiter is** `char` | **no eof | embedded labels | explicit labels | no labels | table is [**`tablename`**]** | **header is** `n` | **header is** `line` | **header is** `n` **lines** | **comment is** `string` | **record is** `n` | **record is** `line` | **record is** `n` **lines** | **no quotes |msq** | **URL is** `string` | **userAgent is** `string`]

Teckenuppsättning

Teckenuppsättning är en filspecifikator för **LOAD**-satsen som anger vilken teckenuppsättning som används i filen.

Specifikatorerna **ansi**, **oem** och **mac** användes i QlikView och fungerar fortfarande. De kommer dock inte att genereras när du skapar **LOAD**-satsen i Qlik Sense.

Syntax:

```
utf8 | unicode | ansi | oem | mac | codepage is
```

Argument:

Argument

Argument	Beskrivning
utf8	UTF-8-teckenuppsättning
unicode	Unicode-teckenuppsättning
ansi	Windows, kodsida 1252
oem	DOS, OS/2, AS400 och andra
mac	Kodsida 10000
codepage is	Med specifikatorn **codepage** går det att använda valfri Windows kodsida som *N*.

Begränsningar:

Konvertering från **oem**-teckenuppsättningen är inte implementerad för macOS. Om ingenting anges, antas kodsida 1252 under Windows.

Exempel:

```
LOAD * from a.txt (utf8, txt, delimiter is ',' , embedded labels)

LOAD * from a.txt (unicode, txt, delimiter is ',' , embedded labels)

LOAD * from a.txt (codepage is 10000, txt, delimiter is ',' , no labels)
```

Se även:

p *Load (page 182)*

Tabellformat

Tabellformatet är en filspecifikator för **LOAD**-satsen som anger vilken filtyp det gäller. Om inget annat är angivet antas standardformatet *.txt*.

Tabellformatstyper

Typ	Beskrivning
txt	I en avgränsad textfil är kolumnerna avgränsade av ett avgränsartecken.

Typ	Beskrivning
fix	I en textfil med fasta postlängder har varje kolumn en bredd på ett visst antal tecken. Vanligtvis innehåller filer med fasta postlängder poster som avgränsas av en radmatning, men det finns mer avancerade alternativ för att ange poststorlek i byte eller för att låta dem sträcka sig över mer än en rad med hjälp av **Record is**. *Om data innehåller tecken på flera byte kan fältbrytningarna bli feljusterade eftersom formatet baserar på en fast längd i byte.*
dif	I en *.dif*-fil (Data Interchange Format) används ett speciellt format för att definiera tabellen.
biff	Qlik Sense kan även tolka data i standard-Excel-filer med hjälp av *biff*-formatet (Binary Interchange File Format).
ooxml	Excel 2007 och senare versioner använder ooxml *.xslx*-formatet. **Tabell är**-specifikatorn kan användas för att definiera arknamnet som ska laddas som en tabell. *Tabell är (page 197)*
html	Om tabellen ingår i en html-sida eller fil bör html användas.
xml	xml (Extensible Markup Language) är ett vanligt märkspråk som används för att representera datastrukturer i textformat. **Tabell är**-specifikatorn kan användas för att definiera sökvägen till den XML som ska laddas som en tabell. *Tabell är (page 197)*
qvd	Formatet *qvd* är det egna filformatet QVD, exporterat från en Qlik Sense-applikation.
qvx	*qvx* är ett filformat/streamingformat för att leverera utdata med hög prestanda till Qlik Sense.
parquet	Apache Parquet är ett kolumnformat som är mycket effektivt för lagring och sökning av stora datauppsättningar. Med Parquet-filer som innehåller nästlade data kan du ange tabellen från Parquet-filen för att ladda med **Table is**-specifikatorn. Till exempel: `LOAD * FROM [lib://DataFiles/company.parquet] (parquet, table is [company:salesrep.salesrep]);`. *Tabell är (page 197)*

Typ	Beskrivning
json	JSON (JavaScript Objekt Notation) är ett språkoberoende dataformat. Om JSON innehåller flera matriser kan specificeraren **Table is** användas för att definiera sökvägen till iteratorn för undertabellen. *Tabell är (page 197)*

Delimiter is

För avgränsade tabellfiler kan en valfri avgränsare anges med hjälp av **delimiter is**-specifikator. Denna specifikator är endast relevant för avgränsade .txt-filer.

Syntax:

```
delimiter is char
```

Argument:

Argument

Argument	Beskrivning
char	Anger ett enstaka tecken av 127 ASCII-tecknen.

Dessutom kan följande värden användas:

Valfria värden

Värde	Beskrivning
'\t'	motsvarar ett tabbtecken, med eller utan citattecken.
'\\'	motsvarar ett omvänt snedstreck (\).
'spaces'	motsvarar alla kombinationer av ett eller flera mellanslag. Tecken som inte kan skrivas ut och har ett ASCII-värde under 32, undantaget CR och LF, tolkas som mellanslag.

Om inget anges, antas **delimiter is ','**.

Exempel:

```
LOAD * from a.txt (utf8, txt, delimiter is ',' , embedded labels);
```

Se även:

p *Load (page 182)*

No eof

Specifikatorn **no eof** används för att ignorera tecknet som betecknar filslutsmarkering när du läser in avgränsade **.txt**-filer.

Syntax:

```
no eof
```

Om no **no eof**-specificeraren används ignoreras tecknen med kodpunkten 26, som annars markerar slutet på filen och kan vara del av ett fältvärde.

Endast relevant för avgränsade textfiler.

Exempel:

```
LOAD * from a.txt (txt, utf8, embedded labels, delimiter is ' ', no eof);
```

Se även:

p *Load (page 182)*

Labels

Labels är en filspecifikator för **LOAD**-satsen som anger var i en fil fältnamnen kan hittas.

Syntax:

```
embedded labels|explicit labels|no labels
```

Fältnamnen kan finnas på olika ställen i filen. Om filens första post innehåller fältnamnen, ska **embedded labels** användas. Om fältnamnen inte finns lagrade i filen, ska **no labels** användas. I *dif*-filer finns ibland en särskild inledning (header, se nedan), skild från datasektionen. I sådana fall bör man använda **explicit labels**. Om ingenting anges antas **embedded labels**, även för *dif*-filer.

Example 1:

```
LOAD * from a.txt (unicode, txt, delimiter is ',' , embedded labels
```

Example 2:

```
LOAD * from a.txt (codePage is 1252, txt, delimiter is ',' , no labels)
```

Se även:

p *Load (page 182)*

Header is

Anger filhuvudets storlek i tabellfiler. En godtycklig inledningslängd kan anges genom specifikatorn **header is**. En sådan inledning är en sektion som inte används i Qlik Sense.

Syntax:

```
header is n
```

```
header is line
```

```
header is n lines
```

Rubriklängden kan anges i bytes (**header is n**), eller i linjer (**header is line** eller **header is n lines**). **n** måste vara ett positivt heltal som representerar rubriklängden. Om inget anges, antas **header is 0**. **header is**-specifikatorn är endast relevant för tabellfiler.

Exempel:

Det här är ett exempel på en datakälla i tabellformat som innehåller en rad med rubriktext som inte ska tolkas som data av Qlik Sense.

```
*Header line
Col1,Col2
a,B
c,D
```

Användning av specifikatorn **header is 1 lines** gör att den första raden inte blir inläst som data. I det här exemplet anger specifikatorn **embedded labels** att Qlik Sense ska tolka den första icke-uteslutna raden som att den innehåller fältetiketter.

```
LOAD Col1, Col2
FROM 'lib://files/header.txt'
(txt, embedded labels, delimiter is ',', msq, header is 1 lines);
```

Resultatet är en tabell med två fält, Col1 and Col2.

Se även:

p *Load (page 182)*

Record is

För fixpostfiler måste postlängden anges med specifikatorn **record is**.

Syntax:

```
Record is n
```

```
Record is line
```

```
Record is n lines
```

Argument:

Argument

Argument	Beskrivning
n	Anger postlängden i bytes.
line	Anger postlängden som en rad.
n lines	Anger postlängden i rader där n är ett positivt heltal som motsvarar postlängden.

Begränsningar:

record is-specifikatorn är endast relevant för **fix**-filer.

Se även:

p *Load (page 182)*

Tabell är

För Excel-, XML-, Parquet- och JSON-filer kan du ange tabellen som du laddar data från i specifikatorn för tabellformat.

Syntax:

```
Table is table name
```

Argument:

Argument

Argument	Beskrivning

table name	Anger tabellens namn. Värdet beror på tabellformatet:
	• Excel: arknamnet.
	• XML: sökvägen som anger den del av XML som ska laddas.
	• Parquet: sökvägen som anger tabellen med formatet *<nod> <nod> <nod>*. Använd **Table is** när du anger en tabell i en nästlad struktur. Om du exempelvis har Parquet-data i följande schema:
	`Schema:`
	`Field(name: "Name", datatype: String),`
	`Field(name: "Age", datatype: Float),`
	`Field(name: "Phone", datatype: List(`
	`Field(name: "Item", datatype: Struct[`
	`Field(name: "Number", datatype: String)`
	Du kan ladda Phone och dess nästlade fält som en tabell med argumentet `Table is [Schema:Phone.Item]`. Då genereras nyckelfältet *%Key_Phone* med tabellen.
	• JSON: i JSON-filer med matriser, sökvägen till iteratorn för undertabellen. Du kan göra en laddning av alla tabeller genom att ange **all tables** istället för **Table is**. Exempel:
	`LOAD * FROM [lib://DataFiles/Sales.json] (json, all tables);`
	Om dina data innehåller flera matriser med identiskt innehåll, använd nyckelordet **DISTINCT** i undertabellens LOAD-satser. Exempel:
	`LOAD DISTINCT * FROM [lib://DataFiles/Sales.json] (json, all tables);`

Exempel: Excel

```
LOAD
	"Item Number",
	"Product Group",
	"Product Line",
```

```
       "Product Sub Group",
       "Product Type"
FROM [lib://AttachedFiles/Item master.xlsx]
(ooxml, embedded labels, table is [Item master]);
```

Exempel: XML

```
LOAD
       city%Table,
       %Key_row_7FAC1F878EC01ECB
FROM [lib://AttachedFiles/cities.xml]
(XmlSimple, table is [root/row/country/city]);
```

Exempel: Parquet

Filen company.parquet innehåller följande schema::

```
company (String)
contact (String)
company:salesrep (List)
       salesrep (Group)
               salesrep (String)
company:headquarter (List)
       headquarter (Group)
               country (String)
               city (String)
               city:region (List)
               region (Group)
                       region (String)
```

Följande laddar innehållet från filen till tabeller. Den första laddningssatsen laddar rotgruppen. Den andra laddningssatsen laddar innehållet i *salesrep*-gruppen som en tabell. Den tredje laddar *headquarter*-gruppen som en tabell. Den fjärde laddar *region*-gruppen som en tabell.

```
LOAD * FROM […] (parquet);
LOAD * FROM […] (parquet, table is [company:salesrep.salesrep]);
LOAD * FROM […] (parquet, table is [company:headquarter.headquarter]
LOAD * FROM […] (parquet, table is [company:headquarter.headquarter.city:region.region]
```

Exempel: JSON

```
LOAD
customers.customerid,
customers.companyname,
customers.contactname,
customers.contacttitle,
customers.address,
customers.city,
customers.postalcode,
customers.country,
customers.phone,
customers.fax,
%Key_customers,
customers.region
FROM [lib://DataFiles/example.json]
(json, table is [/*/customers]);
```

Begränsningar:

Table is-specifikatorn är endast relevant för Excel-, XML-, Parquet, och JSON-filer.

Quotes

Quotes är en filspecifikator för **LOAD**-satsen som anger om citationstecken kan användas och hur citationstecken och avgränsare prioriteras. Gäller endast textfiler.

Syntax:

```
no quotes
```

```
msq
```

Om specifikatorn utelämnas används standardinställningen. Då accepteras både " " och ' ', men bara som första eller sista icke-blankt tecken i ett fältvärde.

Argument:

Argument

Argument	Beskrivning
no quotes	Används om citationstecken inte ska accepteras i en textfil.
msq	Används för att ange Modern Style Quoting, som tillåter flerradigt innehåll i fält. Fält som innehåller radslutstecken måste omslutas av dubbla citattecken. En begränsning av msq-alternativet är att enstaka dubbelcitattecken som visas som första eller sista tecken i fältinnehåll tolkas som början eller slutet på flerradigt innehåll, vilket kan leda till oförutsedda resultat i den inlästa datauppsättningen. I detta fall bör standardcitationstecken användas i stället och specifikatorn bör utelämnas.

XML

Denna skriptspecifikator används när du läser in xml-filer. Giltiga alternativ för **XML**-specifikatorn är listade i syntaxen.

Du kan inte ladda DTD-filer i Qlik Sense.

Syntax:

```
xmlsimple
```

Se även:

p *Load (page 182)*

KML

Denna skriptspecifikator används för att läsa in KML-filer för användning i en kartvisualisering.

Syntax:

```
kml
```

KML-filen kan representera antingen ytdata (till exempel länder eller regioner) som representeras av polygoner, linjedata (till exempel stigar eller vägar) eller datapunkter (till exempel städer eller platser) som representeras av punkter i formatet [long, lat].

URL is

Den här skriptspecifikatorn används för att ange URL:en för en webbfils dataanslutning när den laddar en webbfil.

Syntax:

```
URL is string
```

Argument:

Argument

Argument	Beskrivning
string	Anger URL för den fil som ska laddas. Det åsidosätter den URL som anges i den webbfilskoppling som används.

Begränsningar:

URL is-specifikatorn är endast relevant för webbfiler. Du måste använda en befintlig webbfilsdatakoppling.

Se även:

p *Load (page 182)*

userAgent is

Den här skriptspecifikatorn används för att ange webbläsarens användaragent när den laddar en webbfil.

Syntax:

```
userAgent is string
```

Argument:

Argument

Argument	Beskrivning
string	Anger användaragentsträng för webbläsaren. Den åtsidosätter den standardagentsträngen för webbläsaren "Mozilla/5.0".

Begränsningar:

userAgent is-specifikatorn är endast relevant för webbfiler.

Se även:

p *Load (page 182)*

Let

let-satsen är ett komplement till **set**-satsen som används för att definiera skriptvariabler. I motsats till **set**-satsen utvärderas **let**-satsens uttryck till höger om '=' när skriptet körs, innan det tilldelas variabeln.

Syntax:

```
Let variablename=expression
```

Exempel och resultat:

Exempel	Resultat

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	`$(x)` utvärderas som ' 3+4 '
	
	`$(y)` utvärderas som ' 7 '
	
	`$(z)` utvärderas som ' 8 '
	
	Observera skillnaden mellan **Set**- och **Let**-satserna. **Set**-satsen tilldelar strängen '3+4' till variabeln, medan **Let**-satsen utvärderar strängen och tilldelar variabeln värdet 7.

Let T=now( );	`$(T)` får ett värde som motsvarar aktuell tid.

Loosen Table

En eller flera av Qlik Senses interna datatabeller kan explicit förklaras vara löst kopplade under skriptexekvering med hjälp av satsen **Loosen Table**. När en tabell är löst kopplad tas alla associationer mellan fältvärdena i tabellen bort. Man kan åstadkomma en liknande effekt genom att ladda vart och ett av fälten i den löst kopplade tabellen som fristående, okopplade tabeller. Löst kopplad kan vara användbart under tester för att tillfälligt isolera olika delar av datastrukturen. En löst kopplad tabellen kan identifieras i tabellvyn genom de streckade linjerna. Om en eller flera **Loosen Table**-satser används i skriptet gör detta att Qlik Sense ignorerar alla inställningar för löst kopplade tabeller gjorda innan skriptexekveringen.

Syntax:

```
Loosen Tabletablename [ , tablename2 ...]
```

```
Loosen Tablestablename [ , tablename2 ...]
```

Antingen syntaxen **Loosen Table** eller **Loosen Tables** kan användas.

*Om Qlik Sense hittar cirkelreferenser i datastrukturen som inte kan brytas genom att tabeller interaktivt eller explicit förklaras vara löst kopplade i skriptet, kommer en eller flera ytterligare tabeller att tvingas vara löst kopplade tills inga cirkelreferenser kvarstår. När detta inträffar får man en varning via dialogen **Varning cirkulära referenser**.*

Exempel:

```
Tab1:

SELECT * from Trans;

Loosen Table Tab1;
```

Map

Satsen **map ... using** används för att mappa ett visst fältvärde eller uttryck till värdena i en specifik mappningstabell. Mappningstabellen skapas via satsen **Mapping**.

Syntax:

```
Map fieldlist Using  mapname
```

Automatisk mappning görs för fält som är inlästa efter **Map … Using**-satsen fram till skriptets slut eller tills en **Unmap**-sats påträffas.

Mappningen utförs sist i den kedja av händelser som leder fram till att fältet lagras i den interna Qlik Sense-tabellen. Detta innebär att mappning inte görs varje gång ett fältnamn påträffas som del av ett uttryck, utan när värdet lagras under fältnamnet i den interna tabellen. Om mappning på uttrycksnivå krävs, måste funktionen **Applymap()** användas istället.

Argument:

Argument

Argument	Beskrivning
fieldlist	En kommaavgränsad lista över de fält som ska mappas fr.o.m. den aktuella positionen i skriptet. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken.
mapname	Namnet på en mappningstabell som tidigare lästs in via en **mapping load**- eller **mapping select**-sats.

Exempel och resultat:

Exempel	Resultat
Map Country Using Cmap;	Möjliggör mappning av fältet Country med hjälp av Cmap.
Map A, B, C Using X;	Möjliggör mappning av fälten A, B och C med hjälp av X.
Map * Using GenMap;	Startar mappning av alla fält med hjälp av GenMap.

NullAsNull

Med **NullAsNull**-satsen upphävs konvertering av NULL-värden till strängvärden som dessförinnan har ställts in med **NullAsValue**-satsen.

Syntax:

```
NullAsNull *fieldlist
```

Instruktionen **NullAsValue** fungerar som en omkopplare och kan aktiveras eller inaktiveras flera gånger i skriptet, med hjälp av antingen en **NullAsValue** eller en **NullAsNull** instruktion.

Argument:

Argument

Argument	Beskrivning
*fieldlist	En kommaavgränsad lista över fält för vilka **NullAsNull** ska aktiveras. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken.

Exempel:

```
NullAsNull A,B;

LOAD A,B from x.csv;
```

NullAsValue

Satsen **NullAsValue** anger för vilka fält de NULL-värdena ska konverteras till värden.

Syntax:

```
NullAsValue *fieldlist
```

Qlik Sense betraktar normalt NULL-värden som saknade eller ej angivna enheter. I vissa databasprocesser är det däremot underförstått att NULL-värden ska tolkas som speciella värden och inte som värden som saknas. Det faktum att NULL-värden normalt inte får länka till andra NULL-värden kan upphävas med hjälp av **NullAsValue**-satsen.

NullAsValue-satsen fungerar som växlare och påverkar efterföljande load-satser. Den kan stängas av igen med **NullAsNull**-satsen.

Argument:

Argument

Argument	Beskrivning
*fieldlist	En kommaavgränsad lista över fält för vilka **NullAsValue** ska aktiveras. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken.

Exempel:

```
NullAsValue A,B;

Set NullValue = 'NULL';

LOAD A,B from x.csv;
```

Qualify

Satsen **Qualify** används för att aktivera bestämning av fältnamn, där fältnamn får tabellnamnet som prefix.

Syntax:

```
Qualify *fieldlist
```

Det är möjligt att förhindra automatiska länkningar mellan fält med samma namn i olika tabeller genom att använda en **qualify**-sats. Denna kvalificerar fältnamnet med dess tabellnamn. Fältnamn som förekommer i qualify-satsen döps om när de påträffas i tabeller. Det nya namnet är i formen av *tablename.fieldname*. *Tablename* motsvarar den aktuella tabellens etikett; finns det inget sådant, används istället det namn som förekommer efter **from** i **LOAD**- och **SELECT**-satser.

Bestämningen görs för alla fält som läses in efter **qualify**-satsen.

Som standard gäller att tabellbestämningen är inaktiverad i början av varje skriptexekvering. Tabellbestämningen kan när som helst aktiveras med hjälp av en **qualify**-sats. Bestämningen kan när som helst upphävas med hjälp av en **Unqualify**-sats.

***qualify**-satsen kan inte användas tillsammans med partiell laddning!*

Argument:

Argument

Argument	Beskrivning
*fieldlist	En kommaavgränsad lista över fält för vilka tabellbestämningen ska aktiveras. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken.

Example 1:

```
Qualify B;

LOAD A,B from x.csv;

LOAD A,B from y.csv;
```

De båda tabellerna **x.csv** och **y.csv** associeras endast genom **A**. Resultatet blir tre fält: A, x.B, y.B.

Example 2:

När man arbetar med en okänd databas är det ofta bra att till en början associera endast ett eller ett fåtal fält, som i följande exempel:

```
qualify *;

unqualify TransID;

SQL SELECT * from tab1;

SQL SELECT * from tab2;

SQL SELECT * from tab3;
```

Endast **TransID** kommer att användas för att associera tabellerna *tab1*, *tab2* och *tab3*.

Rem

rem-satsen används för att infoga kommentarer i skriptet, eller för att tillfälligt avaktivera skriptsatser utan att ta bort dem.

Syntax:

```
Rem string
```

Allting mellan **rem** och nästa semikolon **;** betraktas som kommentar.

Du kan skapa kommentarer i skriptet på två olika sätt.

1. Du kan skapa en kommentar var som helst i skriptet – förutom mellan två citationstecken – genom att placera avsnittet i fråga mellan **/*** och ***/**.
2. När **//** anges i skriptet, blir all text som följer till höger på samma rad en kommentar. (Observera undantaget //: som kan användas som en del i en Internetadress.)

Argument:

Argument

Argument	Beskrivning
string	En godtycklig text.

Exempel:

```
Rem ** This is a comment **;

/* This is also a comment */

// This is a comment as well
```

Rename

Skriptnyckelordet **Rename** kan användas för att byta namn på tabeller eller fält som redan laddats.

Rename field

Denna skriptfunktion döper om ett eller flera Qlik Sense-fält efter att de har lästs in.

> *Det rekommenderas inte att du ger en variabel samma namn som ett fält eller en funktion i Qlik Sense.*

Antingen syntaxen **rename field** eller **rename fields** kan användas.

Syntax:

```
Rename Field (using mapname | oldname to newname{ , oldname to newname })

Rename Fields (using mapname | oldname to newname{ , oldname to newname })
```

Argument:

Argument	Beskrivning
mapname	Namnet på en tidigare inläst mappningstabell som innehåller ett eller flera par gamla och nya fältnamn.
oldname	Det gamla fältnamnet.
newname	Det nya fältnamnet.

Begränsningar:

Det går inte att byta namn på två fält så att de får samma namn.

Example 1:

```
Rename Field XAZ0007 to Sales;
```

Example 2:

```
FieldMap:

Mapping SQL SELECT oldnames, newnames from datadictionary;

Rename Fields using FieldMap;
```

Rename table

Denna skriptfunktion döper om ett eller flera interna tabeller i Qlik Sense efter att de har lästs in.

Antingen syntaxen **rename table** eller **rename tables** kan användas.

Syntax:

```
Rename Table (using mapname | oldname to newname{ , oldname to newname })

Rename Tables (using mapname | oldname to newname{ , oldname to newname })
```

Argument:

Argument

Argument	Beskrivning
mapname	Namnet på en tidigare inläst mappningstabell som innehåller ett eller flera par gamla och nya tabellnamn.
oldname	Det gamla tabellnamnet.
newname	Det nya tabellnamnet.

Begränsningar:

Två tabeller med olika namn kan inte byta namn till samma namn. Skriptet genererar ett fel om du försök byta namn på en tabell till samma namn som en befintlig tabell.

Example 1:

```
Tab1:

SELECT * from Trans;
```

```
Rename Table Tab1 to Xyz;
```

Example 2:

```
TabMap:

Mapping LOAD oldnames, newnames from tabnames.csv;

Rename Tables using TabMap;
```

Search

Search-satsen används för att inkludera eller utesluta fält i smartsökningen.

Syntax:

```
Search Include *fieldlist

Search Exclude *fieldlist
```

Du kan använda flera Search satser för att förfina urvalet av fält som ska tas med. Avsnitten utvärderas uppifrån och ned.

Argument:

Argument

Argument	Beskrivning
*fieldlist	En kommaavgränsad lista över fält som ska inkluderas eller uteslutas från sökningar i smartsökningsverktyget. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken.

Exempel:

Exempel på sökning

Sats	Beskrivning
`Search Include *;`	Inkludera alla fält i sökningar i smartsökningsverktyget.
`Search Exclude [*ID];`	Uteslut alla fält som slutar med ID från sökningar i smartsökning.
`Search Exclude '*ID';`	Uteslut alla fält som slutar med ID från sökningar i smartsökning.
`Search Include ProductID;`	Inkludera fältet ProductID i sökningar i smartsökning.

Det kombinerade resultatet av dessa tre satser, i den här ordningen, är att alla fält som slutar med ID utom ProductID utesluts från sökningar i smartsökning.

Section

Med en **section**-sats är det möjligt att definiera om efterföljande **LOAD**- och **SELECT**-satser ska betraktas som data eller som en definition av behörighet.

```
Section (access | application)
```

Om inget anges, antas **section application**. **section**-definitionen används fram tills en ny **section**-sats påträffas.

```
Section access;
Section application;
```

Select

Val av fält från en ODBC-datakälla eller en OLE DB-drivrutin görs via vanliga SQL **SELECT**-satser. Om **SELECT**-satserna accepteras eller ej beror framför allt på den ODBC-drivrutin eller OLE DB-drivrutin som används. Användning av satsen **SELECT** kräver en öppen datakoppling till källan.

Syntax:

```
Select [all | distinct | distinctrow | top n [percent] ] fieldlist

From tablelist

[where criterion ]

[group by fieldlist [having criterion ] ]

[order by fieldlist [asc | desc] ]

[ (Inner | Left | Right | Full) join tablename on fieldref = fieldref ]
```

Dessutom kan flera **SELECT**-satser ibland konkateneras till en enda sats med hjälp av en **union**-operator:

```
selectstatement Union selectstatement
```

SELECT-satsen tolkas av ODBC-drivrutinen eller OLE DB-leverantören, så avvikelser från den allmänna SQL-syntaxen kan uppstå beroende på funktionerna hos ODBC-drivrutinerna eller OLE DB-leverantören, exempelvis:

- **as** är ibland inte tillåtet, dvs. *aliasname* måste följa omedelbart efter *fieldname*.
- **as** är ibland obligatoriskt om ett *aliasname* används.
- **distinct**, **as**, **where**, **group by**, **order by** eller **union** stöds ibland inte.
- ODBC-drivrutinen saknar ibland stöd för några av de citationstecken som beskrivs ovan.

*Detta är ingen komplett beskrivning av SQL **SELECT**-satsen! **SELECT**-satser kan t.ex. kapslas, en **SELECT**-sats kan innehålla flera join-satser, ibland tillåts ett stort antal funktioner i uttrycken osv.*

Argument:

Argument

Argument	Beskrivning
distinct	**distinct** används om duplicerade kombinationer av värden i de valda fälten bara ska läsas in en gång.
distinctrow	**distinctrow** används om duplicerade poster i ursprungstabellen bara ska läsas in en gång.
fieldlist	**`fieldlist ::= (*| field ) {, field }`** En lista över de fält som ska väljas. Genom att använda * som fältlista anger man alla fält i tabellen. **`fieldlist ::= field {, field }`** En lista över ett eller flera kommaavgränsade fält. **`field ::= ( fieldref| expression ) [as aliasname ]`** Uttrycket kan exempelvis vara ett numeriskt uttryck eller en teckensträng från ett eller flera olika fält. Några av de operatorer och funktioner som normalt kan användas: +, -, *, /, & (konkatenering av strängar), sum(fieldname), count(fieldname), avg(fieldname) (average), month(fieldname), osv. Se dokumentationen till ODBC-drivrutinen för mer information. **`fieldref ::= [ tablename. ] fieldname`** **tablename** och **fieldname** är textsträngar som motsvarar det de beskriver. De måste stå inom raka, dubbla citationstecken om de innehåller exempelvis blanktecken. **as**-tillägget används för att tilldela fältet ett nytt namn.
from	**`tablelist ::= table {, table }`** En lista över de tabeller från vilka fälten ska hämtas. **`table ::= tablename [ [as ] aliasname ]`** **tablename** kan sättas inom citationstecken.
where	**where** är ett tillägg som används för att tala om huruvida en post ska inkluderas i valet eller inte. **criterion** är ett logiskt uttryck, som ibland kan vara mycket komplext. En del av de operatorer som godtas är: numeriska operatorer och funktioner, **=**, **<>** eller **#**(ej lika med), **>**, **>=**, **<**, **<=**, **and**, **or**, **not**, **exists**, **some**, **all**, **in** och även nya **SELECT**-satser. Se dokumentationen till ODBC-drivrutinen eller OLE DB-providern för mer information.

Argument	Beskrivning
group by	**group by** är en sats som används för att aggregera (gruppera) flera poster till en. Inom en grupp, för ett visst fält, måste alla poster ha samma värde, eller så kan fältet endast förekomma i uttryck i vilka kollektiva egenskaper räknas ut, exempelvis summa eller medelvärde. Uttrycket, som baseras på ett eller flera fält, definieras i fältsymboluttrycket.
having	**having** är en sats som används för att kvalificera grupper på ett liknande sätt som satsen **where** används för att kvalificera poster.
order by	**order by** är en sats som specificerar sorteringsordningen för den tabell som **SELECT**-satsen resulterar i.
join	**join** är en kvalificerare som talar om att flera tabeller ska länkas samman till en enda. Fältnamn och tabellnamn måste sättas inom citationstecken om de innehåller blanktecken eller å,ä,ö. Om skriptet genereras automatiskt av Qlik Sense, används vanligen de citationstecken som ODBC-drivrutinen eller OLE DB-providern specificerar i datakällans definition av datakällan i **Connect**-satsen.

Example 1:

```
SELECT * FROM `Categories`;
```

Example 2:

```
SELECT `Category ID`, `Category Name` FROM `Categories`;
```

Example 3:

```
SELECT `Order ID`, `Product ID`,
`Unit Price` * Quantity * (1-Discount) as NetSales
FROM `Order Details`;
```

Example 4:

```
SELECT `Order Details`.`Order ID`,
Sum(`Order Details`.`Unit Price` * `Order Details`.Quantity) as `Result`
FROM `Order Details`, Orders
where Orders.`Order ID` = `Order Details`.`Order ID`
group by `Order Details`.`Order ID`;
```

Set

Satsen **set** används för att definiera skriptvariabler. Dessa kan användas för att ersätta strängar, sökvägar, drivrutiner med mera.

Syntax:

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Sleep

sleep-satsen pausar skriptexekveringen under en angiven tidsperiod.

Syntax:

```
Sleep n
```

Argument:

Argument	Beskrivning
n	Anges i millisekunder, där *n* är ett positivt heltal som inte är större än 3600000 (dvs. 1 timme). Värdet kan vara ett uttryck.

Example 1:

```
Sleep 10000;
```

Example 2:

```
Sleep t*1000;
```

SQL

Med **SQL**-satsen kan du skicka ett godtyckligt SQL-kommando via en ODBC- eller OLE DB-koppling.

Syntax:

```
SQL sql_command
```

Om du skickar SQL-satser som uppdaterar databasen kommer ett fel att returneras om Qlik Sense har öppnat ODBC-kopplingen i skrivskyddat läge.

Syntax:

```
SQL SELECT * from tab1;
```

godtas och är den allmänt gällande syntaxen för **SELECT** av konsekvensskäl. SQL-prefixet kommer dock att förbli frivilligt för **SELECT**-satser.

Argument:

Argument	Beskrivning
sql_command	Ett giltigt SQL-kommando.

Example 1:

```
SQL leave;
```

Example 2:

```
SQL Execute <storedProc>;
```

SQLColumns

Satsen **sqlcolumns** returnerar ett antal fält som beskriver kolumnerna i den ODBC- eller OLE DB-datakälla som man gjort en koppling, **connect**, till.

Syntax:

```
SQLcolumns
```

Fälten kan kombineras med fält som genererats av **sqltables**- och **sqltypes**-kommandona för att ge en bra översikt över en viss databas. De tolv standardfälten är följande:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

COLUMN_NAME

DATA_TYPE

TYPE_NAME

PRECISION

LENGTH

SCALE

RADIX

NULLABLE

REMARKS

För en detaljerad beskrivning av dessa fält, se en referenshandbok om ODBC.

Exempel:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';

SQLcolumns;
```

Vissa ODBC-drivrutiner kanske inte stöder det här kommandot. Vissa ODBC-drivrutiner kan ge ytterligare fält.

SQLTables

Satsen **sqltables** returnerar ett antal fält som beskriver tabellerna i den ODBC- eller OLE DB-datakälla som man gjort en koppling, **connect**, till.

Syntax:

```
SQLTables
```

Fälten kan kombineras med fält som genererats av **sqlcolumns**- och **sqltypes**-kommandona för att ge en bra översikt över en viss databas. De fem standardfälten är följande:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

TABLE_TYPE

REMARKS

För en detaljerad beskrivning av dessa fält, se en referenshandbok om ODBC.

Exempel:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
```

```
SQLTables;
```

Vissa ODBC-drivrutiner kanske inte stöder det här kommandot. Vissa ODBC-drivrutiner kan ge ytterligare fält.

SQLTypes

Satsen **sqltypes** returnerar ett antal fält som beskriver typerna i den ODBC- eller OLE DB-datakälla som man gjort en koppling, **connect**, till.

Syntax:

```
SQLTypes
```

Fälten kan kombineras med fält som genererats av **sqlcolumns** och **sqltables**-kommandona för att ge en bra översikt över en viss databas. De femton standardfälten är följande:

TYPE_NAME

DATA_TYPE

PRECISION

LITERAL_PREFIX

LITERAL_SUFFIX

CREATE_PARAMS

NULLABLE

CASE_SENSITIVE

SEARCHABLE

UNSIGNED_ATTRIBUTE

MONEY

AUTO_INCREMENT

LOCAL_TYPE_NAME

MINIMUM_SCALE

MAXIMUM_SCALE

För en detaljerad beskrivning av dessa fält, se en referenshandbok om ODBC.

Exempel:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';

SQLTypes;
```

Vissa ODBC-drivrutiner kanske inte stöder det här kommandot. Vissa ODBC-drivrutiner kan ge ytterligare fält.

Star

Den textsträng som används för att representera den totala uppsättningen av alla värden i ett fält i databasen kan ställas in med hjälp av **star**-satsen. Den påverkar efterföljande **LOAD**- och **SELECT**-satser.

Syntax:

```
Star is[ string ]
```

Argument:

Argument

Argument	Beskrivning
string	En godtycklig text. Observera att strängen måste sättas inom citationstecken om den innehåller blanktecken. Om ingenting anges, antas **star is;**, dvs. stjärnsymbolen måste specificeras explicit för att vara tillgänglig. Definitionen används fram tills en ny **star**-sats skapas.

Om section access används bör inte **Star is**-satsen användas i datadelen av skriptet (under **Section Application**). Däremot kan stjärntecknet användas i de skyddade fälten i **Section Access**-delen av skriptet. I så fall behöver du inte använda den explicita **Star is**-satsen eftersom denna alltid är implicit i section access.

Begränsningar

- Du kan inte använda stjärntecknet i nyckelfält, dvs. fält som länkar tabeller.
- Du kan inte använda stjärntecknet i några fält som påverkas av **Unqualify**-satsen eftersom detta kan påverka fält som länkar tabeller.
- Du kan inte använda stjärntecknet i icke-logiska tabeller, till exempel tabeller av typerna info-load eller mapping-load.
- När stjärntecknet används i ett reduceringsfält (ett fält som länkar till data) i section access representerar det värdena som listas i detta fält i section access. Det representerar inte andra värden som kan ingå i data, men som inte är listade i section access.
- Du kan inte använda stjärntecknet med fält som påverkas av någon typ av datareduktion utanför **Section Access**-området.

Exempel

I exemplet nedan finns ett utdrag av ett dataladdningsskript med section access.

```
Star is *;



Section Access;

LOAD * INLINE [

ACCESS, USERID, OMIT

ADMIN, ADMIN,

USER, USER1, SALES

USER, USER2, WAREHOUSE

USER, USER3, EMPLOYEES

USER, USER4, SALES

USER, USER4, WAREHOUSE

USER, USER5, *

];



Section Application;

LOAD * INLINE [

SALES, WAREHOUSE, EMPLOYEES, ORDERS

1, 2, 3, 4
```

```
];
```

Följande gäller:

- *Star* -tecknet är *.
- Användaren *ADMIN* ser alla fält. Inget har utelämnats.
- Användaren *USER1* kan inte se fältet *SALES*.
- Användaren *USER2* kan inte se fältet *WAREHOUSE* .
- Användaren *USER3* kan inte se fältet *EMPLOYEES*.
- Användaren *USER4* läggs till två gånger i lösningen för att UTESLUTA två fält för den här användaren, *SALES* och *WAREHOUSE*.
- *USER5* har en ”*” som tillägg, vilket betyder att alla fält under OMIT är otillgängliga, d.v.s. användaren *USER5* kan inte se fälten *SALES*, *WAREHOUSE* och *EMPLOYEES* men den här användaren kan se fältet *ORDERS*.

Store

Store-satsen skapar en QVD-, Parquet-,CSV- eller TXT-fil.

```
Store [ fieldlist from] table into filename [ format-spec ];
```

Satsen skapar en explicit namngiven QVD, Parquet, eller textfil.

Satsen kan endast exportera fält från en datatabell, såvida du inte lagrar till Parquet. Om fält från flera tabeller ska exporteras till en QVD-, CSV- eller TXT-fil måste du först göra en explicit join i skriptet för att skapa den datatabell som ska exporteras. Du kan lagra flera tabeller i en enda Parquet genom att nästla data i Parquet-filerna.

Textvärdena exporteras till CSV-filen i UTF-8 med BOM-format. En avgränsare kan anges, se **LOAD**. Om **store** sparar till en CSV-fil stöds inte BIFF-export.

> *I vissa fall med data som inte är väl utformad kommer fälten att omges med dubbla citattecken för att säkerställa att data tolkas korrekt. Detta kan exempelvis hända om fältet innehåller tecken som citattecken, kommatecken, mellanslag eller radbrytningar.*

Argument för kommandot Store

Argument	Beskrivning
*fieldlist::= (* | field) { , field })*	En lista över de fält som ska väljas. Genom att använda * som fältlista anger du alla fält. *field::= fieldname [***as** *aliasname]* *fieldname* är en text som är identisk med fältnamnet i *table*. (Observera att fältnamnet måste omslutas av raka, dubbla citationstecken eller av hakparenteser om det innehåller exempelvis mellanslag eller andra tecken som inte är standard.) *aliasname* är ett alternativt namn för det fält som ska användas i den färdiga QVD- eller CSV-filen.
table	En skriptetikett för en redan inläst tabell som ska användas som datakälla.
filename	Namnet på målfilen inkluderar en giltig sökväg till en befintlig mappdatakoppling. **Exempel:** ***'lib://Table Files/target.qvd'*** I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** ***c:\data\sales.qvd*** • relativ till Qlik Sense applikationens arbetskatalog. **Exempel:** ***data\sales.qvd*** Om sökvägen utelämnas sparar Qlik Sense filen i den mapp som specificerats av en **Directory**-sats. Om det inte finns någon **Directory**-sats, lagrar Qlik Sense filen i arbetskatalogen, *C:\Users\{user}\Documents\Qlik\Sense\Apps*.

Argument	Beskrivning
format-spec ::=((**txt** | **qvd** | **parquet**), **komprimering är** *codec*)	Du kan ställa in formatspecifikationen till något av dessa filformat. Om formatspecifikationen utelämnas antas **qvd**. • **txt** för CSV och TXT-filer. • **qvd** för QVD-filer. • **parquet** för Parquet-filer. Om du använder **parquet** kan du också ställa in vilken komprimeringskodek som ska användas med **komprimering är**. Om du inte anger komprimeringskodeken med **komprimering är** så används `snappy`. Följande komprimeringsinställningar är tillgängliga: • `uncompressed` • `snappy` • `gzip` • `lz4` • `brotli` • `zstd` • `lz4_hadoop` Exempel: `Store mytable into [lib://AttachedFiles/myfile.parquet] (parquet, compression is lz4);`

```
Store mytable into xyz.qvd (qvd);
Store * from mytable into 'lib://FolderConnection/myfile.qvd';
Store Name, RegNo from mytable into xyz.qvd;
Store Name as a, RegNo as b from mytable into 'lib://FolderConnection/myfile.qvd';
Store mytable into myfile.txt (txt);
Store mytable into [lib://FolderConnection/myfile.csv] (txt);
Store mytable into myfile.parquet (parquet);
Store * from mytable into 'lib://FolderConnection/myfile.qvd';
```

Lagra i parquet-filer

Parquet är ett starkt typiserat filformat, där varje fält innehåller en enda specifik typ av data (t.ex. int32, double, tidsmarkör eller text). Qlik Sense lagrar intern data som en löst typiserad dual, där data från olika källor kan blandas i samma fält. Eftersom bara en part av dualen kan lagras i varje fält i Parquet är det viktigt att veta vad varje fält innehåller. Som standard använder Qlik Sense fälttypen för att avgöra hur fältet ska lagras. När data lagras i Parquet-filer i ett specifikt format måste du ange vilken typ av data dina fält är när de laddas. Om du försöker lagra data i inkompatibla fält i en Parquet-fil, till exempel tal i ett textfält eller text i ett tidsmarkörfält kommer du att erhålla nullvärden.

När data laddas som du tänker lagra i Parquet är det möjligt att ändra standardbeteendet. Du kan antingen formatera det så att din datatyp ändras eller tagga det för att tvinga fram specifika kolumntyper i Parquet.

Formatera data för lagring i Parquet

Du kan använda Qlik Sense formateringsfunktioner för att klassificera dina data. Exempelvis kan **Text()**, **Num()**, **Interval()** eller **Timestamp()** tvinga fram dataformat när data lagras i Parquet. Qlik Sense kan lagra data i nästan 20 datatyper beroende på fältattribut och automatiska fälttaggar. Mer information finns i *Tolkningsfunktioner (page 1430)*.

Exempel: Formatera data med Num() och Text()

I följande exempel presenteras dataförberedelse för lagring i Parquet. **Num()** tillämpas på num-fältet. **Text ()** tillämpas på både text och blandade. Vid blandade förhindrar **Text()** det från att behandlas från som ett nummerfält i Parquet och att textvärdena ändras till nullvärden.

```
Data:
LOAD * INLINE [
num, text, mixed
123.321, abc, 123
456.654, def, xyz
789.987, ghi, 321
];

Format:
NoConcatenate
LOAD num, text, Text(mixed) as mixed RESIDENT Data;
STORE Format INTO [lib://AttachedFiles/Tmp.parquet] (parquet);
```

Tagga data för lagring i Parquet

Du taggar dina data med `$parquet`-taggar för att tvinga specifika kolumntyper när data lagras i Parquet. Varje datatyp kan tvingas fram genom att man lägger till motsvarande kontrolltagg. För att exempelvis lagra ett fält som INT32 i Parquet, taggar du det med `$parquet-int32` i laddningsskriptet. Beroende på datatypen kommer antingen strängen eller den numeriska representationen av de duala data att lagras.

Följande Parquet-kontrolltaggar kan användas för att tagga fält för att lagra dem i Parquet-filer.

Parquet-kontrolltaggar

Kontrolltagg	Dual	Fysisk typ	Logisk typ	Konverterad typ
`$parquet-boolean`	Tal	BOOLEAN	Ingen	Ingen
`$parquet-int32`	Tal	INT32	Ingen	Ingen
`$parquet-int64`	Tal	INT64	Ingen	Ingen
`$parquet-float`	Tal	FLOAT	Ingen	Ingen
`$parquet-double`	Tal	DOUBLE	Ingen	Ingen
`$parquet-bytearray`	Sträng	byte-array	Ingen	UTF8
`$parquet-bytearrayfix`	Tal	BYTE_ARRAY_MED_FAST_LÄNGD	Ingen	DECIMAL
`$parquet-`	Tal	INT64	DECIMAL	DECIMAL

Kontrolltagg	Dual	Fysisk typ	Logisk typ	Konverterad typ
`decimal`				
`$parquet-date`	Tal	INT32	DATE	DATE
`$parquet-time`	Tal	INT64	TIME	TIME_MICROS
`$parquet-timestamp`	Tal	INT64	TIMESTAMP	TIDSSTÄMPEL_MIKROSEKUNDER
`$parquet-string`	Sträng	byte-array	STRING	UTF8
`$parquet-enum`	Sträng	byte-array	ENUM	ENUM
`$parquet-interval`	Tal	BYTE_ARRAY_MED_FAST_LÄNGD	INTERVAL	INTERVAL
`$parquet-json`	Sträng	byte-array	JSON	JSON
`$parquet-bson`	Sträng	byte-array	BSON	BSON
`$parquet-uuid`	Sträng	BYTE_ARRAY_MED_FAST_LÄNGD	UUID	Ingen

Exempel: Tagga data för lagring i Parquet

I det här exemplet används två taggar för att definiera data för Parquet. Fältet *num* taggas med `$parquet-int32` för att definiera det som ett talfält som kommer att ställas in som INT32 i Parquet.

```
Data:
LOAD * INLINE [
num, text,
123.321, abc
456.654, def
789.987, ghi
];
TAG num WITH '$parquet-int32';
STORE Format INTO [lib://AttachedFiles/Tmp.parquet] (parquet);
```

Lagra nästlade data i Parquet-filer

Du kan lagra flera tabeller i Parquet-filer genom att nästla dem till strukturerade data. **Store** har stöd för strukturerade noder och listar noder i ett star-schema. Enstaka tabeller kan också lagras i nästlat läge genom att använda **Begränsare är**-specifikatorn.

När du lagrar tabeller ska du ange de tabeller du vill inkludera separerat med komman. Till exempel: `STORE Table1, Table2, Table3 INTO [lib://<file location>/<file name>.parquet] (parquet);`. Du kan styra vilka fält som lagras genom att använda en fältlista i **Store**-satsen. Till exempel `STORE Field1, Field2, FROMTable1, Table2 INTO[lib://file location/file name.parquet] (parquet);`. Alla fält i fältlistan måste finna i en eller flera av de angivna tabellerna. Den första tabellen i **Store**-satsen används som faktatabell i stjärn-schemat.

Fältnamn används för att styra hur grupper skapas och nästlas. Som standar delas fältnamn upp i noder med en punkt (`.`). Avgränsaren kan ändras genom att ställa in systemvariabeln *FieldNameDelimiter* eller genom att använda specificeraren **Avgränsare är**. Specificeraren åsidosätter systemvariabeln..

Fältnamn delas upp efter avgränsare och delarna används för att skapa schemat med nästlade grupper. `STORE Field1, Field1.Field2, Field1.Field3, Field1.Field4 FROM Table1 INTO [nested.parquet] (parquet, delimiter is '.');` kommer exempelvis att skapa två grupper (*Group1* och *Group*) med *Fields1, Fields2* och *Field3, Field4*.

> *Grupper och fält kanske inte har samma namn i en nod i schemat. Exempelvis kanske `STORE Address, Address.Street INTO [nested.parquet] (parquet, delimiter is '.');` inte fungerar eftersom Address är tvetydigt och är både ett datafält och en grupp.*

När nästlade data lagras i Parquet omvandlas nycklar mellan tabeller till länknoder i schemat. Tabeller omvandlas till strukturerade noder i schemat. Du kan åsidosätta standardomvandlingen med fältnamn.

Exempel: Lagra nästlade data i en Parquet-fil

```
company:
LOAD * INLINE [
company, contact
AG, Amanda Honda
Cabro, Cary Frank
Fenwick, Dennis Fisher
Camros, Molly McKenzie
];

salesrep:
LOAD * INLINE [
company, salesrep
AG, Bob Park
Cabro, Cezar Sandu
Fenwick, Ken Roberts
Camros, Max Smith
];

headquarter:
LOAD * INLINE [
company, country, city
AG, USA, Los Angeles
Cabro, USA, Albuquerque
Fenwick, USA, Baltimore
Camros, USA, Omaha
];

region:
LOAD * INLINE [
region, city
West, Los Angeles
Southwest, Albuquerque
East, Baltimore
Central, Omaha
];

STORE company, salesrep, headquarter, region INTO [lib://AttachedFiles/company.parquet]
(parquet)
DROP TABLES company, salesrep, headquarter, region;
```

Den resulterande Parquet-filen har följande schema:

```
company (String)
contact (String)
company:salesrep (List)
       salesrep (Group)
               salesrep (String)
company:headquarter (List)
       headquarter (Group)
               country (String)
               city (String)
               city:region (List)
                       region (Group)
                               region (String)
```

Begränsningar

Att lagra nästlade data i Parquet har följande begränsningar:

- Lagring har inte stöd för mappningsnoder.
- Lagring inkluderar inte nyckelfält som genererats från laddning av nästlade parquet-filer.
- Du kan inte lagra data från tabeller tillsammans om de inte är länkade med nyckelfält.
- Den nästlade filen avnormaliserar datamodellen. Icke-refererade värden kommer inte att sparas och värden som refererats flera gånger kommer att kopieras.

Begränsningar

- **STORE** laddningsskriptsatser stöds inte av Kopplingar till webblagringsleverantörer i Qlik Sense Enterprise on Windows och Qlik Sense Desktop.

Table/Tables

Skriptnyckelorden **Table** och **Tables** används i **Drop**-, **Comment**- och **Rename**-satser, liksom som en formatspecifikator i **Load**-satser.

Tag

Den här skriptsatsen erbjuder ett sätt att tilldela taggar till ett eller flera fält eller tabeller. Om du försöker tagga ett fält eller en tabell som inte finns i applikationen ignoreras taggningen. Om det finns flera förekomster av ett fältnamn eller taggnamn används det senaste värdet.

Syntax:

```
Tag [field|fields] fieldlist with tagname
```

```
Tag [field|fields] fieldlist using mapname
```

```
Tag table tablelist with tagname
```

Argument

Argument	Beskrivning
fieldlist	Ett eller flera fält som ska taggas, i en kommaavgränsad lista.
mapname	Namnet på en mappningstabell som tidigare laddats in via en **mapping Load**- eller **mapping Select**-sats.
tablelist	En kommaavgränsad lista över tabellerna som ska taggas.
tagname	Namnet på den tagg som ska användas på fältet.

Example 1:

```
tagmap:

mapping LOAD * inline [

a,b

Alpha,MyTag

Num,MyTag

];

tag fields using tagmap;
```

Example 2:

```
tag field Alpha with 'MyTag2';
```

Trace

trace-satsen skriver en sträng till fönstret **Skriptexekvering** och skriptets loggfil när den används. Detta är mycket användbart vid felsökning. Med hjälp av dollarteckenexpansion av variabler som beräknas innan **trace**-satsen används kan man anpassa meddelandet.

Syntax:

```
Trace string
```

Example 1:

Följande sats kan användas direkt efter Load-satsen som laddar Main-tabellen.

```
Trace Main table loaded;
```

Texten ”Main table loaded” (Huvudtabell har laddats) visas i skriptkörningsdialogen samt i loggfilen.

Example 2:

Följande satser kan användas direkt efter Load-satsen som laddar Main-tabellen.

```
Let MyMessage = NoOfRows('Main') & ' rows in Main table';
```

```
Trace $(MyMessage);
```
Då visas en text som visar antalet rader i skriptkörningsdialogen och i loggfilen, till exempel ”265,391 rows in Main table” (265 391 rader i huvudtabellen).

Unmap

Satsen **Unmap** avaktiverar fältvärdesmappningar, som har gjorts med hjälp av en tidigare **Map … Using**-sats för fält som läses in efteråt.

Syntax:

```
Unmap *fieldlist
```

Argument:

Argument

Argument	Beskrivning
*fieldlist	En kommaavgränsad lista över de fält som inte längre ska mappas från den aktuella positionen i skriptet. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken.

Exempel och resultat:

Exempel	Resultat
Unmap Country;	Inaktiverar mappning av fältet Country.
Unmap A, B, C;	Inaktiverar mappning av fälten A, B och C.
Unmap * ;	Inaktiverar mappning av alla fält.

Unqualify

Unqualify-satsen upphäver tabellbestämning av fältnamn som tidigare definierats av en **Qualify**-sats.

Syntax:

```
Unqualify *fieldlist
```

Argument:

Argument

Argument	Beskrivning
*fieldlist	En kommaavgränsad lista över fält för vilka tabellbestämningen ska aktiveras. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken. Se dokumentationen till **Qualify**-satsen för ytterligare information.

Example 1:

När man arbetar med en okänd databas är det ofta bra att till en början associera endast ett eller ett fåtal fält, som i följande exempel:

```
qualify *;

unqualify TransID;

SQL SELECT * from tab1;

SQL SELECT * from tab2;

SQL SELECT * from tab3;
```

Först aktiveras kvalificering för alla fält.
Sedan inaktiveras kvalificering för **TransID**.
Endast **TransID** kommer att användas för att associera tabellerna *tab1*, *tab2* och *tab3*. Alla andra fält kommer att kvalificeras med tabellnamnet.

Untag

Den här skriptsatsen erbjuder ett sätt att ta bort taggar från fält eller tabeller. Om du försöker ta bort taggar från ett fält eller en tabell som inte finns i applikationen, ignoreras försöket.

Syntax:

```
Untag [field|fields] fieldlist with tagname

Untag [field|fields] fieldlist using mapname

Untag table tablelist with tagname
```

Argument:

Argument

Argument	Beskrivning
fieldlist	Ett eller flera fält vars taggar ska tas bort, i en kommaavgränsad lista.
mapname	Namnet på en mappningstabell som tidigare har lästs in via en **LOAD**- eller **SELECT**-sats för mappning.
tablelist	En kommaavgränsad lista över tabellerna vars taggar ska tas bort.
tagname	Namnet på den tagg som ska tas bort från fältet.

Example 1:

```
tagmap:

mapping LOAD * inline [
```

```
a,b

Alpha,MyTag

Num,MyTag

];

Untag fields using tagmap;
```

Example 2:

```
Untag field Alpha with MyTag2;
```

Variable/Variables

De **Variable** och **Variables** skriptnyckelorden används i **Drop**-satser.

3.4 Arbetskatalog

Om du refererar till en fil i en skriptsats och sökvägen utelämnas så söker Qlik Sense efter filen i följande ordning:

1. Katalogen som anges av en **Directory**-sats (stöds endast i bakåtkompatibelt skriptläge).
2. Om det inte finns någon **Directory**-sats söker Qlik Sense i arbetskatalogen.

Qlik Sense Desktop-arbetskatalog

I Qlik Sense Desktop är arbetskatalogen *C:\Users\{user}\Documents\Qlik\Sense\Apps*.

Qlik Sense-arbetskatalog

I en Qlik Sense-serverinstallation anges arbetskatalogen i Qlik Sense Repository Service. Som standard är det *C:\ProgramData\Qlik\Sense\Apps*. Se hjälpen till Qlik Hanteringskonsol för mer information.

4 Arbeta med variabler i Skriptredigeraren

En variabel i Qlik Sense är en behållare som lagrar ett statiskt värde eller en beräkning, till exempel ett numeriskt eller ett alfanumeriskt värde. När du använder variabeln i applikationen, tillämpas alla förändringar som görs för variabeln överallt där variabeln används. Du kan definiera variabler med hjälp av variabelöversikten eller i skript med hjälp av Skriptredigeraren. Du ställer in värdet för en variabel med **Let** eller **Set** uttalanden i dataladdningsskriptet.

Du kan även arbeta med Qlik Sense-variablerna från variabelöversikten när du redigerar ett ark.

4.1 Översikt

Om det första tecknet i ett variabelvärde är ett likhetstecken ”=” kommer Qlik Sense att försöka utvärdera värdet som en formel (Qlik Sense-uttryck) och därefter visa eller returnera resultatet snarare än visa den faktiska formeln.

När variabeln används ersätts den av sitt värde. Variabler kan användas för dollarteckenexpansion eller i olika kontrollsatser i skriptet. Det är väldigt praktiskt om samma sträng förekommer flera gånger i skriptet, t.ex. en sökväg.

Vissa speciella systemvariabler ställs dock in av Qlik Sense i början av skriptexekveringen oavsett vilket värde de tidigare hade i layouten.

4.2 Definiera en variabel

Variabler gör det möjligt att lagra statiska värden eller resultatet av en beräkning. När en variabel definieras använder du följande syntax:

```
set variablename = string
```

eller

```
let variable = expression
```

Satsen **Set** används för strängtilldelning. Den tilldelar texten till höger om likhetstecknet till variablerna. Satsen **Let** utvärderar ett uttryck till höger om likhetstecknet under skriptets körningstid och tilldelar resultatet av uttrycket till variabeln.

Variabler är skiftlägeskänsliga.

Det rekommenderas inte att du ger en variabel samma namn som ett fält eller en funktion i Qlik Sense.

`set x = 3 + 4; //`-variabeln får strängen '3 + 4' som värde.

`let x = 3 + 4; //` returnerar 7 som värde.

`set x = Today(); //` returnerar 'Today()' som värde.

`let x = Today(); //` returnerar dagens datum som värde, till exempel '9/27/2021'.

Namnge dina variabler

Överväg som bästa praxis att använda en standardiserad namnkonvention för de variabler som du skapar i en applikation. Exempelvis kan du se till att alla dina variabelnamn börjar med *v*. Exempel: *vUserText*. Detta hjälper till att säkerställa att variabler snabbt identifieras som variabler och särskiljs från mått, fält och funktioner.

4.3 Ytterligare alternativ vid definition av variabler

Detta delavsnitt listar ytterligare funktioner som du kan använda för att förbättra kraften hos variabler som du definierar i laddningsskriptet.

Begränsa variabelvärden

Du kan använda **Constrain** satser för att begränsa skriptvariabler till specifika definitioner, vilket förhindrar att oönskade värden laddas. **Villkora** satser kan användas för kvalitetssäkring och ökad säkerhet för analysapplikationer.

Mer information finns i *Constrain (page 153)*.

Definiera skriptspecifika variabler

Du kan använda systemvariabeln **ScriptOnlyVariables** för att ställa in skriptspecifika variabler. En skriptspecifik variabel är endast tillgänglig i laddningsskriptet och visas inte i till exempel arkvyn i appen.

Mer information finns i *ScriptOnlyVariables (page 237)*.

Släppa variabler

Du kan använda en **Drop variable** sats för att släppa variabler i skriptet. Att släppa en variabel är användbart när du vill använda en variabel endast för att utföra beräkningar eller definiera fält i datamodellens kontext.

Mer information finns i *Drop variable (page 175)*.

4.4 Ta bort en variabel

Om du tar bort en variabel från skriptet och laddar data igen stannar variabeln kvar i applikationen. Om du vill ta bort variabeln helt från applikationen kan du göra något av följande:

- Ta bort variabeln från variabeldialogen.
- Släpp variabeln i skriptet med en Drop variable sats.
- Ange variabler som inte behövs i variabeldialogen som skriptspecifika variabler genom att använda systemvariabeln ScriptOnlyVariables i skriptet.

4.5 Ladda ett variabelvärde som ett fältvärde

Om du vill ladda ett variabelvärde som ett fältvärde i en **LOAD**-sats och resultatet av dollarexpansionen är text i stället för tal eller ett uttryck måste du bifoga den utvidgade variabeln inom enkla citattecken.

Detta exempel laddar systemvariablen som innehåller listan över skriptfel till en tabell. Lägg märke till att expansionen av ScriptErrorCount i **If**-satsen inte kräver citattecken, men att expansionen av ScriptErrorList kräver citattecken.

```
IF $(ScriptErrorCount) >= 1 THEN
   LOAD '$(ScriptErrorList)' AS Error AutoGenerate 1;
END IF
```

4.6 Beräkning med variabler

Det finns flera sätt att använda variabler med beräknade värden i Qlik Sense. Resultatet beror på hur du definierar den och hur du anropar den i ett uttryck.

I detta exempel laddar vi vissa data inline:

```
LOAD * INLINE [
    Dim, Sales
    A, 150
    A, 200
    B, 240
    B, 230
    C, 410
    C, 330
];
```

Nu ska vi definiera två variabler:

```
Let vSales  = 'Sum(Sales)' ;
Let vSales2  = '=Sum(Sales)' ;
```

I den andra variabeln lägger vi till ett likhetstecken framför uttrycket. Nu beräknas variabeln innan den expanderas och innan uttrycket utvärderas.

Om du använder variabeln vSales som den är, exempelvis i ett mått, blir resultatet strängen Sum(Sales). Med andra ord görs ingen beräkning.

Om du lägger till en dollarteckenexpansion och anropar $(vSales) i uttrycket, expanderas variabeln, och summan av Sales visas.

Om du i stället anropar $(vSales2) beräknas variabeln innan den expanderas. Detta innebär att resultatet som visas är totalsumman för Sales. Skillnaden mellan att använda =$(vSales) och =$(vSales2) som måttuttryck visas i diagrammet nedan. Resultat:

Resultat

Dim	$(vSales)	$(vSales2)
A	350	1560

Dim	$(vSales)	$(vSales2)
B	470	1560
C	740	1560

Som du ser resulterar $(vSales) i delsumman för ett dimensionsvärde, medan $(vSales2) resulterar i totalsumman.

Följande skriptvariabler är tillgängliga:

- *Felvariabler (page 305)*
- *Variabler för tolkning av tal (page 241)*
- *Systemvariabler (page 232)*
- *Variabler för värdehantering (page 239)*

4.7 Systemvariabler

Systemvariabler, varav vissa är systemdefinierade, ger information om systemet och Qlik Sense-applikationen.

Översikt av systemvariabler

En del av funktionerna beskrivs mer ingående efter översikten. För de här funktionerna kan du klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

CreateSearchIndexOnReload

Denna variabel definierar om sökindexfiler ska skapas under ominläsning av data.

```
CreateSearchIndexOnReload
```

Diskett

Returnerar enhetsbeteckningen för den första diskettenhet som påträffas, vanligen *a:*. Detta är en systemdefinierad variabel.

```
Floppy
```

Variabeln stöds inte i standardläget.

CD

Returnerar bokstaven på den första cd-rom-enhet som påträffas. Om ingen cd-rom-enhet påträffas, returneras *c:*. Detta är en systemdefinierad variabel.

```
CD
```

Variabeln stöds inte i standardläget.

DöljPrefix

Alla fältnamn som inleds av denna textsträng kommer att döljas på samma sätt som systemfälten. Detta är en användardefinierad variabel.

```
HidePrefix
```

HideSuffix

Alla fältnamn som avslutas av denna textsträng kommer att döljas på samma sätt som systemfälten. Detta är en användardefinierad variabel.

```
HideSuffix
```

Inkludera

Variabeln **Include/Must_Include** specificerar en fil som innehåller text som ska inkluderas i skriptet och utvärderas som skriptkod. Den används inte för att lägga till data. Du kan spara delar av din skriptkod i en separat textfil och återanvända den i flera applikationer. Detta är en användardefinierad variabel.

```
$(Include=filename)
$(Must_Include=filename)
```

Öppna URL-tidsgräns

Denna variabel anger i sekunder den tidsgräns som Qlik Sense ska hålla när data hämtas från URL-källor (t.ex. HTML sidor). Om den utelämnas blir det en tidsgräns på ungefär 20 minuter.

```
OpenUrlTimeout
```

QvPath

Returnerar sökströngen till Qlik Sense-programfilen. Detta är en systemdefinierad variabel.

```
QvPath
```

Variabeln stöds inte i standardläget.

QvRoot

Returnerar rotkatalogen till Qlik Sense-programfilen. Detta är en systemdefinierad variabel.

```
QvRoot
```

Variabeln stöds inte i standardläget.

QvWorkPath

Returnerar sökströngen till den aktuella Qlik Sense applikationen. Detta är en systemdefinierad variabel.

```
QvWorkPath
```

Variabeln stöds inte i standardläget.

QvWorkRoot

Returnerar rotmappen till den aktuella Qlik Sense appen. Detta är en systemdefinierad variabel.

```
QvWorkRoot
```

Variabeln stöds inte i standardläget.

Skriptspecifika variabler

Systemvariabeln **ScriptOnlyVariables** används för att definiera variabler som endast är tillgängliga i skript. Denna variabel anger om en skriptvariabel endast kommer att vara tillgänglig i skriptet, eller om den även kommer att vara tillgänglig (och synlig) utanför skriptet – till exempel i arkvyn i en app.

```
ScriptOnlyVariables
```

Ta bort kommentarer

Om denna variabel är inställd på 0 blir kommentarer av typen /*..*/ och // otillåtna. Om variabeln inte är definierad strippas alltid kommentarer.

```
StripComments
```

Ordagrant

Normalt töms alla fältvärden automatiskt på inledande och avslutande blanktecken (ASCII 32) innan de läses in i Qlik Sense-databasen. Om man sätter variabeln till 1 fördröjs rensningen på tomma blanktecken. Tecknen tabb (ASCII 9) och hårt mellanslag (ANSI 160) rensas aldrig bort.

```
Verbatim
```

WinPath

Returnerar sökströngen till Windows. Detta är en systemdefinierad variabel.

```
WinPath
```

Variabeln stöds inte i standardläget.

WinRoot

Returnerar Windows rotmapp. Detta är en systemdefinierad variabel.

```
WinRoot
```

Variabeln stöds inte i standardläget.

CollationLocale

Anger vilken språkvariant som ska användas för sorteringsordning och sökmatchning. Värdet är kulturnamnet för en språkvariant, till exempel ”en-US”.Detta är en systemdefinierad variabel.

```
CollationLocale
```

CreateSearchIndexOnReload

Denna variabel definierar om sökindexfiler ska skapas under ominläsning av data.

`CreateSearchIndexOnReload`

Du kan definiera om sökindexfiler ska skapas vid omladdning av data eller om de ska skapas efter den första sökbegäran från användaren. Fördelen med att skapa sökindexfiler under omladdning av data är att du undviker den väntetid som den första användaren som gör en sökning annars får. Detta måste vägas mot den längre tiden för omladdning av data som krävs vid skapande av ett sökindex.

Om denna variabel utelämnas skapas inte sökindexfiler vid omladdning av data.

För sessionsapplikationer skapas inte sökindexfiler under omladdning av data, oavsett inställningen för denna variabel.

Nya applikationer inkluderar `set CreateSearchIndexOnReload=1` i standard `SET`-uttalanden. När du skapar skript i aktivitetscentret för Analyser inkluderar inte SET-uttrycken `CreateSearchIndexOnReload` eftersom skripttillgångar inte lagrar data, vilket gör ett indexvärde onödigt.

Example 1: Skapa sökindexfiler under omladdning av data.

```
set CreateSearchIndexOnReload=1;
```

Example 2: Skapa sökindexfiler efter första sökningsbegäran

```
set CreateSearchIndexOnReload=0;
```

HidePrefix

Alla fältnamn som inleds av denna textsträng kommer att döljas på samma sätt som systemfälten. Detta är en användardefinierad variabel.

Syntax:

`HidePrefix`

Exempel:

```
set HidePrefix='_' ;
```

Satsen medför att fältnamn som börjar med ett understrykningstecken inte kommer att visas i fältnamnslistorna om systemfälten är dolda.

HideSuffix

Alla fältnamn som avslutas av denna textsträng kommer att döljas på samma sätt som systemfälten. Detta är en användardefinierad variabel.

Syntax:

```
HideSuffix
```

Exempel:

```
set HideSuffix='%';
```

Satsen medför att fältnamn som avslutas av ett procenttecken inte kommer att visas i fältnamnslistorna om systemfälten är dolda.

Include

Variabeln **Include/Must_Include** specificerar en fil som innehåller text som ska inkluderas i skriptet och utvärderas som skriptkod. Den används inte för att lägga till data. Du kan spara delar av din skriptkod i en separat textfil och återanvända den i flera applikationer. Detta är en användardefinierad variabel.

> *I Qlik Sense och Qlik Sense Desktop stöder denna variabel endast referenser till kopplingar till mappdata. Referenser till filer hos molnbaserade lagringsleverantörer stöds inte.*

```
$(Include=filename)
$(Must_Include=filename)
```

Det finns två versioner av variabeln:

- **Include** genererar inte ett del om det inte går att hitta filen, den misslyckas i tysthet.
- **Must_Include** genererar ett fel om det inte går att hitta filen.

Om du inte anger en sökväg, blir filnamnet relativt till Qlik Sense-applikationens arbetskatalog. Du kan också ange en absolut sökväg till filen, eller en sökväg till lib://-katalogens anslutning. Det får inte finnas något blanksteg före eller efter likhetstecknet.

> *Konstruktionen **set Include =**filename är inte tillämplig.*

```
$(Include=abc.txt);
$(Must_Include=lib://DataFiles/abc.txt);
$(Include=abc.txt);
$(Must_Include=lib://DataFiles/abc.txt);
```

Begränsningar

Begränsad kompatibilitet mellan Windows och Linux för UTF-8-kodade filer.

Det är valfritt att använda UTF-8 med BOM (Byte Order Mark). BOM kan störa användningen av UTF-8 i program som inte förväntar sig icke-ASCII-byte i början av en fil, men som annars skulle kunna hantera textströmmen.

- Windows-system använder BOM i UTF-8 för att identifiera att en fil är UTF-8-kodad, trots att det inte råder någon tveksamhet om hur lagring i byte ska ske.
- Unix/Linux använder UTF-8 för Unicode men använder inte BOM, eftersom detta stör syntaxen i kommandofiler.

Detta medför vissa konsekvenser för Qlik Sense.

- I Windows identifieras alla filer som börjar med en UTF-8 BOM som en UTF-8-skriptfil. Annars antas att ANSI-kodning används.
- I Linux är UTF-8 systemets standardkodsida för 8 bitar. Därför fungerar UTF-8 fastän den inte innehåller någon BOM.

Portabilitet kan därför inte garanteras. En fil som skapas i Windows kan inte alltid tolkas av Linux och vice versa. För UTF-8-kodade filer finns ingen tvärkompatibilitet mellan systemen eftersom de hanterar BOM på olika sätt.

ScriptOnlyVariables

Systemvariabeln **ScriptOnlyVariables** används för att definiera variabler som endast är tillgängliga i skript. Denna variabel anger om en skriptvariabel endast kommer att vara tillgänglig i skriptet, eller om den även kommer att vara tillgänglig (och synlig) utanför skriptet – till exempel i arkvyn i en app.

När ett **ScriptOnlyVariables** värde har definierats –antingen explicit, eller implicit via dess standardvärde 0 – kommer alla variabler som definieras i **Let** eller **Set** satser efter denna definition att använda den definierade inställningen, såvida inte **ScriptOnlyVariables** värdet ändras senare i skriptet. Detta innebär att du kan definiera vissa variabler som endast för skript och tillåta att andra visas utanför skriptet.

Syntax:

```
ScriptOnlyVariables
```

`ScriptOnlyVariables`-värden

Värde	Resultat
0	Variabler visas utanför skriptet, förutom att vara tillgängliga inom själva skriptet. Detta är standardvärdet om **ScriptOnlyVariables** variabelvärdet inte definieras explicit i skriptet.
1	Variabler visas inte utanför skriptet.

Example 1: Visa en variabel utanför skriptet

I det här exemplet är variabeln `vNotScriptOnly` explicit inställd att visas utanför skriptet.

```
SET ScriptOnlyVariables = 0;
SET vNotScriptOnly = "example_text";
```

Example 2: Dölja en variabel utanför skriptet

I det här exemplet kommer variabeln `vScriptOnly` endast att vara tillgänglig under skriptkörning. Den kommer inte att visas någon annanstans, som till exempel i arkvyn.

```
SET ScriptOnlyVariables = 1;
SET vScriptOnly = "example_text";
```

Example 3: Dölja vissa variabler och visa andra

I det här exemplet definieras värdet **ScriptOnlyVariables** flera gånger i hela skriptet för att tillåta olika variabelinställningar. Variabeln `vHideOutsideScript` kommer endast att vara tillgänglig under skriptkörning. Variabeln `vAllowInSheets` kommer att visas i arkvyn.

```
SET ScriptOnlyVariables = 1;
SET vHideOutsideScript = "example_text";
SET ScriptOnlyVariables = 0;
SET vAllowInSheets = "example_text";
```

När ska du använda ScriptOnlyVariables

Endast skriptvariabler hjälper dig att:

- Använd vissa variabler endast för att utföra beräkningar och fältdefinitioner i samband med datamodellen.
- Ta bort variabler som inte behövs för att skapa ark och diagram, vilket förenklar applikationsutvecklingen.

OpenUrlTimeout

Denna variabel anger i sekunder den tidsgräns som Qlik Sense ska hålla när data hämtas från URL-källor (t.ex. HTML sidor). Om den utelämnas blir det en tidsgräns på ungefär 20 minuter.

Syntax:

```
OpenUrlTimeout
```

Exempel:

```
set OpenUrlTimeout=10;
```

StripComments

Om denna variabel är inställd på 0 blir kommentarer av typen /*..*/ och // otillåtna. Om variabeln inte är definierad strippas alltid kommentarer.

Syntax:

```
StripComments
```

Vissa databasdrivrutiner använder sig av /*..*/ som optimeringstips i **SELECT**-satser. Om detta är fallet bör kommentarerna inte rensas bort innan **SELECT**-satsen skickas till databasens drivrutin.

Det rekommenderas att variabeln återställs till 1 omedelbart efter satserna vid behov.

Exempel:

```
set StripComments=0;

SQL SELECT * /* <optimization directive> */ FROM Table ;

set StripComments=1;
```

Verbatim

Normalt töms alla fältvärden automatiskt på inledande och avslutande blanktecken (ASCII 32) innan de läses in i Qlik Sense-databasen. Om man sätter variabeln till 1 fördröjs rensningen på tomma blanktecken. Tecknen tabb (ASCII 9) och hårt mellanslag (ANSI 160) rensas aldrig bort.

Syntax:

```
Verbatim
```

Exempel:

```
set Verbatim = 1;
```

4.8 Variabler för värdehantering

Den här delen beskriver variabler som används för att hantera NULL och andra värden.

Översikt av variabler för värdehantering

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

NullDisplay
Den definierade symbolen kommer att ersätta alla NULL-värden från ODBC, samt kopplingarna, på den lägsta datanivån. Detta är en användardefinierad variabel.

```
NullDisplay
```

NullInterpret
Symbolen som definieras kommer att tolkas som NULL varje gång den förekommer i en textfil, CSV-fil eller i en inline-sats. Detta är en användardefinierad variabel.

```
NullInterpret
```

NullValue
Om **NullAsValue**-satsen används ersätter den definierade symbolen alla NULL-värden i **NullAsValue**-angivna fält med den angivna strängen.

```
NullValue
```

OtherSymbol
Definierar att en symbol som ska behandlas som "alla andra värden" före en **LOAD/SELECT**-sats. Detta är en användardefinierad variabel.

```
OtherSymbol
```

NullDisplay

Den definierade symbolen kommer att ersätta alla NULL-värden från ODBC, samt kopplingarna, på den lägsta datanivån. Detta är en användardefinierad variabel.

Syntax:

```
NullDisplay
```

Exempel:

```
set NullDisplay='<NULL>';
```

NullInterpret

Symbolen som definieras kommer att tolkas som NULL varje gång den förekommer i en textfil, CSV-fil eller i en inline-sats. Detta är en användardefinierad variabel.

Syntax:

```
NullInterpret
```

Exempel:

```
set NullInterpret=' ';

set NullInterpret =;

set NullInterpret ='';
```

NullValue

Om **NullAsValue**-satsen används ersätter den definierade symbolen alla NULL-värden i **NullAsValue**-angivna fält med den angivna strängen.

Syntax:

```
NullValue
```

Exempel:

```
NullAsValue Field1, Field2;

set NullValue='<NULL>';
```

OtherSymbol

Definierar att en symbol som ska behandlas som "alla andra värden" före en **LOAD/SELECT**-sats. Detta är en användardefinierad variabel.

Syntax:

```
OtherSymbol
```

Exempel:

```
set OtherSymbol='+';

LOAD * inline

[X, Y

a, a

b, b];

LOAD * inline

[X, Z

a, a

+, c];
```

fältvärdet Y='b' länkar nu till Z='c' via den andra symbolen.

4.9 Variabler för tolkning av tal

Nummertolkning variabler är systemdefinierade. Variablerna är inkluderade överst i laddningsskriptet och tillämpar inställningar för nummerformatering vid den tidpunkt då skriptet körs. De kan raderas, redigeras eller dupliceras.

Variabler för tolkning av tal genereras automatiskt i överensstämmelse med rådande regionala inställningar i operativsystemet när en ny app skapas. I Qlik Sense Desktop överensstämmer detta med inställningarna för datorns operativsystem. I Qlik Sense, beror det på operativsystemet på servern där Qlik Sense är installerat. Om Qlik Sense-servern du ansluter till är inställd på Sverige så kommer skriptredigeraren använda regionala inställningar för Sverige för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense visas på samma språk som webbläsaren du använder.

Valutaformat

Decimalavgränsare för valuta

Den definierade decimalavgränsaren ersätter decimalsymbolen för valuta som ställts in av dina regionala inställningar.

```
MoneyDecimalSep
```

Valutaformat

Den definierade symbolen ersätter valutasymbolen som ställts in av dina regionala inställningar.

```
MoneyFormat
```

Valutatusenavgränsare

Den definierade tusentalsavgränsaren ersätter siffergrupperingssymbolen för valuta som ställts in av dina regionala inställningar.

```
MoneyThousandSep
```

Talformat

Decimalavgränsare

Den definierade decimalavgränsaren ersätter decimalsymbolen som ställts in av dina regionala inställningar.

```
DecimalSep
```

ThousandSep

Tusentalsavgränsaren som definieras ersätter operativsystemets tusentalsavgränsare (regionala inställningar).

```
ThousandSep
```

Numerisk förkortning

Den numeriska förkortningen ställer in vilken förkortning att använda för siffrors skalprefix, till exempel M för mega eller miljon (10^6) och µ för mikro (10^{-6}).

```
NumericalAbbreviation
```

Tidsformat

DateFormat

Denna miljövariabel definierar datumformatet som används som standard i applikationen. Formatet används både för att tolka och formatera datum. Om variabeln inte är definierad kommer datumformatet enligt regioninställningarna i operativsystemet att hämtas när skriptet körs.

```
DateFormat
```

Tidsformat

Formatet som definieras ersätter operativsystemets format för tid (regionala inställningar).

```
TimeFormat
```

Tidsstämpelformat

Formatet som definieras ersätter operativsystemets format för datum och tid (regionala inställningar).

```
TimestampFormat
```

MonthNames

Det definierade formatet ersätter konventionen med månadsnamn i regionala inställningar.

```
MonthNames
```

Långa namn på månader

Det definierade formatet ersätter konventionen med långa månadsnamn i regionala inställningar.

```
LongMonthNames
```

dagnamn

Det definierade formatet ersätter konventionen med veckodagsnamn som ställts in av dina regionala inställningar.

```
DayNames
```

Långa namn på dagar

Det definierade formatet ersätter konventionen med långa veckodagsnamn i regionala inställningar.

```
LongDayNames
```

Första veckodagen

Heltal som definierar vilken dag som ska användas som den första dagen i veckan.

```
FirstWeekDay
```

Brutna veckor

Den här inställningen definierar om veckor ska brytas eller inte.

```
BrokenWeeks
```

Referensdag

Inställningen definierar vilken dag i januari som ska ställas in som referensdag för att definiera vecka 1.

ReferenceDay

FörstaMånadenPåÅret
Inställningen definierar vilken månad som ska användas som inledande månad på året, och kan användas för att definiera brutna räkenskapsår, exempelvis ett som inleds den 1 april.

> *Den här inställningen används för närvarande inte, utan är reserverad för framtida användning.*

Giltiga inställningar är 1 (januari) till 12 (december). Standardinställningen är 1.

FirstMonthOfYear

```
Set FirstMonthOfYear=4; //Sets the year to start in April
```

BrokenWeeks

Den här inställningen definierar om veckor ska brytas eller inte.

Syntax:

BrokenWeeks

I Qlik Sense, hämtas de regionala inställningarna när applikationen skapas, och motsvarande inställningar lagras i skriptet som miljövariabler.

En nordamerikansk applikationsutvecklare får ofta `Set BrokenWeeks=1;` i skriptet, vilket motsvarar brutna veckor. En europeisk applikationsutvecklare får ofta `Set BrokenWeeks=0;` i skriptet, vilket motsvarar obrutna veckor.

Obrutna veckor innebär att:

- Vissa år börjar vecka 1 i december, och andra år fortsätter den sista veckan från föregående år in i januari.
- Enligt ISO 8601, har vecka 1 alltid minst 4 dagar i januari. I Qlik Sense, kan detta konfigureras med hjälp av variabeln `ReferenceDay`.

Brutna veckor innebär följande:

- Årets sista vecka fortsätter aldrig in i januari.
- Vecka 1 börjar den 1 januari och är i de flesta fall inte en hel vecka.

Följande värden kan användas:

- 0 (= använd obrutna veckor)
- 1 (= använd brutna veckor)

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel:

Om du vill ha ISO-inställningar för veckor och veckonummer, ska du se till att ha följande i skriptet:

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;   //(use unbroken weeks)
Set ReferenceDay=4;
```

Om du vill ha USA-inställningar, ska du se till att ha följande i skriptet:

```
Set FirstWeekDay=6;
Set BrokenWeeks=1;   //(use broken weeks)
Set ReferenceDay=1;
```

DateFormat

Denna miljövariabel definierar datumformatet som används som standard i applikationen och av funktioner som returnerar datum, som `date()` och `date#()`. Formatet används för att tolka och formatera datum. Om variabeln inte är definierad kommer datumformatet enligt regioninställningarna i operativsystemet att hämtas när skriptet körs.

Syntax:

```
DateFormat
```

Exempel på DateFormat-funktion

Exempel	Resultat
`Set DateFormat='M/D/YY'; //(US format)`	Denna användning av `DateFormat`-funktionen definierar datumet som US-format, månad/dag/år.
`Set DateFormat='DD/MM/YY'; //(UK date format)`	Denna användning av `DateFormat`-funktionen definierar datumet som UK-format, dag/månad/år.
`Set DateFormat='YYYY/MM/DD'; //(ISO date format)`	Denna användning av `DateFormat`-funktionen definierar datumet som ISO-format, år/månad/dag.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Standardsystemvariabler

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum.
- `DateFormat`-funktionen som använder US-datumformat.

I det här exemplet laddas en datauppsättning i en tabell som heter `Transactions`. Den inkluderar ett `date`-fält. US `DateFormat`-definitionen används. Detta mönster kommer att användas för implicit text-till-data-konvertering när textdatumen laddas.

Laddningsskript

```
Set DateFormat='MM/DD/YYYY';

Transactions:
LOAD
date,
month(date) as month,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `month`

Skapa det här måttet:

```
=sum(amount)
```

Resultattabell

datum	månad	=sum(amount)
01/01/2022	Jan	1000
2022-02-01	Feb	2123
2022-03-01	Mar	4124
04/01/2022	Apr	2431

`DateFormat`-definitionen MM/DD/ÅÅÅÅ används för implicit konvertering av text till datum, därför är `date`-fältet korrekt tolkat som ett datum. Samma format används för att visa datumet, som visas i resultattabellen.

Exempel 2 – Byt systemvariabel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning från det tidigare exemplet.
- Funktionen `DateFormat` som använder formatet ”DD/MM/ÅÅÅÅ”.

Laddningsskript

```
SET DateFormat='DD/MM/YYYY';
Transactions:
LOAD
date,
month(date) as month,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `month`

Skapa det här måttet:

```
=sum(amount)
```

Resultattabell

datum	månad	=sum(amount)
01/01/2022	Jan	1000
2022-02-01	Jan	2123
2022-03-01	Jan	4124
04/01/2022	Jan	2431

Eftersom definitionen `DateFormat` var inställd på DD/MM/ÅÅÅÅ kan du se att de två siffrorna efter den första ”/”-symbolen har tolkats som månaden, vilket resulterar i alla poster från månaden januari.

Exempel 3 – Datatolkning

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum i numeriskt format.
- Variabeln `DateFormat` som använder formatet ”DD/MM/ÅÅÅÅ”.
- Variabeln `date()`.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
date(numerical_date),
month(date(numerical_date)) as month,
id,
amount
Inline
[
numerical_date,id,amount
43254,1,1000
43255,2,2123
43256,3,4124
43258,4,2431
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `month`

Skapa det här måttet:

```
=sum(amount)
```

Resultattabell

datum	månad	=sum(amount)
06/03/2022	Jun	1000
06/04/2022	Jun	2123
06/05/2022	Jun	4124
06/07/2022	Jun	2431

I laddningsskriptet använder du funktionen `date()` för att konvertera det numeriska datumet till ett datumformat. Eftesom du inte ger ett specificerat format som ett andra argument i funktionen, används formatet `DateFormat`. Detta resulterar i att datafältet använder formatet ”MM/DD/ÅÅÅÅ”.

Exempel 4 – Utländsk dataumformatering

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum.
- `DateFormat`-variabeln, som använder formatet DD/MM/YYYY men inte är kommenterad med snedstreck framåt.

Laddningsskript

```
// SET DateFormat='DD/MM/YYYY';


Transactions:
Load
date,
month(date) as month,
id,
amount
Inline
[
date,id,amount
22-05-2022,1,1000
23-05-2022,2,2123
24-05-2022,3,4124
```

```
25-05-2022,4,2431
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `month`

Skapa det här måttet:

```
=sum(amount)
```

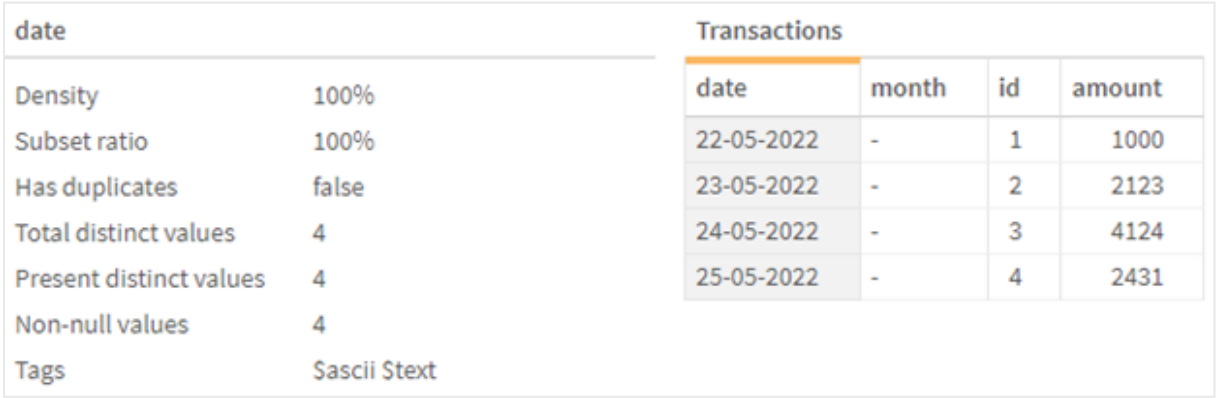
Resultattabell

datum	månad	=sum(amount)
22-05-2022	-	1000
23-05-2022	-	2123
24-05-2022	-	4124
25-05-2022	-	2431

I det första laddningsskriptet är det `DateFormat` som används standarden ”MM/DD/ÅÅÅÅ”. Eftersom `date`-fältet i transaktionsdatauppsättningen inte är i detta format, tolkas fältet inte som ett datum. Detta visas i resultattabellen där `month` -fältvärden är null.

Du kan verifiera de tolkade datatyperna i datamodellvyn genom att inspektera `date`-fältets taggars egenskaper:

Förhandsvisning av `Transactions`-tabellen. Notera taggar för `date`-fältet som indikerar att textinmatningsdata inte implicit har konverterats till ett datum eller tidsmarkör.

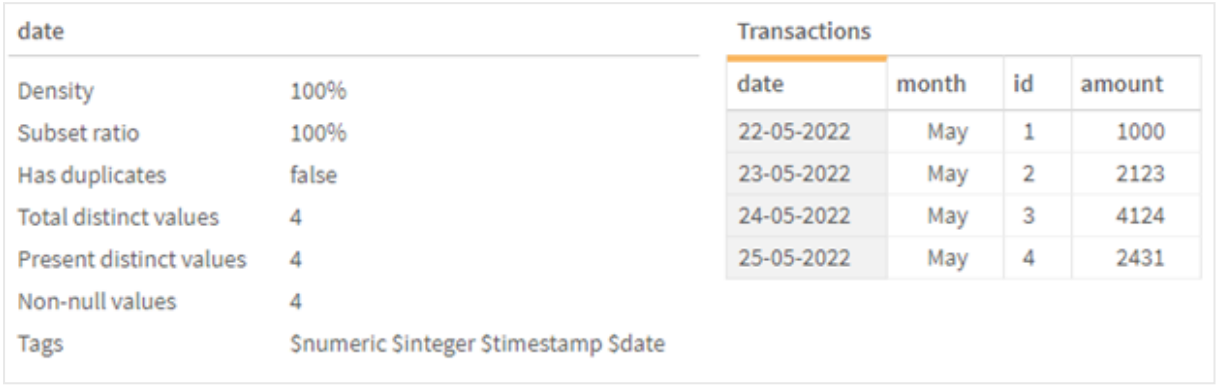


Detta kan lösas genom att aktivera systemvariabeln `DateFormat`:

```
// SET DateFormat='DD/MM/YYYY';
```

Ta bort de dubbla snedstrecket och ladda om data.

Förhandsvisning av `Transactions`-tabellen. Notera taggar för `date`-fältet som indikerar att textinmatningsdata implicit har konverterats till ett datum eller tidsmarkör.



date

Density	100%
Subset ratio	100%
Has duplicates	false
Total distinct values	4
Present distinct values	4
Non-null values	4
Tags	$numeric $integer $timestamp $date

Transactions

date	month	id	amount
22-05-2022	May	1	1000
23-05-2022	May	2	2123
24-05-2022	May	3	4124
25-05-2022	May	4	2431

DayNames

Det definierade formatet ersätter konventionen med veckodagsnamn som ställts in av dina regionala inställningar.

Syntax:

```
DayNames
```

Vid modifiering av variabeln krävs ett semikolon ; för att separera de individuella värdena.

Exempel på DayName-funktioner

Funktionsexempel	Resultatdefinition
`Set DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';`	Denna användning av DayNames-funktionen definierar dagnamn i deras förkortade form.
`Set DayNames='M;Tu;W;Th;F;Sa;Su';`	Denna användning av DayNames-funktionen definierar dagnamn efter deras första bokstav.

`DayNames`-funktionen används ofta i kombination med följande funktioner:

Relaterade funktioner

Funktion	Interaktion
weekday (page 1188)	Skriptfunktion för att returnera `DayNames` som fältvärden.
Date (page 1374)	Skriptfunktion för att returnera `DayNames` som fältvärden.
LongDayNames (page 262)	Långformvärden för .DayNames

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Standardsystemvariabler

Laddningsskript och resultat

Översikt

I det här exemplet är datumen i datamängden inställda på formatet MM/DD/YYYY.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum, som kommer att laddas in i en tabell med namnet, `Transactions`.
- Ett `date`-fält.
- Standarddefinitionen för `DayNames`.

Laddningsskript

```
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';

Transactions:
LOAD
date,
WeekDay(date) as dayname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `dayname`

Skapa det här måttet:

```
sum(amount)
```

Resultattabell

datum	veckodagsnamn	summa(belopp)
01/01/2022	lör	1000
2022-02-01	tis	2123
2022-03-01	tis	4124
04/01/2022	fre	2431

I laddningsskript används `WeekDay`-funktionen med `date`-fältet som tillhandahållet argument. I resultattabellen visar utdata från denna `WeekDay`-funktion veckodagarna i formatet för `DayNames`-definitionen.

Exempel 2 – Byt systemvariabel

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik. Samma datauppsättning och scenario som det första exemplet används.

Men i början av skriptet ändras `DayNames`-definitionen för att använda de förkortade veckodagarna på afrikaans.

Laddningsskript

```
SET DayNames='Ma;Di;Wo;Do;Vr;Sa;So';

Transactions:
Load
date,
WeekDay(date) as dayname,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `dayname`

Skapa det här måttet:

```
sum(amount)
```

Resultattabell

datum	veckodagsnamn	summa(belopp)
01/01/2022	Lö	1000
2022-02-01	Ti	2123
2022-03-01	Ti	4124
04/01/2022	Vr	2431

I resultattabellen visar utdata från denna `WeekDay`-funktion veckodagarna i formatet för `DayNames`-definitionen.

Det är viktigt att komma ihåg att om språket för `DayNames` ändras som det har varit i det här exemplet, skulle `LongDayNames` fortfarande innehålla veckodagar på engelska. Detta skulle också behöva ändras om båda variablerna används i applikationen.

Exempel 3 – Datumfunktion

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum, som kommer att laddas in i en tabell med namnet, `Transactions`.
- Ett `date`-fält.
- Standarddefinitionen för `DayNames`.

Laddningsskript

```
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';

Transactions:
Load
date,
Date(date,'WWW') as dayname,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `dayname`

Skapa det här måttet:

```
sum(amount)
```

Resultattabell

datum	dayname	summa(belopp)
01/01/2022	lör	1000
2022-02-01	tis	2123
2022-03-01	tis	4124
04/01/2022	fre	2431

Standarddefinitionen för `DayNames` används. I laddningsskript används `Date`-funktionen med `date`-fältet som det första argumentet. Det andra argumentet är `WWW`. Denna formatering omvandlar resultatet till de värden som lagras i `DayNames` -definitionen. Detta visas i utdata för resultattabellen.

DecimalSep

Den definierade decimalavgränsaren ersätter decimalsymbolen som ställts in av dina regionala inställningar.

Qlik Sense tolkar automatiskt text som siffror när ett igenkännbart nummermönster påträffas. Systemvariablerna `ThousandSep` och `DecimalSep` bestämmer sammansättningen av mönstren som används när text tolkas som siffror. `ThousandSep`- och `DecimalSep`-variablerna anger standardmönstret för nummerformat när du visualiserar numeriskt innehåll i gränssnittsdiagram och tabeller. Det vill säga, det påverkar direkt **nummerformateringsalternativen** för alla gränssnittsuttryck.

Om vi antar ett tusendelsvgränsaren är `,` och decimalavgränsaren är `.`, är det här exempel på mönster som implicit skulle konverteras till numeriska ekvivalenta värden:

```
0,000.00
```

```
0000.00
```

```
0,000
```

Det här är exempel på mönster som skulle förbli oförändrade som text; det vill säga inte konverterat till numeriskt:

```
0.000,00
```

```
0,00
```

Syntax:

`DecimalSep`

Exempel på funktioner

Exempel	Resultat
`Set DecimalSep='.';`	Ställer in "." som decimalavgränsare.
`Set DecimalSep=',';`	Ställer in "," som decimalavgränsare.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel – Effekt av att ställa in nummeravgränsare på olika indata

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning av summor och datum med summorna inställda i olika formatmönster.
- En tabell med namnet `Transactions`.
- `DecimalSep`-variabeln som är inställd på `.`.
- `ThousandSep`-variabeln som är inställd på `,`.
- `delimiter`-variabeln som är inställd på "|" som avgränsare för de olika fälten på en rad.

Laddningsskript

```
Set ThousandSep=',';
Set DecimalSep='.';

Transactions:
Load date,
id,
amount as amount
Inline
[
```

```
date|id|amount
01/01/2022|1|1.000-45
01/02/2022|2|23.344
01/03/2022|3|4124,35
01/04/2022|4|2431.36
01/05/2022|5|4,787
01/06/2022|6|2431.84
01/07/2022|7|4132.5246
01/08/2022|8|3554.284
01/09/2022|9|3.756,178
01/10/2022|10|3,454.356
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `amount`.

Skapa det här måttet:

```
=sum(amount)
```

Resultattabell

Amount	=Sum(amount)
Totalvärden	20814.7086
1.000-45	
3.756,178	
4124,35	
23.344	23.344
2431.36	2431,36
2431.84	2431.84
3,454.356	3454.356
3554.284	3554.284
4132.5246	4132.5246
4,787	4787

Alla värden som inte tolkas som nummer förblir text och är vänsterjusterade som standard. Alla framgångsrikt konverterade värden högerjusteras och behåller det ursprungliga inmatningsformatet.

Uttryckskolumnen visar den numeriska motsvarigheten, vilken som standard är formaterad med endast en decimalavgränsare `.`. Detta kan åsidosättas med hjälp av inställningen **Talformatering** i rullgardinsmenyn i uttryckskonfigurationen.

FirstWeekDay

Heltal som definierar vilken dag som ska användas som den första dagen i veckan.

Syntax:

```
FirstWeekDay
```

Måndag är den första dagen i veckan enligt ISO 8601, den internationella standarden för representation av datum och tider. Måndag används också som första dag i veckan i ett antal länder, till exempel i Storbritannien, Frankrike, Tyskland och Sverige.

Men i andra länder, som i USA och Kanada, anses söndagen vara början på veckan.

I Qlik Sense hämtas de regionala inställningarna när applikationen skapas, och de motsvarande inställningarna lagras i skriptet som miljövariabler.

En nordamerikansk applikationsutvecklare får ofta `Set FirstWeekDay=6;` i skriptet, vilket motsvarar söndag. En europeisk applikationsutvecklare får ofta `Set FirstWeekDay=0;` i skriptet, vilket motsvarar måndag.

Värden kan därefter anges för FirstWeekDay

Värde	Dag
0	Måndag
1	Tisdag
2	Onsdag
3	Torsdag
4	Fredag
5	Lördag
6	Söndag

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel:

Om du vill ha ISO-inställningar för veckor och veckonummer, ska du se till att ha följande i skriptet:

```
Set FirstWeekDay=0;  // Monday as first week day
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

Om du vill ha USA-inställningar, ska du se till att ha följande i skriptet:

```
Set FirstWeekDay=6;  // Sunday as first week day
Set BrokenWeeks=1;
Set ReferenceDay=1;
```

Exempel 1 – Använda standardvärdet (skript)

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

I det här exemplet använde laddningsskriptet standardvärdet för Qlik Sense-systemvariabeln, `FirstWeekDay=6`. Dessa data innehåller data för de första 14 dagarna under 2020.

Laddningsskript

```
// Example 1: Load Script using the default value of FirstWeekDay=6, i.e. Sunday

SET FirstWeekDay = 6;

Sales:
LOAD
        date,
        sales,
        week(date) as week,
        weekday(date) as weekday
Inline [
date,sales
01/01/2021,6000
01/02/2021,3000
01/03/2021,6000
01/04/2021,8000
01/05/2021,5000
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
01/11/2020,7000
01/12/2020,7000
01/13/2020,7000
01/14/2020,7000
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `week`
- `weekday`

Resultattabell

Datum	vecka	veckodag
2021-01-01	1	ons
02/01/2021	1	tors
03/01/2021	1	fre
04/01/2021	1	lör
05/01/2021	2	sön
01/06/2020	2	mån
01/07/2020	2	tis
01/08/2020	2	ons
01/09/2020	2	tors
01/10/2020	2	fre
01/11/2020	2	lör
01/12/2020	3	sön
01/13/2020	3	mån
01/14/2020	3	tis

Eftersom standardinställningarna används ställs systemvariabeln `FirstWeekDay` in som 6. I resultattabellen kan varje ny vecka ses börja på söndagen (den 5 och 12 januari).

Exempel 2 – Ändra variabeln FirstWeekDay (skript)

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

I det här exemplet ingår de första 14 dagarna i datan. I början av det här skriptet ställer vi in variabeln `FirstWeekDay` som 3.

Laddningsskript

```
// Example 2: Load Script setting the value of FirstWeekDay=3, i.e. Thursday

SET FirstWeekDay = 3;
```

```
Sales:
LOAD
       date,
       sales,
       week(date) as week,
       weekday(date) as weekday
Inline [
date,sales
01/01/2021,6000
01/02/2021,3000
01/03/2021,6000
01/04/2021,8000
01/05/2021,5000
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
01/11/2020,7000
01/12/2020,7000
01/13/2020,7000
01/14/2020,7000
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `week`
- `weekday`

Resultattabell

Datum	vecka	veckodag
2021-01-01	52	ons
02/01/2021	1	tors
03/01/2021	1	fre
04/01/2021	1	lör
05/01/2021	1	sön
01/06/2020	1	mån
01/07/2020	1	tis
01/08/2020	1	ons
01/09/2020	2	tors
01/10/2020	2	fre

Datum	vecka	veckodag
01/11/2020	2	lör
01/12/2020	2	sön
01/13/2020	2	mån
01/14/2020	2	tis

Eftersom systemvariabeln `FirstWeekDay` är inställd som 3, kommer den första dagen i varje vecka att vara en torsdag. I resultattabellen kan varje ny vecka ses börja på torsdag (den 2 och 9 januari).

LongDayNames

Det definierade formatet ersätter konventionen med långa veckodagsnamn i regionala inställningar.

Syntax:

```
LongDayNames
```

Följande exempel på `LongDayNames`-funktionen definierar dagsnamn i sin helhet:

```
Set LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';
```

Vid modifiering av variabeln krävs ett semikolon `;`för att separera de individuella värdena.

`LongDayNames`-funktionen kan användas i kombination med *Date (page 1374)*-funktionen som returnerar `DayNames` som fältvärden.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Standard för systemvariabler

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum, som kommer att laddas in i en tabell med namnet, `Transactions`.
- Ett `date`-fält.
- Standarddefinitionen för `LongDayNames`.

Laddningsskript

```
SET LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';

Transactions:
LOAD
date,
Date(date,'WWWW') as dayname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `dayname`

Skapa det här måttet:

```
=sum(amount)
```

Resultattabell

datum	veckodagsnamn	=sum(amount)
01/01/2022	Lördag	1000
2022-02-01	Tisdag	2123
2022-03-01	Tisdag	4124
04/01/2022	Fredag	2431

I laddningsskriptet, för att skapa ett fält som kallas `dayname`, använder du `Date`-funktionen med `date`-fältet som det första argumentet. Det andra argumentet i funktionen är `WWWW`-formateringen.

Genom att använda denna formatering omvandlas värdena från det första argumentet till motsvarande heldagsnamn som ställs in i `LongDayNames`-variabeln. I resultattabellen visar fältvärdena för vårt skapade `dayname`-fält detta.

Exempel 2 – Byt systemvariabel

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Samma datauppsättning och scenario som det första exemplet används. Men i början av skriptet ändras `LongDayNames`-definitionen för att använda de veckodagarna på spanska.

Laddningsskript

```
SET LongDayNames='Lunes;Martes;Miércoles;Jueves;Viernes;Sábado;Domingo';

Transactions:
LOAD
date,
Date(date,'WWWW') as dayname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `dayname`

Skapa det här måttet:

```
=sum(amount)
```

Resultattabell

datum	veckodagsnamn	=sum(amount)
01/01/2022	Lördag	1000
2022-02-01	Tisdag	2123
2022-03-01	Tisdag	4124
04/01/2022	Fredag	2431

I laddningsskriptet ändras `LongDayNames`-variabeln för att lista veckodagarna på spanska.

Du kan sedan skapa ett fält som kallas, `dayname`, vilket är `Date`-funktionen som används med `date`-fältet som det första argumentet.

Det andra argumentet i funktionen är `WWWW`-formateringen. Genom att använda denna formatering omvandlar Qlik Sense värdena från det första argumentet till motsvarande heldagsnamn som ställs in i `LongDayNames`-variabeln.

I resultattabellen visar fältvärdena för vårt skapade `dayname`-fält veckodagarna skrivna på spanska och i sin helhet.

LongMonthNames

Det definierade formatet ersätter konventionen med långa månadsnamn i regionala inställningar.

Syntax:

```
LongMonthNames
```

När variabeln modifieras måste ; användas för att separera de individuella värdena.

Följande exempel på LongMonthNames -funktionen definierar dagsnamn i sin helhet:

```
Set
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';
```

`LongMonthNames`-funktionen används ofta i kombination med följande funktioner:

Relaterade funktioner

Funktion	Interaktion
Date (page 1374)	Skriptfunktion för att returnera `DayNames` som fältvärden.
LongDayNames (page 262)	Långformvärden för .DayNames

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Standardsystemvariabler

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum läses in i en tabell med namnet `Transactions`.
- Ett `date`-fält.
- Standarddefinitionen för `LongMonthNames`.

Laddningsskript

```
SET
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';

Transactions:
Load
date,
Date(date,'MMMM') as monthname,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,1000.45
01/02/2022,2,2123.34
01/03/2022,3,4124.35
01/04/2022,4,2431.36
01/05/2022,5,4787.78
01/06/2022,6,2431.84
01/07/2022,7,2854.83
01/08/2022,8,3554.28
01/09/2022,9,3756.17
01/10/2022,10,3454.35
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner.

- `date`
- `monthname`

Skapa det här måttet

```
=sum(amount)
```

Resultattabell

datum	monthname	summa(belopp)
01/01/2022	januari	1000,45
01/02/2022	januari	2123,34
01/03/2022	januari	4124,35
01/04/2022	januari	2431,36
01/05/2022	januari	4787,78
01/06/2022	januari	2431,84
01/07/2022	januari	2854,83
2022-01-08	januari	3554,28
09/01/2022	januari	3756,17
01/10/2022	januari	3454,35

Standarddefinitionen för `LongMonthNames` används. I laddningsskriptet, för att skapa ett fält som kallas `month`, använder du `Date`-funktionen med `date`-fältet som det första argumentet. Det andra argumentet i funktionen är `MMMM`-formateringen.

Genom att använda denna formatering omvandlar Qlik Sense värdena från det första argumentet till motsvarande hela månadsnamn som ställs in i `LongMonthNames`-variabeln. I resultattabellen visar fältvärdena för vårt skapade `month`-fält detta.

Exempel 2 – Byt systemvariabel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum läses in i en tabell med namnet `Transactions`.
- Ett `date`-fält.
- `LongMonthNames`-variabeln som är modifierad för att använda de förkortade veckodagarna på spanska.

Laddningsskript

```
SET
LongMonthNames='Enero;Febrero;Marzo;Abril;Mayo;Junio;Julio;Agosto;Septiembre;OctubreNoviembre;
Diciembre';

Transactions:
LOAD
date,
```

```
Date(date,'MMMM') as monthname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till `sum(amount)` och dessa fält som dimensioner:

- `date`
- `monthname`

Skapa det här måttet:

```
=sum(amount)
```

Resultattabell

datum	monthname	summa(belopp)
01/01/2022	Enero	1000,45
01/02/2022	Enero	2123,34
01/03/2022	Enero	4124,35
01/04/2022	Enero	2431,36
01/05/2022	Enero	4787,78
01/06/2022	Enero	2431,84
01/07/2022	Enero	2854,83
2022-01-08	Enero	3554,28
09/01/2022	Enero	3756,17
01/10/2022	Enero	3454,35

I laddningsskriptet ändras `LongMonthNames`-variabeln för att lista namnen på månaderna på spanska. Därefter, för att skapa ett fält som kallas `monthname` används `Date` -funktionen tillsammans med `date`-fältet som det första argumentet. Det andra argumentet i funktionen är `MMMM`-formateringen.

Genom att använda denna formatering omvandlar Qlik Sense värdena från det första argumentet till motsvarande hela månadsnamn som ställs in i `LongMonthNames`-variabeln. I resultattabellen visar fältvärdena för vårt skapade `monthname` -fält månadsnamnenskrivna på spanska.

MoneyDecimalSep

Den definierade decimalavgränsaren ersätter decimalsymbolen för valuta som ställts in av dina regionala inställningar.

Som standard visar Qlik Sense siffror och text annorlunda i tabelldiagram. Siffror är högerjusterade och text är vänsterjusterad. Detta underlättar att hitta konverteringsproblem som gäller text-till-nummer. Alla tabeller på denna sida som visar Qlik Sense resultat använder denna formatering.

Syntax:

```
MoneyDecimalSep
```

Qlik Sense-program kommer att tolka textfält som följer denna formatering som monetära värden. Textfältet måste innehålla den valutasymbol som har definierats i systemvariabeln `MoneyFormat`. `MoneyDecimalSep` är särskilt användbar vid hantering av datakällor som tagits emot från flera olika regionala inställningar.

I följande exempel visas en möjlig användning av systemvariabeln `MoneyDecimalSep`:

```
Set MoneyDecimalSep='.';
```

Den här funktionen används ofta tillsammans med följande funktioner:

Relaterade funktioner

Funktion	Interaktion
`MoneyFormat`	Om tolkning av textfält förekommer kommer `MoneyFormat`-symbolen att användas som en del av tolkningen. För talformat kommer `MoneyFormat`-formateringen att användas av Qlik Sense i diagramobjekt.
`MoneyThousandSep`	Om tolkning av textfält förekommer måste även `MoneyThousandSep`-funktionen vara uppfylld.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – notation med punkt som MoneyDecimalSep (.)

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning läses in i en tabell med namnet `Transactions`.
- Tillhandahållna data som har sitt monetära fält i textformat med punkt "." som decimalavgränsare. Alla poster har också "$"-symbolen som prefix, bortsett från den sista posten som har "£"-symbolen som prefix.

Tänk på att systemvariabeln `MoneyFormat` definierar dollar "$" som standardvaluta.

Laddningsskript

```
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat='$###0.00;-$###0.00';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$14.41'
01/02/2022,2,'$2,814.32'
01/03/2022,3,'$249.36'
01/04/2022,4,'$24.37'
01/05/2022,5,'$7.54'
01/06/2022,6,'$243.63'
01/07/2022,7,'$545.36'
01/08/2022,8,'$3.55'
01/09/2022,9,'$3.436'
01/10/2022,10,'£345.66'
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:`amount`.

Lägg till följande mått:

- `isNum(amount)`
- `sum(amount)`

Granska resultaten nedan som endast visar att alla dollartecken "$" har tolkats korrekt.

Resultattabell

belopp	=isnum(amount)	=Sum(amount)
Totalvärden	0	$3905.98
£345.66	0	$0.00
$3.436	-1	$3.44
$3.55	-1	$3,55
$7.54	-1	$7,54
$14.41	-1	$14,41
$24.37	-1	$24,37
243.63	-1	243,63 $
$249.36	-1	249,36 $
$545.36	-1	545,36 $
$2,814.32	-1	2 814,32 $

I resultattabellen ovan framgår hur `amount`-fältet har tolkats korrekt för alla värden med dollartecken ($) som prefix, medan `amount` med pundtecken (£) som prefix inte har konverterats till ett monetärt värde.

Exempel 2 – notation med komma som MoneyDecimalSep (.)

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning laddas till en tabell som heter `Transactions`.
- Tillhandahållna data som har sitt monetära fält i textformat med komma "," som decimalavgränsare. Alla poster har också "$"-symbolen som prefix, bortsett från den sista posten som felaktigt använder punkt "." som decimalavgränsare.

Tänk på att systemvariabeln `MoneyFormat` definierar dollar "$" som standardvaluta.

Laddningsskript

```
SET MoneyThousandSep='.';
SET MoneyDecimalSep=',';
SET MoneyFormat='$###0.00;-$###0.00';

Transactions:
Load
```

```
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$14,41'
01/02/2022,2,'$2.814,32'
01/03/2022,3,'$249,36'
01/04/2022,4,'$24,37'
01/05/2022,5,'$7,54'
01/06/2022,6,'$243,63'
01/07/2022,7,'$545,36'
01/08/2022,8,'$3,55'
01/09/2022,9,'$3,436'
01/10/2022,10,'$345.66'
];
```

Resultat

Stycketext för resultat.

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:`amount`.

Lägg till följande mått:

- `isNum(amount)`
- `sum(amount)`

Granska nedanstående resultat som visar att alla värden tolkas korrekt, bortsett från det belopp där punkt "." används som decimalavgränsare. I det här fallet skulle ett komma ha använts i stället.

Resultattabell

belopp	=isNum(amount)	=Sum(amount)
Totalvärden	0	3 905,98 USD
345,66 USD	0	$0.00
3 436 USD	-1	3,44 USD
3,55 USD	-1	3,55 $
7,54 $	-1	7,54 $
14,41 $	-1	14,41 $
$24,37	-1	$24.37
$243,63	-1	$243.63
$249,36	-1	$249,36
$545,36	-1	$545,36
$2 814,32	-1	$2 814,32

MoneyFormat

I den här systemvariabeln definieras det formatmönster som används av Qlik vid automatisk översättning av text till tal, där talet har en monetär symbol som prefix. Den definierar också hur mått vars talformategenskaper är inställda till "Pengar" visas i diagramobjekt.

Symbolen som definieras som del av formatmönstret i systemvariabeln `MoneyFormat` ersätter valutasymbolen som ställts in i dina regionala inställningar.

Som standard visar Qlik Sense siffror och text annorlunda i tabelldiagram. Siffror är högerjusterade och text är vänsterjusterad. Detta underlättar att hitta konverteringsproblem som gäller text-till-nummer. Alla tabeller på denna sida som visar Qlik Sense resultat använder denna formatering.

Syntax:

```
MoneyFormat
```

```
Set MoneyFormat='$ #,##0.00; ($ #,##0.00)';
```

Den här formateringen kommer att visas i diagramobjekt när `Number Formatting`-egenskapen för ett numeriskt fält har ställts in till `Money`. När numeriska textfält tolkas av Qlik Sense kommer dessutom Qlik Sense att tolka detta fält som ett monetärt värde om valutasymbolen i textfältet matchar den symbol som har definierats i `MoneyFormat`-variabeln.

Den här funktionen används ofta tillsammans med följande funktioner:

Relaterade funktioner

Funktion	Interaktion
MoneyDecimalSep (page 269)	För talformat kommer `MoneyDecimalSep` att användas i fältformatering av objekt.
MoneyThousandSep (page 277)	För talformat kommer `MoneyThousandSep` att användas i fältformatering av objekt.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – MoneyFormat

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller en datauppsättning som laddas in i en tabell med namnet `Transactions`. Standarddefinitionen för `MoneyFormat`-variabeln används.

Laddningsskript

```
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat='$###0.00;-$###0.00';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,$10000000441
01/02/2022,2,$21237492432
01/03/2022,3,$249475336
01/04/2022,4,$24313369837
01/05/2022,5,$7873578754
01/06/2022,6,$24313884663
01/07/2022,7,$545883436
01/08/2022,8,$35545828255
01/09/2022,9,$37565817436
01/10/2022,10,$3454343566
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `amount`

Lägg till måttet:

```
=Sum(amount)
```

Välj **Pengar** under **Talformat** för att konfigurera `Sum(amount)` som monetärt värde.

Resultattabell

datum	belopp	=Sum(amount)
Totalvärden		$165099674156.00
01/01/2022	$10000000441	$10000000441.00
01/02/2022	$21237492432	$21237492432.00
01/03/2022	$249475336	$249475336.00
01/04/2022	$24313369837	$24313369837.00
01/05/2022	$7873578754	$7873578754.00
01/06/2022	$24313884663	$24313884663.00
01/07/2022	$545883436	$545883436.00
2022-01-08	$35545828255	$35 545 828 255,00
09/01/2022	$37565817436	$37 565 817 436,00
01/10/2022	$3454343566	$3 454 343 566,00

Standarddefinitionen för `MoneyFormat` används. Detta ser ut som följer: `$###0.00;-$###0.00`. I resultattabellen visar formatet för `amount`-fältet valutasymbolen och decimalpunkt och det definierade antalet decimaler.

Exempel 2 – MoneyFormat med tusentalsavgränsare och blandade inmatningsformat

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med blandade inmatningsformat som laddas till en tabell som heter `Transactions`, med tusentalsavgränsare och decimalavgränsare inlagda.
- `MoneyFormat`-definitionen har modifierats genom att komma används som tusentalsavgränsare.
- På en av dataraderna som använder kommma som tusentalsavgränsaren har kommana placerats på fel ställen. Observera hur detta belopp ges som text och inte kan tolkas som ett tal.

Laddningsskript

```
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat = '$#,##0.00;-$#,##0.00';
```

```
Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$10,000,000,441.45'
01/02/2022,2,'$212,3749,24,32.23'
01/03/2022,3,$249475336.45
01/04/2022,4,$24,313,369,837
01/05/2022,5,$7873578754
01/06/2022,6,$24313884663
01/07/2022,7,$545883436
01/08/2022,8,$35545828255
01/09/2022,9,$37565817436
01/10/2022,10,$3454343566
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `amount`

Lägg till måttet:

```
=Sum(amount)
```

Välj **Pengar** under **Talformat** för att konfigurera `Sum(amount)` som monetärt värde.

Resultattabell

datum	Belopp	=Sum(amount)
Totalvärden		$119,548,811,911.90
01/01/2022	$10,000,000,441.45	$10 000 000 441,45
01/02/2022	$212,3749,24,32.23	$0.00
01/03/2022	$249475336.45	$249,475,336.45
01/04/2022	$24	$24.00
01/05/2022	$7873578754	$7,873,578,754.00
01/06/2022	24 313 884 663 USD	24 313 884 663,00 USD
01/07/2022	545 883 436 USD	545 883 436,00 USD
2022-01-08	35 545 828 255 USD	35 545 828 255,00 USD
09/01/2022	37 565 817 436 USD	37 565 817 436,00 USD
01/10/2022	3 454 343 566 USD	3 454 343 566,00 USD

I början av skriptet ändras systemvariabeln `MoneyFormat` så att komma används som tusentalsseparator. I Qlik Sense-tabellen framgår hur denna avgränsare ingår i formateringen. Dessutom har raden med felaktig avgränsare inte tolkats korrekt och är fortfarande text. Detta är varför den inte tas med i summeringen av beloppet.

MoneyThousandSep

Den definierade tusentalsavgränsaren ersätter siffergrupperingssymbolen för valuta som ställts in av dina regionala inställningar.

Som standard visar Qlik Sense siffror och text annorlunda i tabelldiagram. Siffror är högerjusterade och text är vänsterjusterad. Detta underlättar att hitta konverteringsproblem som gäller text-till-nummer. Alla tabeller på denna sida som visar Qlik Sense resultat använder denna formatering.

Syntax:

`MoneyThousandSep`

Qlik Sense-program kommer att tolka textfält som följer denna formatering som monetära värden. Textfältet måste innehålla den valutasymbol som har definierats i systemvariabeln `MoneyFormat`. `MoneyThousandSep` är särskilt användbar vid hantering av datakällor som tagits emot från flera olika regionala inställningar.

I följande exempel visas en möjlig användning av systemvariabeln `MoneyThousandSep`:

```
Set MoneyDecimalSep=',';
```

Den här funktionen används ofta tillsammans med följande funktioner:

Relaterade funktioner

Funktion	Interaktion
`MoneyFormat`	Om tolkning av textfält förekommer kommer `MoneyFormat`-symbolen att användas som en del av tolkningen. För talformat kommer `MoneyFormat`-formateringen att användas av Qlik Sense i diagramobjekt.
`MoneyDecimalSep`	Om tolkning av textfält förekommer måste även `MoneyDecimalSep`-funktionen vara uppfylld.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – notation med komma som MoneyDecimalSep (.)

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning laddas till en tabell som heter `Transactions`.
- Tillhandahållna data som har sitt monetära fält i textformat med komma "," som tusentalsavgränsare. Alla poster har också en "$"-symbol som prefix.

Tänk på att systemvariabeln `MoneyFormat` definierar dollar "$" som standardvaluta.

Laddningsskript

```
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat='$###0.00;-$###0.00';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$10,000,000,441'
01/02/2022,2,'$21,237,492,432'
01/03/2022,3,'$249,475,336'
01/04/2022,4,'$24,313,369,837'
01/05/2022,5,'$7,873,578,754'
01/06/2022,6,'$24,313,884,663'
01/07/2022,7,'$545,883,436'
01/08/2022,8,'$35,545,828,255'
01/09/2022,9,'$37,565,817,436'
01/10/2022,10,'$3.454.343.566'
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:`amount`.

Lägg till följande mått:

- `isNum(amount)`
- `sum(amount)`

Granska nedanstående resultat. Tabellen demonstrerar en korrekt tolkning av alla värden som använder notation med komma "," som tusentalsavgränsare.

`amount`-fältet har tolkats korrekt för alla värden, bortsett från ett värde som använde punkt "." som tusentalsavgränsare.

Resultattabell

belopp	=isNum(amount)	=Sum(amount)
Totalvärden	0	$161645330590.00
$3.454.343.566	0	$0.00
$249,475,336	-1	$249475336.00
$545,883,436	-1	$545883436.00
$7,873,578,754	-1	$7873578754.00
$10,000,000,441	-1	$10000000441.00
$21,237,492,432	-1	$21237492432.00
$24,313,369,837	-1	$24313369837.00
$24,33,884,663	-1	$24313884663.00
$35,545,828,255	-1	$35 545 828 255,00
$37,565,817,436	-1	$37565817436.00

Exempel 2 – notation med punkt som MoneyDecimalSep (.)

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning laddas till en tabell som heter `Transactions`.
- Tillhandahållna data som har sitt monetära fält i textformat med punkt "." som tusentalsavgränsare. Alla poster har också en "$"-symbol som prefix.

Tänk på att systemvariabeln `MoneyFormat` definierar dollar "$" som standardvaluta.

Laddningsskript

```
SET MoneyThousandSep='.';
SET MoneyDecimalSep=',';
```

```
SET MoneyFormat='$###0.00;-$###0.00';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$10.000.000.441'
01/02/2022,2,'$21.237.492.432'
01/03/2022,3,'$249.475.336'
01/04/2022,4,'$24.313.369.837'
01/05/2022,5,'$7.873.578.754'
01/06/2022,6,'$24.313.884.663'
01/07/2022,7,'$545.883.436'
01/08/2022,8,'$35.545.828.255'
01/09/2022,9,'$37.565.817.436'
01/10/2022,10,'$3,454,343,566'
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:`amount`.

Lägg till följande mått:

- `isNum(amount)`
- `sum(amount)`

Granska nedanstående resultat som demonstrerar en korrekt tolkning av alla värden som använder notation med punkt "." som tusentalsavgränsare.

`amount`-fältet har tolkats korrekt för alla värden, bortsett från ett värde som använde komma "," som tusentalsavgränsare.

Resultattabell

belopp	=isNum(amount)	=Sum(amount)
Totalvärden	0	$161645330590.00
$3,545,343,566	0	$0.00
$249.475.336	-1	$249475336.00
$545.883.436	-1	545883436.00
$7.873.578.754	-1	$7873578754.00
$10.000.000.441	-1	$10000000441.00
$21.237.492.432	-1	$21237492432.00
$24.313.884.663	-1	$24313884663.00

belopp	=isNum(amount)	=Sum(amount)
$24.313.884.663	-1	$24313884663.00
$35.545.828.255	-1	$35545828255.00
$37.565.817.436	-1	$37565817436.00

MonthNames

Det definierade formatet ersätter konventionen med månadsnamn i regionala inställningar.

Syntax:

```
MonthNames
```

När variabeln modifieras måste ; användas för att separera de individuella värdena.

Exempel på funktioner

Exempel	Resultat
`Set MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';`	Denna användning av `MonthNames`-funktionen definierar månadsnamn på engelska i deras förkortade form.
`Set MonthNames='Enero;Feb;Marzo;Abr;Mayo;Jun;Jul;Agosto;Set;Oct;Nov;Dic';`	Denna användning av `MonthNames` -funktionen definierar månadsnamn på spanska i deras förkortade form.

`MonthNames`-funktionen kan användas i kombination med följande funktioner:

Relaterade funktioner

Funktion	Interaktion
month (page 1025)	Skriptfunktion för att värden definierade i `MonthNames` som fältvärden.
Date (page 1374)	Skriptfunktion för att värden definierade i `MonthNames` som fältvärden baserat på ett formateringsargument som angetts.
LongMonthNames (page 265)	Långformvärden för MonthNames

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Standardsystemvariabler

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum läses in i en tabell med namnet `Transactions`.
- Ett `date`-fält.
- Standarddefinitionen för `MonthNames`.

Laddningsskript

```
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
LOAD
date,
Month(date) as monthname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000.45
01/02/2022,2,2123.34
01/03/2022,3,4124.35
01/04/2022,4,2431.36
01/05/2022,5,4787.78
01/06/2022,6,2431.84
01/07/2022,7,2854.83
01/08/2022,8,3554.28
01/09/2022,9,3756.17
01/10/2022,10,3454.35
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `monthname`

Skapa det här måttet:

```
=sum(amount)
```

Resultattabell

datum	monthname	summa(belopp)
01/01/2022	jan	1000.45
01/02/2022	jan	2123.34
01/03/2022	jan	4124.35
01/04/2022	jan	2431.36
01/05/2022	Jan	4787.78
01/06/2022	Jan	2431.84
01/07/2022	Jan	2854.83
2022-01-08	Jan	3554.28
09/01/2022	Jan	3756.17
01/10/2022	Jan	3454.35

Standarddefinitionen för `MonthNames` används. I laddningsskriptet används `Month`-funktionen med `date`-fältet som tillhandahållet argument.

I resultattabellen visar utdata från denna `Month`-funktion årets månader i formatet för `MonthNames`-definitionen.

Exempel 2 – Byt systemvariabel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum läses in i en tabell med namnet `Transactions`.
- Ett `date`-fält.
- `MonthNames` -variabeln som är modifierad för att använda de förkortade månadsnamnen på spanska.

Laddningsskript

```
                            Set
MonthNames='Enero;Feb;Marzo;Abr;Mayo;Jun;Jul;Agosto;Set;Oct;Nov;Dic';

Transactions:
LOAD
date,
month(date) as month,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `monthname`

Skapa det här måttet:

```
=sum(amount)
```

Resultattabell

datum	månadsnamn	summa(belopp)
01/01/2022	Januari	1000.45
01/02/2022	Januari	2123,34
01/03/2022	Januari	4124,35
01/04/2022	Januari	2431,36
01/05/2022	Januari	4787.78
01/06/2022	Januari	2431.84
01/07/2022	Januari	2854,83
2022-01-08	Januari	3554,28
09/01/2022	Januari	3756,17
01/10/2022	Januari	3454.35

I laddningsskriptet ändras först `MonthNames`-variabeln för att lista namnen på månaderna i förkortad version på spanska. `Month`-funktionen används med `date`-fältet som tillhandahållet argument.

I resultattabellen visar utdata från denna `Month`-funktion årets månader i formatet för `MonthNames`-definitionen.

Det är viktigt att komma ihåg att om språket för `MonthNames`-variabeln ändras som i det här exemplet, så kommer `LongMonthNames`-variabeln fortfarande att innehålla namn på månader på engelska. `LongMonthNames`-variabeln måste modifieras om båda variablerna används i programmet.

Exempel 3 – Datumfunktion

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum läses in i en tabell med namnet `Transactions`.
- Ett `date`-fält.
- Standarddefinitionen för `MonthNames`.

Laddningsskript

```
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
LOAD
date,
Month(date, 'MMM') as monthname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000.45
01/02/2022,2,2123.34
01/03/2022,3,4124.35
01/04/2022,4,2431.36
01/05/2022,5,4787.78
01/06/2022,6,2431.84
01/07/2022,7,2854.83
01/08/2022,8,3554.28
01/09/2022,9,3756.17
01/10/2022,10,3454.35
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `monthname`

Skapa det här måttet:

```
=sum(amount)
```

Resultattabell

datum	månadsnamn	summa(belopp)
01/01/2022	Jan	1000.45
01/02/2022	jan.	2123.34
01/03/2022	jan.	4124.35
01/04/2022	jan.	2431.36
01/05/2022	jan	4787.78
01/06/2022	jan	2431.84
01/07/2022	Jan	2854.83
2022-01-08	Jan	3554.28
09/01/2022	Jan	3756,17
01/10/2022	jan.	3454,35

Standarddefinitionen för `MonthNames` används. I laddningsskript används `Date`-funktionen med `date`-fältet som det första argumentet. Det andra argumentet är `MMM`.

Genom att använda denna formatering omvandlar Qlik Sense värdena från det första argumentet till motsvarande månadsnamn som ställs in i `MonthNames`-variabeln. I resultattabellen visar fältvärdena för vårt skapade `month`-fält detta.

NumericalAbbreviation

Den numeriska förkortningen ställer in vilken förkortning att använda för siffrors skalprefix, till exempel M för mega eller miljon (10^6) och µ för mikro (10^{-6}).

Syntax:

`NumericalAbbreviation`

Ställ in variabeln `NumericalAbbreviation` till en sträng som innehåller en lista med förkortningsdefinitionspar, separerade med semikolon. Varje förkortningspar ska innehålla skalan (exponenten i decimalbas) och förkortningen separerad av ett kolon, till exempel `6:M` för en miljon.

Standardinställningen är `'3:k;6:M;9:G;12:T;15:P;18:E;21:Z;24:Y;-3:m;-6:µ;-9:n;-12:p;-15:f;-18:a;-21:z;-24:y'`.

Exempel:

Inställningen ändrar prefixet för tusen till t och prefixet för miljard till B. Det är användbart för ekonomiska applikationer där du kan förvänta dig förkortningar som t$, M$ och B$.

```
Set NumericalAbbreviation='3:t;6:M;9:B;12:T;15:P;18:E;21:Z;24:Y;-3:m;-6:µ;-9:n;-12:p;-15:f;-18:a;-21:z;-24:y';
```

ReferenceDay

Inställningen definierar vilken dag i januari som ska anges som referensdag för att definiera vecka 1. Den här inställningen anger med andra ord hur många dagar i vecka 1 som måste vara datum inom januari.

Syntax:

```
ReferenceDay
```

`ReferenceDay` anger hur många dagar som ingår i årets första vecka. `ReferenceDay` kan ställas in på valfritt värde mellan `1` och `7`. Alla värden utanför intervallet `1-7` tolkas som veckans mittpunkt (`4`), vilket motsvarar `ReferenceDay` inställt på `4`.

Om du inte väljer ett värde för `ReferenceDay`-inställningen kommer standardvärdet att visa `ReferenceDay=0` vilket kommer att tolkas som veckans mittpunkt (`4`), som framgår av tabellen med `ReferenceDay`-värden nedan.

`ReferenceDay`-funktionen används ofta i kombination med följande funktioner:

Relaterade funktioner

Variabel	Interaktion
BrokenWeeks (page 244)	Om Qlik Sense-applikationen använder obrutna veckor, kommer variabelinställningen `ReferenceDay` att tillämpas. Om brutna veckor används kommer dock vecka 1 att börja den 1 januari och avslutas i enlighet med inställningen för `FirstWeekDay`-variabeln och ignorera `ReferenceDay`-flaggan.
FirstWeekDay (page 257)	Heltal som definierar vilken dag som ska användas som den första dagen i veckan.

Qlik Sense gör att följande värden kan ställas in för `ReferenceDay`:

ReferenceDay-värden

Värde	Referensdag
0 (standard)	4 januari
1	1 januari
2	Januari 2
3	3 (= 3 januari)
4	4 januari
5	5 januaro
6	6 januari
7	7 januari

I följande exempel definierar `ReferenceDay = 3` 3 januari som referensdagen:

```
SET ReferenceDay=3; //(set January 3 as the reference day)
```

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel:

Om du vill ha ISO-inställningar för veckor och veckonummer, ska du se till att ha följande i skriptet:

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;    // Jan 4th is always in week 1
```

Om du vill ha USA-inställningar, ska du se till att ha följande i skriptet:

```
Set FirstWeekDay=6;
Set BrokenWeeks=1;
Set ReferenceDay=1;     // Jan 1st is always in week 1
```

Exempel 1 – Laddningsskript som använder standardvärdet ReferenceDay = 0

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- `ReferenceDay`-variabeln som är inställd på `0`.
- `BrokenWeeks`-variabeln som är inställd på `0` som tvingar applikationen att använda obrutna veckor.
- En datauppsättning med datum från slutet av 2019 till början av 2020.

Laddningsskript

```
SET BrokenWeeks = 0;
SET ReferenceDay = 0;

Sales:
LOAD
date,
sales,
week(date) as week,
weekday(date) as weekday
```

```
Inline [
date,sales
12/27/2019,5000
12/28/2019,6000
12/29/2019,7000
12/30/2019,4000
12/31/2019,3000
01/01/2020,6000
01/02/2020,3000
01/03/2020,6000
01/04/2020,8000
01/05/2020,5000
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
01/11/2020,7000
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `week`
- `weekday`

Resultattabell

datum	vecka	veckodag
12/27/2019	52	fre
12/28/2019	52	lör
12/29/2019	1	sön
12/30/2019	1	mån
12/31/2019	1	tis
01/01/2020	1	ons
01/02/2020	1	tors
01/03/2020	1	fre
01/04/2020	1	lör
01/05/2020	2	sön
01/06/2020	2	mån
01/07/2020	2	tis
01/08/2020	2	ons

datum	vecka	veckodag
01/09/2020	2	tors
01/10/2020	2	fre
01/11/2020	2	lör

Vecka 52 avslutas lördagen den 28 december. Eftersom `ReferenceDay` kräver att den 4 januari ska ingå i vecka 1, börjar därför vecka 1 den 29 december och avslutas lördagen den 4 januari.

Exempel – ReferenceDay-variabeln inställd på 5

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- `ReferenceDay`-variabeln som är inställd på `5`.
- `BrokenWeeks`-variabeln som är inställd på `0` som tvingar applikationen att använda obrutna veckor.
- En datauppsättning med datum från slutet av 2019 till början av 2020.

Laddningsskript

```
SET BrokenWeeks = 0;
SET ReferenceDay = 5;

Sales:
LOAD
date,
sales,
week(date) as week,
weekday(date) as weekday
Inline [
date,sales
12/27/2019,5000
12/28/2019,6000
12/29/2019,7000
12/30/2019,4000
12/31/2019,3000
01/01/2020,6000
01/02/2020,3000
01/03/2020,6000
01/04/2020,8000
01/05/2020,5000
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
```

```
01/11/2020,7000
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `week`
- `weekday`

Resultattabell

datum	vecka	veckodag
12/27/2019	52	fre
12/28/2019	52	lör
12/29/2019	53	sön
12/30/2019	53	mån
12/31/2019	53	tis
01/01/2020	53	ons
01/02/2020	53	tors
01/03/2020	53	fre
01/04/2020	53	lör
01/05/2020	1	sön
01/06/2020	1	mån
01/07/2020	1	tis
01/08/2020	1	ons
01/09/2020	1	tors
01/10/2020	1	fre
01/11/2020	1	lör

Vecka 52 avslutas lördagen den 28 december. `BrokenWeeks`-variabeln tvingar applikationen att använda obrutna veckor. Referensdagsvärdet på `5` kräver att den 5 januari inkluderas i vecka 1.

Detta är dock åtta dagar efter utgången av vecka 52 föregående år. Därför börjar vecka 53 den 29 december och avslutas den 4 januari. Vecka 1 börjar söndagen den 5 januari.

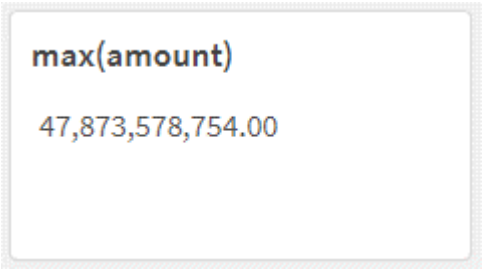
ThousandSep

Tusentalsavgränsaren som definieras ersätter operativsystemets tusentalsavgränsare (regionala inställningar).

Syntax:

```
ThousandSep
```

Qlik Sense-objekt som använder `ThousandSep`-variabeln (med tusentalsavgränsare)



Qlik Sense applikationer tolkar textfält som följer denna formatering som tal. Den här formateringen kommer att visas i diagramobjekt när **talformat**egenskapen för ett numeriskt fält har ställts in till **Tal**.

`ThousandSep` är användbart vid hantering av datakällor som tas emot från flera regionala inställningar.

> *Om `ThousandSep`-variabeln ändras efter att objekt redan har skapats och formaterats i programmet kommer användaren att behöva omformatera alla relevanta fält genom att avmarkera egenskapen **Tal** i **Talformat** och sedan markera den igen.*

I följande exempel visas möjliga användningar av systemvariabeln `ThousandSep`:

```
Set ThousandSep=','; //(for example, seven billion will be displayed as: 7,000,000,000)


Set ThousandSep=' '; //(for example, seven billion will be displayed as: 7 000 000 000)
```

Dessa ämnen kan hjälpa dig att arbeta med den här funktionen:

Relaterade ämnen

Avsnitt	Beskrivning
DecimalSep (page 255)	När tolkning av textfält förekommer måste även inställningarna för decimalavgränsare som tillhandahålls av den här funktionen också efterföljas. För talformat kommer **DecimalSep** att användas av Qlik Sense vid behov.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Standardsystemvariabler

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning laddas till en tabell som heter `Transactions`.
- Standarddefinitionen för `ThousandSep`-variabeln används.

Laddningsskript

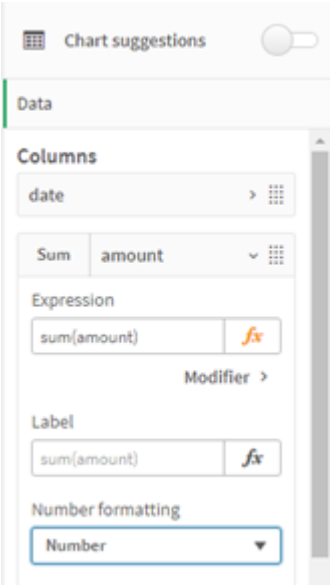
```
Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,10000000441
01/02/2022,2,21237492432
01/03/2022,3,41249475336
01/04/2022,4,24313369837
01/05/2022,5,47873578754
01/06/2022,6,24313884663
01/07/2022,7,28545883436
01/08/2022,8,35545828255
01/09/2022,9,37565817436
01/10/2022,10,3454343566
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:`date`.
2. Lägg till följande mått:
   `=sum(amount)`
3. Välj måttet i egenskapspanelen under **Data**.
4. Under **Talformat**, väljer du **Tal**.

Justera talformat för ett diagrammått



Resultattabell

datum	=sum(amount)
01/01/2022	10,000,000,441.00
01/02/2022	21,237,492,432.00
01/03/2022	41,249,475,336.00
01/04/2022	24,313,369,837.00
01/05/2022	47,873,578,754.00
01/06/2022	24,313,884,663.00
01/07/2022	28,545,883,436.00
2022-01-08	35,545,828,255.00
01/09/2022	37,565,817,436.00
01/10/2022	3,454,343,566.00

I det här exemplet används standarddefinitionen för `ThousandSep`, som är inställd till kommaformat (","). I resultattabellen visar formatet för beloppsfältet ett komma mellan grupperingar av tusental.

Exempel 2 – Byt systemvariabel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning från första exemplet som läses in i en tabell som heter `Transactions`.
- Ändring av `ThousandSep`-definition när ett skript startas så att ett "*"-tecken i tusentalsavgränsaren. Det här är ett extremt exempel och används enbart för att demonstrera variabelns funktion.

Den ändring som används i det här exemplet är extrem och används vanligen inte, men visas är för att demonstrera variabelns funktion.

Laddningsskript

```
SET ThousandSep='*';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,10000000441
01/02/2022,2,21237492432
01/03/2022,3,41249475336
01/04/2022,4,24313369837
01/05/2022,5,47873578754
01/06/2022,6,24313884663
01/07/2022,7,28545883436
01/08/2022,8,35545828255
01/09/2022,9,37565817436
01/10/2022,10,3454343566
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:`date`.
2. Lägg till följande mått:
   `=sum(amount)`
3. Välj måttet i egenskapspanelen under **Data**.
4. Under **Talformat**, väljer du **Anpassat**.

Resultattabell

datum	=sum(amount)
01/01/2022	10*000*000*441.00
01/02/2022	21*237*492*432.00

datum	=sum(amount)
01/03/2022	41*249*475*336.00
01/04/2022	24*313*369*837.00
01/05/2022	47*873*578*754.00
01/06/2022	24*313*884*663.00
01/07/2022	28*545*883*436.00
2022-01-08	35*545*828*255.00
09/01/2022	37*565*817*436.00
01/10/2022	3*454*343*566.00

I början av skriptet ändras systemvariabeln `ThousandSep` till en "*". I resultattabellen framgår att i formatet för beloppsfältet visas en "*" mellan grupperingar av tusental.

Exempel 3 – Datatolkning

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning laddas till en tabell som heter `Transactions`.
- Data som har sitt talfält i textformat med komma som tusentalsavgränsare.
- Standardsystemvariabeln `ThousandSep` används.

Laddningsskript

```
Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'10,000,000,441'
01/02/2022,2,'21,492,432'
01/03/2022,3,'4,249,475,336'
01/04/2022,4,'24,313,369,837'
01/05/2022,5,'4,873,578,754'
01/06/2022,6,'313,884,663'
01/07/2022,7,'2,545,883,436'
01/08/2022,8,'545,828,255'
01/09/2022,9,'37,565,817,436'
```

```
01/10/2022,10,'3,454,343,566'
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:`date`.
2. Lägg till följande mått:
   `=sum(amount)`
3. Välj måttet i egenskapspanelen under **Data**.
4. Under **Talformat**, väljer du **Tal**.
5. Lägg till följande mått för att utvärdera huruvida beloppsfältet är ett numeriskt värde:
   `=isnum(amount)`

Resultattabell

datum	=sum(amount)	=isnum(amount)
01/01/2022	10,000,000,441.00	-1
01/02/2022	21,492,432.00	-1
01/03/2022	4,249,475,336.00	-1
01/04/2022	24,313,369,837.00	-1
01/05/2022	4,873,578,754.00	-1
01/06/2022	313,884,663.00	-1
01/07/2022	2,545,883,436.00	-1
01/08/2022	545,828,255.00	-1
09/01/2022	37,565,817,436.00	-1
01/10/2022	3*454*343*566.00	-1

När data har laddats kan vi se att Qlik Sense har tolkat beloppsfältet som ett numeriskt värde, eftersom data följer `ThousandSep`-variabeln. Detta framgår av `isnum()`-funktionen, som utvärderar varje post till -1, or `TRUE`.

I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.

TimeFormat

Formatet som definieras ersätter operativsystemets format för tid (regionala inställningar).

Syntax:

```
TimeFormat
```

Exempel:

```
Set TimeFormat='hh:mm:ss';
```

TimestampFormat

Formatet som definieras ersätter operativsystemets format för datum och tid (regionala inställningar).

Syntax:

```
TimestampFormat
```

Exempel:

I följande exempel används *1983-12-14T13:15:30Z* som tidsmarkörsdata för att visa resultaten för olika **SET TimestampFormat**-satser. Datumformatet som används är **YYYYMMDD** och tidsformatet är **h:mm:ss TT**. Datumformatet anges i **SET DateFormat**-satsen och tidsformatet anges i **SET TimeFormat**-satsen längst upp i dataladdningsskriptet.

Resultat

Exempel	Resultat
`SET TimestampFormat='YYYYMMDD';`	19831214
`SET TimestampFormat='M/D/YY hh:mm:ss[.fff]';`	12/14/83 13:15:30
`SET TimestampFormat='DD/MM/YYYY hh:mm:ss[.fff]';`	14/12/1983 13:15:30
`SET TimestampFormat='DD/MM/YYYY hh:mm:ss[.fff] TT';`	14/12/1983 13:15:30
`SET TimestampFormat='YYYY-MM-DD hh:mm:ss[.fff] TT';`	1983-12-14 01:15:30

Exempel: laddningsskript

Exempel: laddningsskript

I det första laddningsskriptet används *SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT'*. I det andra laddningsskriptet har formatet för tidsmarkören ändrats till *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'*. De olika resultaten visar hur **SET TimeFormat**-satsen fungerar med olika tidsformat.

Tabellen nedan visar datauppsättningen som används i laddningsskripten som följer. Tabellens andra kolumn innehåller formaten för varje tidsmarkör i datauppsättningen. De första fem tidsmarkörerna följer ISO 8601-normen, men det gör inte den sjätte.

Datauppsättning

Tabellen visar de tidsdata som används och formatet för varje tidsmarkör i datauppsättningen.

transaction_timestamp	time data format
2018-08-30	YYYY-MM-DD
20180830T193614.857	YYYYMMDDhhmmss.sss
20180830T193614.857+0200	YYYYMMDDhhmmss.sss±hhmm
2018-09-16T12:30-02:00	YYYY-MM-DDhh:mm±hh:mm
2018-09-16T13:15:30Z	YYYY-MM-DDhh:mmZ
9/30/18 19:36:14	M/D/YY hh:mm:ss

Gå till **Skriptredigeraren** och skapa ett nytt delavsnitt. Lägg sedan till exempelskriptet och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i din applikation för att se resultatet.

Laddningsskript

```
SET FirstWeekDay=0;
SET BrokenWeeks=1;
SET ReferenceDay=0;
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';
SET DateFormat='YYYYMMDD';
SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT';

Transactions:
Load
*,
Timestamp(transaction_timestamp, 'YYYY-MM-DD hh:mm:ss[.fff]') as LogTimestamp
;

Load * Inline [
transaction_id, transaction_timestamp, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 2018-08-30, 12423.56, 23, 0,2038593, L, Red
3751, 20180830T193614.857, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180830T193614.857+0200, 15.75, 1, 0.22, 5646471, S, blue
3753, 2018-09-16T12:30-02:00, 1251, 7, 0, 3036491, l, Black
3754, 2018-09-16T13:15:30Z, 21484.21, 1356, 75, 049681, xs, Red
3755, 9/30/18 19:36:14, -59.18, 2, 0.333333333333333, 2038593, M, Blue
];
```

Resultat

Qlik Sense-tabell som visar resultaten för TimestampFormat-tolkningsvariabeln som används i laddningsskriptet. Den sista tidsmarkören i datauppsättningen returnerar inte ett korrekt datum.

transaction_id	transaction_timestamp	LogTimeStamp
3750	2018-08-30	2018-08-30 00:00:00
3751	20180830T193614.857	2018-08-30 19:36:14
3752	20180830T193614.857+0200	2018-08-30 17:36:14
3753	2018-09-16T12:30-02:00	2018-09-16 14:30:00
3754	2018-09-16T13:15:30Z	2018-09-16 13:15:30
3755	9/30/18 19:36:14	-

Nästa laddningsskript använder samma datauppsättning. Det använder dock *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'* för att matcha formatet i den sjätte tidsmarkören, som inte följer ISO 8601-normen.

Gå till **Skriptredigeraren** och ersätt det tidigare exempelskriptet med skriptet nedan och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i din applikation för att se resultatet.

Laddningsskript

```
SET FirstWeekDay=0;
SET BrokenWeeks=1;
SET ReferenceDay=0;
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';
SET DateFormat='YYYYMMDD';
SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]';

Transactions:
Load
*,
Timestamp(transaction_timestamp, 'YYYY-MM-DD hh:mm:ss[.fff]') as LogTimestamp
;

Load * Inline [
transaction_id, transaction_timestamp, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 2018-08-30, 12423.56, 23, 0,2038593, L, Red
3751, 20180830T193614.857, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180830T193614.857+0200, 15.75, 1, 0.22, 5646471, S, blue
3753, 2018-09-16T12:30-02:00, 1251, 7, 0, 3036491, l, Black
3754, 2018-09-16T13:15:30Z, 21484.21, 1356, 75, 049681, xs, Red
3755, 9/30/18 19:36:14, -59.18, 2, 0.333333333333333, 2038593, M, Blue
];
```

Resultat

Qlik Sense-tabell som visar resultaten för TimestampFormat-tolkningsvariabeln som används i laddningsskriptet.

transaction_id	transaction_timestamp	LogTimeStamp
3750	2018-08-30	2018-08-30 00:00:00
3751	20180830T193614.857	2018-08-30 19:36:14
3752	20180830T193614.857+0200	2018-08-30 17:36:14
3753	2018-09-16T12:30-02:00	2018-09-16 14:30:00
3754	2018-09-16T13:15:30Z	2018-09-16 13:15:30
3755	9/30/18 19:36:14	2018-09-16 19:36:14

4.10 Variabler i Direkt upptäckt

Systemvariabler i Direkt upptäckt

DirectCacheSeconds

Du kan ställa in en cachelagringsgräns till Direkt upptäckt-frågeresultaten för visualiseringar. När den här tidsgränsen har uppnåtts rensar Qlik Sense cacheminnet när nya Direkt upptäckt-frågor ställs. Qlik Sense ställer frågor om urval till datakällan och skapar cacheminnet igen för den designerade tidsgränsen. Resultatet för varje kombination av urval cachelagras fristående. Det innebär att cacheminnet uppdateras fristående för varje urval, så att ett urval uppdaterar cacheminnet enbart för de valda fälten, och ett andra urval uppdaterar cacheminnet för de berörda fälten. Om det andra urvalet omfattar fält som uppdaterades i det första urvalet uppdateras de inte i cacheminnet igen om inte cachelagringsgränsen har uppnåtts.

Direkt upptäckt-cacheminnet gäller inte för **Tabell**-visualiseringar. Tabellurvalen frågar datakällan varje gång.

Gränsvärdet måste anges i sekunder. Standardgränsen för cachelagring är 1 800 sekunder (30 minuter).

Det värde som används för **DirectCacheSeconds** är det värde som är inställt vid tidpunkten då **DIRECT QUERY**-satsen exekveras. Värdet kan inte ändras under körning.

Exempel:

```
SET DirectCacheSeconds=1800;
```

DirectConnectionMax

Du kan göra asynkrona, parallella anrop mot databasen med hjälp av funktionen för anslutningspoolning. Laddningsskriptsyntaxen för konfigurering av anslutningspoolning ser ut som följer:

```
SET DirectConnectionMax=10;
```

Den numeriska inställningen anger det maximala antalet databaskopplingar som Direct Discovery-koden ska använda vid uppdatering av ett ark. Standardinställningen är 1.

Den här variabeln bör användas försiktigt. Att ställa in den till ett större värde än 1 brukar orsaka problem vid anslutning till Microsoft SQL Server.

DirectUnicodeStrings
Direct Discovery kan stödja urval av utökade Unicode-data med hjälp av SQL -standardformatet för stränglitteraler med utökad teckensträng (N'<extended string>') som krävs av vissa databaser (framför allt SQL Server). Användning av denna syntax kan aktiveras för Direkt upptäckt med hjälp av skriptvariabeln **DirectUnicodeStrings**.

Om variabeln anges till "true" kan du använda den standardmässiga breda ANSI-teckenmarkören "N" framför stränglitteralerna. Alla databaser stöder inte den här standarden. Standardinställningen är "false".

DirectDistinctSupport
Om ett **DIMENSION**-fältvärde väljs i ett Qlik Sense-objekt skapas en fråga för källdatabasen. Om frågan kräver gruppering använder Direkt upptäckt nyckelordet **DISTINCT** för att välja enbart unika värden. Vissa databaser kräver dock nyckelordet **GROUP BY**. Ställ in **DirectDistinctSupport** på 'false' för att generera **GROUP BY** i stället för **DISTINCT** i frågor för unika värden.

```
SET DirectDistinctSupport='false';
```

Om DirectDistinctSupport är inställt på true används **DISTINCT**. Om det inte är inställt är standardbeteendet att använda **DISTINCT**.

DirectEnableSubquery
I scenarier med hög kardinalitet och flera tabeller är det möjligt att generera delfrågor i SQL-frågan istället för att generera en lång IN-sats. Detta aktiveras genom att ställa in **DirectEnableSubquery** på'true'. Standardvärdet är 'false'.

*Om du har aktiverat **DirectEnableSubquery** kan du inte ladda tabeller som inte är i läget Direkt upptäckt.*

```
SET DirectEnableSubquery='true';
```

Variabler för Teradata-querybanding

Teradataquerybanding är en funktion som gör för företagsapplikationer kan samarbeta med den underliggande Teradata-databasen för att ge bättre redovisning, prioritering och hantering av arbetsbelastning. Med hjälp av query banding kan man omsluta en fråga med metadata, exempelvis med autentiseringsuppgifter för användare.

Det finns två varianter. Båda är strängar som utvärderas och skickas till databasen.

SQLSessionPrefix
Den här strängen skickas när en koppling till databasen skapas.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR SESSION;';
```

Om t.ex. **OSuser()** returnerar *WA\sbt*, utvärderas detta till `SET QUERY_BAND = 'Who=WA\sbt;' FOR SESSION;`, vilket i sin tur skickas till databasen när kopplingen skapas.

SQLQueryPrefix
Strängen skickas för varje enskild fråga.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR TRANSACTION;';
```

Direkt upptäckt-teckenvariabler

DirectFieldColumnDelimiter
Du kan ställa in det tecken som används som fältavgränsare i **Direct Query**-satser för databaser som kräver ett annat tecken än komma som fältavgränsare. Det angivna tecknet måste omges av enkla citattecken i **SET**-satsen.

```
SET DirectFieldColumnDelimiter= '|'
```

DirectStringQuoteChar
Du kan ange ett tecken som ska användas för att förse strängarna i en genererad fråga med citattecken. Standardvärdet är enkla citattecken. Det angivna tecknet måste omges av enkla citattecken i **SET**-satsen.

```
SET DirectStringQuoteChar= '"';
```

DirectIdentifierQuoteStyle
Du kan ange att icke-ANSI-citatmarkering av identifierare ska användas i genererade frågor. För närvarande är den enda tillgängliga icke-ANSI-citatmarkeringen GoogleBQ. Standard är ANSI. Versaler, gemener och versaler/gemener blandat kan användas (ANSI, ansi, Ansi).

```
SET DirectIdentifierQuoteStyle="GoogleBQ";
```
ANSI-citatmarkering används exempelvis i följande **SELECT**-sats:

```
SELECT [Quarter] FROM [qvTest].[sales] GROUP BY [Quarter]
```
Om **DirectIdentifierQuoteStyle** är inställd på "GoogleBQ" skulle **SELECT**-satsen använda citatmarkering på följande sätt:

```
SELECT [Quarter] FROM [qvTest.sales] GROUP BY [Quarter]
```

DirectIdentifierQuoteChar
Du kan ange ett tecken för att kontrollera citatmarkeringen av identifierare i en genererad fråga. Detta kan ställas in till antingen ett tecken (exempelvis ett dubbelt citattecken) eller två (exempelvis ett par hakparenteser). Standardvärdet är dubbla citattecken.

```
SET DirectIdentifierQuoteChar='[]';

SET DirectIdentifierQuoteChar='``';

SET DirectIdentifierQuoteChar=' ';

SET DirectIdentifierQuoteChar='""';
```

DirectTableBoxListThreshold

När Direct Discovery-fält används i en **Tabell**-visualisering, ställs ett gränsvärde in för att begränsa det antal rader som visas. Standardgränsen är 1 000 poster. Standardinställningen för gränsvärdet kan ändras med hjälp av variabeln **DirectTableBoxListThreshold** i laddningsskriptet. Exempel:

```
SET DirectTableBoxListThreshold=5000;
```

Gränsinställningen gäller enbart för **Tabell**-visualiseringar som innehåller Direkt upptäckt-fält. **Tabell**-visualiseringar som innehåller endast minnesfält begränsas inte av inställningen **DirectTableBoxListThreshold**.

Inga fält visas i **Tabell**-visualiseringen förrän urvalet har färre poster än gränsinställningen.

Direkt upptäckt-variabler för tolkning av tal

DirectMoneyDecimalSep

Den definierade decimalavgränsaren ersätter decimalsymbolen för valuta i SQL-satsen som genereras för att ladda data med Direkt upptäckt. Det här tecknet måste matcha det tecken som används i **DirectMoneyFormat**.

Standardvärdet är `'.'`.

Exempel:

```
Set DirectMoneyDecimalSep='.';
```

DirectMoneyFormat

Den definierade symbolen ersätter valutaformatet i den SQL-sats som genereras för att ladda data med hjälp av Direkt upptäckt. Valutasymbolen för tusentalsavgränsaren ska inte tas med.

Standardvärdet är `'#.0000'`.

Exempel:

```
Set DirectMoneyFormat='#.0000';
```

DirectTimeFormat

Det definierade tidsformatet ersätter tidsformatet i den SQL-sats som genereras för att ladda data med hjälp av Direkt upptäckt.

Exempel:

```
Set DirectTimeFormat='hh:mm:ss';
```

DirectDateFormat

Det definierade datumformatet ersätter datumformatet i den SQL-sats som genereras för att ladda data med hjälp av Direkt upptäckt.

Exempel:

```
Set DirectDateFormat='MM/DD/YYYY';
```

DirectTimeStampFormat
Det definierade formatet ersätter datum- och tidsformatet i den SQL-sats som genereras för att ladda data med hjälp av Direkt upptäckt.

Exempel:

```
Set DirectTimestampFormat='M/D/YY hh:mm:ss[.fff]';
```

4.11 Felvariabler

Värdena i alla felvariablerna kvarstår efter skriptexekveringen. Den första variabeln, ErrorMode, är indata från användaren. De sista tre är utdata från Qlik Sense med information om fel i skriptet.

Felvariabler – en översikt

Varje variabel beskrivs ytterligare efter översikten. Du kan även klicka på namnet på variabeln i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika variabeln.

Se onlinehjälpen till Qlik Sense för mer information om variabeln

ErrorMode
Denna felvariabel avgör hur Qlik Sense ska reagera när ett fel upptäcks under skriptexekveringen.

```
ErrorMode
```

ScriptError
Denna felvariabel returnerar den senast exekverade skriptsatsens felkod.

```
ScriptError
```

ScriptErrorCount
Denna felvariabel returnerar det totala antalet satser som har orsakat fel under den aktuella skriptexekveringen. Denna variabel återställs alltid till 0 när skriptexekvering påbörjas.

```
ScriptErrorCount
```

ScriptErrorList
Denna felvariabel innehåller en konkatenerad lista över de skriptfel som har inträffat under den senaste exekveringen. Felen avgränsas med en radmatning.

```
ScriptErrorList
```

ErrorMode

Denna felvariabel avgör hur Qlik Sense ska reagera när ett fel upptäcks under skriptexekveringen.

Syntax:

```
ErrorMode
```

Argument:

Argument

Argument	Beskrivning
ErrorMode=1	Standardinställningen. Skriptexekveringen avbryts och användaren ombeds att agera (non-batch mode).
ErrorMode =0	Qlik Sense ignorerar felet och fortsätter skriptexekveringen vid nästa skriptsats.
ErrorMode =2	Qlik Sense utlöser ett felmeddelande "Skriptexekveringen misslyckades..." genast, utan att användaren uppmanas att agera i förväg.

Exempel:

```
set ErrorMode=0;
```

ScriptError

Denna felvariabel returnerar den senast exekverade skriptsatsens felkod.

Syntax:

```
ScriptError
```

Denna variabel återställs till 0 efter varje lyckad skriptexekvering. Om ett fel inträffar kommer en intern Qlik Sense-felkod att användas. Felkoden består av ett dualt värde, ett numeriskt värde och ett textvärde: Följande felkoder finns:

Skriptfelkoder

Felkod	Beskrivning
0	Inget fel. Text för dualt värde är tom.
1	Allmänt fel.
2	Syntaxfel.
3	Allmänt ODBC-fel.
4	Allmänt OLE DB-fel.
5	Generellt fel från anpassad databas.
6	Allmänt XML-fel.
7	Allmänt HTML-fel.
8	Det går inte att hitta filen.

Felkod	Beskrivning
9	Det går inte att hitta databasen.
10	Det går inte att hitta tabellen.
11	Det går inte att hitta fältet.
12	Fel filformat.
16	Semantiskt fel.

Exempel:

```
set ErrorMode=0;

LOAD * from abc.qvf;

if ScriptError=8 then

exit script;

//no file;

end if
```

ScriptErrorCount

Denna felvariabel returnerar det totala antalet satser som har orsakat fel under den aktuella skriptexekveringen. Denna variabel återställs alltid till 0 när skriptexekvering påbörjas.

Syntax:

```
ScriptErrorCount
```

ScriptErrorList

Denna felvariabel innehåller en konkatenerad lista över de skriptfel som har inträffat under den senaste exekveringen. Felen avgränsas med en radmatning.

Syntax:

```
ScriptErrorList
```

5 Skriptuttryck

Uttryck kan användas i både **LOAD**-satser och **SELECT**-satser. De uttryck och funktioner som beskrivs här gäller endast **LOAD**-satsen, och inte **SELECT**-satsen, eftersom den tolkas av ODBC-drivrutinen och inte av Qlik Sense. De flesta ODBC-drivrutiner kan dock tolka ett antal av de funktioner som beskrivs nedan.

Uttryck består av funktioner, fält och operatorer som kombineras i en syntax.

Alla uttryck i Qlik Sense-skriptet resulterar i ett tal och/eller en sträng, beroende på vad som är lämpligt. Logiska funktioner och operatorer returnerar 0 för False och -1 för True. Konverteringar från tal till textsträng eller vice versa är implicita. Logiska operatorer och funktioner tolkar 0 som False och alla andra som True.

Den allmänna syntaxen för ett uttryck är:

Allmän syntax

Uttryck	Fält	Operator
expression ::= (constant	constant	|
expression ::= (constant	fieldref	|
expression ::= (constant	operator1 expression	|
expression ::= (constant	expression operator2 expression	|
expression ::= (constant	function	|
expression ::= (constant	(expression)	)

där:

- **constant** är en sträng (en text, ett datum, en tid) inom enkla, raka citationstecken, eller ett tal. Konstanter skrivs utan tusentalsavgränsare och med decimalkomma som decimalavgränsaren.
- **fieldref** är ett fältnamn i den inlästa tabellen.
- **operator1** är en unär operator (som arbetar med ett uttryck, det till höger).
- **operator2** är en binär operator (som arbetar med två uttryck, ett på varje sida).
- **function ::= functionname(parameters)**

- **parameters ::= expression { , expression }**

Varken typen av parameter eller antalet parametrar är godtyckligt. utan beror på funktionen som används.

Uttryck och funktioner kan således nästlas efter behag, och så länge som uttrycket resulterar i ett värde som går att tolka kommer Qlik Sense inte att ge några felmeddelanden.

6 Diagramuttryck

Ett diagram (visualisering) är en kombination av funktioner, fält, matematiska operatorer (+ * / =) och andra mått. Uttryck används för att bearbeta data i applikationen i syfte att skapa ett resultat som kan ses i en visualisering. De är inte begränsade till användning i mått. Du kan bygga visualiseringar som är mer dynamiska och effektiva med uttryck för rubriker, underrubriker, fotnoter och till och med dimensioner.

Detta innebär till exempel att istället för att låta en visualiserings rubrik vara statisk text kan den skapas från ett uttryck som ger olika resultat beroende på vilka val som görs.

En detaljerad referens om skriptfunktioner och diagramfunktioner finns i Skriptsyntax och diagramfunktioner.

6.1 Definiera aggregeringens omfattning

Det finns vanligtvis två faktorer som tillsammans avgör vilka poster som används för att definiera värdet av en aggregering i ett uttryck. När vi arbetar med visualiseringar är dessa faktorer:

- Dimensionsvärde (för en aggregering i ett diagramuttryck)
- Urval

Tillsammans definierar dessa faktorer omfattningen för aggregeringen. Du kan komma att stöta på situationer där du vill att beräkningen ska bortse från urvalet, dimensionen eller bådadera. I diagramfunktioner kan du åstadkomma detta genom att använda kvalificeraren TOTAL, set-analys eller en kombination av de två. Sammanfattningsvis:

- Använda TOTAL-kvalificeraren för att bortse från eller justera den dimensionella grupperingen
- Använda set-analys för att bortse från eller justera urvalet

Aggregering: metod och beskrivning

Metod	Beskrivning
Kvalificeraren TOTAL	När du använder total-kvalificeraren i en aggregeringsfunktion ignoreras dimensionsvärdet. Total-kvalificeraren kan användas inom en aggregering för att omdefiniera aggregeringens omfattning genom att bortse från dimensionsvärdena. Aggregeringen utförs i stället på alla möjliga fältvärden. Bestämningen **TOTAL** kan följas av en lista med ett eller flera fältnamn inom vinkelparenteser. Exempel: `Sum(TOTAL <Quarter> {<Year={2013}, Quarter={'Q2'}>} Amount)` Dessa fältnamn bör ingå i en underuppsättning av diagrammets dimensioner. Vid uträkningen beaktas i detta fall alla diagrammets dimensioner utom de som ingår i listan, d.v.s. ett värde returneras för varje kombination av fältvärden i dimensionsfälten som finns i listan. Även fält som för närvarande inte utgör en dimension i ett diagram kan ingå i listan. Detta är användbart när man arbetar med grupper av dimensioner där dimensionsfälten inte är fasta. Genom att lista alla dimensioner i gruppen säkerställs att funktionen tillhandahåller konsekventa aggregeringsresultat när den hierarkiska nivån ändras. Ett exempel på hur man lägger till undantag inom hakparenteser finns på *Exempel: TOTAL-kvalificerare och set-analys (page 313)*.
Set-analys	Vid användning av set-analys inuti aggregeringen åsidosätts urvalet. Aggregeringen utförs på alla värden delat över dimensionerna.
Kvalificeraren TOTAL och set-analys	När kvalificeraren **TOTAL** och set-analys används inuti din aggregering åsidosätts alla dimensioner utom de som listas inom hakparanteser efter **TOTAL** -kvalificeraren.

Kvalificeraren ALL	När kvalificeraren **ALL**används inuti aggregeringen ignoreras urvalet och dimensionerna. Motsvarande resultat kan uppnås med set-analys-satsen {1} och kvalificeraren **TOTAL** :
	
	`=sum(All Sales)`
	
	`=sum({1} Total Sales)`

Exempel: TOTAL-kvalificeraren

I följande exempel visas hur TOTAL kan användas för att beräkna en relativ andel. Anta att Q2 har valts. Om du använder TOTAL beräknas summan av alla värden utan hänsyn till dimensionerna.

Exempel: kvalificeraren TOTAL

Year	Quarter	Sum(Amount)	Sum(TOTAL Amount)	Sum(Amount)/Sum(TOTAL Amount)
		3000	3000	100 %
2012	Q2	1700	3000	56,7%
2013	Q2	1300	3000	43,3%

*Om du vill visa talen som en procentandel går du till egenskapspanelen för det mått som du vill visa som ett procentvärde. Under **Talformat** väljer du **Tal** och som **Format** väljer du **Enkel** och något av procentformaten.*

Exempel: set-analys

I följande exempel visas hur set-analys kan användas för att göra en jämförelse mellan datauppsättningar innan något urval har gjorts. Anta att Q2 har valts. Vid användning av set-analys med set-definitionen {1} beräknas summan av alla värden utan hänsyn till urval men delat med dimensionerna.

Exempel: set-analys

Year	Quarter	Sum(Amount)	Sum({1} Amount)	Sum(Amount)/Sum({1} Amount)
		3000	10800	27,8%
2012	Q1	0	1100	0%
2012	Q3	0	1400	0%
2012	Q4	0	1800	0%
2012	Q2	1700	1700	100 %
2013	Q1	0	1000	0%
2013	Q3	0	1100	0%
2013	Q4	0	1400	0%
2013	Q2	1300	1300	100 %

Exempel: kvalificeraren TOTAL och set-analys ({1}-identifierare)

I följande exempel visas hur set-analys och kvalificeraren TOTAL kan kombineras för att göra en jämförelse mellan datauppsättningar innan något urval har gjorts och över alla dimensioner. Anta att Q2 har valts. Vid användning av set-analys med set-definitionen {1} och kvalificeraren TOTAL beräknas summan av alla värden utan hänsyn till urval och utan hänsyn till dimensionerna.

Exempel: kvalificeraren TOTAL och set-analys

Year	Quarter	Sum (Amount)	Sum({1} TOTAL Amount)	Sum(Amount)/Sum({1} TOTAL Amount)
		3000	10800	27,8%
2012	Q2	1700	10800	15,7%
2013	Q2	1300	10800	12%

Exempel: TOTAL kvalificerare och set-analys (jämförelse av data mot ett referensvärde)

I följande exempel framgår hur du kan använda set-analys och kvalificeraren TOTAL för att skapa referenskolumner i tabellen så att du enkelt kan göra jämförelser av mätvärden på radnivå. För att göra uppgifterna lättare att förstå och använda märker vi upp mätkolumnerna.

Uppgifterna från andra kvartalet 2013 används som referens för att jämföra med alla andra värden i tabellen.

Gör följande:

1. Lägg till en tabell i arket.
2. Lägg till följande fält som dimensioner:
    - *Year*
    - *Quarter*
3. Lägg till följande mått:
   `Sum(Amount)`
4. I egenskaperna för det mått som du just har lagt till ger du måttet följande etikett: *Total Sales*
5. Lägg till följande mått:
   `Sum(TOTAL {<Year={2013}, Quarter={'Q2'}>} Amount)`
6. I egenskaperna för det mått som du just har lagt till ger du måttet följande etikett: *Reference Quarter (2013-Q2) Total Sales*
7. Lägg till följande mått:
   `round((sum(Amount) / sum(total {<Year={2013}, Quarter={'Q2'}>} Amount))*100,'0.01') & '%'`
8. I egenskaperna för det mått som du just har lagt till ger du måttet följande etikett: *Rounded Quotient - Total Sales Compared to Reference Quarter (2013-Q2)*

Om inga urval görs bör din tabell se ut som följer:

Exempel: TOTAL-kvalificerare och set-analys (referenskolumn)

Year	Quarter	Total Sales	Reference Quarter (2013-Q2) Total Sales	Rounded Quotient - Total Sales Compared to Reference Quarter (2013-Q2)
2012	Q1	1100	1300	84.62%
2012	Q2	1700	1300	130.77%
2012	Q3	1400	1300	107.69%
2012	Q4	1800	1300	138.46%
2013	Q1	1000	1300	76.92%
2013	Q2	1300	1300	100.00%
2013	Q3	1100	1300	84.62%
2013	Q4	1400	1300	107.69%

Exempel: TOTAL-kvalificerare och set-analys

I följande exempel framgår hur du använder TOTAL, men med ett undantag från denna kvalificerare som anges inom hakparenteser. I exempeluttrycken används även set-analys.

Gör följande:

1. Lägg till en tabell i arket.
2. Lägg till följande fält som dimensioner:
    - *Year*
    - *Quarter*
3. Lägg till följande mått:
   `Sum(Amount)`
4. I egenskaperna för det mått som du just har lagt till ger du måttet följande etikett: *Total Sales*
5. Lägg till följande mått:
   `Sum(TOTAL <Quarter> {<Year={2013}, Quarter={'Q2'}>} Amount)`
6. I egenskaperna för det mått som du just har lagt till ger du måttet följande etikett: *Reference Quarter (2013-Q2) Total Sales*

Exempel: TOTAL-kvalificerare och set-analys

Year	Quarter	Total Sales	Reference Quarter (2013-Q2) Total Sales
2012	Q1	1100	0
2012	Q2	1700	1300
2012	Q3	1400	0
2012	Q4	1800	0

Year	Quarter	Total Sales	Reference Quarter (2013-Q2) Total Sales
2013	Q1	1000	0
2013	Q2	1300	1300
2013	Q3	1100	0
2013	Q4	1400	0

Referenskolumnens uttryck tolkas på följande sätt:

- Set-uttrycket `{<Year={2013}, Quarter={'Q2'}>}` begränsar indata till poster med en `Year` på 2013 och en `Quarter` på Q2.
- `TOTAL`-kvalificeraren säkerställer att alla dimensioner i tabellen ignoreras. Med `<Quarter>`-specificeraren görs dock ett undantag för `Quarter`-dimensionen.

Data som används i exempel på den här sidan

Data som används i exempel:

```
AggregationScope:
LOAD * inline [
Year Quarter Amount
2012 Q1 1100
2012 Q2 1700
2012 Q3 1400
2012 Q4 1800
2013 Q1 1000
2013 Q2 1300
2013 Q3 1100
2013 Q4 1400] (delimiter is ' ');
```

6.2 Set-analys

När du gör ett urval i en app definierar du en delmängd av posterna i data. Aggregeringsfunktioner, som `Sum()`, `Max()`, `Min()`, `Avg()` och `Count()` beräknas baserat på den här delmängden.

Ditt urval definierar med andra ord aggregeringens omfattning; den definierar uppsättningen med poster som beräkningarna görs på.

Set-analys gör det möjligt att definiera en omfattning som är olik den uppsättning med poster som definieras av det aktuella urvalet. Den här nya omfattningen kan även betraktas som ett alternativt urval.

Det här kan vara användbart om du vill jämföra det aktuella urvalet med ett visst värde, till exempel förra årets värde eller den globala marknadsandelen.

Set-uttryck

Uppsättningsuttryck används i aggregeringsfunktioner och inom en klammerparentes.

Exempel: Inre uppsättningsuttryck

```
Sum( {$<Year={2021}>} Sales )
```

Exempel: Yttre uppsättningsuttryck

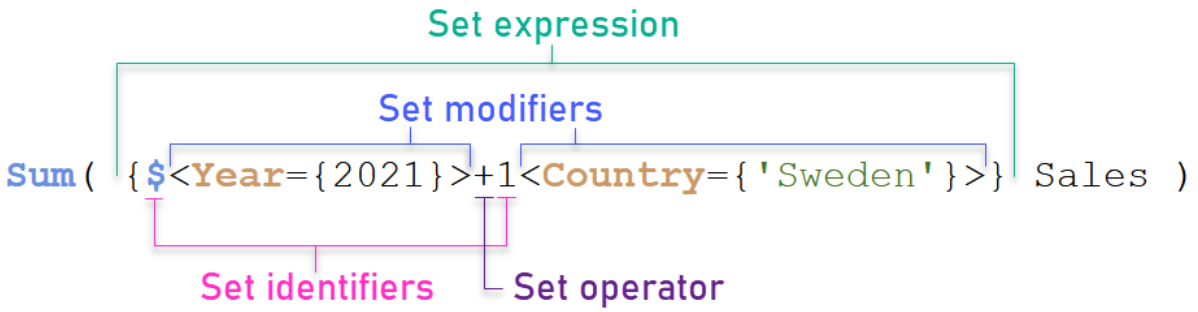
```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

Ett set-uttryck består av en kombination av följande element:

- **Identifierare**. En set-identifierare motsvarar ett urval, som definieras på en annan plats. Den motsvarar även en specifik uppsättning med poster i data. Det kan vara det aktuella urvalet, ett urval från ett bokmärke eller ett urval från ett parallellt tillstånd. Ett enkelt set-uttryck består av en enda identifierare, som t.ex. dollartecknet {$}, vilket innebär alla poster i aktuellt urval.
  Exempel: `$`, `1`, `BookMark1`, `State2`
- **Operatorer**. En set-operator kan användas till att skapa unioner, differenser eller intersektioner mellan set-identifierare. På det sättet kan du skapa en överordnad uppsättning av urvalen som definierats av set-identifierarna.
  Exempel: `+`, `-`, `*`, `/`
- **Modifierare** En set-modifierare kan läggas till i set-identifieraren för att ändra dess urval. En modifierare kan även användas på egen hand och kommer sedan att modifiera standardidentifieraren. En modifierare måste omslutas av vinkelparenteser `<…>`.
  Exempel: `<Year={2020}>`, `<Supplier={ACME}>`
- **Flaggan för att bevara tomma uppsättningar**. Ett avancerat element som kan användas i vilket yttre set-uttryck som helst - utom det sista i en kedja av yttre set-uttryck. Denna flagga åsidosätter implicit radering av uppsättningar på alla uppsättningar som definieras i dessa yttre set-uttryck, vilket förenklar konfigurationen av avancerade uttryck.
  Syntax: `&`

Elementen kombineras för att skapa set-uttryck.

Element i ett set-uttryck



Set-uttrycket ovan är till exempel byggt från aggregeringen `Sum(Sales)`.

Den första operanden returnerar försäljning för året `2021` för det aktuella urvalet, vilket indikeras med set-identifieraren `$` och modifieraren som innehåller urvalet för år `2021`. Den andra operanden returnerar `Sales` för `Sweden` och ignorerar det aktuella urvalet, vilket indikeras med set-identifieraren `1`.

Slutligen returnerar uttrycket en uppsättning som består av posterna som tillhör någon av de två set-operanderna, enligt indikation från set-operatorn +.

Exempel

Exempel som kombinerar set-uttryckselementen ovan är tillgängliga i följande ämnen:

Naturliga uppsättningar

Vanligen motsvarar ett set-uttryck både en uppsättning med poster i datamodellen och ett urval som definierar den här delmängden data. I detta fall kallas uppsättningen en naturlig uppsättning.

Set-identifierare, med eller utan set-modifierare, representerar alltid naturliga uppsättningar.

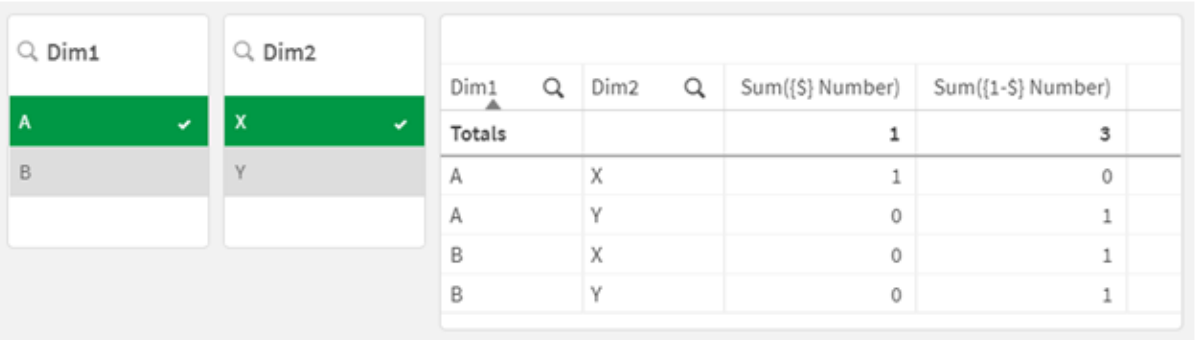
Ett set-uttryck som använder set-operatorer representerar dock även en delmängd av posterna, men kan generellt ändå inte beskrivas som ett urval med fältvärden. Ett sådant uttryck är en icke-naturlig uppsättning.

Till exempel kan uppsättningen som ges av `{1-$}` inte alltid definieras av ett urval. Det innebär att det inte är en naturlig uppsättning. Det går att visa genom att ladda följande data, lägga till dem i en tabell och sedan göra urvalen med filterpaneler.

```
Load * Inline
[Dim1, Dim2, Number
A, X, 1
A, Y, 1
B, X, 1
B, Y, 1];
```

Genom att göra urval för `Dim1` och `Dim2` får du vyn som visas i följande tabell.

Tabell med naturliga och icke-naturliga uppsättningar

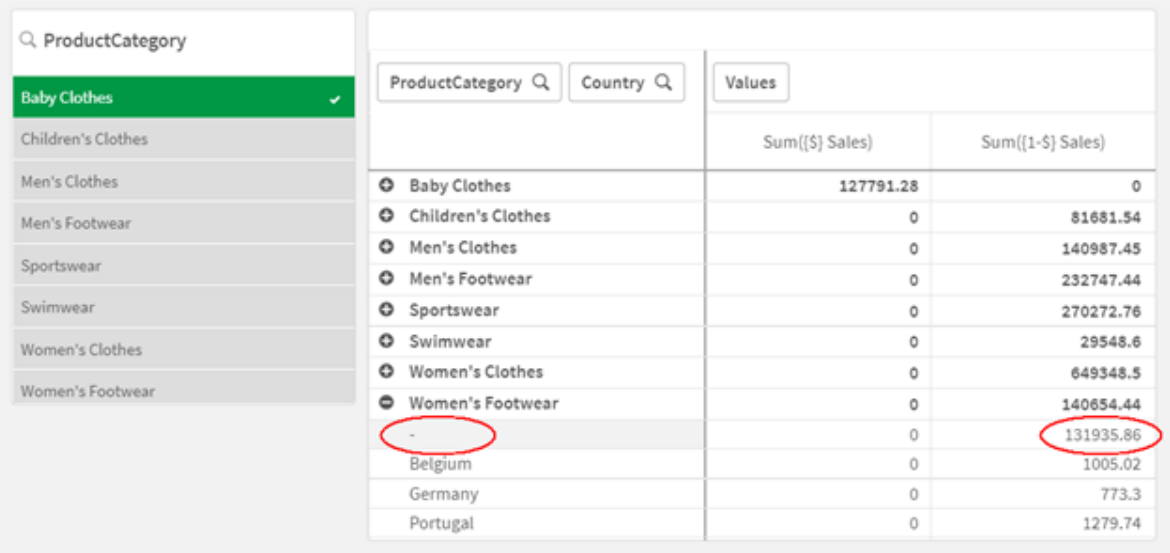


Set-uttrycket i den första åtgärden använder en naturlig uppsättning: det motsvarar urvalet som görs `{$}`.

Den andra åtgärden är annorlunda. Den använder `{1-$}`. Det går inte att göra ett urval som motsvarar den här uppsättningen, så det är en icke-naturlig uppsättning.

Den här distinktionen har ett flertal konsekvenser:

- Set-modifierare går endast att tillämpa på set-identifierare. De går inte att tillämpa på ett godtyckligt set-uttryck. Till exempel är det inte möjligt att använda ett set-uttryck som:
  `{ (BM01 * BM02) <Field={x,y}> }`
  Här betyder de vanliga (runda) parenteserna att intersektionen mellan `BM01` och `BM02` ska utvärderas innan set-modifieraren tillämpas. Skälet är att det inte finns någon uppsättning av element som kan modifieras.
- Du kan inte använda icke-naturliga uppsättningar i `P()`- och `E()`-elementfunktioner. Dessa funktioner returnerar en uppsättning av element, men det går inte att härleda en uppsättning från en icke-naturlig uppsättning.
- Ett mått som använder en icke-naturlig uppsättning kan inte alltid attribueras till rätt dimensionsvärde om datamodellen har flera tabeller. I följande diagram attribueras vissa uteslutna försäljningssiffror till rätt Country, medan andra har NULL som Country.
  Diagram med icke-naturlig uppsättning



  Huruvida tilldelningen är korrekt eller inte beror på datamodellen. I det här fallet kan talet inte tilldelas om det berör ett land som utesluts av urvalet.

Identifierare	Beskrivning
1	Representerar hela uppsättningen av alla poster i applikationen, oavsett vilka urval som har gjorts.
$	Representerar posterna i det aktuella urvalet. Set-uttrycket **{$}** är följaktligen motsvarigheten till att inte ange ett set-uttryck.
$1	Representerar tidigare urval. $2 representerar det näst föregående urvalet och så vidare.
$_1	Representerar nästa urval (framåt). $_2 representerar urvalet efter det nästföljande och så vidare.

Identifierare	Beskrivning
BM01	Du kan använda vilket bokmärkes-ID eller bokmärkesnamn du vill.
MyAltState	Du kan hänvisa till de urval som görs i ett parallellt tillstånd med hjälp av tillståndets namn.

Exempel	Resultat
sum ({1} Sales)	Returnerar den totala försäljningen för appen, oavsett urval men inte oavsett dimensionen.
sum ({$} Sales)	Returnerar försäljning för det aktuella urvalet, alltså detsamma som sum(Sales).
sum ({$1} Sales)	Returnerar försäljning för föregående urval.
sum ({BM01} Sales)	Returnerar försäljningen för bokmärket med namnet*BM01*.

Exempel	Resultat
sum({$<OrderDate = DeliveryDate>} Sales)	Returnerar försäljning för aktuellt urval där OrderDate = DeliveryDate.
sum({1<Region = {US}>} Sales)	Returnerar försäljningen för regionen USA och ignorerar det aktuella urvalet.
sum({$<Region = >} Sales)	Returnerar försäljningen för urvalet, men med urvalet i *Region* borttaget.
sum({<Region = >} Sales)	Returnerar samma resultat som exemplet ovanför. När det set som ska modifieras är utelämnat antas $.
sum({$<Year={2000}, Region={"U*"}>} Sales)	Returnerar försäljning för det aktuella urvalet, men med nya urval både i *Year* och *Region*.

Set-identifierare

En set-identifierare motsvarar en uppsättning med poster i data; antingen alla data eller en delmängd av data. Det är uppsättningen med poster som definieras av ett urval. Det kan vara det aktuella urvalet, alla data (inget urval), ett urval från ett bokmärke eller ett urval från ett parallellt tillstånd.

I det här exemplet `Sum( {$<Year = {2009}>} Sales )` är identifieraren dollartecknet: `$`. Det motsvarar det aktuella urvalet. Det representerar även alla möjliga poster. Den här uppsättningen kan sedan ändras med modifierardelen i set-uttrycket: urvalet `2009` i `Year` läggs till.

Setidentifieraren $ är detsamma som att inte ange någon setidentifierare. I exemplet ovan, är uttrycket `Sum ( {$<Year = {2009}>} Sales )` likvärdigt med `Sum( {<Year = {2009}>} Sales )`.

I ett mer komplicerat set-uttryck går det att använda två identifierare tillsammans med en operator för att bilda en union, en differens eller en skärningspunkt för de två uppsättningarna med poster.

Följande tabell visar några gemensamma identifierare.

Exempel med gemensamma identifierare

Identifierare	Beskrivning
`1`	Representerar hela uppsättningen av alla poster i applikationen, oavsett vilka urval som har gjorts.
`$` (eller ingen setidentifierare)	Representerar posterna i det aktuella urvalet i standardtillståndet. Set-uttrycket `{$}` är följaktligen vanligen motsvarigheten till att inte ange ett set-uttryck.
`$1`	Motsvarar det föregående urvalet i standardtillståndet. `$2` motsvarar det föregående urvalet utom ett och så vidare.
`$_1`	Representerar nästa urval (framåt). `$_2` representerar nästa urval utom ett och så vidare.
`BM01`	Du kan använda vilket bokmärkes-ID eller bokmärkesnamn du vill.
`AltState`	Du kan hänvisa till ett parallellt tillstånd med hjälp av tillståndets namn.
`AltState::BM01`	Ett bokmärke innehåller urval av alla tillstånd och du kan hänvisa till ett specifikt bokmärke genom att kvalificera bokmärkets namn.

Följande tabell visar exempel med olika identifierare.

Exempel med olika identifierare

Exempel	Resultat
`Sum ({1} Sales)`	Returnerar den totala försäljningen för appen, oavsett urval men inte oavsett dimensionen.
`Sum ({$} Sales)`	Returnerar försäljning för det aktuella urvalet, alltså detsamma som `Sum(Sales)`.
`Sum ({$1} Sales)`	Returnerar försäljning för föregående urval.
`Sum ({BM01} Sales)`	Returnerar försäljningen för bokmärket med namnet`BM01`.

Set-operatorer

Set-operatorer används till att inkludera, utesluta eller genomskära datauppsättningar. Alla operatorer använder uppsättningar som operander och returnerar en uppsättning som resultat.

Du kan använda set-operatorer i två olika situationer:

- För att utföra en set-åtgärd på set-identifierare, som motsvarar uppsättningar med poster i data.
- För att utföra en set-åtgärd på elementuppsättningar, på fältvärden eller i en set-modifierare.

Följande tabell visar operatorer som kan användas i set-uttryck.

Operatorer

Operator	Beskrivning
+	Union. Denna binära operation returnerar en uppsättning som består av de poster eller element som tillhör någon av de två set-operanderna.
-	Exklusion. Denna binära operation returnerar en uppsättning som består av de poster eller element som enbart tillhör den första men inte den andra av de två set-operanderna. Den returnerar dessutom det komplementära setet när det används som en unär operator.
*	Intersektion. Denna binära operation returnerar en uppsättning som består av de poster eller element som tillhör båda set-operanderna.
/	Symmetrisk skillnad (XOR). Denna binära operation returnerar en uppsättning som består av de poster eller element som tillhör endera men inte båda set-operanderna.

Följande tabell visar exempel med operatorer.

Exempel med operatorer

Exempel	Resultat
`Sum ({1-$} Sales)`	Returnerar försäljningen för allt som det aktuella urvalet exkluderar.
`Sum ({$*BM01} Sales)`	Returnerar försäljningen för skärningspunkten mellan urvalet och bokmärket #160;`BM01`.
`Sum ({-($+BM01)} Sales)`	Returnerar försäljning som exkluderas av urvalet och bokmärket `BM01`.
`Sum ({$<Year={2009}>+1<Country={'Sweden'}>} Sales)`	Returnerar försäljningen för år 2009 associerat till de aktuella urvalen och lägger till den fullständiga datauppsättningen associerad med landet `Sweden` för alla år.
`Sum ({$<Country={"S*"}+{"*land"}>} Sales)`	Returnerar försäljningen för länder som börjar med `S` eller slutar med `land`.

Set-modifierare

Set-uttryck används för att definiera omfattningen av en beräkning. Den centrala delen av set-uttrycket är set-modifieraren som specificerar ett urval. Det används till att ändra användarens urval eller urvalet i set-identifieraren och resultatet definierar en ny omfattning för beräkningen.

Set-modifieraren består av ett eller flera fältnamn, som vart och ett följs av ett urval som ska göras på just det fältet. Modifieraren omsluts av vinkelparenteser: `< >`

Exempel:

- `Sum ( {$<Year = {2015}>} Sales )`
- `Count ( {1<Country = {Germany}>} distinct OrderID )`

- `Sum ( {$<Year = {2015}, Country = {Germany}>} Sales )`

Uppsättningar av element

En elementuppsättning kan definieras med följande:

- En lista med värden
- En sökning
- En referens till ett annat fält
- En uppsättningsfunktion

Om elementuppsättningsdefinitionen utesluts kommer set-modifieraren att rensa alla urval i det här fältet. Exempel:

```
Sum( {$<Year = >} Sales )
```

Exempel: diagramuttryck för set-modifierare som baseras på elementuppsättningar

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
MyTable:
Load * Inline [
Country, Year, Sales
Argentina, 2014, 66295.03
Argentina, 2015, 140037.89
Austria, 2014, 54166.09
Austria, 2015, 182739.87
Belgium, 2014, 182766.87
Belgium, 2015, 178042.33
Brazil, 2014, 174492.67
Brazil, 2015, 2104.22
Canada, 2014, 101801.33
Canada, 2015, 40288.25
Denmark, 2014, 45273.25
Denmark, 2015, 106938.41
Finland, 2014, 107565.55
Finland, 2015, 30583.44
France, 2014, 115644.26
France, 2015, 30696.98
Germany, 2014, 8775.18
Germany, 2015, 77185.68
];
```

Diagramuttryck

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

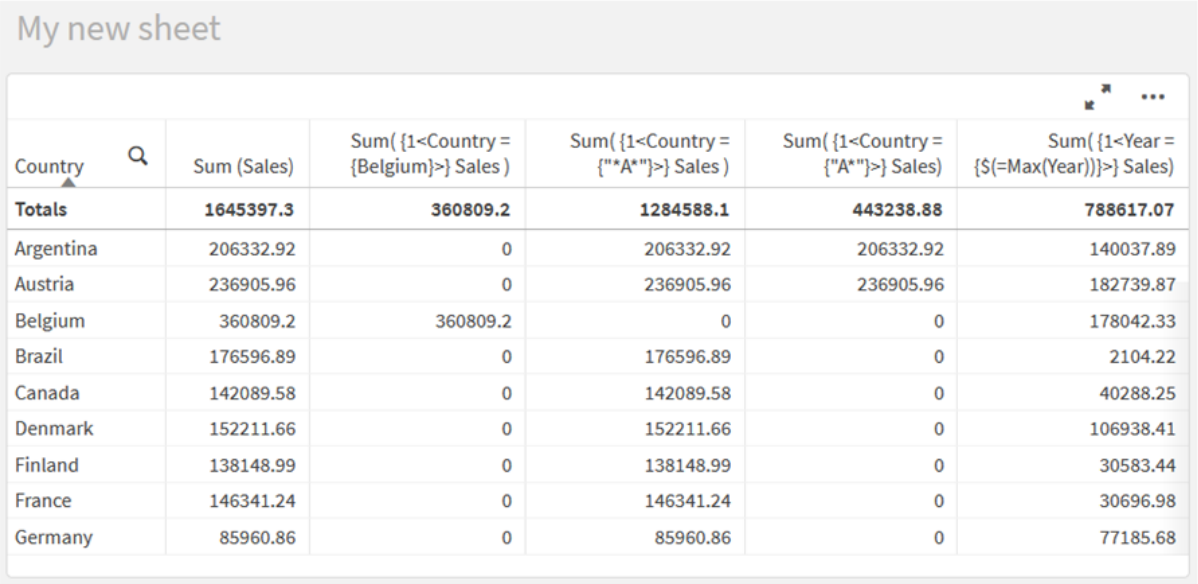
Tabell - set-modifierare baserade på elementuppsättningar

Land	Summa (Försäljning)	Summa ({1<Land={Belgien}>} Försäljning)	Summa ({1<Land={"*A*"}>} Försäljning)	Summa ({1<Land={"A*"}>} Försäljning)	Summa ({1<År={$(=Max (År))}>} Försäljning)
Totalvärden	1645397.3	360 809,2	1284588.1	443238.88	788617.07
Argentina	206332.92	0	206332.92	206332.92	140037.89
Österrike	236905.96	0	236905.96	236905.96	182739.87
Belgien	360 809,2	360 809,2	0	0	178042.33
Brasilien	176 596,89	0	176 596,89	0	2104.22
Kanada	142 089,58	0	142 089,58	0	40288.25
Danmark	152 211,66	0	152 211,66	0	106938.41
Finland	138148,99	0	138148,99	0	30583.44
Frankrike	146341,24	0	146341,24	0	30696.98
Tyskland	85960,86	0	85960,86	0	77185.68

Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Sales)`
    Summa `Sales` utan set-uttryck.
  - `Sum({1<Country={Belgium}>}Sales)`
    Välj `Belgium` och sedan summan som motsvarar `Sales`.
  - `Sum({1<Country={"*A*"}>}Sales)`
    Välj alla länder som har ett `A` och sedan summan som motsvarar `Sales`.
  - `Sum({1<Country={"A*"}>}Sales)`
    Välj alla länder som börjar på `A` och sedan summan som motsvarar `Sales`.
  - `Sum({1<Year={$(=Max(Year))}>}Sales)`
    Beräkna `Max(Year)`, vilket är `2015`, och sedan summan som motsvarar `Sales`.

Set-modifierare baserade på elementuppsättningar


My new sheet

Country	Sum (Sales)	Sum({1<Country = {Belgium}>} Sales)	Sum({1<Country = {"*A*"}>} Sales)	Sum({1<Country = {"A*"}>} Sales)	Sum({1<Year = {$(=Max(Year))}>} Sales)
Totals	**1645397.3**	**360809.2**	**1284588.1**	**443238.88**	**788617.07**
Argentina	206332.92	0	206332.92	206332.92	140037.89
Austria	236905.96	0	236905.96	236905.96	182739.87
Belgium	360809.2	360809.2	0	0	178042.33
Brazil	176596.89	0	176596.89	0	2104.22
Canada	142089.58	0	142089.58	0	40288.25
Denmark	152211.66	0	152211.66	0	106938.41
Finland	138148.99	0	138148.99	0	30583.44
France	146341.24	0	146341.24	0	30696.98
Germany	85960.86	0	85960.86	0	77185.68

Värden i listan

Det vanligaste exemplet på en elementuppsättning är en uppsättning som baseras på en lista med fältvärden som omsluts av klammerparenteser. Exempel:

- `{$<Country = {Canada, Germany, Singapore}>}`
- `{$<Year = {2015, 2016}>}`

De inre klammerparenteserna definierar uppsättningen av element. De individuella värdena separeras med komman.

Citattecken och skiftlägeskänslighet

Om värdena innehåller blanksteg eller specialtecken behöver värdena citeras. Enkla citattecken ger en exakt, skiftlägeskänslig matchning med ett enda fältvärde. Dubbla citattecken innebär en skiftlägesokänslig matchning med ett eller flera fältvärden. Exempel:

- `<Country = {'New Zealand'}>`
  Matchar endast `New Zealand`.
- `<Country = {"New Zealand"}>`
  Matchar `New Zealand`, `NEW ZEALAND` och `new zealand`.

Datum måste stå inom citattecken och använda datumformatet från fältet i fråga. Exempel:

- `<ISO_Date = {'2021-12-31'}>`
- `<US_Date = {'12/31/2021'}>`
- `<UK_Date = {'31/12/2021'}>`

Dubbla citattecken kan bytas ut av hakparenteser eller grava accenter.

Sökningar

Uppsättningar av element kan även skapas genom sökningar. Exempel:

- `<Country = {"C*"}>`
- `<Ingredient = {"*garlic*"}>`
- `<Year = {">2015"}>`
- `<Date = {">12/31/2015"}>`

Jokertecken kan användas i textsökningar: en asterisk (`*`) motsvarar ett obegränsat antal tecken och ett frågetecken (`?`) motsvarar ett enda tecken. Relationsoperatörer kan användas till att definiera numeriska sökningar.

Du bör alltid använda dubbla citattecken för sökningar. Sökningar är inte skiftlägeskänsliga.

Dollarexpansioner

Dollarexpansioner behövs om du vill använda en beräkning i din elementuppsättning. Om du till exempel endast vill titta på det sista möjliga året använder du:

```
<Year = {$(=Max(Year))}>
```

Valda värden i andra fält

Modifierare kan basera sig på de valda värdena i ett annat fält. Exempel:

```
<OrderDate = DeliveryDate>
```

Denna modifierare tar de valda värdena från `DeliveryDate` och applicerar dem som ett val på `OrderDate`. Om det finns många distinkta värden, fler än några hundra, blir denna åtgärd processorkrävande och bör därför undvikas.

Elementuppsättningsfunktioner

Elementuppsättningen kan även baseras på set-funktionerna `P()` (möjliga värden) och `E()` (uteslutna värden).

Om du till exempel vill välja länder där produkten `Cap` har sålts kan du använda:

```
<Country = P({1<Product={Cap}>} Country)>
```

På liknande sätt, om du vill välja ut de länder där produkten `Cap` inte har sålts, kan du använda:

```
<Country = E({1<Product={Cap}>} Country)>
```

Set-modifierare med sökningar

Du kan skapa uppsättningar av element genom sökningar med set-modifierare.

Exempel:

- `<Country = {"C*"}>`
- `<Year = {">2015"}>`
- `<Ingredient = {"*garlic*"}>`

Sökningar bör alltid vara inom dubbla citattecken, hakparenteser eller grava accenter. Du kan använda en blandning av stränglitteraler (enkla citattecken) och sökningar (dubbla citattecken). Exempel:

```
<Product = {'Nut', "*Bolt", Washer}>
```

Textsökningar

Jokertecken och andra symboler kan användas i textsökningar:

- En asterisk (*) kommer att motsvara valfritt antal tecken.
- Ett frågetecken (?) kommer att motsvara ett enda tecken.
- En grav accent (^) kommer att markera början på ett ord.

Exempel:

- `<Country = {"C*", "*land"}>`
  Matcha alla länder som börjar med ett `C` eller slutar med `land`.
- `<Country = {"*^z*"}>`
  Det kommer att matcha alla länder som har ett ord som börjar på `z`, som `New Zealand`.

Numeriska sökningar

Du kan göra numeriska sökningar med hjälp av dessa relationsoperatorer: `>`, `>=`, `<`, `<=`

En numerisk sökning börjar alltid med en av dessa operatorer. Exempel:

- `<Year = {">2015"}>`
  Matcha 2016 och efterföljande år.
- `<Date = {">=1/1/2015<1/1/2016"}>`
  Matcha alla datum under 2015. Notera syntaxen för att beskriva ett tidsintervall mellan de två datumen. Datumformatet behöver matcha datumformatet för fältet i fråga.

Sökningar med hjälp av uttryck

Du kan använda sökuttryck till att göra mer avancerade sökningar. En aggregering utvärderas sedan för varje fältvärde i sökfältet. Alla värden för vilka sökuttrycket returnerar sant väljs.

En uttryckssökning börjar alltid med ett likhetstecken: `=`

Exempel:

```
<Customer = {"=Sum(Sales)>1000"}>
```

Det returnerar alla kunder med ett försäljningsvärde som är större än 1 000. `Sum(Sales)` beräknas utifrån det aktuella urvalet. Det betyder att om du har ett urval i ett annat fält, till exempel fältet `Product`, får du endast kunder som uppfyllt försäljningsvillkoret för de valda produkterna som resultat.

Om du vill att villkoret ska vara oberoende av ditt urval behöver du använda set-analysen inne i söksträngen. Exempel:

```
<Customer = {"=Sum({1} Sales)>1000"}>
```

Uttrycken efter likhetstecknet kommer att tolkas som ett booleskt värde. Det betyder att om det utvärderas till något annat kommer alla tal utom noll att tolkas som sanna, samtidigt som noll och strängar tolkas som falska.

Citat

Använd citattecken när söksträngar innehåller mellanslag eller specialtecken. Enkla citattecken implicerar en exakt, skiftlägeskänslig matchning med ett enda fältvärde. Dubbla citattecken implicerar en sökning som inte är skiftlägeskänslig och som potentiellt matchar flera fältvärden.

Exempel:

- `<Country = {'New Zealand'}>`
  Matcha endast `New Zealand`.
- `<Country = {"New Zealand"}>`
  Matcha `New Zealand`, `NEW ZEALAND` och `new zealand`.

Dubbla citattecken kan bytas ut av hakparenteser eller grava accenter.

> *I tidigare versioner av Qlik Sense hanterades enkla och dubbla citattecken på samma sätt, och alla strängar inom citattecken behandlades som sökningar. För att säkerställa bakåtkompatibiliteten kommer appar som skapats med äldre versioner av Qlik Sense att fungera på samma sätt som i tidigare versioner. Appar som skapats med Qlik Sense från november 2017 och framåt kommer att skilja mellan enkla och dubbla citattecken.*

Exempel: diagramuttryck för set-modifierare med sökningar

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
MyTable:
Load
Year(Date) as Year,
Date#(Date,'YYYY-MM-DD') as ISO_Date,
Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,
Country, Product, Amount
Inline
[Date, Country, Product, Amount
2018-02-20, Canada, Washer, 6
2018-07-08, Germany, Anchor bolt, 10
2018-07-14, Germany, Anchor bolt, 3
2018-08-31, France, Nut, 2
2018-09-02, Czech Republic, Bolt, 1
2019-02-11, Czech Republic, Bolt, 3
2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1
2020-07-12, Canada, Anchor bolt, 8
2020-09-16, France, Washer, 1];
```

Exempel 1: diagramuttryck med textsökningar

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

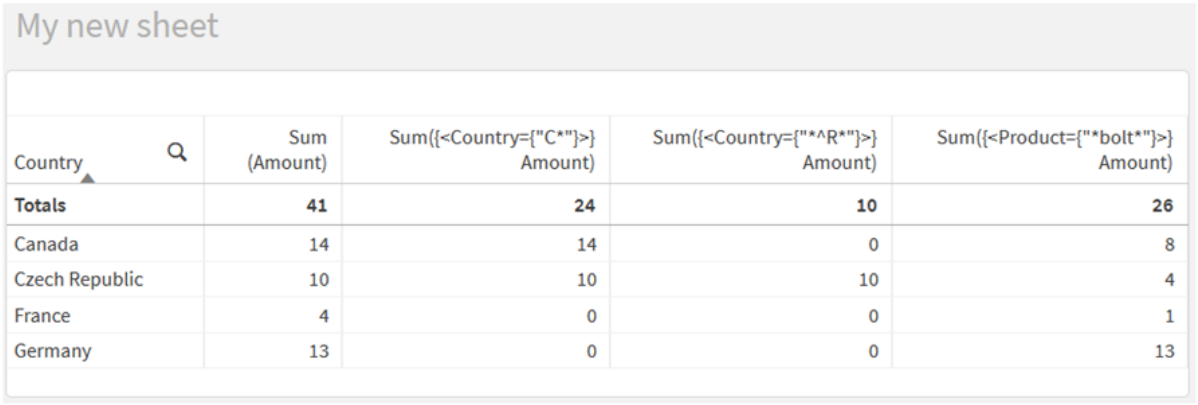
Tabell - set-modifierare med textsökningar

Land	Summa (Belopp)	Summa({<Country={"C*"}>} Belopp)	Summa({<Country={"*^R*"}>} Belopp)	Summa({<Product={"*bolt*"}>} Belopp)
Totalvärden	**41**	**24**	**10**	**26**
Kanada	14	14	0	8
Tjeckien	10	10	10	4
Frankrike	4	0	0	1
Tyskland	13	0	0	13

Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Amount)`
    Summa `Amount` utan set-uttryck.
  - `Sum({<Country={"C*"}>}Amount)`
    Summan `Amount` för alla länder som börjar på `C`, till exempel `Canada` och `Czech Republic`.
  - `Sum({<Country={"*^R*"}>}Amount)`
    Summan `Amount` för alla länder som har ett ord som börjar på `R`, som `Czech Republic`.
  - `Sum({<Product={"*bolt*"}>}Amount)`
    Summan `Amount` för alla produkter som innehåller strängen `bolt`, som `Bolt` och `Anchor bolt`.

Set-modifierare med textsökningar



Exempel: diagramuttryck med numeriska sökningar.

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

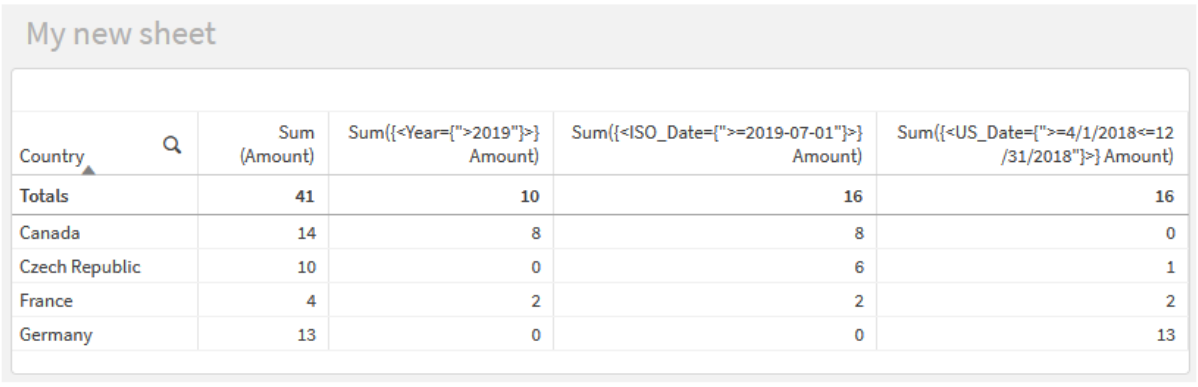
Tabell – set-modifierare med numeriska sökningar

Land	Summa (Belopp)	Summa({<År={">2019"}>} Belopp)	Summa ({<ISO_ Datum={">=2019-07-01"}>} Belopp)	Summa({<US_Datum={">=4/1/2018<=12/31/2018"}>} Belopp)
Totalvärden	**41**	**10**	**16**	**16**
Kanada	14	8	8	0
Tjeckien	10	0	6	1
Frankrike	4	2	2	2
Tyskland	13	0	0	13

Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Amount)`
    Summa `Amount` utan set-uttryck.
  - `Sum({<Year={">2019"}>}Amount)`
    Summan `Amount` för alla år efter `2019`.
  - `Sum({<ISO_Date={">=2019-07-01"}>}Amount)`
    Summan `Amount` för alla datum från och med `2019-07-01`. Formatet på datumet i sökningen måste matcha formatet i fältet.
  - `Sum({<US_Date={">=4/1/2018<=12/31/2018"}>}Amount)`
    Summan `Amount` för alla datum från `4/1/2018` till `12/31/2018`, inklusive start- och slutdatum. Formatet på datumen i sökningen måste matcha formatet i fältet.

Set-modifierare med numeriska sökningar



Exempel 3: diagramuttryck med uttryckssökningar

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

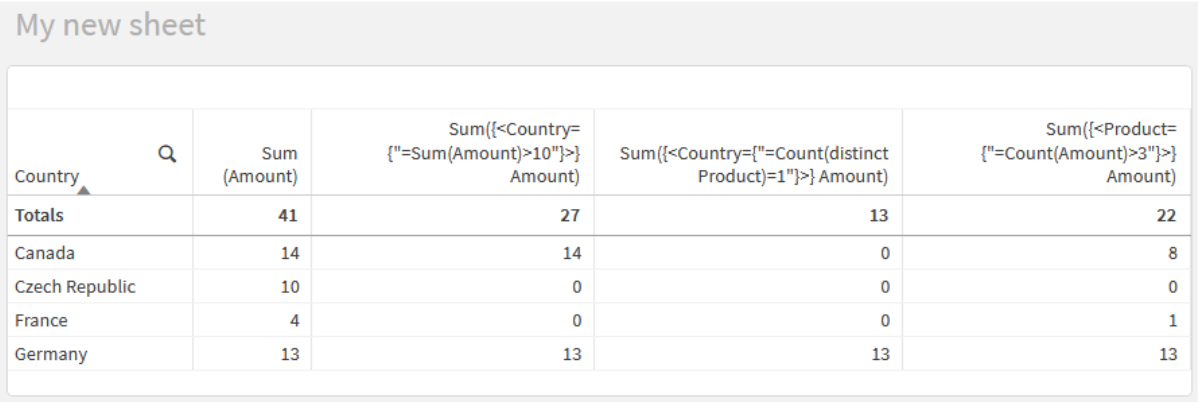
Table - Set modifiers with expression searches

Country	Sum (Amount)	Sum({<Country={"=Sum (Amount)>10"}>} Amount)	Sum({<Country={"=Count(distinct Product)=1"}>} Amount)	Sum({<Product={"=Count (Amount)>3"}>} Amount)
Totals	**41**	**27**	**13**	**22**
Canada	14	14	0	8
Czech Republic	10	0	0	0
France	4	0	0	1
Germany	13	13	13	13

Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Amount)`
    Summa `Amount` utan set-uttryck.
  - `Sum({<Country={"=Sum(Amount)>10"}>}Amount)`
    Summan `Amount` för alla länder som har en aggregerad summa på `Amount` över `10`.
  - `Sum({<Country={"=Count(distinct Product)=1"}>}Amount)`
    Summan `Amount` för alla länder som är kopplade till exakt en distinkt produkt.
  - `Sum({<Product={"=Count(Amount)>3"}>}Amount)`
    Summan `Amount` för alla länder som har över tre transaktioner i data.

Set-modifierare med uttryckssökningar



Exempel	Resultat
sum({$-1<Product = {"*Internal*", "*Domestic*"}>} Sales)	Returnerar försäljningen för det aktuella valet, exklusive transaktioner som hör till produkter med strängen 'Internal' eller 'Domestic' i produktnamnet.
sum({$<Customer = {"=Sum ({1<Year = {2007}>} Sales) > 1000000"}>} Sales)	Returnerar försäljningen för det aktuella valet, men med ett nytt val i fältet "Customer": enbart kunder som under 2007 hade en total försäljning på över 1 000 000.

Set-modifierare med dollarteckenexpansioner

Dollarteckenexpansioner är konstruktioner som beräknas innan uttrycket tolkats och utvärderats. Resultatet matas sedan in i uttrycket istället för `$(…)`. Beräkningen av uttrycket görs sedan med hjälp av resultatet av dollarexpansionen.

Uttrycksredigeraren visar en förhandsgranskning av dollarexpansionen så att du kan bekräfta vad din dollarteckenexpansion utvärderas till.

Förhandsgranskning med dollarteckensexpansion i uttrycksredigeraren

Använd dollarteckensexpansioner när du vill använda en beräkning i din elementuppsättning.

Om du till exempel endast vill titta på det sista möjliga året kan du använda följande konstruktion:

```
<Year = {$(=Max(Year))}>
```

`Max(Year)` beräknas först och resultatet matas in i uttrycket istället för `$(…)`.

Resultatet efter dollarexpansionen kommer att vara ett uttryck som detta:

```
<Year = {2021}>
```

Uttrycket i dollarteckenexpansionen beräknas baserat på det aktuella urvalet. Det betyder att om du har ett urval i ett annat fält kommer resultatet av uttrycket att påverkas.

Om du vill att beräkningen ska vara fristående från urvalet kan du använda set-analysen i dollarexpansionen. Exempel:

```
<Year = {$(=Max({1} Year))}>
```

Strängar

När du vill att dollarexpansionen ska resultera i en sträng gäller normala citatregler. Exempel:

```
<Country = {'$(=FirstSortedValue(Country,Date)'}>
```

Resultatet efter dollarexpansionen kommer att vara ett uttryck som detta:

```
<Country = {'New Zealand'}>
```

Du kommer att få ett syntaxfel om du inte använder citattecken.

Nummer

När du vill att dollarexpansionen ska resultera i en siffra ska du se till att expansionen får samma formatering som fältet. Det betyder att du ibland behöver radbryta uttrycket i en formateringsfunktion.

Exempel:

```
<Amount = {$(=Num(Max(Amount), '###0.00'))}>
```

Resultatet efter dollarexpansionen kommer att vara ett uttryck som detta:

```
<Amount = {12362.00}>
```

Använd en hash för att tvinga expansionen att alltid använda decimalkomma och ingen tusentalsavgränsare. Exempel:

```
<Amount = {$(#=Max(Amount))}>
```

Datum

Om du vill att dollarexpansionen ska resultera i ett datum ska du se till att expansionen har korrekt formatering. Det betyder att du ibland behöver radbryta uttrycket i en formateringsfunktion.

Exempel:

```
<Date = {'$(=Date(Max(Date)))'}>
```

Resultatet efter dollarexpansionen kommer att vara ett uttryck som detta:

```
<Date = {'12/31/2015'}>
```

Precis som för strängar behöver du använda korrekta citattecken.

Ett vanligt exempel kan vara när du vill att beräkningen ska vara begränsad till den senaste månaden (eller året). Då kan du använda en numerisk sökning i kombination med funktionen `AddMonths()`.

Exempel:

```
<Date = {">=$(=AddMonths(Today(),-1))"}>
```

Resultatet efter dollarexpansionen kommer att vara ett uttryck som detta:

```
<Date = {">=9/31/2021"}>
```

Det kommer att välja ut alla händelser som inträffat under den senaste månaden.

Exempel: diagramuttryck för set-modifierare med dollarteckenexpansioner

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
Let vToday = Today();
MyTable:
Load
Year(Date) as Year,
Date#(Date,'YYYY-MM-DD') as ISO_Date,
Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,
Country, Product, Amount
Inline
[Date, Country, Product, Amount
2018-02-20, Canada, Washer, 6
2018-07-08, Germany, Anchor bolt, 10
2018-07-14, Germany, Anchor bolt, 3
2018-08-31, France, Nut, 2
2018-09-02, Czech Republic, Bolt, 1
2019-02-11, Czech Republic, Bolt, 3
2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1
2020-07-12, Canada, Anchor bolt, 8
2021-10-15, France, Washer, 1];
```

Diagramuttryck med dollarteckenexpansioner

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

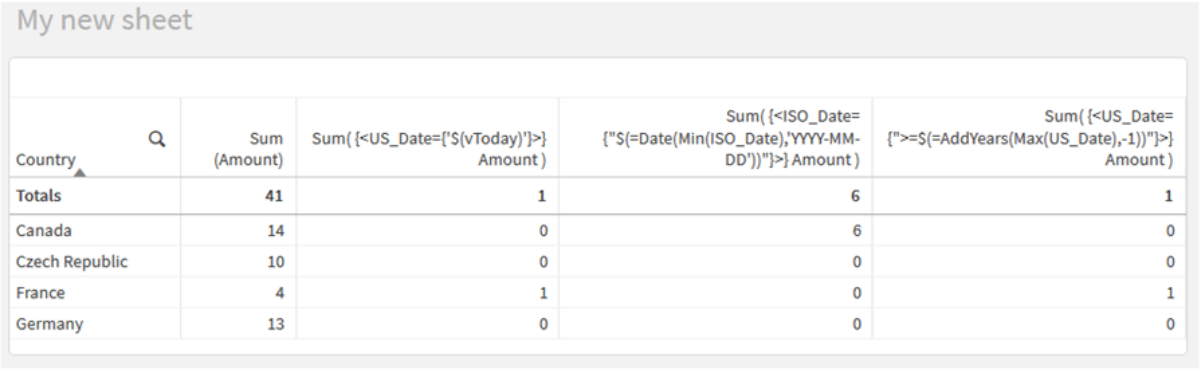
Tabell – set-modifierare med dollarteckenexpansioner

Land	Summa (Belopp)	Summa({<US_Date= {'$(vToday)'}>} Belopp)	Summa({<ISO_Date= {"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>} Belopp)	Summa({<US_Date= {">=$(=AddYears (Max(US_Date),-1))"}>} Belopp)
Totalvärden	**41**	**1**	**6**	**1**
Kanada	14	0	6	0
Tjeckien	10	0	0	0
Frankrike	4	1	0	1
Tyskland	13	0	0	0

Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Amount)`
    Summan `Amount` utan set-uttryck.
  - `Sum({<US_Date={'$(vToday)'}>}Amount)`
    Summan `Amount` för alla poster där `US_Date` är samma som i variabeln `vToday`.
  - `Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>}Amount)`
    Summan `Amount` för alla poster där `ISO_Date` är samma som den första (minsta) möjliga `ISO_Date`. Funktionen `Date()` behövs för att se till att formatet för datumet matchar fältets.
  - `Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>}Amount)`
    Summan `Amount` för alla poster som har ett `US_Date` som är lika med eller kommer efter datumet ett år före det senaste (högsta) möjliga `US_Date`. Funktionen `AddYears()` kommer att returnera ett datum i formatet som anges av variabeln `DateFormat` och det behöver matcha formatet för fältet `US_Date`.

Set-modifierare med dollarteckenexpansioner


My new sheet

Country	Sum (Amount)	Sum({<US_Date={'$(vToday)'}>} Amount)	Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>} Amount)	Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>} Amount)
Totals	**41**	**1**	**6**	**1**
Canada	14	0	6	0
Czech Republic	10	0	0	0
France	4	1	0	1
Germany	13	0	0	0

Exempel	Resultat
sum({$<Year = {$(#vLastYear)}>} Sales)	Returnerar försäljningen för föregående år relaterat till det aktuella urvalet. Här används variabeln vLastYear, som innehåller det relevanta året, för en dollarteckenexpansion.
sum({$<Year = {$(#=Only(Year)-1)}>} Sales)	Returnerar försäljningen för föregående år relaterat till det aktuella urvalet. Här används en dollarteckenexpansion för att beräkna föregående år.

Set-modifierare med set-operatorer

Set-operatorer används till att inkludera, utesluta eller genomskära olika elementuppsättningar. De kombinerar de olika metoderna för att definiera elementuppsättningar.

Operatorer är samma som de som används för set-identifierare.

Operatorer

Operator	Beskrivning
+	Union. Denna binära operation returnerar en uppsättning som består av de poster eller element som tillhör någon av de två set-operanderna.
-	Exklusion. Denna binära operation returnerar en uppsättning som består av de poster eller element som enbart tillhör den första men inte den andra av de två set-operanderna. Den returnerar dessutom det komplementära setet när det används som en unär operator.
*	Intersektion. Denna binära operation returnerar en uppsättning som består av de poster eller element som tillhör båda set-operanderna.
/	Symmetrisk skillnad (XOR). Denna binära operation returnerar en uppsättning som består av de poster eller element som tillhör endera men inte båda set-operanderna.

Till exempel kan de följande två modifierarna definiera samma uppsättning med fältvärden:

- `<Year = {1997, "20*"}>`
- `<Year = {1997} + {"20*"}>`

Båda uttrycken väljer `1997` och åren som börjar med `20`. Med andra ord är det här unionen av de två villkoren.

Set-operatorer möjliggör även mer komplicerade definitioner. Exempel:

```
<Year = {1997, "20*"} - {2000}>
```

Det här uttrycket kommer att välja samma år som de ovan, men dessutom utesluta år `2000`.

.

Exempel: diagramuttryck för set-modifierare med set-operatorer

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
MyTable:
Load
Year(Date) as Year,
Date#(Date,'YYYY-MM-DD') as ISO_Date,
Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,
Country, Product, Amount
Inline
[Date, Country, Product, Amount
2018-02-20, Canada, Washer, 6
2018-07-08, Germany, Anchor bolt, 10
```

```
2018-07-14, Germany, Anchor bolt, 3
2018-08-31, France, Nut, 2
2018-09-02, Czech Republic, Bolt, 1
2019-02-11, Czech Republic, Bolt, 3
2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1
2020-07-12, Canada, Anchor bolt, 8
2020-09-16, France, Washer, 1];
```

Diagramuttryck

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

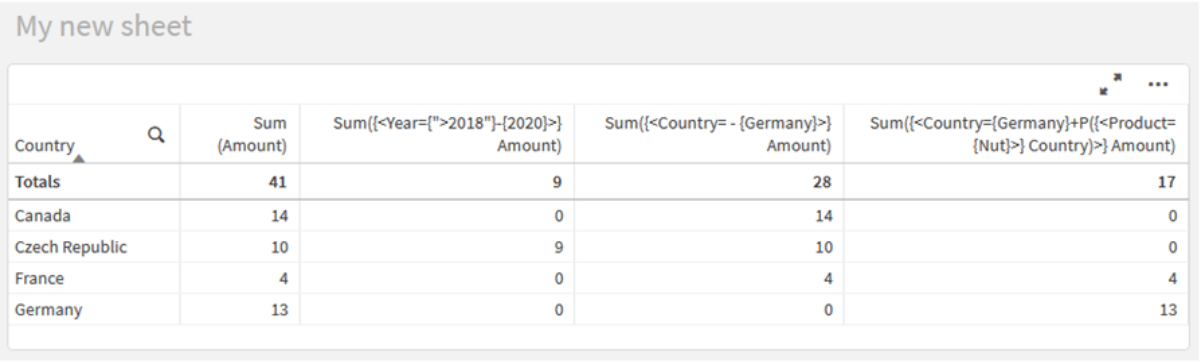
Tabell – set-modifierare med set-operatorer

Land	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum ({<Country=-{Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>}Country)>} Amount)
Totalvärden	**41**	**9**	**28**	**17**
Kanada	14	0	14	0
Tjeckien	10	9	10	0
Frankrike	4	0	4	4
Tyskland	13	0	0	13

Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Amount)`
    Summa `Amount` utan set-uttryck.
  - `Sum({<Year={">2018"}-{2020}>}Amount)`
    Summan `Amount` för alla år efter `2018`, utom `2020`.
  - `Sum({<Country=-{Germany}>}Amount)`
    Summan `Amount` för alla länder utom `Germany`. Observera den unära exklusionsoperatorn.
  - `Sum({<Country={Germany}+P({<Product={Nut}>}Country)>}Amount)`
    Summan `Amount` för `Germany` och alla länder som är kopplade till produkten `Nut`.

Set-modifierare med set-operatorer


My new sheet

Country	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum({<Country= - {Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>} Country)>} Amount)
Totals	**41**	**9**	**28**	**17**
Canada	14	0	14	0
Czech Republic	10	9	10	0
France	4	0	4	4
Germany	13	0	0	13

Exempel	Resultat
sum({$<Product = Product + {OurProduct1} - {OurProduct2} >} Sales)	Returnerar försäljningen för det aktuella valet, men med produkten ”OurProduct1” tillagd i listan över valda produkter och ”OurProduct2” borttagen från densamma.
sum({$<Year = Year + ({“20*”,1997} - {2000}) >} Sales)	Returnerar försäljningen för det aktuella valet, men med ytterligare val i fältet ”Year”: 1997 och alla som börjar på ”20” - dock inte 2000. Observera att om 2000 är inkluderat i det aktuella valet kommer det fortfarande att inkluderas efter modifieringen.
sum({$<Year = (Year + {“20*”,1997}) - {2000} >} Sales)	Returnerar nästan det samma som ovan, men här utesluts 2000, även om det från början ingår i det aktuella urvalet. Detta exempel visar vikten av att ibland använda parenteser för att definiera prioritetsordning.
sum({$<Year = {“*”} - {2000}, Product = {“*bearing*”} >} Sales)	Returnerar försäljningen för det aktuella urvalet men med ett nytt urval i 'Year': alla år utom 2000; och endast för produkter som innehåller strängen 'bearing'.

Set-modifierare med implicita set-operatorer

Standardmetoden för att skriva urval i en set-modifierare är att använda likhetstecken. Exempel:

```
Year = {">2015"}
```

Uttrycket till höger om likhetstecknet i set-modifieraren kallas elementuppsättningen. Det definierar en uppsättning med distinkta fältvärden, med andra ord ett urval.

Denna notation definierar ett nytt urval, utan hänsyn till det aktuella urvalet i fältet. Så om set-identifieraren innehåller ett urval i det här fältet kommer det gamla urvalet att ersättas av det i elementuppsättningen.

När du vill basera ditt urval på det aktuella urvalet i fältet ska du använda ett annat uttryck

Om du till exempel vill fortsätta att använda det gamla urvalet men lägga till kravet att året ska vara efter 2015, kan du skriva följande:

```
Year = Year * {">2015"}
```

Asterisken är en set-operator som definierar en intersektion, så du kommer att få intersektionen mellan det aktuella urvalet i `Year` och det ytterligare kravet att året är efter `2015`. Ett alternativt sätt att skriva det här på är följande:

```
Year *= {">2015"}
```

Det innebär att tilldelningsoperatorn (`*=`) implicit definierar en intersektion.

På ett liknande sätt kan implicita unioner, uteslutningar och symmetriska skillnader definieras med följande: `+=`, `-=`, `/=`

Exempel: diagramuttryck för set-modifierare med implicita set-operatorer

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
MyTable:
Load
Year(Date) as Year,
Date#(Date,'YYYY-MM-DD') as ISO_Date,
Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,
Country, Product, Amount
Inline
[Date, Country, Product, Amount
2018-02-20, Canada, Washer, 6
2018-07-08, Germany, Anchor bolt, 10
2018-07-14, Germany, Anchor bolt, 3
2018-08-31, France, Nut, 2
2018-09-02, Czech Republic, Bolt, 1
2019-02-11, Czech Republic, Bolt, 3
2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1
2020-07-12, Canada, Anchor bolt, 8
2020-09-16, France, Washer, 1];
```

Diagramuttryck med implicita set-operatorer

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

Välj `Canada` och `Czech Republic` från en lista över länder.

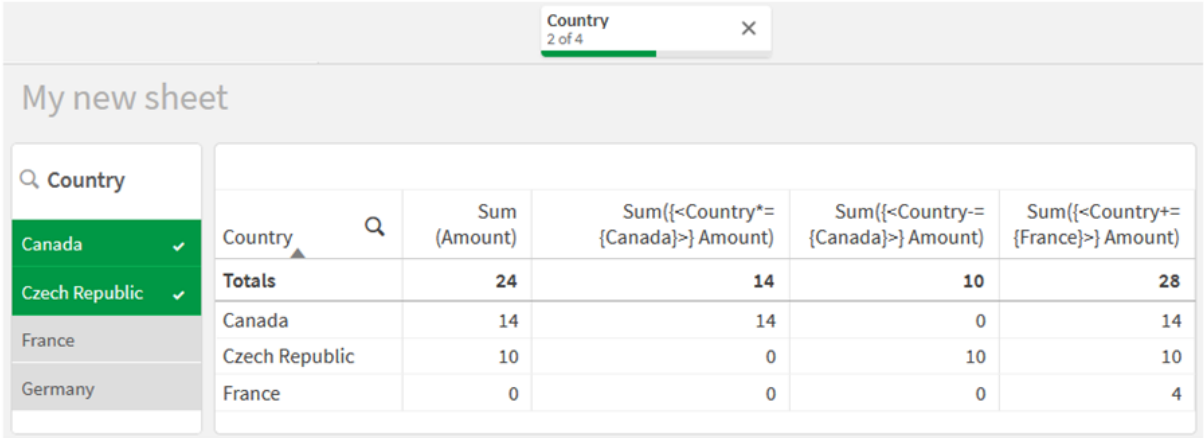
Tabell - Diagramuttryck med implicita set-operatorer

Land	Summa (Belopp)	Summa({<Land*= {Kanada}>} Belopp)	Summa({<Land-= {Kanada}>} Belopp)	Summa({<Land+= {Frankrike}>} Belopp)
Totalvärden	**24**	**14**	**10**	**28**
Kanada	14	14	0	14
Tjeckien	10	0	10	10
Frankrike	0	0	0	4

Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Amount)`
    Summan `Amount` för det aktuella urvalet. Observera att endast `Canada` och `Czech Republic` har värden som inte är noll.
  - `Sum({<Country*={Canada}>}Amount)`
    Summan `Amount` för det aktuella urvalet, som genomskärs av kravet att `Country` ska vara `Canada`. Om `Canada` inte ingår i användarens urval returnerar set-uttrycket en tom uppsättning och kolumnen kommer att ha `0` på alla rader.
  - `Sum({<Country-={Canada}>}Amount)`
    Summan `Amount` för det aktuella urvalet, men uteslut först `Canada` från urvalet `Country`. Om `Canada` inte ingår i användarens urval kommer set-uttrycket inte att ändra några siffror.
  - `Sum({<Country+={France}>}Amount)`
    Summan `Amount` för det aktuella urvalet, men lägg först till `France` i urvalet `Country`. Om `France` redan ingår i användarens urval kommer set-uttrycket inte att ändra några siffror.

Set-modifierare med implicta set-operatorer



Exempel	Resultat
sum({$<Product += {OurProduct1, OurProduct2} >} Sales)	Returnerar försäljningen för det aktuella valet, men med en implicit sammanslagning för att lägga till produkterna "OurProduct1" och "OurProduct2" i listan över de valda produkterna.
sum({$<Year += {“20*”,1997} - {2000} >} Sales)	Returnerar försäljningen för det aktuella urvalet men använder en implicit sammanslagning för att lägga till ett antal år i urvalet: 1997 och alla som börjar på ”20” – dock inte 2000. Observera att om 2000 är inkluderat i det aktuella valet kommer det fortfarande att inkluderas efter modifieringen. Samma som `<Year=Year + ({“20*”,1997}-{2000})>`.
sum({$<Product *= {OurProduct1} >} Sales)	Returnerar försäljningen för det aktuella valet, men bara för intersektionen av det för tillfället valda produkterna samt produkten "OurProduct1".

Set-modifierare som använder set-funktioner

Ibland behöver du definiera en uppsättning fältvärden med hjälp av en nästlad set-definition. Till exempel kanske du vill välja alla kunder som har köpt en viss produkt, utan att välja produkten.

I sådana fall använder du elementuppsättningsfunktionerna `P()` och `E()`. Dessa returnerar elementuppsättningar med möjliga värden respektive uteslutna värden för ett fält. Inom parenteserna kan du ange fältet i fråga och ett set-uttryck som definierar omfattningen. Exempel:

```
P({1<Year = {2021}>} Customer)
```

Det returnerar uppsättningen med kunder som har haft transaktioner under 2021. Du kan använda det här i en set-modifierare. Exempel:

```
Sum({<Customer = P({1<Year = {2021}>} Customer)>} Amount)
```

Det här set-uttrycket kommer att välja dessa kunder men kommer inte att begränsa urvalet till 2021.

Dessa funktioner kan inte användas i andra uttryck.

Dessutom kan endast naturliga uppsättningar användas i elementuppsättningsfunktionerna. Det vill säga, en uppsättning med poster som definieras genom ett enkelt urval.

Till exempel kan den uppsättning som anges med `{1-$}` inte alltid definieras genom urval och det är därför inte en naturlig uppsättning. Användning av dessa funktioner med icke-naturliga uppsättningar returnerar oväntade resultat.

Exempel: diagramuttryck för set-modifierare som använder set-funktioner

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
MyTable:
Load
Year(Date) as Year,
Date#(Date,'YYYY-MM-DD') as ISO_Date,
Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,
Country, Product, Amount
Inline
[Date, Country, Product, Amount
2018-02-20, Canada, Washer, 6
2018-07-08, Germany, Anchor bolt, 10
2018-07-14, Germany, Anchor bolt, 3
2018-08-31, France, Nut, 2
2018-09-02, Czech Republic, Bolt, 1
2019-02-11, Czech Republic, Bolt, 3
2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1
2020-07-12, Canada, Anchor bolt, 8
2020-09-16, France, Washer, 1];
```

Diagramuttryck

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

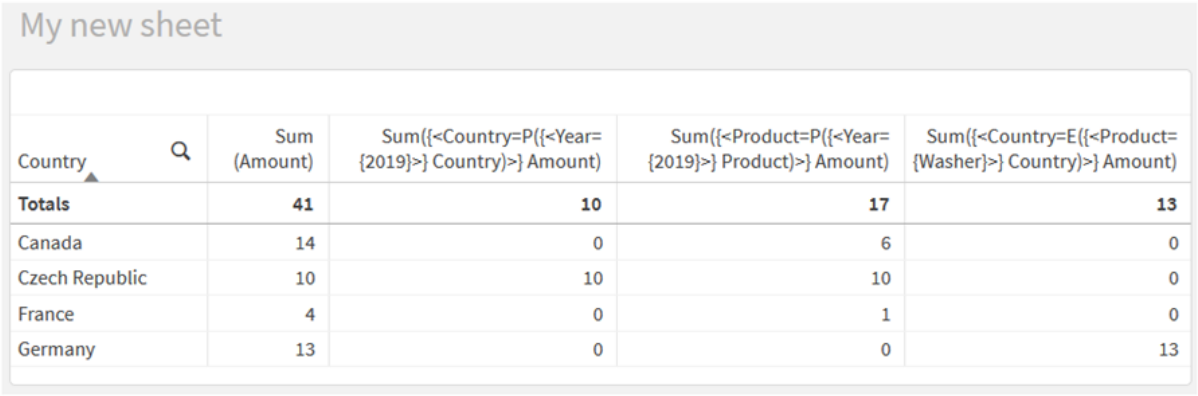
Tabell – Set-modifierare som använder set-funktioner

Land	summa (Belopp)	summa({<Land=P ({<År= {2019}>}Land)>} Belopp)	summa({<Produkt=P ({<År= {2019}>}Produkt)>} Belopp)	summa({<Land=E ({<Produkt= {Washer}>}Land)>} Belopp)
Totalvärden	**41**	**10**	**17**	**13**
Kanada	14	0	6	0
Tjeckien	10	10	10	0
Frankrike	4	0	1	0
Tyskland	13	0	0	13

Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Amount)`
    Summa `Amount` utan set-uttryck.
  - `Sum({<Country=P({<Year={2019}>} Country)>} Amount)`
    Summan `Amount` för länderna som är kopplade till året `2019`. Det kommer dock inte att begränsa beräkningen till `2019`.
  - `Sum({<Product=P({<Year={2019}>} Product)>} Amount)`
    Summan `Amount` för produkterna som är kopplade till året `2019`. Det kommer dock inte att begränsa beräkningen till `2019`.
  - `Sum({<Country=E({<Product={Washer}>} Country)>} Amount)`
    Summan `Amount` för länderna som inte är kopplade till produkten `Washer`.

Set-modifierare som använder set-funktioner



Country	Sum (Amount)	Sum({<Country=P({<Year={2019}>} Country)>} Amount)	Sum({<Product=P({<Year={2019}>} Product)>} Amount)	Sum({<Country=E({<Product={Washer}>} Country)>} Amount)
Totals	**41**	**10**	**17**	**13**
Canada	14	0	6	0
Czech Republic	10	10	10	0
France	4	0	1	0
Germany	13	0	0	13

Exempel	Resultat
sum({$<Customer = P({1<Product= {'Shoe'}>} Customer)>} Sales)	Returnerar försäljningen för det aktuella urvalet, men endast de kunder som någon gång har köpt produkten 'Shoe'. Elementfunktionen P( ) returnerar här en lista över möjliga kunder; de kunder som är inbegripna i urvalet 'Shoe' i fältet Product.
sum({$<Customer = P({1<Product= {'Shoe'}>})>} Sales)	Enligt ovan. Om fältet i elementfunktionen utelämnas returnerar funktionen de möjliga värdena från fältet som angetts i den yttre tilldelningen.

Exempel	Resultat
sum({$<Customer = P({1<Product= {'Shoe'}>} Supplier)>} Sales)	Returnerar försäljningen för det aktuella urvalet, men endast de kunder som någon gång har levererat produkten ”Shoe”, d.v.s. kunden är även leverantör. Elementfunktionen P( ) returnerar här en lista över möjliga leverantörer; de kunder som är inbegripna i urvalet "Shoe" i fältet Product. Listan över leverantörer används sedan som ett urval i fältet Customer.
sum({$<Customer = E({1<Product= {'Shoe'}>})>} Sales)	Returnerar försäljningen för det aktuella urvalet, men endast de kunder som aldrig har köpt produkten 'Shoe'. Elementfunktionen E() returnerar här en lista över uteslutna kunder; de kunder som valts bort i och med urvalet 'Shoe' i fältet Product.

Flagga för att bevara tom mängd

Du kan använda symbolen & (flaggan för att bevara tom mängd) i början av vissa yttre mängduttryck för att åsidosätta implicit radering av urvalsmängd under användaranalys. Flaggan för att bevara tom mängd gör att du kan ändra hur aggregeringar beräknas och representeras när användare gör urval, utan att du behöver skriva om långa, avancerade mängduttryck.

Till exempel, i följande mängduttryck används flaggan för att bevara tom mängd:

```
{&<Country*={'Sweden', 'Denmark'}>} {<ProductCategory={'shirts'}>} Sum( Sales )
```

Bakgrund

Användningsvillkor

Flaggan för att bevara tom mängd kan användas i mängduttryck som har någon av följande komponenter:

- Minst ett inre och ett yttre mängduttryck
- Två eller fler yttre mängduttryck

Använd flaggan för att bevara tom mängd när du vill undvika *Implicit radering av urvalsmängd (page 342)*.

Implicit radering av urvalsmängd

Implicit rensning av urvalsuppsättningar är ett internt beteende som ibland kan vara oönskat för slutanvändaranalys när komplexa kedjor av uppsättningsuttryck används.Implicit rensning av urvalsuppsättningar kan inträffa för uppsättningsuttryck som är strukturerade enligt *Användningsvillkor (page 342)*.

Det oönskade beteendet är som följer:

1. Ett eller flera yttre uppsättningsuttryck i kedjan – specifikt det första (eller enda) och alla efterföljande yttre uppsättningsuttryck utom det sista – definierar datauppsättningar. Exempel: `Country*={'Sweden', 'Denmark'}>}`

2. Något händer som producerar en tom uppsättning för en eller flera av dessa yttre uppsättningsuttryckskomponenter. Oftast görs ett motstridigt urval i appen.
3. För de tomma uppsättningarna raderas de uppsättningar som definieras av dessa uttryck när uttrycket utvärderas. Detta innebär att data som används i aggregeringarna inte längre reduceras till de definierade uppsättningarna, utan istället används all data.
   Å andra sidan respekteras de uppsättningar som definieras i alla yttre uppsättningsuttryck som producerar icke-tomma uppsättningar, samt det sista yttre uppsättningsuttrycket i kedjan, under utvärderingen.

Detta beteende resulterar i aggregeringsvärden som kan vara oväntade och kan leda till inkonsekvent analys. För att kringgå implicit radering av urvalsuppsättningar kan du använda flaggan för att bevara tomma uppsättningar. Se *Lösningen: Använda flaggan för att bevara tomma uppsättningar (page 343)*.

Lösningen: Använda flaggan för att bevara tomma uppsättningar

Använd &-symbolen (flaggan för att bevara tomma uppsättningar) för att åsidosätta implicit radering av urvalsuppsättningar. Denna flagga säkerställer att uppsättningar som definieras i alla yttre uppsättningsuttryck, utom det sista, i en kedja inte raderas, utan istället respekteras under hela utvärderingen av hela uttrycket.

Flaggan för att bevara tomma uppsättningar ger:

- Flexibilitet och tidsbesparing när du har avancerade, långa uppsättningsuttryck som du vill behålla intakta, snarare än att skriva om.
- En möjlighet att ändra aggregeringslogik för att passa avancerade användningsfall.

Syntax

I ett yttre uppsättningsuttryck placeras flaggan för att bevara tomma uppsättningar som ett enda &-tecken i början av uttrycket – det vill säga före uppsättningsmodifierarna, identifierarna och operatorerna.

Exempel:

Exempel – flagga för att bevara tomma uppsättningar

Uttryck	Uttryck utan flagga för att bevara tomma uppsättningar
`{&<Country*={'Sweden', 'Denmark'}>} {<ProductCategory={'shirts'}>} Sum( Sales )`	`{<Country={'Sweden'}>} {<ProductCategory={'shirts'}>} Sum (Sales)`
`{& <Year=, Product={XI345}>} {<Region={Europe}>} Sum (Sales)`	`{<Year=, Product={XI345}>} {<Region={Europe}>} Sum (Sales)`

Detaljer och exempel

För mer djupgående information, gå till *Kedjor av flera yttre och inre uppsättningsuttryck (page 346)*.

Inre och yttre uppsättningsuttryck

Uppsättningsuttryck används i aggregeringsfunktioner och inom en klammerparentes.

När du använder ett uppsättningsuttryck i en aggregeringsfunktion så kan det se ut så här:

Exempel: Inre uppsättningsuttryck

```
Sum( {$<Year={2021}>} Sales )
```

Använd ett uppsättningsuttryck utanför aggregeringsfunktionen om du har uttryck med flera aggregeringar och vill undvika att skriva samma uppsättningsuttryck i varje aggregeringsfunktion.

Om du använder ett yttre uppsättningsuttryck så måste det placeras i början av omfattningen.

Exempel: Yttre uppsättningsuttryck

```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

Om du använder ett uppsättningsuttryck utanför aggregeringsfunktionen så kan du även tillämpa den på befintliga originalmått.

Exempel: Yttre uppsättningsuttryck som tillämpas på originalmåttet

```
{<Year={2021}>} [Master Measure]
```

Ett uppsättningsuttryck som används utanför aggregeringsfunktionerna påverkar hela uttrycket, med undantag för om det är inom parentes eftersom parenteserna då definierar omfattningen. I exemplet på val av lexikal omfattning nedan så tillämpas ett uppsättningsuttryck endast på aggregeringen innanför parenteserna.

Exempel: Val av lexikal omfattning

```
( {<Year={2021}>} Sum(Amount) / Count(distinct Customer) ) - Avg(CustomerSales)
```

Regler

Lexikal omfattning

Uppsättningsuttrycket påverkar hela uttrycket, med undantag för om det är inom parentes. Om detta är fallet så definierar parenteserna den lexikala omfattningen.

Läge

Uppsättningsuttrycket måste placeras i början av den lexikala omfattningen.

Kontext

Kontext är det urval som är relevant för uttrycket. Traditionellt sett har kontexten alltid varit standardtillståndet för det aktuella urvalet. Om ett objekt är inställt på ett parallellt tillstånd så är kontexten det parallella tillståndet för det aktuella urvalet.

Du kan också definiera en kontext genom ett yttre uppsättningsuttryck.

Arv

Inre uppsättningsuttryck ges företräde framför yttre uppsättningsuttryck. Om det inre set-uttrycket innehåller en set-identifierare så ersätter det kontexten som har definierats i det yttre set-uttrycket. I annat fall utvärderas både det inre och det yttre set-uttrycket.

Example 1: Inre set-uttryck med set-identifierare

```
{<Year={2023}>} Sum(Sales) / Count({1} distinct OrderNumber)
```

I ovanstående uttryck består det inre set-uttrycket `Count({1} distinct OrderNumber)` av set-identifieraren `{1}`. När detta inre set-uttryck utvärderas tillämpas inte den omfattning som definieras i det yttre set-uttrycket `{<Year={2023}>}`.

Example 2: Inre set-uttryck utan set-identifierare

```
{<Year={2023}>} Sum ({<Status={'Confirmed'}>} Sales_Stream1) + Sum ({<UpdatedStatus=
{'Confirmed'}>} Sales_Stream2)
```

I ovanstående uttryck innehåller de inre set-uttrycken `Sum ({<Status={'Confirmed'}>} Sales_Stream1)` och `Sum ({<UpdatedStatus={'Confirmed'}>} Sales_Stream2)` inte set-identifierare. Därför tillämpas det yttre set-uttrycket `{<Year={2023}>}` och båda de inre set-uttrycken när resultatet utvärderas.

Tilldelning av elementuppsättningar

Tilldelningen av elemenuppsättningar fastställer hur de två urvalen slås ihop. Om ett vanligt likhetstecken används så har det inre uppsättningsuttrycket högre prioritet. I övrigt används den impicita operatorn för uppsättningen.

- `{<Field={value}>}` – detta inre urval ersätter eventuella yttre urval i "`Field`".
- `{<Field+={value}>}` – detta inre urval slås ihop med det yttre urvalet i "`Field`" med hjälp av unionoperatorn.
- `{<Field*={value}>}` – detta inre urval slås ihop med det yttre urvalet i "`Field`" med hjälp av intersektionsoperatorn.

Arv i flera steg

Arv kan ske i flera steg. Exempel:

- Aktuellt urval → `Sum(Amount)`
  Aggregeringsfunktionen använder kontext, vilket här är aktuellt urval.
- Aktuellt urval → `{<Set1>} Sum(Amount)`
  `Set1` ärver från aktuellt urval och resultatet blir kontexten för aggregeringsfunktionen.
- Aktuellt urval → `{<Set1>} ({<Set2>} Sum(Amount))`
  `Set2` ärver från `Set1`, som i sin tur ärver från aktuellt urval och resultatet blir kontexten för aggregeringsfunktionen.

Arv för kedjor av inre och yttre uppsättningsuttryck

Det finns ytterligare överväganden för arv när man använder flera uppsättningsuttryck i komplexa kedjor, särskilt när ett yttre uppsättningsuttryck följs av andra yttre uppsättningsuttryck och/eller ett inre uppsättningsuttryck. För mer information, se *Kedjor av flera yttre och inre uppsättningsuttryck (page 346)*.

Funktionen Aggr()

`Aggr()`-funktionen skapar en nästlad aggregering som har två oberoende aggregeringar. I exempelet nedan beräknas en `Count()` för varje värde för `Dim` och den resulterande matrisen aggregeras med hjälp av `Sum()`-funktionen.

Exempel:

```
Sum(Aggr(Count(X),Dim))
```

`Count()` är den inre aggregeringen och `Sum()` är den yttre aggregeringen.

- Den inre aggregeringen ärver inte någon kontext från den yttre aggregeringen.
- Den inre aggregeringen ärver kontext från `Aggr()`-funktionen som kan innehålla ett uppsättningsuttryck.
- Både `Aggr()`-funktionen och den yttre aggregeringsfunktionen ärver kontext från ett yttre uppsättningsuttryck.

Kedjor av flera yttre och inre uppsättningsuttryck

Observera att det är möjligt att ha fler än ett yttre uppsättningsuttryck. I det här exemplet har vi två yttre uppsättningsuttryck och ett inre:

```
{<Year={2021}>} {<Region={"Europe"}>} Sum( {$<Product={"XI345"}>} Sales )
```

Uppsättningsuttrycken kommer att tillämpas från vänster till höger. Detta innebär att om två uppsättningsuttryck använder motstridiga set, kommer det uttryck som står längst till höger att ha företräde.

Även om det inte finns någon gräns för antalet yttre uppsättningsuttryck som kedjas ihop på detta sätt, är antalet inre uppsättningsuttryck begränsat till ett. Om fler än ett inre uppsättningsuttryck står bredvid varandra uppstår inget fel, men endast det längst till höger kommer att användas vid utvärderingen.

Arv från uppsättningsuttryck

När ett aggregeringsuttryck innehåller fler än ett uppsättningsuttryck som en del av en kedja, utvärderas uppsättningsuttrycken ett efter ett från vänster till höger. Resultatet av varje uppsättningsuttryck är en grupp av set, potentiellt ett för varje dimension. Seten skickas vidare till nästa uppsättningsuttryck, som kan göra ytterligare ändringar innan de skickas vidare till nästa.

Så länge varje uppsättningsuttryck arbetar med olika dimensioner kommer det inte att finnas några konflikter mellan dem, och alla set kommer att bevaras till slutet av kedjan (och tillämpas under aggregeringen). Det finns ett undantag från denna regel, som ibland gäller när en dimension i ett uppsättningsuttryck har gett ett tomt set. Se följande för mer information:

- *Tomma urvalsuppsättningar (page 346)*
- *Implicit radering av urvalsuppsättningar (page 347)*

Tomma urvalsuppsättningar

När en dimension i ett uppsättningsuttryck utvärderas är det möjligt att setet inte innehåller några data. Detta kallas för ett tomt set. Tomma set kan förekomma i många scenarier, t.ex:

- Felaktig definition av uttryck. Till exempel, när man försöker leta efter ett värde 2025 med uppsättningsuttrycket `{<OrderYear={'2025'}>}`, skrivs uttrycket som `{<OrderYear={'0025'}>}`.
- Under analysen görs ett urval i en annan dimension än den som refereras till i uttrycket och detta urval resulterar i att det inte finns några tillgängliga data för den angivna uttrycksdimensionen. En användare gör t.ex. ett urval för `Region='Europe'` och en uppsättning filter för uppsättningsuttryck för `{<ProductCategory={'Shirts'}>} Sum (Sales)`. Om inga skjortor säljs i Europa innehåller uppsättningsuttrycket ett tomt set.

- Det tomma setet definieras uttryckligen i uttrycket. Detta är inte vanligt eller användbart, men det är tekniskt möjligt. Exempel: `{<Year={}>}`

Implicit radering av urvalsuppsättningar

Om resultatet efter tillämpning av ett uppsättningsuttryck är en tom uppsättning för en av dimensionerna (se *Tomma urvalsuppsättningar (page 346)*) och det finns ytterligare ett uppsättningsuttryck som tillämpas efter detta, kommer urvalet för den dimensionen att raderas precis innan nästa uppsättningsuttryck tillämpas. En raderad urvalsuppsättning innebär att alla värden inkluderas (som om inget urval hade gjorts). Observera dock att denna radering endast sker om det finns ytterligare ett eller flera yttre uppsättningsuttryck som tillämpas efter uppsättningsuttrycket där den tomma uppsättningen produceras.

Här är ett exempel på när en raderad urvalsuppsättning kommer att ske:

```
{<Year={}>} {<Region={"Europe"}>} Sum( Sales )
```

`Year={}` kommer att skapa den tomma uppsättningen för dimensionen `Year`. Detta innebär att urvalsuppsättningen för `Year` raderas när uppsättningen överförs till nästa uppsättning uttryck, som innehåller `Region={Europe}`. Observera att raderingen av urvalet endast sker för en enda dimension. Uppsättningar i andra dimensioner – inklusive uppsättningar inom samma uppsättningsuttryck – kommer fortfarande att vidarebefordras utan ändring.

Här är ett annat exempel på en raderad urvalsuppsättning:

```
{<Year={}, Product={"XI345"}>} {<Region={"Europe"}>} Sum(Sales)
```

I ovanstående exempel händer följande:

- Urvalsuppsättningarna `Product={"XI345"}` och `Region={"Europe"}` tillämpas.
- `Year={}` tillämpas inte. Detta beror på att den uppsättningen raderas när det sista uppsättningsuttrycket – `{<Region={"Europe"}>}` – används.

Om du flyttar en komponent i ett uppsättningsuttryck till det sista yttre uppsättningsuttrycket förhindras implicit radering. Ta det här exemplet:

```
{<Product={"XI345"}>} {<Year={}, Region={"Europe"}>} Sum(Sales)
```

Här har vi flyttat `Year`-uttrycker till det sista uppsättningsuttrycket i kedjan. Detta förhindrar att den implicita raderingen sker, och den tomma uppsättningen för dimensionen `Year` kommer faktiskt att användas för `Sum`-aggregeringen.

I vilket fall som helst kan oönskat beteende inträffa när urvalsuppsättningar raderas. För att förbättra analysupplevelsen och göra det enklare att skriva kedjor av uppsättningsuttryck kan du använda flaggan bevara tom uppsättning. Mer information finns i *Flaggan bevara tom uppsättning (yttre uppsättningsuttryck) (page 347)*.

Flaggan bevara tom uppsättning (yttre uppsättningsuttryck)

Använd symbolen `&` (flaggan bevara tom uppsättning) för att förenkla utvärderingsmodellen för yttre uppsättningsuttryck. Exempel:

```
{& <Year={}, Product={XI345}>} {<Region={Europe}>} Sum( Sales )
```

I det här exemplet definierar `Year={}` en tom urvalsuppsättning för `Year`. Utan symbolen & skulle uttrycket `{<Region={Europe}>}` leda till att den tomma urvalsuppsättningen `Year` raderas, vilket skulle leda till att alla data används i aggregeringsberäkningen. Symbolen & tvingar aggregeringen att respektera den tomma urvalsmängden `Year`, vilket resulterar i mer begripliga utdata.

För fullständig information om flaggan för bevarande av tomma mängder, se *Flagga för att bevara tom mängd (page 342)*.

Yttre set-uttryck med dollarteckenexpansioner

När en (eller flera) dollarteckenexpansioner används i ett större uttryck kommer varje dollarteckenexpansion att utvärderas separat innan resultatet infogas i resten av uttrycket. En konsekvens av den här utvärderingsmetoden är att set-uttryck utanför ett uttryck med dollarteckenexpansion inte påverkar resultatet. Detta skapar en effekt där det sker en dubbel utvärdering av information inom ett uttryck.

Exempel – hur uttryck som innehåller yttre set-uttryck och dollarteckenexpansioner utvärderas

Anta att du har detta exempel på uttryck:

```
{<Year={2024}>} $(=Sum(Sales))
```

Inom uttrycket ovan finns det ett set-uttryck för att välja *Year* värden för *2024*: `{<Year={2024}>}`. När hela exempeluttrycket är utvärderat tillämpas inte detta set-uttryck.

Istället utvärderas uttrycket med dollartecken `$(=Sum(Sales))` först.

I det här exemplet antar vi att `Sum(Sales)` utvärderas till ett resultat på 1500311. Vid utvärdering av uttrycket `{<Year={2024}>} $(=Sum(Sales))` infogas detta tal i det omgivande uttrycket. Denna utvärderingsordning resulterar i följande delvis utvärderade konfiguration (observera att detta inte är ett användbart Qlik Sense-uttryck, och visas endast för att demonstrera utvärderingsordningen):

```
{<Year={2024}>} 1500311
```

Därefter utvärderas uttrycket `{<Year={2024}>} 1500311` i sin helhet. Det uppställda uttrycket `{<Year={2024}>}` har ingen effekt eftersom uttrycket till höger om det bara är ett tal, inte någon typ av aggregerat uttryck.

Om man använder värdet `Sum(Sales)` som anges ovan i exemplet blir därför det utvärderade resultatet för hela uttrycket `{<Year={2024}>} $(=Sum(Sales))` 1500311.

Skapa ett uttryck från delar

Användning av dollarteckenexpansioner i ett uttryck resulterar i en dubbel utvärderingseffekt. Du kan dra nytta av denna dubbla utvärderingseffekt genom att skapa ett uttryck med hjälp av delar.

Exempel

Anta att du har detta exempel på uttryck:

```
{<Year={2024}>} $(='Sum(' & 'Sales' & ')')
```

I den första utvärderingsrundan utvärderas dollarteckenexpansionen `$(='Sum(' & 'Sales' & ')')`. I detta steg kombineras de tre strängarna till en enda sträng `'Sum(Sales)'`. Eftersom det här är en sträng och inte ett uttryck utvärderas inte uttrycket inuti strängen vid detta tillfälle.

I detta skede av utvärderingen ska uttrycket tolkas av Qlik Sense på följande sätt:

```
{<Year={2024}>} Sum(Sales)
```

I det här fallet kommer det yttre set-analysuttrycket att tillämpas på `Sum`-aggregeringen eftersom vi nu har producerat ett normalt uttryck och inte bara ett tal (som var fallet i det första exemplet). Observera att citattecknen använde den första utvärderingsrundan så att uttrycket bevarades (och konkatenerades med `&`) för den andra utvärderingsrundan.

Variabelsökning

När en variabel dollarteckenexpansion används för att göra en variabelsökning infogas innehållet i variabeln i det omgivande uttrycket innan innehållet utvärderas.

Exempel

Anta att du har detta exempel på uttryck:

```
{<Year={2024}>} $(Var1)
```

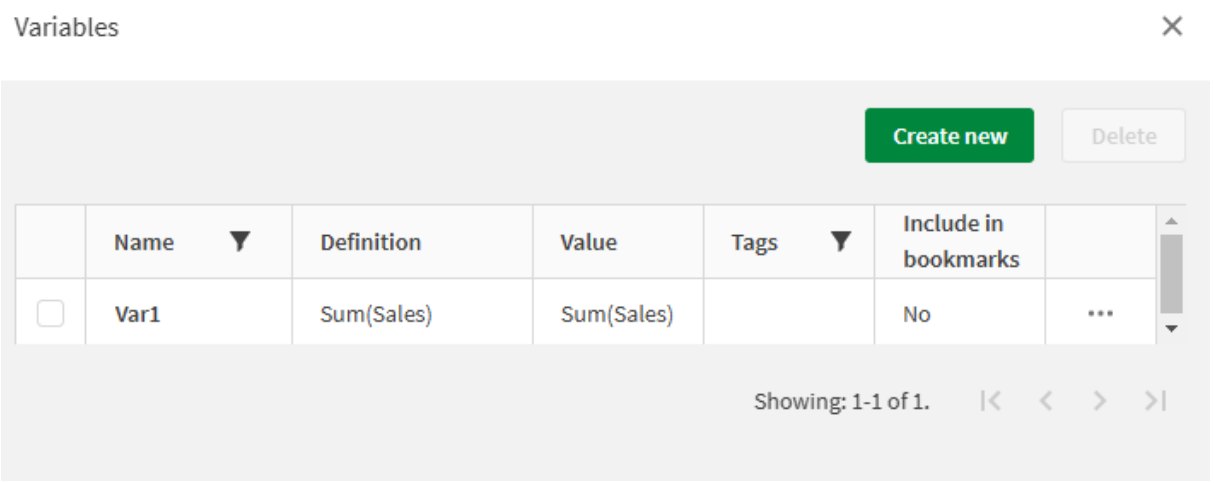
I det här exemplet kommer innehållet i variabeln *Var1* att infogas i det omgivande uttrycket innan innehållet utvärderas.

Utfall A – definition av variabel utan likhetstecken

Om *Var1* definieras som *Sum(Sales)*, skulle det se ut så här efter expansion (observera att konfigurationen nedan är ett delvis utvärderat uttryck):

```
{<Year={2024}>} Sum(Sales)
```

Dialogrutan Variabler visar definition och värde för variabeln Var1, när ett likhetstecken inte placeras i början av variabeldefinitionen.

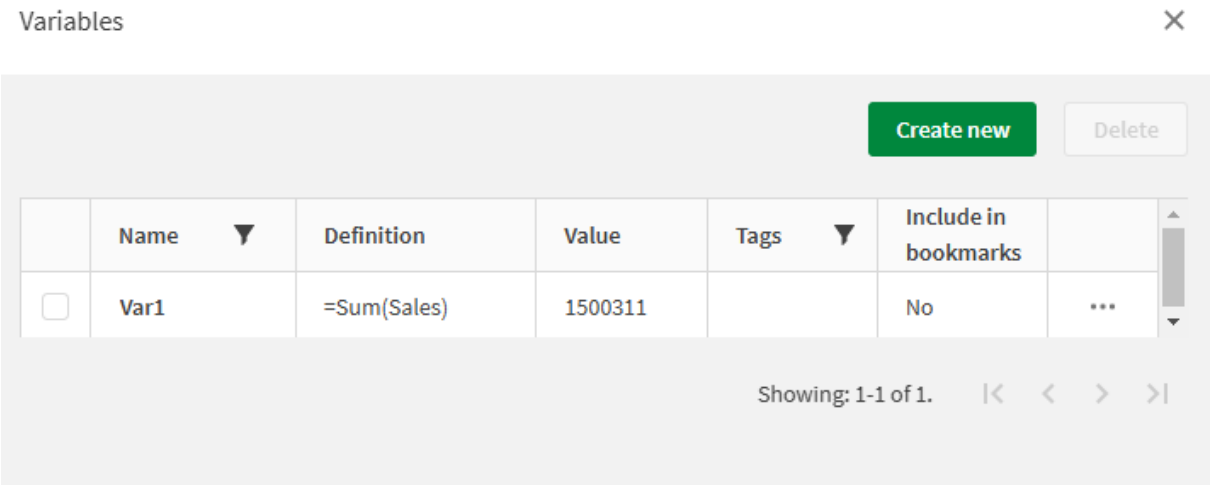


I det här exemplet kommer alltså det yttre set-uttrycket `{<Year={2024}>}` att användas.

Utfall b – definition av variabel med likhetstecken

Om definitionen i *Var1* däremot börjar med ett likhetstecken – det vill säga *=Sum(Sales)* – tillämpas inte det yttre set-uttrycket. Detta beror på att likhetstecknet gör att uttrycket för variabeln beräknas i förväg och isolerat. Detta leder till ett resultat som liknar det första exemplet, där det finns ett yttre set-uttryck som ska utvärderas tillsammans med ett tal, vilket inte ger något användbart resultat.

Dialogrutan Variabler visar definition och värde för variabeln Var1, när ett likhetstecken placeras i början av variabeldefinitionen.



Introduktion – Skapa ett set-uttryck

Du kan bygga set-uttryck i Qlik Sense som stöd för dataanalys. I det här sammanhanget kallas analysen ofta för set-analys. Set-analys gör det möjligt att definiera en omfattning som är olik den uppsättning med poster som definieras av det aktuella urvalet i en app.

Det här får du lära dig

Denna introduktion ger data och diagramuttryck för att bygga set-uttryck med hjälp av set-modifierare, identifierare och operatorer.

Vem är den här introduktionskursen avsedd för?

Den här introduktionskursen är avsedd för apputvecklare som är vana att arbeta med Skriptredigeraren och diagramuttryck.

Vad du behöver göra innan du börjar

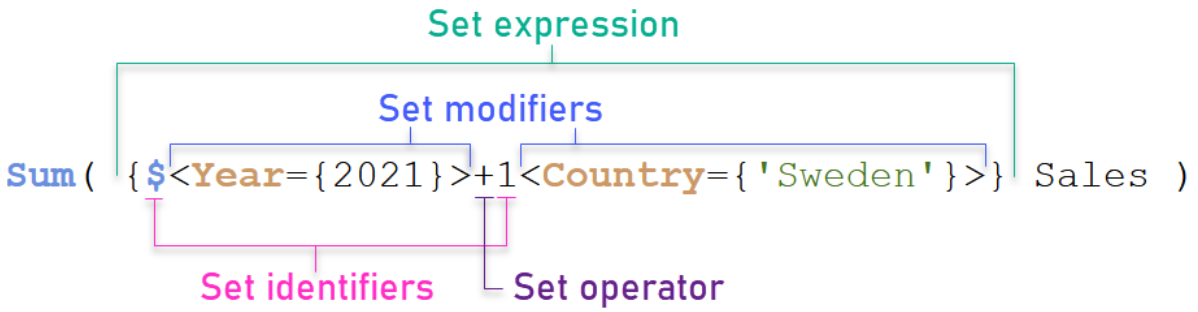
En allokerad Qlik Sense Enterprise Professional-åtkomst, som tillåter att du kan ladda data och skapa appar.

- Set-analys del 1: nybörjarintroduktion
- Set-analys del 2

Element i ett set-uttryck

Set-uttryck omsluts av en aggregeringsfunktion, som `Sum()`, `Max()`, `Min()`, `Avg()` eller `Count()`. Set-uttryck består av byggstenar som kallas för element. Dessa element är set-modifierare, -identifierare och -operatorer.

Element i ett set-uttryck



Set-uttrycket ovan är till exempel byggt från aggregeringen `Sum(Sales)`. Set-uttrycket är omslutet av de yttre klammerparenteserna: `{ }`

Den första operanden i uttrycket är: `$<Year={2021}>`

Denna operand returnerar försäljningen under året `2021` för det aktuella urvalet. Modifieraren, `<Year={2021}>`, innehåller urvalet av år, d.v.s. 2021. Set-identifieraren `$` indikerar att set-uttrycket är baserat på det aktuella urvalet.

Den andra operanden i uttrycket är: `1<Country={'Sweden'}>`

Operanden returnerar Sales för Sweden. Modifieraren, `<Country={'Sweden'}>`, innehåller urvalet av land, d.v.s. Sweden. Set-identifieraren `1` indikerar att urval som görs i appen kommer att ignoreras.

Slutligen indikerar set-operatorn `+` att uttrycket returnerar en uppsättning som består av posterna som tillhör någon av de två set-operanderna.

Introduktionskursen Skapa ett set-uttryck

Slutför följande procedurer för att skapa set-uttrycken som visas i den här introduktionskursen.

Skapa en ny app och ladda data

Gör följande:

1. Skapa en ny -app.
2. Klicka på **Skriptredigeraren**. Alternativt kan du öppna appnavigeringsmenyn och välja **Skriptredigeraren**.
3. Skapa ett nytt delavsnitt i **Skriptredigeraren**.
4. Kopiera följande data och klistra in dem i det nya delavsnittet: *Introduktionsdata för set-uttryck (page 359)*
5. Klicka på **Ladda data**. Data laddas som en inline load.

Skapa set-uttryck med modifierare

Set-modifieraren består av ett eller flera fält namn, som vart och ett följs av ett urval som ska göras på just det fältet. Modifieraren omsluts av vinkelparenteser. Ett exempel är detta set-uttryck:

```
Sum ( {<Year = {2015}>} Sales )
```

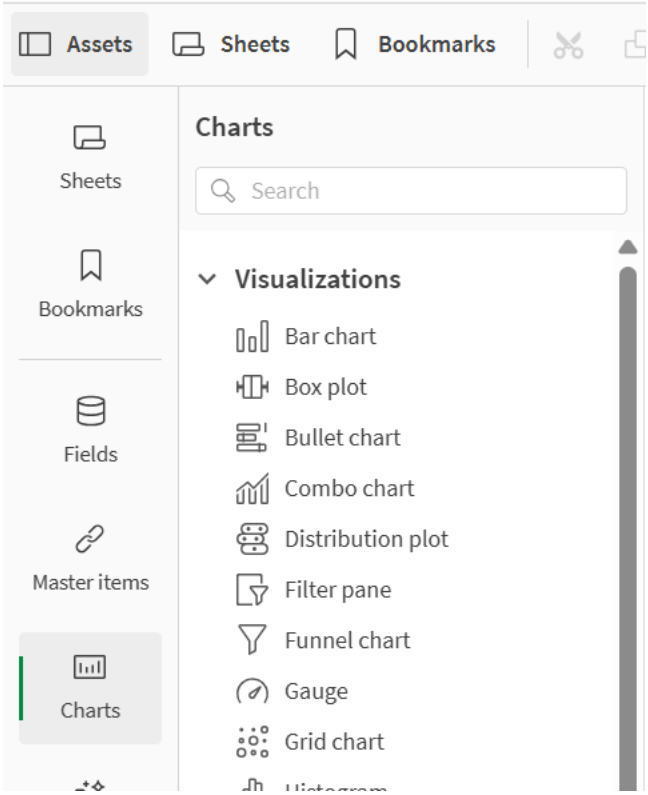
Modifieraren är:

```
<Year = {2015}>
```

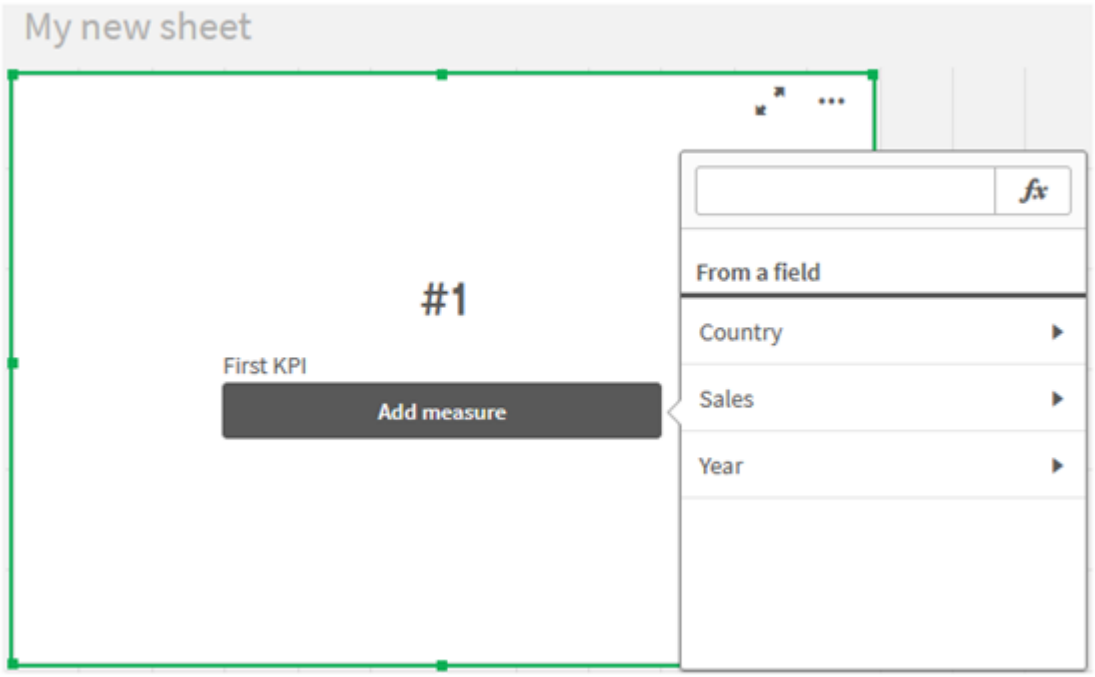
Den här modifieraren anger att data från år 2015 kommer att väljas. Klammerparenteserna som omsluter modifieraren indikerar att det är ett set-uttryck.

Gör följande:

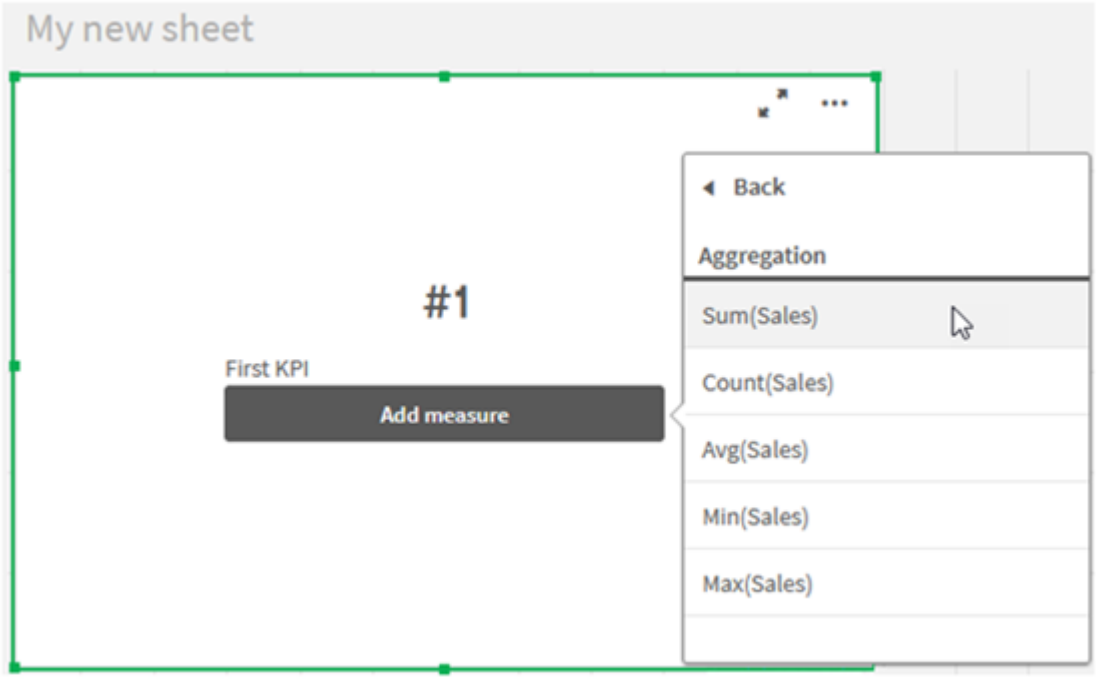
1. I ett ark öppnar du panelen **Resurser** från navigeringsfältet och klickar sedan på **Diagram**.



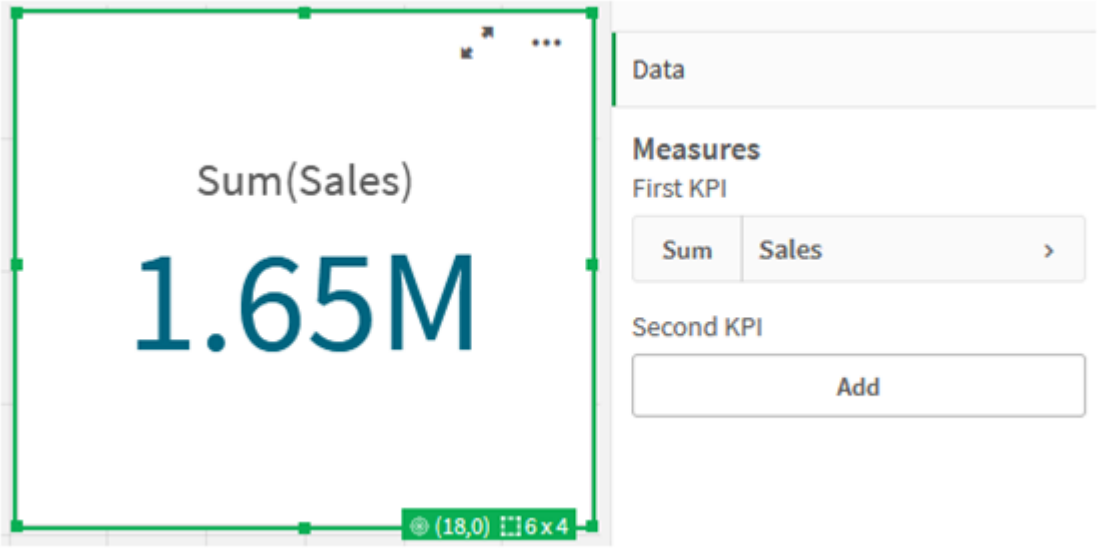
2. Under **Visualiseringar**, dra ett **KPI** till arket och klicka sedan på **Lägg till mått**.



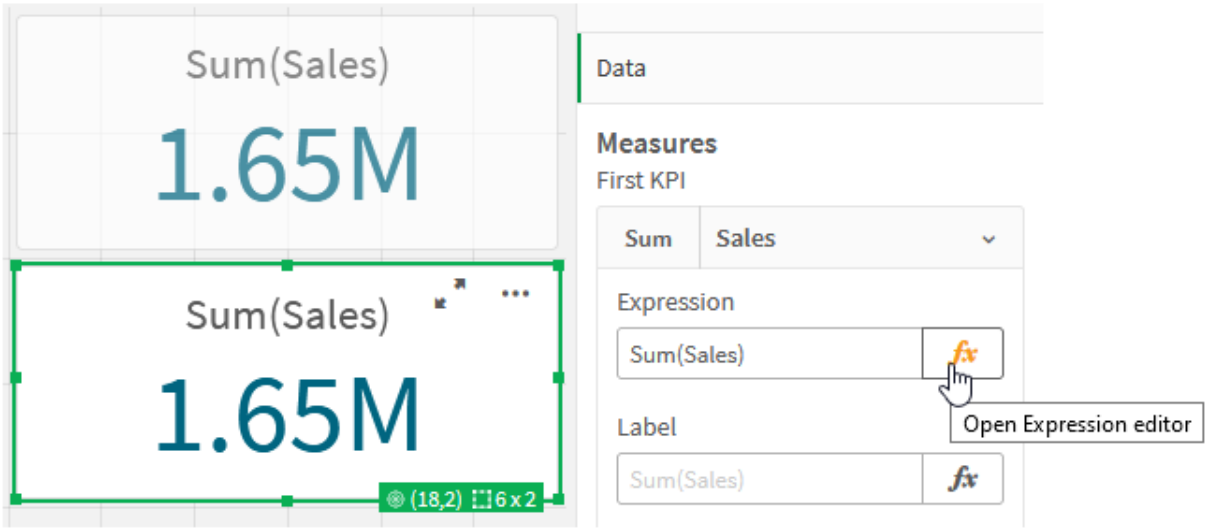
3. Klicka på `Sales` och välj sedan `Sum(Sales)` för aggregeringen.



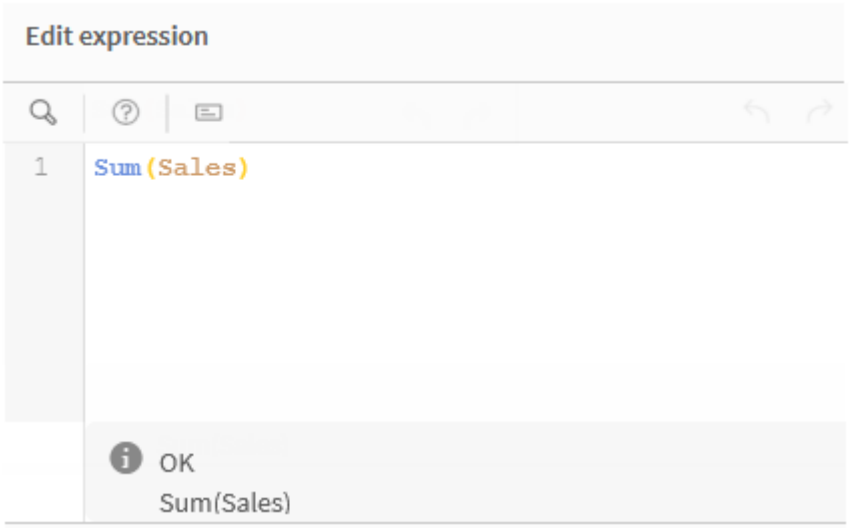
KPI visar summan av försäljningen för alla år.



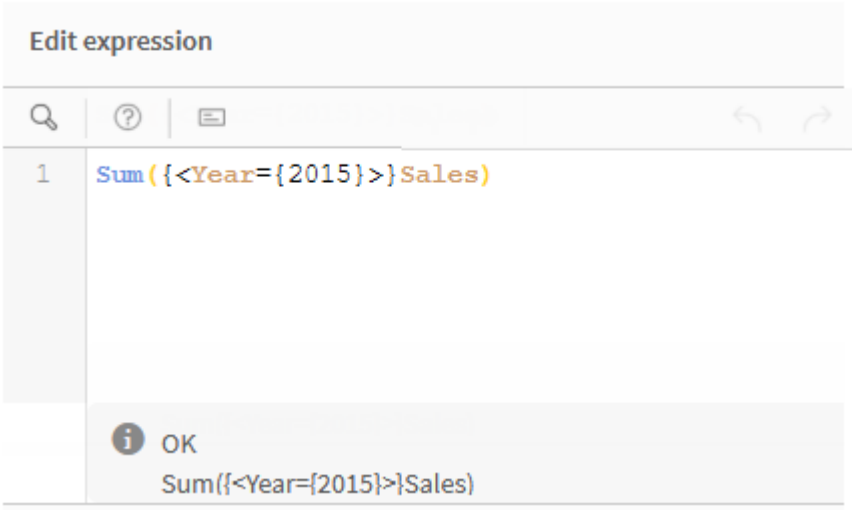
4. Kopiera och klistra in KPI för att skapa en ny KPI.
5. Klicka på det nya KPI, klicka på **Sales** under **Mått** och sedan på **Öppna uttrycksredigeraren**.



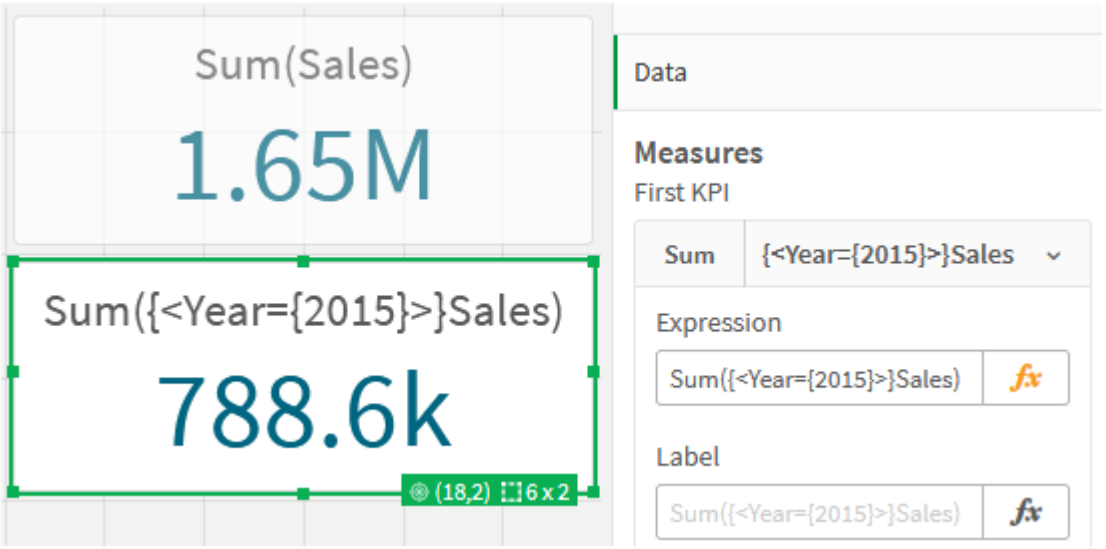
Uttrycksredigeraren öppnas med aggregeringen `Sum(Sales)`.



6. Skapa ett uttryck i uttrycksredigeraren för att få summan av Sales för endast 2015:
   i. Lägg till klammerparenteser för att indikera ett set-uttryck: `Sum({}Sales)`
   ii. Lägg till vinkelparenteser för att indikera en set-modifierare: `Sum({<>}Sales)`
   iii. I vinkelparentesen lägger du till fältet som ska väljas, i det här fallet är fältet `Year`, följt av ett likhetstecken. Sedan ska du omge 2015 med nya klammerparenteser. Det ger följande set-modifierare som resultat: `{<Year={2015}>}`.
   Hela uttrycket är:
   `Sum({<Year={2015}>}Sales)`

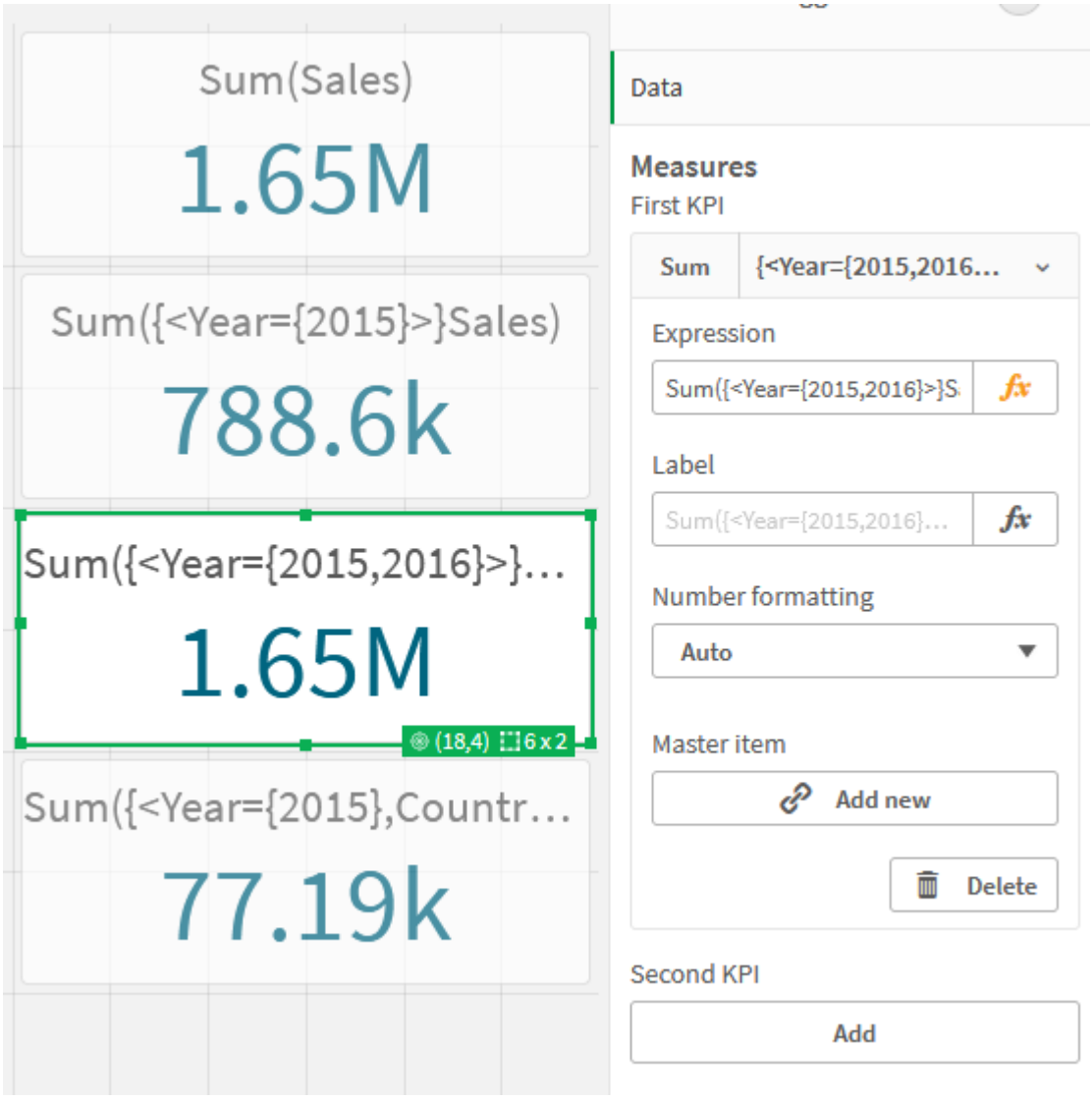


iv. Klicka på **Använd** för att spara uttrycket och avsluta uttrycksredigeraren. Summan av Sales för 2015 visas i KPI.



7. Skapa två KPI:er till med följande uttryck:
   `Sum({<Year={2015,2016}>}Sales)`
   Modifieraren ovan är `<Year={2015,2016}>`. Uttrycket kommer att returnera summan av Sales för 2015 och 2016.
   `Sum({<Year={2015},Country={'Germany'}>} Sales)`
   Modifieraren ovan är `<Year={2015}, Country={'Germany'}>`. Uttrycket kommer att returnera summan Sales för 2015, där 2015 skär Germany.

KPI:er som använder set-modifierare



Lägga till set-identifierare

Set-uttrycken ovan använder de aktuella urvalen som bas, eftersom en identifierare inte användes. Därefter lägger du till identifierare för att specificera beteendet när urval görs.

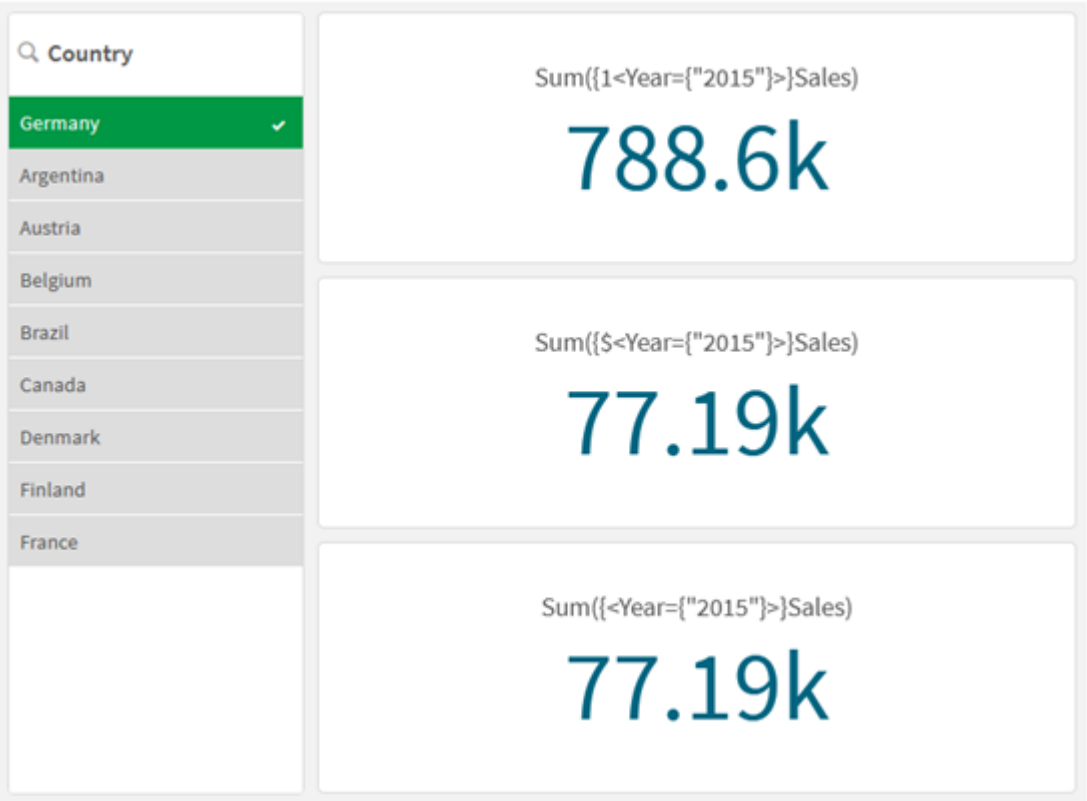
Gör följande:

På ditt ark ska du bygga eller kopiera följande set-uttryck:

`Sum({$<Year={"2015"}>}Sales)`
`$`-identifieraren baserar set-uttrycket på de aktuella urvalen som har gjorts i data. Det här är även standardbeteendet när en identifierare inte används.

`Sum({1<Year={"2015"}>}Sales)`
Identifieraren `1` kommer att orsaka att aggregeringen av `Sum(Sales)` på 2015 ignorerar det aktuella urvalet. Värdet för aggregeringen kommer inte att förändras när användaren gör andra urval. Till exempel när Germany väljs nedan ändras inte värdet för den aggregerade summan för 2015.

KPI:er som använder set-modifierare och set-identifierare



Lägga till operatorer

Set-operatorer används till att inkludera, utesluta eller genomskära datauppsättningar. Alla operatorer använder uppsättningar som operander och returnerar en uppsättning som resultat.

Du kan använda set-operatorer i två olika situationer:

- För att utföra en set-åtgärd på set-identifierare, som motsvarar uppsättningar med poster i data.
- För att utföra en set-åtgärd på elementuppsättningar, på fältvärden eller i en set-modifierare.

Gör följande:

På ditt ark ska du bygga eller kopiera följande set-uttryck:

```
Sum({$<Year={2015}>+1<Country={'Germany'}>}Sales)
```

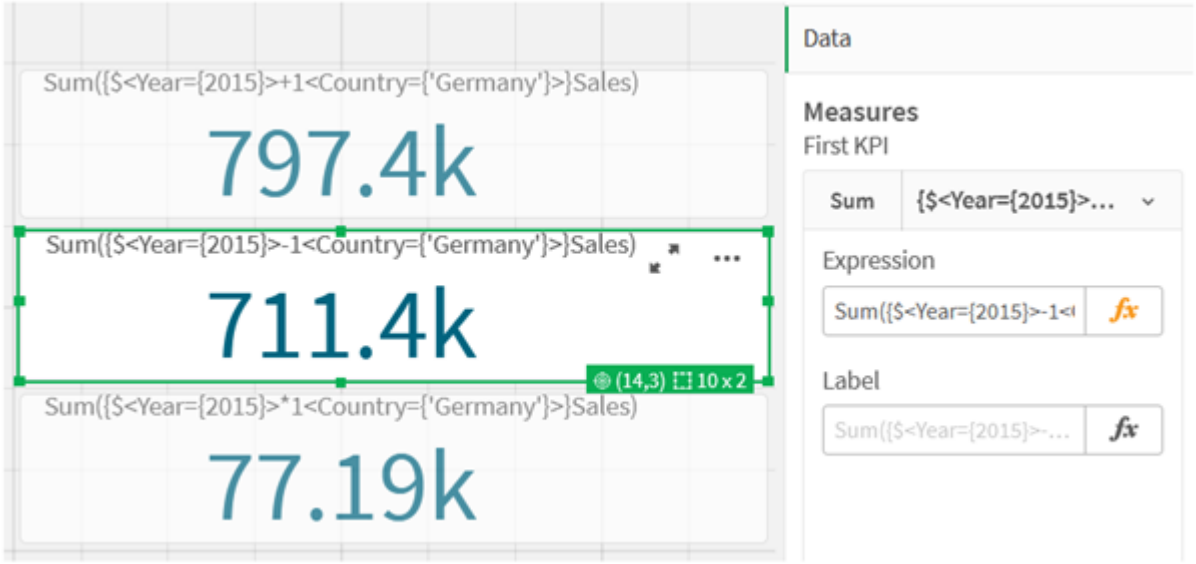
Operatorn med plustecknet `(+)` ger en union av datauppsättningarna för 2015 och Germany. Som vi förklarade för set-identifierarna ovan, innebär identifieraren med dollartecknet `($)` att de aktuella urvalen kommer att användas för den första operanden, `<Year={2015}>`, som respekteras. Identifieraren `1` innebär att urvalet kommer att ignoreras för den andra operanden, `<Country={'Germany'}>`.

KPI som använder operatorn (+)

Alternativt kan du använda ett minustecken `(-)` för att returnera en datauppsättning som består av posterna som tillhör 2015 men inte Germany. Eller så kan du använda en asterisk `(*)` till att returnera en uppsättning som består av poster som tillhör båda uppsättningarna.

```
Sum({$<Year={2015}>-1<Country={'Germany'}>}Sales)
Sum({$<Year={2015}>*1<Country={'Germany'}>}Sales)
```

KPI:er som använder operatorer



Introduktionsdata för set-uttryck

Laddningsskript

Ladda följande data som en inline-laddning och skapa sedan diagramuttrycken i introduktionen.

```
//Create table SalesByCountry
SalesByCountry:
Load * Inline [
Country, Year, Sales
Argentina, 2016, 66295.03
Argentina, 2015, 140037.89
Austria, 2016, 54166.09
Austria, 2015, 182739.87
```

```
Belgium, 2016, 182766.87
Belgium, 2015, 178042.33
Brazil, 2016, 174492.67
Brazil, 2015, 2104.22
Canada, 2016, 101801.33
Canada, 2015, 40288.25
Denmark, 2016, 45273.25
Denmark, 2015, 106938.41
Finland, 2016, 107565.55
Finland, 2015, 30583.44
France, 2016, 115644.26
France, 2015, 30696.98
Germany, 2016, 8775.18
Germany, 2015, 77185.68
];
```

Syntax för set-uttryck

Hela syntaxen (utan det möjliga alternativet med vanliga parenteser för att definiera företräde) beskrivs med Backus-Naur-formalism:

```
set_expression ::= { set_entity { set_operator set_entity } }

set_entity ::= set_identifier [ set_modifier ] | set_modifier

set_identifier ::= 1 | $ | $N | $_N | bookmark_id | bookmark_name

set_operator ::= + | - | * | /

set_modifier ::= < field_selection {, field_selection } >

field_selection ::= field_name [ = | += | -= | *= | /= ] element_set_
expression

element_set_expression ::= [ - ] element_set { set_operator element_set }

element_set ::= [ field_name ] | { element_list } | element_function

element_list ::= element { , element }

element_function ::= ( P | E ) ( [set_expression] [field_name] )

element ::= field_value | " search_mask "
```

6.3 Allmän syntax för diagramuttryck

Den här allmänna syntaxstrukturen kan användas för diagramuttryck, med många valfria parametrar:

```
expression ::= ( constant | expressionname | operator1 expression | expression operator2
expression | function | aggregation function | (expression ) )
```

där:

constant är en sträng (en text, ett datum eller en tid) inom enkla, raka citationstecken, eller ett tal. Konstanter skrivs utan tusentalsavgränsare och med decimalkomma som decimalavgränsare.

expressionname är namnet (etiketten) på ett annat uttryck i samma diagram.

operator1 är en unär operator (som arbetar med ett uttryck, det till höger).

operator2 är en binär operator (som arbetar med två uttryck, ett på varje sida).

```
function ::= functionname ( parameters )
```

```
parameters ::= expression { , expression }
```

Varken typen av parameter eller antalet parametrar är godtyckligt. utan beror på funktionen som används.

```
aggregationfunction ::= aggregationfunctionname ( parameters2 )
```

```
parameters2 ::= aggrexpression { , aggrexpression }
```

Varken typen av parameter eller antalet parametrar är godtyckligt. utan beror på funktionen som används.

6.4 Allmän syntax för aggregeringar

Den här allmänna syntaxstrukturen kan användas för aggregeringar, med många valfria parametrar:

```
aggrexpression ::= ( fieldref | operator1 aggrexpression | aggrexpression operator2
aggrexpression | functioninaggr | ( aggrexpression ) )
```

fieldref är ett fältnamn.

```
functionaggr ::= functionname ( parameters2 )
```

Uttryck och funktioner kan således nästlas efter önskemål. Så länge **fieldref** alltid är omslutet av exakt en aggregeringsfunktion, och under förutsättning att uttrycket returnerar ett värde som kan tolkas, ger Qlik Sense inte några felmeddelanden.

7 Operatorer

Det här avsnittet beskriver operatorerna som kan användas i Qlik Sense. Det finns två sorters operatorer:

- Unära operatorer (tar endast en operand)
- Binära operatorer (tar två operander)

De flesta operatorer är binära.

Följande operatorer kan definieras:

- Bit-operatorer
- Logiska operatorer
- Numeriska operatorer
- Relationsoperatorer
- Strängoperatorer

7.1 Bit-operatorer

Alla bit-operatorer konverterar (trunkerar) operanderna till signerade (32-bitars-) heltal och returnerar resultatet som signerade heltal. Alla operationer sker per bit på bitnivå. Om en operand inte kan tolkas som ett tal, returnerar operationen NULL.

Bit-operatorer

Operator	Fullständigt namn	Beskrivning
bitnot	Bit invers.	Unär operator. Operationen returnerar operandens logiska motsats på bitnivå. **Exempel:** `bitnot 17` returnerar -18
bitand	Bit och.	Operationen returnerar logiskt och mellan operanderna bit för bit. **Exempel:** `17 bitand 7` returnerar 1
bitor	Bit eller.	Operationen returnerar logiskt eller mellan operanderna bit för bit. **Exempel:** `17 bitor 7` returnerar 23

Operator	Fullständigt namn	Beskrivning
bitxor	Bit exklusivt eller.	Operationen returnerar logiskt bitvist exklusivt eller mellan operanderna. **Exempel:** `17 bitxor 7` returnerar 22
>>	Bit högerskift.	Operationen returnerar den första operanden skiftat till höger. Antalet steg anges i den andra operanden. **Exempel:** `8 >> 2` returnerar 2
<<	Bit vänsterskift.	Operationen returnerar den första operanden skiftat till vänster. Antalet steg anges i den andra operanden. **Exempel:** `8 << 2` returnerar 32

7.2 Logiska operatorer

Alla logiska operatorer tolkar operanderna logiskt och returnerar True (-1) eller False (0) som resultat.

Logiska operatorer

Operator	Beskrivning
not	Logisk motsats. En av de få unära operatorerna. Operationen returnerar operandens logiska motsats.
and	Logiskt och. Operationen returnerar operandernas logiska och.
or	Logiskt eller. Operationen returnerar operandernas logiska eller.
Xor	Logiskt exklusivt eller. Operationen returnerar operandernas logiska exklusiva eller. Dvs. som logiskt eller, men med den skillnaden att resultatet är False om båda operanderna är True.

7.3 Numeriska operatorer

Alla numeriska operatorer använder sig av operandernas numeriska värden och resulterar i ett numeriskt värde.

Numeriska operatorer

Operator	Beskrivning
+	Tecken för positivt tal (unär operator) eller aritmetisk addition. Den binära operatorn resulterar i summan av de två operanderna.
-	Tecken för negativt tal (unär operator) eller aritmetisk subtraktion. Den unära operationen resulterar i operanden multiplicerad med -1, den binära i skillnaden mellan de två operanderna.
*	Aritmetisk multiplikation. Operationen resulterar i produkten av de två operanderna.
/	Aritmetisk division. Operationen resulterar i kvoten av de två operanderna.

7.4 Relationsoperatorer

Alla relationsoperatorer jämför operandernas värden och returnerar True (-1) eller False (0) som resultat. Alla relationsoperatorer är binära.

Relationsoperatorer

Operator	Beskrivning
<	Mindre än. En numerisk jämförelse görs om båda operatorerna kan tolkas numeriskt. Operationen resulterar i jämförelsens logiska värde.
<=	Mindre än eller lika med. En numerisk jämförelse görs om båda operatorerna kan tolkas numeriskt. Operationen resulterar i jämförelsens logiska värde.
>	Större än. En numerisk jämförelse görs om båda operatorerna kan tolkas numeriskt. Operationen resulterar i jämförelsens logiska värde.
>=	Större än eller lika med. En numerisk jämförelse görs om båda operatorerna kan tolkas numeriskt. Operationen resulterar i jämförelsens logiska värde.
=	Är lika med. En numerisk jämförelse görs om båda operatorerna kan tolkas numeriskt. Operationen resulterar i jämförelsens logiska värde.
<>	Inte lika med. En numerisk jämförelse görs om båda operatorerna kan tolkas numeriskt. Operationen resulterar i jämförelsens logiska värde.

Operator	Beskrivning
precedes	Till skillnad från med operatorn **<** görs inget försök att tolka argumentens värden numeriskt före jämförelsen. Åtgärden returnerar sant om värdet till vänster om operatorn består av text som, enligt strängjämförelsen, kommer före det högra värdets text. **Exempel:** `'1 ' precedes ' 2'` returnerar FALSE `' 1' precedes ' 2'` returnerar TRUE som ASCII-värdet för ett blanksteg (” ”) har ett mindre värtde än ASCII-värdet för ett tal. Jämför detta med: `'1 ' < ' 2'` returnerar TRUE `' 1' < ' 2'` returnerar TRUE
follows	Till skillnad från med operatorn **>** görs inget försök att tolka argumentens värden numeriskt före jämförelsen. Åtgärden returnerar sant om värdet till vänster om operatorn består av text som, enligt strängjämförelsen, kommer efter det högra värdets text. **Exempel:** `' 2' follows '1'` returnerar FALSE ' 2' follows ' 1' returnerar TRUE som ASCII-värdet för ett blanksteg (” ”) har ett mindre värtde än ASCII-värdet för ett tal. Jämför detta med: `' 2' > ' 1'` returnerar TRUE `' 2' > '1 '` returnerar TRUE

7.5 Strängoperatorer

Det finns två textsträngsoperatorer. Den ena använder sig av operandernas textsträngsvärden och resulterar i en textsträng. Den andra jämför operanderna och returnerar ett booleskt värde som markerar matchning.

&

Konkatenering av strängar. Operationen resulterar i en textsträng som består av de två operandernas textsträngar, den ena efter den andra.

Exempel:

`'abc' & 'xyz'` returnerar 'abcxyz'

like

Jämför strängar med jokertecken. Åtgärden returnerar ett booleskt True (-1) om strängen som föregår operatorn matchas av strängen som följer efter operatorn. Den andra strängen kan innehålla jokertecknen * (valfritt antal av godtyckliga tecken) eller ? (ett godtyckligt tecken).

Exempel:

`'abc' like 'a*'` returnerar True (-1)

`'abcd' like 'a?c*'` returnerar True (-1)

`'abc' like 'a??bc'` returnerar False (0)

8 Skript- och diagramfunktioner

Omvandla och aggregera data med hjälp av funktioner i dataladdningsskript och diagramuttryck.

Många funktioner kan användas på samma sätt både i dataladdningsskript och diagramuttryck, men det finns ett antal undantag:

- Vissa funktioner kan endast användas i dataladdningsskript. Dessa markeras med – skriptfunktion.
- Vissa funktioner kan endast användas i diagramuttryck. Dessa markeras med – diagramfunktion.
- Vissa funktioner kan användas i både dataladdningsskript och diagramuttryck, men med olikheter i parametrar och användning. Dessa beskrivs i separata avsnitt markerade med – skriptfunktion eller – diagramfunktion.

8.1 Analytiska kopplingar för komplement på serversidan (SSE)

Funktioner som aktiveras genom analytiska kopplingar visas bara om du har konfigurerat de analytiska kopplingarna och Qlik Sense har startats.

Du konfigurerar de analytiska kopplingarna i QMC, se avsnittet "Skapa en analytisk koppling" i guiden Hantera Qlik Sense-webbplatser.

I Qlik Sense Desktop konfigurerar du de analytiska kopplingarna genom att redigera filen *Settings.ini*, se avsnittet ”Konfigurera analytiska kopplingar i Qlik Sense Desktop” i guiden Qlik Sense Desktop.

8.2 Aggregeringsfunktioner

Den grupp av funktioner som kallas aggregeringsfunktioner utgörs av funktioner som tar flera fältvärden som indata och returnerar ett enda resultat per grupp, där grupperingen definieras av en diagramdimension eller en **group by**-sats i skriptsatsen.

Aggregeringsfunktionerna omfattar **Sum()**, **Count()**, **Min()**, **Max()** med flera.

De flesta aggregeringsfunktioner kan användas i både dataladdningsskriptet och diagramuttryck, men syntaxen skiljer sig.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

När du namnger ett element, undvik att tilldela samma namn till mer än ett fält, variabel eller mått. Det finns en strikt prioritetsordning för att lösa konflikter mellan enheter med identiska namn. Denna ordning återspeglas i alla objekt eller sammanhang där dessa enheter används. Denna prioritetsordning är

följande:

- Inuti en aggregering prioriteras ett fält framför en variabel. Måttnamn är inte relevanta i aggregeringar och är inte prioriterade.
- Utanför en aggregering prioriteras en måttetikett före en variabel, som i sin tur prioriteras före ett fält.
- Utanför en aggregering kan ett mått dessutom återanvändas genom att referera till dess etikett, såvida inte etiketten faktiskt är en beräknad sådan. I det läget sjunker uttrycket i betydelse för att minska risken för självhänvisning, och i det här fallet kommer namnet alltid att först tolkas som ett måttnamn, därefter som ett fältnamn och för det tredje som ett variabelnamn.

Använda aggregeringsfunktioner i ett dataladdningsskript

Aggregeringsfunktioner kan enbart användas inuti **LOAD** - och **SELECT**-satser.

Använda aggregeringsfunktioner i diagramuttryck

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

En aggregeringsfunktion aggregerar över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan dock definieras med ett s.k. set-uttryck i set-analys.

Så beräknas aggregeringar

En aggregering körs i en slinga över posterna för en specifik tabell och aggregerar posterna i den. Till exempel räknar **Count**(<Field>) antalet poster i tabellen där <Field> finns. Om du bara vill aggregera distinkta fältvärden ska du använda **distinct**-satsen, som **Count**(**distinct** <Field>).

Om aggregeringsfunktionen innehåller fält från olika tabeller kör aggregeringsfunktionen en slinga över posterna från de olika tabellerna för de befintliga fälten. Detta påverkar prestandan och därför bör sådana aggregeringar undvikas, speciellt när du har stora datamängder.

Aggregering av nyckelfält

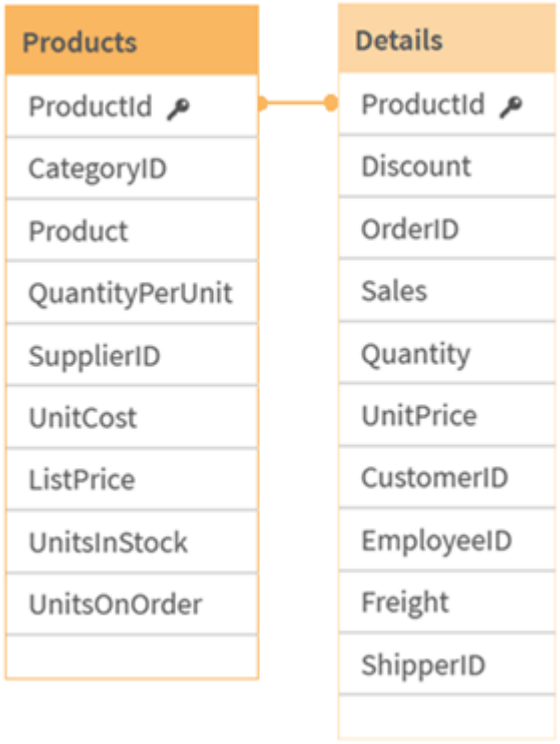
Sättet aggregeringar beräknas på gör att du inte kan aggregera nyckelfält, eftersom det inte framgår vilken tabell som ska användas för aggregeringen. Om fältet <Key> till exempel är länken mellan två tabeller, så framgår det inte om **Count**(<Key>) ska returnera antalet poster från den första eller andra tabellen.

Om du använder **distinct**-satsen är emellertid aggregeringen väldefinierad och kan beräknas för ett nyckelfältet som är länkat över två tabeller.

Om du har ett nyckelfält inuti en aggregeringsfunktion utan **distinct**-satsen returnerar Qlik Sense ett tal som kan vara meningslöst. Lösningen är att antingen använda **distinct**-satsen eller använda en kopia av nyckeln – en kopia som bara finns i en tabell.

I följande tabeller är *ProductId* nyckeln mellan tabellerna.

ProductId-nyckel mellan Products- och Details-tabeller



Count(ProductId) kan räknas antingen i tabellen *Products* (som bara har en post per produkt – *ProductId* är primärnyckeln) eller i tabellen *Details* (som antagligen har flera poster per produkt). Om du vill räkna antalet distinkta produkter ska du använda *Count(distinct ProductId)*. Om du vill räkna antalet rader i en specifik tabell ska du inte använda nyckeln.

Aggregeringar av nyckelfält som finns i tre eller fler tabeller

distinct-prefixet fungerar bara med nyckelfält som länkar upp till två tabeller. När en aggregering grupperas över ett nyckelfält som finns i tre eller fler tabeller kommer alla operationer där frekvensinformation krävs för ett fält returnera NULL. Om ett nyckelfält länkar tre eller fler tabeller måste en icke-nyckelkopia av fältet användas istället.

Grundläggande aggregeringsfunktioner

Grundläggande aggregeringsfunktioner - en översikt

Grundläggande aggregeringsfunktioner är en grupp av de vanligaste aggregeringsfunktionerna.

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Grundläggande aggregeringsfunktioner i dataladdningsskriptet

FirstSortedValue

FirstSortedValue() returnerar värdet från det uttryck som har angetts i **value** som motsvarar resultatet av sorteringen av **sort_weight**-argumentet, exempelvis namnet på produkten med det lägsta enhetspriset. Det n:te värdet i sorteringsordningen kan anges i **rank**. Om fler än ett resultatvärde delar samma **sort_weight** för den angivna **rank** returnerar funktionen NULL. De sorterade värdena itereras över ett antal poster, enligt vad som definieras i en **group by**-sats, eller aggregerat över den fullständiga datauppsättningen om ingen **group by**-sats har definierats.

```
FirstSortedValue ([ distinct ] expression, sort_weight [, rank ])
```

Max

Max() hittar det högsta numeriska värdet för aggregerade data i uttrycket, som definieras av en **group by**-sats. Genom att ange en **rank** n, återfinns det n:te högsta värdet.

```
Max ( expression[, rank])
```

Min

Min() returnerar det lägsta numeriska värdet för aggregerade data i uttrycket, som definieras av en **group by**-sats. Genom att ange en **rank** n återfinns det n:te lägsta värdet.

```
Min ( expression[, rank])
```

Mode

Mode() returnerar det vanligaste värdet, lägesvärdet, för aggregerade data i uttrycket, som definieras av en **group by**-sats. Funktionen **Mode()** kan returnera numeriska värden såväl som textvärden.

```
Mode (expression )
```

Only

Only() returnerar ett värde om det finns ett, och endast ett, möjligt resultat från aggregerade data. Om poster endast innehåller ett värde returneras detta värde. Annars returneras NULL. Använd **group by**-satsen för att utvärdera över flera poster. Funktionen **Only()** kan returnera numeriska värden och textvärden.

```
Only (expression )
```

Sum

Sum() beräknar summan av värden aggregerade i uttrycket, som definieras av en **group by**-sats.

```
Sum ([distinct]expression)
```

Grundläggande aggregeringsfunktioner i diagramuttryck

Aggregeringsfunktioner för diagram kan endast användas på fält i diagramuttryck. Argumentuttrycket för en aggregeringsfunktion får inte innehålla en annan aggregeringsfunktion.

FirstSortedValue

FirstSortedValue() returnerar värdet från det uttryck som har angetts i **value** som motsvarar resultatet av sorteringen av **sort_weight**-argumentet, exempelvis namnet på produkten med det lägsta enhetspriset. Det n:te värdet i sorteringsordningen kan anges i **rank**. Om fler än ett resultatvärde delar samma **sort_weight** för den angivna **rank** returnerar funktionen NULL.

```
FirstSortedValue - diagramfunktion([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] value, sort_weight [,rank])
```

Max

Max() finner det högsta värdet för aggregerade data. Genom att ange en **rank** n, återfinns det n:te högsta värdet.

```
Max - diagramfunktion([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Min

Min() finner det lägsta värdet för aggregerade data. Genom att ange en **rank** n återfinns det n:te lägsta värdet.

```
Min - diagramfunktion([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Mode

Mode() finner det vanligast förekommande värdet, lägesvärdet, i aggregerade data. Funktionen **Mode()** kan behandla både textvärden och numeriska värden.

```
Mode - diagramfunktion ({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Only

Only() returnerar ett värde om det finns ett, och endast ett, möjligt resultat från aggregerade data. Till exempel returnerar en sökning efter den enda produkten med styckpriset =9 NULL om fler än en produkt har styckpriset 9.

```
Only - diagramfunktion([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr)
```

Sum

Sum() beräknar summan av de värden som ges av uttrycket eller fältet över aggregerade data.

```
Sum - diagramfunktion([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr])
```

FirstSortedValue

FirstSortedValue() returnerar värdet från det uttryck som har angetts i **value** som motsvarar resultatet av sorteringen av **sort_weight**-argumentet, exempelvis namnet på produkten med det lägsta enhetspriset. Det n:te värdet i sorteringsordningen kan anges i **rank**. Om fler än ett resultatvärde delar samma **sort_**

weight för den angivna **rank** returnerar funktionen NULL. De sorterade värdena itereras över ett antal poster, enligt vad som definieras i en **group by**-sats, eller aggregerat över den fullständiga datauppsättningen om ingen **group by**-sats har definierats.

Syntax:

```
FirstSortedValue ([ distinct ] value, sort-weight [, rank ])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
value Expression	Funktionen finner värdet av uttrycket **value** som motsvarar resultatet vid sortering av **sort_weight**.
sort-weight Expression	Uttrycket som innehåller data som ska sorteras. Det första (lägsta) värdet i **sort_weight** hittas, varifrån motsvarande värde i uttrycket **value** fastställs. Om du sätter ett minustecken framför **sort_weight** returnerar funktionen det sista (högsta) sorterade värdet i stället.
rank Expression	Genom att ange **rank** "n" som är större än 1 får du det n:te sorterade värdet.
distinct	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår applikation för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Skriptexempel

Exempel	Resultat
Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|CustomerID Astrida|AA|1|10|1 Astrida|AA|7|18|1 Astrida|BB|4|9|1 Astrida|CC|6|2|1 Betacab|AA|5|4|2 Betacab|BB|2|5|2 Betacab|DD|12|25|2 Canutility|AA|3|8|3 Canutility|CC|13|19|3 Divadip|AA|9|16|4 Divadip|AA|10|16|4 Divadip|DD|11|10|4] (delimiter is '|'); FirstSortedValue: LOAD Customer,FirstSortedValue(Product, UnitSales) as MyProductWithSmallestOrderByCustomer Resident Temp Group By Customer;	Customer MyProductWithSmallestOrderByCustomer Astrida CC Betacab AA Canutility AA Divadip DD Funktionen sorterar UnitSales från minsta till största, och letar efter värdet av Customer med det minsta värdet av UnitSales, den minsta ordern. Eftersom CC motsvarar den minsta ordern (värdet UnitSales = 2) för kunden Astrida. AA motsvarar den minsta ordern (4) för kunden Betacab, AA motsvarar den minsta ordern (8) för kunden Canutility, och DD motsvarar den minsta ordern (10) för kunden Divadip..

Exempel	Resultat

Givet att **Temp**-tabellen är laddad som i föregående exempel:
 	
`LOAD Customer,FirstSortedValue(Product, -UnitSales) as MyProductWithLargestOrderByCustomer Resident Temp Group By Customer;`
	`Customer MyProductWithLargestOrderByCustomer`
	
	`Astrida AA`
	
	`Betacab DD`
	
	`Canutility CC`
	
	`Divadip -`
	Argumentet sort_weight inleds med ett minustecken så att funktionen sorterar den största först.
	
	Eftersom AA motsvarar den största ordern (värdet av UnitSales:18) för kunden Astrida, DD motsvarar den största ordern (12) för kunden Betacab, och CC motsvarar den största ordern (13) för kunden Canutility. Det finns två identiska värden för den största ordern (16) för kunden Divadip, därför producerar detta ett null-resultat.

Givet att **Temp**-tabellen är laddad som i föregående exempel:
 	
`LOAD Customer,FirstSortedValue(distinct Product, -UnitSales) as MyProductWithSmallestOrderByCustomer Resident Temp Group By Customer;`
	`Customer MyProductWithLargestOrderByCustomer`
	
	`Astrida AA`
	
	`Betacab DD`
	
	`Canutility CC`
	
	`Divadip AA`
	Detta är samma som i föregående exempel, förutom att distinct kvalificeraren används. Detta gör att dubblettresultatet för Divadip ignoreras, så att ett icke-null-värde kan returneras.

FirstSortedValue - diagramfunktion

FirstSortedValue() returnerar värdet från det uttryck som har angetts i **value** som motsvarar resultatet av sorteringen av **sort_weight**-argumentet, exempelvis namnet på produkten med det lägsta enhetspriset. Det n:te värdet i sorteringsordningen kan anges i **rank**. Om fler än ett resultatvärde delar samma **sort_weight** för den angivna **rank** returnerar funktionen NULL.

Syntax:

```
FirstSortedValue([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] value,
sort_weight [,rank])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
value	Utdatafält. Funktionen finner värdet av uttrycket **value** som motsvarar resultatet vid sortering av **sort_weight**.
sort_weight	Indatafält. Uttrycket som innehåller data som ska sorteras. Det första (lägsta) värdet i **sort_weight** hittas, varifrån motsvarande värde i uttrycket **value** fastställs. Om du sätter ett minustecken framför **sort_weight** returnerar funktionen det sista (högsta) sorterade värdet i stället.
rank	Genom att ange **rank** "n" som är större än 1 får du det n:te sorterade värdet.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel: diagramuttryck

Exempel	Resultat
`FirstSortedValue (Product, UnitPrice)`	Returnerar den `Product` som har lägst `UnitPrice`.
`FirstSortedValue (Product, UnitPrice, 2)`	Returnerar den `Product` som har näst lägst `UnitPrice`.
`FirstSortedValue (Customer, -UnitPrice, 2)`	Returnerar den `Customer` med den `Product` som har näst högst `UnitPrice`.
`FirstSortedValue (Customer, -UnitPrice*UnitSales, 2)`	Returnerar den `Customer` med näst högst säljordervärdet, `UnitPrice` multiplicerat med `UnitSales`.

Exempel – grunderna i FirstSortedValue

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Customer`
  - `Product`
  - `UnitSales`
  - `UnitPrice`

Laddningsskript

```
Example:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som mått:

- `=FirstSortedValue (Product, UnitPrice)`, för att beräkna produkten med det lägsta priset per enhet.
- `=FirstSortedValue (Product, UnitPrice, 2)`, för att beräkna produkten med det näst lägsta priset per enhet.
- `=FirstSortedValue (Customer, -UnitPrice, 2)`, för att beräkna kunden med den produkt som har det näst högsta priset per enhet.
- `=FirstSortedValue (Customer, UnitPrice, 3)`, för att beräkna kunden med den produkt som har det tredje högsta priset per enhet.
- `=FirstSortedValue (Customer, -UnitPrice*UnitSales, 2)`, för att beräkna kunden med näst högst säljordervärdet, `UnitPrice` multiplicerat med `UnitSales`.

Results table

FirstSortedValue (Product, UnitPrice)	FirstSortedValue (Product, UnitPrice, 2)	FirstSortedValue (Customer, -UnitPrice, 2)	FirstSortedValue (Customer, UnitPrice, 3)	FirstSortedValue (Customer, -UnitPrice*UnitSales, 2)
BB	BB	Betacab	-	Canutility

Den första kolumnen returnerar värdet `BB` eftersom detta är den produkt som har lägst `UnitPrice` (9)

Den andra kolumnen returnerar värdet `BB` eftersom detta är produkten med näst lägst `UnitPrice` (10)

Den tredje kolumnen ger värdet `Betacab`, som är den `Customer` med den `Product` som har näst högst `UnitPrice` (20).

Den fjärde kolumnen ger värdet `NULL`, eftersom det finns två värden på `Customer` (`Astrida` och `Canutility`) med samma rangordning (tredje lägsta) `UnitPrice` `15`.

Den femte kolumnen returnerar `Canutility`, som är den `Customer` som har näst högst säljordervärde, `UnitPrice` multiplicerat med `UnitSales` (120).

Exempel – scenario för FirstSortedValue

Diagramuttryck

Översikt

En datauppsättning innehåller försäljningssiffror för årets tre första månader. Regionchefen vill identifiera de bäst och sämst säljande produkterna för att planera för framtida lagerhållning.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Date`
  - `SalesProduct`
  - `Category`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Date,SalesProduct, Category, Sales
2024-01-01, Apple, Fruit, 100
2024-01-15, Banana, Fruit, 80
2024-01-30, Carrot, Vegetable, 50
2024-02-05, Apple, Fruit, 120
```

```
2024-02-20, Banana, Fruit, 90
2024-02-28, Carrot, Vegetable, 60
2024-03-10, Apple, Fruit, 110
2024-03-22, Banana, Fruit, 85
2024-03-31, Carrot, Vegetable, 55
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Category`

Skapa följande mått:

- `=FirstSortedValue(SalesProduct, -Sales)`, för att beräkna den bäst säljande produkten i varje kategori.
- `=FirstSortedValue(TOTAL SalesProduct, -Sales)`, för att beräkna den bäst säljande produkten överlag.
- `=FirstSortedValue(TOTAL SalesProduct, Sales)`, för att beräkna den lägst säljande produkten överlag.

Results table

Category	FirstSortedValue (SalesProduct, -Sales)	FirstSortedValue(TOTAL SalesProduct, -Sales)	FirstSortedValue(TOTAL SalesProduct, Sales)
Fruit	Apple	Apple	Carrot
Vegetable	Carrot	Apple	Carrot

Resultatet visar att `Apple` och `Carrot` är de mest sålda produkterna i sina respektive kategorier, `Fruit` och `Vegetable`. I alla kategorier var dock `Apple` den bäst säljande produkten och `Carrot` den sämst säljande produkten.

Max

Max() hittar det högsta numeriska värdet för aggregerade data i uttrycket, som definieras av en **group by**-sats. Genom att ange en **rank** n, återfinns det n:te högsta värdet.

Syntax:

```
Max ( expr [, rank])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.
rank Expression	Standardvärdet för **rank** är 1, vilket motsvarar det högsta värdet. Om du anger **rank** som 2 returneras det näst högsta värdet. Om **rank** är 3 returneras det tredje högsta värdet.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår applikation för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Exempel:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');


Max:

LOAD Customer, Max(UnitSales) as MyMax Resident Temp Group By Customer;
```

Resultattabell

Kund	MyMax
Astrida	18
Betacab	5
Canutility	8

Exempel:

Givet att **Temp**-tabellen är laddad som i föregående exempel:

```
LOAD Customer, Max(UnitSales,2) as MyMaxRank2 Resident Temp Group By Customer;
```

Resultattabell

Kund	MyMaxRank2
Astrida	10
Betacab	4
Canutility	-

Max - diagramfunktion

Max() finner det högsta värdet för aggregerade data. Genom att ange en **rank** n, återfinns det n:te högsta värdet.

*Titta gärna på **FirstSortedValue** och **RangeMax**, som har liknande funktionalitet som **Max**-funktionen.*

Syntax:

```
Max([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Returnerad datatyp: numeriska

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
rank	Standardvärdet för **rank** är 1, vilket motsvarar det högsta värdet. Om du anger **rank** som 2 returneras det näst högsta värdet. Om **rank** är 3 returneras det tredje högsta värdet.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel: diagramuttryck

Exempel	Resultat
`Max(UnitSales)`	Returnerar det högsta värdet i fältet `UnitSales`.
`Max(UnitSales*UnitPrice)` Värdet för en order beräknas utifrån multiplicering av antal sålda enheter i (`UnitSales`) med priset per enhet.	Returnerar det högsta värdet som blir resultatet av att beräkna alla möjliga värden av (`UnitSales`)*(`UnitPrice`).
`Max(UnitSales, 2)`	Returnerar värdet för det näst högsta värdet.
`Max(TOTAL UnitSales)`	`TOTAL`-kvalificeraren innebär att det högsta möjliga värdet hittas, oavsett diagramdimensionerna. För ett diagram med `Customer` som dimension säkerställer kvalificeraren `TOTAL` att det maximala värdet för hela datauppsättningen returneras, i stället för max `UnitSales` för varje kund.
`Max({1} TOTAL UnitSales)`	Det fasta analysuttrycket `{1}` definierar den uppsättning poster som ska utvärderas som `ALL`, oavsett vilket urval som har gjorts. Om t.ex. en specifik kund väljs kommer den fortfarande att returnera det högsta `UnitSales` över hela datauppsättningen.

Exempel – grunderna i Max

Diagramuttryck

Översikt

En datauppsättning innehåller försäljningssiffror för produkter per månad. Vi vill identifiera det högsta försäljningsvärdet för varje produkt och den månad då detta inträffade.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Customer`
  - `UnitSales`
  - `UnitPrice`

Laddningsskript

```
Example:
LOAD * inline [
Customer|UnitSales|UnitPrice
Astrida|4|16
Astrida|10|15
```

```
Astrida|9|9
Betacab|5|10
Betacab|2|20
Betacab||25
Canutility|8|15
Canutility||19
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=Max(UnitSales)`, för att beräkna det maximala värdet för `UnitSales`.
- `=Max(UnitSales*UnitPrice)`, för att beräkna det maximala värdet efter att ha multiplicerat `UnitSales` med `UnitPrice` rad för rad.
- `=Max(UnitSales, 2)`, för att beräkna det näst högsta värdet.
- `=Max(TOTAL UnitSales)`, för att beräkna den högsta summan oavsett diagramdimension.
- `=Max({1} TOTAL UnitSales)`, för att beräkna den högsta summan som ignorerar alla val.

Results table

Customer	Max (UnitSales)	Max (UnitSales*UnitPrice)	Max (UnitSales, 2)	Max(TOTAL UnitSales)	Max({1} TOTAL UnitSales)
Totals	**10**	**150**	**9**	**10**	**10**
Astrida	10	150	9	10	10
Betacab	5	50	2	10	10
Canutility	8	120	-	10	10

I detta exempel ska du observera följande:

- I det första mätvärdet (`Max(UnitSales)`) är det högsta värdet för `UnitSales` för kunden `Astrida` `10` . Cellen `Totals` i den här kolumnen ger `10` eftersom det här värdet också är det högsta `UnitSales`-värdet i hela datauppsättningen.
- I det andra mätvärdet (`Max(UnitSales*UnitPrice)`) returneras `150` för `Astrida`. Detta är det högsta värdet för den kunden när `UnitSales` multipliceras med `UnitPrice`. Cellen `Totals` för den här kolumnen returnerar också `150` eftersom det även är det högsta beräknade värdet i hela datauppsättningen.
- I det tredje mätvärdet (`Max(UnitSales, 2)`) är det näst högsta värdet för `UnitSales` för kunden `Astrida` `9` . Detta värde är också det näst högsta värdet för alla tre kunderna, och därför returnerar raden `Totals` också `9`.

- Det fjärde mätvärdet (`Max(TOTAL UnitSales)`) returnerar värdet `10`. Detta är det högsta värde som hittats oberoende av kundens dimension, därför returnerar alla rader samma värde. Om du däremot väljer kunden `Betacab` kommer tabellen endast att returnera värden för den kunden, i det här fallet `5` eftersom det är högst `UnitSales` bland alla produkter för `Betacab`.
- Det femte mätvärdet (`Max({1} TOTAL UnitSales)`) returnerar värdet `10` oavsett vilken `Customer` som väljs eftersom set-modifierad `{1}` används. Om du t.ex. väljer `Betacab` kommer denna kolumn att fortsätta att returnera värdet `10`.

Exempel – Max-scenario som beräknar högsta försäljning per månad

Diagramuttryck

Översikt

En datauppsättning innehåller försäljningssiffror för produkter per månad. En försäljningschef vill identifiera det högsta försäljningsvärdet för varje produkt och den månad då detta inträffade.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Product`
  - `Sales`
  - `Month`

Laddningsskript

```
Example:
Load * inline [
Product,Sales,Month
A, 500, 2024-01-01
A, 700, 2024-02-01
A, 600, 2024-03-01
B, 300, 2024-01-01
B, 800, 2024-02-01
B, 450, 2024-03-01
C, 550, 2024-01-01
C, 620, 2024-02-01
C, 710, 2024-03-01
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Product`
- `Month`

Skapa följande mått:

- `=If(Sales = Max(TOTAL <Product> Sales), Sales)`, för att beräkna den maximala försäljningen för varje produkt. I egenskapspanelen för måttet väljer du **Tillägg** > **Datahantering**. Rensa kryssrutan **Inkludera nollvärden** för att ta bort eventuella nollvärden.

Results table

Product	Month	If(Sales = Max(TOTAL <Product> Sales), Sales)
A	2024-02-01	700
B	2024-02-01	800
C	2024-03-01	710

Resultatet av mätningen ger det maximala försäljningsvärdet för varje produkt och den månad då detta värde uppnåddes. Till exempel hade `Product B` den högsta försäljningen (`800`) i `2024-02-01`.

Se även:

p *FirstSortedValue - diagramfunktion (page 374)*
p *RangeMax (page 1591)*

Min

Min() returnerar det lägsta numeriska värdet för aggregerade data i uttrycket, som definieras av en **group by**-sats. Genom att ange en **rank** n återfinns det n:te lägsta värdet.

Syntax:

```
Min ( expr [, rank])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.
rank Expression	Standardvärdet för **rank** är 1, vilket motsvarar det lägsta värdet. Om du anger **rank** som 2, returneras det näst lägsta värdet. Om **rank** är 3, returneras det tredje lägsta värdet och så vidare.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår applikation för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Exempel:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');
Min:

LOAD Customer, Min(UnitSales) as MyMin Resident Temp Group By Customer;
```

Resultattabell

Kund	MyMin
Astrida	2
Betacab	4
Canutility	8

Exempel:

Givet att **Temp**-tabellen är laddad som i föregående exempel:

```
LOAD Customer, Min(UnitSales,2) as MyMinRank2 Resident Temp Group By Customer;
```

Resultattabell

Kund	MyMinRank2
Astrida	9
Betacab	5
Canutility	-

Min - diagramfunktion

Min() finner det lägsta värdet för aggregerade data. Genom att ange en **rank** n återfinns det n:te lägsta värdet.

*Titta gärna på **FirstSortedValue** och **RangeMin**, som har liknande funktionalitet som **Min**-funktionen.*

Syntax:

```
Min({[SetExpression] [TOTAL [<fld {,fld}>]]} expr [,rank])
```

Returnerad datatyp: numeriska

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
rank	Standardvärdet för **rank** är 1, vilket motsvarar det lägsta värdet. Om du anger **rank** som 2, returneras det näst lägsta värdet. Om **rank** är 3, returneras det tredje lägsta värdet och så vidare.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel: diagramuttryck

Exempel	Resultat
`Min(UnitSales)`	Returnerar det lägsta null-värdet i fältet `UnitSales`.
`Min(UnitSales*UnitPrice)` Värdet för en order beräknas utifrån multiplicering av antal sålda enheter i (`UnitSales`) med priset per enhet.	Returnerar det lägsta nullvärdet som blir resultatet av att beräkna alla möjliga värden av (`UnitSales`)*(`UnitPrice`).
`Min(UnitSales, 2)`	Returnerar det näst lägsta värdet i `UnitSales` (efter NULL-värdena).
`Min(TOTAL UnitSales)`	`TOTAL`-kvalificeraren innebär att det lägsta möjliga värdet hittas, oavsett diagramdimensionerna. För ett diagram med `Customer` som dimension säkerställer kvalificeraren `TOTAL` det minsta värdet över den fullständiga datauppsättningen returneras, i stället för minsta `UnitSales` för varje kund.
`Min({1} TOTAL UnitSales)`	Det fasta analysuttrycket `{1}` definierar den uppsättning poster som ska utvärderas som `ALL`, oavsett vilket urval som har gjorts. Om t.ex. en specifik kund väljs kommer den fortfarande att returnera minimi av `UnitSales` över hela datauppsättningen.

Exempel – grunderna i Min

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Customer`
  - `UnitSales`
  - `UnitPrice`

Laddningsskript

```
Example:
LOAD * inline [
Customer|UnitSales|UnitPrice
Astrida|4|16
Astrida|10|15
Astrida|9|9
Betacab|5|10
Betacab|2|20
Betacab||25
Canutility|8|15
Canutility||19
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=Min(UnitSales)`, för att beräkna det minimala värdet för `UnitSales`.
- `=Min(UnitSales*UnitPrice)`, för att beräkna det minimala värdet efter att ha multiplicerat `UnitSales` med `UnitPrice` rad för rad.
- `=Min(UnitSales, 2)`, för att beräkna det näst lägsta värdet.
- `=Min(TOTAL UnitSales)`, för att beräkna den lägsta summan oavsett diagramdimension.
- `=Min({1} TOTAL UnitSales)`, för att beräkna den lägsta summan som ignorerar alla val.

Results table

Customer	Min (UnitSales)	Min (UnitSales*UnitPrice)	Min (UnitSales, 2)	Min(TOTAL UnitSales)	Min({1} TOTAL UnitSales)
Totals	**2**	**40**	**4**	**2**	**2**
Astrida	4	64	9	2	2
Betacab	2	40	5	2	2
Canutility	8	120	-	2	2

I detta exempel ska du observera följande:

- I det första mätvärdet (`Min(UnitSales)`) är det lägsta värdet för `UnitSales` för kunden `Astrida` `4` . Cellen `Totals` i den här kolumnen ger `2` eftersom det här värdet är det lägsta `UnitSales`-värdet i hela datauppsättningen.
- I det andra mätvärdet (`Min(UnitSales*UnitPrice)`) returneras `64` för `Astrida`. Detta är det lägsta värdet för den kunden när `UnitSales` multipliceras med `UnitPrice`. Cellen `Totals` för den här kolumnen returnerar `40` eftersom det även är det lägsta beräknade värdet i hela datauppsättningen.
- I det tredje mätvärdet (`Min(UnitSales, 2)`) är det näst lägsta värdet för `UnitSales` för kunden `Astrida` `9` . Värdet `4` returneras i raden `Totals`, eftersom det är det näst lägsta `UnitSales`-värdet för alla kunder.
- Det fjärde mätvärdet (`Min(TOTAL UnitSales)`) returnerar värdet `2`. Detta är det lägsta möjliga icke-null-värdet oberoende av kundens dimension, därför returnerar alla rader samma värde. Om du däremot väljer kunden `Betacab` kommer tabellen endast att returnera värden för den kunden, i det här fallet `2` eftersom det är lägst `UnitSales` bland alla produkter för `Betacab`.
- Det femte mätvärdet `Min({1} TOTAL UnitSales)`, returnerar `2` oavsett vilket värde som valts i fältet `Customer` eftersom åtgärden beräknar den lägsta totalsumman. Om du t.ex. väljer `Betacab` kommer denna kolumn att fortsätta att returnera värdet `2`.

Exempel – Min-scenario som beräknar lägsta försäljning per månad

Diagramuttryck

Översikt

En datauppsättning innehåller försäljningssiffror för produkter per månad. En försäljningschef vill identifiera det lägsta försäljningsvärdet för varje produkt och den månad då detta inträffade.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:

- `Product`
- `Sales`
- `Month`

Laddningsskript

```
Example:
Load * inline [
Product,Sales,Month
A, 500, 2024-01-01
A, 700, 2024-02-01
A, 600, 2024-03-01
B, 300, 2024-01-01
B, 800, 2024-02-01
B, 450, 2024-03-01
C, 550, 2024-01-01
C, 620, 2024-02-01
C, 710, 2024-03-01
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Product`
- `Month`

Skapa följande mått:

- `=If(Sales = Min(TOTAL <Product> Sales), Sales)`, för att beräkna den minimala försäljningen för varje produkt. I egenskapspanelen väljer du **Tillägg** > **Datahantering**. Rensa kryssrutan **Inkludera nollvärden**.

Results table

Product	Month	If(Sales = Min(TOTAL <Product> Sales), Sales)
A	2024-01-01	500
B	2024-01-01	300
C	2024-01-01	550

Resultatet av mätningen ger det minimala försäljningsvärdet för varje produkt och den månad då detta värde uppnåddes. Till exempel hade `Product B` den lägsta försäljningen (`300`) i `2024-01-01`.

Se även:

p *FirstSortedValue - diagramfunktion (page 374)*
p *RangeMin (page 1595)*

Mode

Mode() returnerar det vanligaste värdet, lägesvärdet, för aggregerade data i uttrycket, som definieras av en **group by**-sats. Funktionen **Mode()** kan returnera numeriska värden såväl som textvärden.

Syntax:

```
Mode ( expr)
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.

Begränsningar:

Om fler än ett värde är lika vanligt förekommande returneras NULL.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår applikation för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Skriptexempel

Exempel	Resultat
Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|CustomerID Astrida|AA|1|10|1 Astrida|AA|7|18|1 Astrida|BB|4|9|1 Astrida|CC|6|2|1 Betacab|AA|5|4|2 Betacab|BB|2|5|2 Betacab|DD Canutility|DD|3|8 Canutility|CC] (delimiter is '|'); Mode: LOAD Customer, Mode(Product) as MyMostOftenSoldProduct Resident Temp Group By Customer;	MyMostOftenSoldProduct AA eftersom AA är den enda produkt som har sålts mer än en gång.

Mode - diagramfunktion

Mode() finner det vanligast förekommande värdet, lägesvärdet, i aggregerade data. Funktionen **Mode()** kan behandla både textvärden och numeriska värden.

Syntax:

```
Mode({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel: diagramuttryck

Exempel	Resultat
`Mode(UnitSales)`	Returnerar det vanligast förekommande värdet i `UnitSales`. Returnerar `NULL (-)`. Inget enskilt värde uppträder oftare än ett annat.
`Mode(Customer)`	Returnerar det vanligast förekommande värdet i `Customer`. Returnerar `NULL (-)`. Inget enskilt värde uppträder oftare än ett annat.
`Mode (TOTAL UnitSales)`	`TOTAL`-kvalificeraren returnerar det vanligast förekommande värdet fortfarande i `Unitsales`, oavsett diagramdimensioner.
`Mode({1} TOTAL UnitSales)`	Det fasta analysuttrycket `{1}` definierar den uppsättning poster som ska utvärderas som `ALL`, oavsett vilket urval som har gjorts. Om t.ex. en specifik kund väljs kommer den fortfarande att returnera det vanligast förekommande värdet av `UnitSales` över hela datauppsättningen.

Exempel – grunderna i Mode

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `Customer`
    - `UnitSales`
    - `UnitPrice`

Laddningsskript

```
Example:
LOAD * inline [
Customer|UnitSales|UnitPrice
Astrida|4|16
Astrida|10|15
Astrida|9|9
Astrida|10|10
Betacab|5|10
Betacab|4|20
Betacab||25
Canutility|8|15
Canutility|4|19
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=Mode(UnitSales)`, för att beräkna det vanligast förekommande `UnitSales` värdet.
- `=Mode(Customer)`, för att beräkna det vanligast förekommande `Customer` värdet.
- `=Mode(TOTAL UnitSales)`, för att beräkna det vanligast förekommande `UnitSales`-värdet i hela datauppsättningen, utan hänsyn till diagramdimensionerna. Om ett val görs kommer värdet att justeras för att återspegla detta val.
- `=Mode({1} TOTAL UnitSales)`, för att beräkna det vanligast förekommande `UnitSales`-värdet, utan hänsyn till urval. Om ett val görs förblir värdet konstant.

Results table

Customer	Mode (UnitSales)	Mode (Customer)	Mode(TOTAL UnitSales)	Mode({1} TOTAL UnitSales)
Totals	**4**	**Astrida**	**4**	**4**
Astrida	10	Astrida	4	4
Betacab	-	Betacab	4	4
Canutility	-	Canutility	4	4

I detta exempel ska du observera följande:

- I det första mätvärdet (`Mode(UnitSales)`) är det vanligast förekommande värdet för kunden `Astrida 10`. Cellen `Totals` returnerar `4`, som är det vanligast förekommande värdet för `UnitSales` i hela datauppsättningen. Det finns inga instanser av de vanligast förekommande värdena för `Betacab` och `Canutility`, så resultatet ger `NULL` för dessa kunder.
- I det andra mätvärdet (`Mode(Customer)`) returnerar resultaten varje kundnamn. Cellen `Totals` returnerar dock `Astrida` eftersom det är det vanligast förekommande kundvärdet i hela datauppsättningen.
- Det tredje mätvärdet (`Mode(TOTAL UnitSales)`) returnerar värdet 4. Om ingen `Customer` väljs kommer alla celler i den här kolumnen att visa värdet 4. Om du väljer en kund, t.ex. `Betacab`, kommer den här kolumnen att ge `NULL` eftersom det inte finns något vanligt förekommande värde för `Betacab`.
- Det fjärde mätvärdet (`Mode({1} TOTAL UnitSales)`) returnerar `4` oavsett eventuella val i fältet `Customer` eftersom uttrycket för set-analys `{1}` definierar den uppsättning poster som ska utvärderas som `ALL`. Om du t.ex. väljer `Betacab` kommer denna kolumn att fortsätta att returnera värdet `4`.

Exempel – Lägesbeskrivning för att fastställa avvikelser i försäljningsdata

Diagramuttryck

Översikt

En datauppsättning innehåller försäljningssiffror per månad. Företaget vill se det vanligast förekommande försäljningsvärdet för varje produkt och i hela datauppsättningen. Man vill också jämföra försäljningen per produkt med hela datauppsättningen för att identifiera eventuella atypiska avvikelser i datan.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Date`
  - `Product`
  - `Category`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Date, Product, Category, Sales
2024-01-01, Apple, Fruit, 100
2024-01-15, Banana, Fruit, 80
2024-01-30, Carrot, Vegetable, 50
2024-02-05, Apple, Fruit, 100
2024-02-20, Banana, Fruit, 90
2024-02-28, Carrot, Vegetable, 60
```

```
2024-03-10, Apple, Fruit, 110
2024-03-22, Banana, Fruit, 80
2024-03-31, Carrot, Vegetable, 50
2024-04-10, Apple, Fruit, 125
2024-04-22, Banana, Fruit, 60
2024-04-31, Carrot, Vegetable, 80
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Category`
- `Product`

Skapa följande mått:

- `=Mode(Sales)`, för att beräkna det vanligast förekommande värdet för varje produkt.
- `=Mode(TOTAL Sales)`, för att beräkna det vanligast förekommande värdet för hela datauppsättningen.
- `If(Mode(Sales) = Mode(TOTAL Sales), 'Typical', 'Atypical')`, för att beräkna om det vanligast förekommande värdet för varje produkt är typiskt jämfört med resultaten i hela datauppsättningen.

Results table

Category	Product	Mode (Sales)	Mode(TOTAL Sales)	If(Mode(Sales) = Mode(TOTAL Sales), 'Typical', 'Atypical')
Fruit	Apple	100	80	Atypical
Fruit	Banana	80	80	Typical
Vegetable	Carrot	50	80	Atypical

Resultatet av funktionen `Mode` ger det vanligaste försäljningsvärdet per produkt, `Mode(Sales)`, och för alla produkter, `Mode(TOTAL Sales)`. Dessa två värden jämförs sedan för att avgöra vilka försäljningsresultat som är typiska eller atypiska.

Se även:

p *Avg - diagramfunktion (page 469)*
p *Median - diagramfunktion (page 507)*

Only

Only() returnerar ett värde om det finns ett, och endast ett, möjligt resultat från aggregerade data. Om poster endast innehåller ett värde returneras detta värde. Annars returneras NULL. Använd **group by**-satsen för att utvärdera över flera poster. Funktionen **Only()** kan returnera numeriska värden och textvärden.

Syntax:

```
Only ( expr )
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår applikation för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');
Only:

LOAD Customer, Only(CustomerID) as MyUniqIDCheck Resident Temp Group By Customer;
```

Resultattabell

Kund	MyUniqIDCheck
Astrida	1 eftersom enbart kund Astrida har fullständiga register som omfattar CustomerID.

Only - diagramfunktion

Only() returnerar ett värde om det finns ett, och endast ett, möjligt resultat från aggregerade data. Till exempel returnerar en sökning efter den enda produkten med styckpriset =9 NULL om fler än en produkt har styckpriset 9.

Syntax:

```
Only([{SetExpression}] [TOTAL [<fld {,fld}>]] expr)
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Använd Only() när du vill ha ett NULL-resultat om det finns flera möjliga värden i exempeldata.

Exempel: diagramuttryck

Exempel	Resultat
`Only(Category)`	Visar `Category` så länge det är unikt i sammanhanget.

Exempel – endast grunderna

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Customer`
  - `Product`
  - `UnitSales`
  - `UnitPrice`

Laddningsskript

```
Example:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
```

```
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell med följande mått:

- `=Only({<UnitPrice={9}>} Product)`, för att beräkna den enda `Product` som har ett `UnitPrice` på `9`.
- `=Only({<Product={DD}>} Customer)`, för att beräkna den enda `Customer` som säljer en `Product` som heter `DD`.
- `=Only({<UnitPrice={20}>} UnitSales)`, för att beräkna antalet `UnitSales` där `UnitPrice` är `20`.
- `=Only({<UnitPrice={15}>} UnitSales)`, för att beräkna värdena på `UnitSales` där `UnitPrice = 15`.

Results table

Only({<UnitPrice={9}>} Product)	Only({<Product={DD}>} Customer)	Only({<UnitPrice={20}>} UnitSales)	Only({<UnitPrice={15}>} UnitSales)
BB	Betacab	2	-

I detta exempel ska du observera följande:

- Den första kolumnen returnerar värdet `BB` eftersom detta är den enda `Product` som har ett `UnitPrice` på `9`.
- `Betacab`, eftersom det är den enda `Customer` som säljer en `Product` som heter `DD`.
- Den tredje kolumnen returnerar `2` eftersom detta är det enda värdet för `UnitSales` där `UnitPrice = 20`.
- Den fjärde kolumnen returnerar `NULL` eftersom det inte finns något enda resultat där `UnitPrice = 15`. I denna datauppsättning finns det två värden för `UnitSales` där `UnitPrice =15`.

Exempel – enda scenario

Diagramuttryck

Översikt

En datauppsättning innehåller produktförsäljning per period. Företaget vill fastställa om en produkt säljs i en enda butik eller i flera butiker.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `Date`
    - `RecordID`
    - `StoreProduct`
    - `Sales`

Laddningsskript

```
Example:
Load * inline [
Date, RecordID, StoreProduct, Sales, Store
2024-01-01, 1, Apple, 100, A
2024-01-15, 2, Banana, 80, B
2024-01-30, 3, Carrot, 50, C
2024-02-05, 4, Apple, 120, A
2024-02-20, 5, Banana, 90, B
2024-02-28, 6, Carrot, 60, D
2024-03-10, 7, Apple, 110, A
2024-03-22, 8, Banana, 85, B
2024-03-31, 9, Carrot, 55, C
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `StoreProduct`

Skapa följande mått:

- `=Only(Store)`, för att beräkna den unika butik där produkten säljs.
- `=If(IsNull(Only(Store)), 'Multiple Stores', 'Single Store')`, för att beräkna om produkten säljs i en enda butik eller i flera butiker.
- `=Sum(Sales)`, för att beräkna den totala produktförsäljningen.

Results table

StoreProduct	Only (Store)	If(IsNull(Only(Store)), 'Multiple Stores', 'Single Store')	Sum (Sales)
Apple	A	Single Store	330
Banana	B	Single Store	255
Carrot	-	Multiple Stores	165

Resultatet visar att `Apples` och `Bananas` endast säljs i en enda butik medan `Carrots` säljs i flera butiker.

Exempel – avancerat scenario för att skapa en tabell med poster för en enskild månad

Diagramuttryck

Översikt

I detta exempel används funktionen `Only` för att visa en tabell med poster rad för rad för en enskild månad. För att uppnå detta används set-analys för att ringa in önskad uppsättning av poster. Uppsättningsanalys kräver minst en åtgärd, men eftersom vi faktiskt inte behöver aggregera postuppsättningen kan vi effektivt använda aggregeringsfunktionen `Only` för att hysa uppsättningsanalysens omfattning.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- En tidigare load-sats som härleder fältet `MonthYear` från fältet `Date` genom att trunkera dagen i månaden, använda funktionen `MonthStart` och returnera en datumrepresentation av den 1:a i varje månad. Detta resultat levereras till funktionen `Date` för att ställa in visningsformatet till `MMM YY`, t.ex. `Jan 24`.
- Följande fält i datatabellen:
  - `Date`
  - `RecordID`
  - `StoreProduct`
  - `Sales`
  - `Store`

Laddningsskript

```
Example:
Load
*,
date(MonthStart (Date),'MMM YY') AS MonthYear
inline [
Date, RecordID, StoreProduct, Sales, Store
2024-01-01, 1, Apple, 100, A
2024-01-15, 2, Banana, 80, B
2024-01-30, 3, Carrot, 50, C
2024-02-05, 4, Apple, 120, A
2024-02-20, 5, Banana, 90, B
2024-02-28, 6, Carrot, 60, D
2024-03-10, 7, Apple, 110, A
2024-03-22, 8, Banana, 85, B
2024-03-31, 9, Carrot, 55, C
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- MonthYear
- RecordID
- Date
- StoreProduct
- Store

Skapa följande mått:

- =Only({$<MonthYear={'$(=MaxString(MonthYear))'}>} Sales), för att utvärdera värdet Sales och endast returnera dem där MonthYear är det senaste (mest aktuella) MonthYear som finns tillgänglig, baserat på de aktuella valen i tabellen. Dollarteckenexpansionen ($) utför en in-line-beräkning med hjälp av funktionen MaxString, $(=MaxString(MonthYear)), som returnerar textrepresentationen av det senaste MonthYear-värdet.

Att försöka filtrera MonthYear baserat på den numeriska representationen, t.ex. genom att använda funktionen Max , Max(MonthYear), skulle inte ge något resultat. Set-analys tillämpar filterval som endast baseras på textrepresentationen av en dubbel datatyp; en datatyp med en numerisk representation och en textrepresentation, t.ex. Date.

Results table

MonthYear	RecordID	Date	StoreProduct	Only({$<MonthYear={'$(=MaxString (MonthYear))'}>} Sales)
Mar 24	7	2024-03-10	Apple	110
Mar 24	8	2024-03-22	Banana	85
Mar 24	9	2024-03-31	Carrot	55

Utdata från funktionen Only genererar en tabell med poster som enligt standard presenterar försäljningsdata från det senaste, eller mest aktuella, MonthYear som finns tillgängligt i uppgifterna. Om du väljer en tidigare månad i kolumnen MonthYear, t.ex. Feb 24, visas det senaste försäljningsvärdet för den månaden.

Sum

Sum() beräknar summan av värden aggregerade i uttrycket, som definieras av en **group by**-sats.

Syntax:

```
sum ( [ distinct] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
distinct	Om predikatet **distinct** förekommer framför uttrycket kommer alla dubbletter att ignoreras.
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår applikation för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');
Sum:

LOAD Customer, Sum(UnitSales) as MySum Resident Temp Group By Customer;
```

Resultattabell

Kund	MySum
Astrida	39
Betacab	9
Canutility	8

Sum - diagramfunktion

Sum() beräknar summan av de värden som ges av uttrycket eller fältet över aggregerade data.

Syntax:

```
Sum([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr)
```

Returnerad datatyp: numeriska

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten. *Även om DISTINCT-kvalificeraren stöds bör du vara mycket försiktig med att använda den eftersom den kan ge läsaren en felaktig bild av att ett totalt värde visas om vissa data har utelämnats.*
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel: diagramuttryck

Exempel	Resultat
`Sum(UnitSales)`	Returnerar summan av värdena i `UnitSales`.
`Sum(UnitSales*UnitPrice)`	Returnerar summan av `UnitPrice` multiplicerat med `UnitSales` aggregerat.
`Sum (TOTAL UnitSales*UnitPrice)`	Returnerar summan för alla rader i tabellen samt totalen, eftersom kvalificeraren `TOTAL` innebär att summan beräknas oavsett diagramdimensioner.
`Sum({1} TOTAL UnitSales*UnitPrice)`	Det fasta analysuttrycket `{1}` definierar den uppsättning poster som ska utvärderas som `ALL`, oavsett vilket urval som har gjorts.

Exempel – grunderna i Sum

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Customer`
  - `Product`
  - `UnitSales`
  - `UnitPrice`

Laddningsskript

```
Example:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Customer`
- `Product`

Skapa följande mått:

- `=Sum(UnitSales)`, för att beräkna summan av `UnitSales`.
- `=Sum(UnitSales*UnitPrice)`, för att beräkna den aggregerade totalen av `UnitSales` multiplicerat med `UnitPrice`.
- `=Sum(TOTAL UnitSales*UnitPrice)`, för att beräkna summan för alla rader i tabellen samt totalen, eftersom kvalificeraren `TOTAL` innebär att summan är densamma för alla celler, oavsett diagramdimensioner.

- `=Sum({1} TOTAL UnitSales*UnitPrice)`, för att beräkna summan, oberoende av urval, eftersom setanalysuttrycket `{1}` definierar den uppsättning poster som ska utvärderas som `ALL`, oavsett vilket urval som har gjorts.

Results table

Customer	Product	Sum (UnitSales)	Sum (UnitSales*UnitPrice)	Sum(TOTAL UnitSales*UnitPrice)	Sum({1} TOTAL UnitSales*UnitPrice)
Totals	**-**	**38**	**505**	**505**	**505**
Astrida	AA	14	214	505	505
Astrida	BB	9	81	505	505
Betacab	BB	5	50	505	505
Betacab	CC	2	40	505	505
Betacab	DD	0	0	505	505
Canutility	AA	8	120	505	505
Canutility	CC	0	0	505	505

I detta exempel ska du observera följande:

- För det första mätvärdet, `Sum(UnitSales`, returnerar cellen `Totals38` , som är summan av värdena i `UnitSales` för alla kunder och produkter. Resultatet för varje kund- och produktkombination sammanställs för att ge den totala summan av `UnitSales` för varje kombination. Till exempel är summan av alla `UnitSales` för produkten `AA` från kunden `Astrida14` .
- För det andra mätvärdet, `Sum(UnitSales*UnitPrice)`, returnerar cellen `Totals505` . Detta är den sammanlagda summan av `UnitSales` multiplicerat med `UnitPrice` för varje kund- och produktkombination.
- För det tredje mätvärdet, `Sum(TOTAL UnitSales*UnitPrice)`, är det värde som returneras i alla rader inklusive `Totals505`, eftersom kvalificeraren `TOTAL` bortser från diagramdimensioner. Men om ett urval görs på en dimension som `Customer`, kommer värdena att förändras. Om du t.ex. väljer `Betacab` kommer den resulterande tabellen bara att ha tre rader, en för var och en av `Betacab`-posterna, men kolumnen kommer att returnera samma totalvärde för alla rader (`90`).
- Det fjärde mätvärdet, `Sum({1} TOTAL UnitSales*UnitPrice)`, returnerar värdet `505` fristående från eventuella val eftersom setanalysuttrycket `{1}` definierar den uppsättning poster som ska utvärderas som `ALL`. Om du väljer en kund, t.ex. `Canutility`, är det värde som returneras fortfarande `505`.

Exempel – Summascenario med aggregering av försäljningsdata

Diagramuttryck

Översikt

En datauppsättning innehåller försäljningssiffror för butiker efter region. Företaget vill aggregera försäljningen.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Month`
  - `SalesProduct`
  - `Region`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Month,SalesProduct, Region, Sales
2024-01-01, T-shirt, East, 100
2024-01-01, T-shirt, West, 120
2024-01-01, Jeans, East, 200
2024-01-01, Jeans, West, 180
2024-02-01, T-shirt, East, 150
2024-02-01, T-shirt, West, 130
2024-02-01, Jeans, East, 250
2024-02-01, Jeans, West, 220
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till dessa fält som raddimensioner:

- `Month`
- `Region`

Lägg till detta fält som en kolumndimension:

- `SalesProduct`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna den totala försäljningen för varje produkt.

Om du vill expandera pivottabellen så att alla rader visas väljer du **Presentation** i egenskapspanelen under **Utseende**. Under **Rader** väljer du **Helt expanderad**.

Results table

Month Region	SalesProduct	
	Jeans	T-shirt
2024-01-01	380	220
East	200	100
West	180	120
2024-02-01	470	280
East	250	150
West	220	130

Resultatet visar försäljningen för varje `SalesProduct` per `Month` och `Region`, med en aggregerad total för varje `Month`.

Räkneaggregeringsfunktioner

Räkneaggregeringsfunktioner returnerar olika slags beräkningar av ett uttryck över ett antal poster i ett dataladdningsskript eller ett antal värden i en dimension i ett diagram.

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Räkneaggregeringsfunktioner i dataladdningsskriptet

Count

Count() returnerar summan av-värden aggregerade i uttrycket, som definieras av en **group by**-sats.

```
Count ([distinct ] expression | * )
```

MissingCount

MissingCount() returnerar summan av-saknade värden aggregerade i uttrycket, som definieras av en **group by**-sats.

```
MissingCount ([ distinct ] expression)
```

NullCount

NullCount() returnerar summan av NULL-värden aggregerade i uttrycket, som definieras av en **group by**-sats.

```
NullCount ([ distinct ] expression)
```

NumericCount

NumericCount() returnerar summan av-numeriska värden aggregerade i uttrycket, som definieras av en **group by**-sats.

```
NumericCount ([ distinct ] expression)
```

TextCount

TextCount() returnerar summan av fältvärden som är icke-numeriska aggregerade i uttrycket, som definieras av en **group by**-sats.

```
TextCount ([ distinct ] expression)
```

Räkneaggregeringsfunktioner i diagramuttryck

Följande räkneaggregeringsfunktioner kan användas i diagram.

Count

Count() används för att aggregera antalet värden, textvärden och numeriska värden, i varje diagramdimension.

```
Count - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]}
expr)
```

MissingCount

MissingCount() används för att aggregera antalet saknade värden i varje diagramdimension. Saknade värden är alltid icke-numeriska.

```
MissingCount - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]] expr)
```

NullCount

NullCount() används för att aggregera antalet null-värden i varje diagramdimension.

```
NullCount - diagramfunktion({[SetExpression][DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

NumericCount

NumericCount() aggregerar antalet numeriska värden i varje diagramdimension.

```
NumericCount - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

TextCount

TextCount() används för att aggregera antalet fältvärden som är icke-numeriska i varje diagramdimension.

```
TextCount - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

Count

Count() returnerar summan av-värden aggregerade i uttrycket, som definieras av en **group by**-sats.

Syntax:

```
Count( [distinct ] expr)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket ignoreras alla dubbletter.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår applikation för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Skriptexempel

Exempel	Resultat

`Temp:`
`LOAD * inline [`
`Customer|Product|OrderNumber|UnitSales|UnitPrice`
`Astrida|AA|1|4|16`
`Astrida|AA|7|10|15`
`Astrida|BB|4|9|9`
`Betacab|CC|6|5|10`
`Betacab|AA|5|2|20`
`Betacab|BB|1|25| 25`
`Canutility|AA|3|8|15`
`Canutility|CC|||19`
`Divadip|CC|2|4|16`
`Divadip|DD|3|1|25`
`] (delimiter is '|');`
 	
`Count1:`
 	
`LOAD Customer,Count(OrderNumber) as OrdersByCustomer Resident Temp Group By Customer;`
	`Customer OrdersByCustomer`
	
	`Astrida 3`
	
	`Betacab 3`
	
	`Canutility 2`
	
	`Divadip 2`
	Så länge dimensionen Customer tas med i tabellen på arket, annars blir resultatet för OrdersByCustomer 3, 2.

Givet att **Temp**-tabellen är laddad som i föregående exempel:
 	
`LOAD Count(OrderNumber) as TotalOrderNumber Resident Temp;`
	`TotalOrderNumber`
	
	`10`

Exempel	Resultat
Givet att **Temp**-tabellen är laddad som i det första exemplet: `LOAD Count(distinct OrderNumber) as TotalOrderNumber Resident Temp;`	`TotalOrderNumber` 8 Eftersom det finns två värden för OrderNumber med samma värde, 1, och ett nollvärde.

Count - diagramfunktion

Count() används för att aggregera antalet värden, textvärden och numeriska värden, i varje diagramdimension.

Syntax:

```
Count({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel: diagramuttryck

Exempel	Resultat
`Count(OrderNumber)`	Returnerar antalet rader som kan ha ett värde för `OrderNumber`. Alla poster, även tomma, räknas. *"0" räknas som ett värde och inte som en tom cell. Om ett värde aggregeras till 0 för en dimension kommer den dimensionen dock inte att tas med i diagram.*

Exempel	Resultat
`Count(Customer)`	Utvärderar antalet förekomster i alla rader.
`Count(DISTINCT [Customer])`	Kvalificeraren `DISTINCT` begränsar `Count` till att endast utvärdera unika förekomster.
`Count (OrderNumber)/Count ({1} TOTAL OrderNumber)`	Uttrycket returnerar antal order från den valda kunden i form av en procentsats av order från alla kunder.
`Count(TOTAL <Product> OrderNumber)`	Returnerar antalet beställningar som gjorts på produkter endast för de valda kunderna. Tomma celler räknas också.

Exempel – grunderna i Count

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Customer`
  - `Product`
  - `OrderNumber`
  - `UnitSales`
  - `UnitPrice`

Laddningsskript

```
Example:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB|1|25| 25
Canutility|AA|3|8|15
Canutility|CC|||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
](delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=Count(OrderNumber)`, för att räkna antalet beställningar.
- `=Count(Customer)`, för att beräkna antalet kundposter.
- `=Count(DISTINCT Customer)`, för att räkna antalet unika kunder.
- `=Count(OrderNumber)/Count({1} TOTAL OrderNumber)`, för att räkna förhållandet mellan de valda kundorderna i förhållande till de totala kundorderna.
- `=Count(TOTAL <Product> OrderNumber)` för att räkna antalet order lagda för produkter för enbart de valda kunderna (tomma celler räknas).

Results table

Customer	Count (OrderNumber)	Count (Customer)	Count (DISTINCT Customer)	Count (OrderNumber)/Count({1} TOTAL OrderNumber)	Count(TOTAL <Product> OrderNumber)
Totals	**10**	**10**	**4**	**1**	**10**
Astrida	3	3	1	0.3	10
Betacab	3	3	1	0.3	10
Canutility	2	2	1	0.2	10
Divadip	2	2	1	0.2	10

När vi tittar på resultaten kan vi konstatera följande:

- Första mätvärdet: det finns `10` orderposter i datauppsättningen.
- Andra mätvärdet: det finns `10` kundposter i datauppsättningen.
- Tredje mätvärdet: det finns `4` distinkta (unika) kunder i datauppsättningen.
- Fjärde mätvärdet: returnerar `1` eftersom inget urval har tillämpats. Om du väljer kunden `Canutility` ändras värdet till `0.2` eftersom det finns `2` orderposter för den här kunden av totalt `10` poster i datauppsättningen.
- Femte mätvärdet: returnerar `10` eftersom inget urval har tillämpats. Om du väljer ut kunderna `Astrida` och `Canutility`, ändras värdet till `5` eftersom det är antalet order lagda för produkter för enbart de valda kunderna (tomma celler räknas).

Exempel – Använda Antal för att ta reda på antalet försäljningar per produkt

Diagramuttryck

Översikt

En datauppsättning innehåller data för butiksförsäljning. Du vill ta reda på antalet försäljningar per produkt.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Product`
  - `Sales`
  - `Region`
  - `Status`

Laddningsskript

```
Example:
Load * inline [
Product,Sales,Region,Status
Apple, 100, North, Completed
Banana, 80, South, Pending
Carrot, 50, East, Completed
Apple, 120, North, Completed
Banana, 90, South, Completed
Carrot, 60, West, Pending
Apple, 110, North, Completed
Banana, 85, South, Completed
Carrot, 55, East, Pending
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Product`

Skapa följande mått:

- `=Count(Sales)`, för att beräkna antalet försäljningar per produkt.
- `=Count({<Status={'Completed'}>} Sales)`, för att beräkna antalet försäljningar med slutförda order.
- `=Count({<Status={'Pending'}>} Sales)`, för att beräkna antalet försäljningar med väntande order.
- `=Count({<Region={'North'}>} Sales)`, för att beräkna antalet order i region `North`.

Results table

Product	Count (Sales)	Count({<Status={'Completed'}>} Sales)	Count({<Status={'Pending'}>} Sales)	Count({<Region={'North'}>} Sales)
Totals	**9**	**6**	**3**	**3**
Apple	3	3	0	3
Banana	3	2	1	0
Carrot	3	1	2	0

Utdata från funktionen `Count` returnerar en uppdelning av det totala antalet försäljningsorder per produkt, samt antalet order som har slutförts och antalet väntande. Det sista mätvärdet returnerar ett antal försäljningsposter för region `North`.

MissingCount

MissingCount() returnerar summan av-saknade värden aggregerade i uttrycket, som definieras av en **group by**-sats.

Syntax:

```
MissingCount ( [ distinct ] expr)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket ignoreras alla dubbletter.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår applikation för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Skriptexempel

Exempel	Resultat
Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|UnitPrice Astrida|AA|1|4|16 Astrida|AA|7|10|15 Astrida|BB|4|9|9 Betacab|CC|6|5|10 Betacab|AA|5|2|20 Betacab|BB||| 25 Canutility|AA|||15 Canutility|CC| ||19 Divadip|CC|2|4|16 Divadip|DD|3|1|25] (delimiter is '|'); MissCount1: LOAD Customer,MissingCount(OrderNumber) as MissingOrdersByCustomer Resident Temp Group By Customer; Load MissingCount(OrderNumber) as TotalMissingCount Resident Temp;	Customer MissingOrdersByCustomer Astrida 0 Betacab 1 Canutility 2 Divadip 0 Den andra satsen ger: TotalMissingCount 3 i en tabell med den dimensionen.
Givet att **Temp**-tabellen är laddad som i föregående exempel: LOAD MissingCount(distinct OrderNumber) as TotalMissingCountDistinct Resident Temp;	TotalMissingCountDistinct 1 Eftersom det finns enbart ett saknat OrderNumber-värde.

MissingCount - diagramfunktion

MissingCount() används för att aggregera antalet saknade värden i varje diagramdimension. Saknade värden är alltid icke-numeriska.

Syntax:

```
MissingCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.

Argument	Beskrivning
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel: diagramuttryck

Exempel	Resultat
`MissingCount(OrderNumber)`	Returnerar antalet saknade värden där fälten `OrderNumber` är tomma. *"0" räknas som ett värde och inte som en tom cell. Om ett värde aggregeras till 0 för en dimension kommer den dimensionen dock inte att tas med i diagram.*
`MissingCount (OrderNumber)/MissingCount ({1} TOTAL OrderNumber)`	Uttrycket returnerar antalet ofullständiga order från den valda kunden i form av en decimaldel av ofullständiga order från alla kunder.

Exempel – grunderna i MissingCount

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Customer`
  - `Product`
  - `OrderNumber`
  - `UnitSales`
  - `UnitPrice`

Laddningsskript

```
Example:
LOAD * inline [
Customer, Product, OrderNumber, UnitSales, UnitPrice
Astrida,AA, 1,4,16
Astrida, AA,7,10,15
Astrida, BB,4,9,9
```

```
Betacab, BB,6,5,10
Betacab, CC,5,2,20
Betacab ,DD,,,25
Canutility, AA,,,15
Canutility, CC,,,19
Divadip, AA,2,4,16
Divadip, BB,,4,16
Divadip, DD,3,,25
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=MissingCount(OrderNumber)`, för att beräkna antalet order som saknar ett `OrderNumber`.
- `=MissingCount(OrderNumber)/MissingCount ({1} Total [OrderNumber])`, för att returnera antalet ofullständiga order från den valda kunden som en bråkdel av ofullständiga order från alla kunder. Om du vill visa värdet i procent väljer du **Nummer > Enkel formatering > 12,34%** under **Nummerformatering**.

Results table

Customer	MissingCount (OrderNumber)	MissingCount(OrderNumber)/MissingCount ({1} Total [OrderNumber])
Totals	**4**	**100.00%**
Astrida	0	0.00%
Betacab	1	25.00%
Canutility	2	50.00%
Divadip	1	25.00%

I det första mätvärdet returnerar funktionen `MissingCount4` eftersom `4` av `11` posterna i `OrderNumber`-fältet är tomma.

> *"0" räknas som ett värde och inte som en tom cell. Om ett värde aggregeras till 0 för en dimension kommer den dimensionen dock inte att tas med i diagram.*

I det andra mätvärdet returnerar utdata från funktionen `MissingCount` antalet ofullständiga order från den valda kunden i form av en procentdel av ofullständiga order från alla kunder. Om du t.ex. väljer `Canutility` från dimensionen `Customer` returnerar åtgärden `50%` eftersom hälften av alla ofullständiga order (`2` av `4`) är hos den här kunden.

Exempel – Använd MissingCount för att identifiera saknade produktdata

Diagramuttryck

Översikt

En datauppsättning innehåller månatliga uppgifter om viktiga produktindikatorer (KPI). En produktchef vill ta reda på vilka KPI:er som saknar data.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Month`
  - `KPI`
  - `Value`

Laddningsskript

```
Example:
Load * inline [
Month, KPI, Value
2024-01-01, Revenue, 5000
2024-02-01, Revenue,
2024-03-01, Revenue, 5500
2024-04-01, Revenue,
2024-01-01, Expenses, 3000
2024-02-01, Expenses, 3100
2024-03-01, Expenses,
2024-04-01, Expenses, 3200
2024-01-01,Customer Count, 200
2024-02-01,Customer Count, 210
2024-03-01,Customer Count,
2024-04-01,Customer Count,
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `KPI`

Skapa följande mått:

- `=MissingCount(Value)`, för att beräkna antalet saknade värden.

Results table

KPI	MissingCount(Value)
Totals	**5**
Customer Count	2
Expenses	1
Revenue	2

Resultatet visar att 5 poster i datauppsättningen saknar data samt de KPI-kategorier som dessa poster avser.

NullCount

NullCount() returnerar summan av NULL-värden aggregerade i uttrycket, som definieras av en **group by**-sats.

Syntax:

```
NullCount ( [ distinct ] expr)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket ignoreras alla dubbletter.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår applikation för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Skriptexempel

Exempel	Resultat
Set NULLINTERPRET = NULL; Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|CustomerID Astrida|AA|1|10|1 Astrida|AA|7|18|1 Astrida|BB|4|9|1 Astrida|CC|6|2|1 Betacab|AA|5|4|2 Betacab|BB|2|5|2 Betacab|DD||| Canutility|AA|3|8| Canutility|CC|NULL||] (delimiter is '|'); Set NULLINTERPRET=; NullCount1: LOAD Customer,NullCount(OrderNumber) as NullOrdersByCustomer Resident Temp Group By Customer; LOAD NullCount(OrderNumber) as TotalNullCount Resident Temp;	Customer NullOrdersByCustomer Astrida 0 Betacab 0 Canutility 1 Den andra satsen ger: TotalNullCount 1 i en tabell med den dimensionen, eftersom endast en post innehåller ett null-värde.

NullCount - diagramfunktion

NullCount() används för att aggregera antalet null-värden i varje diagramdimension.

Syntax:

```
NullCount({[SetExpression][DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.

Argument	Beskrivning
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel: diagramuttryck

Exempel	Resultat
`NullCount(OrderNumber)`	Returnerar antalet null-värden i fältet `OrderNumber`.
`NullCount (OrderNumber)/NullCount ({1} Total OrderNumber)`	Uttrycket returnerar antalet order med `NULL` när `OrderNumber` från den valda kunden som en bråkdel av samtliga order från kunder med `NULL` som `OrderNumber`.

Exempel – grunderna i NullCount

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Värdehanteringsvariabeln `NullInterpret` som är inställd på `NULL`. Variabeln kommer att tolka alla poster med texten `NULL` som nollvärden.
- Följande fält i datatabellen:
  - `Customer`
  - `Product`
  - `OrderNumber`

Laddningsskript

```
Set NULLINTERPRET = NULL;
Example:
LOAD * inline [
Customer|Product|OrderNumber
Astrida|AA|1
Astrida|AA|7
Astrida|BB|4
Astrida|CC|6
Betacab|AA|5
Betacab|BB|2
Betacab|DD|NULL
```

```
Canutility|AA|3
Canutility|AA|NULL
Canutility|CC|NULL
] (delimiter is '|');
```

För att återställa variabeln `NullInterpret` till standardinställningen, inkludera följande variabelsatser i slutet av laddningsskriptet: `Set NULLINTERPRET =;`

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=NullCount(OrderNumber)`, för att beräkna antalet order som har `NULL` som `OrderNumber`.
- `=NullCount (OrderNumber)/NullCount ({1} Total OrderNumber)`, för att returnera antalet order (från den valda kunden) med `NULL` som ett `OrderNumber` som en bråkdel av samtliga order från kunder med `NULL` som `OrderNumber`. Om du vill visa värdet i procent väljer du **Nummer > Enkel formatering > 12,34%** under **Nummerformatering**.

Results table

Customer	NullCount (OrderNumber)	NullCount (OrderNumber)/NullCount ({1} Total OrderNumber)
Totals	**3**	**100.00%**
Astrida	0	0.00%
Betacab	1	33.33%
Canutility	2	66.67%

Det första mätvärdet returnerar `3` eftersom det finns `3` totalt `OrderNumber`-poster med ett `NULL`-värde (`1` post med kunden `Betacab` och `2` poster med `Canutility`).

Det andra mätvärdet returnerar antalet `NULL`-order som en bråkdel av `NULL`-order från alla kunder. Till exempel tillskrivs `66.67%` av alla ordernummer med värdet `NULL` till kunden `Canutility`.

Exempel – Använda NullCount för att identifiera nullvärden i KPI-data

Diagramuttryck

Översikt

En datauppsättning innehåller månatliga uppgifter om viktiga produktindikatorer (KPI). En produktchef vill ta reda på vilka KPI:er som har nullvärden i sina data.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Värdehanteringsvariabeln `NullInterpret` som är inställd på `NULL`. Variabeln kommer att tolka alla poster med texten `NULL` som nollvärden.
- Följande fält i datatabellen:
  - `Month`
  - `KPI`
  - `Value`

Laddningsskript

```
Set NULLINTERPRET = NULL;
Example:
LOAD * inline [
Month, KPI, Value
2024-01-01, Revenue, 5000
2024-02-01, Revenue, NULL
2024-03-01, Revenue, 5500
2024-04-01, Revenue, NULL
2024-01-01, Expenses, 3000
2024-02-01, Expenses, 3100
2024-03-01, Expenses, NULL
2024-04-01, Expenses, 3200
2024-01-01,Customer Count, 200
2024-02-01,Customer Count, 210
2024-03-01,Customer Count, NULL
2024-04-01,Customer Count, NULL
];
```

För att återställa variabeln `NullInterpret` till standardinställningen, inkludera följande variabelsatser i slutet av laddningsskriptet: `Set NULLINTERPRET =;`

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en raddimension:

- `KPI`

Skapa följande mått:

- `=NullCount(Value)`, för att beräkna antalet saknade värden.

Results table

KPI	NullCount(Value)
Customer Count	2
Expenses	1
Revenue	2

Resultatet visar att 5 poster i datauppsättningen har NULL-värden och de KPI-kategorier som dessa poster avser.

NumericCount

NumericCount() returnerar summan av-numeriska värden aggregerade i uttrycket, som definieras av en **group by**-sats.

Syntax:

```
NumericCount ( [ distinct ] expr)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket ignoreras alla dubbletter.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår applikation för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Skriptexempel

Exempel	Resultat
`LOAD NumericCount(OrderNumber) as TotalNumericCount Resident Temp;`	Den andra satsen ger: TotalNumericCount 7 i en tabell med den dimensionen.
Givet att **Temp**-tabellen är laddad som i föregående exempel: `LOAD NumericCount(distinct OrderNumber) as TotalNumericCountDistinct Resident Temp;`	TotalNumericCountDistinct 6 Eftersom det finns ett OrderNumber som duplicerar ett annat, innebär det att resultatet är 6 som inte är dubbletter.

Exempel:

```
Temp:

LOAD * inline [

Customer|Product|OrderNumber|UnitSales|UnitPrice
```

```
Astrida|AA|1|4|16

Astrida|AA|7|10|15

Astrida|BB|4|9|9

Betacab|CC|6|5|10

Betacab|AA|5|2|20

Betacab|BB||| 25

Canutility|AA|||15

Canutility|CC| ||19

Divadip|CC|2|4|16

Divadip|DD|7|1|25

] (delimiter is '|');

NumCount1:

LOAD Customer,NumericCount(OrderNumber) as NumericCountByCustomer Resident Temp Group By
Customer;
```

Resultattabell

Kund	NumericCountByCustomer
Astrida	3
Betacab	2
Canutility	0
Divadip	2

NumericCount - diagramfunktion

NumericCount() aggregerar antalet numeriska värden i varje diagramdimension.

Syntax:

```
NumericCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Argument	Beskrivning
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel: diagramuttryck

Exempel	Resultat
`NumericCount (OrderNumber)`	Returnerar antalet rader i fältet `OrderNumber` som har numeriska värden. *"0" räknas som ett värde och inte som en tom cell. Om ett värde aggregeras till 0 för en dimension kommer den dimensionen dock inte att tas med i diagram.Tomma celler utvärderas som icke-numeriska värden och räknas inte av `NumericCount`.*
`NumericCount (Product)`	Returnerar antalet produktnamn som är numeriska värden. Du kan använda detta för att kontrollera att inga textfält har fått numeriskt innehåll.
`NumericCount (DISTINCT OrderNumber)/Count (DISTINCT OrderNumber)`	Räknar det fullständiga antalet distinkta numeriska ordernummer och delar dem med antalet numeriska och icke-numeriska ordernummer. Detta blir 1 om alla fältvärden är numeriska.

Exempel – grunderna i NumericCount

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:

- Customer
- Product
- OrderNumber

Laddningsskript

```
Example:
LOAD * inline [
Customer|Product|OrderNumber
Astrida|AA|1
Astrida|AA|7
Astrida|BB|4
Betacab|CC|6
Betacab|AA|5
Betacab|BB|
Canutility|AA|
Canutility|CC|
Divadip|CC|2
Divadip|DD|3
](delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- OrderNumber

Skapa följande mått:

- =NumericCount(OrderNumber), för att beräkna antalet order som innehåller numeriska värden.
- =NumericCount(Product), för att returnera antalet produktnamn som är helt numeriska värden.
- =NumericCount(DISTINCT OrderNumber)/Count(DISTINCT OrderNumber), för att beräkna det totala antalet ordernummer med distinkta numeriska värden och dividera detta belopp med det totala antalet ordernummer med distinkta numeriska och icke-numeriska värden. Detta blir 1 om alla fältvärden är numeriska.

Results table

OrderNumber	NumericCount (OrderNumber)	NumericCount (Product)	NumericCount (DISTINCT OrderNumber)/Count(DISTINCT OrderNumber)
Totals	**7**	**0**	**0.875**
1	1	0	1
2	1	0	1
3	1	0	1
4	1	0	1

OrderNumber	NumericCount (OrderNumber)	NumericCount (Product)	NumericCount (DISTINCT OrderNumber)/Count(DISTINCT OrderNumber)
Totals	**7**	**0**	**0.875**
5	1	0	1
6	1	0	1
7	1	0	1
	0	0	0

Utdata från det första mätvärdet returnerar `7` eftersom tre av de tio posterna i `OrderNumber` är tomma och de återstående är numeriska.

Det andra mätvärdet returnerar `0` eftersom det inte finns några numeriska värden; alla produktnamn är text. Du kan använda denna typ av mätvärde för att kontrollera att inga textfält har fått numeriskt innehåll.

Det tredje mätvärdet räknar antalet distinkta numeriska ordernummer och delar dem med antalet numeriska och icke-numeriska ordernummer. Detta blir `1` om alla fältvärden är numeriska. Du kan använda detta för att kontrollera att alla fältvärden är numeriska. I exemplet finns det `7` distinkta numeriska värden för `OrderNumber` av `8` distinkta numeriska och icke-numeriska värden, så att uttrycket returnerar `0.875`.

Exempel – NumericCount-scenario

Diagramuttryck

Översikt

En datauppsättning innehåller månatliga försäljningsdata från flera butiker. Butikschefen vill ta reda på vilka butiker och produkter som har försäljningsvärden som inte är numeriska. Detta indikerar vilka försäljningsposter som har ogiltiga värden, till exempel att posten var tom eller innehåller icke-numeriska data.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Store`
  - `StoreProduct`
  - `Sales`

Laddningsskript

```
Example:
LOAD * inline [
Store, StoreProduct, Sales
```

```
A, Widget A, 1200
B, Widget A, No data
C, Widget A, 1300
A, Widget B, 800
B, Widget B,
C, Widget B, "Error"
A, Widget C, 500
B, Widget C, 600
C, Widget C, 700
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en raddimension:

- `Store`

Lägg till detta fält som en kolumndimension:

- `StoreProduct`

Skapa följande mått:

- `=NumericCount(Sales)`, för att beräkna butiksförsäljningen som har numeriska data.

Results table

Store	StoreProduct		
	Widget A	Widget B	Widget C
A	1	1	1
B	0	0	1
C	1	0	1

Resultatet visar att det finns `3` poster där det numeriska antalet är `0`. Det är dessa försäljningsposter som måste undersökas eftersom de innehåller icke-numeriska värden.

TextCount

TextCount() returnerar summan av fältvärden som är icke-numeriska aggregerade i uttrycket, som definieras av en **group by**-sats.

Syntax:

```
TextCount ( [ distinct ] expr)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket ignoreras alla dubbletter.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår applikation för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Exempel:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
TextCount1:

LOAD Customer,TextCount(Product) as ProductTextCount Resident Temp Group By Customer;
```

Resultattabell

Kund	ProductTextCount
Astrida	3
Betacab	3
Canutility	2
Divadip	2

Exempel:

```
LOAD Customer,TextCount(OrderNumber) as OrderNumberTextCount Resident Temp Group By Customer;
```

Resultattabell

Kund	OrderNumberTextCount
Astrida	0
Betacab	1
Canutility	2
Divadip	0

TextCount - diagramfunktion

TextCount() används för att aggregera antalet fältvärden som är icke-numeriska i varje diagramdimension.

Syntax:

```
TextCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel: diagramuttryck

Exempel	Resultat
`TextCount (Product)`	Returnerar antalet produktnamn som har textvärden.

Exempel	Resultat
`TextCount (OrderNumber)`	Returnerar antalet rader i fältet `OrderNumber` som har textvärden. *"0" räknas som ett värde och inte som en tom cell. Om ett värde aggregeras till 0 för en dimension kommer den dimensionen dock inte att tas med i diagram. Tomma celler utvärderas som icke-text och räknas inte av `TextCount`.*
`TextCount (DISTINCT Product)/Count (Product)`	Räknar alla antal distinkta textvärden i `Product` och delar det med det totala antalet värden i `Product`.

Exempel – grunderna i TextCount

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Customer`
  - `Product`
  - `OrderNumber`

Laddningsskript

```
Set NULLINTERPRET = ' ';
Example:
LOAD * inline [
Customer|Product|OrderNumber
Astrida|AA|1
Astrida|AA|7
Astrida|BB|4
Betacab|CC|6
Betacab|AA|5
Betacab|BB|
Canutility|AA|
Canutility|CC|
Divadip|CC|2
Divadip|DD|3
](delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till följande mått:

- `=TextCount(Product)`, för att returnera antalet produktnamn som är textvärden.
- `=TextCount(OrderNumber)`, för att beräkna antalet ordernummer som är textvärden.
- `=TextCount(DISTINCT Product)/Count(Product)`, för att räkna det totala antal distinkta textvärden för `Product` och dela det med det totala antalet värden i `Product`.

Results table

TextCount(Product)	TextCount(OrderNumber)	TextCount(DISTINCT Product)/Count(Product)
10	3	0.4

Det första mätvärdet returnerar `10` eftersom alla tio poster i `Product` är text.

Det andra mätvärdet returnerar `3` eftersom tomma celler behandlas som text och räknas. Den här typen av mätvärde brukar användas för att kontrollera att inga numeriska fält har fått textinnehåll eller är tomma.

Det tredje mätvärdet returnerar `0.4`. Det här resultatet beräknas genom att dela det totala antalet distinkta textvärden för `Product` (`4`) med det totala antalet värden för `Product` (`10`).

Exempel – TextCount-scenario

Diagramuttryck

Översikt

En datauppsättning innehåller responsiva data från undersökningar med både reklamationer och allmän feedback per produkt. Du vill bara granska de responsiva svar som har textvärden.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `CustomerID`
  - `ProductPurchased`
  - `Complaint`
  - `Feedback`

Laddningsskript

```
Set NullInterpret = '';
Example:
Load * Inline [
CustomerID, ProductPurchased, Complaint, Feedback
1, Laptop, Slow performance, "The laptop is slow, poor battery, slow graphics"
```

```
2, Mobile, Heating issue, "Mobile heats up, average camera"
3, Laptop, Battery problem, "Battery not working, slow to charge"
4, Tablet, Screen issue, "Screen is broken, unresponsive touch"
5, Mobile, No issues, "Excellent performance"
6, Mobile, 123, "123"
7, Laptop, ,
8, Tablet, Warranty expired,
9, Mobile, , "Contact customer service"
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ProductPurchased`

Skapa följande mått:

- `=TextCount(ProductPurchased)`, för att beräkna antalet produktnamn med icke-numeriska värden.
- `=TextCount(Complaint)`, för att beräkna antalet reklamationer med icke-numeriska värden.
- `=TextCount(Feedback)`, för att beräkna antalet feedback-värden med icke-numeriska värden.

Results table

ProductPurchased	TextCount (ProductPurchased)	TextCount (Complaint)	TextCount (Feedback)
Laptop	3	2	2
Mobile	4	2	3
Tablet	2	2	1

Resultatet visar att det för produkten `Laptop` finns `3` poster varav `2` har klagomål och `2` har feedback. På samma sätt finns det för produkten `Mobile` `4` poster, inklusive `2` klagomål och `3` feedbacksvar. Numeriska värden eller tomma värden räknas inte eftersom det förväntas att fälten `ProductPurchased`, `Complaint` och `Feedback` är textvärden.

Finansiella aggregeringsfunktioner

Den här delen beskriver aggregeringsfunktioner för finansiella operationer vad gäller betalning och kassaflöde.

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Finansiella aggregeringsfunktioner i dataladdningsskriptet

IRR

IRR() returnerar den aggregerade internräntan för en serie kassaflöden, representerade av talen i uttryck itererade över ett antal poster enligt vad som definierats i en group by-sats.

```
IRR (expression)
```

XIRR

XIRR() returnerar den aggregerade internräntan (per år) för en tablå av kassaflöden (som inte nödvändigtvis är periodisk), representerad av parvisa tal i **pmt** och **date** itererade över ett antal poster enligt vad som definierats i en group by-sats. Alla betalningar diskonteras utifrån ett 365-dagarsår.

```
XIRR (valueexpression, dateexpression )
```

NPV

NPV()-skriptfunktionen tar en rabattsats och multiplicerar värden ordnade efter period. Inflöden (intäkter) är positiva och utflöden (framtida betalningar) antas vara negativa värden för dessa beräkningar. Dessa inträffar i slutet av varje period.

```
NPV (rate, expression)
```

XNPV

XNPV() returnerar det aggregerade aktuella nettovärdet för en tablå av kassaflöden (som inte nödvändigtvis är periodisk), representerade av parvisa tal i **pmt** och **date**. Alla betalningar diskonteras utifrån ett 365-dagarsår.

```
XNPV (rate, valueexpression, dateexpression)
```

Finansiella aggregeringsfunktioner i diagramuttryck

Dessa finansiella aggregeringsfunktioner kan användas i diagram.

IRR

IRR() returnerar den aggregerade avkastningsgraden för en serie kassaflöden som motsvaras av talen i det uttryck som ges av **value** itererat över diagrammets dimensioner.

```
IRR - diagramfunktion[TOTAL [<fld {,fld}>]] value)
```

NPV

NPV() returnerar det aggregerade aktuella nettovärdet för en investering baserat på **discount_rate** per period och en serie framtida betalningar (negativa värden) och inkomster (positiva värden). Dessa motsvaras av talen i **value** itererat över diagrammets dimensioner. Betalningar och inkomster förväntas i slutet av varje period.

```
NPV - diagramfunktion([TOTAL [<fld {,fld}>]] discount_rate, value)
```

XIRR

XIRR() returnerar den aggregerade avkastningsgraden (per år) för ett kassaflödesschema (inte nödvändigtvis periodiskt) motsvarat av parvisa tal i de uttryck som ges av **pmt** och **date** itererat över diagrammets dimensioner. Alla betalningar diskonteras utifrån ett 365-dagarsår.

```
XIRR - diagramfunktion([TOTAL [<fld {,fld}>]] pmt, date)
```

XNPV

XNPV() returnerar det aktuella aggregerade nettovärdet för ett kassaflödesschema (inte nödvändigtvis periodiskt) motsvarat av parvisa tal i de uttryck som ges av **pmt** och **date** itererat över diagramdimensionerna. Alla betalningar diskonteras utifrån ett 365-dagarsår.

```
XNPV - diagramfunktion([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

IRR

IRR() returnerar den aggregerade internräntan för en serie kassaflöden, representerade av talen i uttryck itererade över ett antal poster enligt vad som definierats i en group by-sats.

Dessa kassaflöden måste inte vara jämna vilket krävs för annuiteter. Kassaflödena måste dock infalla med jämna intervall (som månatligen eller årligen). Internränta är den räntesats som ges vid en investering i form av betalning (negativa värden) och inkomst (positiva värden) som infaller regelbundet. För att funktionen ska kunna beräknas krävs minst ett negativt och ett positivt värde.

Funktionen använder en förenklad version av Newton-metoden för att beräkna intern avkastningsgrad (IRR).

Syntax:

```
IRR(value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Det uttryck eller fält som innehåller de data som ska mätas.

Begränsningar:

Textvärden, NULL-värden samt saknade värden ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

Exempel och resultat:

Exempel och resultat

Exempel	År	IRR2013
Cashflow: LOAD 2013 as Year, * inline [Date|Discount|Payments 2013-01-01|0.1|-10000 2013-03-01|0.1|3000 2013-10-30|0.1|4200 2014-02-01|0.2|6800] (delimiter is '|'); Cashflow1: LOAD Year,IRR(Payments) as IRR2013 Resident Cashflow Group By Year;	2013	0.1634

IRR - diagramfunktion

IRR() returnerar den aggregerade avkastningsgraden för en serie kassaflöden som motsvaras av talen i det uttryck som ges av **value** itererat över diagrammets dimensioner.

Dessa kassaflöden måste inte vara jämna vilket krävs för annuiteter. Kassaflödena måste dock infalla med jämna intervall (som månatligen eller årligen). Internränta är den räntesats som ges vid en investering i form av betalning (negativa värden) och inkomst (positiva värden) som infaller regelbundet. För att funktionen ska kunna beräknas krävs minst ett negativt och ett positivt värde.

Funktionen använder en förenklad version av Newton-metoden för att beräkna intern avkastningsgrad (IRR).

Syntax:

```
IRR([TOTAL [<fld {,fld}>]] value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Det uttryck eller fält som innehåller de data som ska mätas.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {,fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden samt saknade värden ignoreras.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`IRR (Payments)`	0.1634 Betalningarna antas vara periodiska till sin natur, exempelvis månatliga. *Datum-fältet används i XIRR-exemplet där betalningarna kan vara icke-periodiska så länge du uppger de datum då betalningarna gjordes.*

Data som används i exempel:

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Discount|Payments
2013-01-01|0.1|-10000
2013-03-01|0.1|3000
2013-10-30|0.1|4200
2014-02-01|0.2|6800
] (delimiter is '|');
```

Se även:

p *XIRR - diagramfunktion (page 449)*
p *Aggr - diagramfunktion (page 612)*

NPV

NPV()-skriptfunktionen tar en rabattsats och multiplicerar värden ordnade efter period. Inflöden (intäkter) är positiva och utflöden (framtida betalningar) antas vara negativa värden för dessa beräkningar. Dessa inträffar i slutet av varje period.

Det aktuella nettovärdet, eller NPV (Net Present Value), används för att beräkna det aktuella totala värdet för en framtida ström av kassaflöden. För att beräkna NPV behöver vi beräkna framtida kassaflöden för varje period och fastställa korrekt rabattsats. **NPV()**-skriptfunktionen tar en rabattsats och multiplicerar värden ordnade efter period. Inflöden (intäkter) är positiva och utflöden (framtida betalningar) antas vara negativa värden för dessa beräkningar. Dessa inträffar i slutet av varje period.

Syntax:

```
NPV(discount_rate, value)
```

Returnerad datatyp: numeriska. Som standard formateras resultatet som valuta.

Formeln för att beräkna det aktuella nettovärdet är:

$$NPV = \sum_{t=1}^{n} \frac{R_t}{(1+i)^t}$$

där:

- R_t = nettokassainflöden och -utflöden under en enstaka period `t`
- `i` = rabattsats eller avkastning som kan tjänas eller erhållas genom alternativa investeringar
- `t` = antal timerperioder

Argument

Argument	Beskrivning
discount_ rate	**discount_rate** är procentandelen för rabatt som tillämpas. Ett värde på 0,1 anger en rabattsats på 10 %.
value	Det här fältet innehåller värden för flera perioder sorterade efter period. Det första värdet antas vara kassaflödet vid slutet av period 1 och så vidare.

Begränsningar:

`NPV()`-funktionen har följande begränsningar:

- Textvärden, NULL-värden samt saknade värden ignoreras.
- Kassaflödesvärden måste sorteras efter stigande period.

Användning

`NPV()` är en finansiell funktion som används för att kontrollera lönsamheten för projekt och för att härleda andra mått. Den här funktionen är användbar när kassaflöden finns tillgängliga som rådata.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – enstaka betalning (skript)

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med ett projekt och dess kassaflöde för en period som laddas till en tabell som heter `CashFlow`.
- En resident load från `CashFlow`-tabellen, som används för att beräkna NPV-fältet för projektet i en tabell som heter `NPV`.
- En hårdkodad rabattsats på 10 % som används i NPV-beräkningen.
- En `Group By`-sats som används för att gruppera alla betalningar för projektet.

Laddningsskript

```
CashFlow:
Load
*
Inline
[
PrjId,PeriodId,Values
1,1,1000
];

NPV:
Load
       PrjId,
       NPV(0.1,Values) as NPV //Discount Rate of 10%
Resident CashFlow
Group By PrjId;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `PrjId`
- `NPV`

Resultattabell

PrjId	NPV
1	$909.09

För en enstaka betalning på 1000 USD som ska erhållas på slutet av en period, till en rabattsats på 10 % per period, är NPV lika med 1000 USD delat med (1 + rabattsatsen). Effektiv NPV motsvarar 909,09 USD.

Exempel 2 – flera betalningar (skript)

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med ett projekt och dess kassaflöde för flera perioder som laddas till en tabell som heter `CashFlow`.
- En resident load från `CashFlow`-tabellen, som används för att beräkna NPV-fältet för projektet i en tabell som heter `NPV`.
- En hårdkodad rabattsats på 10 % (0,1) används i NPV-beräkningen.
- En `Group By`-sats som används för att gruppera alla betalningar för projektet.

Laddningsskript

```
CashFlow:
Load
*
Inline
[
PrjId,PeriodId,Values
1,1,1000
1,2,1000
];

NPV:
Load
       PrjId,
       NPV(0.1,Values) as NPV //Discount Rate of 10%
Resident CashFlow
Group By PrjId;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `PrjId`
- `NPV`

Resultattabell

PrjId	NPV
1	$1735.54

För betalningar på 1000 USD som ska erhållas på slutet av två period, till en rabattsats på 10 % per period, är effektiv NPV lika med 1735,54 USD.

Exempel 3 – flera betalningar (skript)

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Rabattsatser för två projekt som laddas till en tabell som heter `Project`.
- Kassaflöden för flera perioder för varje projekt efter projekt-ID och period-ID. Detta period-ID kan användas för att sortera posterna om uppgifterna inte är sorterade.
- Kombinationen av `NoConcatenate`, resident loads, och `Left Join`-funktionen för att skapa en temporär tabell, `tmpNPV`. Tabellen kombinerar posterna från tabellerna `Project` och `CashFlow` till en platt tabell. Rabattsatserna kommer att upprepas i den här tabellen för varje period.
- En resident load från `tmpNPV`-tabellen, som används för att beräkna NPV-fältet för varje projekt i en tabell som heter `NPV`.
- Den enstaka rabattsatsen som är kopplad till varje projekt. Detta hämtas med `only()`-funktionen och används i NPV-beräkningen för varje projekt.
- En `Group By`-sats som används för att gruppera alla betalningar för varje projekt efter projekt-ID.

För att undvika att syntetiska eller redundanta data laddas till datamodellen släpps `tmpNPV`-tabellen på slutet av skriptet.

Laddningsskript

```
Project:
Load * inline [
PrjId,Discount_Rate
1,0.1
2,0.15
];

CashFlow:
Load
*
Inline
[
PrjId,PeriodId,Values
1,1,1000
1,2,1000
```

```
1,3,1000
2,1,500
2,2,500
2,3,1000
2,4,1000
];

tmpNPV:
NoConcatenate Load *
Resident Project;
Left Join
Load *
Resident CashFlow;

NPV:
Load
       PrjId,
       NPV(Only(Discount_Rate),Values) as NPV //Discount Rate will be 10% for Project 1 and 15% for
Project 2
Resident tmpNPV
Group By PrjId;

Drop table tmpNPV;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- PrjId
- NPV

Resultattabell

PrjId	NPV
1	$2486.85
2	$2042.12

I projekt-ID 1 förväntas betalningar på 1000 USD erhållas på slutet av tre perioder, till en rabattsats på 10 % per period. Därför blir effektiv NPV 2486,85 USD.

I projekt-ID 2 förväntas två betalningar på 500 USD och ytterligare två betalningar på 1000 USD över fyra perioder med en rabattsats på 15 %. Därför blir effektiv NPV 2042,12 USD.

Exempel 4 – exempel på lönsamhet för ett projekt (skript)

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Rabattsatser och initiala investeringar (period 0) för två projekt som laddas till en tabell som heter `Project`.
- Kassaflöden för flera perioder för varje projekt efter projekt-ID och period-ID. Detta period-ID kan användas för att sortera posterna om uppgifterna inte är sorterade.
- Kombinationen av `NoConcatenate`, resident loads, och `Left Join`-funktionen för att skapa en temporär tabell, `tmpNPV`. Tabellen kombinerar posterna från tabellerna `Project` och `CashFlow` till en platt tabell. Rabattsatserna kommer att upprepas i den här tabellen för varje period.
- Den enstaka rabattsatsen som är kopplad till varje projekt, som hämtas med `only()`-funktionen och används i NPV-beräkningen för varje projekt.
- En resident load från tmpNPV-tabellen används för att beräkna NPV-fältet för varje projekt i en tabell som heter NPV.
- Ett extra fält som delar upp NPV efter den inledande investeringen för varje projekt används för att beräkna projektets lönsamhetsindex.
- En gruppsats som grupperar efter projekt-ID används för att gruppera alla betalningar för varje projekt.

För att undvika att syntetiska eller redundanta data laddas till datamodellen släpps `tmpNPV`-tabellen på slutet av skriptet.

Laddningsskript

```
Project:
Load * inline [
PrjId,Discount_Rate, Initial_Investment
1,0.1,100000
2,0.15,100000
];

CashFlow:
Load
*
Inline
[
PrjId,PeriodId,Values,
1,1,35000
1,2,35000
1,3,35000
2,1,30000
2,2,40000
2,3,50000
2,4,60000
];

tmpNPV:
NoConcatenate Load *
Resident Project;
Left Join
Load *
Resident CashFlow;

NPV:
```

```
Load
        PrjId,
        NPV(Only(Discount_Rate),Values) as NPV, //Discount Rate will be 10% for Project 1 and
15% for Project 2
    NPV(Only(Discount_Rate),Values)/ Only(Initial_Investment) as Profitability_Index
Resident tmpNPV
Group By PrjId;

Drop table tmpNPV;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- PrjId
- NPV

Skapa följande mått:

```
=only(Profitability_Index)
```

Resultattabell

PrjId	NPV	=endast(Lönsamhetsindex)
1	$87039.82	0.87
2	$123513.71	1.24

Projekt-ID 1 har ett effektivt NPV på 87 039,82 USD och en initial investering på 100 000 USD. Därför är lönsamhetsindexet lika med 0,87. Eftersom det är mindre än 1 är projektet inte lönsamt.

Projekt-ID 2 har ett effektivt NPV på 123 513,71 USD och en initial investering på 100 000 USD. Därför är lönsamhetsindexet lika med 1.24. Eftersom det är större än 1 är projektet lönsamt.

NPV - diagramfunktion

NPV() returnerar det aggregerade aktuella nettovärdet för en investering baserat på **discount_rate** per period och en serie framtida betalningar (negativa värden) och inkomster (positiva värden). Dessa motsvaras av talen i **value** itererat över diagrammets dimensioner. Betalningar och inkomster förväntas i slutet av varje period.

Syntax:

```
NPV([TOTAL [<fld {,fld}>]] discount_rate, value)
```

Returnerad datatyp: numeriska Som standard formateras resultatet som valuta.

Argument:

Argument

Argument	Beskrivning
discount_rate	**discount_rate** är procentandelen för rabatt som tillämpas.
value	Det uttryck eller fält som innehåller de data som ska mätas.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena. Bestämningen **TOTAL** kan följas av en lista med ett eller flera fältnamn inom vinkelparenteser. Dessa fältnamn bör ingå i en underuppsättning av diagrammets dimensionsvariabler. Vid uträkningen beaktas i detta fall alla diagrammets dimensionsvariabler utom de som ingår i listan, d.v.s. ett värde returneras för varje kombination av fältvärden i dimensionsfälten som finns i listan. Även fält som för närvarande inte utgör en dimension i ett diagram kan ingå i listan. Detta är användbart när man arbetar med grupper av dimensioner där dimensionsfälten inte är fasta. Genom att lista alla variabler i gruppen säkerställs att funktionen fungerar när den hierarkiska nivån ändras.

Begränsningar:

discount_rate och **value** får inte innehålla aggregeringsfunktioner, såvida inte dessa inre aggregeringar innehåller kvalificeraren **TOTAL**. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden samt saknade värden ignoreras.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`NPV(Discount, Payments)`	-540,12 $

Data som används i exempel:

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Discount|Payments
```

```
2013-01-01|0.1|-10000
2013-03-01|0.1|3000
2013-10-30|0.1|4200
2014-02-01|0.2|6800
] (delimiter is '|');
```

Se även:

p *XNPV - diagramfunktion (page 459)*
p *Aggr - diagramfunktion (page 612)*

XIRR

XIRR() returnerar den aggregerade internräntan (per år) för en tablå av kassaflöden (som inte nödvändigtvis är periodisk), representerad av parvisa tal i **pmt** och **date** itererade över ett antal poster enligt vad som definierats i en group by-sats. Alla betalningar diskonteras utifrån ett 365-dagarsår.

Qliks XIRR-funktionalitet (**XIRR()**- och **RangeXIRR()**-funktionerna) använder följande ekvation för att lösa värdet `Rate` och fastställa korrekt XIRR-värde:

```
XNPV(Rate, pmt, date) = 0
```

Ekvationen löses med en förenklad version av Newton-metoden.

Syntax:

```
XIRR(pmt, date )
```

Returnerad datatyp: numeriska

Argument

Argument	Beskrivning
pmt	Betalningar. Uttrycket eller fältet som innehåller de kassaflöden som motsvarar det betalningsschema som anges i **date**.
date	Uttrycket eller fältet som innehåller de datascheman som motsvarar kassaflödesbetalningar som ges i **pmt**.

När du arbetar med den här funktionen gäller följande begränsningar:

- Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.
- Denna funktion kräver minst en giltig negativ och minst en giltig positiv betalning (med motsvarande giltiga datum). Om dessa betalningar inte anges returneras ett NULL-värde.

Dessa ämnen kan hjälpa dig att arbeta med den här funktionen:

- *XNPV (page 452)*: använd den här funktionen för att beräkna det aggregerade aktuella nettovärdet för ett schema med kassaflöden.

- *RangeXIRR (page 1616)*: **RangeXIRR()** är den ekvivalenta intervallfunktionen för **XIRR()**-funktionen.

Den underliggande algoritmen som används av den här funktionen varierar mellan olika versioner av För klienthanterat Qlik Sense. Mer information om de senaste uppdateringarna av algoritmen finns i supportartikeln Korrigeringar och uppdateringar av XIRR-funktioner*.*

Exempel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Transaktionsdata för en serie kassaflöden.
- Användningen av **XIRR()**-funktionen för att beräkna den interna årliga avkastningsgraden för dessa kassaflöden.

Laddningsskript

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Payments
2013-01-01|-10000
2013-03-01|3000
2013-10-30|4200
2014-02-01|6800
] (delimiter is '|');

Cashflow1:
LOAD Year,XIRR(Payments, Date) as XIRR2013  Resident Cashflow Group By Year;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Year`
- `XIRR2013`

Resultattabell

År	XIRR2013
2013	0.5385

Tolkning av XIRR-returvärdet

XIRR-funktionen används vanligtvis för att analysera en investering där det finns en utgående (negativ) betalning i början och sedan en serie mindre inkomstbetalningar (positiva) senare. Här är ett förenklat exempel med endast en negativ och en positiv betalning:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|-100
2024-01-01|110
] (delimiter is '|');
```

Vi gör en första betalning på 100 och får tillbaka 110 efter exakt ett år. Detta motsvarar en avkastningsgrad på 10 % per år. `XIRR(Payments, Date)` returnerar ett värde på 0,1.

Returvärdet för XIRR-funktionen kan vara positivt eller negativt. När det gäller en investering visar ett negativt resultat att investeringen är en förlust. Vinst- eller förlustbeloppet kan beräknas helt enkelt genom att göra en summering av betalningsfältet.

I exemplet ovan lånar vi ut våra pengar i ett år. Avkastningsgraden kan ses som en ränta. Det är också möjligt att använda XIRR:s funktion när du är på den andra sidan av transaktionen (t.ex. om du är låntagare i stället för långivare).

Ta det här exemplet:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|100
2024-01-01|-110
] (delimiter is '|');
```

Detta är samma sak som i det första exemplet, men i omvänd form. Här lånar vi 100 euro i ett år och betalar tillbaka dem med 10 % ränta. I det här exemplet ger XIRR-beräkningen 0,1 (10 %), samma värde som i det första exemplet.

Observera att vi i det första exemplet fick en vinst på 10 och i det andra exemplet en förlust på 10, men att returvärdet för XIRR-funktionen är positivt för båda dessa exempel. Detta beror på att XIRR-funktionen beräknar den dolda räntan i transaktionen, oavsett vilken sida du är på i transaktionen.

Begränsningar med flera lösningar

Qliks XIRR-funktion definieras av följande ekvation, där värdet `Rate` löses:

```
XNPV(Rate, pmt, date) = 0
```

Det är ibland möjligt att denna ekvation har fler än en lösning. Detta är känt som "multipel IRR-problemet" och orsakas av ett icke-normalt kassaflöde (även kallat okonventionellt kassaflöde). Följande laddningsskript visar ett exempel på detta:

```
Cashflow:
LOAD * inline [
Date|Payments
2021-01-01|-200
```

```
2022-01-01|500
2023-01-01|-250
] (delimiter is '|');
```

I detta exempel finns det en negativ lösning och en positiv lösning (`Rate`= -0,3 och `Rate`= 0,8). **XIRR()** kommer att returnera 0,8.

När Qliks XIRR-funktion söker efter en lösning börjar den vid `Rate` = 0 och ökar värdet stegvis tills den hittar en lösning. Om det finns fler än en positiv lösning returneras den första lösningen. Om den inte kan hitta en positiv lösning kommer den att återställa `Rate` tillbaka till noll och börja söka efter en lösning i den negativa riktningen.

Observera att ett "normalt" kassaflöde garanterat har endast en lösning. Ett "normalt" kassaflöde innebär att alla betalningar med samma tecken (positiva eller negativa) ingår i en kontinuerlig grupp.

Se även:

p *XNPV (page 452)*
p *RangeXIRR (page 1616)*
≤ Korrigeringar och uppdateringar av XIRR-funktioner

XIRR - diagramfunktion

XIRR() returnerar den aggregerade avkastningsgraden (per år) för ett kassaflödesschema (inte nödvändigtvis periodiskt) motsvarat av parvisa tal i de uttryck som ges av **pmt** och **date** itererat över diagrammets dimensioner. Alla betalningar diskonteras utifrån ett 365-dagarsår.

Qliks XIRR-funktionalitet (**XIRR()**- och **RangeXIRR()**-funktionerna) använder följande ekvation för att lösa värdet `Rate` och fastställa korrekt XIRR-värde:

```
XNPV(Rate, pmt, date) = 0
```

Ekvationen löses med en förenklad version av Newton-metoden.

Syntax:

```
XIRR([TOTAL [<fld {,fld}>]] pmt, date)
```

Returnerad datatyp: numeriska

Argument

Argument	Beskrivning
pmt	Betalningar. Uttrycket eller fältet som innehåller de kassaflöden som motsvarar det betalningsschema som anges i **date**.
date	Uttrycket eller fältet som innehåller de datascheman som motsvarar kassaflödesbetalningar som ges i **pmt**.

Argument	Beskrivning
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

När du arbetar med den här funktionen gäller följande begränsningar:

- **pmt** och **date** får inte innehålla aggregeringsfunktioner, såvida inte dessa inre aggregeringar innehåller kvalificeraren **TOTAL**. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.
- Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.
- Denna funktion kräver minst en giltig negativ och minst en giltig positiv betalning (med motsvarande giltiga datum). Om dessa betalningar inte anges returneras ett NULL-värde.

Dessa ämnen kan hjälpa dig att arbeta med den här funktionen:

- *XNPV - diagramfunktion (page 459)*: använd den här funktionen för att beräkna det aggregerade aktuella nettovärdet för ett schema med kassaflöden.
- *RangeXIRR (page 1616)*: **RangeXIRR()** är den ekvivalenta intervallfunktionen för **XIRR()**-funktionen.

> *Den underliggande algoritmen som används av den här funktionen varierar mellan olika versioner av För klienthanterat Qlik Sense. Mer information om de senaste uppdateringarna av algoritmen finns i supportartikeln* Korrigeringar och uppdateringar av XIRR-funktioner*.*

Exempel

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller kassaflödestransaktioner.
- Information som lagras i en tabell med benämningen `Cashflow`.

Laddningsskript

```
Cashflow:
LOAD 2013 as Year, * inline [
```

```
Date|Payments
2013-01-01|-10000
2013-03-01|3000
2013-10-30|4200
2014-02-01|6800
] (delimiter is '|');
```

Resultat

Gör följande:

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till följande beräkning som ett mått:

```
=XIRR(Payments, Date)
```

Resultattabell

=XIRR(betalningar, datum)
0.5385

Tolkning av XIRR-returvärdet

XIRR-funktionen används vanligtvis för att analysera en investering där det finns en utgående (negativ) betalning i början och sedan en serie mindre inkomstbetalningar (positiva) senare. Här är ett förenklat exempel med endast en negativ och en positiv betalning:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|-100
2024-01-01|110
] (delimiter is '|');
```

Vi gör en första betalning på 100 och får tillbaka 110 efter exakt ett år. Detta motsvarar en avkastningsgrad på 10 % per år. `XIRR(Payments, Date)` returnerar ett värde på 0,1.

Returvärdet för XIRR-funktionen kan vara positivt eller negativt. När det gäller en investering visar ett negativt resultat att investeringen är en förlust. Vinst- eller förlustbeloppet kan beräknas helt enkelt genom att göra en summering av betalningsfältet.

I exemplet ovan lånar vi ut våra pengar i ett år. Avkastningsgraden kan ses som en ränta. Det är också möjligt att använda XIRR:s funktion när du är på den andra sidan av transaktionen (t.ex. om du är låntagare i stället för långivare).

Ta det här exemplet:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|100
2024-01-01|-110
] (delimiter is '|');
```

Detta är samma sak som i det första exemplet, men i omvänd form. Här lånar vi 100 euro i ett år och betalar tillbaka dem med 10 % ränta. I det här exemplet ger XIRR-beräkningen 0,1 (10 %), samma värde som i det första exemplet.

Observera att vi i det första exemplet fick en vinst på 10 och i det andra exemplet en förlust på 10, men att returvärdet för XIRR-funktionen är positivt för båda dessa exempel. Detta beror på att XIRR-funktionen beräknar den dolda räntan i transaktionen, oavsett vilken sida du är på i transaktionen.

Begränsningar med flera lösningar

Qliks XIRR-funktion definieras av följande ekvation, där värdet `Rate` löses:

```
XNPV(Rate, pmt, date) = 0
```

Det är ibland möjligt att denna ekvation har fler än en lösning. Detta är känt som "multipel IRR-problemet" och orsakas av ett icke-normalt kassaflöde (även kallat okonventionellt kassaflöde). Följande laddningsskript visar ett exempel på detta:

```
Cashflow:
LOAD * inline [
Date|Payments
2021-01-01|-200
2022-01-01|500
2023-01-01|-250
] (delimiter is '|');
```

I detta exempel finns det en negativ lösning och en positiv lösning (`Rate`= -0,3 och `Rate`= 0,8). **XIRR()** kommer att returnera 0,8.

När Qliks XIRR-funktion söker efter en lösning börjar den vid `Rate` = 0 och ökar värdet stegvis tills den hittar en lösning. Om det finns fler än en positiv lösning returneras den första lösningen. Om den inte kan hitta en positiv lösning kommer den att återställa `Rate` tillbaka till noll och börja söka efter en lösning i den negativa riktningen.

Observera att ett "normalt" kassaflöde garanterat har endast en lösning. Ett "normalt" kassaflöde innebär att alla betalningar med samma tecken (positiva eller negativa) ingår i en kontinuerlig grupp.

Se även:

p *IRR - diagramfunktion (page 436)*
p *Aggr - diagramfunktion (page 612)*
≤ Korrigeringar och uppdateringar av XIRR-funktioner

XNPV

XNPV() returnerar det aggregerade aktuella nettovärdet för en tablå av kassaflöden (som inte nödvändigtvis är periodisk), representerade av parvisa tal i **pmt** och **date**. Alla betalningar diskonteras utifrån ett 365-dagarsår.

Syntax:

```
XNPV(discount_rate, pmt, date)
```

Returnerad datatyp: numeriska

Som standard formateras resultatet som valuta.

Formeln för att beräkna XNPV visas nedan:

XNPV-aggregeringsformel

$$XNPV = \sum_{i=1}^{n} \frac{P_i}{(1+rate)^{(di-d1)/365}}$$

där:

- P_i = nettokassainflöden och -utflöden under en enstaka period i
- d_1 = det första betalningsdatumet
- d_i = det i:e betalningsdatumet
- $rate$ = rabattsats

Det aktuella nettovärdet, eller NPV (Net Present Value), används för att beräkna det aktuella totala värdet för en framtida ström av kassaflöden baserat på en räntesats. För att beräkna XNPV måste vi uppskatta framtida kassaflöden med motsvarande datum. Därefter tillämpar vi för varje betalning den effektiva diskonteringsräntan baserat på betalningsdatumet.

Att utföra XNPV-aggregeringen över en serie betalningar liknar att utföra en summaaggregering över dessa betalningar. Skillnaden är att varje belopp ändras (eller "diskonteras") beroende på vald diskonteringsränta (liknande räntesats) och hur långt in i framtiden betalningen ligger. Om XNPV utförs med parametern **discount_rate** satt till noll kommer XNPV att vara likvärdigt med en summering (betalningarna kommer inte att ändras innan de summeras). I allmänhet gäller att ju närmare noll **discount_rate** är satt, desto mer liknar XNPV-resultatet resultatet av en summaaggregering.

Argument

Argument	Beskrivning
discount_rate	**discount_rate** är den årliga ränta som betalningarna ska diskonteras med. Ett värde på 0,1 anger en rabattsats på 10 %.

Argument	Beskrivning
pmt	Betalningar. Uttrycket eller fältet som innehåller de kassaflöden som motsvarar det betalningsschema som anges i **date**. Positiva värden antas vara inflöden och negativa värden antas vara utflöden. *XNPV() diskonterar inte det första kassaflödet eftersom det alltid kommer att inträffa på startdatumet. Efterföljande betalningar rabatteras utifrån ett 365-dagarsår. Detta skiljer sig från **NPV()**, där även den första betalningen diskonteras.*
date	Uttrycket eller fältet som innehåller de datascheman som motsvarar kassaflödesbetalningar som ges i **pmt**. Det första värdet används som startdatum för att beräkna tidsförskjutningar för framtida kassaflöden.

När du arbetar med den här funktionen gäller följande begränsningar:

- Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Användning

- `XNPV()` används vid finansiell modellering för beräkning av det aktuella nettovärdet (NPV) för en investeringsmöjlighet.
- På grund av sin högre precision är XNPV att föredra framför NPV för alla typer av finansiella modeller.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – enstaka betalning (skript)

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med ett projekt och dess kassaflöde för ett år, i en tabell som heter `CashFlow`. Det initiala datumet för beräkningen är inställt till 1 juli 2022, med ett nettokassaflöde på 0. Efter ett år inträffar ett kassaflöde på 1000 USD.
- En resident load från `CashFlow`-tabellen, som används för att beräkna `XNPV`-fältet för projektet i en tabell som heter `XNPV`.
- En hårdkodad rabattsats på 10 % (0,1) används i XNPV-beräkningen.
- En `Group By`-sats används för att gruppera alla betalningar för projektet.

Laddningsskript

```
CashFlow:
Load
*
Inline
[
PrjId,Dates,Values
1,'07/01/2022',0
1,'07/01/2023',1000
];

XNPV:
Load
       PrjId,
       XNPV(0.1,Values,Dates) as XNPV //Discount Rate of 10%
Resident CashFlow
Group By PrjId;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `PrjId`
- `XNPV`

Resultattabell

PrjId	XNPV
1	$909.09

Enligt formeln är XNPV-värdet för den första posten 0, och för den andra posten är XNPV-värdet 909,09 USD. Totalt XNPV är 909,09 USD.

Exempel 2 – flera betalningar (skript)

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med ett projekt och dess kassaflöde för ett år, i en tabell som heter `CashFlow`.
- En resident load från `CashFlow`-tabellen, som används för att beräkna `XNPV`-fältet för projektet i en tabell som heter `XNPV`.
- En hårdkodad rabattsats på 10 % (0,1) används i XNPV-beräkningen.
- En `Group By`-sats används för att gruppera alla betalningar för projektet.

Laddningsskript

```
CashFlow:
Load
*
Inline
[
PrjId,Dates,Values
1,'07/01/2022',0
1,'07/01/2024',500
1,'07/01/2023',1000
];

XNPV:
Load
       PrjId,
       XNPV(0.1,Values,Dates) as XNPV //Discount Rate of 10%
Resident CashFlow
Group By PrjId;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `PrjId`
- `XNPV`

Resultattabell

PrjId	XNPV
1	$1322.21

I det här exemplet erhålls en betalning på 1000 USD på slutet av det första året och en betalning på 500 USD erhålls på slutet av andra året. Med en rabattsats på 10 % per period blir effektiv XNPV 1322,21 USD.

Observerar att endast den första raden med data ska hänvisa till basdatumet för beräkningar. För resten av raderna spelar ordningen ingen roll, eftersom datumparametern används för att beräkna den förbrukade tiden.

Exempel 3 – flera betalningar och oregelbundna kassaflöden (skript)

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Rabattsatser för två projekt som i en tabell som heter `Project`.
- Kassaflöden för flera perioder för varje projekt efter projekt-ID och datum. `Dates`-fältet används för att beräkna hur länge respektive rabattsatsen tillämpas för kassaflödet. Bortsett från från den första posten (initialt kassaflöde och datum) spelar posternas ordning ingen roll och om den ändras påverkas inte beräkningarna.
- Med en kombination av `NoConcatenate`, resident loads och `Left Join`-funktionen skapas en temporär tabell `tmpNPV` som kombinerar posterna i tabellerna `Project` och `CashFlow` till en platt tabell. Den här tabellen kommer att ha repeterade rabattsatser för varje kassaflöde.
- En resident load från `tmpNPV`-tabellen, som används för att beräkna `XNPV`-fältet för varje projekt i en tabell som heter `XNPV`.
- Den enstaka rabattsatsen som är kopplad till varje projekt hämtas med `only()`-funktionen och används i `XNPV`-beräkningen för varje projekt.
- En `Group By`-sats som grupperar efter projekt-ID används för att gruppera alla betalningar och motsvarande datum för varje projekt.
- För att undvika att syntetiska eller redundanta data laddas till datamodellen släpps `tmpXNPV`-tabellen på slutet av skriptet.

Laddningsskript

```
Project:
Load * inline [
PrjId,Discount_Rate
1,0.1
2,0.15
];

CashFlow:
Load
*
Inline
[
PrjId,Dates,Values
1,'07/01/2021',0
1,'07/01/2022',1000
1,'07/01/2023',1000
```

```
2,'07/01/2020',0
2,'07/01/2023',500
2,'07/01/2024',1000
2,'07/01/2022',500
];

tmpXNPV:
NoConcatenate Load *
Resident Project;
Left Join
Load *
Resident CashFlow;

XNPV:
Load
       PrjId,
       XNPV(Only(Discount_Rate),Values,Dates) as XNPV //Discount Rate will be 10% for Project 1 and
15% for Project 2
Resident tmpXNPV
Group By PrjId;

Drop table tmpXNPV;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `PrjId`
- `XNPV`

Resultattabell

PrjId	XNPV
1	$1735.54
2	$278.36

Projekt-ID har ett initialt kassaflöde på 0 USD den 1 juli 2021. Det finns två betalningar på 1000 USD som ska erhållas på slutet av två på varandra följande år, med en rabattsats på 10 % per period. Därför blir effektiv XNPV 1735.54 USD.

Projekt-ID 2 har ett initialt utflöde av 1000 USD (därför används minustecken) den 1 juli 2020. Efter två år förväntas en betalning på 500 USD. Efter två år förväntas ytterligare en betalning på 500 USD. Slutligen förväntas en betalning på 1000 USD den 1 juli 2024. Med rabattsatsen på 15 % blir effektiv XNPV 278,36 USD.

Se även:

p *Drop table (page 176)*
p *group by (page 187)*
p *Join (page 73)*
p *Max (page 378)*

p *NoConcatenate (page 103)*
p *NPV - diagramfunktion (page 444)*
p *Only (page 394)*

XNPV - diagramfunktion

XNPV() returnerar det aktuella aggregerade nettovärdet för ett kassaflödesschema (inte nödvändigtvis periodiskt) motsvarat av parvisa tal i de uttryck som ges av **pmt** och **date** itererat över diagramdimensionerna. Alla betalningar diskonteras utifrån ett 365-dagarsår.

Syntax:

```
XNPV([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

Returnerad datatyp: numeriska

> *Som standard formateras resultatet som valuta.*

Formeln för att beräkna XNPV visas nedan:

XNPV-aggregeringsformel

$$XNPV = \sum_{i=1}^{n} \frac{P_i}{(1+rate)^{(di-d1)/365}}$$

där:

- P_i = nettokassainflöden och -utflöden under en enstaka period `i`
- d_1 = det första betalningsdatumet
- d_i = det `i`:e betalningsdatumet
- `rate` = rabattsats

Det aktuella nettovärdet, eller NPV (Net Present Value), används för att beräkna det aktuella totala värdet för en framtida ström av kassaflöden baserat på en räntesats. För att beräkna XNPV måste vi uppskatta framtida kassaflöden med motsvarande datum. Därefter tillämpar vi för varje betalning den effektiva diskonteringsräntan baserat på betalningsdatumet.

Att utföra XNPV-aggregeringen över en serie betalningar liknar att utföra en summaaggregering över dessa betalningar. Skillnaden är att varje belopp ändras (eller "diskonteras") beroende på vald diskonteringsränta (liknande räntesats) och hur långt in i framtiden betalningen ligger. Om XNPV utförs med parametern **discount_rate** satt till noll kommer XNPV att vara likvärdigt med en summering (betalningarna kommer inte att ändras innan de summeras). I allmänhet gäller att ju närmare noll **discount_rate** är satt, desto mer liknar XNPV-resultatet resultatet av en summaaggregering.

Argument

Argument	Beskrivning
discount_rate	**discount_rate** är den årliga ränta som betalningarna ska diskonteras med. Ett värde på 0,1 anger en rabattsats på 10 %.
pmt	Betalningar. Uttrycket eller fältet som innehåller de kassaflöden som motsvarar det betalningsschema som anges i **date**. Positiva värden antas vara inflöden och negativa värden antas vara utflöden. *XNPV() diskonterar inte det första kassaflödet eftersom det alltid kommer att inträffa på startdatumet. Efterföljande betalningar rabatteras utifrån ett 365-dagarsår. Detta skiljer sig från **NPV()**, där även den första betalningen diskonteras.*
date	Uttrycket eller fältet som innehåller de datascheman som motsvarar kassaflödesbetalningar som ges i **pmt**. Det första värdet används som startdatum för att beräkna tidsförskjutning för framtida kassaflöden.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {,fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

När du arbetar med den här funktionen gäller följande begränsningar:

- **discount_rate**,**pmt** och **date** får inte innehålla aggregeringsfunktioner, såvida inte dessa inre aggregeringar innehåller kvalificerarna **TOTAL** eller **ALL**. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.
- Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Användning

- `XNPV()` används vid finansiell modellering för beräkning av det aktuella nettovärdet (NPV) för en investeringsmöjlighet.
- På grund av sin högre precision är XNPV att föredra framför NPV för alla typer av finansiella modeller.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan

vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller kassaflödestransaktioner.
- Information som lagras i en tabell med benämningen `Cashflow`.

Laddningsskript

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Payments
2013-01-01|-10000
2013-03-01|3000
2013-10-30|4200
2014-02-01|6800
] (delimiter is '|');
```

Resultat

Gör följande:

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till följande beräkning som ett mått:

```
=XNPV(0.09, Payments, Date)
```

Resultattabell

=XNPV(0,09, betalningar, datum)
$3062.49

Se även:

p *NPV - diagramfunktion (page 444)*
p *Aggr - diagramfunktion (page 612)*

Statistiska aggregeringsfunktioner

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Statistiska aggregeringsfunktioner i dataladdningsskriptet

Följande statistiska aggregeringsfunktioner kan användas i skript.

Avg

Avg() hittar medelvärdet för aggregerade data i uttrycket över ett antal poster som definieras av en **group by**-sats.

```
Avg ([distinct] expression)
```

Correl

Correl() returnerar den aggregerade korrelationskoefficienten för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definierats i en **group by**-sats.

```
Correl (x-expression, y-expression)
```

Fractile

Fractile() hittar värdet som motsvarar den inklusiva fraktilen (kvantilen) för aggregerade data i uttrycket över ett antal poster som definieras av en **group by**-sats.

```
Fractile (expression, fractile)
```

FractileExc

FractileExc() hittar värdet som motsvarar den exklusiva fraktilen (kvantilen) för aggregerade data i uttrycket över ett antal poster som definieras av en **group by**-sats.

```
FractileExc (expression, fractile)
```

Kurtosis

Kurtosis() returnerar graden av kurtosis av data i uttrycket över ett antal poster som definieras av en **group by**-sats.

```
Kurtosis ([distinct ] expression )
```

LINEST_B

LINEST_B() returnerar det aggregerade b-värdet (y-intercept) hos en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras i en **group by**-sats.

```
LINEST_B (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_df

LINEST_DF() returnerar den aggregerade frihetsgraden hos en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras i en**group by**-sats.

```
LINEST_DF (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_f

Denna skriptfunktion returnerar det aggregerade F-värdet ($r^2/(1-r^2)$) hos en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras av en **group by**-sats.

```
LINEST_F (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_m

LINEST_M() returnerar det aggregerade m-värdet (lutning) hos en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras i en **group by**-sats.

```
LINEST_M (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_r2

LINEST_R2() returnerar det aggregerade r^2-värdet (bestämningskoefficienten) av en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som representeras av parvisa tal i x-expression och y-expression itererat över ett antal poster som definieras av en **group by**-sats.

```
LINEST_R2 (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_seb

LINEST_SEB() returnerar det aggregerade standardfelet hos b-värdet i en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt definitionen av en **group by**-sats.

```
LINEST_SEB (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sem

LINEST_SEM() returnerar det aggregerade standardfelet hos m-värdet i en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt definitionen av en **group by**-sats.

```
LINEST_SEM (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sey

LINEST_SEY() returnerar det aggregerade standardfelet hos y-uppskattningen i en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt definitionen av en **group by**-sats.

```
LINEST_SEY (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_ssreg

LINEST_SSREG returnerar den aggregerade restsumman av en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som representeras av parvisa tal i x-expression och y-expression itererat över ett antal poster som definieras av en **group by**-sats.

```
LINEST_SSREG (y-expression, x-expression [, y0 [, x0 ]])
```

Linest_ssresid

LINEST_SSRESID() returnerar den aggregerade restsumman av en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som representeras av parvisa tal i x-expression och y-expression itererat över ett antal poster som definieras av en **group by**-sats.

```
LINEST_SSRESID (y-expression, x-expression [, y0 [, x0 ]])
```

Median

Median() returnerar den aggregerade medianen av värdena i uttrycket över ett antal poster som definieras av en **group by**-sats.

```
Median (expression)
```

Skew

Skew() returnerar skevheten i uttryck över ett antal poster enligt vad som definierats i en **group by**-sats.

```
Skew ([ distinct] expression)
```

Stdev

Stdev() returnerar värdenas standardavvikelse i uttrycket över ett antal poster som definieras av en **group by**-sats.

```
Stdev ([distinct] expression)
```

Sterr

Sterr() returnerar det aggregerade standardfelet (stdev/sqrt(n)) för en serie värden, representerade av uttryck över ett antal poster enligt vad som definierats i en **group by**-sats.

```
Sterr ([distinct] expression)
```

STEYX

STEYX() returnerar det aggregerade standardfelet hos det predikterade y-värdet för varje x-värde i regressionen för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definierats i en **group by**-sats.

```
STEYX (y-expression, x-expression)
```

Statistiska aggregeringsfunktioner i diagramuttryck

Följande statistiska aggregeringsfunktioner kan användas i diagram:

Avg

Avg() returnerar det aggregerade medelvärdet av uttrycket eller fältet itererat över diagrammets dimensioner.

```
Avg - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} expr)
```

Correl

Correl() returnerar den aggregerade korrelationskoefficienten för två datauppsättningar. Korrelationsfunktionen är ett mått på förhållandet mellan datauppsättningarna och aggregeras för (x,y)-värdeparen itererat över diagrammets dimensioner.

```
Correl - diagramfunktion({[SetExpression] [TOTAL [<fld {, fld}>]]} value1, value2 )
```

Fractile

Fractile() finner det värde som motsvarar den inklusiva fraktilen (kvantilen) av aggregerade data i det intervall som ges av uttrycket itererat över diagrammets dimensioner.

```
Fractile - diagramfunktion({[SetExpression] [TOTAL [<fld {, fld}>]]} expr, fraction)
```

FractileExc

FractileExc() finner det värde som motsvarar den exklusiva fraktilen (kvantilen) av aggregerade data i det intervall som ges av uttrycket itererat över diagrammets dimensioner.

```
FractileExc - diagramfunktion({[SetExpression] [TOTAL [<fld {, fld}>]]} expr, fraction)
```

Kurtosis

Kurtosis() finner kurtosis av dataintervallet aggregerat i uttrycket eller fältet itererat över diagrammets dimensioner.

```
Kurtosis - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} expr)
```

LINEST_b

LINEST_B() returnerar det aggregerade b-värdet (y-intercept) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_R2 - diagramfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_value, x_value[, y0_const[, x0_const]])
```

LINEST_df

LINEST_DF() returnerar de aggregerade frihetsgraderna för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_DF - diagramfunktion({[SetExpression] [TOTAL [<fld{, fld}>]]} y_value, x_value [, y0_const [, x0_const]])
```

LINEST_f

LINEST_F() returnerar det aggregerade F-värdet (r2/(1-r2)) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_F - diagramfunktion({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_m

LINEST_M() returnerar det aggregerade m-värdet (lutningen) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_M - diagramfunktion({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_r2

LINEST_R2() returnerar det aggregerade r2-värdet (bestämningskoefficienten) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_R2 - diagramfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_value,
x_value[, y0_const[, x0_const]])
```

LINEST_seb

LINEST_SEB() returnerar det aggregerade standardfelet för b-värdet av en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_SEB - diagramfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_sem

LINEST_SEM() returnerar det aggregerade standardfelet för m-värdet av en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_SEM - diagramfunktion([{set_expression}][ distinct ] [total [<fld
{,fld}>] ] y-expression, x-expression [, y0 [, x0 ]] )
```

LINEST_sey

LINEST_SEY() returnerar det aggregerade standardfelet för y-estimatet av en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_SEY - diagramfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssreg

LINEST_SSREG() returnerar den aggregerade regressionskvadratsumman av en linjär regression definierad genom ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_SSREG - diagramfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssresid

LINEST_SSRESID() returnerar den aggregerade residualkvadratsumman hos en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_SSRESID - diagramfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

Median

Median() returnerar medianvärdet för värdeintervallet som aggregeras i uttrycket, itererat över diagrammets dimensioner.

```
Median - diagramfunktion({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MutualInfo

MutualInfo beräknar ömsesidig information (MI) mellan två fält eller mellan aggregerade värden i **Aggr()**.

```
MutualInfo - diagramfunktion{[SetExpression] [DISTINCT] [TOTAL target, driver
[, datatype [, breakdownbyvalue [, samplesize ]]])
```

Skew

Skew() returnerar den aggregerade skevheten av uttrycket eller fältet itererat över diagrammets dimensioner.

```
Skew - diagramfunktion{[SetExpression] [DISTINCT] [TOTAL [<fld{ ,fld}>]]}
expr)
```

Stdev

Stdev() finner standardavvikelsen för dataintervallet som aggregerats i uttrycket eller fältet itererat över diagrammets dimensioner.

```
Stdev - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]}
expr)
```

Sterr

Sterr() finner värdet av standardfelet av medelvärdet (stdev/sqrt(n)) för värdeserien som aggregerats i uttrycket itererat över diagrammets dimensioner.

```
Sterr - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL[<fld{, fld}>]]}
expr)
```

STEYX

STEYX() returnerar det aggregerade standardfelet vid prediktering av y-värden för varje x-värde i en linjär regression som ges av en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **y_value** och **x_value**.

```
STEYX - diagramfunktion{[SetExpression] [TOTAL [<fld{, fld}>]]} y_value, x_
value)
```

Avg

Avg() hittar medelvärdet för aggregerade data i uttrycket över ett antal poster som definieras av en **group by**-sats.

Syntax:

```
Avg([DISTINCT] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
DISTINCT	Om predikatet **distinct** förekommer framför uttrycket kommer alla dubbletter att ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

Resulterande data

Exempel	Resultat
Temp: crosstable (Month, Sales) load * inline [Customer|Jan|Feb|Mar||Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec Astrida|46|60|70|13|78|20|45|65|78|12|78|22 Betacab|65|56|22|79|12|56|45|24|32|78|55|15 Canutility|77|68|34|91|24|68|57|36|44|90|67|27 Divadip|36|44|90|67|27|57|68|47|90|80|94] (delimiter is '|'); Avg1: LOAD Customer, Avg(Sales) as MyAverageSalesByCustomer Resident Temp Group By Customer;	Customer MyAverageSalesByCustomer Astrida 48.916667 Betacab 44.916667 Canutility 56.916667 Divadip 63.083333 Detta kan kontrolleras på arket genom att skapa en tabell som innehåller måttet: Sum(Sales)/12
Givet att **Temp**-tabellen är laddad som i föregående exempel: LOAD Customer,Avg(DISTINCT Sales) as MyAvgSalesDistinct Resident Temp Group By Customer;	Customer MyAverageSalesByCustomer Astrida 43.1 Betacab 43.909091 Canutility 55.909091 Divadip 61 Enbart distinkta värden räknas. Dela summan med antalet värden som inte är dubbletter.

Avg - diagramfunktion

Avg() returnerar det aggregerade medelvärdet av uttrycket eller fältet itererat över diagrammets dimensioner.

Syntax:

```
Avg([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Argument	Beskrivning
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {,fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Exempel och resultat:

Example table

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Astrida	46	60	70	13	78	20	45	65	78	12	78	22
Betacab	65	56	22	79	12	56	45	24	32	78	55	15
Canutility	77	68	34	91	24	68	57	36	44	90	67	27
Divadip	57	36	44	90	67	27	57	68	47	90	80	94

Exempel på funktioner

Exempel	Resultat
`Avg(Sales)`	Om en tabell innehåller dimensionen `Customer` och måttet `Avg([Sales])` har den resultatet 2566 om **Totalvärden** visas.
`Avg([TOTAL (Sales))`	53.458333 för alla värden för `Customer`, eftersom TOTAL-kvalificeraren innebär att dimensionerna ignoreras.
`Avg(DISTINCT (Sales))`	51.862069 som totalvärde, eftersom Distinct-kvalificeraren innebär att endast unika värden i `Sales` för varje `Customer` utvärderas.

Data som används i exempel:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
```

```
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Se även:

p *Aggr - diagramfunktion (page 612)*

Correl

Correl() returnerar den aggregerade korrelationskoefficienten för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definierats i en **group by**-sats.

Syntax:

```
Correl(value1, value2)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value1, value2	Uttrycken eller fälten som innehåller de två stickprovsuppsättningarna för vilka korrelationskoefficienten ska mätas.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

Resulterande data

Exempel	Resultat
Salary: Load *, 1 as Grp; LOAD * inline ["Employee name"|Gender|Age|Salary Aiden Charles|Male|20|25000 Brenda Davies|Male|25|32000 Charlotte Edberg|Female|45|56000 Daroush Ferrara|Male|31|29000 Eunice Goldblum|Female|31|32000 Freddy Halvorsen|Male|25|26000 Gauri Indu|Female|36|46000 Harry Jones|Male|38|40000 Ian Underwood|Male|40|45000 Jackie Kingsley|Female|23|28000] (delimiter is '|'); Correl1: LOAD Grp, Correl(Age,Salary) as Correl_ Salary Resident Salary Group By Grp;	I en tabell med dimensionen `Correl_Salary` visas resultatet av Correl()-beräkningen i dataladdningsskriptet: 0.9270611

Correl - diagramfunktion

Correl() returnerar den aggregerade korrelationskoefficienten för två datauppsättningar. Korrelationsfunktionen är ett mått på förhållandet mellan datauppsättningarna och aggregeras för (x,y)-värdeparen itererat över diagrammets dimensioner.

Syntax:

```
Correl([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] value1, value2 )
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value1, value2	Uttrycken eller fälten som innehåller de två stickprovsuppsättningarna för vilka korrelationskoefficienten ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`Correl(Age, Salary)`	Om en tabell innehåller dimensionen `Employee name` och måttet `Correl(Age, Salary)` är resultatet 0,9270611. Resultaten visas endast för totalcellen.

Exempel	Resultat
`Correl(TOTAL Age, Salary))`	0.927. Detta och följande resultat visas med tre decimaler för bättre läsbarhet. Om du skapar en filterruta med dimensionen Gender och gör val från den, ser du resultatet 0,951 när Female väljs och 0,939 om Male väljs. Detta beror på att valet utesluter alla resultat som inte hör till det andra värdet för Gender.
`Correl({1} TOTAL Age, Salary))`	0.927. Oberoende av urval. Detta beror på att set-uttrycket {1} bortser från alla urval och dimensioner.
`Correl(TOTAL <Gender> Age, Salary))`	0,927 i totalcellen, 0,939 för alla värden för Male och 0,951 för alla värden för Female. Det här motsvarar resultaten om du gör valen i en filterruta baserad på Gender.

Data som används i exempel:

```
Salary:

LOAD * inline [

"Employee name"|Gender|Age|Salary

Aiden Charles|Male|20|25000

Brenda Davies|Male|25|32000

Charlotte Edberg|Female|45|56000

Daroush Ferrara|Male|31|29000

Eunice Goldblum|Female|31|32000

Freddy Halvorsen|Male|25|26000

Gauri Indu|Female|36|46000

Harry Jones|Male|38|40000

Ian Underwood|Male|40|45000

Jackie Kingsley|Female|23|28000

] (delimiter is '|');
```

Se även:

p *Aggr - diagramfunktion (page 612)*
p *Avg - diagramfunktion (page 469)*
p *RangeCorrel (page 1582)*

Fractile

Fractile() hittar värdet som motsvarar den inklusiva fraktilen (kvantilen) för aggregerade data i uttrycket över ett antal poster som definieras av en **group by**-sats.

Du kan använda FractileExc (page 479) för att beräkna den exklusiva fraktilen.

Syntax:

```
Fractile(expr, fraction)
```

Returnerad datatyp: numeriska

Funktionen returnerar värdet som motsvarar rangordningen enligt definitionen `rank = fraction * (N-1) + 1` där `N` är antal värden i `expr`. Om `rank` inte är ett heltal görs en interpolering mellan de två närmaste värdena.

Argument:

Argument

Argument	Beskrivning
expr	Uttrycket eller fältet som innehåller de data som ska användas för beräkning av fraktilen.
fraction	Ett tal mellan 0 och 1 som motsvarar den fraktil (kvantil uttryckt som bråkdel) som ska beräknas.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

Resulterande data

Exempel	Resultat

Table1:
Crosstable (Type, Value)
Load recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2] (delimiter is '|');
 	
Fractile1:
LOAD Type,
Fractile(Value,0.75) as MyFractile
Resident Table1 Group By Type;
	I en tabell med dimensionerna `Type` och `MyFractile` blir resultaten av Fractile()-beräkningarna i dataladdningsskriptet:
	
	`Type MyFractile`
	
	`Comparison 27.5`
	
	`Observation 36`

Fractile - diagramfunktion

Fractile() finner det värde som motsvarar den inklusiva fraktilen (kvantilen) av aggregerade data i det intervall som ges av uttrycket itererat över diagrammets dimensioner.

Du kan använda FractileExc - diagramfunktion (page 480) för att beräkna den exklusiva fraktilen.

Syntax:

```
Fractile([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr, fraction)
```

Returnerad datatyp: numeriska

Funktionen returnerar värdet som motsvarar rangordningen enligt definitionen `rank = fraction * (N-1) + 1` där `N` är antal värden i `expr`. Om `rank` inte är ett heltal görs en interpolering mellan de två närmaste värdena.

Argument:

Argument

Argument	Beskrivning
expr	Uttrycket eller fältet som innehåller de data som ska användas för beräkning av fraktilen.
fraction	Ett tal mellan 0 och 1 som motsvarar den fraktil (kvantil uttryckt som bråkdel) som ska beräknas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Exempel och resultat:

Example table

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Astrida	46	60	70	13	78	20	45	65	78	12	78	22
Betacab	65	56	22	79	12	56	45	24	32	78	55	15
Canutility	77	68	34	91	24	68	57	36	44	90	67	27
Divadip	57	36	44	90	67	27	57	68	47	90	80	94

Exempel på funktioner

Exempel	Resultat
`Fractile (Sales, 0.75)`	Om en tabell innehåller dimensionen `Customer` och måttet `Fractile([Sales])` har den resultatet 71.75 om **Totalvärden** visas. Detta är punkten i värdefördelningen för `Sales` som 75 % av värdena faller under.
`Fractile (TOTAL Sales, 0.75))`	71.75 för alla värden för `Customer`, eftersom TOTAL-kvalificeraren innebär att dimensionerna ignoreras.
`Fractile (DISTINCT Sales, 0.75)`	70 som totalvärde, eftersom DISTINCT-kvalificeraren innebär att endast unika värden i `Sales` för varje `Customer` utvärderas.

Data som används i exempel:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Se även:

p *Aggr - diagramfunktion (page 612)*

FractileExc

FractileExc() hittar värdet som motsvarar den exklusiva fraktilen (kvantilen) för aggregerade data i uttrycket över ett antal poster som definieras av en **group by**-sats.

Du kan använda Fractile (page 475) för att beräkna den inkluderade fraktilen.

Syntax:

```
FractileExc(expr, fraction)
```

Returnerad datatyp: numeriska

Funktionen returnerar värdet som motsvarar rangordningen enligt definitionen `rank = fraction * (N+1)` där `N` är antal värden i `expr`. Om `rank` inte är ett heltal görs en interpolering mellan de två närmaste värdena.

Argument:

Argument

Argument	Beskrivning
expr	Uttrycket eller fältet som innehåller de data som ska användas för beräkning av fraktilen.
fraction	Ett tal mellan 0 och 1 som motsvarar den fraktil (kvantil uttryckt som bråkdel) som ska beräknas.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

Resulterande data

Exempel	Resultat

Table1:
Crosstable (Type, Value)
Load recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2] (delimiter is '|');
 	
Fractile1:
LOAD Type,
FractileExc(Value,0.75) as
MyFractile Resident Table1 Group
By Type;
	I en tabell med dimensionerna `Type` och `MyFractile` blir resultaten av FractileExc()-beräkningarna i dataladdningsskriptet:
	
	`Type MyFractile`
	
	`Comparison 28.5`
	
	`Observation 38`

FractileExc - diagramfunktion

FractileExc() finner det värde som motsvarar den exklusiva fraktilen (kvantilen) av aggregerade data i det intervall som ges av uttrycket itererat över diagrammets dimensioner.

Du kan använda Fractile - diagramfunktion (page 476) för att beräkna den inkluderade fraktilen.

Syntax:

```
FractileExc([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr,
fraction)
```

Returnerad datatyp: numeriska

Funktionen returnerar värdet som motsvarar rangordningen enligt definitionen `rank = fraction * (N+1)` där `N` är antal värden i `expr`. Om `rank` inte är ett heltal görs en interpolering mellan de två närmaste värdena.

Argument:

Argument

Argument	Beskrivning
expr	Uttrycket eller fältet som innehåller de data som ska användas för beräkning av fraktilen.
fraction	Ett tal mellan 0 och 1 som motsvarar den fraktil (kvantil uttryckt som bråkdel) som ska beräknas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Exempel och resultat:

Example table

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Astrida	46	60	70	13	78	20	45	65	78	12	78	22
Betacab	65	56	22	79	12	56	45	24	32	78	55	15
Canutility	77	68	34	91	24	68	57	36	44	90	67	27
Divadip	57	36	44	90	67	27	57	68	47	90	80	94

Exempel på funktioner

Exempel	Resultat
`FractileExc (Sales, 0.75)`	Om en tabell innehåller dimensionen `Customer` och måttet `FractileExc([Sales])` har den resultatet 75.25 om **Totalvärden** visas. Detta är punkten i värdefördelningen för `Sales` som 75 % av värdena faller under.
`FractileExc (TOTAL Sales, 0.75))`	75.25 för alla värden för `Customer`, eftersom TOTAL-kvalificeraren innebär att dimensionerna ignoreras.
`FractileExc (DISTINCT Sales, 0.75)`	73.50 som totalvärde, eftersom DISTINCT-kvalificeraren innebär att endast unika värden i `Sales` för varje `Customer` utvärderas.

Data som används i exempel:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Se även:

p *Aggr - diagramfunktion (page 612)*

Kurtosis

Kurtosis() returnerar graden av kurtosis av data i uttrycket över ett antal poster som definieras av en **group by**-sats.

Syntax:

```
Kurtosis([distinct ] expr )
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket kommer alla dubbletter att ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

Resulterande data

Exempel	Resultat
Table1: Crosstable (Type, Value) Load recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|'); Kurtosis1: LOAD Type, Kurtosis(Value) as MyKurtosis1, Kurtosis(DISTINCT Value) as MyKurtosis2 Resident Table1 Group By Type;	I en tabell med dimensionerna Type, MyKurtosis1 och MyKurtosis2 blir resultaten av Kurtosis()-beräkningarna i dataladdningsskriptet: Type MyKurtosis1 MyKurtosis2 Comparison -1.1612957 -1.4982366 Observation -1.1148768 -0.93540144

Kurtosis - diagramfunktion

Kurtosis() finner kurtosis av dataintervallet aggregerat i uttrycket eller fältet itererat över diagrammets dimensioner.

Syntax:

```
Kurtosis([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Argument	Beskrivning
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {,fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Exempel och resultat:

Example table

Type	Value																			
Comparison	2	27	38	31	1	19	1	34	3	1	2	3	2	1	2	1	3	29	37	2
Observation	35	40	12	15	21	14	46	10	28	48	16	30	32	48	31	22	12	39	19	25

Exempel på funktioner

Exempel	Resultat
`Kurtosis (Value)`	Om du har en tabell med dimensionen `Type` och måttet `Kurtosis(Value)` och **Totalvärden** visas för tabellen och talformatet är inställt på 3 signifikanta siffror är resultatet 1,252. För `Comparison` är det 1,161 och för `Observation` är det 1,115.
`Kurtosis (TOTAL Value))`	1.252 för alla värden för `Type`, eftersom TOTAL-kvalificeraren innebär att dimensionerna ignoreras.

Data som används i exempel:

```
Table1:
Crosstable (Type, Value)
Load recno() as ID, * inline [
Observation|Comparison
```

```
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2 ] (delimiter is '|');
```

Se även:

p *Avg - diagramfunktion (page 469)*

LINEST_B

LINEST_B() returnerar det aggregerade b-värdet (y-intercept) hos en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras i en **group by**-sats.

Syntax:

```
LINEST_B (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*

LINEST_B - diagramfunktion

LINEST_B() returnerar det aggregerade b-värdet (y-intercept) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_B([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [ , x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y0_const, x0_const	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.

Argument	Beskrivning
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*
p *Avg - diagramfunktion (page 469)*

LINEST_DF

LINEST_DF() returnerar den aggregerade frihetsgraden hos en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras i en**group by**-sats.

Syntax:

```
LINEST_DF (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*

LINEST_DF - diagramfunktion

LINEST_DF() returnerar de aggregerade frihetsgraderna för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_DF([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value [, y0_const [, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*

Argument	Beskrivning
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {,fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*
p *Avg - diagramfunktion (page 469)*

LINEST_F

Denna skriptfunktion returnerar det aggregerade F-värdet ($r^2/(1-r^2)$) hos en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras av en **group by**-sats.

Syntax:

```
LINEST_F (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*

LINEST_F - diagramfunktion

LINEST_F() returnerar det aggregerade F-värdet (r2/(1-r2)) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_F([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*
p *Avg - diagramfunktion (page 469)*

LINEST_M

LINEST_M() returnerar det aggregerade m-värdet (lutning) hos en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras i en **group by**-sats.

Syntax:

```
LINEST_M (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*

LINEST_M - diagramfunktion

LINEST_M() returnerar det aggregerade m-värdet (lutningen) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_M([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*
p *Avg - diagramfunktion (page 469)*

LINEST_R2

LINEST_R2() returnerar det aggregerade r^2-värdet (bestämningskoefficienten) av en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som representeras av parvisa tal i x-expression och y-expression itererat över ett antal poster som definieras av en **group by**-sats.

Syntax:

```
LINEST_R2 (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*

LINEST_R2 - diagramfunktion

LINEST_R2() returnerar det aggregerade r2-värdet (bestämningskoefficienten) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_R2([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*
p *Avg - diagramfunktion (page 469)*

LINEST_SEB

LINEST_SEB() returnerar det aggregerade standardfelet hos b-värdet i en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt definitionen av en **group by**-sats.

Syntax:

```
LINEST_SEB (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*

LINEST_SEB - diagramfunktion

LINEST_SEB() returnerar det aggregerade standardfelet för b-värdet av en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_SEB([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*
p *Avg - diagramfunktion (page 469)*

LINEST_SEM

LINEST_SEM() returnerar det aggregerade standardfelet hos m-värdet i en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt definitionen av en **group by**-sats.

Syntax:

```
LINEST_SEM (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*

LINEST_SEM - diagramfunktion

LINEST_SEM() returnerar det aggregerade standardfelet för m-värdet av en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_SEM([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*
p *Avg - diagramfunktion (page 469)*

LINEST_SEY

LINEST_SEY() returnerar det aggregerade standardfelet hos y-uppskattningen i en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt definitionen av en **group by**-sats.

Syntax:

```
LINEST_SEY (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*

LINEST_SEY - diagramfunktion

LINEST_SEY() returnerar det aggregerade standardfelet för y-estimatet av en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_SEY([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*
p *Avg - diagramfunktion (page 469)*

LINEST_SSREG

LINEST_SSREG returnerar den aggregerade restsumman av en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som representeras av parvisa tal i x-expression och y-expression itererat över ett antal poster som definieras av en **group by**-sats.

Syntax:

```
LINEST_SSREG (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*

LINEST_SSREG - diagramfunktion

LINEST_SSREG() returnerar den aggregerade regressionskvadratsumman av en linjär regression definierad genom ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_SSREG([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*
p *Avg - diagramfunktion (page 469)*

LINEST_SSRESID

LINEST_SSRESID() returnerar den aggregerade restsumman av en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som representeras av parvisa tal i x-expression och y-expression itererat över ett antal poster som definieras av en **group by**-sats.

Syntax:

```
LINEST_SSRESID (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*

LINEST_SSRESID - diagramfunktion

LINEST_SSRESID() returnerar den aggregerade residualkvadratsumman hos en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value,
x_value[, y0_const[, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionslinjen tvingas att passera en viss koordinat.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

p *Exempel på användning av linest-funktioner (page 528)*
p *Avg - diagramfunktion (page 469)*

Median

Median() returnerar den aggregerade medianen av värdena i uttrycket över ett antal poster som definieras av en **group by**-sats.

Syntax:

```
Median (expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Exempel: skriptuttryck som använder medianen

Exempel – skriptuttryck

Laddningsskript

Ladda följande inline-data och skriptuttryck i skriptredigeraren för det här exemplet.

```
Table 1:
Load RecNo() as RowNo, Letter, Number Inline
[Letter, Number
A,1
A,3
A,4
A,9
B,2
B,8
B,9];

Median:
LOAD Letter,
Median(Number) as MyMedian
Resident Table1 Group By Letter;
```

Skapa en visualisering

Skapa en tabellvisualisering i ett Qlik Sense-ark med **Letter** och **MyMedian** som dimensioner.

Resultat

Letter	MyMedian
A	3.5
B	8

Förklaring

Medianen betraktas som värdet ”i mitten” när de har sorterats från lägsta till högsta. Om datauppsättningen har ett jämnt antal värden kommer funktionen att returnera genomsnittet för de två mittenvärdena. I det här exemplet beräknas medianen för varje uppsättning med värden för **A** och **B**, som är 3,5 respektive 8.

Median - diagramfunktion

Median() returnerar medianvärdet för värdeintervallet som aggregeras i uttrycket, itererat över diagrammets dimensioner.

Syntax:

```
Median([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Exempel: diagramuttryck som använder medianen

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplet nedan.

```
Load RecNo() as RowNo, Letter, Number Inline
[Letter, Number
A,1
A,3
A,4
A,9
B,2
B,8
B,9];
```

Skapa en visualisering

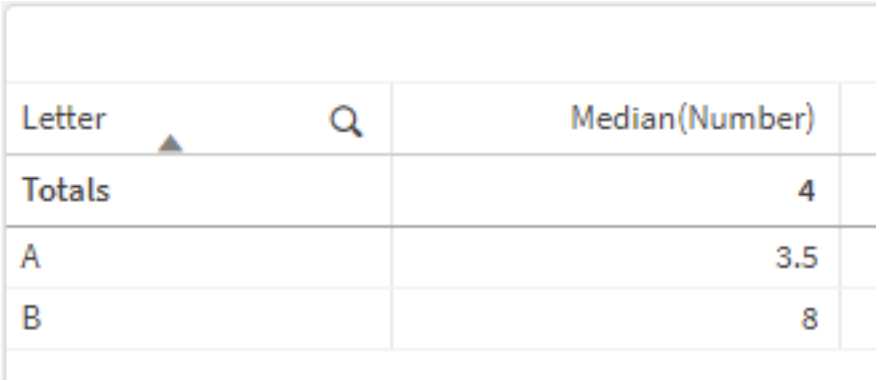
Skapa en tabellvisualisering i ett Qlik Sense-ark med **Letter** som dimension.

Diagramuttryck

Lägg till följande uttryck i tabellen som ett mått:

```
Median(Number)
```

Resultat



Letter	Median(Number)
Totals	4
A	3.5
B	8

Förklaring

Medianen betraktas som värdet ”i mitten” när de har sorterats från lägsta till högsta. Om datauppsättningen har ett jämnt antal värden kommer funktionen att returnera genomsnittet för de två mittenvärdena. I det här exemplet beräknas medianen för varje uppsättning med värden för **A** och **B**, som är 3,5 respektive 8.

Medianen för **Totalvärden** beräknas för alla värden, vilket är lika med 4.

Se även:

p *Avg - diagramfunktion (page 469)*

MutualInfo - diagramfunktion

MutualInfo beräknar ömsesidig information (MI) mellan två fält eller mellan aggregerade värden i **Aggr()**.

MutualInfo returnerar den aggregerade gemensamma informationen för två datauppsättningar. Detta möjliggör analys av de viktigaste drivkrafterna mellan ett fält och en potentiell drivkraft. Gemensam information mäter förhållandet mellan datauppsättningarna och aggregeras för (x,y)-parvärden, vilket upprepas för diagrammets dimensioner. Gemensam information har ett mått mellan 0 och 1 och kan formateras som ett percentilvärde. **MutualInfo** definieras av antingen urval eller ett set-uttryck.

MutualInfo tillåter olika typer av MI-analys:

- Gemensam MI: beräkna MI i ett drivande fält och ett målfält.
- Uppdelning av drivkraft efter värde: MI beräknas med individuella fältvärden i drivande fält och målfält.

- Funktionsurval: använd **MutualInfo** i ett rutnätsdiagram om du vill skapa en matris där alla fält jämförs med varandra baserat på MI.

MutualInfo visar inte nödvändigtvis på kausalitet mellan fält med gemensam information. Även om information är gemensam för två fält behöver de inte driva varandra lika mycket. Om du till exempel jämför glassförsäljning och utomhustemperatur, visar **MutualInfo** att de har gemensam information. Detta visar inte om det är utomhustemperaturen som driver glassförsäljningen, vilket är troligt, eller om det är glassförsäljningen som driver utomhustemperaturen, vilket är osannolikt.

Vid beräkning av ömsesidig information påverkar associationer överensstämmelsen mellan och frekvensen av värden från fält som kommer från olika tabeller.

Värdena som returneras för samma fält eller urval kan variera något. Detta beror på att varje **MutualInfo**-anrop verkar på slumpmässigt utvalda exempeldata och på **MutualInfo**-algoritmens inbyggda slumpmässighet.

MutualInfo kan tillämpas på funktionen **Aggr()**.

Syntax:

```
MutualInfo({SetExpression}] [DISTINCT] [TOTAL] field1, field2 , datatype [,
breakdownbyvalue [, samplesize ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
field1, field2	Uttrycken eller fälten som innehåller de två exempeluppsättningarna den gemensamma informationen ska mätas för.
datatype	Datatyperna som finns i målet och drivaren, 1 eller 'dd' för diskret:diskret 2 eller 'cc' för kontinuerlig:kontinuerlig 3 eller 'cd' för kontinuerlig:diskret 4 eller 'dc' för diskret:kontinuerlig Datatyper är inte skiftlägeskänsliga.
breakdownbyvalue	Ett statiskt värde som motsvarar ett värde i drivaren. Om det tillhandahålls beräknas MI-bidraget för det värdet. Du kan använda **ValueList()** eller **ValueLoop()**. Om **Null()** läggs till beräknas total MI för alla värden i drivaren. För uppdelning efter värde måste drivaren innehålla diskreta data.

Argument	Beskrivning
samplesize	Antal värden som ska ingå i exempeldata från mål och drivare. Val av exempeldata är slumpmässigt. **MutualInfo** kräver en exempelstorlek på minst 80. Som standard använder **MutualInfo** endast upp till 10 000 datapar som exempeldata eftersom **MutualInfo** kan kräva mycket resurser. Du kan specificera fler datapar i exempeldatastorleken. Om **MutualInfo** når tidsgränsen minskar du mängden exempeldata.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

Exempel på funktioner

Exempel	Resultat
`mutualinfo(Age, Salary, 1)`	Om en tabell innehåller dimensionen `Employee name` och måttet `mutualinfo(Age, Salary, 1)` är resultatet 0,99820986. Resultaten visas endast för totalcellen.
`mutualinfo (TOTAL Age, Salary, 1, null (), 81)`	Om du skapar en filterruta med dimensionen Gender och gör urval från den, ser du resultatet 0.99805677 när Female är valt och 0.99847373 om Male är valt. Detta beror på att valet utesluter alla resultat som inte hör till det andra värdet för Gender.
`mutualinfo (TOTAL Age, Gender, 1, ValueLoop (25,35))`	0.68196996. Alla värden som väljs från Gender ändrar detta till 0.

Exempel	Resultat
`mutualinfo({1} TOTAL Age, Salary, 1, null ())`	0.99820986. Detta är oberoende av urval. Set-uttrycket {1} bortser från alla urval och dimensioner.

Data som används i exempel:

```
Salary:

LOAD * inline [

"Employee name"|Age|Gender|Salary

Aiden Charles|20|Male|25000

Ann Lindquist|69|Female|58000

Anna Johansen|37|Female|36000

Anna Karlsson|42|Female|23000

Antonio Garcia|20|Male|61000

Benjamin Smith|42|Male|27000

Bill Yang|49|Male|50000

Binh Protzmann|69|Male|21000

Bob Park|51|Male|54000

Brenda Davies|25|Male|32000

Celine Gagnon|48|Female|38000

Cezar Sandu|50|Male|46000

Charles Ingvar Jönsson|27|Male|58000

Charlotte Edberg|45|Female|56000

Cindy Lynn|69|Female|28000

Clark Wayne|63|Male|31000

Daroush Ferrara|31|Male|29000

David Cooper|37|Male|64000

David Leg|58|Male|57000

Eunice Goldblum|31|Female|32000
```

```
Freddy Halvorsen|25|Male|26000

Gauri Indu|36|Female|46000

George van Zaant|59|Male|47000

Glenn Brown|58|Male|40000

Harry Jones|38|Male|40000

Helen Brolin|52|Female|66000

Hiroshi Ito|24|Male|42000

Ian Underwood|40|Male|45000

Ingrid Hendrix|63|Female|27000

Ira Baumel|39|Female|39000

Jackie Kingsley|23|Female|28000

Jennica Williams|36|Female|48000

Jerry Tessel|31|Male|57000

Jim Bond|50|Male|58000

Joan Callins|60|Female|65000

Joan Cleaves|25|Female|61000

Joe Cheng|61|Male|41000

John Doe|36|Male|59000

John Lemon|43|Male|21000

Karen Helmkey|54|Female|25000

Karl Berger|38|Male|68000

Karl Straubaum|30|Male|40000

Kaya Alpan|32|Female|60000

Kenneth Finley|21|Male|25000

Leif Shine|63|Male|70000

Lennart Skoglund|63|Male|24000

Leona Korhonen|46|Female|50000
```

```
Lina André|50|Female|65000

Louis Presley|29|Male|36000

Luke Langston|50|Male|63000

Marcus Salvatori|31|Male|46000

Marie Simon|57|Female|23000

Mario Rossi|39|Male|62000

Markus Danzig|26|Male|48000

Michael Carlen|21|Male|45000

Michelle Tyson|44|Female|69000

Mike Ashkenaz|45|Male|68000

Miro Ito|40|Male|39000

Nina Mihn|62|Female|57000

Olivia Nguyen|35|Female|51000

Olivier Simenon|44|Male|31000

Östen Ärlig|68|Male|57000

Pamala Garcia|69|Female|29000

Paolo Romano|34|Male|45000

Pat Taylor|67|Female|69000

Paul Dupont|34|Male|38000

Peter Smith|56|Male|53000

Pierre Clouseau|21|Male|37000

Preben Jørgensen|35|Male|38000

Rey Jones|65|Female|20000

Ricardo Gucci|55|Male|65000

Richard Ranieri|30|Male|64000

Rob Carsson|46|Male|54000

Rolf Wesenlund|25|Male|51000
```

```
Ronaldo Costa|64|Male|39000

Sabrina Richards|57|Female|40000

Sato Hiromu|35|Male|21000

Sehoon Daw|57|Male|24000

Stefan Lind|67|Male|35000

Steve Cioazzi|58|Male|23000

Sunil Gupta|45|Male|40000

Sven Svensson|45|Male|55000

Tom Lindwall|46|Male|24000

Tomas Nilsson|27|Male|22000

Trinity Rizzo|52|Female|48000

Vanessa Lambert|54|Female|27000

] (delimiter is '|');
```

Skew

Skew() returnerar skevheten i uttryck över ett antal poster enligt vad som definierats i en **group by**-sats.

Syntax:

```
Skew([ distinct] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
DISTINCT	Om predikatet **distinct** förekommer framför uttrycket kommer alla dubbletter att ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. Bygg sedan en rak tabell med `Type` och *MySkew* som dimensioner.

Resulterande data

Exempel	Resultat
Table1: Crosstable (Type, Value) Load recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|'); Skew1: LOAD Type, Skew(Value) as MySkew Resident Table1 Group By Type;	Resultaten av Skew()-beräkningen är: • Type är MySkew • Comparison är 0.86414768 • Observation är 0.32625351

Skew - diagramfunktion

Skew() returnerar den aggregerade skevheten av uttrycket eller fältet itererat över diagrammets dimensioner.

Syntax:

```
Skew([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Argument	Beskrivning
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {,fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Bygg sedan en rak tabell med `Type` som dimension och `Skew (Value)` som mått.

`Totals` ska vara aktiverat i tabellegenskaperna.

Exempel	Resultat
Table1: Crosstable (Type, Value) Load recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|');	Resultaten av Skew(Value)-beräkningen är: • Totalär 0.23522195 • Comparison är 0.86414768 • Observation är 0.32625351

Se även:

p *Avg - diagramfunktion (page 469)*

Stdev

Stdev() returnerar värdenas standardavvikelse i uttrycket över ett antal poster som definieras av en **group by**-sats.

Syntax:

```
Stdev([distinct] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket kommer alla dubbletter att ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Bygg sedan en rak tabell med `Type` och `MyStdev`som dimensioner.

Resulterande data

Exempel	Resultat

`Table1:`
`Crosstable (Type, Value)`
`Load recno() as ID, * inline [`
`Observation|Comparison`
`35|2`
`40|27`
`12|38`
`15|31`
`21|1`
`14|19`
`46|1`
`10|34`
`28|3`
`48|1`
`16|2`
`30|3`
`32|2`
`48|1`
`31|2`
`22|1`
`12|3`
`39|29`
`19|37`
`25|2 ] (delimiter is '|');`
 	
`Stdev1:`
`LOAD Type,`
`Stdev(Value) as MyStdev`
`Resident Table1 Group By Type;`
	Resultaten av Stdev()-beräkningen är:
	• `Type` är `MyStdev`
	• `Comparison` är 14.61245
	• `Observation` är 12.507997

Stdev - diagramfunktion

Stdev() finner standardavvikelsen för dataintervallet som aggregerats i uttrycket eller fältet itererat över diagrammets dimensioner.

Syntax:

```
Stdev([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Bygg sedan en rak tabell med `Type` som dimension och `Stdev (Value)` som mått.

`Totals` ska vara aktiverat i tabellegenskaperna.

Exempel	Resultat

`Stdev(Value)`
`Table1:`
`Crosstable (Type, Value)`
`Load recno() as ID, * inline [`
`Observation|Comparison`
`35|2`
`40|27`
`12|38`
`15|31`
`21|1`
`14|19`
`46|1`
`10|34`
`28|3`
`48|1`
`16|2`
`30|3`
`32|2`
`48|1`
`31|2`
`22|1`
`12|3`
`39|29`
`19|37`
`25|2 ] (delimiter is '|');`
	Resultaten av Stdev(Value)-beräkningen är:
	• `Total`är 15.47529
	• `Comparison` är 14.61245
	• `Observation` är 12.507997

Se även:

p *Avg - diagramfunktion (page 469)*
p *STEYX - diagramfunktion (page 526)*

Sterr

Sterr() returnerar det aggregerade standardfelet (stdev/sqrt(n)) för en serie värden, representerade av uttryck över ett antal poster enligt vad som definierats i en **group by**-sats.

Syntax:

```
Sterr ([distinct] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket kommer alla dubbletter att ignoreras.

Begränsningar:

Textvärden, NULL-värden samt saknade värden ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

Resulterande data

Exempel	Resultat

`Table1:`
`Crosstable (Type, Value)`
`Load recno() as ID, * inline [`
`Observation|Comparison`
`35|2`
`40|27`
`12|38`
`15|31`
`21|1`
`14|19`
`46|1`
`10|34`
`28|3`
`48|1`
`16|2`
`30|3`
`32|2`
`48|1`
`31|2`
`22|1`
`12|3`
`39|29`
`19|37`
`25|2 ] (delimiter is '|');`
 	
`Sterr1:`
`LOAD Type,`
`Sterr(Value) as MySterr`
`Resident Table1 Group By Type;`
	I en tabell med dimensionerna `Type` och `MySterr` blir resultaten av Sterr()-beräkningen i dataladdningsskriptet:
	
	`Type MySterr`
	
	`Comparison 3.2674431`
	
	`Observation 2.7968733`

Sterr - diagramfunktion

Sterr() finner värdet av standardfelet av medelvärdet (stdev/sqrt(n)) för värdeserien som aggregerats i uttrycket itererat över diagrammets dimensioner.

Syntax:

```
Sterr([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden samt saknade värden ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Bygg sedan en rak tabell med `Type` som dimension och `Sterr (Value)` som mått.

`Totals` ska vara aktiverat i tabellegenskaperna.

Exempel	Resultat
Table1: Crosstable (Type, Value) Load recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|');	Resultaten av Sterr(Value)-beräkningen är: • Totalär 2.4468583 • Comparison är 3.2674431 • Observation är 2.7968733

Se även:

p *Avg - diagramfunktion (page 469)*
p *STEYX - diagramfunktion (page 526)*

STEYX

STEYX() returnerar det aggregerade standardfelet hos det predikterade y-värdet för varje x-värde i regressionen för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definierats i en **group by**-sats.

Syntax:

```
STEYX (y_value, x_value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

Resulterande data

Exempel	Resultat
Trend: Load *, 1 as Grp; LOAD * inline [Month|KnownY|KnownX Jan|2|6 Feb|3|5 Mar|9|11 Apr|6|7 May|8|5 Jun|7|4 Jul|5|5 Aug|10|8 Sep|9|10 Oct|12|14 Nov|15|17 Dec|14|16] (delimiter is '|'); STEYX1: LOAD Grp, STEYX(KnownY, KnownX) as MySTEYX Resident Trend Group By Grp;	I en tabell med dimensionen MySTEYX är resultatet i beräkningen STEYX() i dataladdningsskriptet 2.0714764.

STEYX - diagramfunktion

STEYX() returnerar det aggregerade standardfelet vid prediktering av y-värden för varje x-värde i en linjär regression som ges av en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **y_value** och **x_value**.

Syntax:

```
STEYX([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller intervallet av kända y-värden som ska mätas.
x_value	Uttrycket eller fältet som innehåller intervallet av kända x-värden som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {,fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Bygg sedan en enkel tabell med `KnownY` och `KnownX` som dimension och `Steyx(KnownY,KnownX)` som mått.

`Totals` ska vara aktiverat i tabellegenskaperna.

Exempel	Resultat
Trend: LOAD * inline [Month|KnownY|KnownX Jan|2|6 Feb|3|5 Mar|9|11 Apr|6|7 May|8|5 Jun|7|4 Jul|5|5 Aug|10|8 Sep|9|10 Oct|12|14 Nov|15|17 Dec|14|16] (delimiter is '|');	Resultatet av STEYX(KnownY,KnownX)-beräkningen är 2,071 (om nummerformatering är inställt på tre decimaler.)

Se även:

p *Avg - diagramfunktion (page 469)*
p *Sterr - diagramfunktion (page 522)*

Exempel på användning av linest-funktioner

Funktioner av typen linest används för att hitta värden associerade med analys av linjär regression. I det här avsnittet beskrivs hur du kan bygga visualiseringar med hjälp av sticksprovsdata för att hitta värdena för linest-funktionerna som finns i Qlik Sense. linest-funktionerna kan användas i dataladdningsskriptet och diagramuttryck.

Se de individuella avsnitten för linest-diagram- och skriptfunktionerna för beskrivningar av syntax och argument.

Uttryck för data och skript används i exemplen

Ladda följande inline-data och skriptuttryck i skriptredigeraren för linest()-exemplen nedan.

```
T1:
LOAD *, 1 as Grp;
LOAD * inline [
X|Y
1|0
2|1
3|3
4|8
5|14
6|20
7|0
8|50
9|25
10|60
11|38
12|19
13|26
14|143
15|98
16|27
17|59
18|78
19|158
20|279 ] (delimiter is '|');

R1:
LOAD
Grp,
linest_B(Y,X) as Linest_B,
linest_DF(Y,X) as Linest_DF,
linest_F(Y,X) as Linest_F,
linest_M(Y,X) as Linest_M,
linest_R2(Y,X) as Linest_R2,
linest_SEB(Y,X,1,1) as Linest_SEB,
linest_SEM(Y,X) as Linest_SEM,
linest_SEY(Y,X) as Linest_SEY,
linest_SSREG(Y,X) as Linest_SSREG,
linest_SSRESID(Y,X) as Linest_SSRESID
resident T1 group by Grp;
```

Exempel: skriptuttryck som använder linest

Exempel: skriptuttryck

Skapa en visualisering från beräkningar av dataladdningsskript

Skapa en tabellvisualisering i ett Qlik Sense-ark med följande fält som kolumner:

- Linest_B
- Linest_DF
- Linest_F

- Linest_M
- Linest_R2
- Linest_SEB
- Linest_SEM
- Linest_SEY
- Linest_SSREG
- Linest_SSRESID

Resultat

Tabellen med resultat från linest-beräkningarna som gjordes för dataladdningsskriptet ska se ut så här:

Resultattabell

Linest_B	Linest_DF	Linest_F	Linest_M	Linest_R2	Linest_SEB
-35.047	18	20.788	8.605	0.536	22.607

Resultattabell

Linest_SEM	Linest_SEY	Linest_SSREG	Linest_SSRESID
1.887	48.666	49235.014	42631.186

Exempel 2: diagramuttryck som använder linest

Exempel: diagramuttryck

Skapa en tabellvisualisering i ett Qlik Sense-ark med följande fält som dimensioner:

```
ValueList('Linest_b', 'Linest_df','Linest_f', 'Linest_m','Linest_r2','Linest_SEB','Linest_
SEM','Linest_SEY','Linest_SSREG','Linest_SSRESID')
```

Nu använder uttrycket funktionen för syntetiska dimensioner för att skapa etiketter för dimensionerna med namnen på linest-funktionerna. Du kan ändra etiketten till **Linest functions** för att spara plats.

Lägg till följande uttryck i tabellen som ett mått:

```
Pick(Match(ValueList('Linest_b', 'Linest_df','Linest_f', 'Linest_m','Linest_r2','Linest_
SEB','Linest_SEM','Linest_SEY','Linest_SSREG','Linest_SSRESID'),'Linest_b', 'Linest_
df','Linest_f', 'Linest_m','Linest_r2','Linest_SEB','Linest_SEM','Linest_SEY','Linest_
SSREG','Linest_SSRESID'),Linest_b(Y,X),Linest_df(Y,X),Linest_f(Y,X),Linest_m(Y,X),Linest_r2
(Y,X),Linest_SEB(Y,X,1,1),Linest_SEM(Y,X),Linest_SEY(Y,X),Linest_SSREG(Y,X),Linest_SSRESID
(Y,X) )
```

Detta uttryck visar värdet av resultatet för varje linest-funktion mot motsvarande namn i den syntetiska dimensionen. Resultatet för `Linest_b(Y,X)` visas jämte **linest_b** och så vidare.

Resultat

Resultattabell

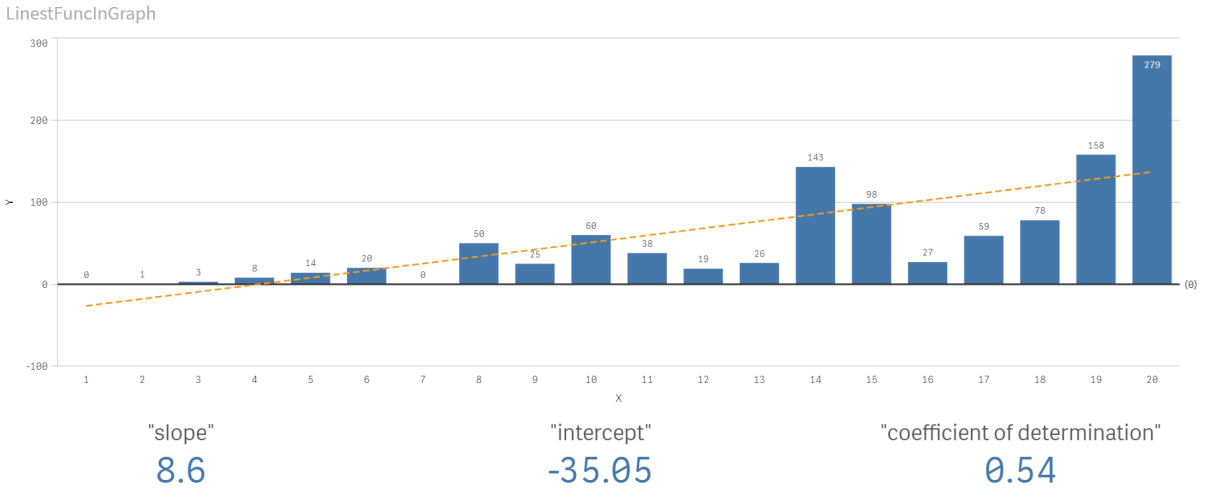
Linest functions	Linest function results
Linest_b	-35.047
Linest_df	18
Linest_f	20.788
Linest_m	8.605
Linest_r2	0.536
Linest_SEB	22.607
Linest_SEM	1.887
Linest_SEY	48.666
Linest_SSREG	49235.014
Linest_SSRESID	42631.186

Exempel 3: diagramuttryck som använder linest

Exempel: diagramuttryck

1. Skapa en visualisering med stapeldiagram i ett Qlik Sense-ark med **X** som dimension och **Y** som mått.
2. Lägg till en linjär trendlinje i Y-måttet.
3. Lägg till en KPI-visualisering på arket.
    1. Lägg till *lutning* som en etikett för KPI.
    2. Lägg till `sum(Linest_M)` som ett uttryck för KPI.
4. Lägg till en andra KPI-visualisering på arket.
    1. Lägg till *skärningspunkt* som en etikett för KPI.
    2. Lägg till `Sum(Linest_B)` som ett uttryck för KPI.
5. Lägg till en tredje KPI-visualisering på arket.
    1. Lägg till *bestämningskoefficient* som en etikett för KPI.
    2. Lägg till `Sum(Linest_R2)` som ett uttryck för KPI.

Resultat



"slope"	"intercept"	"coefficient of determination"
8.6	-35.05	0.54

Förklaring

Stapeldiagrammet visar hur X- och Y-data ritas ut. Relevanta linest()-funktioner ger värden till den linjära regressionsekvationen som trendlinjen baseras på, nämligen y = m * x + b. Ekvationen använder minsta kvadratmetoden för att beräkna en rak linje (trendlinje) genom att returnera en matris som beskriver en linje som passar bäst för data.

KPI:er visar resultaten för linest()-funktionerna **sum(Linest_M)** för lutning och **sum(Linest_B)** för Y-skärningspunkten, som är variabler i den linjära regressionsekvationen och motsvarande aggregerade R2-värde för bestämningskoefficienten.

Statistiska testfunktioner

Statistiska testfunktioner kan användas i både dataladdningsskriptet och diagramuttryck, men syntaxen är olika.

Chi-2-testfunktioner

Används allmänt i studien av kvalitativa variabler. Man kan jämföra observerade frekvenser i en enkelriktad frekvenstabell med förväntade frekvenser, eller studera sambandet mellan två variabler i en reservtabell.

T-testfunktioner

T-testfunktioner används för statistisk undersökning av två populationsmedelvärden. Ett t-test med två stickprov undersöker om två stickprov skiljer sig åt och används vanligtvis när två normalfördelningar har okända varianser och när en liten urvalsstorlek används vid ett experiment.

Z-testfunktioner

En statistisk undersökning av två populationsmedelvärden. Ett z-test med två stickprov som undersöker om två stickprov skiljer sig åt och används vanligtvis när två normalfördelningar har kända varianser och när ett experiment använder en stor urvalsstorlek.

Chi2-testfunktioner

Används allmänt i studien av kvalitativa variabler. Man kan jämföra observerade frekvenser i en enkelriktad frekvenstabell med förväntade frekvenser, eller studera sambandet mellan två variabler i en reservtabell.Chi-squared test functions are used to determine whether there is a statistically significant difference between the expected frequencies and the observed frequencies in one or more groups. Often a histogram is used, and the different bins are compared to an expected distribution.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Chi2Test_chi2
Chi2Test_chi2() returnerar det aggregerade värdet av ett chi^2-test för en eller två värdeserier.

```
Chi2Test_chi2(col, row, actual_value[, expected_value])
```

Chi2Test_df
Chi2Test_df() returnerar det aggregerade df-värdet (frihetsgrader) av ett chi^2-test för en eller två värdeserier.

```
Chi2Test_df(col, row, actual_value[, expected_value])
```

Chi2Test_p
Chi2Test_p() returnerar det aggregerade p-värdet (signifikansen) av ett chi^2-test för en eller två värdeserier.

```
Chi2Test_p - diagramfunktion(col, row, actual_value[, expected_value])
```

Se även:

p *T-testfunktioner (page 536)*
p *Z-testfunktioner (page 569)*

Chi2Test_chi2

Chi2Test_chi2() returnerar det aggregerade värdet av ett chi^2-test för en eller två värdeserier.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Alla Qlik Sense chi^2-testfunktioner har samma argument.

Syntax:

```
Chi2Test_chi2(col, row, actual_value[, expected_value])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
col, row	Den angivna kolumnen och raden i den matris av värden som ska testas.
actual_value	Det observerade värdet av data vid angiven **col** och **row**.
expected_value	Det väntade värdet för fördelningen vid angiven **col** och **row**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
Chi2Test_chi2( Grp, Grade, Count )

Chi2Test_chi2( Gender, Description, Observed, Expected )
```

Se även:

p *Exempel på användning av chi2-test-funktioner i diagram (page 584)*
p *Exempel på användning av chi2-test-funktioner i dataladdningsskriptet (page 588)*

Chi2Test_df

Chi2Test_df() returnerar det aggregerade df-värdet (frihetsgrader) av ett chi^2-test för en eller två värdeserier.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Alla Qlik Sense chi^2-testfunktioner har samma argument.

Syntax:

```
Chi2Test_df(col, row, actual_value[, expected_value])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
col, row	Den angivna kolumnen och raden i den matris av värden som ska testas.
actual_value	Det observerade värdet av data vid angiven **col** och **row**.
expected_value	Det väntade värdet för fördelningen vid angiven **col** och **row**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
Chi2Test_df( Grp, Grade, Count )

Chi2Test_df( Gender, Description, Observed, Expected )
```

Se även:

p *Exempel på användning av chi2-test-funktioner i diagram (page 584)*
p *Exempel på användning av chi2-test-funktioner i dataladdningsskriptet (page 588)*

Chi2Test_p - diagramfunktion

Chi2Test_p() returnerar det aggregerade p-värdet (signifikansen) av ett chi^2-test för en eller två värdeserier. Testet kan göras antingen på värdena i **actual_value**, för att testa variationer inom den angivna **col**- och **row**-matrisen, eller genom att jämföra värden i **actual_value** med motsvarande värden i **expected_value**, om så anges.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

> *Alla Qlik Sense chi^2-testfunktioner har samma argument.*

Syntax:

```
Chi2Test_p(col, row, actual_value[, expected_value])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
col, row	Den angivna kolumnen och raden i den matris av värden som ska testas.
actual_value	Det observerade värdet av data vid angiven **col** och **row**.
expected_value	Det väntade värdet för fördelningen vid angiven **col** och **row**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
Chi2Test_p( Grp, Grade, Count )
```

```
Chi2Test_p( Gender, Description, Observed, Expected )
```

Se även:

p *Exempel på användning av chi2-test-funktioner i diagram (page 584)*
p *Exempel på användning av chi2-test-funktioner i dataladdningsskriptet (page 588)*

T-testfunktioner

T-testfunktioner används för statistisk undersökning av två populationsmedelvärden. Ett t-test med två stickprov undersöker om två stickprov skiljer sig åt och används vanligtvis när två normalfördelningar har okända varianser och när en liten urvalsstorlek används vid ett experiment.

I följande avsnitt grupperas de statistiska t-testfunktionerna enligt exempeltestet som gäller för funktionen.

Skapa en typisk t-test-rapport (page 590)

T-test med två oberoende stickprov

Följande funktioner gäller student-t-test med två oberoende stickprov.

ttest_conf

TTest_conf returnerar det aggregerade värdet av konfidensintervallet för ett t-test för två oberoende stickprov.

```
TTest_conf returnerar det aggregerade värdet av konfidensintervallet för ett t-test för två oberoende stickprov. ( grp, value [, sig[, eq_var]])
```

ttest_df

TTest_df() returnerar det aggregerade värdet (frihetsgrader) av ett t-test för två oberoende värdeserier.

```
TTest_df() returnerar det aggregerade värdet (frihetsgrader) av ett t-test
för två oberoende värdeserier. (grp, value [, eq_var)
```

ttest_dif

TTest_dif() är en numerisk funktion som returnerar den aggregerade skillnaden i medelvärde för ett t-test med två oberoende värdeserier.

```
TTest_dif() är en numerisk funktion som returnerar den aggregerade skillnaden
i medelvärde för ett t-test med två oberoende värdeserier. (grp, value)
```

ttest_lower

TTest_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

```
TTest_lower() returnerar det aggregerade värdet för konfidensintervallets
nedre gräns för två oberoende värdeserier. (grp, value [, sig[, eq_var]])
```

ttest_sig

TTest_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för två oberoende värdeserier.

```
TTest_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-
test för två oberoende värdeserier. (grp, value [, eq_var])
```

ttest_sterr

TTest_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för två oberoende värdeserier.

```
TTest_sterr() returnerar det aggregerade standardfelet för skillnaden i
medelvärde för ett t-test för två oberoende värdeserier. (grp, value [, eq_
var])
```

ttest_t

TTest_t() returnerar det aggregerade t-värdet för två oberoende värdeserier.

```
TTest_t() returnerar det aggregerade t-värdet för två oberoende värdeserier.
(grp, value [, eq_var])
```

ttest_upper

TTest_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

```
TTest_upper() returnerar det aggregerade värdet för konfidensintervallets
övre gräns för två oberoende värdeserier. (grp, value [, sig [, eq_var]])
```

T-test med två oberoende viktade stickprov

Följande funktioner gäller student-t-test med två oberoende stickprov där indataserien anges i viktat tvåkolumnsformat.

ttestw_conf

TTestw_conf() returnerar det aggregerade t-värdet för två oberoende värdeserier.

```
TTestw_conf() returnerar det aggregerade t-värdet för två oberoende
värdeserier. (weight, grp, value [, sig[, eq_var]])
```

ttestw_df

TTestw_df() returnerar det aggregerade df-värdet (frihetsgrader) av ett t-test för två oberoende värdeserier.

```
TTestw_df() returnerar det aggregerade df-värdet (frihetsgrader) av ett t-
test för två oberoende värdeserier. (weight, grp, value [, eq_var])
```

ttestw_dif

TTestw_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-test med två oberoende värdeserier.

```
TTestw_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-
test med två oberoende värdeserier. ( weight, grp, value)
```

ttestw_lower

TTestw_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

```
TTestw_lower() returnerar det aggregerade värdet för konfidensintervallets
nedre gräns för två oberoende värdeserier. (weight, grp, value [, sig[, eq_
var]])
```

ttestw_sig

TTestw_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för två oberoende värdeserier.

```
TTestw_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-
test för två oberoende värdeserier. ( weight, grp, value [, eq_var])
```

ttestw_sterr

TTestw_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för två oberoende värdeserier.

```
TTestw_sterr() returnerar det aggregerade standardfelet för skillnaden i
medelvärde för ett t-test för två oberoende värdeserier. (weight, grp, value
[, eq_var])
```

ttestw_t

TTestw_t() returnerar det aggregerade t-värdet för två oberoende värdeserier.

```
TTestw_t() returnerar det aggregerade t-värdet för två oberoende värdeserier.
(weight, grp, value [, eq_var])
```

ttestw_upper

TTestw_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

```
TTestw_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier. (weight, grp, value [, sig [, eq_var]])
```

T-test med ett stickprov

Följande funktioner gäller student-t-test med ett stickprov.

ttest1_conf

TTest1_conf() returnerar det aggregerade konfidensintervallvärdet för en värdeserie.

```
TTest1_conf() returnerar det aggregerade konfidensintervallvärdet för en värdeserie. (value [, sig])
```

ttest1_df

TTest1_df() returnerar det aggregerade df-värdet (frihetsgrader) för ett t-test för en värdeserie.

```
TTest1_df() returnerar det aggregerade df-värdet (frihetsgrader) för ett t-test för en värdeserie. (value)
```

ttest1_dif

TTest1_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-test för en värdeserie.

```
TTest1_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-test för en värdeserie. (value)
```

ttest1_lower

TTest1_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för en värdeserie.

```
TTest1_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för en värdeserie. (value [, sig])
```

ttest1_sig

TTest1_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för en värdeserie.

```
TTest1_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för en värdeserie. (value)
```

ttest1_sterr

TTest1_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för en värdeserie.

```
TTest1_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för en värdeserie. (value)
```

ttest1_t

TTest1_t() returnerar det aggregerade t-värdet för en värdeserie.

TTest1_t() returnerar det aggregerade t-värdet för en värdeserie. (value)

ttest1_upper

TTest1_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för en värdeserie.

TTest1_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för en värdeserie. (value [, sig])

T-test med ett viktat stickprov

Följande funktioner gäller student-t-test med ett stickprov där indataserien anges i viktat tvåkolumnsformat.

ttest1w_conf

TTest1w_conf() är en **numerisk** funktion som returnerar det aggregerade konfidensintervallvärdet för en värdeserie.

TTest1w_conf() är en numerisk funktion som returnerar det aggregerade konfidensintervallvärdet för en värdeserie. (weight, value [, sig])

ttest1w_df

TTest1w_df() returnerar det aggregerade df-värdet (frihetsgrader) för ett t-test för en värdeserie.

TTest1w_df() returnerar det aggregerade df-värdet (frihetsgrader) för ett t-test för en värdeserie. (weight, value)

ttest1w_dif

TTest1w_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-test för en värdeserie.

TTest1w_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-test för en värdeserie. (weight, value)

ttest1w_lower

TTest1w_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för en värdeserie.

TTest1w_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för en värdeserie. (weight, value [, sig])

ttest1w_sig

TTest1w_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för en värdeserie.

TTest1w_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för en värdeserie. (weight, value)

ttest1w_sterr

TTest1w_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för en värdeserie.

TTest1w_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för en värdeserie. (weight, value)

ttest1w_t

TTest1w_t() returnerar det aggregerade t-värdet för en värdeserie.

```
TTest1w_t() returnerar det aggregerade t-värdet för en värdeserie. ( weight, value)
```

ttest1w_upper

TTest1w_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för en värdeserie.

```
TTest1w_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för en värdeserie. (weight, value [, sig])
```

TTest_conf

TTest_conf returnerar det aggregerade värdet av konfidensintervallet för ett t-test för två oberoende stickprov.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_conf ( grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_conf( Group, Value )

TTest_conf( Group, Value, Sig, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest_df

TTest_df() returnerar det aggregerade värdet (frihetsgrader) av ett t-test för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_df (grp, value [, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_df( Group, Value )

TTest_df( Group, Value, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest_dif

TTest_dif() är en numerisk funktion som returnerar den aggregerade skillnaden i medelvärde för ett t-test med två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_dif (grp, value [, eq_var] )
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_dif( Group, Value )
```

```
TTest_dif( Group, Value, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest_lower

TTest_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_lower (grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_lower( Group, Value )
```

```
TTest_lower( Group, Value, Sig, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest_sig

TTest_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_sig (grp, value [, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_sig( Group, Value )

TTest_sig( Group, Value, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest_sterr

TTest_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_sterr (grp, value [, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_sterr( Group, Value )
```

```
TTest_sterr( Group, Value, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest_t

TTest_t() returnerar det aggregerade t-värdet för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_t(grp, value[, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_t( Group, Value, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest_upper

TTest_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_upper (grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_upper( Group, Value )
```

```
TTest_upper( Group, Value, sig, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTestw_conf

TTestw_conf() returnerar det aggregerade t-värdet för två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTestw_conf (weight, grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_conf( Weight, Group, Value )
```

```
TTestw_conf( Weight, Group, Value, sig, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTestw_df

TTestw_df() returnerar det aggregerade df-värdet (frihetsgrader) av ett t-test för två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTestw_df (weight, grp, value [, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_df( Weight, Group, Value )
```

```
TTestw_df( Weight, Group, Value, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTestw_dif

TTestw_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-test med två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTestw_dif (weight, grp, value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_dif( Weight, Group, Value )
```

```
TTestw_dif( Weight, Group, Value, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTestw_lower

TTestw_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTestw_lower (weight, grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_lower( Weight, Group, Value )
```

```
TTestw_lower( Weight, Group, Value, sig, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTestw_sig

TTestw_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTestw_sig ( weight, grp, value [, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_sig( Weight, Group, Value )
```

```
TTestw_sig( Weight, Group, Value, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTestw_sterr

TTestw_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTestw_sterr (weight, grp, value [, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_sterr( Weight, Group, Value )
```

```
TTestw_sterr( Weight, Group, Value, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTestw_t

TTestw_t() returnerar det aggregerade t-värdet för två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ttestw_t (weight, grp, value [, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_t( Weight, Group, Value )
```

```
TTestw_t( Weight, Group, Value, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTestw_upper

TTestw_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTestw_upper (weight, grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_upper( Weight, Group, Value )
```

```
TTestw_upper( Weight, Group, Value, sig, false )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1_conf

TTest1_conf() returnerar det aggregerade konfidensintervallvärdet för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_conf (value [, sig ])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_conf( Value )

TTest1_conf( Value, 0.005 )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1_df

TTest1_df() returnerar det aggregerade df-värdet (frihetsgrader) för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_df (value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_df( Value )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1_dif

TTest1_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_dif (value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_dif( Value )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1_lower

TTest1_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_lower (value [, sig])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_lower( Value )

TTest1_lower( Value, 0.005 )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1_sig

TTest1_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_sig (value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_sig( Value )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1_sterr

TTest1_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_sterr (value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_sterr( Value )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1_t

TTest1_t() returnerar det aggregerade t-värdet för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_t (value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_t( Value )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1_upper

TTest1_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_upper (value [, sig])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_upper( Value )
```

```
TTest1_upper( Value, 0.005 )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1w_conf

TTest1w_conf() är en **numerisk** funktion som returnerar det aggregerade konfidensintervallvärdet för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_conf (weight, value [, sig ])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_conf( Weight, Value )

TTest1w_conf( Weight, Value, 0.005 )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1w_df

TTest1w_df() returnerar det aggregerade df-värdet (frihetsgrader) för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_df (weight, value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_df( Weight, Value )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1w_dif

TTest1w_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_dif (weight, value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_dif( Weight, Value )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1w_lower

TTest1w_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_lower (weight, value [, sig ])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_lower( Weight, Value )
```

```
TTest1w_lower( Weight, Value, 0.005 )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1w_sig

TTest1w_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_sig (weight, value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_sig( Weight, Value )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1w_sterr

TTest1w_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_sterr (weight, value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_sterr( Weight, Value )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1w_t

TTest1w_t() returnerar det aggregerade t-värdet för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_t ( weight, value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_t( Weight, Value )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

TTest1w_upper

TTest1w_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_upper (weight, value [, sig])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.

Argument	Beskrivning
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_upper( Weight, Value )

TTest1w_upper( Weight, Value, 0.005 )
```

Se även:

p *Skapa en typisk t-test-rapport (page 590)*

Z-testfunktioner

En statistisk undersökning av två populationsmedelvärden. Ett z-test med två stickprov som undersöker om två stickprov skiljer sig åt och används vanligtvis när två normalfördelningar har kända varianser och när ett experiment använder en stor urvalsstorlek.

Z-testens statistiska testfunktioner grupperas enligt typ av indataserier som gäller för funktionen.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Exempel på användning av z-test-funktioner (page 594)

Funktioner i enkolumnsformat

Följande funktioner gäller för z-test med enkla indataserier.

ztest_conf

ZTest_conf() returnerar det aggregerade z-värdet för en värdeserie.

```
ZTest_conf() returnerar det aggregerade z-värdet för en värdeserie. (value [, sigma [, sig ])
```

ztest_dif

ZTest_dif() returnerar den aggregerade skillnaden i medelvärde för ett z-test för en värdeserie.

ZTest_dif() returnerar den aggregerade skillnaden i medelvärde för ett z-test för en värdeserie. (value [, sigma])

ztest_sig

ZTest_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett z-test för en värdeserie.

ZTest_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett z-test för en värdeserie. (value [, sigma])

ztest_sterr

ZTest_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett z-test för en värdeserie.

ZTest_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett z-test för en värdeserie. (value [, sigma])

ztest_z

ZTest_z() returnerar det aggregerade z-värdet för en värdeserie.

ZTest_z() returnerar det aggregerade z-värdet för en värdeserie. (value [, sigma])

ztest_lower

ZTest_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

ZTest_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier. (grp, value [, sig [, eq_var]])

ztest_upper

ZTest_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

ZTest_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier. (grp, value [, sig [, eq_var]])

Funktioner i viktat tvåkolumnsformat

Följande funktioner gäller z-test där indataserien anges i viktat tvåkolumnsformat.

ztestw_conf

ZTestw_conf() returnerar det aggregerade z-konfidensintervallvärdet för en värdeserie.

ZTestw_conf() returnerar det aggregerade z-konfidensintervallvärdet för en värdeserie. (weight, value [, sigma [, sig]])

ztestw_dif

ZTestw_dif() returnerar den aggregerade skillnaden i medelvärde för ett z-test för en värdeserie.

ZTestw_dif() returnerar den aggregerade skillnaden i medelvärde för ett z-test för en värdeserie. (weight, value [, sigma])

ztestw_lower

ZTestw_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

```
ZTestw_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier. (weight, value [, sigma])
```

ztestw_sig

ZTestw_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett z-test för en värdeserie.

```
ZTestw_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett z-test för en värdeserie. (weight, value [, sigma])
```

ztestw_sterr

ZTestw_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett z-test för en värdeserie.

```
ZTestw_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett z-test för en värdeserie. (weight, value [, sigma])
```

ztestw_upper

ZTestw_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

```
ZTestw_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier. (weight, value [, sigma])
```

ztestw_z

ZTestw_z() returnerar det aggregerade z-värdet för en värdeserie.

```
ZTestw_z() returnerar det aggregerade z-värdet för en värdeserie. (weight, value [, sigma])
```

ZTest_z

ZTest_z() returnerar det aggregerade z-värdet för en värdeserie.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_z(value[, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Populationsmedelvärdet 0 förutsätts. Om testet ska utföras kring ett annat medelvärde dras det medelvärdet av från stickprovsvärdena.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTest_z( Value-TestValue )
```

Se även:

p *Exempel på användning av z-test-funktioner (page 594)*

ZTest_sig

ZTest_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett z-test för en värdeserie.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_sig(value[, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Populationsmedelvärdet 0 förutsätts. Om testet ska utföras kring ett annat medelvärde dras det medelvärdet av från stickprovsvärdena.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTest_sig(Value-TestValue)
```

Se även:

p *Exempel på användning av z-test-funktioner (page 594)*

ZTest_dif

ZTest_dif() returnerar den aggregerade skillnaden i medelvärde för ett z-test för en värdeserie.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_dif(value[, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Populationsmedelvärdet 0 förutsätts. Om testet ska utföras kring ett annat medelvärde dras det medelvärdet av från stickprovsvärdena.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTest_dif(Value-TestValue)
```

Se även:

p *Exempel på användning av z-test-funktioner (page 594)*

ZTest_sterr

ZTest_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett z-test för en värdeserie.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_sterr(value[, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Populationsmedelvärdet 0 förutsätts. Om testet ska utföras kring ett annat medelvärde dras det medelvärdet av från stickprovsvärdena.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTest_sterr(Value-TestValue)
```

Se även:

p *Exempel på användning av z-test-funktioner (page 594)*

ZTest_conf

ZTest_conf() returnerar det aggregerade z-värdet för en värdeserie.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_conf(value[, sigma[, sig]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Populationsmedelvärdet 0 förutsätts. Om testet ska utföras kring ett annat medelvärde dras det medelvärdet av från stickprovsvärdena.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTest_conf(Value-TestValue)
```

Se även:

p *Exempel på användning av z-test-funktioner (page 594)*

ZTest_lower

ZTest_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_lower (grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTest_lower( Group, Value )
```

```
ZTest_lower( Group, Value, sig, false )
```

Se även:

p *Exempel på användning av z-test-funktioner (page 594)*

ZTest_upper

ZTest_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_upper (grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTest_upper( Group, Value )
```

```
ZTest_upper( Group, Value, sig, false )
```

Se även:

p *Exempel på användning av z-test-funktioner (page 594)*

ZTestw_z

ZTestw_z() returnerar det aggregerade z-värdet för en värdeserie.

Funktionen gäller z-test där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTestw_z (weight, value [, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Värdena bör returneras per **value**. För stickproven förutsätts medelvärdet 0. Om testet ska utföras kring ett annat medelvärde dras det värdet av från stickprovsvärdena.
weight	Varje stickprovsvärde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTestw_z( Weight, Value-TestValue)
```

Se även:

p *Exempel på användning av z-test-funktioner (page 594)*

ZTestw_sig

ZTestw_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett z-test för en värdeserie.

Funktionen gäller z-test där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTestw_sig (weight, value [, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Värdena bör returneras per **value**. För stickproven förutsätts medelvärdet 0. Om testet ska utföras kring ett annat medelvärde dras det värdet av från stickprovsvärdena.
weight	Varje stickprovsvärde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTestw_sig( Weight, Value-TestValue)
```

Se även:

p *Exempel på användning av z-test-funktioner (page 594)*

ZTestw_dif

ZTestw_dif() returnerar den aggregerade skillnaden i medelvärde för ett z-test för en värdeserie.

Funktionen gäller z-test där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTestw_dif ( weight, value [, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Värdena bör returneras per **value**. För stickproven förutsätts medelvärdet 0. Om testet ska utföras kring ett annat medelvärde dras det värdet av från stickprovsvärdena.
weight	Varje stickprovsvärde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTestw_dif( Weight, Value-TestValue)
```

Se även:

p *Exempel på användning av z-test-funktioner (page 594)*

ZTestw_sterr

ZTestw_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett z-test för en värdeserie.

Funktionen gäller z-test där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTestw_sterr (weight, value [, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Värdena bör returneras per **value**. För stickproven förutsätts medelvärdet 0. Om testet ska utföras kring ett annat medelvärde dras det värdet av från stickprovsvärdena.
weight	Varje stickprovsvärde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTestw_sterr( Weight, Value-TestValue)
```

Se även:

p *Exempel på användning av z-test-funktioner (page 594)*

ZTestw_conf

ZTestw_conf() returnerar det aggregerade z-konfidensintervallvärdet för en värdeserie.

Funktionen gäller z-test där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_conf(weight, value[, sigma[, sig]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Populationsmedelvärdet 0 förutsätts. Om testet ska utföras kring ett annat medelvärde dras det medelvärdet av från stickprovsvärdena.
weight	Varje stickprovsvärde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTestw_conf( Weight, Value-TestValue)
```

Se även:

p *Exempel på användning av z-test-funktioner (page 594)*

ZTestw_lower

ZTestw_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTestw_lower (grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTestw_lower( Group, Value )
```

```
ZTestw_lower( Group, Value, sig, false )
```

Se även:

p *Exempel på användning av z-test-funktioner (page 594)*

ZTestw_upper

ZTestw_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTestw_upper (grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTestw_upper( Group, Value )

ZTestw_upper( Group, Value, sig, false )
```

Se även:

p *Exempel på användning av z-test-funktioner (page 594)*

Exempel på statistiska testfunktioner

Detta avsnitt visar exempel på statistiska testfunktioner när de tillämpas på diagram och dataladdningsskript.

Exempel på användning av chi2-test-funktioner i diagram

Funktioner av typen chi2-test används för att hitta värden associerade med statistisk analys med chi2.

I det här avsnittet beskrivs hur du kan bygga visualiseringar med hjälp av stickprovsdata för att hitta värdena för fördelningstestfunktionerna med chi2 som finns i Qlik Sense. Se de individuella avsnitten för chi2-test-diagramfunktionerna för beskrivningar av syntax och argument.

Ladda data för exemplen

Det finns tre uppsättningar med stickprovsdata som beskriver de tre olika statistiska stickproven som laddas till skriptet.

Gör följande:

1. Skapa en ny app.

   I laddningen anger du följande:

   ```
   // Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the top of the script.
   ```

2.

   ```
   Sample_1:

   LOAD * inline [

   Grp,Grade,Count

   I,A,15

   I,B,7

   I,C,9

   I,D,20

   I,E,26

   I,F,19

   II,A,10

   II,B,11

   II,C,7

   II,D,15

   II,E,21

   II,F,16

   ];

   // Sample_2 data is pre-aggregated: If raw data is used, it must be aggregated using count()...

   Sample_2:

   LOAD * inline [

   Sex,Opinion,OpCount
   ```

```
1,2,58

1,1,11

1,0,10

2,2,35

2,1,25

2,0,23 ] (delimiter is ',');

// Sample_3a data is transformed using the crosstable statement...

Sample_3a:

crosstable(Gender, Actual) LOAD

Description,

[Men (Actual)] as Men,

[Women (Actual)] as Women;

LOAD * inline [

Men (Actual),Women (Actual),Description

58,35,Agree

11,25,Neutral

10,23,Disagree ] (delimiter is ',');

// Sample_3b data is transformed using the crosstable statement...

Sample_3b:

crosstable(Gender, Expected) LOAD

Description,

[Men (Expected)] as Men,

[Women (Expected)] as Women;

LOAD * inline [

Men (Expected),Women (Expected),Description

45.35,47.65,Agree

17.56,18.44,Neutral
```

```
16.09,16.91,Disagree ] (delimiter is ',');

// Sample_3a and Sample_3b will result in a (fairly harmless) Synthetic Key...
```

3. Klicka på om du vill ladda data.

Skapa visualiseringar för chi2-test-diagramfunktioner

Exempel: Exempel 1

Gör följande:

1. I dataimporten klickar du på för att komma till appvyn. Klicka sedan på det ark som du skapade tidigare.
   Arkvyn öppnas.
2. Klicka på **Redigera ark** för att redigera arket.
3. Från **Diagram** lägger du till en tabell och från **Fält** lägger du till Grp, Grade och Count som dimensioner.
   Den här tabellen visar dina stickprovsdata.
4. Lägg till ytterligare en tabell med följande uttryck som en dimension:
   `ValueList('p','df','Chi2')`
   Nu används funktionen för syntetiska dimensioner för att skapa etiketter för dimensionerna med namnen på de tre chi2-test-funktionerna.
5. Lägg till följande uttryck i tabellen som ett mått:

```
IF(ValueList('p','df','Chi2')='p',Chi2Test_p(Grp,Grade,Count),

IF(ValueList('p','df','Chi2')='df',Chi2Test_df(Grp,Grade,Count),

Chi2Test_Chi2(Grp,Grade,Count)))
```

   Detta gör att resultatvärdet för varje chi2-test-funktion sätts in i tabellen jämte dess associerade syntetiska dimension.
6. Ställ in **Talformat** för måttet på **Tal** och **3Gällande siffror**.

> *I uttrycket för måttet kan du använda följande uttryck istället:* `Pick(Match(ValueList('p','df','Chi2'),'p','df','Chi2'),Chi2Test_p(Grp,Grade,Count),Chi2Test_df(Grp,Grade,Count),Chi2Test_Chi2(Grp,Grade,Count))`

Resultat:

Resultattabellen för chi2-test-funktionerna för Exempel 1-data kommer att innehålla följande värden:

Resultattabell

p	df	Chi2
0.820	5	2.21

Exempel: Exempel 2

Gör följande:

1. I arket som du redigerade med Exempel 1 lägger du till en tabell från **Diagram** och från **Fält** lägger du till Sex, Opinion och OpCount som dimensioner.
2. Gör en kopia av resultattabellen från Exempel 1 genom att använda kommandona **Kopiera** och **Klistra in**. Redigera uttrycket i måttet och ersätt argumenten i alla tre chi2-test-funktioner med namnen på fälten som användes i data för Exempel 2, till exempel: `Chi2Test_p (Sex,Opinion,OpCount)`

Resultat:

Resultattabellen för chi2-test-funktionerna för Exempel 2-data kommer att innehålla följande värden:

Resultattabell

p	df	Chi2
0.000309	2	16.2

Exempel: Exempel 3

Gör följande:

1. Skapa två ytterligare tabeller på samma sätt som i exemplen för data i Exempel 1 och 2. Använd följande fält som dimensioner i dimensionstabellen: Gender, Description, Actual och Expected.
2. I resultattabellen använder du namnen på fälten som användes i data för Exempel 3, till exempel: `Chi2Test_p(Gender,Description,Actual,Expected)`

Resultat:

Resultattabellen för chi2-test-funktionerna för Exempel 3-data kommer att innehålla följande värden:

Resultattabell

p	df	Chi2
0.000308	2	16.2

Exempel på användning av chi2-test-funktioner i dataladdningsskriptet

Funktioner av typen chi2-test används för att hitta värden associerade med statistisk analys med chi2. I det här avsnittet finns en beskrivning av hur man använder de chi-kvadratiska distributionstestfunktioner som finns tillgängliga i Qlik Sense-dataladdningsskriptet. Se de individuella avsnitten om chi2-test-skriptfunktionerna för beskrivningar av syntax och argument.

I det här exemplet används en tabell som innehåller antalet studenter som har fått ett betyg (A-F) för två grupper med studenter (I och II).

Data table

Group	A	B	C	D	E	F
I	15	7	9	20	26	19
II	10	11	7	15	21	16

Ladda exempeldata

Gör följande:

1. Skapa en ny applikation.

   I Skriptredigeraren anger du följande:

   ```
   // Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the top of the script.
   ```

2.

   ```
   Sample_1:

   LOAD * inline [

   Grp,Grade,Count

   I,A,15

   I,B,7

   I,C,9

   I,D,20

   I,E,26

   I,F,19

   II,A,10

   II,B,11

   II,C,7

   II,D,15

   II,E,21

   II,F,16

   ];
   ```

3. Klicka på om du vill ladda data.

Du har nu laddat exempeldata.

Ladda chi2-test-funktionsvärdena

Nu laddar vi chi2-test-värdena utifrån exempeldata i en ny tabell, grupperade efter Grp.

Gör följande:

Lägg till följande i slutet av skriptet i Skriptredigeraren:

```
// Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the top of the script.
```

1.

```
Chi2_table:

LOAD Grp,

Chi2Test_chi2(Grp, Grade, Count) as chi2,

Chi2Test_df(Grp, Grade, Count) as df,

Chi2Test_p(Grp, Grade, Count) as p

resident Sample_1 group by Grp;
```

2. Klicka på om du vill ladda data.

Du har nu laddat chi2-test-värdena i en tabell med namnet Chi2_table.

Resultat

Du kan visa de resulterande chi2-test-värdena i datamodellvyn under **Förhandsgranskning**. De bör se ut så här:

Results

Grp	chi2	df	p
I	16.00	5	0.007
II	9.40	5	0.094

Skapa en typisk t-test-rapport

En typisk t-test-rapport kan innehålla tabeller med **Group Statistics** och **Independent Samples Test**-resultat.

I följande avsnitt kommer vi att bygga upp de här tabellerna med hjälp av Qlik Senset-test-funktioner som tillämpas på två fristående grupper av stickprov, Observation och Comparison. Motsvarande tabeller för dessa stickprov skulle se ut så här:

Statistik för gruppen

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Fristående stickprov

Type	conf	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval (Lower)	95% Confidence Interval (Upper)
Equal Variance not Assumed	0	3.534	37.116717335823	0.001	15.2	4.30101	6.48625	23.9137
Equal Variance Assumed	8.706939	3.534	38	0.001	15.2	4.30101	6.49306	23.9069

Ladda exempeldata

Gör följande:

1. Skapa en ny app med ett nytt ark.
2. Ange följande i Skriptredigeraren:

```
Table1:
Crosstable (Type, Value)
Load recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2 ] (delimiter is '|');
```

I detta laddningsskript är **recno()** inkluderat eftersom **crosstable** kräver tre argument. Det innebär att **recno()** helt enkelt ger ett extra argument, i det här fallet ett ID för varje rad. Utan det skulle **Comparison**-stickprovsvärdena inte läsas in.

3. Klicka på om du vill ladda data.

Skapa tabellen Group statistics

Gör följande:

1. I Skriptredigeraren klickar du på för att komma till appvyn. Klicka sedan på det ark som du skapade tidigare.
   Arkvyn öppnas.
2. Klicka på **Redigera ark** för att redigera arket.
3. Från **Diagram** lägger du till en tabell och från **Fält** lägger du till Type som en dimension i tabellen.
4. Lägg till följande uttryck som mått.

Exempel på uttryck

Etikett	Uttryck
N	Count(Value)
Mean	Avg(Value)
Standard Deviation	Stdev(Value)
Standard Error Mean	Sterr(Value)

5. Klicka på **Sortering** och kontrollera att Type är högst upp i sorteringslistan.

Resultat:

En Group statistics-tabell för dessa stickprov skulle se ut så här:

Statistik för gruppen

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Skapa tabellen Independent sample test

Gör följande:

1. Klicka på **Redigera ark** för att redigera arket.
2. Från **Diagram** lägger du till en tabell med följande uttryck som en dimension i tabellen. `=ValueList(Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1))` och ge den etiketten Type.

3. Lägg till följande uttryck som mått:

Exempel på uttryck

Etikett	Uttryck
conf	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_conf(Type, Value),TTest_conf(Type, Value, 0))
t	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_t(Type, Value),TTest_t(Type, Value, 0))
df	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_df(Type, Value),TTest_df(Type, Value, 0))
Sig. (2-tailed)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_sig(Type, Value),TTest_sig(Type, Value, 0))
Mean Difference	TTest_dif(Type, Value)
Standard Error Difference	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_sterr(Type, Value),TTest_sterr(Type, Value, 0))
95% Confidence Interval (Lower)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_lower(Type, Value,(1-(95)/100)/2),TTest_lower(Type, Value,(1-(95)/100)/2, 0))
95% Confidence Interval (Upper)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_upper(Type, Value,(1-(95)/100)/2),TTest_upper(Type, Value,(1-(95)/100)/2, 0))

Resultat:

Fristående stickprov

Type	conf	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval (Lower)	95% Confidence Interval (Upper)
Equal Variance not Assumed	0	3.534	37.116717335823	0.001	15.2	4.30101	6.48625	23.9137

Type	conf	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval (Lower)	95% Confidence Interval (Upper)
Equal Variance Assumed	8.706939	3.534	38	0.001	15.2	4.30101	6.49306	23.9069

Exempel på användning av z-test-funktioner

z-test-funktionerna används för att hitta värden associerade med statistiska analyser av z-test-typ för stora uppsättningar av stickprovsdata, vanligtvis större än 30, och där du vet att varians förekommer.

I det här avsnittet beskrivs hur du kan bygga visualiseringar med hjälp av sticksprovsdata för att hitta värdena för z-test-funktionerna som finns i Qlik Sense. Se de individuella avsnitten för z-test-diagramfunktionerna för beskrivningar av syntax och argument.

Ladda exempeldata

Stickprovsdata som används här är samma som används i exemplen för t-test-funktionen. Datauppsättningen skulle normalt sett anses för liten för att använda vid z-test-analyser, men den är tillräcklig för att illustrera användningen av de olika z-test-funktionerna i Qlik Sense.

Gör följande:

1. Skapa en ny app med ett nytt ark.

   Om du skapade en app för t-test -funktionerna, kunde du använda den och skapa ett nytt ark för de här funktionerna.

2. I Skriptredigeraren anger du följande:

```
Table1:
Crosstable (Type, Value)
Load recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
```

```
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2 ] (delimiter is '|');
```

I detta laddningsskript är **recno()** inkluderat eftersom **crosstable** kräver tre argument. Det innebär att **recno()** helt enkelt ger ett extra argument, i det här fallet ett ID för varje rad. Utan det skulle **Comparison**-stickprovsvärdena inte läsas in.

3. Klicka på om du vill ladda data.

Skapa tabellen z-test

Gör följande:

1. I Skriptredigeraren klickar du på för att komma till appvyn. Klicka sedan på det ark som du skapade ovan.
   Arkvyn öppnas.
2. Klicka på **Redigera ark** för att redigera arket.
3. Från **Diagram** lägger du till en tabell och från **Fält** lägger du till Type som en dimension.
4. Lägg till följande uttryck i tabellen som ett mått.

Exempel på uttryck

Etikett	Uttryck
ZTest Conf	ZTest_conf(Value)
ZTest Dif	ZTest_dif(Value)
ZTest Sig	ZTest_sig(Value)
ZTest Sterr	ZTest_sterr(Value)
ZTest Z	ZTest_z(Value)

*Du kanske måste justera talformatet för måtten för att se meningsfulla värden. Tabellen blir enklare att läsa om du ställer in talformat på de flesta av måtten till **Tal > Enkel**, i stället för **Auto**. Men för ZTest Sig, till exempel, använd talformateringen: **Anpassad**, och justera sedan formatmönstret till **#.######**.*

Resultat:

Den resulterande tabellen för z-test-funktionerna för exempeldata kommer att innehålla följande värden:

z-test resultattabell

Type	ZTest Conf	ZTest Dif	ZTest Sig	ZTest Sterr	ZTest Z
Comparison	6.40	11.95	0.000123	3.27	3.66
Observation	5.48	27.15	0.000000	2.80	9.71

Skapa tabellen z-testw

z-testw-funktionerna används när indataserien visas i viktat tvåkolumnsformat. Uttrycken kräver ett värde för weight.

I exemplen här används värdet 2 genomgående, men du kan använda ett uttryck som ska definiera ett värde för weight för varje observation.

Gör följande:

1. I Skriptredigeraren klickar du på för att komma till appvyn. Klicka sedan på det ark som du skapade ovan.
   Arkvyn öppnas.
2. Klicka på **Redigera ark** för att redigera arket.
3. Från **Diagram** lägger du till en tabell och från **Fält** lägger du till Type som en dimension.
4. Lägg till följande uttryck i tabellen som ett mått.

   Exempel på uttryck

Etikett	Uttryck
ZTestw Conf	ZTestw_conf(2,Value)
ZTestw Dif	ZTestw_dif(2,Value)
ZTestw Sig	ZTestw_sig(2,Value)
ZTestw Sterr	ZTestw_sterr(2,Value)
ZTestw Z	ZTestw_z(2,Value)

   Använd samma nummerformatering som i z-test-funktionsexemplet.

Resultat:

Den resulterande tabellen för z-testw-funktionerna kommer att innehålla följande värden:

z-testw resultattabell

Type	ZTestw Conf	ZTestw Dif	ZTestw Sig	ZTestw Sterr	ZTestw Z
Comparison	4.47	11.95	8.037185e-08	2.28	5.24
Observation	3.83	27.15	0	1.95	13.91

Strängaggregeringsfunktioner

Den här delen beskriver strängrelaterade aggregeringsfunktioner.

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Strängaggregeringsfunktioner i dataladdningsskriptet

Concat

Concat() används för att kombinera strängvärden. Denna skriptfunktion returnerar den aggregerade sträng-konkateneringen av alla värden i uttryck itererat över ett antal poster enligt vad som definierats i en **group by**-sats.

```
Concat ([ distinct ] expression [, delimiter [, sort-weight]])
```

FirstValue

FirstValue() returnerar värdet som laddades först från posterna som definierats av uttrycket, sorterade efter en **group by**-sats.

> *Funktionen finns endast som skriptfunktion.*

```
FirstValue (expression)
```

LastValue

LastValue() returnerar värdet som laddades sist från posterna som definierats av uttrycket, sorterade efter en **group by**-sats.

> *Funktionen finns endast som skriptfunktion.*

```
LastValue (expression)
```

MaxString

MaxString() hittar strängvärden i uttrycket och returnerar det senaste textvärdet alfabetiskt sorterat under ett antal poster, som definieras av en **group by**-sats.

```
MaxString (expression )
```

MinString

MinString() hittar strängvärden i uttrycket och returnerar det första textvärdet alfabetiskt sorterat under ett antal poster, som definieras av en **group by**-sats.

```
MinString (expression )
```

Strängaggregeringsfunktioner i diagram

Följande diagramfunktioner kan användas för att aggregera strängar i diagram.

Concat

Concat() används för att kombinera strängvärden. Funktionen returnerar den aggregerade strängkonkateneringen av alla värden av uttrycket utvärderat över varje dimension.

```
Concat - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]
string[, delimiter[, sort_weight]])
```

MaxString

MaxString() finner strängvärden i uttrycket eller fältet och returnerar det senaste textvärdet i alfabetisk sorteringsordning.

```
MaxString - diagramfunktion({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MinString

MinString() finner strängvärden i uttrycket eller fältet och returnerar det första textvärdet i alfabetisk sorteringsordning.

```
MinString - diagramfunktion({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Concat

Concat() används för att kombinera strängvärden. Denna skriptfunktion returnerar den aggregerade sträng-konkateneringen av alla värden i uttryck itererat över ett antal poster enligt vad som definierats i en **group by**-sats.

Syntax:

```
Concat ([ distinct ] string [, delimiter [, sort-weight]])
```

Returnerad datatyp: sträng

Argument:

Det uttryck eller fält som innehåller den sträng som ska behandlas.

Argument

Argument	Beskrivning
string	Det uttryck eller fält som innehåller den sträng som ska behandlas.
delimiter	Varje värde kan avgränsas med den sträng som finns i delimiter.
sort-weight	Konkanteneringens ordning kan avgöras av värdet på dimensionen **sort-weight**, om sådant finns, där strängen motsvarar det lägsta värde som visas först i konkateneringen.
distinct	Om predikatet **distinct** förekommer framför uttrycket ignoreras alla dubbletter.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

Exempel och resultat

Exempel	Resultat	Resultat som har lagts till på ett ark

`TeamData:`
`LOAD * inline [`
`SalesGroup|Team|Date|Amount`
`East|Gamma|01/05/2013|20000`
`East|Gamma|02/05/2013|20000`
`West|Zeta|01/06/2013|19000`
`East|Alpha|01/07/2013|25000`
`East|Delta|01/08/2013|14000`
`West|Epsilon|01/09/2013|17000`
`West|Eta|01/10/2013|14000`
`East|Beta|01/11/2013|20000`
`West|Theta|01/12/2013|23000`
`] (delimiter is '|');`
 	
`Concat1:`
 	
`LOAD SalesGroup,Concat(Team) as TeamConcat1`
`Resident TeamData Group By SalesGroup;`
	SalesGroup
	
	East
	
	West
		TeamConcat1
		
		AlphaBetaDeltaGammaGamma
		
		EpsilonEtaThetaZeta

Givet att **TeamData**-tabellen är laddad som i föregående exempel: `LOAD SalesGroup,Concat(distinct Team,'-') as TeamConcat2 Resident TeamData Group By SalesGroup;`	SalesGroup East West	TeamConcat2 Alpha-Beta-Delta-Gamma Epsilon-Eta-Theta-Zeta
Givet att **TeamData**-tabellen är laddad som i föregående exempel. Eftersom argumentet för **sort-weight** läggs till ordnas resultaten efter värdet på dimensionen Amount: `LOAD SalesGroup,Concat(distinct Team,'-',Amount) as TeamConcat2 Resident TeamData Group By SalesGroup;`	SalesGroup East West	TeamConcat2 Delta-Beta-Gamma-Alpha Eta-Epsilon-Zeta-Theta

Concat - diagramfunktion

Concat() används för att kombinera strängvärden. Funktionen returnerar den aggregerade strängkonkateneringen av alla värden av uttrycket utvärderat över varje dimension.

Syntax:

```
Concat({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} string[, delimiter
[, sort_weight]])
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
string	Det uttryck eller fält som innehåller den sträng som ska behandlas.
delimiter	Varje värde kan avgränsas med den sträng som finns i delimiter.
sort-weight	Konkanteneringens ordning kan avgöras av värdet på dimensionen **sort-weight**, om sådant finns, där strängen motsvarar det lägsta värde som visas först i konkateneringen.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel och resultat:

Results table

SalesGroup	Amount	Concat(Team)	Concat(TOTAL <SalesGroup> Team)
East	25000	Alpha	AlphaBetaDeltaGammaGamma
East	20000	BetaGammaGamma	AlphaBetaDeltaGammaGamma
East	14000	Delta	AlphaBetaDeltaGammaGamma
West	17000	Epsilon	EpsilonEtaThetaZeta
West	14000	Eta	EpsilonEtaThetaZeta
West	23000	Theta	EpsilonEtaThetaZeta
West	19000	Zeta	EpsilonEtaThetaZeta

Exempel på funktioner

Exempel	Resultat
`Concat(Team)`	Tabellen är konstruerad av dimensionerna SalesGroup och Amount, samt variationer av måttet Concat(Team). Bortse från totalvärderesultatet och observera att trots att det finns data för åtta värden av Team spridda över två värden av SalesGroup, är det enda resultatet av måttet Concat(Team) som konkatenerar fler än ett Team-strängvärde i tabellen raden med dimensionen Amount 20 000, vilket ger resultatet BetaGammaGamma. Detta beror på att det finns tre värden för Amount 20 000 i indata. Alla andra resultat förblir okonkatenerade när måttet omfattar dimensionerna eftersom det bara finns ett värde av Team för varje kombination av SalesGroup och Amount.
`Concat (DISTINCT Team,', ')`	Beta, Gamma. eftersom DISTINCT-kvalificeraren betyder att dublettresultatet för Gamma ignoreras. Dessutom definieras avgränsarargumentet som ett komma följt av mellanslag.
`Concat (TOTAL <SalesGroup> Team)`	Alla strängvärden av Team konkateneras om TOTAL-kvalificeraren används. Med fältvalet <SalesGroup> angivet delar detta resultatet i två värden av dimensionen SalesGroup. För SalesGroupEast är resultaten AlphaBetaDeltaGammaGamma. För SalesGroupWest är resultaten EpsilonEtaThetaZeta.
`Concat (TOTAL <SalesGroup> Team,';', Amount)`	Om du lägger till argumentet för **sort-weight**: Amount ordnas resultaten efter värdet på dimensionen Amount. Resultaten blir DeltaBetaGammaGammaAlpha och EtaEpsilonZEtaTheta.

Data som används i exemplet:

```
TeamData:
LOAD * inline [
SalesGroup|Team|Date|Amount
East|Gamma|01/05/2013|20000
East|Gamma|02/05/2013|20000
West|Zeta|01/06/2013|19000
East|Alpha|01/07/2013|25000
East|Delta|01/08/2013|14000
West|Epsilon|01/09/2013|17000
West|Eta|01/10/2013|14000
East|Beta|01/11/2013|20000
West|Theta|01/12/2013|23000
] (delimiter is '|');
```

FirstValue

FirstValue() returnerar värdet som laddades först från posterna som definierats av uttrycket, sorterade efter en **group by**-sats.

Funktionen finns endast som skriptfunktion.

Syntax:

```
FirstValue ( expr)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Begränsningar:

Om inget textvärde hittas returneras NULL.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

Resulterande data

Exempel	Resultat	Resultat på ett ark

`TeamData:`
`LOAD * inline [`
`SalesGroup|Team|Date|Amount`
`East|Gamma|01/05/2013|20000`
`East|Gamma|02/05/2013|20000`
`West|Zeta|01/06/2013|19000`
`East|Alpha|01/07/2013|25000`
`East|Delta|01/08/2013|14000`
`West|Epsilon|01/09/2013|17000`
`West|Eta|01/10/2013|14000`
`East|Beta|01/11/2013|20000`
`West|Theta|01/12/2013|23000`
`] (delimiter is '|');`
 	
`FirstValue1:`
 	
`LOAD SalesGroup,FirstValue(Team) as FirstTeamLoaded`
`Resident TeamData Group By SalesGroup;`
	SalesGroup
	
	East
	
	West
		FirstTeamLoaded
		
		Gamma
		
		Zeta

LastValue

LastValue() returnerar värdet som laddades sist från posterna som definierats av uttrycket, sorterade efter en **group by**-sats.

Funktionen finns endast som skriptfunktion.

Syntax:

```
LastValue ( expr )
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Begränsningar:

Om inget textvärde hittas returneras NULL.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår applikation för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Exempel	Resultat	Resultat med anpassad sortering

`TeamData:`
`LOAD * inline [`
`SalesGroup|Team|Date|Amount`
`East|Gamma|01/05/2013|20000`
`East|Gamma|02/05/2013|20000`
`West|Zeta|01/06/2013|19000`
`East|Alpha|01/07/2013|25000`
`East|Delta|01/08/2013|14000`
`West|Epsilon|01/09/2013|17000`
`West|Eta|01/10/2013|14000`
`East|Beta|01/11/2013|20000`
`West|Theta|01/12/2013|23000`
`] (delimiter is '|');`
 	
`LastValue1:`
 	
`LOAD SalesGroup,LastValue(Team) as LastTeamLoaded`
`Resident TeamData Group By SalesGroup;`
	SalesGroup
	
	East
	
	West
		LastTeamLoaded
		
		Beta
		
		Theta

MaxString

MaxString() hittar strängvärden i uttrycket och returnerar det senaste textvärdet alfabetiskt sorterat under ett antal poster, som definieras av en **group by**-sats.

Syntax:

```
MaxString ( expr )
```

Returnerad datatyp: dual

Argument:

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Begränsningar:

Om inget textvärde hittas returneras NULL.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

Exempel	Resultat	

`TeamData:`
`LOAD * inline [`
`SalesGroup|Team|Date|Amount`
`East|Gamma|01/05/2013|20000`
`East|Gamma|02/05/2013|20000`
`West|Zeta|01/06/2013|19000`
`East|Alpha|01/07/2013|25000`
`East|Delta|01/08/2013|14000`
`West|Epsilon|01/09/2013|17000`
`West|Eta|01/10/2013|14000`
`East|Beta|01/11/2013|20000`
`West|Theta|01/12/2013|23000`
`] (delimiter is '|');`
 	
`Concat1:`
 	
`LOAD SalesGroup,MaxString(Team) as MaxString1 Resident TeamData Group By SalesGroup;`
	SalesGroup
	
	East
	
	West
		MaxString1
		
		Gamma
		
		Zeta

Givet att **TeamData**-tabellen är laddad som i föregående exempel och ditt dataladdningsskript har SET-satsen: `SET DateFormat='DD/MM/YYYY';`: `LOAD SalesGroup,MaxString(Date) as MaxString2 Resident TeamData Group By SalesGroup;`	SalesGroup East West	MaxString2 01/11/2013 01/12/2013

MaxString - diagramfunktion

MaxString() finner strängvärden i uttrycket eller fältet och returnerar det senaste textvärdet i alfabetisk sorteringsordning.

Syntax:

```
MaxString({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Om uttrycket inte innehåller några värden med strängrepresentation, returneras NULL.

Exempel och resultat:

Resultattabell

SalesGroup	Amount	MaxString(Team)	MaxString(Date)
East	14000	Delta	2013/08/01
East	20000	Gamma	2013/11/01
East	25000	Alpha	2013/07/01
West	14000	Eta	2013/10/01
West	17000	Epsilon	2013/09/01
West	19000	Zeta	2013/06/01
West	23000	Theta	2013/12/01

Exempel på funktioner

Exempel	Resultat
`MaxString (Team)`	Det finns tre värden för 20 000 för dimensionen Amount: två på Gamma (på olika datum) och ett på Beta. Resultatet av måttet MaxString (Team) är därför Gamma eftersom detta är det högsta värdet i de sorterade strängarna.
`MaxString (Date)`	2013/11/01 är det största Date-värdet av de tre som är associerade med dimensionen Amount. Detta förutsätter att ditt skript har SET-satsen `SET DateFormat='YYYY-MM-DD';'`

Data som används i exemplet:

```
TeamData:
LOAD * inline [
SalesGroup|Team|Date|Amount
East|Gamma|01/05/2013|20000
East|Gamma|02/05/2013|20000
West|Zeta|01/06/2013|19000
East|Alpha|01/07/2013|25000
East|Delta|01/08/2013|14000
West|Epsilon|01/09/2013|17000
West|Eta|01/10/2013|14000
East|Beta|01/11/2013|20000
West|Theta|01/12/2013|23000
] (delimiter is '|');
```

MinString

MinString() hittar strängvärden i uttrycket och returnerar det första textvärdet alfabetiskt sorterat under ett antal poster, som definieras av en **group by**-sats.

Syntax:

```
MinString ( expr )
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Begränsningar:

Om inget textvärde hittas returneras NULL.

Exempel och resultat:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

Resulterande data

Exempel	Resultat	

`TeamData:`
`LOAD * inline [`
`SalesGroup|Team|Date|Amount`
`East|Gamma|01/05/2013|20000`
`East|Gamma|02/05/2013|20000`
`West|Zeta|01/06/2013|19000`
`East|Alpha|01/07/2013|25000`
`East|Delta|01/08/2013|14000`
`West|Epsilon|01/09/2013|17000`
`West|Eta|01/10/2013|14000`
`East|Beta|01/11/2013|20000`
`West|Theta|01/12/2013|23000`
`] (delimiter is '|');`
 	
`Concat1:`
 	
`LOAD SalesGroup,MinString(Team) as MinString1 Resident`
`TeamData Group By SalesGroup;`
	SalesGroup
	
	East
	
	West
		MinString1
		
		Alpha
		
		Epsilon

Givet att **TeamData**-tabellen är laddad som i föregående exempel och ditt dataladdningsskript har SET-satsen:
`SET DateFormat='DD/MM/YYYY';`:
 	
`LOAD SalesGroup,MinString(Date) as MinString2 Resident`
`TeamData Group By SalesGroup;`
	SalesGroup
	
	East
	
	West
		MinString2
		
		01/05/2013
		
		01/06/2013

MinString - diagramfunktion

MinString() finner strängvärden i uttrycket eller fältet och returnerar det första textvärdet i alfabetisk sorteringsordning.

Syntax:

```
MinString({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel och resultat:

Exempeldata

SalesGroup	Amount	MinString(Team)	MinString(Date)
East	14000	Delta	2013/08/01
East	20000	Beta	2013/05/01
East	25000	Alpha	2013/07/01
West	14000	Eta	2013/10/01
West	17000	Epsilon	2013/09/01
West	19000	Zeta	2013/06/01
West	23000	Theta	2013/12/01

Exempel på funktioner

Exempel	Resultat
`MinString (Team)`	Det finns tre värden för 20 000 för dimensionen Amount: två på Gamma (på olika datum) och ett på Beta. Resultatet av måttet MinString (Team) är därför Beta eftersom detta är det första värdet i de sorterade strängarna.
`MinString (Date)`	2013/11/01 är det tidigaste Date-värdet av de tre som är associerade med dimensionen Amount. Detta förutsätter att ditt skript har SET-satsen `SET DateFormat='YYYY-MM-DD';'`

Data som används i exemplet:

```
TeamData:
LOAD * inline [
SalesGroup|Team|Date|Amount
East|Gamma|01/05/2013|20000
East|Gamma|02/05/2013|20000
West|Zeta|01/06/2013|19000
East|Alpha|01/07/2013|25000
East|Delta|01/08/2013|14000
West|Epsilon|01/09/2013|17000
West|Eta|01/10/2013|14000
East|Beta|01/11/2013|20000
West|Theta|01/12/2013|23000
] (delimiter is '|');
```

Syntetiska dimensionsfunktioner

En syntetisk dimension skapas i appen av värden som genererats från de syntetiska dimensionsfunktionerna och inte direkt från fält i datamodellen. När värden som genererats från en syntetisk dimensionsfunktion används i ett diagram som en beräknad dimension skapar detta en syntetisk dimension. Syntetiska dimensioner gör att du kan skapa till exempel diagram med dimensioner med värden som kommer från dina data, det vill säga, dynamiska dimensioner.

Syntetiska dimensioner påverkas inte av urval.

Följande syntetiska dimensionsfunktioner kan användas i diagram.

ValueList

ValueList() returnerar en uppsättning listade värden, som när de används i en beräknad dimension bildar en syntetisk dimension.

```
ValueList - diagramfunktion (v1 {, Expression})
```

ValueLoop

ValueLoop() returnerar en uppsättning itererade värden, som när de används i en beräknad dimension bildar en syntetisk dimension.

```
ValueLoop - diagramfunktion(from [, to [, step ]])
```

ValueList - diagramfunktion

ValueList() returnerar en uppsättning listade värden, som när de används i en beräknad dimension bildar en syntetisk dimension.

*I diagram som har en syntetisk dimension skapad med **ValueList**-funktionen är det möjligt att referera till det dimensionsvärde som motsvarar en specifik uttryckscell genom att åberopa **ValueList**-funktionen med samma parametrar i diagramuttrycket. Funktionen kan självfallet användas överallt i layouten. Utöver användningen i syntetiska dimensioner är den endast meningsfull inuti en aggregeringsfunktion.*

Syntetiska dimensioner påverkas inte av urval.

Syntax:

```
ValueList(v1 {,...})
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
v1	Statiskt värde (vanligtvis en sträng, men kan vara ett tal).
{,...}	Valbar lista över statiska värden.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`ValueList ('Number of Orders', 'Average Order Size', 'Total Amount')`	När det används för att skapa en dimension i en tabell till exempel, resulterar det här i tre strängvärden som radetiketter i tabellen. Dessa kan refereras i ett uttryck.
`=IF( ValueList ('Number of Orders', 'Average Order Size', 'Total Amount') = 'Number of Orders', count (SaleID), IF( ValueList ('Number of Orders', 'Average Order Size', 'Total Amount') = 'Average Order Size', avg (Amount), sum (Amount) ))`	Det här uttrycket tar värdena från den skapade dimensionen och refererar dem i en nästlad IF-sats som indata för tre aggregeringsfunktioner: **ValueList()** Created dimension / Year / Added expression 522.00 Number of Orders 2012 5.00 Number of Orders 2013 7.00 Average Order Size 2012 13.20 Average Order Size 2013 15.43 Total Amount 2012 66.00 Total Amount 2013 108.00

Data som används i exempel:

```
SalesPeople:
LOAD * INLINE [
SaleID|SalesPerson|Amount|Year
1|1|12|2013
2|1|23|2013
```

```
3|1|17|2013
4|2|9|2013
5|2|14|2013
6|2|29|2013
7|2|4|2013
8|1|15|2012
9|1|16|2012
10|2|11|2012
11|2|17|2012
12|2|7|2012
] (delimiter is '|');
```

ValueLoop - diagramfunktion

ValueLoop() returnerar en uppsättning itererade värden, som när de används i en beräknad dimension bildar en syntetisk dimension.
De genererade värdena börjar med **from**-värdet och avslutas med **to**-värdet inklusive de mellanliggande värdena i stegvis ordning.

> *I diagram som har en syntetisk dimension skapad med **ValueLoop**-funktionen är det möjligt att referera till det dimensionsvärde som motsvarar en specifik uttryckscell genom att åberopa **ValueLoop**-funktionen med samma parametrar i diagramuttrycket. Funktionen kan självfallet användas överallt i layouten. Utöver användningen i syntetiska dimensioner är den endast meningsfull inuti en aggregeringsfunktion.*

> *Syntetiska dimensioner påverkas inte av urval.*

Syntax:

```
ValueLoop(from [, to [, step ]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
from	Startvärdet i den värdeuppsättning som ska genereras.
to	Slutvärdet i den värdeuppsättning som ska genereras.
step	Storleken på stegen mellan värdena.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`ValueLoop (1, 10)`	Detta skapar en dimension i en tabell till exempel, som kan användas för syften som numrerade etiketter. Exemplet här resulterar i värden mellan 1 och 10. Dessa värden kan sedan refereras i ett uttryck.
`ValueLoop (2, 10,2)`	Detta exempel resulterar i värdena 2, 4, 6, 8 och 10 eftersom argumentet step har värdet 2.

Nästlade aggregeringar

Ibland kan man behöva applicera en aggregering på resultatet av en annan aggregering. Detta kallas nästlade aggregeringar.

Det går inte att nästla aggregeringar i de flesta diagramuttrycken. Det går dock att nästla aggregeringar om du använder kvalificeraren **TOTAL** i den inre aggregeringsfunktionen.

> *Det går inte att nästla mer än 100 nivåer.*

Nästlade aggregeringar med TOTAL-kvalificeraren

Exempel:

Säg att du till exempel vill beräkna summan för fältet **Sales** men bara vill inkludera transaktioner med **OrderDate** som motsvarar föregående år. Föregående år kan nås via aggregeringsfunktionen **`Max(TOTAL`** `Year(OrderDate)`**`)`**.

Följande aggregering skulle returnera det önskade resultatet:

```
Sum(If(Year(OrderDate)=Max(TOTAL Year(OrderDate)), Sales))
```

Qlik Sense kräver att kvalificeraren **TOTAL** inkluderas i den här typen av nästling. Det krävs för den önskade jämförelsen. Den här sortens nästling är ganska vanlig och utgör en god vana.

Se även:

p *Aggr - diagramfunktion (page 612)*

8.3 Aggr - diagramfunktion

Aggr() returnerar en uppsättning värden för uttrycket beräknade över den angivna dimensionen eller de angivna dimensionerna. Exempelvis det maximala värdet av försäljningen, per kund, per region.

Aggr-funktionen används för nästlade aggregeringar där den första parametern (den inre aggregeringen) beäknas en gång per dimensionsvärde. Dimensionerna anges i den andra parametern (och efterföljande parametrar).

Dessutom ska **Aggr**-funktionen omslutas av en yttre aggregeringsfunktion där matrisen med resultat från **Aggr**-funktionen används som indata till den aggregering som den är nästlad i.

Syntax:

```
Aggr({SetExpression}[DISTINCT] [NODISTINCT ] expr, StructuredParameter{, StructuredParameter})
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Ett uttryck som består av en aggregeringsfunktion. Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet.
StructuredParameter	StructuredParameter består av en dimension2 och ett (valfritt) sorteringskriterium i formatet: `(Dimension(Sort-type, Ordering))` Dimensionen är ett enda fält och kan inte vara ett uttryck. Dimensionen används för att bedöma vilken uppsättning med värden som Aggr-uttrycket ska beräknas på. Om sorteringskriterier används, sorteras den uppsättning med värden som skapas av Aggr-funktionen och som beräknats för dimensionen. Detta är viktigt när sorteringsordningen påverkar resultatet av det uttryck som Aggr-funktionen är innesluten i. Mer information om hur du använder sorteringskriterier finns i Lägga till sorteringskriterier till dimensionen i den strukturerade parametern.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om uttrycksargumentet föregås av kvalificeraren **distinct**, eller om ingen kvalificerare alls används, kommer varje distinkt kombination av dimensionsvärden endast att generera ett returvärde. Det är normalt så här aggregeringar skapas - varje distinkt kombination av dimensionsvärden ger en rad i diagrammet.

Argument	Beskrivning
NODISTINCT	Om uttrycksargumentet föregås av kvalificeraren **nodistinct** kan varje kombination av dimensionsvärden generera mer än ett returvärde, beroende på den underliggande datastrukturen. Om det bara finns en dimension, returnerar funktionen **aggr** en uppsättning med samma antal element som det finns rader i källdata.

Grundläggande aggregeringsfunktioner, såsom **Sum**, **Min** och **Avg**, returnerar ett enda numeriskt värde, medan funktionen Aggr() kan jämföras med att skapa en temporär uppsättning med resultat (en virtuell tabell), över vilken en annan aggregering kan göras. Detta kan exempelvis göras genom att beräkna ett genomsnittligt försäljningsvärde genom att summera försäljningen efter kund i en **Aggr()**-sats och sedan beräkna genomsnittet av de summerade resultaten: **Avg(TOTAL Aggr(Sum(Sales),Customer))**.

Använd Aggr()-funktionen i beräknade dimensioner om du vill skapa kapslade diagramaggregeringar på flera nivåer.

Begränsningar:

Varje dimension i en Aggr()-funktion måste bestå av ett enda fält. Den får inte vara ett uttryck (beräknad dimension).

Lägga till sorteringskriterier till dimensionen i den strukturerade parametern

I sin grundläggande form är argumentet StructuredParameter i syntaxen för Aggr-funktionen en enda dimension. Uttrycket: Aggr(Sum(Sales, Month)) hittar det totala värdet för försäljning per månad. Om det innesluts i en annan aggregeringsfunktion kan det dock ge oväntade resultat om inte sorteringskriterier används. Det beror på att en del dimensioner kan sorteras numeriskt eller alfabetiskt, och så vidare.

I StructuredParameter-argumentet i Aggr-funktionen kan du ange sorteringskriterier för dimensionen i dina uttryck. På så sätt fastställer du en sorteringsordning för den virtuella tabell som produceras av Aggr-funktionen.

Argumentet StructuredParameter har följande syntax:

```
(FieldName, (Sort-type, Ordering))
```

Strukturerade parametrar kan kapslas:

```
(FieldName, (FieldName2, (Sort-type, Ordering)))
```

Sorteringstypen kan vara: NUMERIC, TEXT, FREQUENCY eller LOAD_ORDER.

Ordningstyperna som är associerade med varje sorteringstyp är som följer:

Tillåtna ordningstyper

Sorteringstyp	Tillåtna ordningstyper
NUMERIC	ASCENDING, DESCENDING eller REVERSE
TEXT	ASCENDING, A2Z, DESCENDING, REVERSE eller Z2A
FREQUENCY	DESCENDING, REVERSE eller ASCENDING
LOAD_ORDER	ASCENDING, ORIGINAL, DESCENDING eller REVERSE

Ordningstyperna REVERSE och DESCENDING är motsvarigheter.

För sorteringstypen TEXT är ordningstyperna ASCENDING och A2Z motsvarigheter, och DESCENDING, REVERSE och Z2A är motsvarigheter.

För sorteringstypen LOAD_ORDER är ordningstyperna ASCENDING och ORIGINAL motsvarigheter.

Exempel: diagramuttryck som använder Aggr

Exempel – diagramuttryck

Diagramuttryck exempel 1

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplet nedan.

```
ProductData:

LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD|25|25
Canutility|AA|8|15
Canutility|CC|0|19
] (delimiter is '|');
```

Diagramuttryck

Skapa en KPI-visualisering i ett Qlik Sense-ark. Lägg till följande uttryck i KPI-visualiseringen som ett mått:

```
Avg(Aggr(Sum(UnitSales*UnitPrice), Customer))
```

Resultat

376.7

Förklaring

Uttrycket `Aggr(Sum(UnitSales*UnitPrice), Customer)` hittar det totala värdet för försäljning utifrån **Customer**, och returnerar en uppsättning med värden: 295, 715 och 120 för de tre **Customer**-värdena.

Vi har i praktiken skapat en tillfällig lista med värden utan att ha behövt skapa en explicit tabell eller kolumn som innehåller dessa värden.

De här tre värdena används som indata för funktionen **Avg()** som räknar ut medelvärdet för försäljning, 376.7.

Diagramuttryck exempel 2

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplet nedan.

```
ProductData:

LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|BB|7|12
Betacab|CC|2|22
Betacab|CC|4|20
Betacab|DD|25|25
Canutility|AA|8|15
Canutility|AA|5|11
Canutility|CC|0|19
] (delimiter is '|');
```

Diagramuttryck

Skapa en tabellvisualisering i ett Qlik Sense-ark med **Customer**, **Product**, **UnitPrice** och **UnitSales** som dimensioner. Lägg till följande uttryck i tabellen som ett mått:

```
Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
```

Resultat

Customer	Product	UnitPrice	UnitSales	Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
Astrida	AA	15	10	16
Astrida	AA	16	4	16
Astrida	BB	9	9	15

Customer	Product	UnitPrice	UnitSales	Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
Astrida	BB	15	10	15
Betacab	BB	10	5	12
Betacab	BB	12	7	12
Betacab	CC	20	4	22
Betacab	CC	22	2	22
Betacab	DD	25	25	25
Canutility	AA	11	5	15
Canutility	AA	15	8	15
Canutility	CC	19	0	19

Förklaring

En uppsättning med värden: 16, 16, 15, 15, 12, 12, 22, 22, 25, 15, 15 och 19. Kvalificeraren **nodistinct** innebär att uppsättningen innehåller ett element för varje rad i källdata: varje är maximalt **UnitPrice** för varje **Customer** och **Product**.

Diagramuttryck exempel 3

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplet nedan.

```
Set vNumberOfOrders = 1000;

OrderLines:
Load
       RowNo() as OrderLineID,
       OrderID,
       OrderDate,
       Round((Year(OrderDate)-2005)*1000*Rand()*Rand()*Rand1) as Sales
       While Rand()<=0.5 or IterNo()=1;
Load * Where OrderDate<=Today();
Load
       Rand() as Rand1,
       Date(MakeDate(2013)+Floor((365*4+1)*Rand())) as OrderDate,
       RecNo() as OrderID
       Autogenerate vNumberOfOrders;

Calendar:
Load distinct
       Year(OrderDate) as Year,
       Month(OrderDate) as Month,
```

```
OrderDate
Resident OrderLines;
```

Diagramuttryck

Skapa en tabellvisualisering i ett Qlik Sense-ark med **Year** och **Month** som dimensioner. Lägg till följande uttryck i tabellen som mått:

- `Sum(Sales)`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))` har etiketten Structured Aggr() i tabellen.

Resultat

Year	Month	Sum(Sales)	Structured Aggr()
2013	Jan	53495	53495
2013	Feb	48580	102075
2013	Mar	25651	127726
2013	Apr	36585	164311
2013	May	61211	225522
2013	Jun	23689	249211
2013	Jul	42311	291522
2013	Aug	41913	333435
2013	Sep	28886	362361
2013	Oct	25977	388298
2013	Nov	44455	432753
2013	Dec	64144	496897
2014	Jan	67775	67775

Förklaring

Det här exemplet visar aggregerade värden under en tolvmånadersperiod för varje år i kronologiskt stigande ordning, vilket förklarar de strukturerade parametrarna (Numeric, Ascending) i uttrycket **Aggr()**. Två specifika dimensioner krävs som strukturerade parametrar: **Year** och **Month**, sorterat (1) **Year** (numeriskt) och (2) **Month** (numeriskt). Dessa två dimensioner måste användas i tabellen eller diagramvisualiseringen. Det krävs för att dimensionslistan i funktionen **Aggr()** ska motsvara dimensionerna i objektet som används i visualiseringen.

Du kan jämföra skillnaden mellan dessa mått i en tabell eller i separata linjediagram:

- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year), (Month) ))`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))`

Det bör vara tydligt att endast det senare uttrycket utför den önskade ackumuleringen av aggregerade värden.

Se även:

p *Grundläggande aggregeringsfunktioner (page 369)*

8.4 Färgfunktioner

Dessa funktioner kan användas i uttryck som används för att ange och utvärdera färgegenskaper för diagramobjekt, samt i datainläsningsskript.

> *Qlik Sense har stöd för färgfunktionerna **Color()**, **qliktechblue** och **qliktechgray** för att säkerställa bakåtkompatibilitet, men dessa funktioner bör helst inte användas.*

ARGB

ARGB() används i uttryck för att ange eller utvärdera färgegenskaperna hos ett diagramobjekt där färgen definieras med en röd komponent **r**, en grön komponent **g** och en blå komponent **b**, med en alfafaktor (opacitet) på **alpha**.

```
ARGB (alpha, r, g, b)
```

HSL

HSL() används i uttryck för att ange eller utvärdera färgegenskaperna för ett diagramobjekt där färgen definieras av värden för **hue**, **saturation** och **luminosity**, mellan 0 och 1.

```
HSL (hue, saturation, luminosity)
```

RGB

RGB() returnerar ett heltal som motsvarar färgkoden för den färg som definieras av tre parametrar: den röda komponenten r, den gröna komponenten g och den blå komponenten b. Komponenterna måste ha heltalsvärden mellan 0 och 255. Funktionen kan användas i uttryck för att ställa in eller utvärdera färgegenskaper för ett diagramobjekt.

```
RGB (r, g, b)
```

Colormix1

Colormix1() används i uttryck för att returnera en ARGB-färgrepresentation från en tvåfärgad toning baserad på ett värde mellan 0 och 1.

```
Colormix1 (Value , ColorZero , ColorOne)
```

Value är ett reellt tal mellan 0 och 1.

- Om Value = 0 returneras ColorZero .
- Om Value = 1 returneras ColorOne .
- Om 0 < Value < 1 returneras motsvarande färgblandning.

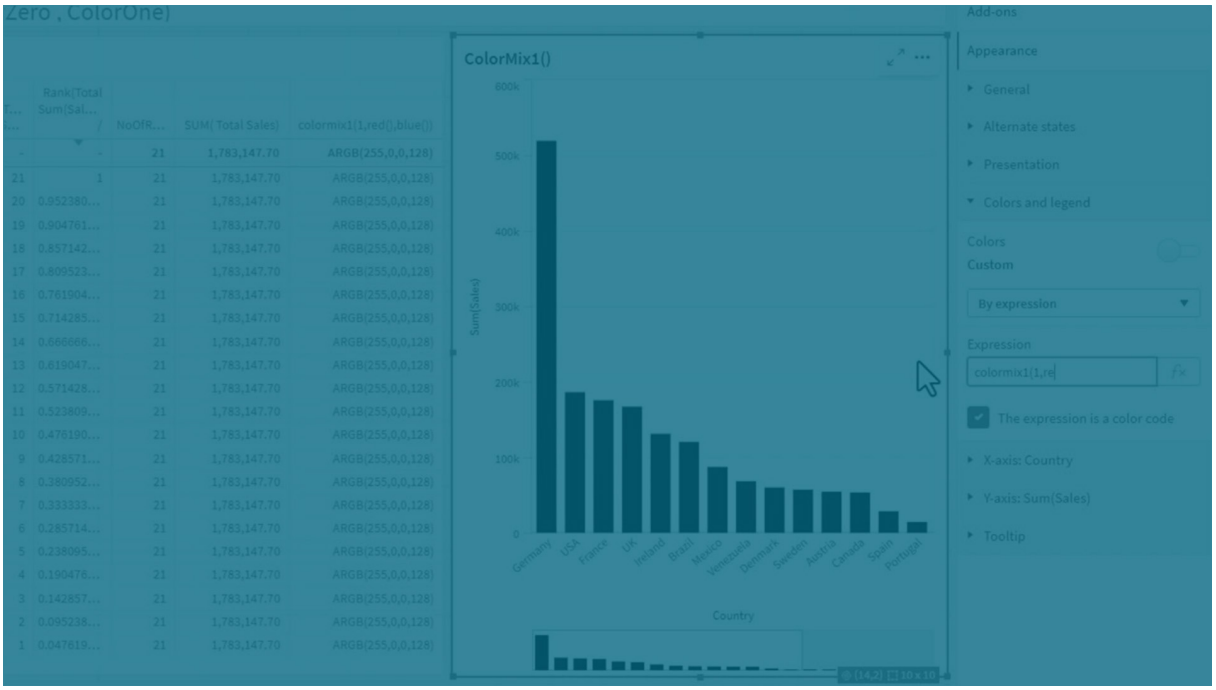
ColorZero är en giltig RGB-färgrepresentation för den färg som ska associeras med intervallets lägsta värde.

ColorOne är en giltig RGB-färgrepresentation för den färg som ska associeras med intervallets högsta värde.

Exempel:

```
Colormix1(0.5, red(), blue())
```
returnerar:

```
ARGB(255,64,0,64) (purple)
```



Colormix2

Colormix2() används i uttryck för att returnera en ARGB-färgrepresentation från en tvåfärgad toning baserat på ett värde mellan -1 och 1, med möjlighet att ange en mellanliggande färg för intervallets mitt (0)

```
Colormix2 (Value ,ColorMinusOne , ColorOne[ , ColorZero])
```
Value är ett reellt tal mellan -1 och 1.

- Om Value = -1 returneras den första färgen.
- Om Value = 1 returneras den andra färgen.
- Om -1 < Value< 1 returneras den lämpliga färgblandningen.

ColorMinusOne är en giltig RGB-färgrepresentation för den färg som ska associeras med intervallets lägsta värde.

ColorOne är en giltig RGB-färgrepresentation för den färg som ska associeras med intervallets högsta värde.

ColorZero är en alternativ giltig RGB-färgrepresentation för den färg som ska associeras med intervallets mittersta värde.

SysColor

SysColor() returnerar ARGB-färgåtergivningen för Windows systemfärg nr, där nr motsvarar parametern till Windows API-funktion **GetSysColor(nr)**.

```
SysColor (nr)
```

ColorMapHue

ColorMapHue() returnerar ett ARGB-värde av en färg från en färgkarta som varierar nyanskomponenten för HSV-färgmodellen. Färgkartan börjar med röd, övergår sedan till gul, grön, cyan, blå, magenta och återgår till röd. x måste vara ett värde mellan 0 och 1.

```
ColorMapHue (x)
```

ColorMapJet

ColorMapJet() returnerar ett ARGB-värde för en färg från en färgkarta som börjar med blå, övergår till cyan, gul och orange och återgår till röd. x måste vara ett värde mellan 0 och 1.

```
ColorMapJet (x)
```

Fördefinierade färgfunktioner

Följande funktioner kan användas i uttryck för fördefinierade färger. Varje funktion returnerar en RGB-färgrepresentation.

En parameter för alfafaktorn kan också ges, vilket leder till att en ARGB-färgrepresentation returneras. Alfafaktorn 0 motsvarar full genomskinlighet. Alfafaktorn 255 motsvarar full opacitet. Om inget värde anges för alfa antas det vara 255.

Fördefinierade färgfunktioner

Färgfunktion	RGB -värde
black ([alpha])	(0,0,0)
blue([alpha])	(0,0,128)
brown([alpha])	(128,128,0)
cyan([alpha])	(0,128,128)
darkgray([alpha])	(128,128,128)
green([alpha])	(0,128,0)
lightblue([alpha])	(0,0,255)

lightcyan([alpha])	(0,255,255)
lightgray([alpha])	(192,192,192)
lightgreen([alpha])	(0,255,0)
lightmagenta([alpha])	(255,0,255)
lightred([alpha])	(255,0,0)
magenta([alpha])	(128,0,128)
red([alpha])	(128,0,0)
white([alpha])	(255,255,255)
yellow([alpha])	(255,255,0)

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Blue()`	RGB(0,0,128)
`Blue(128)`	ARGB(128,0,0,128)

ARGB

ARGB() används i uttryck för att ange eller utvärdera färgegenskaperna hos ett diagramobjekt där färgen definieras med en röd komponent **r**, en grön komponent **g** och en blå komponent **b**, med en alfafaktor (opacitet) på **alpha**.

Syntax:

```
ARGB(alpha, r, g, b)
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
alpha	Genomskinlighet i intervallet 0-255. 0 representerar full transparens och 255 motsvarar helt avsaknad av transparens.
r, g, b	Röda, gröna och blå komponentvärden. En färgkomponent med värdet 0 motsvarar ingen påverkan och värdet 255 motsvarar full påverkan.

Alla argument måste vara uttryck som resulterar i heltal i intervallet 0 till 255.

Om den numeriska komponenten tolkas och formateras i hexadecimal notation blir det lättare att se värdena på färgkomponenterna. Exempelvis har ljusgrön numret 4 278 255 360, som i hexadecimal notation är FF00FF00. De första två positionerna 'FF' (255) anger **alpha**-kanalen. De följande två positionerna "00" visar mängden för **röd**, de därpå följande två positionerna "FF" visar mängden för **grön** och de två sista positionerna "00" visar mängden för **blå**.

Exempel: diagramuttryck

Exempel	Resultat
`ARGB(128, 0, 128, 0)`	Returnerar färgen grön med semitransparens.

Exempel – grunderna i ARGB

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Product`
  - `Sales`

Laddningsskript

```
Example:
LOAD * Inline [
Product, Sales
Product A, 200
Product B, 500
Product C, 100
Product D, 300
Product E, 700
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Product`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.

I panelen **Egenskaper** för mätvärdet `Sum(Sales)` anger du följande uttryck som **Bakgrundsfärguttryck**: Detta kommer att tillämpa cellfärg enligt försäljningskalkylen för produkten.

- `=If(Sales > 400, ARGB(128, 0, 128, 0), If(Sales >= 200 and Sales <= 400, ARGB(128, 255, 255, 0), ARGB(128, 255, 0, 0)) )`

Results table

Product	Sum(Sales)
Product A	200
Product B	500
Product C	100
Product D	300
Product E	700

I detta exempel ska du observera följande:

- Försäljningsvärden över `400` är markerade med grön bakgrund.
- Försäljningsvärden mellan `200` och `400` är markerade med gul bakgrund.
- Alla övriga försäljningsvärden är markerade med röd bakgrund.

Exempel – markera vinstdata i ett diagram

Diagramuttryck

Översikt

En datauppsättning innehåller försäljningssiffror per kund. Vi vill analysera uppgifterna och lyfta fram vinstmarginalerna beroende på om de är låga, måttliga eller höga.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Customer`
  - `SalesValue`
  - `Cost`
  - `ProfitMargin`

Laddningsskript

```
Example:
LOAD * Inline [
Customer, SalesValue, Cost, ProfitMargin
Customer A, 500, 300, 40%
Customer B, 800, 750, 6.25%
Customer C, 400, 300, 25%
Customer D, 600, 550, 8.33%
```

```
Customer E, 900, 700, 22.22%
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=Sum(SalesValue)`, för att beräkna summan av försäljningen.
- `=Sum(Cost)`, för att beräkna det totala värdet av kostnaden för sålda varor.
- `=Sum(ProfitMargin)`, för att summera vinstmarginalen. Om du vill visa värdet i procent väljer du **Nummer > Enkel formatering > 12,34%** under **Nummerformatering**.

I panelen **Egenskaper** för mätvärdet `Sum(ProfitMargin)` anger du följande uttryck som **Bakgrundsfärguttryck**:

```
If(ProfitMargin > 0.3, ARGB(128, 0, 128, 0),   // Green for high-profit customers
  If(ProfitMargin >= 0.1 and ProfitMargin <= 0.3, ARGB(128, 255, 165, 0), // Orange for
moderate-profit customers
    ARGB(128, 255, 0, 0) // Red for low-profit customers
  )
)
```

I panelen **Utseende**, under **Presentation**, stänger du av **Totalsummor**.

Results table

Customer	Sum(SalesValue)	Sum(Cost)	Sum(Profit Margin)
Customer A	500	300	40.00%
Customer B	800	750	6.25%
Customer C	400	300	25.00%
Customer D	600	550	8.33%
Customer E	900	700	22.22%

Utdata från funktionen `ARGB` återger följande färger enligt kundens vinstmarginal:

- Grön, `ARGB(128, 0, 128, 0)`, för kunder med vinstmarginaler högre än 30%.
- Orange, `ARGB(128, 255, 165, 0)`, för kunder med måttliga vinstmarginaler mellan 10% och 30%.
- Röd, `ARGB(128, 255, 0, 0)`, för kunder med en vinstmarginal under 10%.

Exempel – ARGB diagramscenario

Diagramuttryck

Översikt

En datauppsättning innehåller faktiska och prognosticerade försäljningsdata per land. En försäljningschef vill visa data i ett stapeldiagram och lyfta fram de försäljningssiffror som är högre än prognosvärdet för perioden.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `Country`
    - `ActualSales`
    - `ProjectedSales`

Laddningsskript

```
Example:
Load * Inline
[Country, ActualSales, ProjectedSales
Sweden, 100000, 50000
Germany, 125000, 175000
Norway, 74850, 68500
Ireland, 45000, 48000
Sweden, 98000, 50000
Germany, 115000, 175000
Norway, 71850, 68500
Ireland, 31000, 48000
];
```

Resultat

Ladda data och öppna ett ark. Skapa ett nytt stapeldiagram och lägg till det här fältet som en dimension:

- `Country`

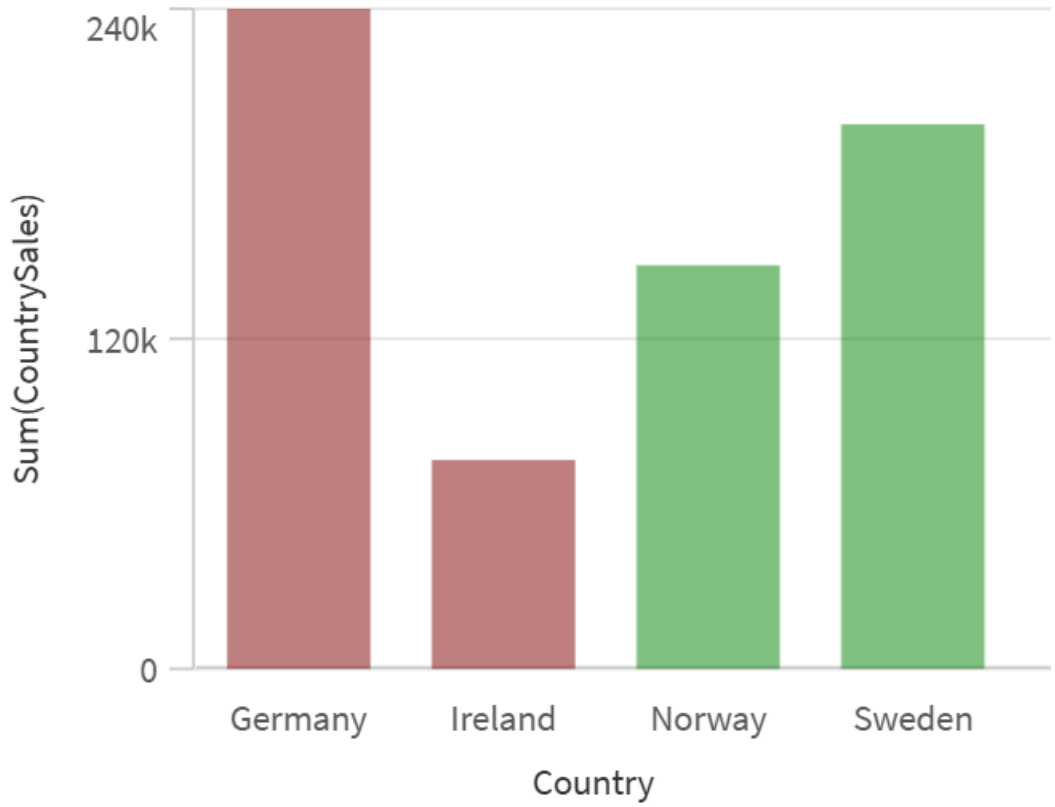
Skapa följande mått:

- `=Sum(ActualSales)`, för att beräkna det totala värdet av försäljning för landet.

Lägg till ett färguttryck för att jämföra faktisk försäljning med prognostiserad. I egenskapspanelen, under **Utseende** > **Färger och teckenförklaring**, inaktiverar du av **Färger**: **Auto** för att växla till **Anpassad** och välj sedan **Efter uttryck** i listan. Ange följande uttryck:

- `=If(Sum(ActualSales)>Sum(ProjectedSales),ARGB(128,0,128,0),ARGB(128,128,0,0))`, för att i grönt markera förekomster där faktisk försäljning överstiger prognostiserad försäljning och i rött de tillfällen då den prognostiserade försäljningen överstiger den faktiska.

Resultat: Stapeldiagram över försäljning per land



Vi kan se att stapeln är grön för de länder vars försäljning överstiger det prognostiserade värdet, medan de länder vars försäljning understiger det prognostiserade värdet visas i rött. Till exempel är `Sweden` grönt eftersom försäljningsvärdet på `98,000` är större än det prognostiserade värdet på `50,000`.

RGB

RGB() returnerar ett heltal som motsvarar färgkoden för den färg som definieras av tre parametrar: den röda komponenten r, den gröna komponenten g och den blå komponenten b. Komponenterna måste ha heltalsvärden mellan 0 och 255. Funktionen kan användas i uttryck för att ställa in eller utvärdera färgegenskaper för ett diagramobjekt.

Syntax:

```
RGB (r, g, b)
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
r, **g**, **b**	Röda, gröna och blå komponentvärden. En färgkomponent med värdet 0 motsvarar ingen påverkan och värdet 255 motsvarar full påverkan.

Alla argument måste vara uttryck som resulterar i heltal i intervallet 0 till 255.

Om den numeriska komponenten tolkas och formateras i hexadecimal notation blir det lättare att se värdena på färgkomponenterna. Exempelvis har ljusgrön numret 4 278 255 360, som i hexadecimal notation är FF00FF00. De första två positionerna 'FF' (255) anger **alpha**-kanalen. I funktionerna **RGB** och **HSL** är denna alltid "FF" (ogenomskinlig). De följande två positionerna "00" visar mängden för **röd**, de därpå följande två positionerna "FF" visar mängden för **grön** och de två sista positionerna "00" visar mängden för **blå**.

Exempel: diagramuttryck

Exempel	Resultat
`If(Sum(Sales)>Sum(Budget),RGB (255,0,0),RGB(100,80,120))`	Returnerar lila eller rött beroende på värdet av jämförelsen mellan `Sales` och `Budget`.

Exempel – grunderna i RGB

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Product`
  - `Sales`

Laddningsskript

```
Example:
LOAD * Inline [
Product, Sales
Product A, 200
Product B, 500
Product C, 100
Product D, 300
```

```
Product E, 700
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Product`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.

I panelen **Egenskaper** för mätvärdet `Sum(Sales)` anger du följande uttryck som **Bakgrundsfärguttryck**: Detta kommer att tillämpa cellfärg enligt försäljningskalkylen för produkten.

- `=If(Sales > 400, RGB(0, 128, 0), If(Sales >= 200 and Sales <= 400, RGB( 255, 255, 0), RGB(255, 0, 0)) )`

Results table

Product	Sum(Sales)
Product A	200
Product B	500
Product C	100
Product D	300
Product E	700

I detta exempel ska du observera följande:

- Försäljningsvärden över `400` är markerade med grön bakgrund.
- Försäljningsvärden mellan `200` och `400` är markerade med gul bakgrund.
- Alla övriga försäljningsvärden är markerade med röd bakgrund.

Exempel – markera vinstdata i ett diagram

Diagramuttryck

Översikt

En datauppsättning innehåller försäljningssiffror per kund. Vi vill analysera uppgifterna och lyfta fram vinstmarginalerna beroende på om de är låga, måttliga eller höga.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Customer`
  - `SalesValue`
  - `Cost`
  - `ProfitMargin`

Laddningsskript

```
Example:
LOAD * Inline [
Customer, SalesValue, Cost, ProfitMargin
Customer A, 500, 300, 40%
Customer B, 800, 750, 6.25%
Customer C, 400, 300, 25%
Customer D, 600, 550, 8.33%
Customer E, 900, 700, 22.22%
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=Sum(SalesValue)`, för att beräkna summan av försäljningen.
- `=Sum(Cost)`, för att beräkna det totala värdet av kostnaden för sålda varor.
- `=Sum(ProfitMargin)`, för att summera vinstmarginalen. Om du vill visa värdet i procent väljer du **Nummer > Enkel formatering > 12,34%** under **Nummerformatering**.

I panelen **Egenskaper** för mätvärdet `Sum(ProfitMargin)` anger du följande uttryck som **Bakgrundsfärguttryck**:

```
If(ProfitMargin > 0.3, RGB(0, 128, 0),   // Green for high-profit customers
  If(ProfitMargin >= 0.1 and ProfitMargin <= 0.3, RGB(255, 165, 0), // Orange for moderate-
profit customers
    RGB(255, 0, 0) // Red for low-profit customers
  )
)
```

I panelen **Utseende**, under **Presentation**, stänger du av **Totalsummor**.

Results table

Customer	Sum(SalesValue)	Sum(Cost)	Sum(Profit Margin)
Customer A	500	300	40.00%
Customer B	800	750	6.25%

Customer	Sum(SalesValue)	Sum(Cost)	Sum(Profit Margin)
Customer C	400	300	25.00%
Customer D	600	550	8.33%
Customer E	900	700	22.22%

Utdata från funktionen RGB återger följande färger enligt kundens vinstmarginal:

- Grön, `RGB(0, 128, 0)`, för kunder med vinstmarginaler högre än 30%.
- Orange, `RGB(255, 165, 0)`, för kunder med måttliga vinstmarginaler mellan 10% och 30%.
- Röd, `RGB(255, 0, 0)`, för kunder med en vinstmarginal under 10%.

Exempel – RGB diagramscenario

Diagramuttryck

Översikt

En datauppsättning innehåller faktiska och prognosticerade försäljningsdata per land. En försäljningschef vill visa data i ett stapeldiagram och lyfta fram de försäljningssiffror som är högre än prognosvärdet för perioden.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Country`
  - `ActualSales`
  - `ProjectedSales`

Laddningsskript

```
Example:
Load * Inline
[Country, ActualSales, ProjectedSales
Sweden, 100000, 50000
Germany, 125000, 175000
Norway, 74850, 68500
Ireland, 45000, 48000
Sweden, 98000, 50000
Germany, 115000, 175000
Norway, 71850, 68500
Ireland, 31000, 48000
];
```

Resultat

Ladda data och öppna ett ark. Skapa ett nytt stapeldiagram och lägg till det här fältet som en dimension:

- `Country`

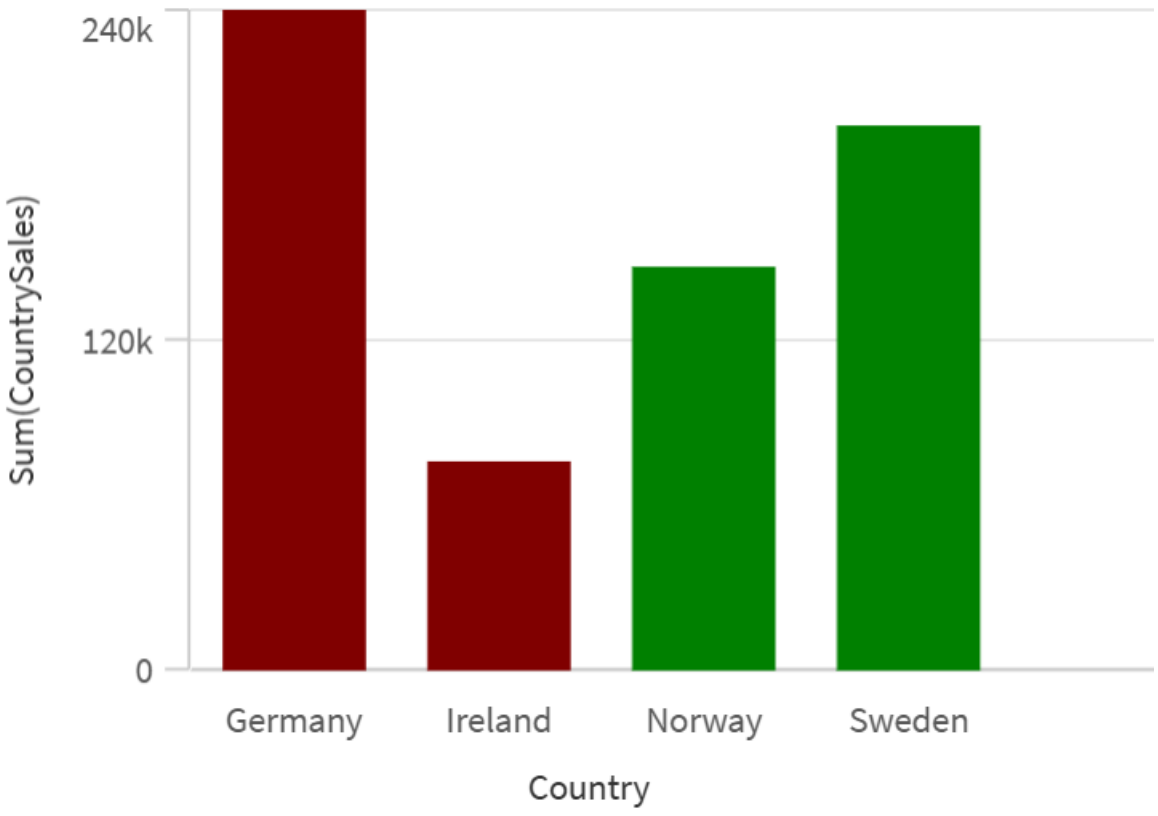
Skapa följande mått:

- `=Sum(ActualSales)`, för att beräkna det totala värdet av försäljning för landet.

Lägg till ett anpassat färguttryck för att jämföra faktisk försäljning med prognostiserad. I egenskapspanelen, under **Utseende** > **Färger och teckenförklaring**, inaktiverar du av **Färger**: **Auto** för att växla till **Anpassad** och välj sedan **Efter uttryck** i listan. Ange följande uttryck:

- `=If(Sum(ActualSales)>Sum(ProjectedSales),RGB(0,128,0),RGB(128,0,0))`, för att i grönt markera förekomster där faktisk försäljning överstiger prognostiserad försäljning och i rött de tillfällen då den prognostiserade försäljningen överstiger den faktiska.

Resultat: Stapeldiagram över försäljning per land



Vi kan se att stapeln är grön för de länder vars försäljning överstiger det prognostiserade värdet, medan de länder vars försäljning understiger det prognostiserade värdet visas i rött. Till exempel är `Sweden` grönt eftersom försäljningsvärdet på `98,000` är större än det prognostiserade värdet på `50,000`.

HSL

HSL() används i uttryck för att ange eller utvärdera färgegenskaperna för ett diagramobjekt där färgen definieras av värden för **hue**, **saturation** och **luminosity**, mellan 0 och 1.

Syntax:

```
HSL (hue, saturation, luminosity)
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
hue, **saturation**, **luminosity**	Hue, saturation och luminosity komponentvärden mellan 0 och 1.

Alla argument måste vara uttryck som resulterar i heltal i intervallet 0 till 1.

Om den numeriska komponenten tolkas och formateras i hexadecimal notation blir det lättare att se värdena RGB på färgkomponenterna. Exempelvis har ljusgrön numret 4 278 255 360, som i hexadecimal notation är FF00FF00 och RGB (0,255,0). Detta motsvarar HSL (80/240, 240/240, 120/240) , ett HSL - värde på (0.33, 1, 0.5).

Exempel: diagramuttryck

Exempel	Resultat
`If(Sum(Sales)>Sum(Budget),HSL( 0.33, 1, 0.25),HSL(0, 1, 0.5))`	Returnerar lila eller rött beroende på värdet av jämförelsen mellan `Sales` och `Budget`.

Exempel – grunderna i HSL

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Product`
  - `Sales`

Laddningsskript

```
Example:
LOAD * Inline [
```

```
Product, Sales
Product A, 200
Product B, 500
Product C, 100
Product D, 300
Product E, 700
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Product`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.

I panelen **Egenskaper** för mätvärdet `Sum(Sales)` anger du följande uttryck som **Bakgrundsfärguttryck**: Detta kommer att tillämpa cellfärg enligt försäljningskalkylen för produkten.

- `=If(Sales > 400, HSL(.33, 1, 0.25), If(Sales >= 200 and Sales <= 400, HSL(.167, 1, 0.5), HSL(0, 1, 0.5)) )`

Results table

Product	Sum(Sales)
Product A	200
Product B	500
Product C	100
Product D	300
Product E	700

I detta exempel ska du observera följande:

- Försäljningsvärden över `400`, som `Product B` och `E`, är markerade med grön bakgrund.
- Försäljningsvärden mellan `200` och `400`, som `Product A` och `D`, är markerade med gul bakgrund.
- Alla övriga försäljningsvärden är markerade med röd bakgrund.

Exempel – markera vinstdata i ett diagram

Diagramuttryck

Översikt

En datauppsättning innehåller försäljningssiffror. Vi vill analysera uppgifterna och lyfta fram vinstmarginalerna beroende på om de är låga, måttliga eller höga.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Customer`
  - `SalesValue`
  - `Cost`
  - `ProfitMargin`

Laddningsskript

```
Example:
LOAD * Inline [
Customer, SalesValue, Cost, ProfitMargin
Customer A, 500, 300, 40%
Customer B, 800, 750, 6.25%
Customer C, 400, 300, 25%
Customer D, 600, 550, 8.33%
Customer E, 900, 700, 22.22%
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=Sum(SalesValue)`, för att beräkna summan av försäljningen.
- `=Sum(Cost)`, för att beräkna det totala värdet av kostnaden för sålda varor.
- `=Sum(ProfitMargin)`, för att summera vinstmarginalen. Om du vill visa värdet i procent väljer du **Nummer > Enkel formatering > 12,34%** under **Nummerformatering**.

I panelen **Egenskaper** för mätvärdet `Sum(ProfitMargin)` anger du följande uttryck som **Bakgrundsfärguttryck**:

```
If(ProfitMargin > 0.3, HSL( 0.33, 1, 0.25),   // Green for high-profit customers
  If(ProfitMargin >= 0.1 and ProfitMargin <= 0.3, HSL( 0.11, 1, 0.5),   // Orange for
moderate-profit customers
    HSL(0, 1, 0.5)   // Red for low-profit customers
  )
)
```

I panelen **Utseende**, under **Presentation**, stänger du av **Totalsummor**.

Results table

Customer	Sum(SalesValue)	Sum(Cost)	Sum([Profit Margin])
Customer A	500	300	40.00%

Customer	Sum(SalesValue)	Sum(Cost)	Sum([Profit Margin])
Customer B	800	750	6.25%
Customer C	400	300	25.00%
Customer D	600	550	8.33%
Customer E	900	700	22.22%

Utdata från funktionen `HSL` återger följande färger enligt kundens vinstmarginal:

- Grön, `HSL( 0.33, 1, 0.25)`, för kunder med vinstmarginaler högre än 30%.
- Orange, `HSL( 0.11, 1, 0.5)`, för kunder med måttliga vinstmarginaler mellan 10% och 30%.
- Röd, `HSL(0, 1, 0.5)`, för kunder med en vinstmarginal under 10%.

Exempel – HSL diagramscenario

Diagramuttryck

Översikt

En datauppsättning innehåller faktiska och prognosticerade försäljningsdata per land. En försäljningschef vill visa data i ett stapeldiagram och lyfta fram de försäljningssiffror som är högre än prognosvärdet för perioden.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Country`
  - `ActualSales`
  - `ProjectedSales`

Laddningsskript

```
Example:
Load * Inline
[Country, ActualSales, ProjectedSales
Sweden, 100000, 50000
Germany, 125000, 175000
Norway, 74850, 68500
Ireland, 45000, 48000
Sweden, 98000, 50000
Germany, 115000, 175000
Norway, 71850, 68500
Ireland, 31000, 48000
];
```

Resultat

Ladda data och öppna ett ark. Skapa ett nytt stapeldiagram och lägg till det här fältet som en dimension:

- `Country`

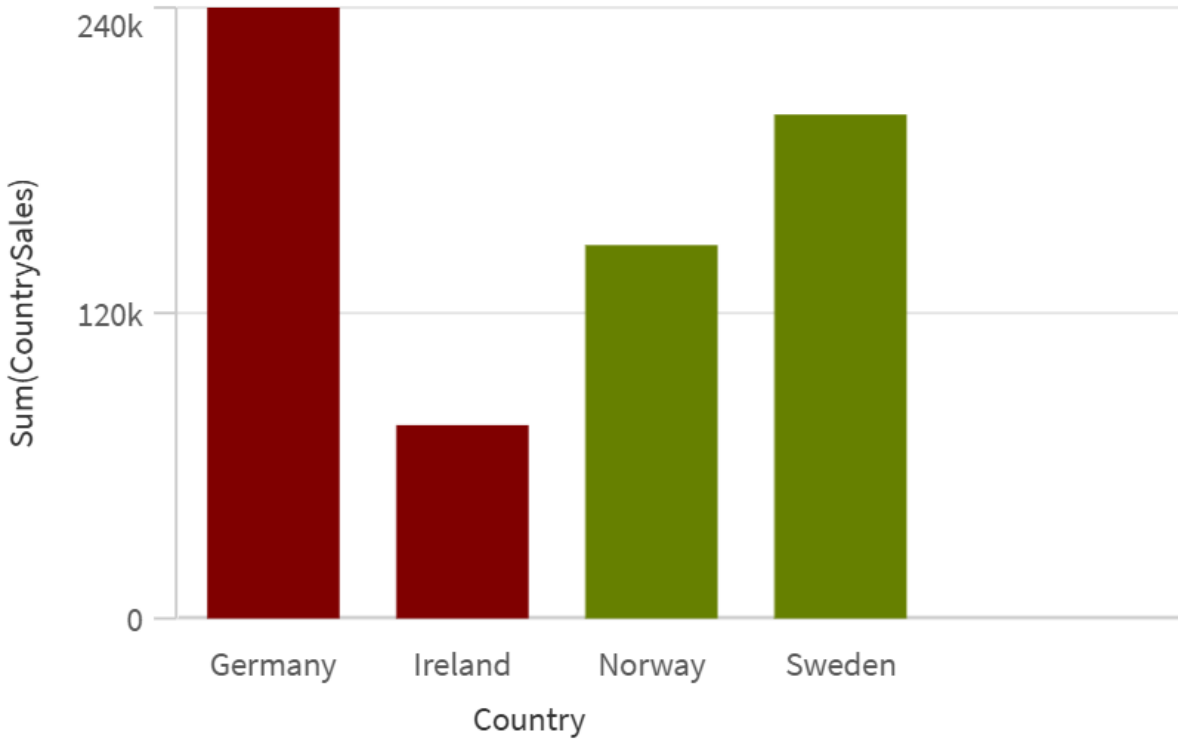
Skapa följande mått:

- `=Sum(ActualSales)`, för att beräkna det totala värdet av försäljning för landet.

Lägg till ett färguttryck för att jämföra faktisk försäljning med prognostiserad. I egenskapspanelen, under **Utseende** > **Färger och teckenförklaring**, inaktiverar du av **Färger**: **Auto** för att växla till **Anpassad** och välj sedan **Efter uttryck** i listan. Ange följande uttryck:

- `=If(Sum(ActualSales)>Sum(ProjectedSales),HSL(1.2,1.0,0.25),HSL(0, 1.0, 0.25))`, för att i grönt markera förekomster där faktisk försäljning överstiger prognostiserad försäljning och i rött de tillfällen då den prognostiserade försäljningen överstiger den faktiska.

Resultat: Stapeldiagram över försäljning per land



Vi kan se att stapeln är grön för de länder vars försäljning överstiger det prognostiserade värdet, medan de länder vars försäljning understiger det prognostiserade värdet visas i rött. Till exempel är `Sweden` grönt eftersom försäljningsvärdet på `98,000` är större än det prognostiserade värdet på `50,000`.

8.5 Villkorsfunktioner

Alla villkorsfunktioner utvärderar ett villkor. Sedan returnerar de olika svar beroende på villkorets värde. Funktionerna kan användas i dataladdningsskriptet och diagramuttryck.

Villkorsfunktioner – en översikt

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

alt

alt-funktionen returnerar den första av parametrarna som har ett giltigt numeriskt värde. Om ingen av parametrarna har ett giltigt numeriskt värde, returneras den sista parametern. Det finns inga begränsningar vad gäller antalet parametrar.

```
alt (expr1[ , expr2 , expr3 , ...] , else)
```

class

class-funktionen kopplar den första parametern till ett klassintervall. Resultatet är ett dualt värde med a<=x<b som textvärde, där a och b är de övre och nedre gränserna av bin, och det nedre gränsvärdet ett numeriskt värde.

```
class (expression, interval [ , label [ , offset ]])
```

coalesce

Coalesce-funktionen returnerar den första av parametrarna som har en giltig non-NULL-representation. Det finns inga begränsningar vad gäller antalet parametrar.

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

if

if-funktionen returnerar ett värde beroende på om villkoret som hör samman med funktionen utvärderas som True eller False.

```
if (condition , then , else)
```

match

Funktionen **match** jämför den första parametern med alla de följande och returnerar den numeriska platsen för uttryck som matchar. Jämförelsen är skiftlägeskänslig.

```
match ( str, expr1 [ , expr2,...exprN ])
```

mixmatch

Funktionen **mixmatch** jämför den första parametern med alla de följande och returnerar den numeriska platsen för uttryck som matchar. Jämförelsen är inte skiftlägeskänslig och är okänslig för de japanska teckensystemen hiragana och katakana.

```
mixmatch ( str, expr1 [ , expr2,...exprN ])
```

pick

Funktionen pick returnerar det *n*:te uttrycket i listan.

```
pick (n, expr1[ , expr2,...exprN])
```

wildmatch

Funktionen **wildmatch** jämför den första parametern med alla de följande och returnerar antalet uttryck som matchar. Den gör det möjligt att använda jokertecken (***** och **?**) i jämförelsesträngarna. ***** matchar alla följder av tecken. **?** matchar alla enstaka tecken. Jämförelsen är inte skiftlägeskänslig och är okänslig för de japanska teckensystemen hiragana och katakana.

```
wildmatch ( str, expr1 [ , expr2,...exprN ])
```

alt

alt-funktionen returnerar den första av parametrarna som har ett giltigt numeriskt värde. Om ingen av parametrarna har ett giltigt numeriskt värde, returneras den sista parametern. Det finns inga begränsningar vad gäller antalet parametrar.

Syntax:

```
alt(expr1[ , expr2 , expr3 , ...] , else)
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
expr1	Det första uttrycket som ska kontrolleras om det har ett giltigt numeriskt värde.
expr2	Det andra uttrycket som ska kontrolleras om det har ett giltigt numeriskt värde.
expr3	Det tredje uttrycket som ska kontrolleras om det har ett giltigt numeriskt värde.
else	Värde som ska returneras om ingen av de föregående parametrarna har ett giltigt numeriskt värde.

Funktionen alt används ofta tillsammans med funktioner för tal- eller datumtolkning. På så vis kan Qlik Sense testa olika datumformat i en prioriterad ordningsföljd. Den kan även användas för att hantera NULL-värden i numeriska uttryck.

Exempel: diagramuttryck

Exempel	Resultat

`alt( date#( dat , 'YYYY/MM/DD' ),`
 	
`date#( dat , 'MM/DD/YYYY' ),`
 	
`date#( dat , 'MM/DD/YY' ),`
 	
`'No valid date' )`
	Uttrycket testar om fältet date innehåller ett datum enligt något av de tre angivna datumformaten. Om så är fallet, returneras ett dualt värde som innehåller den ursprungliga strängen och ett giltigt numeriskt värde som motsvarar ett datum. Om ingen matchning hittas returneras texten `No valid date` (utan giltig numerisk representation).

`alt(Sales,0) + alt(Margin,0)`	Det här uttrycket lägger till fälten `Sales` och `Margin`, och ersätter eventuella saknade värden (NULL) med `0`.

Exempel – grunderna i alt

Diagramuttryck

Översikt

En datauppsättning innehåller ofullständiga försäljnings- och prognosdata. Du vill ersätta saknade värden med giltiga alternativa värden för att fylla i luckorna.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `ID`
    - `Sales`
    - `Forecast`

Laddningsskript

```
Example:
LOAD * INLINE [
ID, Sales, Forecast
1, 500, 450
2, , 300
3, 200, 250
4, , 400
5, 350,
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ID`
- `Sales`
- `Forecast`

Skapa följande mått:

- `=Sum(alt(Sales, Forecast))` för att returnera en giltig talrepresentation genom att ersätta saknade värden med giltiga alternativa värden.

Resultattabell

ID	Sales	Forecast	Sum(alt(Sales, Forecast))
Totals			**1750**
1	500	450	500
2		300	300
3	200	250	200
4		400	400
5	350		350

Funktionen `alt` returnerar `Forecast`-värdet i måttberäkningen för `ID`-raderna med saknade `Sales`-värden. Genom att ersätta saknade värden med giltiga alternativa värden kan du generera en fullständigt ifylld datauppsättning.

Exempel – Använda alt för att ersätta saknade värden

Diagramuttryck

Översikt

Ett företag samlar in temperaturdata från flera sensorer. Du vill ersätta saknade data från en sensor med data från den andra sensorn.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Time`
  - `Temp_Sensor1`
  - `Temp_Sensor2`

Laddningsskript

```
Example:
LOAD * INLINE [
Time, Temp_Sensor1, Temp_Sensor2
08:00, 23.5, 24.0
09:00, , 23.8
10:00, 24.1,
11:00, , 24.5
12:00, 23.9, 24.1
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Time`
- `Temp_Sensor1`
- `Temp_Sensor2`

Skapa följande mått:

- `=Sum(alt(Temp_Sensor1, Temp_Sensor2))` för att returnera en giltig talrepresentation.

Resultattabell

Time	Temp_Sensor1	Temp_Sensor2	Sum(alt(Temp_Sensor1, Temp_Sensor2))
Totals			**119.8**
08:00	23.5	24.0	23.5

Time	Temp_Sensor1	Temp_Sensor2	Sum(alt(Temp_Sensor1, Temp_Sensor2))
Totals			**119.8**
09:00		23.8	23.8
10:00	24.1		24.1
11:00		24.5	24.5
12:00	23.9	24.1	23.9

Om man tittar på resultaten har funktionen `alt` ersatt saknade sensordata med giltiga alternativa resultat från den andra sensorn. Detta kompletterar datauppsättningen genom att fylla i luckor.

Exempel – alt-scenario

Diagramuttryck

Översikt

Det här scenariot visar olika sätt att använda funktionen `alt` för att ersätta saknade värden eller nollvärden.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ProductID`
  - `ProductName`
  - `SalesAmount`
  - `ReturnAmount`
  - `Profit`
  - `Discount`
  - `Category`
  - `Year`
  - `Region`
  - `CustomerRating`

Laddningsskript

```
Set NullInterpret = '';
Example:
LOAD * INLINE [
ProductID, ProductName, SalesAmount, ReturnAmount, Profit, Discount, Category, Year, Region,
CustomerRating
101, "Laptop", 1200, 200, 500, , "Electronics", 2023, "North",
102, "Tablet", 800, , 300, 50, "Electronics", 2023, "South", 4
103, "Smartphone", 1500, , 600, 100, , 2022, "East", 5
104, "Printer", 400, , , 20, "Accessories", 2021, , 3
```

```
105, , 1000, 100, , 30, "Electronics", 2023, "West", 4
106, "Monitor", , 150, 250, , "Accessories", 2022, "North", 5
107, "Keyboard", 200, , 80, 5, , 2022, "South", 2
108, "Mouse", , 50, , , "Accessories", , "East",
109, , 600, , , , "Electronics", 2023, , 4
110, "Headphones", 300, 30, 100, , "Accessories", 2021, "West", 3
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `ProductID`

Skapa följande beräknade dimension:

- `=Coalesce( ProductName, 'Unknown Product')`, för att tillhandahålla beskrivningen `Unknown Product` för alla produkter som inte har något värde för `ProductName`.

Skapa följande mått:

- `=Sum(alt(SalesAmount, ReturnAmount, Profit))`, för att returnera ett giltigt nummer om ingen `SalesAmount` hittas.
- `=avg(alt(CustomerRating, 3))`, för att returnera ett giltigt standardiserat kundbetyg om inget sådant ursprungligen angetts.
- `=Sum(alt(Discount, SalesAmount * 0.05))`, för att returnera en giltig rabatt.
- `=Sum(alt(Profit, SalesAmount * 0.1))`, för att returnera ett giltigt vinstvärde.
- `=Sum(alt(SalesAmount, avg(TOTAL SalesAmount)))`, för att returnera ett giltigt försäljningsvärde.
- `=Sum(alt(Discount, avg(TOTAL Discount)))`, för att returnera ett giltigt rabattvärde.

Resultattabell

Produkt-ID	Coalesce(Produktnamn, 'Okänd produkt')	Summa(alt (Försäljningsbelopp, Returbelopp, Vinst))	medel (alt (Kundbetyg, 3))	Summa(alt (Rabatt, Försäljningsbelopp * 0.05))	Summa(alt (Vinst, Försäljningsbelopp * 0.1))	Summa(alt (Försäljningsbelopp, medel (TOTALT Försäljningsbelopp)))	Summa (alt (Rabatt, medel (TOTAL Rabatt)))
Totalvärden		**6200**	**3.6**	**310**	**2030**	**7500**	**410**
101	Bärbar dator	1200	3	60	500	1200	41

Produkt-ID	Coalesce(Produktnamn, 'Okänd produkt')	Summa(alt (Försäljningsbelopp, Returbelopp, Vinst))	medel (alt (Kundbetyg, 3))	Summa(alt (Rabatt, Försäljningsbelopp * 0.05))	Summa(alt (Vinst, Försäljningsbelopp * 0.1))	Summa(alt (Försäljningsbelopp, medel (TOTALT Försäljningsbelopp)))	Summa (alt (Rabatt, medel (TOTAL Rabatt)))
Totalvärden		**6200**	**3.6**	**310**	**2030**	**7500**	**410**
102	Surfplatta	800	4	50	300	500	50
103	Smarttelefon	1500	5	100	600	800	100
104	Skrivare	400	3	20	40	400	20
105	Okänd produkt	1000	4	30	100	1000	30
106	Övervaka	150	5	0	250	750	41
107	Tangentbord	200	2	5	80	200	5
108	Mus	50	3	0	0	750	41
109	Okänd produkt	600	4	30	60	600	41
110	Hörlurar	300	3	15	100	300	41

Funktionen `alt` säkerställer att alternativa värden kan beräknas och returneras när värden saknas. Observera följande resultat:

- I kolumn 2 används funktionen `Coalesce` för att returnera värdet `Unknown Product` för alla null-värden. Funktionen `Alt` kan inte användas eftersom den endast gäller för numeriska värden.
- Kolumn 3 returnerar värdet för `SalesAmount`. Om inget värde hittas returneras värdet `ReturnAmount`. Om inget `ReturnAmount`-värde hittas returneras värdet `Profit`. Detta bidrar till att säkerställa att kolumnen fylls i med ett värde om `SalesAmount` inte har något värde.
- Kolumn 4 returnerar `CustomerRating`. Om inget värde hittas är standardvärdet `3`. Eftersom produkten `Laptop` t.ex. inte har någon kundbedömning returneras `3`. Måttet beräknar sedan genomsnittet för kolumnen.
- Kolumn 5 returnerar rabattvärdet. Om inget värde finns tillgängligt beräknas det som 5% av värdet på `SalesAmount`.

- Kolumn 6 returnerar värdet för `Profit`. Om inget värde finns tillgängligt returneras 10% av `SalesAmount`.
- Kolumn 7 returnerar `SalesAmount`. Om inget värde anges returneras det genomsnittliga `SalesAmount` för hela datauppsättningen. Till exempel har produkterna `Monitor` och `Mouse` inget värde för `SalesAmount`, så båda returnerar `750`, som är genomsnittet av alla `SalesAmount`-värden.
- Kolumn 8 returnerar värdet för `Discount`. Om inget värde hittas beräknar och returnerar systemet den genomsnittliga rabatten för hela datauppsättningen.

class

class-funktionen kopplar den första parametern till ett klassintervall. Resultatet är ett dualt värde med a<=x<b som textvärde, där a och b är de övre och nedre gränserna av bin, och det nedre gränsvärdet ett numeriskt värde.

Syntax:

```
class(expression, interval [ , label [ , offset ]])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
interval	Ett tal som anger bin-bredden.
label	Valfri sträng som kan ersätta 'x' i den resulterande texten.
offset	Ett tal som kan användas som förskjutning från standardstartpunkten i klassificeringen. Startpunkten är normalt 0.

Exempel: diagramuttryck

Exempel	Resultat
`class( var,10 )` med `var = 23`	Returnerar `20<=x<30`
`class( var,5,'value' )` med `var = 23`	Returnerar `20<= value <25`
`class( var,10,'x',5 )` med `var = 23`	Returnerar `15<=x<25`

Exempel – grunderna i class

Diagramuttryck

Översikt

En datatabell laddas som innehåller namn och åldrar. Du vill klassificera personerna efter åldersgrupper (i tioårsintervall) och räkna antalet personer i varje grupp.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Age`
  - `Name`

Laddningsskript

```
Example:
LOAD * INLINE [
Age, Name
25, John
42, Karen
53, Yoshi
43, Bob
27, Charles
];
```

Resultat

1. Ladda data och öppna ett ark. Skapa två nya tabeller.
2. I den första tabellen ska du göra följande:
   a. Lägg till de här dimensionerna:
      - `Age`
      - `Name`
   b. Skapa följande mått:
      - `=class(Age, 10, 'age')`, för att klassificera data i ålderskategorier.
3. I den andra tabellen ska du göra följande:
   a. Lägg till detta uttryck som en dimension
      - `=class(Age, 10, 'age')`
   b. Skapa följande mått:
      - `=Count(Age)`, för att räkna antalet personer i åldersgruppen.

Den första tabellen visar rådata för `Name` och `Age`, och vilken ålderskategori de tillhör.

Resultattabell 1

Name	Age	class(Age, 10, 'age')
Bob	43	40 <= age < 50
Charles	27	20 <= age < 30
John	25	20 <= age < 30
Karen	42	40 <= age < 50
Yoshi	53	50 <= age < 60

I den andra tabellen kan du se hur data nu klassificeras och sammanfattas. Resultatet av funktionen `class` skapade tre ålderskategorier i den första mätningen. Det andra måttet räknar antalet personer i varje ålderskategori.

Resultattabell 2

class(Age, 10, 'age')	Count(Age)
Totals	**5**
20 <= age < 30	2
40 <= age < 50	2
50 <= age < 60	1

Laddningsskript

Översikt

En datatabell laddas som innehåller namn och åldrar. Vi vill lägga till ett fält som klassificerar varje person utifrån en åldersgrupp med ett tioårsintervall. Den ursprungliga källtabellen ser ut enligt följande.

Ursprunglig källtabell

Name	Age
John	25
Karen	42
Yoshi	53

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En föregående LOAD-sats som använder funktionen **class** för att skapa ett nytt fält som heter `Agegroup`.

Laddningsskript

```
LOAD *,
class(Age, 10, 'age') As Agegroup;

LOAD * INLINE
[ Age, Name
25, John
42, Karen
53, Yoshi];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Name`
- `Age`
- `Agegroup`

Resultattabell

Name	Age	Agegroup
John	25	20 <= age < 30
Karen	42	40 <= age < 50
Yoshi	53	50 <= age < 60

Det nya fältet `Agegroup` klassificerar varje person efter ålderskategori.

Exempel – class-scenario

Diagramuttryck

Översikt

En datatabell laddas som innehåller kunddata. Du vill klassificera kunder efter antalet intjänade poäng för livstidsvärde och ange antalet kunder i varje kategori.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `CustomerID`
  - `CustomerName`
  - `LifetimeValue`

Laddningsskript

```
Example:
Load * Inline [
CustomerID, CustomerName, LifetimeValue
1, John Doe, 12500
2, Jane Smith, 36000
3, Sam Brown, 15000
4, Lisa White, 50000
5, Tom Harris, 22000
];
```

Resultat

1. Ladda data och öppna ett ark. Skapa två nya tabeller.
2. I den första tabellen ska du göra följande:
    a. Lägg till de här dimensionerna:
        - `CustomerID`
        - `CustomerName`
        - `LifetimeValue`
    b. Skapa följande beräknade dimension:
        - `=Class(LifetimeValue, 10000, 'Lifetime Value Group')`, för att klassificera data baserat på fältet `LifetimeValue`.
3. I den andra tabellen ska du göra följande:
    a. Lägg till detta uttryck som en dimension
        - `=Class(LifetimeValue, 10000, 'Lifetime Value Group')`
    b. Skapa följande mått:
        - `=Count(CustomerID)`, för att räkna antalet personer i åldersgruppen.

Den första tabellen visar de rådata som har matats in och resultatet av att använda klassfunktionen.

Resultattabell 1

CustomerID	CustomerName	LifetimeValue	Class(LifetimeValue, 10000, 'Lifetime Value Group')
1	John Doe	12500	10000 <= Lifetime Value Group < 20000
2	Jane Smith	36000	30000 <= Lifetime Value Group < 40000
3	Sam Brown	15000	10000 <= Lifetime Value Group < 20000
4	Lisa White	50000	50000 <= Lifetime Value Group < 60000
5	Tom Harris	22000	20000 <= Lifetime Value Group < 30000

Den andra tabellen visar hur dessa data nu klassificeras och sammanfattas. Uppgifterna kategoriseras i fyra grupper med hjälp av funktionen `class` och räknas sedan för att visa antalet kunder i varje grupp.

Resultattabell 2

Class(LifetimeValue, 10000, 'Lifetime Value Group')	Count(CustomerID)
Totals	**5**
10000 <= Lifetime Value Group < 20000	2
20000 <= Lifetime Value Group < 30000	1
30000 <= Lifetime Value Group < 40000	1
50000 <= Lifetime Value Group < 60000	1

coalesce

Coalesce-funktionen returnerar den första av parametrarna som har en giltig non-NULL-representation. Det finns inga begränsningar vad gäller antalet parametrar.

Syntax:

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
expr1	Det första uttrycket som ska kontrolleras om det har något giltigt icke-NULL-värde.
expr2	Det andra uttrycket som ska kontrolleras om det har något giltigt icke-NULL-värde.
expr3	Det tredje uttrycket som ska kontrolleras om det har något giltigt icke-NULL-värde.

Exempel: diagramuttryck

Exempel	Resultat
`Coalesce(ProductDescription, ProductName, ProductCode, 'no description available')`	Det här uttrycket väljer mellan tre olika produktbeskrivningsfält när vissa fält inte har värden för produkten. Genom att följa den ordning som anges i funktionen returneras det första fältet med ett icke-nullvärde. Om inget av fälten innehåller ett värde blir resultatet `no description available`.
`Coalesce(TextBetween(FileName, '"', '"'), FileName)`	Det här uttrycket tar bort eventuella citattecken runt fältet `FileName`. Om det `FileName` som används omges av citattecken tas de bort, och `FileName` returneras utan citattecknen. Om funktionen `TextBetween` inte hittar avgränsarna returnerar den null, vilket **Coalesce** avvisar, och returnerar `FileName` i råformat.

Exempel – grunderna i Coalesce

Diagramuttryck

Översikt

En datatabell laddas som innehåller produkt- och prisdata. Du vill se till att alla null-värden ersätts med ett alternativt värde.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ProductID`
  - `ProductName`
  - `Price`
- En `NullInterpret`-variabelsats för att säkerställa att eventuella saknade värden behandlas som NULL.

Laddningsskript

```
Set NullInterpret = '';
Example:
LOAD * INLINE [
ProductID, ProductName, Price
1, Laptop, 1000
2, , 800
3, Desktop, 1200
4, , 500
5, Tablet, 600
];
```

Resultat

1. Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:
   - `ProductID`
   - `ProductName`
2. Skapa följande beräknade dimension:
   - `=Coalesce(ProductName, 'Unknown Product')`
3. Skapa följande mått:
   - `=Sum(Price)` för att returnera en giltig talrepresentation.
4. Skapa en ny tabell och lägg till följande uttryck som en beräknad dimension:
   - `=Coalesce(ProductName, 'Unknown Product')`

5. Skapa följande mått:
    - `=Sum(Price)` för att returnera en giltig talrepresentation.

Resultaten i de följande tabellerna visar hur funktionen `Coalesce` kan ersätta nollvärdena i den ursprungliga datauppsättningen med det alternativa värdet `Unknown Product`.

Resultattabell 1

ProductID	ProductName	Coalesce(ProductName, 'Unknown Product')	Sum(Price)
Totals			**4100**
1	Laptop	Laptop	1000
2	-	Unknown Product	800
3	Desktop	Desktop	1200
4	-	Unknown Product	500
5	Tablet	Tablet	600

Resultattabell 2

Coalesce(ProductName, 'Unknown Product')	Sum(Price)
Totals	**5**
Desktop	600
Laptop	1000
Tablet	600
Unknown Product	1300

Exempel – Ersätt nollvärden med ett prioriterat alternativvärde

Diagramuttryck

Översikt

I detta exempel används funktionen `Coalesce` för att ersätta saknade uppgifter om plats med ett alternativt värde enligt den angivna prioritetslogiken.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `LocationID`
    - `Country`

- State
- City
- En NullInterpret-variabelsats för att säkerställa att eventuella saknade värden behandlas som NULL.

Laddningsskript

```
Set NullInterpret = '';
Example:
LOAD * INLINE [
LocationID, Country, State, City
L001, USA, California, Los Angeles
L002, USA, ,
L003, Canada, Ontario,
L004, , New South Wales, Sydney
L005, UK, , London
L006, , , London
];
```

Resultat

1. Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:
   - LocationID
   - Country
   - State
   - City
2. Skapa följande beräknade dimension:
   - =Coalesce(Country, State, City), för att fylla i eventuella nollvärden med alternativ.

Resultattabell

LocationID	Country	State	City	Coalesce(Country, State, City)
L001	USA	California	Los Angeles	USA
L002	USA	-	-	USA
L003	Canada	Ontario	-	Canada
L004	-	New South Wales	Sydney	New South Wales
L005	UK	-	London	UK
L006	-	-	London	London

Resultaten visar att även om det finns saknade värden kan du använda funktionen Coalesce för att tillhandahålla logik om hur du fyller i dessa instanser med ett alternativt värde för att komplettera datauppsättningen. Till exempel saknar L004 ett värde för Country. Funktionen Coalesce returnerar New South Wales eftersom State är det andra fältet i prioritetslogiken som ska användas när värdet för Country saknas.

if

if-funktionen returnerar ett värde beroende på om villkoret som hör samman med funktionen utvärderas som True eller False.

Syntax:

```
if(condition , then [, else])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
condition	Uttryck som tolkas logiskt.
then	Uttrycket kan vara av vilken typ som helst. Om *condition* är True, returnerar funktionen if värdet för uttrycket *then*.
else	Uttrycket kan vara av vilken typ som helst. Om *condition* är False, returnerar funktionen if värdet för uttrycket *else*. Den här parametern är valfri. Om *condition* är False returneras NULL om du inte har specificerat else.

Exempel: diagramuttryck

Exempel	Resultat
`if( Amount>= 0, 'OK', 'Alarm' )`	Uttrycket testar om beloppet är ett positivt tal (0 eller större) och returnerar `OK` om det är det. Om beloppet är mindre än 0 returneras `Alarm`.

Exempel – Laddningsskript som använder if

Laddningsskript

Översikt

If kan användas i laddningsskript med andra metoder och objekt, inklusive variabler. I det här exemplet används variabeln `threshold` och inkluderar ett nytt fält i datamodellen som baseras på detta threshold.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i två datatabeller som heter:
  - `Transactions`: Innehåller transaktionsfält
  - `Transaction_Buckets`: Inkluderar ett `transaction_id`-fält och skapar ett nytt fält, `[Compared to Threshold]`, med hjälp av `if`-funktionen för att jämföra fältet `transaction_amount` med

variabeln `threshold`.

- En `threshold`-variabelsats som ställs in på `100`.

Laddningsskript

```
Transactions:
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, m, orange
3752, 20180916, 5.75, 1, 5646471, S, blue
3753, 20180922, 125.00, 7, 3036491, l, Black
3754, 20180922, 484.21, 13, 049681, xs, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
];

set threshold = 100;

/* Create new table called Transaction_Buckets
Compare transaction_amount field from Transaction table to threshold of 100.
Output results into a new field called Compared to Threshold
 */

Transaction_Buckets:
Load
   transaction_id,
   If(transaction_amount > $(threshold),'Greater than $(threshold)','Less than $(threshold)')
as [Compared to Threshold]
Resident Transactions;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `transaction_id`
- `[Compared to Threshold]`

Resultattabellen visar utdata vid användning av *if*-funktionen i laddningsskriptet.

transaktions-ID	Jämfört med tröskel
3750	Mindre än 100
3751	Större än 100
3752	Mindre än 100
3753	Större än 100

transaktions-ID	Jämfört med tröskel
3754	Större än 100
3756	Mindre än 100
3757	Större än 100

Exempel – Utvärdera produktdata med hjälp av if

Diagramuttryck

Översikt

En datauppsättning innehåller produktdata. Du vill identifiera och kategorisera produkter beroende på om de är förbrukningsbara eller inte.

Laddningsskript

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ProductID`
  - `ProductName`
  - `ProductCategory`

```
Example:
LOAD * INLINE [
ProductID, ProductName, ProductCategory
1, "Laptop", "Electronics"
2, "Banana", "Fruits"
3, "Shampoo", "Toiletries"
4, "TV", "Electronics"
5, "Apple", "Fruits"
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ProductID`
- `ProductName`
- `ProductCategory`

Skapa följande mått:

- `=If(ProductCategory = 'Toiletries' or ProductCategory = 'Fruits', 'Consumable', 'Other')`, för att returnera de värden som krävs.

Resultattabell

ProductID	ProductName	ProductCategory	If(ProductCategory = 'Toiletries' or ProductCategory = 'Fruits', 'Consumable', 'Other')
1	Laptop	Electronics	Other
2	Banana	Fruits	Consumable
3	Shampoo	Toiletries	Consumable
4	TV	Electronics	Other
5	Apple	Fruits	Consumable

Du kan se i resultatet att `if`-funktionen utvärderar uttrycket och returnerar de värden som krävs. Till exempel klassificeras produkter i kategorin `Fruits` eller `Toiletries` som `Consumable`.

Exempel – Utvärdera och klassificera kunddata med hjälp av if

Diagramuttryck

Översikt

En datauppsättning innehåller kunddata. Du vill identifiera och kategorisera de kunder vars region och branschkategori uppfyller dina klassificeringskrav.

Laddningsskript

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ClientID`
  - `ClientName`
  - `Region`
  - `Industry`

```
Example:
LOAD * INLINE [
ClientID, ClientName, Region, Industry
1, "Acme Corp", "North", "Manufacturing"
2, "Beta Ltd", "East", "Technology"
```

```
3, "Gamma Inc", "West", "Retail"
4, "Delta LLC", "North", "Technology"
5, "Epsilon GmbH", "South", "Healthcare"
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ClientID`
- `ClientName`
- `Region`
- `Industry`

Skapa följande mått:

- `=If((Region = 'North' or Region = 'East') AND (Industry= 'Technology' or Industry = 'Manufacturing'), 'Target Segment', 'Secondary Segment')`, för att returnera det värde som krävs.

Resultattabell

ClientID	ClientName	Region	Industry	If((Region = 'North' or Region = 'East') AND (Industry= 'Technology' or Industry = 'Manufacturing'), 'Target Segment', 'Secondary Segment')
1	Acme Corp	North	Manufacturing	Target Segment
2	Beta Ltd	East	Technology	Target Segment
3	Gamma Inc	West	Retail	Secondary Segment
4	Delta LLC	North	Technology	Target Segment
5	Epsilon GmbH	Sound	Healthcare	Secondary Segment

Du kan se i resultatet att `if`-funktionen utvärderar uttrycket och returnerar de värden som krävs. Exempelvis klassificeras `Acme Corp` som `Target Segment` eftersom det uppfyller kraven för att ingå i region `North` eller `East` och branschen `Manufacturing` eller `Technology`.

Exempel – Utvärdera incidentdata med hjälp av if

Diagramuttryck

Översikt

Använd `if`-funktionen för att kategorisera en datauppsättning med incidentrapporter.

Laddningsskript

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `Date`
    - `Location`
    - `Incidents`

```
Example:
LOAD * inline [
Date, Location, Incidents
1/3/2016, Beijing, 0
1/3/2016, Boston, 12
1/3/2016, Stockholm, 3
1/3/2016, Toronto, 0
1/4/2016, Beijing, 0
1/4/2016, Boston, 8];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- Datum
- Plats
- Incidenter

Skapa följande beräknade dimensioner:

- `=if(Incidents>=10, 'Critical', 'Ok' )`
- `=if(Incidents>=10, 'Critical', If( Incidents>=1 and Incidents<10, 'Warning', 'Ok'))`

Resultattabell

Date	Location	Incidents	if(Incidents>=10, 'Critical', 'Ok')	if(Incidents>=10, 'Critical', If(Incidents>=1 and Incidents<10, 'Warning', 'Ok'))
1/3/2016	Beijing	0	Ok	Ok
1/3/2016	Boston	12	Critical	Critical
1/3/2016	Stockholm	3	Ok	Warning
1/3/2016	Toronto	0	Ok	Ok
1/4/2016	Beijing	0	Ok	Ok
1/4/2016	Boston	8	Ok	Warning

Exempel – Kategorisera datum med hjälp av if

Diagramuttryck

Översikt

Använd `if`-funktionen för att kategorisera datum.

Laddningsskript

Öppna Skriptredigeraren och lägg till laddningsskriptet under ett nytt delavsnitt.

```
SET FirstWeekDay=0;
Load
Date(MakeDate(2022)+RecNo()-1) as Date
Autogenerate 14;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Date`

Skapa följande beräknade dimensioner:

- `WeekDay(Date)`
- `=If(WeekDay(Date)>=5,'WeekEnd','Normal Day')`

Resultattabell

Date	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/1/2022	Sat	WeekEnd

Date	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/2/2022	Sun	WeekEnd
1/3/2022	Mon	Normal Day
1/4/2022	Tue	Normal Day
1/5/2022	Wed	Normal Day
1/6/2022	Thu	Normal Day
1/7/2022	Fri	Normal Day
1/8/2022	Sat	WeekEnd
1/9/2022	Sun	WeekEnd
1/10/2022	Mon	Normal Day
1/11/2022	Tue	Normal Day
1/12/2022	Wed	Normal Day
1/13/2022	Thu	Normal Day
1/14/2022	Fri	Normal Day

match

Funktionen **match** jämför den första parametern med alla de följande och returnerar den numeriska platsen för uttryck som matchar. Jämförelsen är skiftlägeskänslig.

Syntax:

```
match( str, expr1 [ , expr2,...exprN ])
```

Returnerad datatyp: heltal

*Om du vill använda ej skiftlägeskänslig jämförelse använder du funktionen **mixmatch**. Om du vill använda ej skiftlägeskänslig jämförelse och jokertecken använder du funktionen **wildmatch**.*

Argument

Argument	Beskrivning
str	Strängen som ska utvärderas.
expr1	Det första uttrycket som ska kontrolleras om det har ett giltigt numeriskt värde.
expr2	Det andra uttrycket som ska kontrolleras om det har ett giltigt numeriskt värde.
expr3	Det tredje uttrycket som ska kontrolleras om det har ett giltigt numeriskt värde.

Exempel: diagramuttryck

Exempel	Resultat
`match( ColorCode,'Blue','Black' )`	Detta uttryck returnerar indexet för matchningen i fältet `ColorCode`. Om värdet för `ColorCode` är `Blue` returnerar det `1`. Om värdet för `ColorCode` är `Black` returnerar det `2`. Om ingen motsvarighet påträffas, returneras `0`.
`match( Region, 'North', 'East' )`	Detta uttryck returnerar indexet för matchningen i fältet `Region`. Om värdet för `Region` är `North` returnerar det `1`. Om värdet för `Region` är `East` returnerar det `2`. Om ingen motsvarighet påträffas, returneras `0`.

Exempel – Kategorisera produkter med hjälp av match

Diagramuttryck

Översikt

En datauppsättning innehåller produktdata. Du vill identifiera och kategorisera förbrukningsbara produkter.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ProductID`
  - `ProductName`
  - `ProductCategory`

Laddningsskript

```
Example:
LOAD * INLINE [
ProductID, ProductName, ProductCategory
1, "Laptop", "Electronics"
2, "Banana", "Fruits"
3, "Shampoo", "Toiletries"
4, "TV", "Electronics"
5, "Apple", "Fruits"
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ProductID`
- `ProductName`
- `ProductCategory`

Skapa följande mått:

- `=match(ProductCategory,'Toiletries', 'Fruits')`, för att returnera det index där `ProductCategory` har värdet `Toiletries` eller `Fruits`.
- `=If(match(ProductCategory, 'Toiletries', 'Fruits'), 'Consumable', 'Other')`, för att returnera det index där `ProductCategory` har värdet `Toiletries` eller `Fruits`. Om någon av dem hittas returneras `Consumable`. I annat fall returneras `Other`.

Resultattabell

ProductID	ProductName	ProductCategory	match (ProductCategory,'Toiletries', 'Fruits')	If(match (ProductCategory, 'Toiletries', 'Fruits'), 'Consumable', 'Other')
1	Laptop	Electronics	0	Other
2	Banana	Fruits	2	Consumable
3	Shampoo	Toiletries	1	Consumable
4	TV	Electronics	0	Other
5	Apple	Fruits	2	Consumable

`match`-funktionen returnerar indexet för söksträngen i det första måttet. I den andra åtgärden används detta resultat sedan i en `If`-funktion för att kategorisera produkter som `Consumable` eller `Other`.

Exempel – Utvärdera och klassificera kunddata med hjälp av match

Diagramuttryck

Översikt

En datauppsättning innehåller kunddata. Du vill identifiera och kategorisera de kunder vars region och branschkategori uppfyller dina klassificeringskrav.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ClientID`
  - `ClientName`
  - `Region`
  - `Industry`

Laddningsskript

```
Example:
LOAD * INLINE [
ClientID, ClientName, Region, Industry
1, "Acme Corp", "North", "Manufacturing"
2, "Beta Ltd", "East", "Technology"
3, "Gamma Inc", "West", "Retail"
4, "Delta LLC", "North", "Technology"
5, "Epsilon GmbH", "South", "Healthcare"
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ClientID`
- `ClientName`
- `Region`
- `Industry`

Skapa följande mått:

`=IF(match(Region, 'North', 'East') AND match(Industry, 'Technology', 'Manufacturing'), 'Target Segment', 'Secondary Segment')`, för att returnera indexet för söksträngen. Om det hittas returneras `Target Segment`. Om det inte hittas returneras `Secondary Segment`.

Resultattabell

ClientID	ClientName	Region	Industry	IF(match(Region, 'North', 'East') AND match(Industry, 'Technology', 'Manufacturing'), 'Target Segment', 'Secondary Segment')
1	Acme Corp	North	Manufacturing	Target Segment
2	Beta Ltd	East	Technology	Target Segment
3	Gamma Inc	West	Retail	Secondary Segment
4	Delta LLC	North	Technology	Target Segment
5	Epsilon GmbH	South	Healthcare	Secondary Segment

`match`-funktionen söker efter text och returnerar ett index som sedan kan användas för att utföra ytterligare analyser och dataklassificering. Exempelvis klassificeras `Acme Corp` som ett `Target Segment` eftersom det uppfyller `match`-funktionens sökkrav för fälten `Region` och `Industry`.

Exempel: laddningsskript som använder match

Laddningsskript

Översikt

Du kan använda match för att ladda en delmängd av dina data. Du kan till exempel returnera ett numeriskt värde för ett uttryck i funktionen. Sedan kan du begränsa vilka data som laddas utifrån det numeriska värdet. Match-funktionen returnerar 0 om det inte finns någon match. Alla uttryck som inte har någon matchning i det här exemplet returnerar därför 0 och `WHERE`-satsen utesluter dem från dataladdningen.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i två datatabeller som heter:
  - `Transactions`: Innehåller transaktionsfält
  - `Transaction_Buckets`: Skapar två nya fält `Customer` och `Color Code Blue and Black`. `where`-satsen läser in resultaten från `match`-funktionen.

Laddningsskript

```
Transactions:
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, m, orange
3752, 20180916, 5.75, 1, 5646471, S, blue
3753, 20180922, 125.00, 7, 3036491, l, Black
3754, 20180922, 484.21, 13, 049681, xs, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
];

/*
Create new table called Transaction_Buckets
Create new fields called Customer, and Color code Blue and Black
Load Transactions table.
Match returns 1 for 'Blue', 2 for 'Black'.
Does not return a value for 'blue' because match is case sensitive.
Only values that returned numeric value greater than 0
are loaded by WHERE statement into Transactions_Buckets table.
*/

Transaction_Buckets:
Load
 customer_id,
```

```
 customer_id as [Customer],
 color_code as [Color Code Blue and Black]
Resident Transactions
Where match(color_code,'Blue','Black') > 0;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `[Color Code Blue and Black]`
- `Customer`

Resultattabellen visar utdata vid användning av match-funktionen i laddningsskriptet

Color Code Blue and Black	Customer
Black	203521
Black	3036491
Blue	2038593

Exempel – diagramuttryck som använder match

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Cities`
  - `Count`

Laddningsskript

```
Example:
Load * inline [
Cities, Count
Toronto, 123
Toronto, 234
Toronto, 231
Boston, 32
Boston, 23
Boston, 1341
Beijing, 234
Beijing, 45
Beijing, 235
```

```
Stockholm, 938
Stockholm, 39
Stockholm, 189
zurich, 2342
zurich, 9033
zurich, 0039];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Cities`

Skapa följande mått:

- `=match( Cities,'Toronto','Boston','Beijing','Zurich')`, för att returnera indexet för söksträngen.
- `=match( Cities,'Toronto','Boston','Beijing','Stockholm','zurich')`, för att returnera indexet för söksträngen.
- `=Cities & ' - ' & match ( Cities, 'Toronto','Boston', 'Beijing','Stockholm','zurich')`, för att returnera `Cities`-fältet och indexet för söksträngen.

Det första måttet i tabellen nedan returnerar `0` för `Stockholm`, eftersom `Stockholm`inte finns med i listan med uttryck i `match`-funktionen. Den returnerar också `0` för `Zurich` eftersom `match`-jämförelsen är skiftlägeskänslig.

Resultattabell

Cities	match(Cities,'Toronto','Boston','Beijing','Zurich')	match(Cities,'Toronto','Boston','Beijing','Stockholm','zurich')	Cities & ' - ' & match (Cities, 'Toronto','Boston', 'Beijing','Stockholm','zurich')
Beijing	3	3	Beijing - 3
Boston	2	2	Boston - 2
Stockholm	0	4	Stockholm - 4
Toronto	1	1	Toronto - 1
zurich	0	5	zurich - 5

Exempel – anpassat diagramuttryck som använder match

Diagramuttryck

Översikt

Du kan använda `match` för att utföra en anpassad sortering för ett uttryck. I det här exemplet används samma datauppsättning som i föregående exempel.

Som standard sorteras kolumner numeriskt eller alfabetiskt, beroende på datatyp.

Resultattabell som visar standardsorteringsordningen

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Gör följande för att ändra sorteringsordningen:

1. Öppna delavsnittet **Sortering** för ditt diagram i **egenskapspanelen**.
2. Inaktivera automatisk sortering för den kolumn du vill använda anpassad sortering för.
3. Radera kryssrutorna **Sortera numeriskt** och **Sortera alfabetiskt**.
4. Välj **Sortera efter uttryck** och ange sedan ett uttryck som liknar följande:
   `=match( Cities, 'Toronto','Boston','Beijing','Stockholm','zurich')`
   Sorteringsordningen för kolumnen Cities ändras.

Resultattabell som visar sorteringsordningen med hjälp av *match*-funktionen

Cities
Toronto
Boston
Beijing
Stockholm
zurich

mixmatch

Funktionen **mixmatch** jämför den första parametern med alla de följande och returnerar den numeriska platsen för uttryck som matchar. Jämförelsen är inte skiftlägeskänslig och är okänslig för de japanska teckensystemen hiragana och katakana.

Syntax:

```
mixmatch( str, expr1 [ , expr2,...exprN ])
```

Returnerad datatyp: heltal

Om du vill använda skiftlägeskänslig jämförelse använder du funktionen **match**. Om du vill använda ej skiftlägeskänslig jämförelse och jokertecken använder du funktionen **wildmatch**.

Argument

Argument	Beskrivning
str	Strängen som ska utvärderas.
expr1	Det första uttrycket som ska kontrolleras om det har ett giltigt numeriskt värde.
expr2	Det andra uttrycket som ska kontrolleras om det har ett giltigt numeriskt värde.
expr3	Det tredje uttrycket som ska kontrolleras om det har ett giltigt numeriskt värde.

Exempel: diagramuttryck

Exempel	Resultat
`mixmatch( ColorCode,'black','Blue' )`	Detta uttryck returnerar `1` om värdet av `ColorCode` är `Black` eller `2` om värdet av `ColorCode` är `Blue`.
`mixmatch( Cities,'toronto','Boston','Beijing','Zurich' )`	Detta uttryck returnerar indexet för det objekt som matchar värdet i `Cities`-fältet.

Exempel – Kategorisera produkter med hjälp av mixmatch

Diagramuttryck

Översikt

En datauppsättning innehåller produktdata. Du vill identifiera och kategorisera förbrukningsbara produkter.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ProductID`
  - `ProductName`
  - `ProductCategory`

Laddningsskript

```
Example:
LOAD * INLINE [
ProductID, ProductName, ProductCategory
1, "Laptop", "Electronics"
2, "Banana", "Fruits"
3, "Shampoo", "toiletries"
4, "TV", "Electronics"
5, "Apple", "fruits"
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ProductID`
- `ProductName`
- `ProductCategory`

Skapa följande mått:

- `=mixmatch(ProductCategory,'Toiletries', 'Fruits')`, för att returnera det index där `ProductCategory` har värdet `Toiletries` eller `Fruits`.
- `=If(mixmatch(ProductCategory, 'Toiletries', 'Fruits'), 'Consumable', 'Other')`, för att översätta en `ProductCategory`-matchning mot antingen `Toiletries` eller `Fruits` som `Consumable`. Om ingen motsvarighet påträffas, returneras `Other`.

Funktionen `mixmatch` returnerar ett positivt värde som inte är noll om en matchning hittas, annars noll om ingen matchning hittas. `if`-satsen tolkar i sin tur resultat som inte är noll som `true` och resultat som är noll som `false`. Genom att kombinera en `if`-sats med funktionerna `mixmatch`, `wildmatch` eller `match`, är det möjligt att enkelt avkoda matchningar och felmatchningar.

Resultattabell

ProductID	ProductName	ProductCategory	mixmatch (ProductCategory,'Toiletries', 'Fruits')	If(mixmatch (ProductCategory, 'Toiletries', 'Fruits'), 'Consumable', 'Other')
1	Laptop	Electronics	0	Other
2	Banana	Fruits	2	Consumable
3	Shampoo	toiletries	1	Consumable

ProductID	ProductName	ProductCategory	mixmatch (ProductCategory,'Toiletries', 'Fruits')	If(mixmatch (ProductCategory, 'Toiletries', 'Fruits'), 'Consumable', 'Other')
4	TV	Electronics	0	Other
5	Apple	fruits	2	Consumable

`mixmatch`-funktionen returnerar indexet för söksträngen i det första måttet. I den andra åtgärden används detta resultat sedan i en `If`-funktion för att ge anpassade resultat. Observera att funktionen `mixmatch` inte är skiftlägeskänslig.

Exempel – Utvärdera och klassificera kunddata med hjälp av mixmatch

Diagramuttryck

Översikt

En datauppsättning innehåller kunddata. Du vill identifiera och kategorisera de kunder vars region och branschkategori uppfyller dina klassificeringskrav.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ClientID`
  - `ClientName`
  - `Region`
  - `Industry`

Laddningsskript

```
Example:
LOAD * INLINE [
ClientID, ClientName, Region, Industry
1, "Acme Corp", "north", "manufacturing"
2, "Beta Ltd", "East", "Technology"
3, "Gamma Inc", "west", "Retail"
4, "Delta LLC", "North", "Technology"
5, "Epsilon GmbH", "South", "healthcare"
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ClientID`
- `ClientName`
- `Region`
- `Industry`

Skapa följande mått:

`=If(mixmatch(Region, 'North', 'East') AND mixmatch(Industry, 'Technology', 'Manufacturing'), 'Target Segment', 'Secondary Segment')`, för att returnera indexet för söksträngen. Om det hittas returneras `Target Segment`. Om det inte hittas returneras `Secondary Segment`.

Resultattabell

ClientID	ClientName	Region	Industry	If(mixmatch (Region, 'North', 'East') AND mixmatch (Industry, 'Technology', 'Manufacturing'), 'Target Segment', 'Secondary Segment')
1	Acme Corp	north	manufacturing	Target Segment
2	Beta Ltd	East	Technology	Target Segment
3	Gamma Inc	west	Retail	Secondary Segment
4	Delta LLC	North	Technology	Target Segment
5	Epsilon GmbH	South	healthcare	Secondary Segment

`mixmatch`-funktionen söker efter text och returnerar ett index som sedan kan användas för att utföra ytterligare analyser och dataklassificering. Exempelvis klassificeras `Acme Corp` som ett `Target Segment` eftersom det uppfyller `mixmatch`-funktionens sökkrav för fälten `Region` och `Industry`. Observera att funktionen `mixmatch` inte är skiftlägeskänslig.

Exempel – Laddningsskript som använder mixmatch

Laddningsskript

Översikt

Du kan använda mixmatch för att ladda en delmängd av dina data. Du kan till exempel returnera ett numeriskt värde för ett uttryck i funktionen. Sedan kan du begränsa vilka data som laddas utifrån det numeriska värdet. Mixmatch returnerar 0 om det inte finns någon matchning. Alla uttryck som inte har någon matchning i det här exemplet returnerar därför 0 och `WHERE`-satsen utesluter dem från dataladdningen.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i två datatabeller som heter:
  - `Transactions`: Innehåller transaktionsfält
  - `Transaction_Buckets`: Skapar två nya fält `Customer` och `Color Code - Black, Blue, blue`. `where`-satsen läser in resultaten från `mixmatch`-funktionen.

Laddningsskript

```
Transactions:
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, m, orange
3752, 20180916, 5.75, 1, 5646471, S, blue
3753, 20180922, 125.00, 7, 3036491, l, Black
3754, 20180922, 484.21, 13, 049681, xs, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
];

/*
Create new table called Transaction_Buckets
Create new fields called Customer, and Color code - Black, Blue, blue
Load Transactions table.
Mixmatch returns 1 for 'Black', 2 for 'Blue'.
Also returns 3 for 'blue' because mixmatch is not case sensitive.
Only values that returned numeric value greater than 0
are loaded by WHERE statement into Transactions_Buckets table.
*/

Transaction_Buckets:
Load
   customer_id,
   customer_id as [Customer],
   color_code as [Color Code - Black, Blue, blue]
```

```
Resident Transactions
Where mixmatch(color_code,'Black','Blue') > 0;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `[Color Code - Black, Blue, blue]`
- `Customer`

Resultattabellen visar utdata vid användning av mixmatch-funktionen i laddningsskriptet.

Color Code - Black, Blue, blue	Customer
Black	203521
Black	3036491
Blue	2038593
blue	5646471

Exempel – diagramuttryck som använder mixmatch

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Cities`
  - `Count`

Laddningsskript

```
Example:
Load * inline [Cities, Count
Toronto, 123
Toronto, 234
Toronto, 231
Boston, 32
Boston, 23
Boston, 1341
Beijing, 234
Beijing, 45
Beijing, 235
Stockholm, 938
Stockholm, 39
```

```
Stockholm, 189
zurich, 2342
zurich, 9033
zurich, 0039];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Cities`

Skapa följande mått:

- `=mixmatch( Cities,'Toronto','Boston','Beijing','Zurich')`, för att returnera indexet för söksträngen.
- `=mixmatch( Cities,'Toronto','Boston','Beijing','Stockholm','Zurich')`, för att returnera indexet för söksträngen.
- `=Cities & ' - ' & mixmatch ( Cities, 'Toronto','Boston', 'Beijing','Stockholm','Zurich')`, för att returnera `Cities`-fältet och indexet för söksträngen.

Det första måttet i tabellen nedan returnerar `0` för `Stockholm`, eftersom `Stockholm`inte finns med i listan med uttryck i `mixmatch`-funktionen. Den returnerar också `4` för `Zurich` eftersom `mixmatch`-jämförelsen inte är skiftlägeskänslig.

Resultattabell

Cities	mixmatch(Cities,'Toronto','Boston','Beijing','Zurich')	mixmatch(Cities,'Toronto','Boston','Beijing','Stockholm','Zurich')	Cities & ' - ' & mixmatch (Cities, 'Toronto','Boston', 'Beijing','Stockholm','Zurich')
Beijing	3	3	Beijing - 3
Boston	2	2	Boston - 2
Stockholm	0	4	Stockholm - 4
Toronto	1	1	Toronto - 1
zurich	4	5	zurich - 5

Exempel – anpassat diagramuttryck som använder mixmatch

Diagramuttryck

Översikt

Du kan använda `mixmatch` för att utföra en anpassad sortering för ett uttryck. I det här exemplet används samma datauppsättning som i föregående exempel.

Som standard sorteras kolumner numeriskt eller alfabetiskt, beroende på datatyp.

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Cities`

Resultattabell som visar standardsorteringsordningen

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Gör följande för att ändra sorteringsordningen:

1. Öppna delavsnittet **Sortering** för ditt diagram i **egenskapspanelen**.
2. Inaktivera automatisk sortering för den kolumn du vill använda anpassad sortering för.
3. Radera kryssrutorna **Sortera numeriskt** och **Sortera alfabetiskt**.
4. Välj **Sortera efter uttryck** och ange sedan ett uttryck som liknar följande:
   `=mixmatch( Cities, 'Toronto','Boston','Beijing','Stockholm','Zurich')`
   Sorteringsordningen för kolumnen Cities ändras.

Resultattabell som visar sorteringsordningen med hjälp av *mixmatch*-funktionen

Cities
Toronto
Boston
Beijing
Stockholm
zurich

pick

Funktionen pick returnerar det *n*:te uttrycket i listan.

Syntax:

```
pick(n, expr1[ , expr2,...exprN])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
n	*n* är ett heltal mellan 1 och N.

Exempel: diagramuttryck

Exempel	Resultat
`pick( N, 'A','B',4, 6 )`	returnerar `B` om `N = 2` returnerar `4` om `N = 3`

Exempel – Tilldela etiketter till produkter med hjälp av pick

Diagramuttryck

Översikt

En datauppsättning innehåller produktdata. Du vill tilldela kategorietiketter till produkter baserat på kategorikoden.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ProductID`
  - `ProductName`
  - `CategoryCode`

Laddningsskript

```
Example:
LOAD * INLINE [
ProductID, ProductName, CategoryCode
1, "Laptop", 1
2, "Headphones", 2
3, "Printer", 3
4, "Smartphone", 1
5, "Tablet", 1
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ProductID`
- `ProductName`
- `CategoryCode`

Skapa följande mått:

- `=Pick(CategoryCode, 'Electronics', 'Accessories', 'Office Equipment')`, för att returnera ett giltigt uttryck från listan.

Resultattabell

ProductID	ProductName	CategoryCode	Pick (CategoryCode, 'Electronics', 'Accessories', 'Office Equipment')
1	Laptop	1	Electronics
2	Headphone	2	Accessories
3	Printer	3	Office Equipment
4	Smartphone	1	Electronics
5	Tablet	1	Electronics

Funktionen `pick` returnerar en lista med värden som är associerade med motsvarande index från det numeriska fältet `CategoryCode`. Till exempel är etiketten `Electronics` den första parametern i funktionen och returneras för alla produkter med `CategoryCode 1`.

Exempel – Tilldela etiketter till regioner med hjälp av pick

Diagramuttryck

Översikt

En datauppsättning innehåller lands- och regionkoder. Du vill tilldela regionetiketter till länder baserat på regionkoden.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Country`
  - `RegionCode`

Laddningsskript

```
Example:
LOAD * INLINE [
Country, RegionCode
"USA", 1
"Canada", 1
"Germany", 2
"Japan", 3
"Australia", 4
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Country`
- `RegionCode`

Skapa följande mått:

- `=Pick(RegionCode, 'North America', 'Europe', 'Asia', 'Australia')`, för att returnera ett giltigt objekt från listan.

Resultattabell

Country	RegionCode	Pick (RegionCode, 'North America', 'Europe', 'Asia', 'Australia')
Australia	4	Australia
Canada	1	North America
Germany	2	Europe
Japan	3	Asia
USA	1	North America

Funktionen `pick` returnerar en lista med värden som är associerade med motsvarande index från det numeriska fältet `RegionCode`. Till exempel är etiketten `North America` den första parametern i funktionen och returneras för alla länder med `RegionCode 1`.

Exempel – Kombination av pick och match för att översätta listor med alternativa textmatchningar

Diagramuttryck

Översikt

Detta avancerade scenario beskriver hur man effektivt använder en kombination av funktionerna `pick` och `match` för att mappa textmatchningar in-line. I programmeringstermer har detta ett liknande syfte som en `Case`-sats. Scenariot utforskar denna funktionalitet genom att mappa landsnamn till regionekvivalenta namn och samtidigt tillhandahålla ett standardvärde för `Region` för alla tidigare ej mappade `Country`-namn.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Countries`.
    - Datatabellen har ett fält som heter `Country`.

Laddningsskript

```
Countries:
LOAD * INLINE [
Country
"USA"
"Canada"
"Germany"
"Japan"
"Australia"
"South Africa"
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Country`
- `RegionCode`

Skapa följande mått:

- `=Pick(1+match(Country,'USA','Canada','Germany','Japan','Australia'), 'Unmapped', 'North America', 'North America', 'Europe', 'Asia', 'Australia')`

`match`-funktionen returnerar indexmatchningen av de kommaseparerade alternativen som anges, med början från `1 = 'USA'`, `2 = 'Canada'` och så vidare. Om ingen motsvarighet påträffas, returneras noll. Detta resultat ökas sedan med `1` och matas in i funktionen `pick`, så att `1 = default value` om ingen matchning hittas, `2 = 'USA'`, `3 = 'Canada'` och så vidare.

Denna kombination av funktionsanropen `Match` och `Pick` kommer sedan att avkoda de matchande `Country`-namnen till ett motsvarande `Region`-namn. Alla omappade `Country`-värden kommer att returnera standardvärdet för `Region`, `Unmapped`.

Resultattabell

Country	Pick(1+match(Country,'USA','Canada','Germany','Japan','Australia'), 'Unmapped', 'North America', 'North America', 'Europe', 'Asia', 'Australia')
Australia	Australia
Canada	North America
Germany	Europe
Japan	Asia
South Africa	Unmapped
USA	North America

wildmatch

Funktionen **wildmatch** jämför den första parametern med alla de följande och returnerar antalet uttryck som matchar. Den gör det möjligt att använda jokertecken (***** och **?**) i jämförelsesträngarna. ***** matchar alla följder av tecken. **?** matchar alla enstaka tecken. Jämförelsen är inte skiftlägeskänslig och är okänslig för de japanska teckensystemen hiragana och katakana.

Syntax:

```
wildmatch( str, expr1 [ , expr2,...exprN ])
```

Returnerad datatyp: heltal

Om du vill använda en jämförelse utan jokertecken, ska du använda funktionerna **match** eller **mixmatch**.

Argument

Argument	Beskrivning
str	Strängen som ska utvärderas.
expr1	Det första uttrycket som ska kontrolleras om det har ett giltigt numeriskt värde.
expr2	Det andra uttrycket som ska kontrolleras om det har ett giltigt numeriskt värde.
expr3	Det tredje uttrycket som ska kontrolleras om det har ett giltigt numeriskt värde.

Exempel: diagramuttryck

Exempel	Resultat
`wildmatch( ColorCode,'black','*Blue*' )`	Detta uttryck returnerar `1` om värdet för `ColorCode` är `Black` eller `2` om värdet för `ColorCode` är `Blue` eller `Light Blue` eller `blue`.
`wildmatch( Cities,'toronto','Boston','Beijing','Zurich')`	Detta uttryck returnerar indexet för det objekt som matchar värdet i `Cities`-fältet.

Exempel – Kategorisera produkter med hjälp av wildmatch

Diagramuttryck

Översikt

En datauppsättning innehåller produktdata. Du vill identifiera och kategorisera förbrukningsbara produkter.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ProductID`
  - `ProductName`
  - `ProductCategory`

Laddningsskript

```
Example:
LOAD * INLINE [
ProductID, ProductName, ProductCategory
1, "Laptop", "Electronics"
2, "Banana", "fruit"
3, "Shampoo", "toiletries"
4, "TV", "Electronics"
5, "Apple", "Fruits and vegetables"
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ProductID`
- `ProductName`
- `ProductCategory`

Skapa följande mått:

- `=wildmatch(ProductCategory, 'Toiletries*', 'Fruit*')`, för att returnera det index där `ProductCategory` har värdet som börjar med `Toiletries` eller börjar med `Fruit`.
- `=If(wildmatch(ProductCategory, 'Toiletries*', 'Fruit*'), 'Consumable', 'Other')`, för att returnera det index där `ProductCategory` har värdet som börjar med `Toiletries` eller börjar med `Fruits`. Om någon av dem hittas returneras `Consumable`. I annat fall returneras `Other`.

Funktionen `wildmatch` returnerar ett positivt värde som inte är noll om en matchning hittas, annars noll om ingen matchning hittas. `if`-satsen tolkar i sin tur resultat som inte är noll som `true` och resultat som är noll som `false`. Genom att kombinera en `if`-sats med funktionerna `mixmatch`, `wildmatch` eller `match`, är det möjligt att enkelt avkoda matchningar och felmatchningar.

Resultattabell

ProductID	ProductName	ProductCategory	wildmatch (ProductCategory, 'Toiletries*', 'Fruit*')	If(wildmatch (ProductCategory, 'Toiletries*', 'Fruit*')
1	Laptop	Electronics	0	Other
2	Banana	fruit	2	Consumable
3	Shampoo	toiletries	1	Consumable
4	TV	Electronics	0	Other
5	Apple	Fruits and vegetables	2	Consumable

`wildmatch`-funktionen returnerar indexet för söksträngen i det första måttet. I den andra åtgärden används detta resultat sedan i en `If`-funktion för att ge anpassade resultat. Observera att funktionen `wildmatch` inte är skiftlägeskänslig och tillåter jokertecken (`*` och `?`).

Exempel – Utvärdera och klassificera kunddata med hjälp av wildmatch

Diagramuttryck

Översikt

En datauppsättning innehåller kunddata. Du vill identifiera och kategorisera de kunder vars region och branschkategori uppfyller dina klassificeringskrav.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:

- ClientID
- ClientName
- Region
- Industry

Laddningsskript

```
Example:
LOAD * INLINE [
ClientID, ClientName, Region, Industry
1, "Acme Corp", "north territory", "Vehicle Manufacturing"
2, "Beta Ltd", "East location", "Science & Technology"
3, "Gamma Inc", "west", "Retail Sales"
4, "Delta LLC", "North", "Technology"
5, "Epsilon GmbH", "South", "healthcare"
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- ClientID
- ClientName
- Region
- Industry

Skapa följande mått:

`=If(wildmatch(Region, 'North*', 'East*') AND wildmatch(Industry, '*Technology*', '*Manufacturing*'), 'Target Segment', 'Secondary Segment')`, för att returnera indexet för sökströngen. Om det hittas returneras `Target Segment`. Om det inte hittas returneras `Secondary Segment`.

Resultattabell

ClientID	ClientName	Region	Industry	If(wildmatch(Region, 'North*', 'East*') AND wildmatch(Industry, '*Technology*', '*Manufacturing*'), 'Target Segment', 'Secondary Segment')
1	Acme Corp	north territory	Vehicle Manufacturing	Target Segment
2	Beta Ltd	East location	Science & Technology	Target Segment
3	Gamma Inc	west	Retail Sales	Secondary Segment

ClientID	ClientName	Region	Industry	If(wildmatch(Region, 'North*', 'East*') AND wildmatch(Industry, '*Technology*', '*Manufacturing*'), 'Target Segment', 'Secondary Segment')
4	Delta LLC	North	Technology	Target Segment
5	Epsilon GmbH	South	healthcare	Secondary Segment

`wildmatch`-funktionen söker efter text och returnerar ett index som sedan kan användas för att utföra ytterligare analyser och dataklassificering. Exempelvis klassificeras `Acme Corp` som ett `Target Segment` eftersom det uppfyller `wildmatch`-funktionens sökkrav för fälten `Region` och `Industry`. Observera att funktionen `wildmatch` inte är skiftlägeskänslig och tillåter jokertecken (`*` och `?`).

Exempel: laddningsskript som använder wildmatch

Laddningsskript

Översikt

Du kan använda wildmatch för att ladda en delmängd av dina data. Du kan till exempel returnera ett numeriskt värde för ett uttryck i funktionen. Sedan kan du begränsa vilka data som laddas utifrån det numeriska värdet. Wildmatch returnerar 0 om det inte finns någon matchning. Alla uttryck som inte har någon matchning i det här exemplet returnerar därför 0 och `WHERE`-satsen utesluter dem från dataladdningen.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i två datatabeller som heter:
  - `Transactions`: Innehåller transaktionsfält
  - `Transaction_Buckets`: Skapar två nya fält `Customer` och `Color code - Black, Blue, blue, red`. `where`-satsen läser in resultaten från `wildmatch`-funktionen.

Laddningsskript

```
Transactions:
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, m, orange
3752, 20180916, 5.75, 1, 5646471, S, blue
3753, 20180922, 125.00, 7, 3036491, l, Black
3754, 20180922, 484.21, 13, 049681, xs, Red
```

```
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
];

/*
Create new table called Transaction_Buckets
Create new fields called Customer, and Color code - Black, Blue, blue, red
Load Transactions table.
Wildmatch returns 1 for 'Black', 'Blue', and 'blue', and 2 for 'Red'.
Only values that returned numeric value greater than 0
are loaded by WHERE statement into Transactions_Buckets table.
*/

Transaction_Buckets:
Load
   customer_id,
   customer_id as [Customer],
   color_code as [Color Code Black, Blue, blue, Red]
Resident Transactions
Where wildmatch(color_code,'Bl*','R??') > 0;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `[Color code - Black, Blue, blue, red]`
- `Customer`

Resultattabellen visar utdata vid användning av wildmatch-funktionen i laddningsskriptet.

Color code - Black, Blue, blue, red	Customer
Black	203521
Black	3036491
Blue	2038593
blue	5646471
Red	049681
Red	2038593

Exempel – diagramuttryck som använder wildmatch

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `Cities`
    - `Count`

Laddningsskript

```
Example:
Load * inline [Cities, Count
Toronto, 123
Toronto, 234
Toronto, 231
Boston, 32
Boston, 23
Boston, 1341
Beijing, 234
Beijing, 45
Beijing, 235
Stockholm, 938
Stockholm, 39
Stockholm, 189
zurich, 2342
zurich, 9033
zurich, 0039];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Cities`

Skapa följande mått:

- `=wildmatch( Cities,'Tor*','?ton','Beijing','*urich')` , för att returnera indexet för söksträngen.
- `=wildmatch( Cities,'Tor*','???ton','Beijing','Stockholm','*urich')`, för att returnera indexet för söksträngen.
- `=Cities & ' - ' & wildmatch ( Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')`, för att returnera `Cities`-fältet och indexet för söksträngen.

Det första måttet i tabellen nedan returnerar `0` för `Stockholm`, eftersom `Stockholm`inte finns med i listan med uttryck i `wildmatch`-funktionen. Den returnerar också `0` för `Boston` eftersom `?` bara matchar på ett enda tecken.

Resultattabell

Cities	wildmatch(Cities,'Tor*','?ton','Beijing','*urich')	wildmatch(Cities,'Tor*','???ton','Beijing','Stockholm','*urich')	Cities & ' - ' & wildmatch (Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')
Beijing	3	3	Beijing - 3
Boston	0	2	Boston - 2
Stockholm	0	4	Stockholm - 4
Toronto	1	1	Toronto - 1
zurich	4	5	zurich - 5

Exempel – anpassat diagramuttryck som använder wildmatch

Diagramuttryck

Översikt

Du kan använda `wildmatch` för att utföra en anpassad sortering för ett uttryck. I det här exemplet används samma datauppsättning som i föregående exempel.

Som standard sorteras kolumner numeriskt eller alfabetiskt, beroende på datatyp.

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Cities`

Resultattabell som visar standardsorteringsordningen

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Gör följande för att ändra sorteringsordningen:

1. Öppna delavsnittet **Sortering** för ditt diagram i **egenskapspanelen**.
2. Inaktivera automatisk sortering för den kolumn du vill använda anpassad sortering för.

3. Radera kryssrutorna **Sortera numeriskt** och **Sortera alfabetiskt**.
4. Välj **Sortera efter uttryck** och ange sedan ett uttryck som liknar följande:
   `=wildmatch( Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')`
   Sorteringsordningen för kolumnen Cities ändras.

Resultattabell som visar sorteringsordningen med hjälp av *wildmatch*-funktionen

Cities
Toronto
Boston
Beijing
Stockholm
zurich

8.6 Räknefunktioner

I det här avsnittet beskrivs funktioner relaterade till posträknare under utvärdering av **LOAD**-satser i dataladdningsskriptet. Den enda funktion som kan användas i diagramuttryck är **RowNo()**.

En del räknefunktioner har inga parametrar, men de avslutande parenteserna är ändå obligatoriska.

Räknefunktioner - en översikt

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

autonumber

Denna skriptfunktion returnerar ett unikt heltal för varje distinkt utvärderat värde av *expression* som påträffas under skriptexekveringen. Funktionen kan exempelvis användas för att skapa en kompakt minnesrepresentation av en komplex nyckel.

```
autonumber (expression[ , AutoID])
```

autonumberhash128

Denna skriptfunktion beräknar en 128-bitars hashning av de kombinerade indatauttrycksvärdena och returnerar ett unikt heltalsvärde för varje distinkt hash-värde som påträffas under skriptexekveringen. Funktionen kan exempelvis användas för att skapa en kompakt minnesrepresentation av en komplex nyckel.

```
autonumberhash128 (expression {, expression})
```

autonumberhash256

Denna skriptfunktion beräknar en 256-bitars hashning av de kombinerade indatauttrycksvärdena och returnerar ett unikt heltalsvärde för varje distinkt hash-värde som påträffas under skriptexekveringen. Funktionen kan exempelvis användas för att skapa en kompakt minnesrepresentation av en komplex nyckel.

```
autonumberhash256 (expression {, expression})
```

IterNo

Denna skriptfunktion returnerar ett heltal som anger för vilken gång i ordningen en post utvärderas i en **LOAD**-sats med ett **while**-tillägg. Den första upprepningen får nummer 1. Funktionen **IterNo** är endast meningsfull om den används tillsammans med en **while**-sats.

```
IterNo ( )
```

RecNo

Denna skriptfunktion returnerar ett heltal som motsvarar numret på den rad i den aktuella tabellen som läses vid det givna tillfället. Första posten får nummer 1.

```
RecNo ( )
```

RowNo - script function

Denna funktion returnerar ett heltal som anger den aktuella radens placering i den resulterande interna Qlik Sense-tabellen. Första raden får nummer 1.

```
RowNo ( )
```

RowNo - chart function

RowNo() returnerar numret på den aktuella raden i det aktuella kolumnsegmentet i en tabell. För bitmappsdiagram returnerar **RowNo()** numret på den aktuella raden i diagrammets raka tabellmotsvarighet.

```
RowNo - diagramfunktion([TOTAL])
```

autonumber

Denna skriptfunktion returnerar ett unikt heltal för varje distinkt utvärderat värde av *expression* som påträffas under skriptexekveringen. Funktionen kan exempelvis användas för att skapa en kompakt minnesrepresentation av en komplex nyckel.

> *Du kan enbart koppla **autonumber**-nycklar som har skapats i samma dataladdning, eftersom heltalet genereras i enlighet med den ordning som tabellen laddas i. Om du behöver använda nycklar som är varaktiga mellan dataladdningar, fristående från källdatasortering, bör du använda funktionerna **hash128**, **hash160** eller **hash256**.*

Syntax:

```
autonumber(expression[ , AutoID])
```

Argument:

Argument	Beskrivning
AutoID	För att skapa flera räkneinstanser i de fall där funktionen **autonumber** används för flera olika nycklar inom ett skript, kan man använda den frivilliga parametern *AutoID* för att namnge instanserna.

Exempel: Skapa en sammansatt nyckel

I det här exemplet skapar vi en sammansatt nyckel med hjälp av funktionen **autonumber** för att spara minne. Exemplet är kortfattat eftersom det är avsett som en illustration, men blir meningsfullt med en tabell som innehåller ett stort antal rader.

Exempeldata

Region	Year	Month	Sales
North	2014	May	245
North	2014	May	347
North	2014	June	127
South	2014	June	645
South	2013	May	367
South	2013	May	221

Källdata laddas med inline-data. Sedan lägger vi till en föregående load som skapar en sammansatt nyckel från fälten Region, Year och Month.

```
RegionSales:
LOAD *,
AutoNumber(Region&Year&Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May, 367
South, 2013,   May,    221
];
```

Den resulterande tabellen ser ut så här:

Resultattabell

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1
North	2014	June	127	2
South	2014	June	645	3
South	2013	May	367	4
South	2013	May	221	4

I det här exemplet kan du hänvisa till RYMkey, exempelvis 1, i stället för strängen "North2014May" om du behöver länka till en annan tabell.

Nu laddar vi en källtabell med kostnader på liknande sätt. Fälten Region, Year och Month undantas i föregående laddning för att undvika att skapa en syntetisk nyckel. Vi skapar redan en sammansatt nyckel med funktionen **autonumber** som länkar tabellerna.

```
RegionCosts:
LOAD Costs,
AutoNumber(Region&Year&Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Costs
South, 2013,   May,    167
North, 2014,   May,    56
North, 2014,   June,   199
South, 2014,   June,   64
South, 2013,   May, 172
South, 2013,   May,    126
];
```

Nu kan vi lägga till en tabellvisualisering på ett ark och lägga till fälten Region, Year och Month samt summamått för försäljning och kostnader. Tabellen ser då ut så här:

Resultattabell

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

autonumberhash128

Denna skriptfunktion beräknar en 128-bitars hashning av de kombinerade indatauttrycksvärdena och returnerar ett unikt heltalsvärde för varje distinkt hash-värde som påträffas under skriptexekveringen. Funktionen kan exempelvis användas för att skapa en kompakt minnesrepresentation av en komplex nyckel.

*Du kan enbart koppla **autonumberhash128**-nycklar som har skapats i samma dataladdning, eftersom heltalet genereras i enlighet med den ordning som tabellen laddas i. Om du behöver använda nycklar som är varaktiga mellan dataladdningar, fristående från källdatasortering, bör du använda funktionerna **hash128**, **hash160** eller **hash256**.*

Syntax:

```
autonumberhash128(expression {, expression})
```

Exempel: Skapa en sammansatt nyckel

I det här exemplet skapar vi en sammansatt nyckel med hjälp av funktionen **autonumberhash128** för att spara minne. Exemplet är kortfattat eftersom det är avsett som en illustration, men blir meningsfullt med en tabell som innehåller ett stort antal rader.

Exempeldata

Region	Year	Month	Sales
North	2014	May	245
North	2014	May	347
North	2014	June	127
South	2014	June	645
South	2013	May	367
South	2013	May	221

Källdata laddas med inline-data. Sedan lägger vi till en föregående load som skapar en sammansatt nyckel från fälten Region, Year och Month.

```
RegionSales:
LOAD *,
AutoNumberHash128(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May, 367
South, 2013,   May,    221
];
```

Den resulterande tabellen ser ut så här:

Resultattabell

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1
North	2014	June	127	2

Region	Year	Month	Sales	RYMkey
South	2014	June	645	3
South	2013	May	367	4
South	2013	May	221	4

I det här exemplet kan du hänvisa till RYMkey, exempelvis 1, i stället för strängen "North2014May" om du behöver länka till en annan tabell.

Nu laddar vi en källtabell med kostnader på liknande sätt. Fälten Region, Year och Month undantas i föregående laddning för att undvika att skapa en syntetisk nyckel. Vi skapar redan en sammansatt nyckel med funktionen **autonumberhash128** som länkar tabellerna.

```
RegionCosts:
LOAD Costs,
AutoNumberHash128(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Costs
South, 2013,   May,    167
North, 2014,   May,    56
North, 2014,   June,   199
South, 2014,   June,   64
South, 2013,   May, 172
South, 2013,   May,    126
];
```

Nu kan vi lägga till en tabellvisualisering på ett ark och lägga till fälten Region, Year och Month samt summamått för försäljning och kostnader. Tabellen ser då ut så här:

Resultattabell

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

autonumberhash256

Denna skriptfunktion beräknar en 256-bitars hashning av de kombinerade indatauttrycksvärdena och returnerar ett unikt heltalsvärde för varje distinkt hash-värde som påträffas under skriptexekveringen. Funktionen kan exempelvis användas för att skapa en kompakt minnesrepresentation av en komplex nyckel.

*Du kan enbart koppla **autonumberhash256**-nycklar som har skapats i samma dataladdning, eftersom heltalet genereras i enlighet med den ordning som tabellen laddas i. Om du behöver använda nycklar som är varaktiga mellan dataladdningar, fristående från källdatasortering, bör du använda funktionerna **hash128**, **hash160** eller **hash256**.*

Syntax:

```
autonumberhash256(expression {, expression})
```

Exempel: Skapa en sammansatt nyckel

I det här exemplet skapar vi en sammansatt nyckel med hjälp av funktionen **autonumberhash256** för att spara minne. Exemplet är kortfattat eftersom det är avsett som en illustration, men blir meningsfullt med en tabell som innehåller ett stort antal rader.

Exempeltabell

Region	Year	Month	Sales
North	2014	May	245
North	2014	May	347
North	2014	June	127
South	2014	June	645
South	2013	May	367
South	2013	May	221

Källdata laddas med inline-data. Sedan lägger vi till en föregående load som skapar en sammansatt nyckel från fälten Region, Year och Month.

```
RegionSales:
LOAD *,
AutoNumberHash256(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May, 367
South, 2013,   May,    221
];
```

Den resulterande tabellen ser ut så här:

Resultattabell

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1
North	2014	June	127	2
South	2014	June	645	3
South	2013	May	367	4
South	2013	May	221	4

I det här exemplet kan du hänvisa till RYMkey, exempelvis 1, i stället för strängen "North2014May" om du behöver länka till en annan tabell.

Nu laddar vi en källtabell med kostnader på liknande sätt. Fälten Region, Year och Month undantas i föregående laddning för att undvika att skapa en syntetisk nyckel. Vi skapar redan en sammansatt nyckel med funktionen **autonumberhash256** som länkar tabellerna.

```
RegionCosts:
LOAD Costs,
AutoNumberHash256(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Costs
South, 2013,   May,    167
North, 2014,   May,    56
North, 2014,   June,   199
South, 2014,   June,   64
South, 2013,   May, 172
South, 2013,   May,    126
];
```

Nu kan vi lägga till en tabellvisualisering på ett ark och lägga till fälten Region, Year och Month samt summamått för försäljning och kostnader. Tabellen ser då ut så här:

Resultattabell

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

IterNo

Denna skriptfunktion returnerar ett heltal som anger för vilken gång i ordningen en post utvärderas i en **LOAD**-sats med ett **while**-tillägg. Den första upprepningen får nummer 1. Funktionen **IterNo** är endast meningsfull om den används tillsammans med en **while**-sats.

Syntax:

```
IterNo( )
```

Exempel och resultat:

Exempel:

```
LOAD
   IterNo() as Day,
   Date( StartDate + IterNo() - 1 ) as Date
   While StartDate + IterNo() - 1 <= EndDate;



LOAD * INLINE

[StartDate, EndDate

2014-01-22, 2014-01-26

];
```

Denna **LOAD**-sats genererar en post per datum inom det intervall som är angivet med **StartDate** och **EndDate**.

I det här exemplet ser den resulterande tabellen ut så här:

Resultattabell

Day	Date
1	2014-01-22
2	2014-01-23
3	2014-01-24
4	2014-01-25
5	2014-01-26

RecNo

Denna skriptfunktion returnerar ett heltal som motsvarar numret på den rad i den aktuella tabellen som läses vid det givna tillfället. Första posten får nummer 1.

Syntax:

```
RecNo( )
```

I motsats till **RowNo( )**, som räknar rader i den resulterande Qlik Sense-tabellen, räknar **RecNo( )** posterna i rådatatabellen och nollställs när en rådatatabell konkateneras med en annan.

Exempel: Dataladdningsskript

Laddning av rådatatabell

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Laddar post och radnummer för markerade rader:

```
QTab:

LOAD *,

RecNo( ),

RowNo( )

resident Tab1 where A<>2;



LOAD

C as A,

D as B,

RecNo( ),

RowNo( )

resident Tab2 where A<>5;
```

```
//We don't need the source tables anymore, so we drop them

Drop tables Tab1, Tab2;
```

Resultatet blir en intern tabell i Qlik Sense:

Resultattabell

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo

Denna funktion returnerar ett heltal som anger den aktuella radens placering i den resulterande interna Qlik Sense-tabellen. Första raden får nummer 1.

Syntax:

```
RowNo( [TOTAL])
```

I motsats till **RecNo( )**, som räknar posterna i rådatatabellen, räknar **RowNo( )**-funktionen inte poster som uteslutits av **where**-tillägg och börjar inte om på 1 när rådatatabeller konkateneras med varandra.

> *Om du använder föregående load, det vill säga ett antal **LOAD** -flervärdessatser som läses från samma tabell kan du bara använda **RowNo( )** i den översta **LOAD** -satsen. Om du använder **RowNo( )** i de följande **LOAD** -satserna returneras 0.*

Exempel: Dataladdningsskript

Laddning av rådatatabell

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Laddar post och radnummer för markerade rader:

```
QTab:

LOAD *,

RecNo( ),

RowNo( )

resident Tab1 where A<>2;


LOAD

C as A,

D as B,

RecNo( ),

RowNo( )

resident Tab2 where A<>5;


//We don't need the source tables anymore, so we drop them

Drop tables Tab1, Tab2;
```

Resultatet blir en intern tabell i Qlik Sense:

Resultattabell

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo - diagramfunktion

RowNo() returnerar numret på den aktuella raden i det aktuella kolumnsegmentet i en tabell. För bitmappsdiagram returnerar **RowNo()** numret på den aktuella raden i diagrammets raka tabellmotsvarighet.

Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner, utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

Kolumnsegment

Region	Country	Population	Rank(Population)
Americas	Mexico	128,932,753	2
Americas	Canada	37,742,154	3
Americas	United States of America	331,002,651	1
Europe	Sweden	10,099,265	4
Europe	United Kingdom	67,886,011	2
Europe	France	65,273,511	3
Europe	Germany	83,783,942	1

Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen.

Syntax:

```
RowNo([TOTAL])
```

Returnerad datatyp: heltal

Argument:

Argument	Beskrivning
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

Exempel: diagramuttryck som använder RowNo

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB|1|25| 25
Canutility|AA|3|8|15
Canutility|CC|5|4|19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

Diagramuttryck

Skapa en tabellvisualisering i ett Qlik Sense-ark med **Customer** och **UnitSales** som dimensioner. Lägg till `RowNo( )` och `RowNo(TOTAL)` som mått med etiketten **Rad i segment** respektive **Row Number**. Lägg till följande uttryck i tabellen som ett mått:

```
If( RowNo( )=1, 0, UnitSales / Above( UnitSales ))
```

Resultat

Customer	UnitSales	Row in Segment	Row Number	If(RowNo( )=1, 0, UnitSales / Above(UnitSales))
Astrida	4	1	1	0
Astrida	9	2	2	2.25
Astrida	10	3	3	1.1111111111111
Betacab	2	1	4	0
Betacab	5	2	5	2.5
Betacab	25	3	6	5
Canutility	4	1	7	0
Canutility	8	2	8	2
Divadip	1	1	9	0
Divadip	4	2	10	4

Förklaring

Kolumnen **Row in Segment** visar resultatet 1,2,3 för kolumnsegmentet som innehåller värdena i UnitSales för kunden Astrida. Radnumreringen börjar sedan på 1 igen för nästa kolumnsegment, som är Betacab.

Kolumnen **Row Number** bortser från dimensionerna på grund av argumentet `TOTAL` för `RowNo()` och räknar raderna i tabellen.

Det här uttrycket returnerar 0 för den första raden i varje kolumnsegment, så kolumnen visar:

0, 2,25, 1,1111111, 0, 2,5, 5, 0, 2, 0 och 4.

Se även:

p *Above - diagramfunktion (page 1459)*

8.7 Datum- och tidsfunktioner

Datum- och tidsfunktionerna i Qlik Sense används för att konvertera datum- och tidsvärden. Alla funktioner kan användas både i dataladdningsskriptet och diagramuttryck.

Funktionerna bygger på ett serienummer för datum och tid som motsvarar antalet dagar sedan den 30 december 1899. Heltalet motsvarar dagen och fraktionen motsvarar tiden på dagen.

Qlik Sense använder parameterns numeriska värde, så ett tal är giltigt som parameter även när det inte är formaterat som ett datum eller en tidspunkt. Om parametern inte motsvarar ett numeriskt värde, exempelvis eftersom den är en sträng, försöker Qlik Sense tolka strängen enligt datum- och tidsmiljövariablerna.

Om du använder ett tidformat i parametern som inte motsvarar operativsystemets inställningar, kan Qlik Sense inte göra en korrekt tolkning. För att lösa detta kan du antingen ändra inställningarna eller använda en tolkningsfunktion.

I exemplen för varje funktion antas standardformaten för tid och datum hh:mm:ss och YYYY-MM-DD (ISO 8601).

Qlik Sense ignorerar alla parametrar för sommartid när en tidsmarkör bearbetas med en datum- eller tidsfunktion, såvida inte datum- eller tidsfunktionen innehåller en geografisk position.

Exempelvis skulle `ConvertToLocalTime( filetime('Time.qvd'), 'Paris')` *använda parametrar för sommartid, medan* `ConvertToLocalTime(filetime('Time.qvd'), 'GMT-01:00')` *inte skulle använda parametrar för sommartid.*

Datum- och tidsfunktioner – en översikt

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Tidsuttryck

second

Denna funktion returnerar ett heltal som motsvarar sekunden om decimaldelen av **expression** tolkas som en tidpunkt enligt standardtolkningen av tal.

```
second (expression)
```

minute

Denna funktion returnerar ett heltal som motsvarar minuten om decimaldelen av **expression** tolkas som tid enligt standardtolkningen av tal.

```
minute (expression)
```

hour

Denna funktion returnerar ett heltal som motsvarar timmen om decimaldelen av **expression** tolkas som en tidpunkt enligt standardtolkningen av tal.

```
hour (expression)
```

day

Denna funktion returnerar ett heltal som motsvarar dagen om decimaldelen av **expression** tolkas som datum enligt standardtolkningen av tal.

```
day (expression)
```

week

Denna funktion returnerar ett heltal som motsvarar veckonumret enligt ISO 8601. Veckonumret beräknas utifrån datumtolkningen av uttrycket, enligt standardtalformatet.

```
week (expression)
```

month

Denna funktion returnerar ett dualt värde: månadsnamnet som det är definierat i miljövariabeln **MonthNames** och ett heltal mellan 1 och 12. Månaden beräknas utifrån datumtolkningen av uttrycket, enligt standardtalformatet.

```
month (expression)
```

quarter

Funktionen **quarter** returnerar ett heltal som representerar kvartalet för det angivna datumet. Det returnerade värdet är mellan 1 och 4. Funktionen kan användas i både laddningsskript och diagramuttryck.

```
quarter (date [,
first_month_of_year]

)
```

year

Denna funktion returnerar ett heltal som motsvarar året om **expression** tolkas som ett datum enligt standardtolkningen av tal.

```
year (expression)
```

weekyear

Funktionen returnerar det år som veckonumret hör till enligt miljövariablerna. Veckonummer går från 1 till cirka 52.

```
weekyear (expression)
```

weekday

Denna funktion returnerar ett dualt värde med:

- Ett namn på en dag som definierat i miljövariabeln **DayNames**.
- Ett heltal mellan 0 och 6 som motsvarar den nominella veckodagen (0-6).

```
weekday (date)
```

Tidsmarkörer

now

I För klienthanterat Qlik Sense, returnerar denna funktion en tidsmarkör för aktuell tid. Funktionen returnerar värden i systemvariabelformatet **TimeStamp**. Som standard är värdet för **timer_mode** 1.

```
now ([ timer_mode])
```

today

Denna funktion returnerar aktuell tid. Funktionen returnerar värden i systemvariabelformatet `DateFormat`.

```
today ([timer_mode])
```

LocalTime

Denna funktion returnerar en tidsmarkör för aktuell tid från systemklockan för en angiven tidszon.

```
localtime ([timezone [, ignoreDST ]])
```

Make

makedate

Denna funktion returnerar ett datum beräknat utifrån året **YYYY**, månaden **MM** och dagen **DD**.

```
makedate (YYYY [ , MM [ , DD ] ])
```

makeweekdate

Funktionen returnerar ett datum som beräknats från året, veckonumret och veckodagen.

```
makeweekdate (YYYY [ , WW [ , D ] ])
```

maketime

Denna funktion returnerar en tid beräknat utifrån timmar **hh**, minuter **mm** och sekunder **ss**.

```
maketime (hh [ , mm [ , ss [ .fff ] ] ])
```

Övriga datumfunktioner

AddMonths

Denna funktion returnerar det datum som infaller **n** månader efter **startdate** eller, om **n** är negativt, det datum som infaller **n** månader före **startdate**.

```
addmonths (startdate, n , [ , mode])
```

AddYears

Denna funktion returnerar det datum som infaller **n** år efter **startdate** eller, om **n** är negativt, det datum som infaller**n** år före **startdate**.

```
addyears (startdate, n)
```

yeartodate

Denna funktion räknar ut om inmatningstidsmarkören hamnar inom året för datumet då skriptet senast lästes in och returnerar True om så är fallet, False om så inte är fallet.

```
yeartodate (date [ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Tidszoner

timezone

I För klienthanterat Qlik Sense returnerar den här funktionen tidszonen, enligt definitionen för den dator som kör Qlik-motorn.

```
timezone ( )
```

GMT

Denna funktion returnerar aktuell Greenwich Mean Time utifrån de regionala inställningarna.

```
GMT ( )
```

UTC

Returnerar aktuell Coordinated Universal Time.

```
UTC ( )
```

daylightsaving

Returnerar ett dubbelt värde som motsvarar den aktuella justeringen för sommartid, enligt systemtiden. I För klienthanterat Qlik Sense returnerar funktionen olika värden beroende på årstiden och aktuell systemtid. I Qlik Cloud använder funktionen alltid UTC-tid så det returnerade värdet är alltid ett dubbelt värde som indikerar inga tidsjusteringar.

```
daylightsaving ( )
```

converttolocaltime

Konverterar en UTC- eller GMT-tidsmarkör till lokal tid i form av ett dualt värde. Ett antal platser (städer, platser och tidszoner) i hela världen kan användas.

```
converttolocaltime (timestamp [, place [, ignore_dst=false]])
```

Ange tid

setdateyear

Den här funktionen tar som indata en **timestamp** och ett **year** och uppdaterar **timestamp** med det **year** som har angetts i indata.

```
setdateyear (timestamp, year)
```

setdateyearmonth

Den här funktionen tar som indata en **timestamp**, en **month** och ett **year** och uppdaterar **timestamp** med det **year** och den **month** som har angetts i indata.

```
setdateyearmonth (timestamp, year, month)
```

In...

inyear

Denna funktion returnerar True om **timestamp** ligger inom det år som innehåller **base_date**.

```
inyear (date, basedate , shift [, first_month_of_year = 1])
```

inyeartodate

Denna funktion returnerar True om **timestamp** ligger inom den del av året som innehåller **base_date** fram till och inklusive den sista millisekunden av **base_date**.

```
inyeartodate (date, basedate , shift [, first_month_of_year = 1])
```

inquarter

Denna funktion returnerar True om **timestamp** ligger inom det kvartal som innehåller **base_date**.

```
inquarter (date, basedate , shift [, first_month_of_year = 1])
```

inquartertodate

Denna funktion returnerar True om **timestamp** ligger inom den del av det kvartal som innehåller **base_date** fram till och inklusive den sista millisekunden av **base_date**.

```
inquartertodate (date, basedate , shift [, first_month_of_year = 1])
```

inmonth

Denna funktion returnerar True om **timestamp** ligger inom den månad som innehåller **base_date**.

```
inmonth (date, basedate , shift)
```

inmonthtodate

Returnerar True om **date** ligger inom den del av månaden som innehåller **basedate** fram till och inklusive den sista millisekunden av **basedate**.

```
inmonthtodate (date, basedate , shift)
```

inmonths

Med den här funktionen får vi reda på om en tidsmarkör faller inom samma månad, tvåmånadersperiod, fyramånadersperiod eller halvår som basdatum. Det går även att se om tidsmarkören finns inom en föregående eller senare tidsperiod.

```
inmonths (n, date, basedate , shift [, first_month_of_year = 1])
```

inmonthstodate

Med den här funktionen får vi reda på om en tidsmarkör finns inom delen av månaden, tvåmånadersperioden, kvartalet, fyramånadersperioden eller halvåret fram till och inklusive den sista millisekunden av `base_date`. Det går även att se om tidsmarkören finns inom en föregående eller senare tidsperiod.

```
inmonthstodate (n, date, basedate , shift [, first_month_of_year = 1])
```

inweek

Denna funktion returnerar True om **timestamp** ligger inom den vecka som innehåller **base_date**.

```
inweek (date, basedate , shift [, weekstart])
```

inweektodate

Denna funktion returnerar True om **timestamp** ligger inom den del av veckan som innehåller **base_date** fram till och inklusive den sista millisekunden av **base_date**.

```
inweektodate (date, basedate , shift [, weekstart])
```

inlunarweek

Denna funktion fastställer om **timestamp** ligger inom den sjudagarsperiod som innehåller **base_date**. Sjudagarsperioder i Qlik Sense definieras genom att räkna 1 januari som den första dagen i veckan. Bortsett från årets sista vecka kommer varje vecka att ha exakt sju dagar.

```
inlunarweek (date, basedate , shift [, weekstart])
```

inlunarweektodate

Denna funktion tar reda på om **timestamp** ligger inom delen av sjudagarsperioden fram till och inklusive den sista millisekunden av **base_date**. Sjudagarsperioder i Qlik Sense definieras genom att räkna 1 januari som den första dagen i veckan. Bortsett från årets sista vecka kommer varje vecka att ha exakt sju dagar.

```
inlunarweektodate (date, basedate , shift [, weekstart])
```

inday

Denna funktion returnerar True om **timestamp** ligger inom den dag som innehåller **base_timestamp**.

```
inday (timestamp, basetimestamp , shift [, daystart])
```

indaytotime

Denna funktion returnerar True om **timestamp** ligger inom den del av dagen som innehåller **base_timestamp** fram till och inklusive den exakta millisekunden för **base_timestamp**.

```
indaytotime (timestamp, basetimestamp , shift [, daystart])
```

Start... end

yearstart

Denna funktion returnerar en tidsmarkör som motsvarar starten av den första dagen i det år som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

```
yearstart ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör med den sista millisekunden av den sista dagen i det år som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

```
yearend ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearname

Denna funktion returnerar ett fyrsiffrigt år som visningsvärde med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden av den första dagen på det år som innehåller **date**.

```
yearname (date [, shift = 0 [, first_month_of_year = 1]] )
```

quarterstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden i det kvartal som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

```
quarterstart (date [, shift = 0 [, first_month_of_year = 1]])
```

quarterend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden i det kvartal som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

```
quarterend (date [, shift = 0 [, first_month_of_year = 1]])
```

quartername

Denna funktion returnerar ett visningsvärde med kvartalets månader (formaterat enligt skriptvariabeln **MonthNames**) och år med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden av kvartalets första dag.

```
quartername (date [, shift = 0 [, first_month_of_year = 1]])
```

monthstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden av den första dagen i den månad som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

```
monthstart (date [, shift = 0])
```

monthend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör med den sista millisekunden av den sista dagen i den månad som innehåller `date`. Det förvalda utdataformatet blir det `DateFormat` som har definierats i skriptet.

```
monthend (date [, shift = 0])
```

monthname

Denna funktion returnerar ett visningsvärde med månaden (formaterat enligt skriptvariabeln **MonthNames**) och året med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden av månadens första dag.

```
monthname (date [, shift = 0])
```

monthsstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden i månaden, tvåmånadersperioden, kvartalet, fyramånadersperioden eller halvåret som innehåller ett basdatum. Det går även att hitta tidsmarkören för en tidigare eller senare tidsperiod. Standardformatet för utdata är det

DateFormat som har angetts i skriptet.

```
monthsstart (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden i månaden, tvåmånadersperioden, kvartalet, fyramånadersperioden eller halvåret som innehåller ett basdatum. Det går även att hitta tidsmarkören för en föregående eller senare tidsperiod.

```
monthsend (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsname

Denna funktion returnerar ett visningsvärde som representerar månadsintervallet i perioden (formaterat enligt skriptvariabeln **MonthNames**) liksom året. Det underliggande numeriska värdet motsvarar en tidsmarkör för den första millisekunden i månaden, tvåmånadersperioden, kvartalet, fyramånadersperioden eller halvåret som innehåller ett basdatum.

```
monthsname (n, date [, shift = 0 [, first_month_of_year = 1]])
```

weekstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden på den första dagen (måndag) i den kalendervecka som innehåller **date**. Det förvalda utdataformatet är det **DateFormat** som har definierats i skriptet.

```
weekstart (date [, shift = 0 [,weekoffset = 0]])
```

weekend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden på den sista dagen i den kalendervecka som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

```
weekend (date [, shift = 0 [,weekoffset = 0]])
```

weekname

Denna funktion returnerar ett värde som visar år och veckonummer med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden på den första dagen i veckan som innehåller **date**.

```
weekname (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden för den första dagen i den sjudagarsperiod som innehåller **date**. Sjudagarsperioder i Qlik Sense definieras genom att räkna 1 januari som den första dagen i veckan. Bortsett från årets sista vecka kommer varje vecka att ha exakt sju dagar.

```
lunarweekstart (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden för den sista dagen i den sjudagarsperiod som innehåller **date**. Sjudagarsperioder i Qlik Sense definieras genom att räkna 1 januari som den första dagen i veckan. Bortsett från årets sista vecka kommer varje vecka att ha exakt sju dagar.

```
lunarweekend (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekname

Denna funktion returnerar ett visningsvärde som visar året och sjudagarsperiodsnumret som motsvarar en tidsmarkör för den första millisekunden på den första dagen i sjudagarsperioden som innehåller **date**. Sjudagarsperioder i Qlik Sense definieras genom att räkna 1 januari som den första dagen i veckan. Bortsett från årets sista vecka kommer varje vecka att ha exakt sju dagar.

```
lunarweekname (date [, shift = 0 [,weekoffset = 0]])
```

daystart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör med den första millisekunden av den dag som ingår i argumentet **time**. Det förvalda utdataformatet blir det **TimestampFormat** som har definierats i skriptet.

```
daystart (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden av den dag som ingår i **time**. Det förvalda utdataformatet blir det **TimestampFormat** som har definierats i skriptet.

```
dayend (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayname

Denna funktion returnerar ett värde som visar datumet med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden på dagen som innehåller **time**.

```
dayname (timestamp [, shift = 0 [, dayoffset = 0]])
```

Dagsnumrering

age

Funktionen **age** returnerar åldern vid tidpunkten för **timestamp** (i fyllda år) för någon som är född på **date_of_birth**.

```
age (timestamp, date_of_birth)
```

networkdays

Funktionen **networkdays** returnerar antalet arbetsdagar (måndag-fredag) mellan och inklusive **start_date** och **end_date** med hänsyn till eventuella **holiday**.

```
networkdays (start:date, end_date {, holiday})
```

firstworkdate

Funktionen **firstworkdate** returnerar senast möjliga startdatum för att uppnå **no_of_workdays** (måndag-fredag) som tar slut senast **end_date** med hänsyn till alla eventuella helgdagar. **end_date** och **holiday** ska vara giltiga datum eller tidsmarkörer.

```
firstworkdate (end_date, no_of_workdays {, holiday} )
```

lastworkdate

Funktionen **lastworkdate** returnerar det tidigaste slutdatumet för att uppnå **no_of_workdays** (måndag-fredag) med början vid **start_date** med hänsyn till alla eventuella **holiday**. **start_date** och **holiday** ska vara giltiga datum eller tidsmarkörer.

```
lastworkdate (start_date, no_of_workdays {, holiday})
```

daynumberofyear

Beräknar dagens nummer på året där tidsmarkören finns. Beräkningen är gjord från den första millisekunden av den första dagen på året, men startpunkten för den första månaden kan flyttas.

```
daynumberofyear (date[,firstmonth])
```

daynumberofquarter

Beräknar dagens nummer i kvartalet där tidsmarkören finns. Den här funktionen används när du skapar en Master Calendar

```
daynumberofquarter (date[,firstmonth])
```

addmonths

Denna funktion returnerar det datum som infaller **n** månader efter **startdate** eller, om **n** är negativt, det datum som infaller **n** månader före **startdate**.

Syntax:

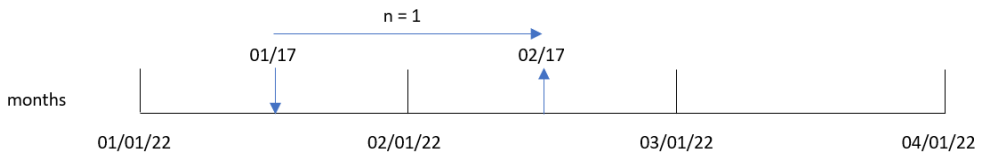
```
AddMonths(startdate, n , [ , mode])
```

Returnerad datatyp: dual

Funktionen `addmonths()` lägger till eller drar av ett definierat antal månader, `n`, från ett `startdate` och returnerar resulterande data.

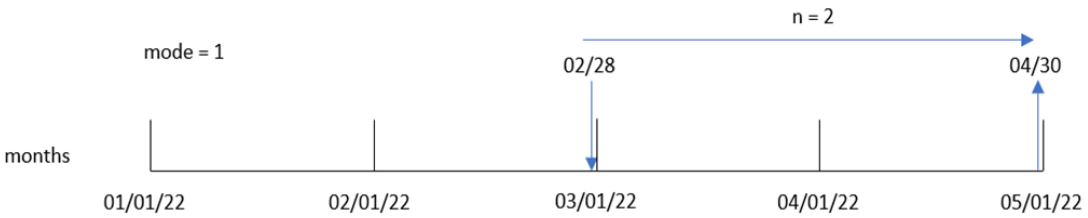
Argumentet `mode` kommer att påverka värdena `startdate` den 28 eller senare i månaden. Genom att sätta argumentet `mode` till 1 returnerar funktionen `addmonths()` ett datum som är lika i relativt avstånd till månadens slut som `startdate`.

Exempeldiagram för `addmonths()`-funktionen



Exempel: 28 februari är den sista dagen i månaden. Om funktionen `addmonths()`, med ett `mode` av 1, används för att returnera datumet två månader senare kommer funktionen att returnera den sista dagen i april, 30 april.

Exempeldiagram för `addmonths()`-funktionen med `mode=1`



Argument

Argument	Beskrivning
startdate	Startdatumet som en tidsmarkör, exempelvis '2012-10-12'.
n	Antal månader som ett positivt eller negativt heltal.
mode	Anger om månaden läggs till relativt månadens början eller månadens slut. Standardläget är 0 när den läggs till relativt månadens början. Ange läget 1 när den läggs till relativt månadens slut. När läget är inställt på 1 och indatadatumet är 28 eller högre, kontrollerar funktionen hur många dagar som återstår före månadens slut på startdatumet. Samma antal datar som krävs för att nå månadens slut anges på det returnerade datumet.

Användning

Funktionen `addmonths()` används oftast i ett uttryck för att hitta ett datum ett angivet antal månader före eller efter en tidsperiod.

Funktionen `addmonths()` kan exempelvis användas för att identifiera slutdatumet för mobiltelefoniavtal.

Exempel på funktioner

Exempel	Resultat
`addmonths ('01/29/2003' ,3)`	Returnerar 04/29/2003.
`addmonths ('01/29/2003',3,0)`	Returnerar 04/29/2003.
`addmonths ('01/29/2003',3,1)`	Returnerar 04/28/2003.
`addmonths ('01/29/2003',1,0)`	Returnerar 02/28/2003.
`addmonths ('01/29/2003',1,1)`	Returnerar 02/26/2003.
`addmonths ('02/28/2003',1,0)`	Returnerar 03/28/2003.
`addmonths ('02/28/2003',1,1)`	Returnerar 03/31/2003.
`addmonths ('01/29/2003',-3)`	Returnerar 10/29/2002.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner mellan 2020 och 2022 som laddas i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i `DateFormat`-systemvariabeln.
- Ett fält `two_months_later` skapas som returnerar datumet två månader efter att transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        addmonths(date,2) as two_months_later
        ;
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
```

```
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `two_months_later`

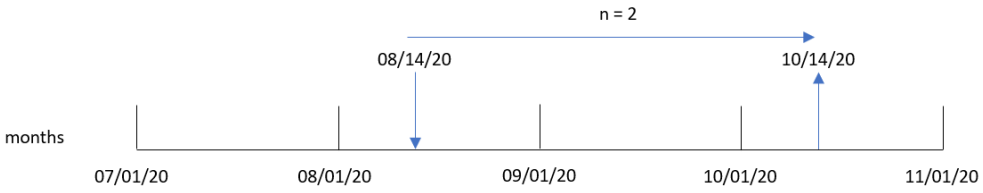
Resultattabell

datum	två månader senare
01/10/2020	2020-03-10
02/28/2020	2020-04-28
04/09/2020	2020-06-09
04/16/2020	2020-06-16
05/21/2020	2020-07-21
08/14/2020	2020-10-14
10/07/2020	2020-12-07
12/05/2020	2021-02-05
01/22/2021	2021-03-22
02/03/2021	2021-04-03
03/17/2021	2021-05-17
04/23/2021	23/06/2021
05/04/2021	04/07/2021
06/30/2021	30/08/2021
07/26/2021	26/09/2021
12/27/2021	27/02/2022

datum	två månader senare
02/02/2022	2022-04-02
02/26/2022	2022-04-26
03/07/2022	2022-05-07
03/11/2022	2022-05-11

`two_months_later`-fältet skapas i föregående LOAD-sats med hjälp av funktionen `addmonths()`. Det första argumentet som anges identifierar vilket datum som utvärderas. Det andra argumentet är antalet månader som ska läggas till eller dras från `startdate`. I det här fallet anges värdet 2.

Diagram för `addmonths()`-funktionen, exempel utan ytterligare argument



Transaktion 8193 ägde rum 14 augusti. Därför returnerar funktionen `addmonths()` 14 oktober 2020 för fältet `two_months_later`.

Exempel 2 – relativt månadsslut

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner från månadens slut under 2022, som laddas i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i systemvariabeln `DateFormat`.
- Ett fält `relative_two_months_prior` skapas som returnerar det relativa datumet vid månadens slut två månader före att transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        addmonths(date,-2,1) as relative_two_months_prior
        ;
```

```
Load
*
Inline
[
id,date,amount
8188,'01/28/2022',37.23
8189,'01/31/2022',57.54
8190,'02/28/2022',17.17
8191,'04/29/2022',88.27
8192,'04/30/2022',57.42
8193,'05/31/2022',53.80
8194,'08/14/2022',82.06
8195,'10/07/2022',40.39
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

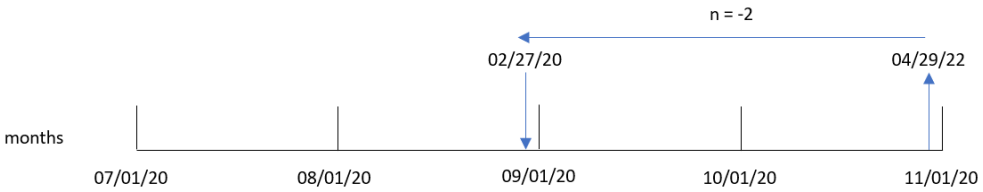
- `date`
- `relative_two_months_prior`

Resultattabell

datum	två månader tidigare
28/01/2022	11/27/2021
01/31/2022	11/30/2021
02/28/2022	12/31/2021
04/29/2022	2022-02-27
04/30/2022	02/28/2022
05/31/2022	03/31/2022
08/14/2022	06/14/2022
10/07/2022	08/07/2022

`relative_two_months_prior`-fältet skapas i föregående LOAD-sats med hjälp av funktionen `addmonths()`. Det första argumentet som anges identifierar vilket datum som utvärderas. Det andra argumentet är antalet månader som ska läggas till eller dras från `startdate`. I det här fallet anges värdet -2. Det sista argumentet är läget, med värdet 1, som tvingar funktionen att beräkna det relativa datumet vid månadens slut för alla datum större än eller lika med 28.

Diagram för funktionen addmonths(), exempel med n=-2



Transaktion 8191 ägde rum den 29 april 2022. Initialt skulle två månader tidigare ange månaden till februari. Därefter beräknar funktionen det relativa värdet för månadsslutet, eftersom funktionens tredje argumentet sätter läget till 1 och dagens värde till lägre än den 27:e. Funktionen identifierar att 29:e är den näst sista dagen i april och returnerar därför den näst sista dagen i februari, den 27:e.

Exempel 3 – Exempel på diagramobjekt

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som returnerar datumet två månader efter att transaktionen ägde rum skapas som en mätning i ett diagramobjekt.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
```

```
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

Skapa följande mått:

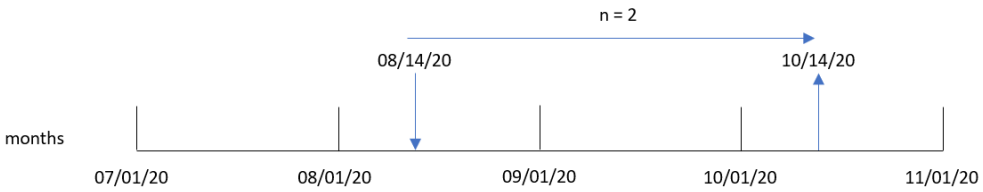
```
=addmonths(date,2)
```

Resultattabell

datum	=addmonths(date,2)
01/10/2020	2020-03-10
02/28/2020	2020-04-28
04/09/2020	2020-06-09
04/16/2020	2020-06-16
05/21/2020	2020-07-21
08/14/2020	2020-10-14
10/07/2020	2020-12-07
12/05/2020	2021-02-05
01/22/2021	2021-03-22
02/03/2021	2021-04-03
03/17/2021	2021-05-17
04/23/2021	23/06/2021
05/04/2021	04/07/2021
06/30/2021	30/08/2021
07/26/2021	26/09/2021
12/27/2021	27/02/2022
02/02/2022	2022-04-02
02/26/2022	2022-04-26
03/07/2022	2022-05-07
03/11/2022	2022-05-11

`two_months_later`-måttet skapas i ett diagramobjekt med hjälp av `addmonths()`-funktionen. Det första argumentet som anges identifierar vilket datum som utvärderas. Det andra argumentet är antalet månader som ska läggas till eller dras från `startdate`. I det här fallet anges värdet 2.

Diagram med exempel på `addmonths()`-funktionen, diagramobjekt



Transaktion 8193 ägde rum 14 augusti. Därför returnerar `addmonths()`-funktionen 14 oktober 2020 för `two_months_later`-fältet.

Exempel 4 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning läses in i en tabell som heter `Mobile_Plans`.
- Information med avtals-ID, startdatum, avtalslängd och månadsavgift.

Slutanvändaren vill ha ett diagramobjekt som visar slutdatum för varje telefoniavtal, sorterat på avtals-ID.

Laddningsskript

```
Mobile_Plans:
Load
*
Inline
[
contract_id,start_date,contract_length,monthly_fee
8188,'01/13/2020',18,37.23
8189,'02/26/2020',24,17.17
8190,'03/27/2020',36,88.27
8191,'04/16/2020',24,57.42
8192,'05/21/2020',24,53.80
8193,'08/14/2020',12,82.06
8194,'10/07/2020',18,40.39
8195,'12/05/2020',12,87.21
8196,'01/22/2021',12,95.93
8197,'02/03/2021',18,45.89
8198,'03/17/2021',24,36.23
8199,'04/23/2021',24,25.66
8200,'05/04/2021',12,82.77
```

```
8201,'06/30/2021',12,69.98
8202,'07/26/2021',12,76.11
8203,'12/27/2021',36,25.12
8204,'06/06/2022',24,46.23
8205,'07/18/2022',12,84.21
8206,'11/14/2022',12,96.24
8207,'12/12/2022',18,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `contract_id`
- `start_date`
- `contract_length`

Skapa följande mått för att beräkna slutet på varje avtal:

```
=addmonths(start_date,contract_length, 0)
```

Resultattabell

kontrakts-ID	startdatum	contract_length	=addmonths(start_date,contract_length,0)
8188	01/13/2020	18	07/13/2021
8189	02/26/2020	24	02/26/2022
8190	03/27/2020	36	03/27/2023
8191	04/16/2020	24	04/16/2022
8192	05/21/2020	24	05/21/2022
8193	08/14/2020	12	08/14/2021
8194	10/07/2020	18	04/07/2022
8195	12/05/2020	12	12/05/2021
8196	01/22/2021	12	2022-01-22
8197	02/03/2021	18	08/03/2022
8198	03/17/2021	24	03/17/2023
8199	04/23/2021	24	04/23/2023
8200	05/04/2021	12	05/04/2022
8201	06/30/2021	12	06/30/2022
8202	07/26/2021	12	07/26/2022
8203	12/27/2021	36	12/27/2024
8204	06/06/2022	24	06/06/2024

kontrakts-ID	startdatum	contract_length	=addmonths(start_date,contract_length,0)
8205	07/18/2022	12	07/18/2023
8206	11/14/2022	12	11/14/2023
8207	12/12/2022	18	06/12/2024

addyears

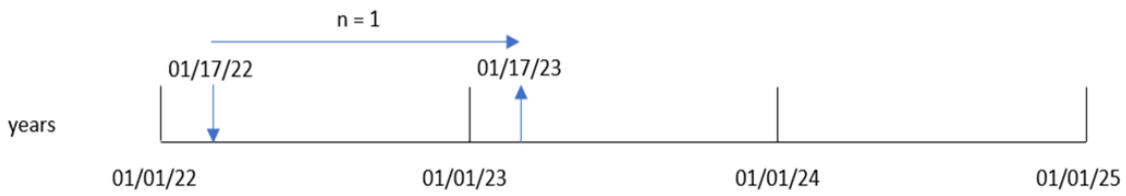
Denna funktion returnerar det datum som infaller **n** år efter **startdate** eller, om **n** är negativt, det datum som infallern år före **startdate**.

Syntax:

```
AddYears(startdate, n)
```

Returnerad datatyp: dual

Exempeldiagram för `addyears()`-funktionen



`addyears()`-funktionen lägger till eller drar av ett definierat antal år, `n`, från ett `startdate`. Därefter returneras det resulterande datumet.

Argument

Argument	Beskrivning
startdate	Startdatumet som en tidsmarkör, exempelvis '2012-10-12'.
n	Antal år som ett positivt eller negativt heltal.

Exempel på funktioner

Exempel	Resultat
`addyears ('01/29/2010',3)`	Returnerar 01/29/2013.
`addyears ('01/29/2010',-1)`	Returnerar 01/29/2009.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan

vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – enkelt exempel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner mellan 2020 och 2022 som laddas i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i `DateFormat`-systemvariabeln.
- Ett fält `two_years_later` skapas som returnerar datumet två år efter att transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        addyears(date,2) as two_years_later
        ;
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
```

```
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

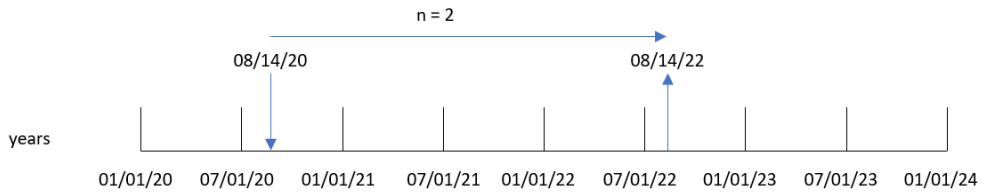
- `date`
- `two_years_later`

Resultattabell

datum	two_years_later
01/10/2020	01/10/2022
02/28/2020	02/28/2022
04/09/2020	04/09/2022
04/16/2020	04/16/2022
05/21/2020	2022-05-21
08/14/2020	08/14/2022
10/07/2020	10/07/2022
12/05/2020	12/05/2022
01/22/2021	01/22/2023
02/03/2021	02/03/2023
03/17/2021	03/17/2023
04/23/2021	04/23/2023
05/04/2021	05/04/2023
06/30/2021	06/30/2023
07/26/2021	07/26/2023
12/27/2021	12/27/2023
02/02/2022	02/02/2024
02/26/2022	02/26/2024
03/07/2022	03/07/2024
03/11/2022	03/11/2024

`two_years_later`-fältet skapas i föregående LOAD-sats med hjälp av funktionen `addyears()`. Det första argumentet som anges identifierar vilket datum som utvärderas. Det andra argumentet är antalet år som ska läggas till eller dras av från startdatumet. I det här fallet anges värdet 2.

Diagram för `addyears()`-funktionen, grundläggande exempel



Transaktion 8193 ägde rum 14 augusti 2020. Därför returnerar `addyears()`-funktionen 14 augusti 2022 för `two_years_later`-fältet.

Exempel 2 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner mellan år 2020 och 2022 som laddas i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i systemvariabeln `DateFormat`.

Skapa ett mått `prior_year_date` i diagramobjektet som returnerar datumet ett år före transaktionen äger rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
```

```
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

Skapa följande mått för att beräkna datumet ett år före varje transation:

```
=addyears(date,-1)
```

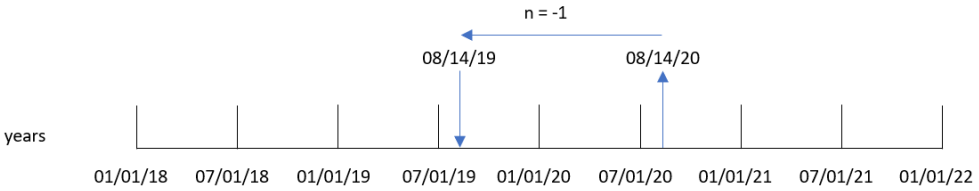
Resultattabell

datum	=addyears(date,-1)
01/10/2020	01/10/2019
02/28/2020	02/28/2019
04/09/2020	04/09/2019
04/16/2020	04/16/2019
05/21/2020	05/21/2019
08/14/2020	08/14/2019
10/07/2020	10/07/2019
12/05/2020	12/05/2019
01/22/2021	01/22/2020
02/03/2021	02/03/2020
03/17/2021	03/17/2020
04/23/2021	04/23/2020
05/04/2021	05/04/2020
06/30/2021	06/30/2020
07/26/2021	07/26/2020
12/27/2021	12/27/2020
02/02/2022	02/02/2021

datum	=addyears(date,-1)
02/26/2022	02/26/2021
03/07/2022	03/07/2021
03/11/2022	03/11/2021

`one_year_prior`-måttet skapas i ett diagramobjekt med hjälp av `addyears()`-funktionen. Det första argumentet som anges identifierar vilket datum som utvärderas. Det andra argumentet är antalet år som ska läggas till eller dras av från `startdate`. I det här fallet anges värdet -1.

Diagram med exempel på `addyears()`-funktionen, diagramobjekt



Transaktion 8193 ägde rum 14 augusti. Därför returnerar `addyears()`-funktionen 14 augusti 2019 för `one_year_prior`-fältet.

Exempel 3 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning läses in i en tabell som heter `Warranties`.
- Information med produkt-ID, inköpsdatum, garantilängd och inköpspriset.

Slutanvändaren vill ha ett diagramobjekt som visar garantins slutdatum för varje produkt, sorterat på produkt-ID.

Laddningsskript

```
Warranties:
Load
*
Inline
[
product_id,purchase_date,warranty_length,purchase_price
8188,'01/13/2020',4,32000
8189,'02/26/2020',2,28000
8190,'03/27/2020',3,41000
8191,'04/16/2020',4,17000
```

```
8192,'05/21/2020',2,25000
8193,'08/14/2020',1,59000
8194,'10/07/2020',2,12000
8195,'12/05/2020',3,12000
8196,'01/22/2021',4,24000
8197,'02/03/2021',1,50000
8198,'03/17/2021',2,80000
8199,'04/23/2021',3,10000
8200,'05/04/2021',4,30000
8201,'06/30/2021',3,30000
8202,'07/26/2021',4,20000
8203,'12/27/2021',4,10000
8204,'06/06/2022',2,25000
8205,'07/18/2022',1,32000
8206,'11/14/2022',1,30000
8207,'12/12/2022',4,22000
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `product_id`
- `purchase_date`
- `warranty_length`

Skapa följande mått för att beräkna slutdatumet för varje produkts garanti:

```
=addyears(purchase_date,warranty_length)
```

Resultattabell

product_id	purchase_date	warranty_length	=addyears(purchase_date,warranty_length)
8188	01/13/2020	4	01/13/2024
8189	02/26/2020	2	02/26/2022
8190	03/27/2020	3	03/27/2023
8191	04/16/2020	4	04/16/2024
8192	05/21/2020	2	05/21/2022
8193	08/14/2020	1	08/14/2021
8194	10/07/2020	2	10/07/2022
8195	12/05/2020	3	12/05/2023
8196	01/22/2021	4	01/22/2025
8197	02/03/2021	1	02/03/2022
8198	03/17/2021	2	03/17/2023

product_id	purchase_date	warranty_length	=addyears(purchase_date,warranty_length)
8199	04/23/2021	3	04/23/2024
8200	05/04/2021	4	05/04/2025
8201	06/30/2021	3	06/30/2024
8202	07/26/2021	4	07/26/2025
8203	12/27/2021	4	12/27/2025
8204	06/06/2022	2	06/06/2024
8205	07/18/2022	1	07/18/2023
8206	11/14/2022	1	11/14/2023
8207	12/12/2022	4	12/12/2026

age

Funktionen **age** returnerar åldern vid tidpunkten för **timestamp** (i fyllda år) för någon som är född på **date_of_birth**.

Syntax:

```
age(timestamp, date_of_birth)
```

Kan vara ett uttryck.

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
timestamp	Tidsmarkören, eller uttrycksresultatet som tidsmarkör, som antalet fyllda år ska beräknas fram till.
date_of_birth	Födelsedatum för personen vars ålder beräknas. Kan vara ett uttryck.

Exempel och resultat:

I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`age('25/01/2014', '29/10/2012')`	Returnerar 1.
`age('29/10/2014', '29/10/2012')`	Returnerar 2.

Exempel:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
Employees:

LOAD * INLINE [

Member|DateOfBirth

John|28/03/1989

Linda|10/12/1990

Steve|5/2/1992

Birg|31/3/1993

Raj|19/5/1994

Prita|15/9/1994

Su|11/12/1994

Goran|2/3/1995

Sunny|14/5/1996

Ajoa|13/6/1996

Daphne|7/7/1998

Biffy|4/8/2000

] (delimiter is |);

AgeTable:

Load *,

age('20/08/2015', DateOfBirth) As Age

Resident Employees;

Drop table Employees;
```

Den resulterande tabellen visar de returnerade värdena för age för varje post i tabellen.

Resultattabell

Member	DateOfBirth	Age
John	28/03/1989	26
Linda	10/12/1990	24
Steve	5/2/1992	23
Birg	31/3/1993	22
Raj	19/5/1994	21
Prita	15/9/1994	20
Su	11/12/1994	20
Goran	2/3/1995	20
Sunny	14/5/1996	19
Ajoa	13/6/1996	19
Daphne	7/7/1998	17
Biffy	4/8/2000	15

converttolocaltime

Konverterar en UTC- eller GMT-tidsmarkör till lokal tid i form av ett dualt värde. Ett antal platser (städer, platser och tidszoner) i hela världen kan användas.

Syntax:

```
ConvertToLocalTime(timestamp [, place [, ignore_dst=false]])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
timestamp	Tidsmarkören, eller uttrycksresultatet som tidsmarkör som ska konverteras.

Argument	Beskrivning
place	En plats eller tidszon från tabellen över giltiga platser och tidszoner nedan. Du kan också använda GMT eller UTC för att definiera lokal tid. Följande värden och tidsförskjutningsvärden är giltiga: • GMT • GMT-12:00 - GMT-01:00 • GMT+01:00 - GMT+14:00 • UTC • UTC-12:00 - UTC-01:00 • UTC+01:00 - UTC+14:00 *Om du använder en DST-förskjutning (d.v.s. att du anger ett **ignore_dst**-argumentvärde som utvärderas till False) måste du ange ett ställe snarare än en GMT-förskjutning i **place**-argumentet. Det beror på att vid justering av sommartid krävs information om breddgrad i tillägg till informationen om längdgrad från en GMT-förskjutning. Se Använder GMT-förskjutningar i kombination med DST (page 735) för information.* *Du kan bara använda standardinställda tidsförskjutningar. Det går inte att använda en godtycklig tidsförskjutning, till exempelGMT-04:27.*
ignore_dst	Om det här argumentet utvärderas till True ignoreras DST (sommartid). Giltiga argumentvärden som utvärderas till True inkluderar `-1` och `True()`. Om det här argumentet utvärderas till False justeras tidsstämpeln för sommartid. Giltiga argumentvärden som utvärderas till False inkluderar `0` och `False()`. Om **ignore_dst**-argumentvärdet är ogiltigt utvärderar funktionen uttrycket som om **ignore_dst**-värdet utvärderas till True. Om **ignore_dst**-argumentvärdet inte har angetts utvärderar funktionen uttrycket som om **ignore_dst**-värdet utvärderas till False.

Giltiga platser och tidszoner

A-C	D-K	L-R	S-Z
Abu Dhabi	Darwin	La Paz	Samoa
Adelaide	Dhaka	Lima	Santiago
Alaska	Eastern Time (US & Canada)	Lisbon	Sapporo
Amsterdam	Edinburgh	Ljubljana	Sarajevo
Arizona	Ekaterinburg	London	Saskatchewan

A-C	D-K	L-R	S-Z
Astana	Fiji	Madrid	Seoul
Athens	Georgetown	Magadan	Singapore
Atlantic Time (Canada)	Greenland	Mazatlan	Skopje
Auckland	Greenwich Mean Time : Dublin	Melbourne	Sofia
Azores	Guadalajara	Mexico City	Solomon Is.
Baghdad	Guam	Mid-Atlantic	Sri Jayawardenepura
Baku	Hanoi	Minsk	St. Petersburg
Bangkok	Harare	Monrovia	Stockholm
Beijing	Hawaii	Monterrey	Sydney
Belgrade	Helsinki	Moscow	Taipei
Berlin	Hobart	Mountain Time (US & Canada)	Tallinn
Bern	Hong Kong	Mumbai	Tashkent
Bogota	Indiana (East)	Muscat	Tbilisi
Brasilia	International Date Line West	Nairobi	Tehran
Bratislava	Irkutsk	New Caledonia	Tokyo
Brisbane	Islamabad	New Delhi	Urumqi
Brussels	Istanbul	Newfoundland	Warsaw
Bucharest	Jakarta	Novosibirsk	Wellington
Budapest	Jerusalem	Nuku'alofa	West Central Africa
Buenos Aires	Kabul	Osaka	Vienna
Cairo	Kamchatka	Pacific Time (US & Canada)	Vilnius
Canberra	Karachi	Paris	Vladivostok
Cape Verde Is.	Kathmandu	Perth	Volgograd
Caracas	Kolkata	Port Moresby	Yakutsk
Casablanca	Krasnoyarsk	Prague	Yerevan
Central America	Kuala Lumpur	Pretoria	Zagreb

A-C	D-K	L-R	S-Z
Central Time (US & Canada)	Kuwait	Quito	-
Chennai	Kyiv	Riga	-
Chihuahua	-	Riyadh	-
Chongqing	-	Rome	-
Copenhagen	-	-	-

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`ConvertToLocalTime('2023-08-14 08:39:47','Paris')`	Returnerar '2023-08-14 10:39:47' och motsvarande interna tidsmarkör.
`ConvertToLocalTime(UTC(), 'Stockholm')`	Returnerar tiden för Stockholm, justerat för sommartid.
`ConvertToLocalTime(UTC(), 'Stockholm', -1)`	Returnerar tiden för Stockholm, utan justering för sommartid.
`ConvertToLocalTime(UTC(), 'GMT-05:00')`	Returnerar tidsangivelsen för Nordamerikas ostkust, t.ex. New York. Ingen justering görs för sommartid eftersom en GMT-förskjutning, snarare än en plats, anges.
`ConvertToLocalTime(UTC(), 'New York', -1)`	Returnerar tidsangivelsen för Nordamerikas ostkust (New York) utan justering för sommartid.
`ConvertToLocalTime(UTC(), 'New York', True())`	Returnerar tidsangivelsen för Nordamerikas ostkust (New York) utan justering för sommartid.
`ConvertToLocalTime(UTC(), 'New York', 0)`	Returnerar tidsangivelsen för Nordamerikas ostkust (New York) justerat för sommartid.
`ConvertToLocalTime(UTC(), 'New York', False())`	Returnerar tidsangivelsen för Nordamerikas ostkust (New York) justerat för sommartid.

Använder GMT-förskjutningar i kombination med DST

Enligt implementeringen av biblioteken för International Components for Unicode (ICU) i Qlik Sense krävs ytterligare information om breddgrad när förskjutningar av GMT (Greenwich Mean Time) i kombination med DST (Daylight Saving Time, sommartid) används.

GMT är en förskjutning av längdgraden (öst-västlig), medan DST är en förskjutning av breddgraden (nord-sydlig). Helsingfors (Finland) och Johannesburg (Sydafrika) har till exempel samma förskjutning GMT+02:00, men de har inte samma DST-förskjutning. Det innebär att förutom GMT-förskjutningen krävs för DST-förskjutningar information om breddgraden för den lokala tidszonen (inmatad geografisk tidszon) för att få fullständig information om lokala DST-förhållanden.

day

Denna funktion returnerar ett heltal som motsvarar dagen om decimaldelen av **expression** tolkas som datum enligt standardtolkningen av tal.

Funktionen returnerar dagen i månaden för ett visst datum. Den används vanligtvis för att härleda ett dagfält som en del av en kalenderdimension.

Syntax:

```
day(expression)
```

Returnerad datatyp: heltal

Exempel på funktioner

Exempel	Resultat
`day( 1971-10-12 )`	returnerar 12
`day( 35648 )`	returnerar 6, eftersom 35648 = 1997-08-06

Exempel 1 – DateFormat datauppsättning (skript)

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- En datauppsättning av datum benämnd `Master_Calendar`. Systemvariabeln `DateFormat` är inställd på DD/MM/ÅÅÅÅ.
- En föregående laddning som skapar ett ytterligare fält benämnt `day_of_month`, med användning av `day()`-funktionen.
- Ett ytterligare fält benämnt `long_date`, som använder `date()`-funktionen för att uttrycka månadens hela namn.

Laddningsskript

```
SET DateFormat='DD/MM/YYYY';

Master_Calendar:
Load
```

```
        date,
        date(date,'dd-MMMM-YYYY') as long_date,
        day(date) as day_of_month
Inline
[
date
03/11/2022
03/12/2022
03/13/2022
03/14/2022
03/15/2022
03/16/2022
03/17/2022
03/18/2022
03/19/2022
03/20/2022
03/21/2022
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `long_date`
- `day_of_month`

Resultattabell

datum	långt_datum	dag_i_månaden
03/11/2022	11-mars- 2022	11
03/12/2022	12-mars- 2022	12
03/13/2022	13-mars- 2022	13
03/14/2022	14-mars- 2022	14
03/15/2022	15-mars- 2022	15
03/16/2022	16-mars- 2022	16
03/17/2022	17-mars- 2022	17
2022-03-18	18-mars- 2022	18
2022-03-19	19-mars- 2022	19
03/20/2022	20-mars- 2022	20
03/21/2022	21-mars- 2022	21

Dag i månaden är korrekt utvärderad av `day()`-funktionen i skriptet.

Exempel 2 – ANSI-datum (skript)

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- En datauppsättning av datum benämnd `Master_Calendar`. DateFormat-systemvariabeln DD/MM/ÅÅÅÅ används. Emellertid är de datum som är inkluderade i datauppsättningen i ANSI:s standarddatumformat.
- En föregående laddning som skapar ett ytterligare fält benämnt `day_of_month`, med användning av `date()`-funktionen.
- Ett ytterligare fält benämnt `long_date` som använder `date()`-funktionen för att uttrycka datumet med månadens hela namn.

Laddningsskript

```
SET DateFormat='DD/MM/YYYY';
Master_Calendar:
Load
       date,
       date(date,'dd-MMMM-YYYY') as long_date,
       day(date) as day_of_month

Inline
[
date
2022-03-11
2022-03-12
2022-03-13
2022-03-14
2022-03-15
2022-03-16
2022-03-17
2022-03-18
2022-03-19
2022-03-20
2022-03-21
 ];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `long_date`
- `day_of_month`

Resultattabell

datum	långt_datum	dag_i_månaden
03/11/2022	11-mars- 2022	11
03/12/2022	12-mars- 2022	12
03/13/2022	13-mars- 2022	13
03/14/2022	14-mars- 2022	14
03/15/2022	15-mars- 2022	15
03/16/2022	16-mars- 2022	16
03/17/2022	17-mars- 2022	17
2022-03-18	18-mars- 2022	18
2022-03-19	19-mars- 2022	19
03/20/2022	20-mars- 2022	20
03/21/2022	21-mars- 2022	21

Dag i månaden är korrekt utvärderad av `day()`-funktionen i skriptet.

Exempel 3 – Oformaterade datum (skript)

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- En datauppsättning av datum benämnd `Master_Calendar`. `DateFormat`-systemvariabeln DD/MM/ÅÅÅÅ används.
- En föregående laddning som skapar ett ytterligare fält benämnt `day_of_month`, med användning av `day()`-funktionen.
- Det ursprungliga oformaterade datumet, benämnt `unformatted_date`.
- Ett extrafält, benämnt `long_date`, som använder `date()` används för att konvertera det numeriska datumet till ett formaterat datafält.

Laddningsskript

```
SET DateFormat='DD/MM/YYYY';

Master_Calendar:
Load
      unformatted_date,
      date(unformatted_date,'dd-MMMM-YYYY') as long_date,
      day(date) as day_of_month
```

```
Inline
[
unformatted_date
44868
44898
44928
44958
44988
45018
45048
45078
45008
45038
45068
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `unformatted_date`
- `long_date`
- `day_of_month`

Resultattabell

oformaterat_datum	långt_datum	dag_i_månaden
44868	03-november- 2022	3
44898	03-december- 2022	3
44928	02-januari- 2023	2
44958	01-februari- 2023	1
44988	03-mars- 2023	3
45008	23-mars- 2023	23
45018	2 april 2023	2
45038	22 april 2023	22
45048	02-maj- 2023	2
45068	22-maj- 2023	22
45078	01-juni- 2023	1

Dag i månaden är korrekt utvärderad av `day()`-funktionen i skriptet.

Exempel 4 – Beräkna utgångsmånad (diagram)

Laddningsskript och diagramuttryck

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- En datauppsättning av ordrar lagda i mars, benämnda `Orders`. Tabellen innehåller tre fält:
  - id
  - orderdatum
  - belopp

Laddningsskript

```
Orders:
Load
        id,
        order_date,
        amount
Inline
[
id,order_date,amount
1,03/01/2022,231.24
2,03/02/2022,567.28
3,03/03/2022,364.28
4,03/04/2022,575.76
5,03/05/2022,638.68
6,03/06/2022,785.38
7,03/07/2022,967.46
8,03/08/2022,287.67
9,03/09/2022,764.45
10,03/10/2022,875.43
11,03/11/2022,957.35
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:`order_date`.

För att beräkna leveransdatumet skapar du detta mått: `=day(order_date+5)`.

Resultattabell

orderdatum	=day(orderdatum+5)
03/11/2022	16

orderdatum	=day(orderdatum+5)
03/12/2022	17
03/13/2022	18
03/14/2022	19
03/15/2022	20
03/16/2022	21
03/17/2022	22
2022-03-18	23
2022-03-19	24
03/20/2022	25
03/21/2022	26

Funktionen `day()` avgör korrekt att en order som lagts den 11 mars ska levereras den 16 baserat på en 5 dagars leveranstid.

dayend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden av den dag som ingår i **time**. Det förvalda utdataformatet blir det **TimestampFormat** som har definierats i skriptet.

Syntax:

```
DayEnd(time[, [period_no[, day_start]])
```

Användning

`dayend()`-funktionen används vanligtvis som en del av ett uttryck när användaren vill att beräkningen ska använda den del av dagen som ännu inte har inträffat. Till exempel för att beräkna de totala utgifter som fortfarande återstår under dagen.

Returnerad datatyp: dual

Argument

Argument	Beskrivning
time	Tidsmarkören som ska evalueras.
period_no	**period_no** är ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den dag som innehåller **time**. Negativa värden i **period_no** anger föregående dagar och positiva värden anger efterföljande dagar.

Argument	Beskrivning
day_start	Om du vill ange att dagar inte startar vid midnatt anger du en startpunkt som delar av en dag i **day_start**. Till exempel betecknar 0,125 03.00. Med andra ord, för att skapa förskjutningen ska du dividera starttiden med 24 timmar. Till exempel, för en dag som börjar klockan 07:00, använd bråket 7/24.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
dayend('01/25/2013 16:45:00')	Returnerar 01/25/2013 23:59:59. e.m.
dayend('01/25/2013 16:45:00', -1)	Returns 01/24/2013 23:59:59. e.m.
dayend('01/25/2013 16:45:00', 0, 0.5)	Returnerar 01/26/2013 11:59:59. e.m.

Exempel 1 – basskript

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- En datauppsättning med en lista med datum läses in i en tabell med namnet "Kalender".
- `DateFormat`-standardsystemvariabel `(MM/DD/YYYY)`.
- En föregående laddning för att skapa ytterligare ett fält benämnt "`EOD_timestamp`" med hjälp av `dayend()`-funktionen.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Calendar:
    Load
```

```
      date,
      dayend(date) as EOD_timestamp
      ;
Load
date
Inline
[
date
03/11/2022 1:47:15 AM
03/12/2022 4:34:58 AM
03/13/2022 5:15:55 AM
03/14/2022 9:25:14 AM
03/15/2022 10:06:54 AM
03/16/2022 10:44:42 AM
03/17/2022 11:33:30 AM
03/18/2022 12:58:14 PM
03/19/2022 4:23:12 PM
03/20/2022 6:42:15 PM
03/21/2022 7:41:16 PM
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `EOD_timestamp`

Resultattabell

datum	EOD_timestamp
03/11/2022 1:47:15 AM	3/11/2022 23:59:59
03/12/2022 4:34:58 AM	2022-03-12 23:59:59
03/13/2022 5:15:55 AM	2022-03-13 23:59:59
03/14/2022 9:25:14 AM	2022-03-14 23:59:59
03/15/2022 10:06:54 AM	2022-03-15 23:59:59
03/16/2022 10:44:42 AM	2022-03-16 23:59:59
03/17/2022 11:33:30 AM	2022-03-17 23:59:59
03/18/2022 12:58:14 PM	2022-03-18 23:59:59
03/19/2022 4:23:12 PM	2022-03-19 23:59:59
03/20/2022 6:42:15 PM	2022-03-20 23:59:59
03/21/2022 7:41:16 PM	2022-03-21 23:59:59

Som du kan se i tabellen ovan genereras tidsmarkören för dagens slut för varje datum i vår datauppsättning. Tidsmarkören har formatet för systemvariabeln `TimestampFormat M/D/YYYY h:mm:ss [.fff] TT`.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Du laddar en datauppsättning med tjänstebokningar till en tabell med namnet "Services".

Datauppsättningen innehåller följande fält:

- `service_id`
- `service_date`
- `amount`

Du skapar två nya fält i tabellen:

- `deposit_due_date`: Datumet när depositionen ska vara inne. Det här är slutet på dagen tre dagar innan `service_date`.
- `final_payment_due_date`: Datumet när slutbetalningen ska vara inne. Det här är slutet på dagen sju dagar efter `service_date`.

De två fälten ovan skapas i en tidigare laddning med hjälp av funktionen `dayend()` och de ger de två första parametrarna, `time` och `period_no`.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Services:
   Load
     *,
     dayend(service_date,-3) as deposit_due_date,
     dayend(service_date,7) as final_payment_due_date
     ;
Load
service_id,
service_date,
amount
Inline
[
service_id, service_date,amount
1,03/11/2022 9:25:14 AM,231.24
2,03/12/2022 10:06:54 AM,567.28
3,03/13/2022 10:44:42 AM,364.28
4,03/14/2022 11:33:30 AM,575.76
5,03/15/2022 12:58:14 PM,638.68
6,03/16/2022 4:23:12 PM,785.38
7,03/17/2022 6:42:15 PM,967.46
8,03/18/2022 7:41:16 PM,287.67
9,03/19/2022 8:14:15 PM,764.45
```

```
10,03/20/2022 9:23:51 PM,875.43
11,03/21/2022 10:04:41 PM,957.35
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `service_date`
- `deposit_due_date`
- `final_payment_due_date`

Resultattabell

service_date	deposit_due_date	final_payment_due_date
03/11/2022 9:25:14 AM	2022-03-08 23:59:59	2022-03-18 23:59:59
03/12/2022 10:06:54 AM	2022-03-09 23:59:59	2022-03-19 23:59:59
03/13/2022 10:44:42 AM	2022-03-10 23:59:59	2022-03-20 23:59:59
03/14/2022 11:33:30 AM	3/11/2022 23:59:59	2022-03-21 23:59:59
03/15/2022 12:58:14 PM	2022-03-12 23:59:59	2022-03-22 23:59:59
03/16/2022 4:23:12 PM	2022-03-13 23:59:59	2022-03-23 23:59:59
03/17/2022 6:42:15 PM	2022-03-14 23:59:59	2022-03-2423:59:59
03/18/2022 7:41:16 PM	2022-03-15 23:59:59	2022-03-25 23:59:59
03/19/2022 8:14:15 PM	2022-03-16 23:59:59	2022-03-26 23:59:59
03/20/2022 9:23:51 PM	2022-03-17 23:59:59	2022-03-27 23:59:59
03/21/2022 10:04:41 PM	2022-03-18 23:59:59	3/28/2022 11:59:59 PM

Värdena på de nya fältena finns i `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Eftersom funktionen `dayend()` användes är tidsmarkören den sista millisekunden av dagen.

Värdena för depositionens förfallodag är tre dagar innan tjänstedatumet, eftersom det andra argumentet som skickades i funktionen `dayend()` är negativt.

Värdena för den slutliga betalningens förfallodag är sju dagar efter tjänstedatumet, eftersom det andra argumentet som skickades i funktionen `dayend()` är positivt.

Exempel 3 – day_start script

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Datauppsättningen och scenariot som används i det här examplet är samma som i föregående exempel.

Precis som i föregående exempel skapar du två nya fält:

- `deposit_due_date`: Datumet när depositionen ska vara inne. Det här är slutet på dagen tre dagar innan `service_date`.
- `final_payment_due_date`: Datumet när slutbetalningen ska vara inne. Det här är slutet på dagen sju dagar efter `service_date`.

Men ditt företag vill ha en policy där arbetsdagen börjar kl. 17.00 och slutar kl. 17.00 nästa dag. Ditt företag kan då övervaka transaktioner som sker under dessa arbetstider.

För att uppnå dessa krav skapas de två fälten ovan i en tidigare inläsning med hjälp av funktionen `dayend()` och genom att använda alla de tre argumenten `time`, `period_no` och `day_start`.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Services:
   Load
     *,
     dayend(service_date,-3,17/24) as deposit_due_date,
     dayend(service_date,7,17/24) as final_payment_due_date
     ;
Load
service_id,
service_date,
amount
Inline
[
service_id, service_date,amount
1,03/11/2022 9:25:14 AM,231.24
2,03/12/2022 10:06:54 AM,567.28
3,03/13/2022 10:44:42 AM,364.28
4,03/14/2022 11:33:30 AM,575.76
5,03/15/2022 12:58:14 PM,638.68
6,03/16/2022 4:23:12 PM,785.38
7,03/17/2022 6:42:15 PM,967.46
8,03/18/2022 7:41:16 PM,287.67
9,03/19/2022 8:14:15 PM,764.45
10,03/20/2022 9:23:51 PM,875.43
11,03/21/2022 10:04:41 PM,957.35
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `service_date`
- `deposit_due_date`
- `final_payment_due_date`

Resultattabell

service_date	deposit_due_date	final_payment_due_date
03/11/2022 9:25:14 AM	3/8/2022 4:59:59 PM	3/18/2022 4:59:59 PM
03/12/2022 10:06:54 AM	3/9/2022 4:59:59 PM	3/19/2022 4:59:59 PM
03/13/2022 10:44:42 AM	3/10/2022 4:59:59 PM	3/20/2022 4:59:59 PM
03/14/2022 11:33:30 AM	3/11/2022 4:59:59 PM	3/21/2022 4:59:59 PM
03/15/2022 12:58:14 PM	3/12/2022 4:59:59 PM	3/22/2022 4:59:59 PM
03/16/2022 4:23:12 PM	3/13/2022 4:59:59 PM	3/23/2022 4:59:59 PM
03/17/2022 6:42:15 PM	3/14/2022 4:59:59 PM	3/24/2022 4:59:59 PM
03/18/2022 7:41:16 PM	3/15/2022 4:59:59 PM	3/25/2022 4:59:59 PM
03/19/2022 8:14:15 PM	3/16/2022 4:59:59 PM	3/26/2022 4:59:59 PM
03/20/2022 9:23:51 PM	3/17/2022 4:59:59 PM	3/27/2022 4:59:59 PM
03/21/2022 10:04:41 PM	3/18/2022 4:59:59 PM	3/28/2022 4:59:59 PM

Trots att datumen är desamma som i exempel 2, har de nu tidsmarkören för den sista millisekunden innan kl. 17.00 eftersom värdet på det tredje argumentet, `day_start`, som skickades till funktionen `dayend()` är 17/24.

Exempel 4 – diagramexempel

Laddningsskript och diagramuttryck

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Datauppsättningen och scenariot som används i det här examplet är samma som i de två föregående exemplen. Företaget vill ha en policy där arbetsdagen börjar kl. 17.00 och slutar kl. 17.00 nästa dag.

Precis som i föregående exempel skapar du två nya fält:

- `deposit_due_date`: Datumet när depositionen ska vara inne. Det här är slutet på dagen tre dagar innan `service_date`.
- `final_payment_due_date`: Datumet när slutbetalningen ska vara inne. Det här är slutet på dagen sju dagar efter `service_date`.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Services:
Load
service_id,
service_date,
```

```
amount
Inline
[
service_id, service_date,amount
1,03/11/2022 9:25:14 AM,231.24
2,03/12/2022 10:06:54 AM,567.28
3,03/13/2022 10:44:42 AM,364.28
4,03/14/2022 11:33:30 AM,575.76
5,03/15/2022 12:58:14 PM,638.68
6,03/16/2022 4:23:12 PM,785.38
7,03/17/2022 6:42:15 PM,967.46
8,03/18/2022 7:41:16 PM,287.67
9,03/19/2022 8:14:15 PM,764.45
10,03/20/2022 9:23:51 PM,875.43
11,03/21/2022 10:04:41 PM,957.35
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

`service_date`.

För att skapa fältet `deposit_due_date` skapar du det här måttet:

`=dayend(service_date,-3,17/24)`.

För att sedan skapa fältet `final_payment_due_date` skapar du det här måttet:

`=dayend(service_date,7,17/24)`.

Resultattabell

service_date	=dayend(service_date,-3,17/24)	=dayend(service_date,7,17/24
03/11/2022	3/8/2022 16:59:59 PM	3/18/2022 16:59:59 PM
03/12/2022	3/9/2022 16:59:59 PM	3/9/2022 16:59:59 PM
03/13/2022	3/10/2022 16:59:59 PM	3/20/2022 16:59:59 PM
03/14/2022	3/11/2022 16:59:59 PM	3/21/2022 16:59:59 PM
03/15/2022	3/12/2022 16:59:59 PM	3/22/2022 16:59:59 PM
03/16/2022	3/13/2022 16:59:59 PM	3/23/2022 16:59:59 PM
03/17/2022	3/14/2022 16:59:59 PM	3/24/2022 16:59:59 PM
2022-03-18	3/15/2022 16:59:59 PM	3/25/2022 16:59:59 PM
2022-03-19	3/16/2022 16:59:59 PM	3/26/2022 16:59:59 PM
03/20/2022	3/17/2022 16:59:59 PM	3/27/2022 16:59:59 PM
03/21/2022	3/18/2022 16:59:59 PM	2022-03-28 16:59:59

Värdena på de nya fältena finns i `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Eftersom funktionen `dayend()` användes är tidsmarkören den sista millisekunden av dagen.

Värdena för betalningens förfallodag är tre dagar innan tjänstedatumet, eftersom det andra argumentet som skickades i funktionen `dayend()` är negativt.

Värdena för den slutliga betalningens förfallodag är sju dagar efter tjänstedatumet, eftersom det andra argumentet som skickades i funktionen `dayend()` är positivt.

Datumen har tidsmarkören för den sista millisekunden innan kl. 17.00 eftersom värdet på det tredje argumentet, `day_start`, som skickades till funktionen `dayend()` är 17/24.

Argument

Argument	Beskrivning
time	Tidsmarkören som ska evalueras.
period_no	**period_no** är ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den dag som innehåller **time**. Negativa värden i **period_no** anger föregående dagar och positiva värden anger efterföljande dagar.
day_start	Om du vill ange att dagar inte startar vid midnatt anger du en startpunkt som delar av en dag i **day_start**. Till exempel betecknar 0,125 03.00.

daylightsaving

Returnerar ett dubbelt värde som motsvarar den aktuella justeringen för sommartid, enligt systemtiden. I För klienthanterat Qlik Sense returnerar funktionen olika värden beroende på årstiden och aktuell systemtid. I Qlik Cloud använder funktionen alltid UTC-tid så det returnerade värdet är alltid ett dubbelt värde som indikerar inga tidsjusteringar.

Syntax:

```
DaylightSaving( )
```

Returnerad datatyp: dual

Exempel:

```
daylightsaving( )
```

dayname

Denna funktion returnerar ett värde som visar datumet med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden på dagen som innehåller **time**.

Syntax:

```
DayName(time[, period_no [, day_start]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
time	Tidsmarkören som ska evalueras.
period_no	**period_no** är ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den dag som innehåller **time**. Negativa värden i **period_no** anger föregående dagar och positiva värden anger efterföljande dagar.
day_start	Om du vill ange att dagar inte startar vid midnatt anger du en startpunkt som delar av en dag i **day_start**. Till exempel betecknar 0,125 03.00.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`dayname('25/01/2013 16:45:00')`	Returnerar 25/01/2013.
`dayname('25/01/2013 16:45:00', -1)`	Returnerar 24/01/2013.
`dayname('25/01/2013 16:45:00', 0, 0.5 )`	Returnerar 25/01/2013. När du visar hela tidsmarkören kan du se att den motsvarar25/01/2013 12:00:00.000.

Exempel:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

I det här exemplet skapas dagsnamnet utifrån tidsmarkören som markerar början på dagen efter varje faktureringsdatum i tabellen.

```
TempTable:

LOAD RecNo() as InvID, * Inline [

InvDate

28/03/2012

10/12/2012
```

```
5/2/2013

31/3/2013

19/5/2013

15/9/2013

11/12/2013

2/3/2014

14/5/2014

13/6/2014

7/7/2014

4/8/2014

];



InvoiceData:

LOAD *,

DayName(InvDate, 1) AS DName

Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för dayname()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	DName
28/03/2012	29/03/2012 00:00:00
10/12/2012	11/12/2012 00:00:00
5/2/2013	07/02/2013 00:00:00
31/3/2013	01/04/2013 00:00:00
19/5/2013	20/05/2013 00:00:00
15/9/2013	16/09/2013 00:00:00
11/12/2013	12/12/2013 00:00:00
2/3/2014	03/03/2014 00:00:00

InvDate	DName
14/5/2014	15/05/2014 00:00:00
13/6/2014	14/06/2014 00:00:00
7/7/2014	08/07/2014 00:00:00
4/8/2014	05/08/2014 00:00:00

daynumberofquarter

Beräknar dagens nummer i kvartalet där tidsmarkören finns. Den här funktionen används när du skapar en Master Calendar

Syntax:

```
DayNumberOfQuarter(timestamp[,start_month])
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
timestamp	Det datum eller den tidsmarkör som ska utvärderas.
start_month	Genom att ange en **start_month** mellan 2 och 12 (1 om parametern utelämnas), kan årets början flyttas fram till första dagen på valfri månad. Om du exempelvis vill börja ett budgetår den 1 mars anger du **start_month** som 3.

I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Exempel på funktioner

Exempel	Resultat
`DayNumberOfQuarter('12/09/2014')`	Returnerar 74, dagens nummer för innevarande kvartal.
`DayNumberOfQuarter ('12/09/2014',3)`	Returnerar 12, dagens nummer för innevarande kvartal. I det här fallet börjar det första kvartalet med mars (eftersom start_month har angetts som 3). Det innebär att det innevarande kvartalet är det tredje kvartalet, som började den 1 september.

Exempel 1 – januari, början på året (skript)

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- En enkel datauppsättning med en lista med datum som laddas till en tabell med namnet `Calendar`. `DateFormat`-standardsystemvariabeln MM/DD/YYYY används.
- En föregående laddning som skapar ett ytterligare fält benämnt `DayNrQtr`, med användning av `DayNumberOfQuarter()`-funktionen.

Utöver datumet ges funktionen inga fler parametrar.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
        date,
        DayNumberOfQuarter(date) as DayNrQtr
        ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
03/31/2022
04/01/2022
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `daynrqtr`

Resultattabell

datum	daynrqtr
01/01/2022	1
01/10/2022	10
01/31/2022	31
2022-02-01	32

datum	daynrqtr
2022-02-10	41
02/28/2022	59
2022-03-01	61
03/31/2022	91
04/01/2022	1

Årets första dag är 1 januari, eftersom inget annat argument har skickats till funktionen `DayNumberOfQuarter()`.

Den 1 januari är kvartalets första dag medan den 1 februari är kvartalets 32:a dag. Den 31:a mars är den 91:a och sista dagen i kvartalet medan den 1 april är det andra kvartalets första dag.

Exempel 2 – februari, början på året (skript)

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- Samma datauppsättning som i det första exemplet.
- `DateFormat`-standardsystemvariabeln MM/DD/YYYY används.
- Ett `start_month`-argument börjar den 1 februari. Därför inleds budgetåret den 1 februari.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
       date,
       DayNumberOfQuarter(date,2) as DayNrQtr
       ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
```

```
03/31/2022
04/01/2022
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `daynrqtr`

Resultattabell

datum	daynrqtr
01/01/2022	62
01/10/2022	71
01/31/2022	92
2022-02-01	1
2022-02-10	10
02/28/2022	28
2022-03-01	30
03/31/2022	60
04/01/2022	61

Årets första dag är 1 februari, eftersom det andra argumentet som skickades till funktionen `DayNumberOfQuarter()` var 2.

Årets första kvartal är mellan februari och april medan det fjärde kvartalet är mellan november och januari. Det här visas i resultattabellen där 1 februari är kvartalets första dag medan den 1 januari är kvartalets 92:a och sista dag.

Exempel 3 – januari, början på året (diagram)

Laddningsskript och diagramuttryck

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- Samma datauppsättning som i det första exemplet.
- `DateFormat`-standardsystemvariabeln MM/DD/YYYY används.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Värdet på kvartalets dag beräknas via ett mått i ett diagramobjekt.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
03/31/2022
04/01/2022
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

Skapa följande mått:

```
=daynumberofquarter(date)
```

Resultattabell

datum	=daynumberofquarter(date)
01/01/2022	1
01/10/2022	10
01/31/2022	31
2022-02-01	32
2022-02-10	41
02/28/2022	59
2022-03-01	61
03/31/2022	91
04/01/2022	1

Årets första dag är den 1 januari, eftersom inget annat argument skickades till funktionen `DayNumberOfQuarter()`.

Den 1 januari är kvartalets första dag medan den 1 februari är kvartalets 32:a dag. Den 31:a mars är den 91:a och sista dagen i kvartalet medan den 1 april är det andra kvartalets första dag.

Exempel 4 – februari, början på året (diagram)

Laddningsskript och diagramuttryck

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- Samma datauppsättning som i det första exemplet.
- `DateFormat`-standardsystemvariabeln MM/DD/YYYY används.
- Budgetåret löper från den 1 februari till den 31 januari.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Värdet på kvartalets dag beräknas via ett mått i ett diagramobjekt.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
03/31/2022
04/01/2022
];
```

Diagramobjekt

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

Skapa följande mått:

```
=daynumberofquarter(date,2)
```

Resultat

Resultattabell

datum	=daynumberofquarter(date,2)
01/01/2022	62
01/10/2022	71
01/31/2022	92
2022-02-01	1
2022-02-10	10
02/28/2022	28
2022-03-01	30
03/31/2022	60
04/01/2022	61

Årets första dag är den 1 januari, eftersom det andra argumentet som skickades till funktionen `DayNumberOfQuarter()` var 2.

Årets första kvartal är mellan februari och april medan det fjärde kvartalet är mellan november och januari. Det här syns i resultattabellen där 1 februari är kvartalets första dag medan den 1 januari är kvartalets 92:a och sista dag.

daynumberofyear

Beräknar dagens nummer på året där tidsmarkören finns. Beräkningen är gjord från den första millisekunden av den första dagen på året, men startpunkten för den första månaden kan flyttas.

Syntax:

```
DayNumberOfYear(timestamp[,start_month])
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
timestamp	Det datum eller den tidsmarkör som ska utvärderas.
start_month	Genom att ange en **start_month** mellan 2 och 12 (1 om parametern utelämnas), kan årets början flyttas fram till första dagen på valfri månad. Om du exempelvis vill börja ett budgetår den 1 mars anger du **start_month** som 3.

I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Exempel på funktioner

Exempel	Resultat
`DayNumberOfYear( '12/09/2014' )`	Returnerar 256, dagens nummer räknat från den första dagen på året.
`DayNumberOfYear( '12/09/2014',3 )`	Returnerar 196, dagens nummer räknat från 1 mars.

Exempel 1 – januari, början på året (skript)

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- En enkel datauppsättning med en lista med datum som laddas till en tabell med namnet `Calendar`. `DateFormat`-standardsystemvariabeln MM/DD/YYYY används.
- En föregående laddning som skapar ett ytterligare fält benämnt `daynryear`, med användning av `DayNumberOfYear()`-funktionen.

Utöver datumet ges funktionen inga fler parametrar.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
       date,
       DayNumberOfYear(date) as daynryear
       ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `daynryear`

Resultattabell

datum	dagnummerår
01/01/2022	1
01/10/2022	10
01/31/2022	31
2022-02-01	32
2022-02-10	41
06/30/2022	182
07/26/2022	208
10/31/2022	305
11/01/2022	306
12/31/2022	366

Årets första dag är den 1 januari, eftersom inget annat argument har skickats till funktionen `DayNumberOfYear()`.

Den 1 januari är kvartalets första dag medan den 1 februari är årets 32:a dag. Den 30 juni är den 182:a dagen medan den 31 december är den 366:e och sista dagen på året.

Exempel 2 – november, början på året (skript)

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- Samma datauppsättning som i det första exemplet.
- `DateFormat`-standardsystemvariabeln MM/DD/YYYY används
- Ett `start_month`-argument börjar den 1 november. Därför inleds budgetåret den 1 november.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
       date,
       DayNumberOfYear(date,11) as daynryear
       ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `daynryear`

Resultattabell

datum	dagnummerår
01/01/2022	62
01/10/2022	71
01/31/2022	92
2022-02-01	93
2022-02-10	102
06/30/2022	243
07/26/2022	269
10/31/2022	366
2022-11-01	1
12/31/2022	61

Årets första dag är den 1 november, eftersom det andra argumentet som skickades till funktionen `DayNumberOfYear()` var 11.

Den 1 januari är kvartalets första dag medan den 1 februari är årets 32:a dag. Den 30 juni är den 182:a dagen medan den 31 december är den 366:e och sista dagen på året.

Exempel 3 – januari, början på året (diagram)

Laddningsskript och diagramuttryck

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- Samma datauppsättning som i det första exemplet.
- `DateFormat`-standardsystemvariabeln MM/DD/YYYY används.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Värdet på kvartalets dag beräknas via ett mått i ett diagramobjekt.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

Skapa följande mått:

```
=daynumberofyear(date)
```

Resultattabell

datum	=dagnummerpåår(datum)
01/01/2022	1
01/10/2022	10
01/31/2022	31
2022-02-01	32
2022-02-10	41
06/30/2022	182
07/26/2022	208
10/31/2022	305
11/01/2022	306
12/31/2022	366

Årets första dag är den 1 januari, eftersom inget annat argument har skickats till funktionen `DayNumberOfYear()`.

Den 1 januari är årets första dag medan den 1 februari är årets 32:a dag. Den 30 juni är den 182:a dagen medan den 31 december är den 366:e och sista dagen på året.

Exempel 4 – november, början på året (diagram)

Laddningsskript och diagramuttryck

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- Samma datauppsättning som i det första exemplet.
- `DateFormat`-standardsystemvariabeln MM/DD/YYYY används.
- Budgetåret löper från den 1 november till den 31 oktober.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Värdet på dag på året beräknas via ett mått i ett diagramobjekt.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
Calendar:
Load
date
Inline
[
```

```
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

Skapa följande mått:

```
=daynumberofyear(date)
```

Resultattabell

datum	=daynumberofyear(date,11)
01/01/2022	62
01/10/2022	71
01/31/2022	92
2022-02-01	93
2022-02-10	102
06/30/2022	243
07/26/2022	269
10/31/2022	366
11/01/2022	1
12/31/2022	61

Årets första dag är den 1 november, eftersom det andra argumentet som skickades till funktionen `DayNumberOfYear()` var 11.

Budgetåret är mellan november och oktober. Det här visas i resultattabellen där 1 november är årets första dag medan den 1 oktober är årets 366:e och sista dag.

daystart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör med den första millisekunden av den dag som ingår i argumentet **time**. Det förvalda utdataformatet blir det **TimestampFormat** som har definierats i skriptet.

Syntax:

```
DayStart(time[, [period_no[, day_start]])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
time	Tidsmarkören som ska evalueras.
period_no	**period_no** är ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den dag som innehåller **time**. Negativa värden i **period_no** anger föregående dagar och positiva värden anger efterföljande dagar.
day_start	Om du vill ange att dagar inte startar vid midnatt anger du en startpunkt som delar av en dag i **day_start**. Till exempel betecknar 0,125 03.00. Med andra ord, för att skapa förskjutningen ska du dividera starttiden med 24 timmar. Till exempel, för en dag som börjar klockan 07:00, använd bråket 7/24.

Användning

`daystart()`-funktionen används vanligtvis som en del av ett uttryck när användaren vill att beräkningen ska använda den del av dagen som har förflutit hittills. Det kan exempelvis användas för att beräkna de totala lönerna som de anställda har tjänat hittills under dagen.

I de här exemplen används tidsmarkörsformatet `'M/D/YYYY h:mm:ss[.fff] TT'`. Tidsmarkörsformatet anges i `SET TimeStamp`-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Exempel på funktioner

Exempel	Resultat
`daystart('01/25/2013 4:45:00 PM')`	Returnerar 1/25/2013 12:00:00 AM.
`daystart('1/25/2013 4:45:00 PM', -1)`	Returnerar 1/24/2013 12:00:00 AM.
`daystart('1/25/2013 16:45:00',0,0.5 )`	Returnerar 1/25/2013 12:00:00 PM.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så

kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – enkelt exempel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En enkel datauppsättning med en lista med datum som laddas till en tabell med namnet `Calendar`.
- `TimeStampFormat`-standardsystemvariabeln (`(M/D/YYYY h:mm:ss[.fff] TT`) används.
- En föregående laddning som skapar ett ytterligare fält benämnt `SOD_timestamp`, med användning av `daystart()`-funktionen.

Utöver datumet ges funktionen inga fler parametrar.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Calendar:
    Load
        date,
        daystart(date) as SOD_timestamp
        ;
Load
date
Inline
[
date
03/11/2022 1:47:15 AM
03/12/2022 4:34:58 AM
03/13/2022 5:15:55 AM
03/14/2022 9:25:14 AM
03/15/2022 10:06:54 AM
03/16/2022 10:44:42 AM
03/17/2022 11:33:30 AM
03/18/2022 12:58:14 PM
03/19/2022 4:23:12 PM
03/20/2022 6:42:15 PM
03/21/2022 7:41:16 PM
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `SOD_timestamp`

Resultattabell

datum	SOD_timestamp
03/11/2022 1:47:15 AM	3/11/2022 12:00:00 AM
03/12/2022 4:34:58 AM	2022-03-12 00:00:00
03/13/2022 5:15:55 AM	3/13/2022 12:00:00 AM
03/14/2022 9:25:14 AM	3/14/2022 12:00:00 AM
03/15/2022 10:06:54 AM	3/15/2022 12:00:00 AM
03/16/2022 10:44:42 AM	2022-03-16 00:00:00
03/17/2022 11:33:30 AM	3/17/2022 12:00:00 AM
03/18/2022 12:58:14 PM	3/18/2022 12:00:00 AM
03/19/2022 4:23:12 PM	2022-03-19 00:00:00
03/20/2022 6:42:15 PM	3/20/2022 12:00:00 AM
03/21/2022 7:41:16 PM	3/21/2022 12:00:00 AM

Som framgår av tabellen ovan genereras tidsmarkören för dagens slut för varje datum i vår datauppsättning. Tidsmarkören har formatet för systemvariabeln `TimestampFormat M/D/YYYY h:mm:ss [.fff] TT`.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med en lista med parkeringsböter som laddas till en tabell med namnet `Fines`. Datauppsättningen innehåller följande fält:
  - `id`
  - `due_date`
  - `number_plate`
  - `amount`
- En föregående laddning som använder `daystart()`-funktionen och som tillhandahåller alla tre parametrarna: `time`, `period_no` och `day_start`. Den föregående laddningen skapar de två följande nya datumfälten:

- Ett `early_repayment_period`-datumfält som börjar sju dagar innan inbetalningen ska göras.
- Ett `late_penalty_period`-datumfält som börjar 14 dagar efter inbetalningen ska göras.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Fines:
    Load
        *,
        daystart(due_date,-7) as early_repayment_period,
        daystart(due_date,14) as late_penalty_period
        ;
Load
*
Inline
[
id, due_date, number_plate,amount
1,02/11/2022, 573RJG,50.00
2,03/25/2022, SC41854,50.00
3,04/14/2022, 8EHZ378,50.00
4,06/28/2022, 8HSS198,50.00
5,08/15/2022, 1221665,50.00
6,11/16/2022, EAK473,50.00
7,01/17/2023, KD6822,50.00
8,03/22/2023, 1GGLB,50.00
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `due_date`
- `early_repayment_period`
- `late_penalty_period`

Resultattabell

due_date	early_repayment_period	late_penalty_period
02/11/2022 9:25:14 AM	2/4/2022 12:00:00 AM	2/25/2022 12:00:00 AM
03/25/2022 10:06:54 AM	3/18/2022 12:00:00 AM	4/8/2022 12:00:00 AM
04/14/2022 10:44:42 AM	4/7/2022 12:00:00 AM	4/28/2022 12:00:00 AM
06/28/2022 11:33:30 AM	6/21/2022 12:00:00 AM	7/12/2022 12:00:00 AM
08/15/2022 12:58:14 PM	8/8/2022 12:00:00 AM	8/29/2022 12:00:00 AM
11/16/2022 4:23:12 PM	11/9/2022 12:00:00 AM	11/30/2022 12:00:00 AM
01/17/2023 6:42:15 PM	1/10/2023 12:00:00 AM	1/31/2023 12:00:00 AM
03/22/2023 7:41:16 PM	3/15/2023 12:00:00 AM	4/5/2023 12:00:00 AM

Värdena på de nya fältena finns i `TimestampFormat M/DD/YYYY tt`. Eftersom `daystart()`-funktionen användes är tidsmarkörerna den första millisekunden av dagen.

Värdena för förtidsåterbetalningsperioden är sju dagar före datumet då inbetalningen ska göras, eftersom det andra argumentet som skickats i `daystart()`-funktionen är negativt.

Värdena för den försenade återbetalningsperioden är 14 dagar efter datumet då inbetalningen ska göras, eftersom det andra argumentet som skickats i `daystart()`-funktionen är negativt.

Exempel 3 – day_start

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i föregående exempel.
- Samma föregående laddning som i föregående exempel.

I det här exemplet ställer vi in arbetsdagen till att börja och sluta 7.00 varje dag.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Fines:
    Load
        *,
        daystart(due_date,-7,7/24) as early_repayment_period,
        daystart(due_date,14, 7/24) as late_penalty_period
        ;
Load
*
Inline
[
id, due_date, number_plate,amount
1,02/11/2022, 573RJG,50.00
2,03/25/2022, SC41854,50.00
3,04/14/2022, 8EHZ378,50.00
4,06/28/2022, 8HSS198,50.00
5,08/15/2022, 1221665,50.00
6,11/16/2022, EAK473,50.00
7,01/17/2023, KD6822,50.00
8,03/22/2023, 1GGLB,50.00
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- due_date
- early_repayment_period
- late_penalty_period

Resultattabell

due_date	early_repayment_period	late_penalty_period
2022-02-11	2/3/2022 7:00:00 AM	2/24/2022 7:00:00 AM
03/25/2022	3/17/2022 7:00:00 AM	4/7/2022 7:00:00 AM
04/14/2022	4/6/2022 7:00:00 AM	4/27/2022 7:00:00 AM
06/28/2022	6/20/2022 7:00:00 AM	7/11/2022 7:00:00 AM
08/15/2022	8/7/2022 7:00:00 AM	8/28/2022 7:00:00 AM
11/16/2022	11/8/2022 7:00:00 AM	11/29/2022 7:00:00 AM
01/17/2023	1/9/2023 7:00:00 AM	1/30/2023 7:00:00 AM
03/22/2023	3/14/2023 7:00:00 AM	4/4/2023 7:00:00 AM

Datumen har nu tidsmarkören 7.00 eftersom värdet på `day_start`-argumentet som skickades till funktionen `daystart()` var 7/24. Detta ställer in början av dagen till 7.00.

Eftersom `due_date`-fältet inte har en tidsstämpel behandlas det som 0.00 som därför fortfarande räknas till den föregående dagen, eftersom dagarna börjar och slutar 7.00. Förtidsåterbetalningsperioden för en bot som förfaller 11 februari börjar därför 3 februari 7.00.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

I det här exemplet används samma datauppsättning och scenario som i föregående exempel.

Men det är bara den ursprungliga `Fines`-tabellen som laddas i programmet, med de två ytterligare förfallodatumen som beräknas i ett diagramobjekt.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Fines:
    Load
*
Inline
[
id, due_date, numer_plate,amount
```

```
1,02/11/2022 9:25:14 AM, 573RJG,50.00
2,03/25/2022 10:06:54 AM, SC41854,50.00
3,04/14/2022 10:44:42 AM, 8EHZ378,50.00
4,06/28/2022 11:33:30 AM, 8HSS198,50.00
5,08/15/2022 12:58:14 PM, 1221665,50.00
6,11/16/2022 4:23:12 PM, EAK473,50.00
7,01/17/2023 6:42:15 PM, KD6822,50.00
8,03/22/2023 7:41:16 PM, 1GGLB,50.00
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `due_date`.
2. För att skapa fältet `early_repayment_period` skapar du följande mått: `=daystart(due_date,-7,7/24)`
3. För att skapa fältet `late_penalty_period` skapar du följande mått: =daystart(due_date,14,7/24)

Resultattabell

due_date	=daystart(due_date,-7,7/24)	=daystart(due_date,14,7/24)
02/11/2022 9:25:14 AM	2/4/2022 7:00:00 AM	2/25/2022 7:00:00 AM
03/25/2022 10:06:54 AM	3/18/2022 7:00:00 AM	4/8/2022 7:00:00 AM
04/14/2022 10:44:42 AM	4/7/2022 7:00:00 AM	4/28/2022 7:00:00 AM
06/28/2022 11:33:30 AM	6/21/2022 7:00:00 AM	7/12/2022 7:00:00 AM
08/15/2022 12:58:14 PM	8/8/2022 7:00:00 AM	8/29/2022 7:00:00 AM
11/16/2022 4:23:12 PM	11/9/2022 7:00:00 AM	11/30/2022 7:00:00 AM
01/17/2023 6:42:15 PM	1/10/2023 7:00:00 AM	1/31/2023 7:00:00 AM
03/22/2023 7:41:16 PM	3/15/2023 7:00:00 AM	4/5/2023 7:00:00 AM

Värdena på de nya fältena finns i `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Eftersom `daystart()`-funktionen användes motsvarar tidsmarkörerna den första millisekunden av dagen.

Värdena för förtidsåterbetalningsperioden är sju dagar före datumet då inbetalningen ska göras, eftersom det andra argumentet som skickades i `daystart()`-funktionen var negativt.

Värdena för den försenade återbetalningsperioden är 14 dagar efter datumet då inbetalningen ska göras, eftersom det andra argumentet som skickades i `daystart()`-funktionen var positivt.

Datumen har en tidsmarkör på 7.00 eftersom värdet på det tredje argumentet som skickades till `daystart()`-funktionen, `day_start` var 7/24.

firstworkdate

Funktionen **firstworkdate** returnerar senast möjliga startdatum för att uppnå **no_of_workdays** (måndag-fredag) som tar slut senast **end_date** med hänsyn till alla eventuella helgdagar. **end_date** och **holiday** ska vara giltiga datum eller tidsmarkörer.

Syntax:

```
firstworkdate(end_date, no_of_workdays {, holiday} )
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
end_date	Tidsmarkör för det slutdatum som ska utvärderas.
no_of_ workdays	Antalet arbetsdagar som ska uppnås.
holiday	Ledighetsperioder som ska undantas från arbetsdagar. En helgdag anges som ett strängkonstant datum. Du kan ange flera semesterdatum, avgränsade med kommatecken. **Exempel:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Exempel och resultat:

I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`firstworkdate ('29/12/2014', 9)`	Returnerar 17/12/2014.
`firstworkdate ('29/12/2014', 9, '25/12/2014', '26/12/2014')`	Returnerar 15/12/2014 eftersom en ledighetsperiod på två dagar räknas in.

Exempel:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
ProjectTable:

LOAD *, recno() as InvID, INLINE [

EndDate
```

```
28/03/2015

10/12/2015

5/2/2016

31/3/2016

19/5/2016

15/9/2016

] ;

NrDays:

Load *,

FirstWorkDate(EndDate,120) As StartDate

Resident ProjectTable;

Drop table ProjectTable;
```

Den resulterande tabellen visar de returnerade värdena för FirstWorkDate för varje post i tabellen.

Resultattabell

InvID	EndDate	StartDate
1	28/03/2015	13/10/2014
2	10/12/2015	26/06/2015
3	5/2/2016	24/08/2015
4	31/3/2016	16/10/2015
5	19/5/2016	04/12/2015
6	15/9/2016	01/04/2016

GMT

Denna funktion returnerar aktuell Greenwich Mean Time utifrån de regionala inställningarna. Funktionen returnerar värden i systemvariabelformatet `TimestampFormat`.

När applikationen laddas om, kommer alla laddningsskripttabeller, variabler eller diagramobjekt som använder `GMT`-funktionen att justeras till den senaste aktuella Greenwich Mean Time som hämtats från systemklockan.

Syntax:

```
GMT( )
```

Returnerad datatyp: dual

I de här exemplen används tidsmarkörsformatet `M/D/YYYY h:mm:ss[.fff] TT`. Datumformatet anges i `SET TimestampFormat`-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Exempel på funktioner

Exempel	Resultat
`GMT()`	3/28/2022 2:47:36 PM

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Variabel (skript)

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik. I det här exemplet ställs aktuell Greenwich Mean time som en variabel i laddningsskriptet med hjälp av `GMT`-funktionen.

Laddningsskript

```
LET vGMT = GMT();
```

Resultat

Ladda data och skapa ett ark. Skapa en textruta genom att använda diagramobjektet **Text och bild**.

Lägg till det här måttet i textrutan:

```
=vGMT
```

Textrutan ska innehålla en textrad med datum och tid som ser ut som den som visas nedan:

```
3/28/2022 2:47:36 PM
```

Exempel 2 – november, början på året (skript)

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med en lista med försenade biblioteksböcker som laddas i en tabell med namnet `Overdue`. `DateFormat`-standardsystemvariabeln MM/DD/YYYY används.
- Skapa ett nytt fält benämnt `days_overdue`, som beräknar hur många dagar varje bok är försenad.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Overdue:
    Load
        *,
        Floor(GMT()-due_date) as days_overdue
        ;
Load
*
Inline
[
cust_id,book_id,due_date
1,4,01/01/2021,
2,24,01/10/2021,
6,173,01/31/2021,
31,281,02/01/2021,
86,265,02/10/2021,
52,465,06/30/2021,
26,537,07/26/2021,
92,275,10/31/2021,
27,455,11/01/2021,
27,46,12/31/2021
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `due_date`
- `book_id`
- `days_overdue`

Resultattabell

due_date	book_id	days_overdue
2021-01-01	4	455
01/10/2021	24	446
01/31/2021	173	425
02/01/2021	281	424
02/10/2021	265	415
06/30/2021	465	275
07/26/2021	537	249
10/31/2021	275	152
11/01/2021	455	151
12/31/2021	46	91

Värdena i `days_overdue`-fältet beräknas genom att finna skillnaden mellan aktuell Greenwich Mean Time, med hjälp av `GMT()`-funktionen, och det ursprungliga förfallodatumet. För att enbart dagarna ska beräknas avrundas resultaten till närmaste heltal med `Floor()`-funktionen.

Exempel 3 – diagramobjekt (diagram)

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik. Laddningsskriptet innehåller samma datauppsättning som i föregående exempel. `DateFormat`-standardsystemvariabeln MM/DD/YYYY används.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Värdet för antalet dagars försening beräknas via ett mått i ett diagramobjekt.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Overdue:
Load
*
Inline
[
cust_id,book_id,due_date
1,4,01/01/2021,
2,24,01/10/2021,
6,173,01/31/2021,
31,281,02/01/2021,
```

```
86,265,02/10/2021,
52,465,06/30/2021,
26,537,07/26/2021,
92,275,10/31/2021,
27,455,11/01/2021,
27,46,12/31/2021
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- due_date
- book_id

Skapa följande mått:

```
=Floor(GMT() - due_date)
```

Resultattabell

due_date	book_id	=Floor(GMT()-due_date)
2021-01-01	4	455
01/10/2021	24	446
01/31/2021	173	425
02/01/2021	281	424
02/10/2021	265	415
06/30/2021	465	275
07/26/2021	537	249
10/31/2021	275	152
11/01/2021	455	151
12/31/2021	46	91

Värdena i `days_overdue`-fältet beräknas genom att finna skillnaden mellan aktuell Greenwich Mean Time, med hjälp av `GMT()`-funktionen, och det ursprungliga förfallodatumet. För att enbart dagarna ska beräknas avrundas resultaten till närmaste heltal med `Floor()`-funktionen.

hour

Denna funktion returnerar ett heltal som motsvarar timmen om decimaldelen av **expression** tolkas som en tidpunkt enligt standardtolkningen av tal.

Syntax:

```
hour(expression)
```

Returnerad datatyp: heltal

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`hour( '09:14:36' )`	Den tillhandahållna textsträngen konverteras implicit till en tidsmarkör eftersom den matchar tidsmarkörformatet som definieras i variabeln TimestampFormat. Uttrycket returnerar 9.
`hour( '0.5555' )`	Uttrycket returnerar 13 (eftersom 0.5555 = 13:19:55)

Exempel 1 – Variabel (skript)

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- En datauppsättning innehåller transaktioner enligt tidsmarkör.
- `TimeStamp`-standardsystemvariabel `(M/D/YYYY h:mm:ss[.fff] TT)`.

Skapa ett fält, "`hour`", och räkna sedan ut när köpen skedde.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
  Load
    *,
    hour(date) as hour
    ;
Load
```

```
*
Inline
[
id,date,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
9499,'2022-01-03 05:40:49',73.53,
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `hour`

Resultattabell

datum	timme
2022-01-01 22:10:22	22
2022-01-02 08:35:54	8
2022-01-03 05:40:49	5
2022-01-03 14:21:53	14
2022-01-04 18:49:38	18
2022-01-04 22:58:34	22
2022-01-05 19:04:57	19
2022-01-05 19:34:46	19
2022-01-06 08:49:09	8
2022-01-06 11:29:38	11

Värdena i timfältet skapas genom att använda funktionen `hour()` och skicka datumet som ett uttryck i föregående load-sats.

Exempel 2 – Diagramobjekt (diagram)

Laddningsskript och diagramuttryck

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- Samma datauppsättning som i det första exemplet.
- `TimeStamp`-standardsystemvariabel `(M/D/YYYY h:mm:ss[.fff] TT)`.

I det här exemplet har dock datauppsättningen (oförändrad) laddats till programmet. Värdena på `hour` beräknas via ett mått i ett diagramobjekt.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
Load
*
Inline
[
id,date,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
9499,'2022-01-03 05:40:49',73.53,
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

För att beräkna "`hour`" skapar du följande mått:

```
=hour(date)
```

Resultattabell

due_date	=hour(date)
2022-01-01 22:10:22	22
2022-01-02 08:35:54	8
2022-01-03 05:40:49	5
2022-01-03 14:21:53	14
2022-01-04 18:49:38	18
2022-01-04 22:58:34	22

due_date	=hour(date)
2022-01-05 19:04:57	19
2022-01-05 19:34:46	19
2022-01-06 08:49:09	8
2022-01-06 11:29:38	11

Värdena för 'hour' skapas genom att använda funktionen `hour()` och skicka datumet som ett uttryck i ett mått för diagramobjektet.

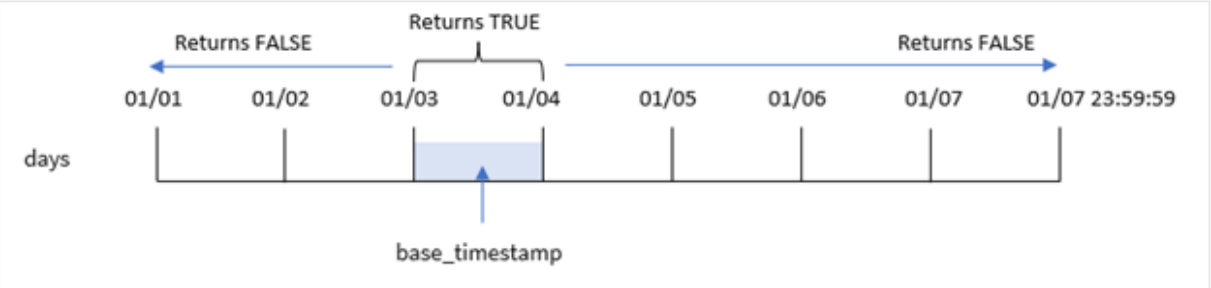
inday

Denna funktion returnerar True om **timestamp** ligger inom den dag som innehåller **base_timestamp**.

Syntax:

```
InDay (timestamp, base_timestamp, period_no[, day_start])
```

Diagram för funktionen `inday`



Funktionen `inday()` använder argumentet `base_timestamp` för att identifiera vilken dag tidsmarkören faller inom. Starttiden på dagen är som standard midnatt, men du kan ändra starttiden på dagen genom att använda argument `day_start` i funktionen `inday()`. När denna dag har definierats returnerar funktionen booleska resultat när de föreskrivna tidsmarkörsvärdena jämförs med den dagen.

Användning

Funktionen `inday()` returnerar ett booleskt resultat. Vanligtvis kommer denna typ av funktion att användas som ett villkor i en `if expression`. Detta returnerar en aggregering eller beräkning beroende på om ett utvärderat datum inträffade på dagen för den aktuella tidsmarkören.

Till exempel kan funktionen `inday()` användas för att identifiera all utrustning som tillverkas under en specifik dag.

Returnerad datatyp: Booleansk

I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.

Argument

Argument	Beskrivning
timestamp	Det datum och den tid du vill jämföra mot base_timestamp.
base_timestamp	Datum och tid som används för att utvärdera tidsmarkören.
period_no	Dagens startpunkt kan flyttas med period_no. period_no är ett heltal där värdet 0 anger den dag som innehåller base_timestamp. Negativa värden i period_no anger föregående dagar och positiva värden anger efterföljande dagar.
day_start	Om du vill använda dagar som inte startar vid midnatt kan du ange en startpunkt i delar av en dag med day_start, till exempel 0,125 för att beteckna 03.00.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i SET DateFormat-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	Returnerar True
inday ('01/12/2006 12:23:00 PM', '01/13/2006 12:00:00 AM', 0)	Returnerar False
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	Returnerar False
inday ('01/11/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	Returnerar True
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0, 0.5)	Returnerar False
inday ('01/12/2006 11:23:00 AM', '01/12/2006 12:00:00 AM', 0, 0.5)	Returnerar True

Exempel 1 – Laddningsskript (skript)

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner efter tidsmarkör som läses in i en tabell som heter `Transactions`.
- Ett datumfält som anges i `TimeStamp`-systemvariabeln i formatet `(M/D/YYYY h:mm:ss[.fff] TT)`.
- En föregående load som innehåller funktionen `inday()` som är inställd som fältet `in_day`.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        inday(date,'01/05/2022 12:00:00 AM', 0) as in_day
        ;
Load
*
Inline
[
id,date,amount
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_day`

Resultattabell

datum	inday
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	0
01/04/2022 10:58:34 PM	0

datum	inday
01/05/2022 5:40:49 AM	-1
01/05/2022 11:29:38 AM	-1
01/05/2022 7:04:57 PM	-1
01/06/2022 8:49:09 AM	0

Fältet skapades i den föregående load-satsen genom att använda funktionen och skicka datafälten, en hårdkodad tidsmarkör för den 5 januari och på 0, som funktionens argument.in_dayinday()period_no0

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Laddningsskriptet använder samma datauppsättning och scenario som det som användes i det första exemplet.

I det här exemplet är uppgiften dock att beräkna om transaktionsdatumet var två dagar innan den 5 januari.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        inday(date,'01/05/2022 12:00:00 AM', -2) as in_day
        ;
Load
*
Inline
[
id,date,amount
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- date
- in_day

Resultattabell

datum	inday
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	-1
01/04/2022 6:49:38 PM	0
01/04/2022 10:58:34 PM	0
01/05/2022 5:40:49 AM	0
01/05/2022 11:29:38 AM	0
01/05/2022 7:04:57 PM	0
01/06/2022 8:49:09 AM	0

Eftersom period_no i -2 användes som förskjutningsargument i funktionen inday(), avgör funktionen i den här instansen om en transaktion utfördes den 3 januari. Detta kan verifieras i utdatatabellen där en transaktion returnerar ett booleskt resultat på TRUE.

Exempel 3 – day_start

Laddningsskript och resultat

Översikt

Laddningsskriptet använder samma datauppsättning och scenario som de som användes i de föregående exemplen.

I det här exemplet är företagspolicyn dock att arbetsdagen börjar och slutar kl. 7 på morgonen.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        inday(date,'01/05/2022 12:00:00 AM', 0, 7/24) as in_day
        ;
Load
*
Inline
[
id,date,amount
```

```
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_day`

Resultattabell

datum	inday
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	-1
01/04/2022 10:58:34 PM	-1
01/05/2022 5:40:49 AM	-1
01/05/2022 11:29:38 AM	0
01/05/2022 7:04:57 PM	0
01/06/2022 8:49:09 AM	0

Eftersom argumentet `start_day` 7/24, dvs. 07.00, används i funktionen `inday()` avgör funktionen i den här instansen om en transaktion skedde mellan den 4 januari kl. 7 på morgonen och den 5 januari kl. 7 på morgonen.

Detta kan verifieras i utdatatabellen där transaktioner som sker efter kl. 7 på morgonen den 4 januari returnerar ett booleskt resultat, TRUE, medan transaktioner som sker efter kl. 7 på morgonen den 5 januari returnerar ett booleskt resultat FALSE.

Exempel 4 – Diagramobjekt

Laddningsskript och diagramuttryck

Översikt

Laddningsskriptet använder samma datauppsättning och scenario som de som användes i de föregående exemplen.

I det här exemplet är dock datauppsättningen oförändrad och har lästs in i programmet. Du gör beräkningen för att bestämma om en transaktion sker den 5 januari genom att skapa ett mått i ett diagramobjekt.

Laddningsskript

```
Transactions:
Load
*
Inline
[
id,date,amount
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `date`

För att räkna ut om en transaktion sker den 5 januari skapar du följande mått:

```
=inday(date,'01/05/2022 12:00:00 AM',0)
```

Resultattabell

datum	inday(date,'01/05/2022 12:00:00',0)
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	0
01/04/2022 10:58:34 PM	0

datum	inday(date,'01/05/2022 12:00:00',0)
01/05/2022 5:40:49 AM	-1
01/05/2022 11:29:38 AM	-1
01/05/2022 7:04:57 PM	-1
01/06/2022 8:49:09 AM	0

Exempel 5 – Scenario

Laddningsskript och resultat

Översikt

I det här exemplet har man identifierat att utrustningsfel var orsaken till att produkter som tillverkades den 5 januari var defekta. Slutanvändaren vill ha ett diagramobjekt som visar, efter datum, status för vilka produkter som tillverkades som var "defekta" eller "felfria", samt kostnaden för de produkter som tillverkades den 5 januari.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en tabell som heter "Produkter".
- Tabellen innehåller följande fält:
    - produkt-ID
    - tillverkningstid
    - kostnadspris

Laddningsskript

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

```
=dayname(manufacture_date)
```

Skapa följande mått:

- `=if(only(InDay(manufacture_date,makedate(2022,01,05),0)),'Defective','Faultless')`
- `=sum(cost_price)`

Ange måttens **Nummerformatering** till **Pengar**.

Under **Utseende**, stänger du av **Total**.

Resultattabell

dayname (tillverkningsdatum)	=if(only(InDay(tillverkningsdatum,makedate (2022,01,05),0)),'Defekt','Felfri')	=sum(cost_ price)
01/01/2022	Felfri	44.67
01/02/2022	Felfri	36.34
01/03/2022	Felfri	51.75
01/04/2022	Felfri	89.69
01/05/2022	Defekt	170.78
01/06/2022	Felfri	74.23

Funktionen `inday()` returnerar ett booleskt värde vid utvärdering av tillverkningsdatumen för var och en av produkterna. För alla produkter som tillverkats den 5 januari returnerar funktionen `inday()` ett booleskt värde på TRUE och markerar produkterna som "Defekta". För alla produkter som returnerar ett värde på FALSE, och därför inte tillverkas den dagen, markerar den produkterna som "Felfria".

indaytotime

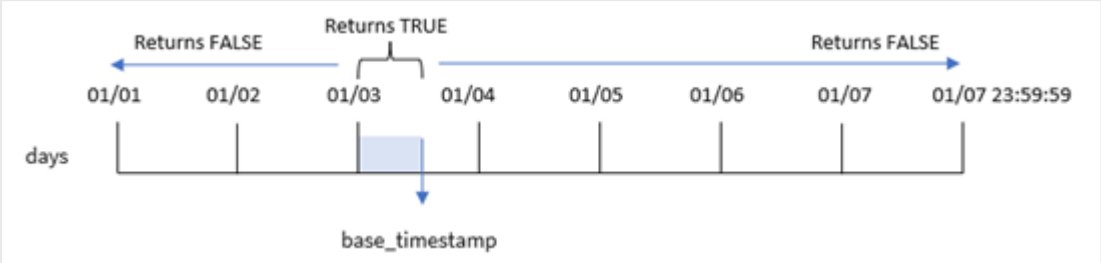
Denna funktion returnerar True om **timestamp** ligger inom den del av dagen som innehåller **base_timestamp** fram till och inklusive den exakta millisekunden för **base_timestamp**.

Syntax:

```
InDayToTime (timestamp, base_timestamp, period_no[, day_start])
```

Funktionen `indaytotime()` returnerar ett booleskt resultat beroende på när ett tidsmarkörsvärde inträffar under den dagens segment. Startgränsen för detta segment är starten på dagen, som är angiven till midnatt som standard. Dagens början kan ändras av argument `day_start` i funktionen `indaytotime()`. Slutgränsen för dagssegmentet avgörs av funktionens `base_timestamp`-argument.

Diagram för `indaytotime`-funktionen.



Användning

Funktionen `indaytotime()` returnerar ett booleskt resultat. Vanligtvis kommer denna typ av funktion att användas som ett villkor i en `if expression`. Funktionen `indaytotime()` returnerar en aggregering eller beräkning beroende på om en tidsmarkör inträffade i segmentet under denna dag fram till och med tiden för bastidsmarkören.

Till exempel kan funktionen `indaytotime()` används för att visa summan av biljettförsäljningen för föreställningar som har ägt rum hittills idag.

Returnerad datatyp: Booleansk

I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.

Argument

Argument	Beskrivning
`timestamp`	Det datum och den tid du vill jämföra mot `base_timestamp`.
`base_timestamp`	Datum och tid som används för att utvärdera tidsmarkören.
`period_no`	Dagens startpunkt kan flyttas med `period_no`. `period_no` är ett heltal där värdet 0 anger den dag som innehåller `base_timestamp`. Negativa värden i `period_no` anger föregående dagar och positiva värden anger efterföljande dagar.
`day_start`	(valfritt) Om du vill använda dagar som inte startar vid midnatt, kan du ange en förskjutning som en del av en dag i `day_start`. Du kan till exempel använda 0,125 för att hänvisa till 03.00.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`indaytotime` ('01/12/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', 0)	Returnerar True
`indaytotime` ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	Returnerar False
`indaytotime` '01/11/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', -1)	Returnerar True

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- En datauppsättning med en uppsättning transaktioner för perioden 4 till 5 januari. läses in i en tabell som heter "Transaktioner".
- Ett datumfält som anges i `TimeStamp`-systemvariabeln i formatet `(M/D/YYYY h:mm:ss[.fff] TT)`.
- En föregående load som innehåller funktionen `indaytotime()` som är inställd som fältet `'in_day_to_time'` , vilket bestämmer om var och en av transaktionerna äger rum före 09:00.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        indaytotime(date,'01/05/2022 9:00:00 AM',0) as in_day_to_time
        ;
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
8193,'01/04/2022 1:54:10 PM',45.89
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
```

```
8196,'01/04/2022 10:00:49 PM',76.11
8197,'01/05/2022 7:45:37 AM',82.06
8198,'01/05/2022 8:44:36 AM',17.17
8199,'01/05/2022 11:26:08 AM',40.39
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Resultat

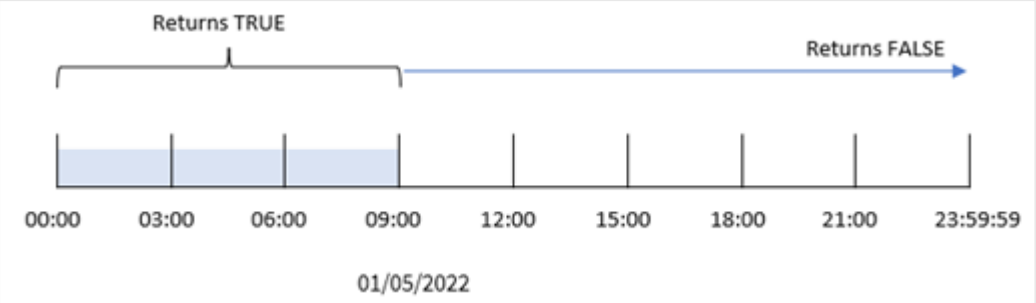
Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_day_to_time`

Resultattabell

datum	in_day_to_time
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	-1
01/05/2022 8:44:36 AM	-1
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Exempel 1-diagram för funktionen `indaytotime` med gränsen 09.00.



`in_day_to_time field` skapades i den föregående load-satsen genom att använda funktionen `indaytotime()` och skicka datumfältet, en hårdkodad tidsmarkör för kl. 9.00 den 5 januari och förskjutning på 0, som funktionens argument. Alla transaktioner som sker mellan midnatt och 09.00 den 5 januari returnerar TRUE.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Laddningsskriptet använder samma datauppsättning och scenario som det som användes i det första exemplet.

I det här exemplet kommer du dock att beräkna om transaktionsdatumet skedde en dag innan kl. 09.00 den 5 januari.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        indaytotime(date,'01/05/2022 9:00:00 AM', -1) as in_day_to_time
        ;
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
8193,'01/04/2022 1:54:10 PM',45.89
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
8196,'01/04/2022 10:00:49 PM',76.11
8197,'01/05/2022 7:45:37 AM',82.06
```

```
8198,'01/05/2022 8:44:36 AM',17.17
8199,'01/05/2022 11:26:08 AM',40.39
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Resultat

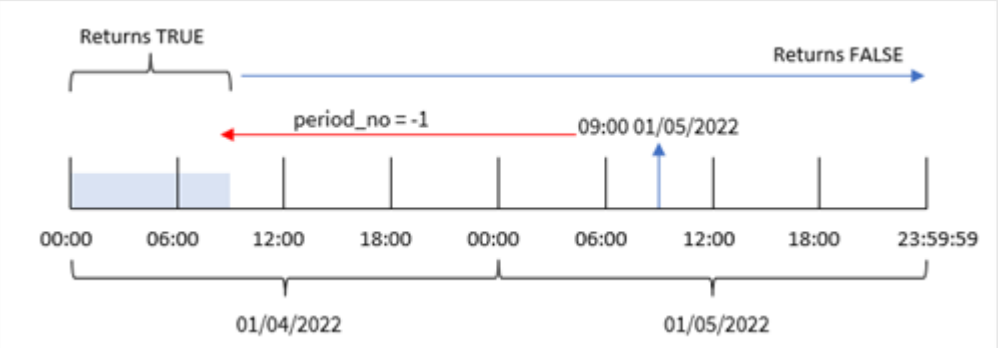
Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_day_to_time`

Resultattabell

datum	in_day_to_time
01/04/2022 3:41:54 AM	-1
01/04/2022 4:19:43 AM	-1
01/04/2022 04:53:47 AM	-1
01/04/2022 8:38:53 AM	-1
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	0
01/05/2022 8:44:36 AM	0
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Exempel 2-diagram på funktionen `indaytotime` med transaktioner från den 4 januari.



Eftersom en förskjutning på 1 användes som förskjutningsargument i funktionen `indaytotime()` avgör funktionen i det här exemplet om en transaktion skedde innan kl. 09.00 den 4 januari. Det kan verifieras i utdatatabellen där en transaktion returnerar ett booleskt resultat på TRUE.

Exempel 3 – day_start

Laddningsskript och resultat

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är företagspolicyn dock sådan att arbetsdagen börjar och slutar kl. 08.00 på morgonen.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        indaytotime(date,'01/05/2022 9:00:00 AM', 0,8/24) as in_day_to_time
        ;
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
8193,'01/04/2022 1:54:10 PM',45.89
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
8196,'01/04/2022 10:00:49 PM',76.11
8197,'01/05/2022 7:45:37 AM',82.06
8198,'01/05/2022 8:44:36 AM',17.17
8199,'01/05/2022 11:26:08 AM',40.39
```

```
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Resultat

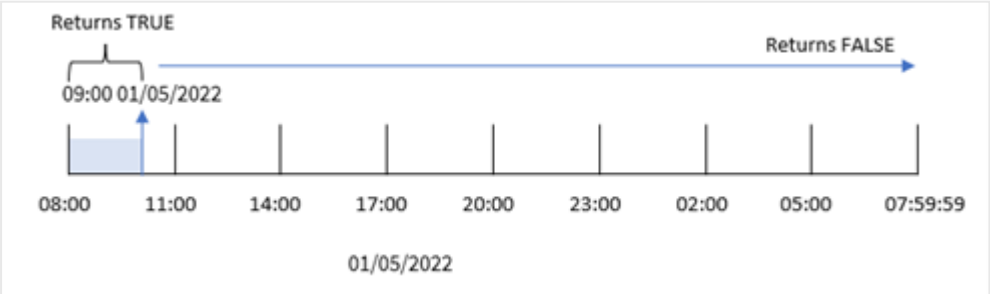
Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_day_to_time`

Resultattabell

datum	in_day_to_time
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	0
01/05/2022 8:44:36 AM	-1
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Exempel 3-diagram på funktionen `indaytotime` med transaktioner mellan kl. 08.00 och 09.00.



Eftersom argumentet `start_day` på 8/24, som är lika med 08.00, används i funktionen `indaytotime()`, börjar och slutar varje dag kl. 08.00. Därför kommer funktionen `indaytotime()` att returnera ett booleskt resultat på TRUE för alla transaktioner som har ägt rum mellan 08.00 och 09.00 den 5 januari.

Exempel 4 – Diagramobjekt

Laddningsskript och diagramuttryck

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är dock datauppsättningen oförändrad och har lästs in i programmet. Du gör beräkningen för att bestämma om en transaktion sker den 5 januari innan kl. 09.00 genom att skapa ett mått i ett diagramobjekt.

Laddningsskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
8193,'01/04/2022 1:54:10 PM',45.89
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
8196,'01/04/2022 10:00:49 PM',76.11
8197,'01/05/2022 7:45:37 AM',82.06
8198,'01/05/2022 8:44:36 AM',17.17
8199,'01/05/2022 11:26:08 AM',40.39
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

`date`.

För att bestämma om en transaktion sker den 5 januari innan kl. 09.00 skapar du följande mått:

```
=indaytotime(date,'01/05/2022 9:00:00 AM',0)
```

	Resultattabell
datum	**=inday(date,'01/05/2022 09:00:00',0)**
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	-1
01/05/2022 8:44:36 AM	-1
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Måttet `in_day_to_time` skapades i den diagramobjektet genom att använda funktionen `indaytotime()` och skicka datumfältet, en hårdkodad tidsmarkör för kl. 09.00 den 5 januari och en förskjutning på `0`, som funktionens argument. Alla transaktioner som sker mellan midnatt och 09.00 den 5 januari returnerar TRUE. Det här valideras i resultattabellen.

Exempel 5 – Scenario

Laddningsskript och resultat

Översikt

I det här exemplet läser man in en datauppsättning som innehåller biljettförsäljning för en lokal biograf i en tabell som heter Ticket_Sales. Dagens datum är den 3 maj, 2022 och klockan är 11.00.

Användaren vill ha ett KPI-diagramobjekt för att visa intäkterna från alla shower som har ägt rum hittills idag.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Ticket_Sales:
Load
*
Inline
```

```
[
sale ID, show time, ticket price
1,05/01/2022 09:30:00 AM,10.50
2,05/03/2022 05:30:00 PM,21.00
3,05/03/2022 09:30:00 AM,10.50
4,05/03/2022 09:30:00 AM,31.50
5,05/03/2022 09:30:00 AM,10.50
6,05/03/2022 12:00:00 PM,42.00
7,05/03/2022 12:00:00 PM,10.50
8,05/03/2022 05:30:00 PM,42.00
9,05/03/2022 08:00:00 PM,31.50
10,05/04/2022 10:30:00 AM,31.50
11,05/04/2022 12:00:00 PM,10.50
12,05/04/2022 05:30:00 PM,10.50
13,05/05/2022 05:30:00 PM,21.00
14,05/06/2022 12:00:00 PM,21.00
15,05/07/2022 09:30:00 AM,42.00
16,05/07/2022 10:30:00 AM,42.00
17,05/07/2022 10:30:00 AM,10.50
18,05/07/2022 05:30:00 PM,10.50
19,05/08/2022 05:30:00 PM,21.00
20,05/11/2022 09:30:00 AM,10.50
];
```

Resultat

Gör följande:

1. Skapa ett KPI-objekt.
2. Skapa ett mått som visar summan av all biljettförsäljning för föreställningar som har ägt rum idag hittills med funktionen `indaytotime()`:

```
=sum(if(indaytotime([show time],'05/03/2022 11:00:00 AM',0),[ticket price],0))
```

3. Skapa en etikett för KPI-objektet, "Aktuella intäkter".
4. Ange måttens **Nummerformatering** till **Pengar**.

Summan av biljettförsäljningen fram till kl. 11:00 den 3 maj 2022 är 52,50 USD.

Funktionen `indaytotime ()` returnerar ett booleskt värde när man jämför visningstiderna för varje biljettförsäljning med den aktuella tiden ('05/03/2022 11:00:00 AM'). För alla föreställningar den 3 maj före kl. 11:00 returnerar funktionen `indaytotime ()` ett booleskt värde på TRUE. och dess biljettpris kommer att inkluderas i summan.

inlunarweek

Denna funktion fastställer om **timestamp** ligger inom den sjudagarsperiod som innehåller **base_date**. Sjudagarsperioder i Qlik Sense definieras genom att räkna 1 januari som den första dagen i veckan. Bortsett från årets sista vecka kommer varje vecka att ha exakt sju dagar.

Syntax:

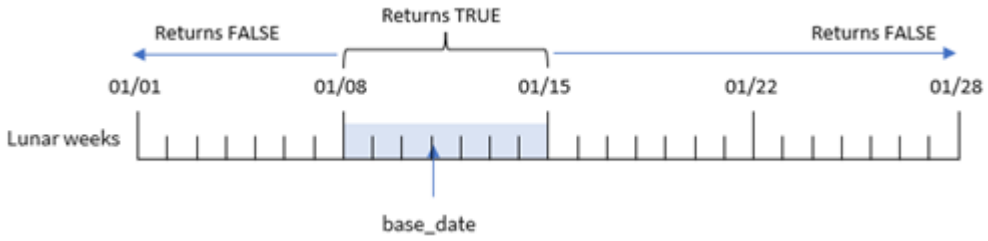
```
InLunarWeek (timestamp, base_date, period_no[, first_week_day])
```

Returnerad datatyp: Booleansk

> *I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.*

`inlunarweek()`-funktionen avgör vilken sjudagarsperiod `base_date` infaller i. Den returnerar sedan ett booleskt värde när det har avgjort huruvida varje tidsmarkör inträffar under samma sjudagarsperiod som `base_date`.

Diagram för funktionen `inlunarweek()`



Användning

Funktionen `inlunarweek()` returnerar ett booleskt resultat. Vanligtvis kommer denna typ av funktion att användas som ett villkor i ett IF-uttryck. Detta returnerar en aggregering eller beräkning beroende på om ett utvärderat datum inträffade under sjudagarsperioden i fråga.

Till exempel kan `inlunarweek()`-funktionen användas för att identifiera all utrustning som tillverkats under en specifik sjudagarsperiod.

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Data som används för att utvärdera sjudagarsperioden.
period_no	Sjudagarsperiodens startpunkt kan flyttas med **period_no**. period_no är ett heltal där värdet 0 anger den sjudagarsperiod som innehåller **base_date**. Negativa värden i **period_no** anger föregående sjudagarsperioder och positiva värden anger efterföljande sjudagarsperioder.
first_week_day	En förflyttning av startpunkten som kan vara större eller mindre än noll. Detta flyttar början på året med det angivna antalet dagar och/eller delar av en dag.

Exempel på funktioner

Exempel	Resultat
`inlunarweek ('01/12/2013', '01/14/2013', 0)`	Returnerar `TRUE`, eftersom värdet för `timestamp`, 01/12/2013, infaller under veckan 01/08/2013 till 01/14/2013.
`inlunarweek ('01/12/2013', '01/07/2013', 0)`	Returnerar `FALSE`, eftersom `base_date` 01/07/2013 är i den sjudagarsperiod som definieras som 01/01/2013 till 01/07/2013.
`inlunarweek ('01/12/2013', '01/14/2013', -1)`	Returnerar `FALSE`. Om värdet period_no anges till -1 växlar veckan till föregående vecka, 01/01/2013 till 01/07/2013.
`inlunarweek ('01/07/2013', 01/14/2013', -1)`	Returnerar `TRUE`. Jämfört med föregående exempel är `timestamp` i följande vecka, efter att ha tagit med flytten bakåt i beräkningen.
`inlunarweek ('01/11/2006', '01/08/2006', 0, 3)`	Returnerar `FALSE`. Om värdet 3 anges för `first_week_day` innebär det att startvärdet för året beräknas från 01/04/2013. Därför infaller värdet för `base_date` under den första veckan, och värdet för `timestamp` inträffar under veckan 01/11/2013 till 01/17/2013.

`inlunarweek()`-funktionen används ofta i kombination med följande funktioner:

Relaterade funktioner

Funktion	Interaktion
lunarweekname (page 973)	Den här funktionen används för att fastställa sjudagarsperiodens nummer under det år som ett inmatningsdatum inträffar.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med transaktioner för månaden januari som laddas i en tabell som heter `Transactions`.
- Datumfältet har tillhandahållits i formatet (MM/DD/ÅÅÅÅ) i Systemvariabeln `DateFormat`.

Skapa ett fält, `in_lunar_week`, som avgör huruvida transaktionerna ägde rum under samma sjudagarsperiod som 10 januari.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inlunarweek(date,'01/10/2022', 0) as in_lunar_week
        ;
Load
*
Inline
[
id,date,amount
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Resultat

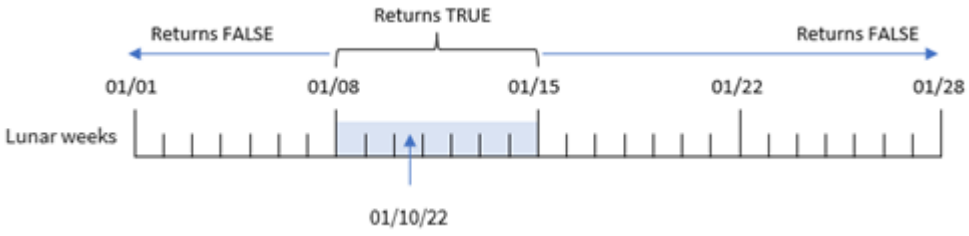
Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_lunar_week`

Resultattabell

datum	in_lunar_week
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	-1
1/9/2022	-1
1/10/2022	-1
1/11/2022	-1
1/12/2022	-1
1/13/2022	-1
1/14/2022	-1
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	0
1/23/2022	0

`inlunarweek()`-funktion, grundläggande exempel



in_lunar_week-fältet skapades i den föregående load-satsen genom att använda inlunarweek()-funktionen och sedan skicka det följande som funktionens argument:

- date-fältet
- Ett hårdkodat datum för 10 januari som base_date
- Ett period_no på 0

Eftersom sjudagarsperioderna börjar 1 januari skulle 10 januari infalla under den sjudagarsperiod som börjar 8 januari och slutar 14 januari. Därför skulle alla transaktioner som inträffar mellan dessa datum i januari returnera det booleska värdet TRUE. Det här valideras i resultattabellen.

Exempel 2 – period_no

Exempel och resultat:

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Datumfältet har tillhandahållits i formatet (MM/DD/ÅÅÅÅ) i Systemvariabeln DateFormat.

Men i det här exemplet är uppgiften att skapa ett fält, 2_lunar_weeks_later, som avgör huruvida transaktionerna ägde rum två sjudagarsperioder efter 10 januari eller inte.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inlunarweek(date,'01/10/2022', 2) as [2_lunar_weeks_later]
        ;
Load
*
Inline
[
id,date,amount
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
```

```
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Resultat

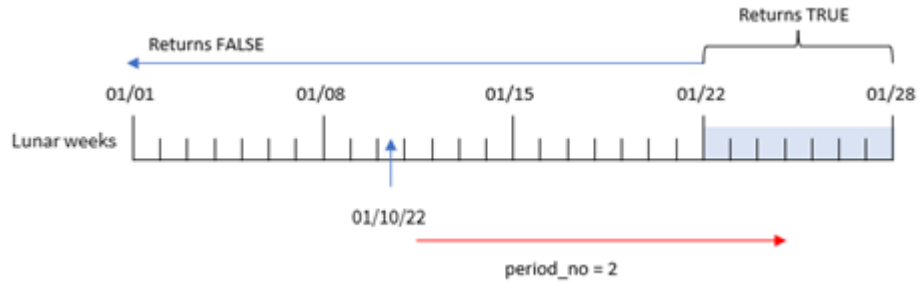
Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `2_lunar_weeks_later`

Resultattabell

datum	2_lunar_weeks_later
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	0
1/9/2022	0
1/10/2022	0
1/11/2022	0
1/12/2022	0
1/13/2022	0
1/14/2022	0
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	-1
1/23/2022	-1

Exempel på `inlunarweek()`-funktion, period_no



I det här fallet definierar funktionen veckan med början 22 januari som den sjudagarsperiod som transaktioner ska valideras mot, eftersom en `period_no` på 2 användes mot förskjutningsargumentet i `inlunarweek()`-funktionen. Därför kommer alla transaktioner som äger rum mellan 22 januari och 28 januari returnera ett booleskt resultat på `TRUE`.

Exempel 3 – first_week_day

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet använder samma datauppsättning och scenario som i det första exemplet. Men i det här exemplet ställer vi in sjudagarsperioderna till att börja 6 januari.

- Samma datauppsättning och scenario som i det första exemplet.
- `DateFormat`-standardsystemvariabeln MM/DD/YYYY används.
- Ett `first_week_day`-argument på 5. Detta ställer in sjudagarsperioderna till att börja 5 januari.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inlunarweek(date,'01/10/2022', 0,5) as in_lunar_week
        ;
Load
*
Inline
[
id,date,amount
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
```

```
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

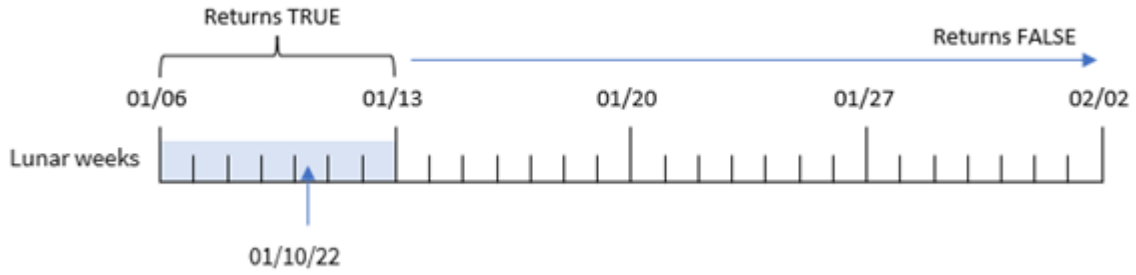
- `date`
- `in_lunar_week`

Resultattabell

datum	in_lunar_week
1/5/2022	0
1/6/2022	-1
1/7/2022	-1
1/8/2022	-1
1/9/2022	-1
1/10/2022	-1
1/11/2022	-1
1/12/2022	-1
1/13/2022	0
1/14/2022	0
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0

datum	in_lunar_week
1/22/2022	0
1/23/2022	0

Exempel på `inlunarweek()`-funktion, first_week_day



I det här fallet förskjuts början på sjudagarsperiodskalendern till 6 januari eftersom ett `first_week_date`-argument på 5 används i `inlunarweek()`-funktion. Därför infaller 10 januari i den sjudagarsperiod som börjar 6 januari och slutar 12 januari. Alla transaktioner som infaller mellan de här två datumen returnerar det booleska värdet `TRUE`.

Exempel 4 – Diagramobjekt

Laddningsskript och diagramuttryck:

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Datumfältet har tillhandahållits i formatet (MM/DD/ÅÅÅÅ) i Systemvariabeln `DateFormat`.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som avgör om transaktionerna skedde under samma sjudagarsperiod som 10 januari skapas som ett mått i ett diagramobjekt i programmet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8183,'1/5/2022',42.32
```

```
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

För att räkna ut om en transaktion sker under den sjudagarsperiod som innehåller 10 januari skapar du följande mått:

```
= inlunarweek(date,'01/10/2022', 0)
```

Resultattabell

datum	=inlunarweek(date,'01/10/2022', 0)
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	-1
1/9/2022	-1
1/10/2022	-1
1/11/2022	-1
1/12/2022	-1
1/13/2022	-1
1/14/2022	-1
1/15/2022	0
1/16/2022	0

datum	=inlunarweek(date,'01/10/2022', 0)
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	0
1/23/2022	0

Exempel 5 – Scenario

Laddningsskript och diagramuttryck:

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning läses in i en tabell som heter `Products`.
- Information bestående av produkt-ID, tillverkningsdatum och självkostnadspris.

I det här exemplet har man identifierat att utrustningsfel var orsaken till att produkter som tillverkades under den sjudagarsperiod där 12 januari ingick var defekta. Slutanvändaren vill ha ett diagramobjekt som visar, efter sjudagarsperiodens namn, status för vilka produkter som tillverkades som var "defekta" eller "felfria", samt kostnaden för de produkter som tillverkades den månaden.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
```

```
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell.
2. Skapa en dimension för att visa månadsnamnen:
   `=lunarweekname(manufacture_date)`
3. Skapa ett mått för att identifiera vilka av produkterna som är defekta och vilka som är felfria med `inlunarweek()`-funktionen:
   `=if(only(inlunarweek(manufacture_date,makedate(2022,01,12),0)), 'Defective','Faultless')`
4. Skapa ett mått för att summera `cost_price` av produkterna:
   `=sum(cost_price)`
5. Ange måttens **Nummerformatering** till **Pengar**.
6. Under **Utseende**, stänger du av **Total**.

Resultattabell

lunarweekname (manufacture_date)	=if(only(inlunarweek(manufacture_date,makedate (2022,01,12),0)), 'Defective','Faultless')	=sum(cost_ price)
2022/01	Felfri	$125.79
2022/02	Defekt	$316.38
2022/03	Felfri	$455.75
2022/04	Felfri	$146.09

Funktionen `inlunarweek()` returnerar ett booleskt värde vid utvärdering av tillverkningsdatumen för var och en av produkterna. För alla produkter som tillverkats under den sjudagarsperiod där 10 januari ingår returnerar funktionen `inlunarweek()` det booleska värdet `TRUE` och markerar produkterna som "Defekta". För alla produkter som returnerar ett värde på `FALSE` och därför inte tillverkats under den veckan, markerar den produkterna som "Felfria".

inlunarweektodate

Denna funktion tar reda på om **timestamp** ligger inom delen av sjudagarsperioden fram till och inklusive den sista millisekunden av **base_date**. Sjudagarsperioder i Qlik Sense definieras genom att räkna 1 januari som den första dagen i veckan. Bortsett från årets sista vecka kommer varje vecka att ha exakt sju dagar.

Syntax:

```
InLunarWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Returnerad datatyp: Booleansk

> *I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.*

Exempeldiagram för `inlunarweektodate()`-funktionen

Funktionen `inlunarweektodate()` fungerar som slutpunkt i sjudagarsperioden. `inlunarweek()`-funktionen å andra sidan avgör vilken sjudagarsperiod `base_date` infaller i. Om `base_date` exempelvis är 5 januari skulle alla tidsmarkörer mellan 1 januari och 5 januari returnera det booleska resultatet `TRUE`, medan datumen 6 och 7 januari och senare skulle returnera det booleska resultatet `FALSE`.

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Data som används för att utvärdera sjudagarsperioden.
period_no	Sjudagarsperiodens startpunkt kan flyttas med **period_no**. period_no är ett heltal där värdet 0 anger den sjudagarsperiod som innehåller **base_date**. Negativa värden i **period_no** anger föregående sjudagarsperioder och positiva värden anger efterföljande sjudagarsperioder.
first_week_day	En förflyttning av startpunkten som kan vara större eller mindre än noll. Detta flyttar början på året med det angivna antalet dagar och/eller delar av en dag.

Användning

Funktionen `inlunarweektodate()` returnerar ett booleskt resultat. Vanligtvis kommer denna typ av funktion att användas som ett villkor i ett IF-uttryck. `inlunarweektodate()`-funktionen används när användaren vill att beräkningen ska returnera en aggregering eller beräkning, beroende på om det beräknade datumet inträffade under ett visst segment av veckan ifråga.

Till exempel kan `inlunarweektodate()`-funktionen användas för att identifiera all utrustning som tillverkats under en viss vecka fram till och med ett visst datum.

Exempel på funktioner

Exempel	Resultat
`inlunarweektodate ('01/12/2013', '01/13/2013', 0)`	Returnerar `TRUE`, eftersom värdet för `timestamp`, 01/12/2013, infaller under delen av veckan 01/08/2013 till 01/13/2013.
`inlunarweektodate ('01/12/2013', '01/11/2013', 0)`	Returnerar `FALSE`, eftersom värdet för `timestamp` är senare än värdet `base_date`, trots att båda datumen infaller under samma sjudagarsperiod före 01/12/2012.
`inlunarweektodate ('01/12/2006', '01/05/2006', 1)`	Returnerar `TRUE`. Att ange värdet 1 för `period_no` flyttar `base_date` framåt en vecka, så att värdet för `timestamp` Infaller under en del av sjudagarsperioden.

`inlunarweektodate()`-funktionen används ofta i kombination med följande funktioner:

Relaterade funktioner

Funktion	Interaktion
lunarweekname (page 973)	Den här funktionen används för att fastställa sjudagarsperiodens nummer under det år som ett inmatningsdatum inträffar.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner för månaden januari som laddas i en tabell som heter `Transactions`. `DateFormat`-standardsystemvariabeln MM/DD/YYYY används.

- Skapa ett fält, `in_lunar_week_to_date`, som avgör huruvida transaktionerna ägde rum under sjudagarsperioden fram till 10 januari.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inlunarweektodate(date,'01/10/2022', 0) as in_lunar_week_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
8193,'1/21/2022',82.06
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_lunar_week_to_date`

Resultattabell

datum	in_lunar_week_to_date
1/1/2022	0
1/4/2022	0
1/10/2022	-1
1/11/2022	0
1/12/2022	0

datum	in_lunar_week_to_date
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	0
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

`inlunarweektodate()`-funktionen, inga ytterligare argument

Fältet `in_lunar_week_to_date` skapades i den föregående load-satsen genom att använda funktionen `inlunarweektodate()` och skicka `date`-fältet, ett hårdkodat datum för 10 januari som vår `base_date`, och en förskjutning på 0 som funktionens argument.

Eftersom sjudagarsperioderna börjar 1 januari skulle 10 januari infalla under den sjudagarsperiod som börjar 8 januari, och eftersom vi använder `inlunarweektodate()`-funktionen skulle den sjudagarsperioden sluta den 10:e. Därför skulle alla transaktioner som inträffar mellan dessa datum i januari returnera det booleska värdet `TRUE`. Det här valideras i resultattabellen.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet. Men i det här fallet är uppgiften att skapa ett fält, `2_lunar_weeks_later`, som avgör huruvida transaktionerna ägde rum två veckor efter sjudagarsperioden fram till 1 januari.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
Transactions:
    Load
        *,
        inlunarweektodate(date,'01/10/2022', 2) as [2_lunar_weeks_later]
        ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
8193,'1/21/2022',82.06
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

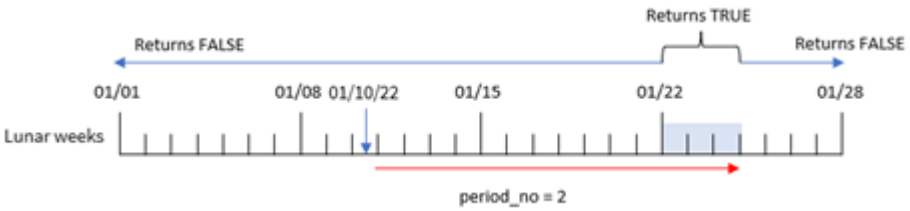
- `date`
- `2_lunar_weeks_later`

Resultattabell

datum	2_lunar_weeks_later
1/1/2022	0
1/4/2022	0
1/10/2022	0
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0

datum	2_lunar_weeks_later
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	-1
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Exempel på `inlunarweektodate()`-funktion, period_no



I det här fallet fastställer `inlunarweektodate()`-funktionen att sjudagarsperioden fram till 10 januari motsvarar tre dagar (8, 9 och 10 januari). Eftersom ett `period_no` på 2 användes som förskjutningsargument, ändras den här sjudagarsperioden med 14 dagar. Därför definieras denna tredagarsvecka så att den innehåller dagarna 22, 23 och 24 januari. Alla transaktioner som äger rum mellan 22 januari och 24 januari returnerar det booleska resultatet `TRUE`.

Exempel 3 – first_week_day

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- `DateFormat`-standardsystemvariabeln MM/DD/YYYY används.
- Ett `first_week_date`-argument på 3. Detta ställer in sjudagarsperioderna till att börja 3 januari.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
```

```
    Load
        *,
        inlunarweek(date,'01/10/2022', 0,3) as in_lunar_week_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
8193,'1/21/2022',82.06
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

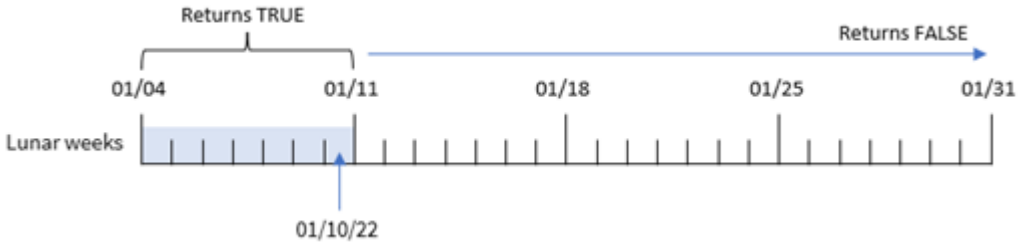
- `date`
- `in_lunar_week_to_date`

Resultattabell

datum	in_lunar_week_to_date
1/1/2022	0
1/4/2022	-1
1/10/2022	-1
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0

datum	in_lunar_week_to_date
1/23/2022	0
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Exempel på `inlunarweektodate()`-funktion, first_week_day



I det här fallet kommer den första sjudagarsperioden att vara mellan 3 januari och 10 januari, eftersom `the first_week_date`-argumentet 3 används i `inlunarweek()`-funktionen. Eftersom 10 januari även är `base_date` kommer alla transaktioner som infaller mellan de här två datumen att returnera det booleska värdet `TRUE`.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som avgör om transaktionerna skedde under sjudagarsperioden fram till 10 januari skapas som ett mått i ett diagramobjekt i programmet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
```

```
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
8193,'1/21/2022',82.06
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

Skapa följande mått:

```
=inlunarweektodate(date,'01/10/2022', 0)
```

Resultattabell

datum	=inlunarweektodate(date,'01/10/2022', 0)
1/1/2022	0
1/4/2022	0
1/10/2022	-1
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	0
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Exempel på `inlunarweektodate()`-funktionen, diagramobjekt

Måttet `in_lunar_week_to_date` skapades i diagramobjektet genom att använda `inlunarweektodate()`-funktionen och skicka datumfältet, ett hårdkodat datum för 10 januari som vår `base_date` och en förskjutning på 0 som funktionens argument.

Eftersom sjudagarsperioderna börjar 1 januari skulle 10 januari infalla under den sjudagarsperiod som börjar 8 januari. Dessutom skulle den sjudagarsperioden sluta den 10:e, eftersom vi använder `inlunarweektodate()`-funktionen. Därför skulle alla transaktioner som inträffar mellan dessa datum i januari returnera det booleska värdet `TRUE`. Det här valideras i resultattabellen.

Exempel 5 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning läses in i en tabell som heter `Products`.
- Information bestående av produkt-ID, tillverkningsdatum och självkostnadspris.

I det här exemplet har man identifierat att utrustningsfel var orsaken till att produkter som tillverkades under sjudagarsperioden den 12 januari var defekta. Frågan löstes den 13 januari. Slutanvändaren vill ha ett diagramobjekt som visar, efter vecka, status för vilka produkter som tillverkades som var "defekta" eller "felfria", samt kostnaden för de produkter som tillverkades den veckan.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';

Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'01/02/2022 12:22:06',37.23
8189,'01/05/2022 01:02:30',17.17
8190,'01/06/2022 15:36:20',88.27
8191,'01/08/2022 10:58:35',57.42
8192,'01/09/2022 08:53:32',53.80
```

```
8193,'01/10/2022 21:13:01',82.06
8194,'01/11/2022 00:57:13',40.39
8195,'01/12/2022 09:26:02',87.21
8196,'01/13/2022 15:05:09',95.93
8197,'01/14/2022 18:44:57',45.89
8198,'01/15/2022 06:10:46',36.23
8199,'01/16/2022 06:39:27',25.66
8200,'01/17/2022 10:44:16',82.77
8201,'01/18/2022 18:48:17',69.98
8202,'01/26/2022 04:36:03',76.11
8203,'01/27/2022 08:07:49',25.12
8204,'01/28/2022 12:24:29',46.23
8205,'01/30/2022 11:56:56',84.21
8206,'01/30/2022 14:40:19',96.24
8207,'01/31/2022 05:28:21',67.67
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell.
2. Skapa en dimension för att visa veckonamnen.
   `=weekname(manufacture_date)`
3. Skapa sedan en dimension som använder `inlunarweektodate()`-funktionen för att identifiera vilka av produkterna som är defekta och vilka som är felfria:
   `=if(inlunarweektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
4. Skapa ett mått för att summera `cost_price` av produkterna:
   `=sum(cost_price)`
5. Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

=lunarweekname (manufacture_date)	=if(InLunarWeekToDate(manufacture_date,makedate (2022,01,12),0),'Defective','Faultless')	=Sum(cost_ price)
2022/01	Felfri	$142.67
2022/02	Defekt	$320.88
2022/02	Felfri	$141.82
2022/03	Felfri	$214.64
2022/04	Felfri	$147.46
2022/05	Felfri	$248.12

Funktionen `inlunarweektodate()` returnerar ett booleskt värde vid utvärdering av tillverkningsdatumen för var och en av produkterna. För de som returnerar ett booleskt värde på `TRUE`, markerar det produkterna som `'Defective'`. För alla produkter som returnerar värdet `FALSE`, och därför inte har tillverkats under sjudagarsperioden fram till och med den 12 januari, markeras produkterna som `'Faultless'`.

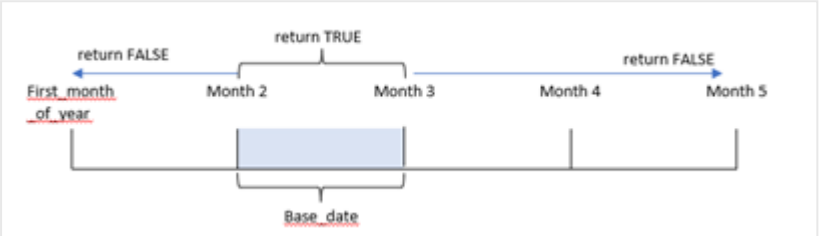
inmonth

Denna funktion returnerar True om **timestamp** ligger inom den månad som innehåller **base_date**.

Syntax:

```
InMonth (timestamp, base_date, period_no)
```

Diagram för `indaytotime`-funktionen.



Med andra ord så avgör `inmonth()`-funktionen om en uppsättning datum infaller i denna månad och returnerar ett booleskt värde baserat på en `base_date` som identifierar månaden.

Användning

Funktionen `inmonth()` returnerar ett booleskt resultat. Vanligtvis kommer denna typ av funktion att användas som ett villkor i en `if expression`. Detta returnerar en aggregering eller beräkning beroende på om ett utvärderat datum inträffade på dagen för den aktuella tidsmarkören.

Till exempel kan `inmonth()`-funktionen användas för att identifiera all utrustning som tillverkas under en specifik dag.

Returnerad datatyp: Booleansk

I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.

Argument

Argument	Beskrivning
tidsmarkör	Det datum du vill jämföra mot `base_date`.
base_date	Datum som används för att utvärdera månaden. Det är viktigt att notera att `base_date` kan vara vilken dag som helst inom en månad.
period_no	Månadens startpunkt kan flyttas med `period_no`. `period_no` är ett heltal där värdet 0 anger den månad som innehåller `base_date`. Negativa värden i `period_no` anger föregående månader och positiva värden anger efterföljande månader.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra

formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`inmonth ('25/01/2013', '01/01/2013', 0)`	Returnerar True
`inmonth('25/01/2013', '23/04/2013', 0)`	Returnerar False
`inmonth ('25/01/2013', '01/01/2013', -1)`	Returnerar False
`inmonth ('25/12/2012', '17/01/2013', -1)`	Returnerar True

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner för första halvåret 2022.
- En föregående laddning med en extra variabel, ”`in_month`”, som avgör om transaktioner ägde rum i april.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
  Load
    *,
    inmonth(date,'04/01/2022', 0) as in_month
    ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/14/2022',17.17
8190,'1/20/2022',88.27
```

```
8191,'1/22/2022',57.42
8192,'2/1/2022',53.80
8193,'2/2/2022',82.06
8194,'2/20/2022',40.39
8195,'4/11/2022',87.21
8196,'4/13/2022',95.93
8197,'4/15/2022',45.89
8198,'4/25/2022',36.23
8199,'5/20/2022',25.66
8200,'5/22/2022',82.77
8201,'6/19/2022',69.98
8202,'6/22/2022',76.11
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_month`

Exempel på funktioner

datum	in_month
1/10/2022	0
1/14/2022	0
1/20/2022	0
1/22/2022	0
2/1/2022	0
2/2/2022	0
2/20/2022	0
4/11/2022	-1
4/13/2022	-1
4/15/2022	-1
4/25/2022	-1
5/20/2022	0
22/5/2022	0
6/19/2022	0
6/22/2022	0

`in_month`-fältet skapades i den föregående load-satsen genom att använda `inmonth()`-funktionen och skicka datafälten, en hårdkodad tidsmarkör för den 1 april som funktionens argument `base_date` och en `period_no` för `0` som funktionens argument.

`base_date` identifierar månaden som kommer att returnera ett booleskt resultat av TRUE. Därför returnerar alla transaktioner som inträffade i april TRUE vilket valideras i resultattabellen.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Samma datauppsättning och scenario som det första exemplet används.

Men i det här exemplet kommer du att skapa ett fält, `2_months_prior`, som avgör om transaktionerna ägde rum två månader före april.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
   *,
   inmonth(date,'04/01/2022', -2) as [2_months_prior]
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/14/2022',17.17
8190,'1/20/2022',88.27
8191,'1/22/2022',57.42
8192,'2/1/2022',53.80
8193,'2/2/2022',82.06
8194,'2/20/2022',40.39
8195,'4/11/2022',87.21
8196,'4/13/2022',95.93
8197,'4/15/2022',45.89
8198,'4/25/2022',36.23
8199,'5/20/2022',25.66
8200,'5/22/2022',82.77
8201,'6/19/2022',69.98
8202,'6/22/2022',76.11
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `2_months_prior`

Exempel på funktioner

datum	2_months_prior
1/10/2022	0
1/14/2022	0
1/20/2022	0
1/22/2022	0
2/1/2022	-1
2/2/2022	-1
2/20/2022	-1
4/11/2022	0
4/13/2022	0
4/15/2022	0
4/25/2022	0
5/20/2022	0
22/5/2022	0
6/19/2022	0
6/22/2022	0

Om du använder -2 som `period_no`-argument i `inmonth()`-funktionen förskjuts månaden som definieras av `base_date`-argumentet två månader tidigare. I det här exemplet ändras den definierade månaden från april till februari.

Därför kommer varje transaktion som äger rum i februari att returnera det booleska värdet TRUE.

Exempel 3 – Diagramobjekt

Laddningsskript och diagramuttryck

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är dock datauppsättningen oförändrad och har lästs in i programmet. Beräkningen som avgör om transaktioner har skett mellan 1 juli och 26 juli skapas som ett mått i ett diagramobjekt i programmet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
```

```
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/14/2022',17.17
8190,'1/20/2022',88.27
8191,'1/22/2022',57.42
8192,'2/1/2022',53.80
8193,'2/2/2022',82.06
8194,'2/20/2022',40.39
8195,'4/11/2022',87.21
8196,'4/13/2022',95.93
8197,'4/15/2022',45.89
8198,'4/25/2022',36.23
8199,'5/20/2022',25.66
8200,'5/22/2022',82.77
8201,'6/19/2022',69.98
8202,'6/22/2022',76.11
];
```

Diagramobjekt

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

```
date
```

För att räkna ut om en transaktion sker den 5 januari skapar du följande mått:

```
=inmonth(date,'04/01/2022', 0)
```

Resultat

Exempel på funktioner

datum	=inmonth(date,'04/01/2022', 0)
1/10/2022	0
1/14/2022	0
1/20/2022	0
1/22/2022	0
2/1/2022	0
2/2/2022	0
2/20/2022	0
4/11/2022	-1
4/13/2022	-1
4/15/2022	-1
4/25/2022	-1

datum	=inmonth(date,'04/01/2022', 0)
5/20/2022	0
22/5/2022	0
6/19/2022	0
6/22/2022	0

Exempel 4 – Scenario

Laddningsskript och resultat

Översikt

I det här exemplet laddas en datauppsättning i en tabell som heter `Products`. Tabellen innehåller följande fält:

- Produkt-ID
- Tillverkningsdatum
- Kostnadspris

På grund av utrustningsfel var produkter som tillverkades i juli månad 2022 defekta. Problemet löstes den 27 juli 2022.

Slutanvändaren vill ha ett diagramobjekt som visar status för tillverkade produkter som var "defekta" (booleskt värde: TRUE) eller "felfria" (booleskt värde: FALSE), efter datum, samt kostnaden för de produkter som tillverkades den månaden.

Laddningsskript

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
```

```
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

```
=monthname(manufacture_date)
```

Skapa följande mått

- `=sum(cost_price)`
- `=if(only(inmonth(manufacture_date,makedate(2022,07,01),0)),'Defective','Faultless')`

1. Ange måttens **Nummerformatering** till **Pengar**.
2. Under **Utseende**, stänger du av **Total**.

Resultattabell

monthname (manufacture_date)	=if(only(inmonth(manufacture_date,makedate (2022,07,01),0)),'Defective','Faultless')	=sum(cost_ price)
Jan 2022	Felfri	$54.40
Feb 2022	Felfri	$145.69
Mar 2022	Felfri	$53.80
Apr 2022	Felfri	$82.06
Maj 2022	Felfri	$127.60
Jun 2022	Felfri	$141.82
Jul 2022	Defekt	$214.64
Aug 2022	Felfri	$147.46
Sep 2022	Felfri	$84.21
Okt 2022	Felfri	$163.91

Funktionen `inmonth()` returnerar ett booleskt värde vid utvärdering av tillverkningsdatumen för var och en av produkterna. För alla produkter som tillverkats i juli 2022 returnerar funktionen `inmonth()` ett booleskt värde på TRUE och markerar produkterna som "Defekta". För alla produkter som returnerar ett värde på FALSE, och därför inte tillverkas i juli, markerar den produkterna som "Felfria".

inmonths

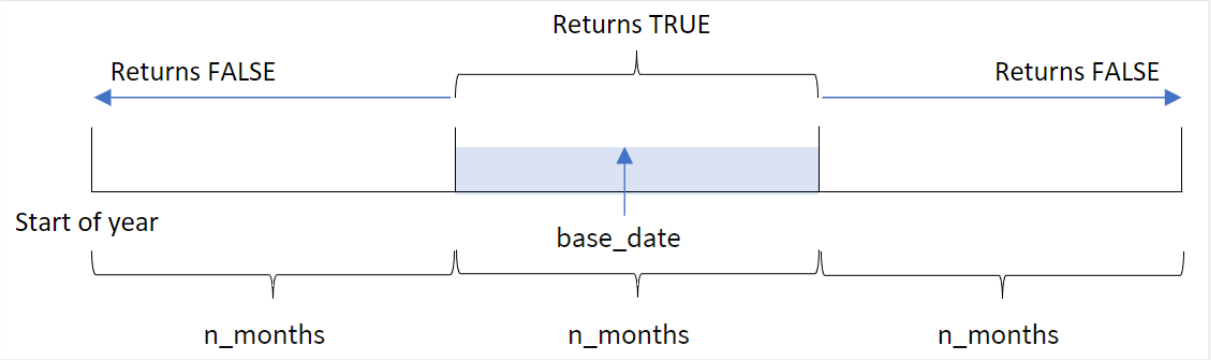
Med den här funktionen får vi reda på om en tidsmarkör faller inom samma månad, tvåmånadersperiod, fyramånadersperiod eller halvår som basdatum. Det går även att se om tidsmarkören finns inom en föregående eller senare tidsperiod.

```
InMonths(n_months, timestamp, base_date, period_no [, first_month_of_year])
```

Returnerad datatyp: Booleansk

I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.

Diagram för `inmonths()`-funktionen.



`inmonths()`-funktionen delar in året i segment baserat på det `n_months`-argument som tillhandahålls. Därefter avgör den huruvida varje tidsmarkör som utvärderas infaller i samma segment som `base_date`-argumentet. Men om ett `period_no`-argument tillhandahålls avgör funktionen huruvuda tidsmarkörerna infaller under en tidigare eller följande period från `base_date`.

Följande segment av året är tillgängliga i funktionen som `n_month`-argument.

`n_month` argument

Period	Antal månader
månad	1
två månader	2
kvartal	3
fyra månader	4
halvår	6

Användning

Funktionen `inmonths()` returnerar ett booleskt resultat. Vanligtvis kommer denna typ av funktion att användas som ett villkor i en `if expression`. Genom att använda `inmonths()`-funktionen kan du välja den period som du vill utvärdera. Exempelvis, låta användaren identifiera produkter som tillverkats under en månad, ett kvartal eller ett halvår i en viss period.

Returnerad datatyp: Booleansk

I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.

Argument

Argument	Beskrivning
n_months	Antalet månader som definierar perioden. Ett heltal eller uttryck vars resultat blir ett heltal som ska vara antingen: 1 (motsvaras av inmonth()-funktionen), 2 (tvåmånadersperiod), 3 (motsvaras av inquarter()-funktionen), 4 (fyramånadersperiod) eller 6 (halvår).
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera perioden.
period_no	Periodens startpunkt kan flyttas med **period_no**, ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den period som innehåller **base_date**. Negativa värden i **period_no** anger föregående perioder och positiva värden anger efterföljande perioder.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Du kan använda följande värden för att ställa in den första månaden på året i argumentet `first_month_of_year`:

first_month_of_year values

Månad	värde
Februari	2
mars	3
april	4
Maj	5
juni	6
juli	7
augusti	8
september	9
oktober	10
November	11
december	12

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan

vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`inmonths(4, '01/25/2013', '04/25/2013', 0)`	Returnerar TRUE Eftersom värdet för timestamp, 01/25/2013, befinner sig inom fyramånadersperioden 2013-01-01 till 2013-04-30 där värdet för `base_date`, 04/25/2013 finns.
`inmonths(4, '05/25/2013', '04/25/2013', 0)`	Returnerar FALSE Eftersom 05/25/2013 är utanför samma period som i förra exemplet.
`inmonths(4, '11/25/2012', '02/01/2013', -1 )`	Returnerar TRUE Eftersom värdet för `period_no`, `-1`, flyttar sökperioden bakåt en period på fyra månader (värdet för n-months), vilket gör sökperioden till 09/01/2012 till 12/31/2012.
`inmonths(4, '05/25/2006', '03/01/2006', 0, 3)`	Returnerar TRUE Eftersom värdet för `first_month_of_year` är angett som `3`, vilket gör sökperioden till 03/01/2006 to 07/30/2006 istället för 01/01/2006 till 04/30/2006.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner för 2022 läses in i en tabell som heter ‘`Transactions`’.
- En föregående laddning med en extra variabel, `in_months`, som avgör vilka transaktioner som ägde rum under samma kvartal som 15 maj 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
```

```
       inmonths(3,date,'05/15/2022', 0) as in_months
       ;
Load
*
Inline
[
id,date,amount
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
8190,'3/30/2022',88.27
8191,'4/5/2022',57.42
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_months`

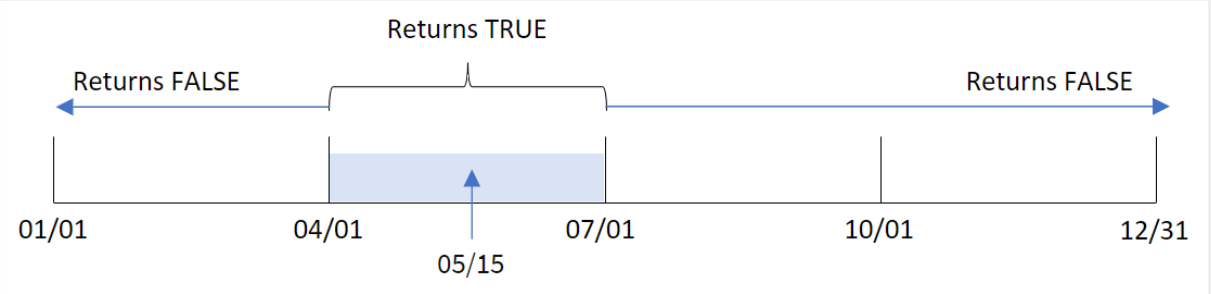
Resultattabell

Datum	in_months
2022-02-19	0
2022-03-07	0
2022-03-30	0
2022-04-05	-1
2022-04-16	-1
2022-05-01	-1
5/7/2022	-1
22/5/2022	-1

Datum	in_months
6/15/2022	-1
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

'`in_months`' fältet skapas i föregående LOAD-sats med hjälp av funktionen `inmonths()`. Det första argumentet som tillhandahålls är `3` som delar upp året i kvartalssegment. Det andra argumentet som anges identifierar vilket fält som utvärderas, datumfältet i det här exemplet. Det tredje argumentet är ett hårdkodat datum för 15 maj, som är `base_date` och ett `period_no` av `0` är det sista argumentet.

Diagrammet över `inmonths()`-funktionen med kvartalssegment.



Maj infaller under årets andra kvartal. Därför kommer alla transaktioner som sker mellan 1 april och 30 juni att returnera det booleska resultatet TRUE. Det här valideras i resultattabellen.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner för 2022 laddas i en tabell som heter 'Transactions'.
- En föregående laddning med en extra variabel, previous_quarter, som avgör huruvida transaktionerna ägde rum under kvartalet före 15 maj 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inmonths(3,date,'05/15/2022', -1) as previous_quarter
       ;
Load
*
Inline
[
id,date,amount
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
8190,'3/30/2022',88.27
8191,'4/5/2022',57.42
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

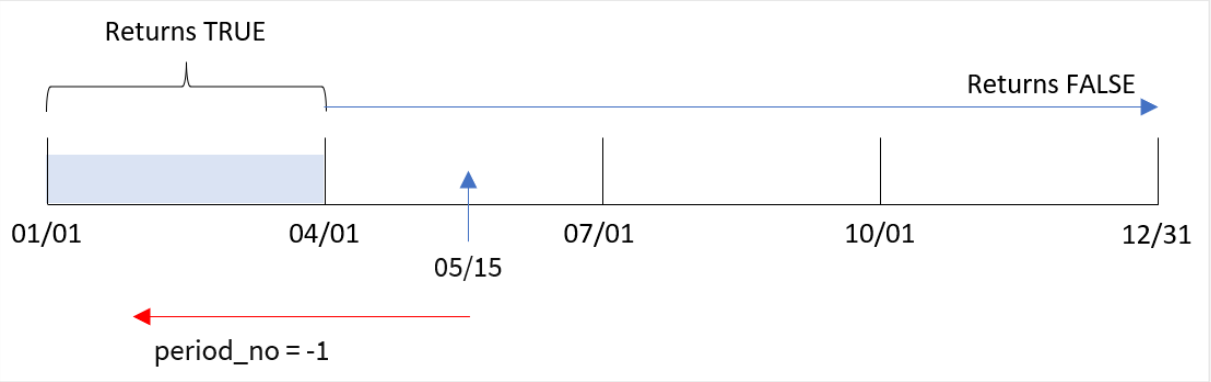
- date
- previous_quarter

Resultattabell

Datum	föregående kvartal
2022-02-19	-1
2022-03-07	-1
2022-03-30	-1
2022-04-05	0
2022-04-16	0
2022-05-01	0
5/7/2022	0
22/5/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Funktionen utvärderar huruvuda transaktioner inträffade under årets första kvartal genom att använda `-1` som `period_no` argument i `inmonths()`-funktionen. 15 maj är `base_date` och infaller under årets andra kvartal (april-juni).

Diagram över `inmonths()` funktion med kvartalssegment och `period_no` inställd på `-1`



Därför kommer alla transaktioner som sker mellan januari och mars returnera det booleska resultatet TRUE.

Exempel 3 – first_month_of_year

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner för 2022 laddas i en tabell som heter '`Transactions`'.
- En föregående laddning med en extra variabel, `in_months`, som avgör vilka transaktioner som ägde rum under samma kvartal som 15 maj 2022.

I det här exemplet är organisationspolicyn att mars är den första månaden i räkenskapsåret.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inmonths(3,date,'05/15/2022', 0, 3) as in_months
       ;
Load
*
Inline
[
id,date,amount
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
8190,'3/30/2022',88.27
8191,'4/5/2022',57.42
```

```
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_months`

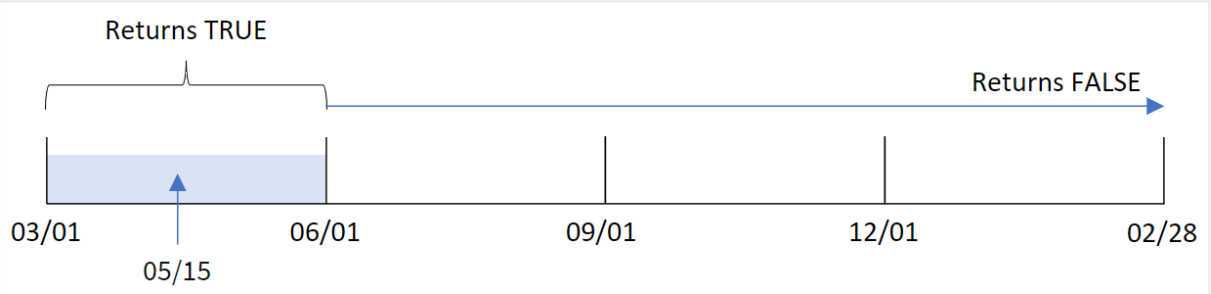
Resultattabell

Datum	in_months
2022-02-19	0
2022-03-07	-1
2022-03-30	-1
2022-04-05	-1
2022-04-16	-1
2022-05-01	-1
5/7/2022	-1
22/5/2022	-1
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0

Datum	in_months
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Genom att använda `3` som `first_month_of_year`-argumentet i `inmonths()`-funktionen, börjar funktionen året den 1 mars. Funktionen `inmonths()` delar sedan året i kvartal: mar-maj, jun-aug, sep-nov och dec-feb. Därför infaller 15 maj under årets första kvartal (mars-maj).

Diagram för `inmonths()`-funktionen med mars inställt som årets första månad.



Alla transaktioner som inträffar under dessa månader returnerar det booleska värdet TRUE.

Exempel 4 - Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är dock datauppsättningen oförändrad och har lästs in i programmet. Beräkningen som avgör om transaktioner har skett i samma kvartal som den 15 maj 2022 skapas som ett mått i ett diagram i applikationen.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
```

```
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
8190,'3/30/2022',88.27
8191,'4/5/2022',57.42
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `date`

För att beräkna om transaktioner ägde rum under samma kvartal som 15 maj skapar du följande mått:

```
=inmonths(3,date,'05/15/2022', 0)
```

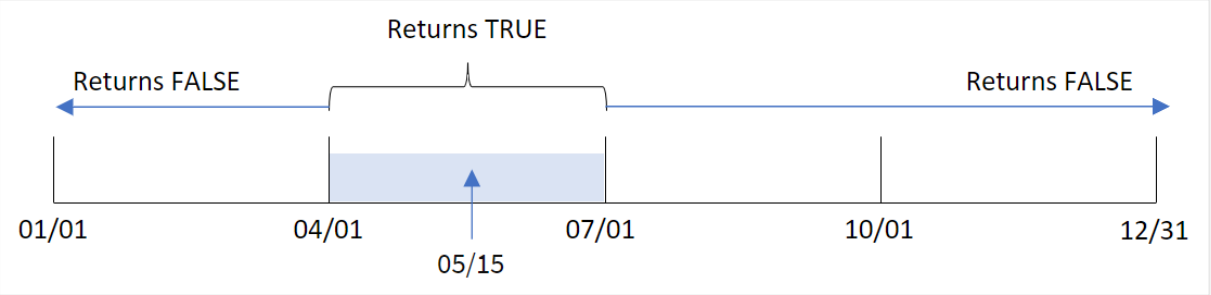
Resultattabell

datum	=inmonths(3,date,'05/15/2022', 0)
2022-02-19	0
2022-03-07	0
2022-03-30	0
2022-04-05	-1
2022-04-16	-1
2022-05-01	-1
5/7/2022	-1
22/5/2022	-1
6/15/2022	-1
6/26/2022	-1
7/9/2022	0

datum	=inmonths(3,date,'05/15/2022', 0)
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

`in_months`-fältet skapas i diagrammet genom att använda `inmonths()`-funktionen. Det första argumentet som tillhandahålls är `3` som delar upp året i kvartalssegment. Det andra argumentet som anges identifierar vilket fält som utvärderas, datumfältet i det här exemplet. Det tredje argumentet är ett hårdkodat datum för 15 maj, som är `base_date` och ett `period_no` av `0` är det sista argumentet.

Diagrammet över `inmonths()`-funktionen med kvartalssegment.



Maj infaller under årets andra kvartal. Därför kommer alla transaktioner som sker mellan 1 april och 30 juni att returnera det booleska resultatet TRUE. Det här valideras i resultattabellen.

Exempel 5 – Scenario

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning laddas i en tabell som heter `Products`.
- Tabellen innehåller följande fält:

- produkt-ID
- produkttyp
- tillverkningsdatum
- kostnadspris

Slutanvändaren vill ha ett diagramobjekt som visar, per produkttyp, kostnaden för produkterna som tillverkats under det första segmentet av 2021. Användaren vill kunna definiera det här segmentets längd.

Laddningsskript

```
SET vPeriod = 1;

Products:
Load
*
Inline
[
product_id,product_type,manufacture_date,cost_price
8188,product A,'2/19/2022',37.23
8189,product D,'3/7/2022',17.17
8190,product C,'3/30/2022',88.27
8191,product B,'4/5/2022',57.42
8192,product D,'4/16/2022',53.80
8193,product D,'5/1/2022',82.06
8194,product A,'5/7/2022',40.39
8195,product B,'5/22/2022',87.21
8196,product C,'6/15/2022',95.93
8197,product B,'6/26/2022',45.89
8198,product C,'7/9/2022',36.23
8199,product D,'7/22/2022',25.66
8200,product D,'7/23/2022',82.77
8201,product A,'7/27/2022',69.98
8202,product A,'8/2/2022',76.11
8203,product B,'8/8/2022',25.12
8204,product B,'8/19/2022',46.23
8205,product B,'9/26/2022',84.21
8206,product C,'10/14/2022',96.24
8207,product D,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark.

I början av laddningsskriptet har en variabel, `vPeriod`, skapats som är kopplad till variabelinmatningskontrollen.

Gör följande:

1. I resurspanelen klickar du på **Diagram**.
2. Under **Instrumentpanelsobjekt**, drar du en **Variabelinmatning** till arket.

3. Ange en titel för diagramobjektet.
4. Under **Variabel** väljer du **vPeriod** som namn och ställer in objektet så att det visas som en **rullgardinsmeny**.
5. Under **Värden** klickar du på **Dynamiska värden**. Ange följande:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. Lägg till en ny tabell i arket.
7. I **egenskap**spanelen under Data lägger du till `product_type` som en dimension.
8. Lägg till följande uttryck som ett mått:
   `=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))`
9. Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

product_type	**=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))**
produkt A	$88.27
produkt B	$37.23
produkt C	$17.17
produkt D	$0.00

`inmonths()`-funktionen använder användarindata som sitt argument för att definiera storleken på startsegmentet för året. Funktionen skickar tillverkningsdatumet för var och en av produkterna som `inmonths()`-funktionens andra argument. Genom att använda 1 januari som det tredje argumentet i `inmonths()`-funktionen kommer produkter med tillverkningsdatum under årets öppningssegment att returnera det booleska värdet TRUE och därför kommer summafunktionen att lägga till kostnaden för de här produkterna.

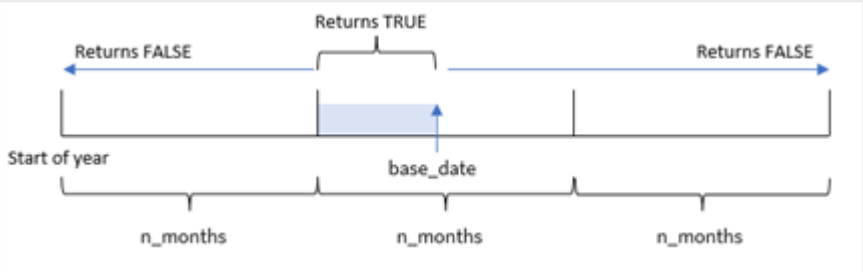
inmonthstodate

Med den här funktionen får vi reda på om en tidsmarkör finns inom delen av månaden, tvåmånadersperioden, kvartalet, fyramånadersperioden eller halvåret fram till och inklusive den sista millisekunden av `base_date`. Det går även att se om tidsmarkören finns inom en föregående eller senare tidsperiod.

```
InMonths (n_months, timestamp, base_date, period_no[, first_month_of_year ])
```

Returnerad datatyp: Booleansk

Diagram för `inmonthstodate`-funktionen.



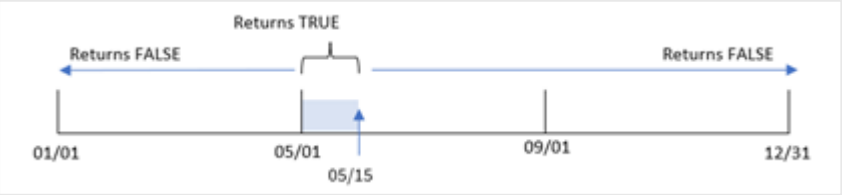
Argument

Argument	Beskrivning
n_months	Antalet månader som definierar perioden. Ett heltal eller uttryck vars resultat blir ett heltal som ska vara antingen: 1 (motsvaras av inmonth()-funktionen), 2 (tvåmånadersperiod), 3 (motsvaras av inquarter()-funktionen), 4 (fyramånadersperiod) eller 6 (halvår).
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera perioden.
period_no	Periodens startpunkt kan flyttas med **period_no**, ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den period som innehåller **base_date**. Negativa värden i **period_no** anger föregående perioder och positiva värden anger efterföljande perioder.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

I `inmonthstodate()`-funktionen fungerar `base_date` som slutpunkt för det aktuella årssegmentet som det ingår i.

Om året exempelvis delas upp i tredjedelssegment, och `base_date` var 15 maj skulle alla tidsmarkörer mellan början av januari och slutet av april returnera det booleska resultatet FALSE. Datum mellan 1 maj och 15 maj skulle returnera TRUE. Resten av året skulle returnera FALSE.

Diagram för det booleska resultatintervallet från `inmonthstodate`-funktionen.



Följande segment av året är tillgängliga i funktionen som `n_month`-argument.

`n_month` argument

Period	Antal månader
månad	1
två månader	2
kvartal	3
tertial	4
halvår	6

Användning

Funktionen `inmonthstodate()` returnerar ett booleskt resultat. Vanligtvis används denna typ av funktion som ett villkor i en `if expression`. Genom att använda `inmonthstodate()`-funktionen kan du välja den period som du vill utvärdera. Exempelvis, tillhandahålla en indatavariabel som låter användaren identifiera de produkter som tillverkats under en månad, ett kvartal eller ett halvår i en viss period, fram till ett visst datum.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`inmonthstodate(4, '01/25/2013', '04/25/2013', 0)`	Returnerar True, eftersom värdet för timestamp, 01/25/2013, befinner sig inom fyramånadersperioden 01/01/2013 till slutet av 04/25/2013, där värdet för base_date, 04/25/2013 finns.
`inmonthstodate(4, '04/26/2013', '04/25/2006', 0)`	Returnerar False, eftersom 04/26/2013 är utanför samma period som i förra exemplet.
`inmonthstodate(4, '09/25/2005', '02/01/2006', -1)`	Returnerar True, eftersom värdet för period_no, -1 flyttar sökperioden bakåt en period på fyra månader (värdet för n-months), vilket innebär att sökperioden är 01/09/2005 till 02/01/2006.
`inmonthstodate(4, '04/25/2006', '06/01/2006', 0, 3)`	Returnerar True eftersom värdet för first_month_of_year är angett till 3, vilket innebär att sökperioden är 03/01/2006 till 06/01/2006 i stället för 05/01/2006 till 06/01/2006.

Exempel 1 – inga ytterligare argument

Ladda skript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner för 2022 som läses in i en tabell som heter ”`Transactions`”.
- Ett datumfält i systemvariabeln `DateFormat` i formatet `(MM/DD/YYYY)`.
- En föregående LOAD-sats som innehåller följande:
    - `inmonthstodate()`-funktionen som är inställt som fältet `in_months_to_date`. Detta avgör vilka transaktioner som ägde rum under kvartalet fram till den 15 maj 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    inmonthstodate(3,date,'05/15/2022', 0) as in_months_to_date
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_months_to_date`

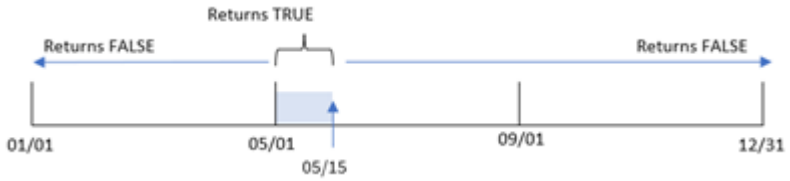
Resultattabell

datum	in_months_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

‘`in_months_to_date`’-fältet skapas i den föregående load-satsen genom att använda `inmonthstodate()`-funktionen.

Det första argumentet som tillhandahålls är `3`, som delar upp året i kvartalssegment. Det andra argumentet anger vilket fält som utvärderas. Det tredje argumentet är ett hårdkodat datum för 15 maj, vilket är `base_date` som definierar slutgränsen för segmentet. Ett `period_no` på `0` är det sista argumentet.

Diagram för `inmonthstodate`-funktionen utan ytterligare argument



Alla transaktioner som sker mellan 1 april och 15 maj returnerar det booleska resultatet TRUE. Transaktionsdatum utanför denna period returnerar FALSE.

Exempel 2 – period_no

Ladda skript och resultat

Översikt

Samma datauppsättning och scenario som det första exemplet används.

Men i det här exemplet är uppgiften att skapa ett fält, `previous_qtr_to_date`, som avgör huruvida transaktionerna ägde rum ett kvartal före 15 maj.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    inmonthstodate(3,date,'05/15/2022', -1) as previous_qtr_to_date
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
```

```
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `previous_qtr_to_date`

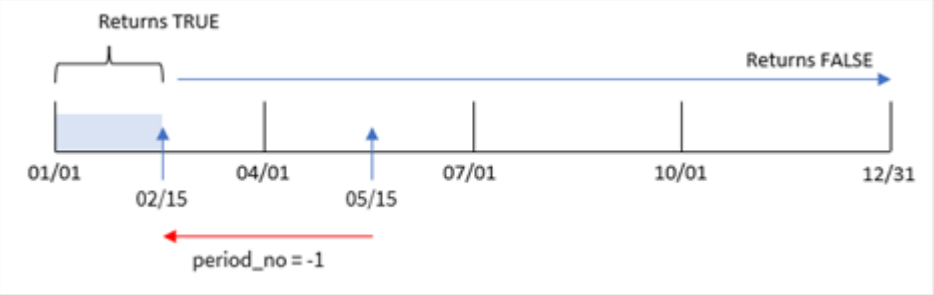
Resultattabell

datum	previous_qtr_to_date
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Genom att använda `-1` som `period_no`-argument i `inmonthstodate()`-funktionen skiftar funktionerna gränserna för segmentet i jämförelseåret med ett kvartal.

15 maj infaller under årets andra kvartal och därför motsvarar segmentet initialt mellan 1 april och 15 maj. `period_no`-argumentet förskjuter det här segmentet tre månader bakåt. Datumgränserna blir 1 januari till 15 februari.

Diagram för `inmonthstodate`-funktionen med period_no inställt till -1.



Därför kommer alla transaktioner som sker mellan 1 januari och 15 februari att returnera ett booleskt resultat som är TRUE.

Exempel 3 – first_month_of_year

Ladda skript och resultat

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är organisationspolicyn att mars är den första månaden i räkenskapsåret.

Skapa ett fält, `in_months_to_date`, som bestämmer vilka transaktioner som ägde rum under samma kvartal fram till den 15 maj 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    inmonthstodate(3,date,'05/15/2022', 0,3) as in_months_to_date
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
```

```
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_months_to_date`

Resultattabell

datum	previous_qtr_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	-1
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0

datum	previous_qtr_to_date
9/26/2022	0
10/14/2022	0
10/29/2022	0

Genom att använda `3` som `first_month_of_year`-argument i `inmonthstodate()`-funktionen startar funktionen året den 1 mars och delar sedan upp året i kvartal baserat på det första angivna argumentet. Därför är kvartalssegmenten:

- mar-maj
- jun-aug
- sep-nov
- dec-feb

`base_date` för 15 maj segmenterar då kvartalet mar-maj genom att ställa in slutdatum på 15 maj.

Diagram för `inmonthstodate`-funktionen med mars inställt som årets första månad.

Därför kommer alla transaktioner som sker mellan 1 mars och 15 maj att returnera det booleska resultatet TRUE, och transaktioner med datum utanför dessa gränser kommer att returnera värdet FALSE.

Exempel 4 – diagramexempel

Laddningsskript och diagramuttryck

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är datauppsättningen oförändrad och har lästs in i applikationen. Uppgiften är att skapa en beräkning som avgör om transaktioner ägde rum under samma kvartal som den 15 maj som ett mått i ett diagram i applikationen.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
```

```
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

```
date
```

För att beräkna om transaktioner ägde rum under samma kvartal som 15 maj skapar du följande mått:

```
=inmonthstodate(3,date,'05/15/2022', 0)
```

Resultattabell

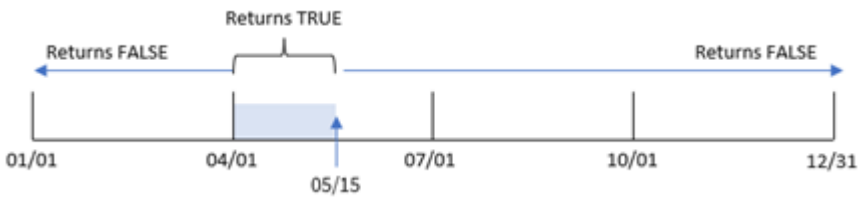
datum	=inmonthstodate(3,date,'05/15/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0

datum	=inmonthstodate(3,date,'05/15/2022', 0)
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

‘`in_months_to_date`-måttet skapas i diagrammet med hjälp av `inmonthstodate()`-funktionen.

Det första argumentet som tillhandahålls är `3`, som delar upp året i kvartalssegment. Det andra argumentet anger vilket fält som utvärderas. Det tredje argumentet är ett hårdkodat datum, 15 maj, vilket är `base_date` som definierar slutgränsen för det segmentet. Ett `period_no` på `0` är det sista argumentet.

Diagrammet över `inmonthstodate`-funktionen med kvartalssegment.



Alla transaktioner som sker mellan 1 april och 15 maj returnerar det booleska resultatet TRUE. Transaktionsdatum utanför detta segment returnerar FALSE.

Exempel 5 – Scenario

Ladda skript och resultat

Översikt

I det här exemplet laddas en datauppsättning i en tabell som heter `Sales`. Tabellen innehåller följande fält:

- Produkt-ID
- Produkttyp

- Försäljningsdatum
- Försäljningspris

Slutanvändaren vill ha ett diagram som visar försäljningen av produkter under perioden fram till 24 december 2022, efter produkttyp. Användaren vill kunna definiera den här periodens längd.

Laddningsskript

```
SET vPeriod = 1;

Products:
Load
*
Inline
[
product_id,product_type,sales_date,sales_price
8188,product A,'9/19/2022',37.23
8189,product D,'10/27/2022',17.17
8190,product C,'10/30/2022',88.27
8191,product B,'10/31/2022',57.42
8192,product D,'11/16/2022',53.80
8193,product D,'11/28/2022',82.06
8194,product A,'12/2/2022',40.39
8195,product B,'12/5/2022',87.21
8196,product C,'12/15/2022',95.93
8197,product B,'12/16/2022',45.89
8198,product C,'12/19/2022',36.23
8199,product D,'12/22/2022',25.66
8200,product D,'12/23/2022',82.77
8201,product A,'12/24/2022',69.98
8202,product A,'12/24/2022',76.11
8203,product B,'12/26/2022',25.12
8204,product B,'12/27/2022',46.23
8205,product B,'12/27/2022',84.21
8206,product C,'12/28/2022',96.24
8207,product D,'12/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark.

I början av laddningsskriptet har en variabel, `vPeriod`, skapats som är kopplad till variabelinmatningskontrollen.

Gör följande:

1. I resurspanelen klickar du på **Diagram**.
2. Under **Instrumentpanelsobjekt**, drar du en **Variabelinmatning** till arket.
3. Ange en titel för diagrammet.

4. Under **Variabel** väljer du **vPeriod** som namn och ställer in objektet så att det visas som en **rullgardinsmeny**.
5. Under **Värden** klickar du på **Dynamiska värden**. Ange följande:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. Lägg till en ny tabell i arket.
7. I **egenskap**spanelen under Data lägger du till `product_type` som en dimension.
8. Lägg till följande uttryck som ett mått:
   `=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))`
9. Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

product_type	=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))
produkt A	$186.48
produkt B	$190.52
produkt C	$220.43
produkt D	$261.46

`inmonthstodate()`-funktionen använder användarindata som sitt argument för att definiera storleken på startsegmentet för året.

Funktionen skickar försäljningsdatumet för var och en av produkterna som `inmonthstodate()`-funktionens andra argument. Genom att använda 24 december som tredje argument i `inmonthstodate()`-funktionen returnerar produkter med försäljningsdatum som inträffar under den definierade perioden fram till och inklusive 24 december returnerar det booleska värdet TRUE. Summafunktionen lägger till försäljningen av produkterna.

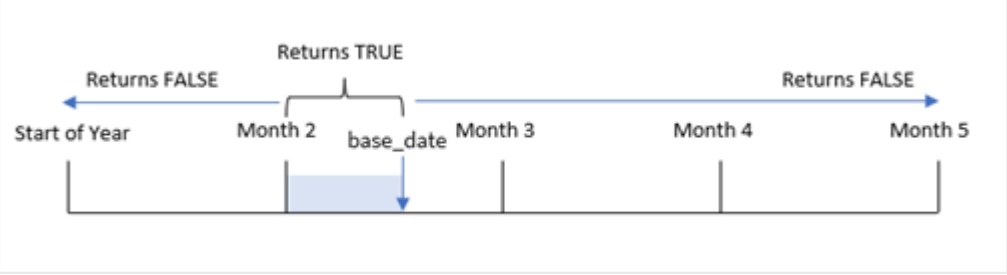
inmonthtodate

Returnerar True om **date** ligger inom den del av månaden som innehåller **basedate** fram till och inklusive den sista millisekunden av **basedate**.

```
InMonthToDate (timestamp, base_date, period_no)
```

Returnerad datatyp: Booleansk

Diagram för `inmonthtodate`-funktionen.



`inmonthtodate()`-funktionen identifierar en vald månad som ett segment. Startgränsen är månadens början. Slutgränsen kan ställas in till ett senare datum i månaden. Funktionen fastställer sedan huruvida en uppsättning datum ligger inom det här segmentet eller inte och returnerar det booleska värdet TRUE eller FALSE.

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera månaden.
period_no	Månadens startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger den månad som innehåller **base_date**. Negativa värden i **period_no** anger föregående månader och positiva värden anger efterföljande månader.

Användning

Funktionen `inmonthtodate()` returnerar ett booleskt resultat. Vanligtvis används denna typ av funktion som ett villkor i en `if expression`. `inmonthtodate()`-funktionen returnerar en aggregering eller beräkning som beror på om ett datum inträffade under månaden fram till och inklusive det aktuella datumet.

Till exempel kan `inmonthtodate()`-funktionen användas för att identifiera all utrustning som tillverkats under en månad fram till ett specifikt datum.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`inmonthtodate ('01/25/2013', '25/01/2013', 0)`	Returnerar True
`inmonthtodate ('01/25/2013', '24/01/2013', 0)`	Returnerar False
`inmonthtodate ('01/25/2013', '28/02/2013', -1)`	Returnerar True

Exempel 1 – inga ytterligare argument

Ladda skript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner för 2022 läses in i en tabell som heter 'Transactions'.
- Ett datumfält anges i DateFormat-systemvariabeln (MM/DD/YYYY)-formatet.
- En föregående LOAD-sats som innehåller följande:
  - inmonthtodate()-funktionen som är inställt som fältet 'in_month_to_date'. Detta avgör vilka transaktioner som ägde rum mellan 1 juli och 26 juli 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    inmonthtodate(date,'07/26/2022', 0) as in_month_to_date
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- date
- in_month_to_date

Resultattabell

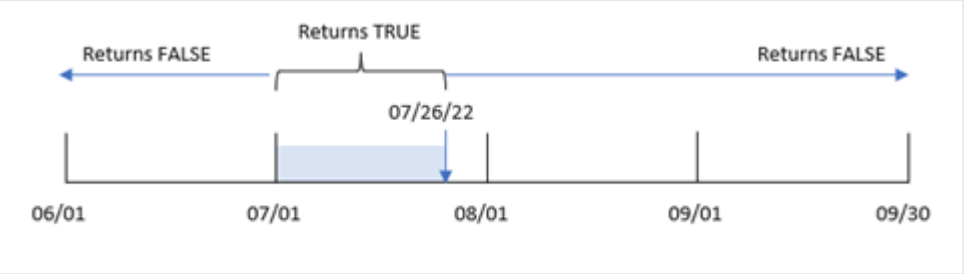
datum	in_month_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	-1
7/22/2022	-1
7/23/2022	-1
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

‘in_month_to_date’-fältet skapas i den föregående load-satsen genom att använda inmonthtodate()-funktionen.

Det första argumentet anger vilket fält som utvärderas. Det andra argumentet är ett hårdkodat datum, 26 juli, som är base_date. Det här base_date-argumentet identifierar vilken månad som är segmenterad och slutgränsen för det segmentet.

Ett period_no på 0 är det sista argumentet, vilket betyder att funktionen inte jämför månaderna före eller efter den segmenterade månaden.

Diagram för `inmonthtodate`-funktionen utan ytterligare argument



Som följd returnerar alla transaktioner som sker mellan 1 juli och 26 juli det booleska resultatet TRUE. Alla transaktioner som inträffar i juli efter 26 juli returnerar det booleska resultatet FALSE, liksom alla transaktioner under någon annan månad under året.

Exempel 2 – period_no

Ladda skript och resultat

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är uppgiften att skapa ett fält, `six_months_prior`, som avgör vilka transaktioner som ägde rum sex hela månader före 1 juli och 26 juli.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    inmonthtodate(date,'07/26/2022', -6) as six_months_prior
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
```

```
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

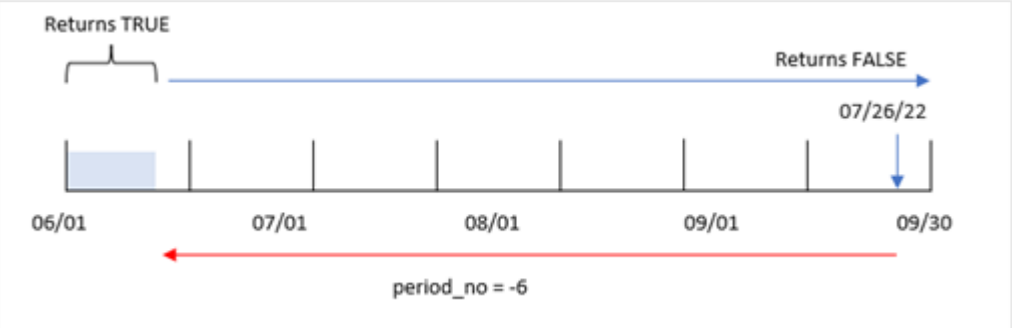
- `date`
- `six_months_prior`

Resultattabell

datum	six_months_prior
1/7/2022	-1
1/19/2022	-1
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Genom att använda `-6` som `period_no`-argument i `inmonthtodate()`-funktionen skiftar funktionerna gränserna för segmentet i jämförelsemånaden med sex månader. Initialt är månadssegmentet mellan 1 juli och 26 juli. `period_no` förskjuter sedan detta segment sex månader bakåt och datumgränserna skiftas och ligger mellan 1 januari och 26 januari.

Diagram för `inmonthtodate`-funktionen med period_no inställt till -6.



Som följd returnerar alla transaktioner som sker mellan 1 januari och 26 januari det booleska resultatet TRUE.

Exempel 3 – Diagramexempel

Laddningsskript och diagramuttryck

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är datauppsättningen oförändrad och har lästs in i applikationen. Uppgiften är att skapa en beräkning som avgör om transaktioner har skett mellan 1 juli och 26 juli som ett mått i ett diagramobjekt i applikationen.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
```

```
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

```
date
```

För att beräkna om transaktioner ägde rum mellan 1 juli och 26 juli skapar du följande mått:

```
=inmonthtodate(date,'07/26/2022', 0)
```

Resultattabell

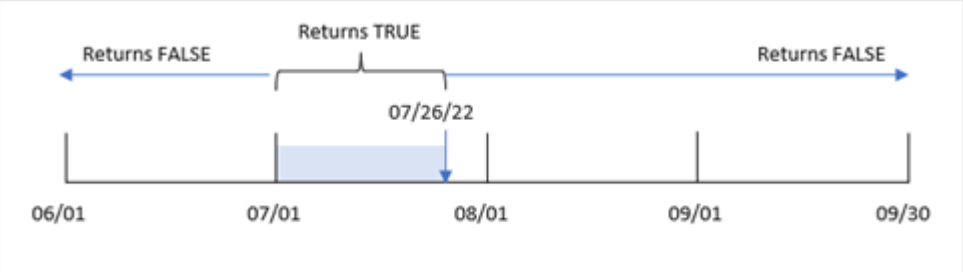
datum	=inmonthtodate(date,'07/26/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	-1
7/22/2022	-1
7/23/2022	-1
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0

datum	=inmonthtodate(date,'07/26/2022', 0)
9/26/2022	0
10/14/2022	0
10/29/2022	0

`in_month_to_date`-fältmåttet skapas i diagrammet genom att använda `inmonthtodate()`-funktionen.

Det första argumentet anger vilket fält som utvärderas. Det andra argumentet är ett hårdkodat datum, 26 juli, som är `base_date`. Det här base_date-argumentet identifierar vilken månad som är segmenterad och slutgränsen för det segmentet. Ett `period_no` på 0 är det sista argumentet. Detta innebär att funktionen inte jämför månaderna före eller efter den segmenterade månaden.

Diagram för `inmonthtodate`-funktionen utan ytterligare argument



Som följd returnerar alla transaktioner som sker mellan 1 juli och 26 juli det booleska resultatet TRUE. Alla transaktioner som inträffar i juli efter 26 juli returnerar det booleska resultatet FALSE, liksom alla transaktioner under någon annan månad under året.

Exempel 4 – Scenario

Ladda skript och resultat

Översikt

I det här exemplet laddas en datauppsättning i en tabell som heter `Products`. Tabellen innehåller följande fält:

- Produkt-ID
- Tillverkningsdatum
- Kostnadspris

På grund av utrustningsfel var produkter som tillverkades i juli månad 2022 defekta. Problemet löstes den 27 juli 2022.

Slutanvändaren vill ha ett diagramobjekt som visar status för tillverkade produkter som var "defekta" (booleskt värde: TRUE) eller "felfria" (booleskt värde: FALSE), efter datum, samt kostnaden för de produkter som tillverkades den månaden.

Laddningsskript

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `=monthname(manufacture_date)`
- `=if(Inmonthtodate(manufacture_date,makedate(2022,07,26),0),'Defective','Faultless')`

För att beräkna den sammanlagda kostnaden för produkten skapar du följande mått:

```
=sum(cost_price)
```

Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

monthname (manufacture_date)	if(Inmonthtodate(manufacture_date,makedate (2022,07,26),0),'Defective','Faultless')	=sum(cost_ price)
Jan 2022	Felfri	$54.40
Feb 2022	Felfri	$145.69
Mar 2022	Felfri	$53.80

monthname (manufacture_date)	if(Inmonthtodate(manufacture_date,makedate (2022,07,26),0),'Defective','Faultless')	=sum(cost_ price)
Apr 2022	Felfri	$82.06
maj 2022	Felfri	$127.60
Jun 2022	Felfri	$141.82
Jul 2022	Defekt	$144.66
Jul 2022	Felfri	$69.98
Aug 2022	Felfri	$147.46
Sep 2022	Felfri	$84.21
Okt 2022	Felfri	$163.91

Funktionen `inmonthtodate()` returnerar ett booleskt värde vid utvärdering av tillverkningsdatumen för var och en av produkterna.

För de datum som returnerar det booleska värdet TRUE markeras produkten som "Defekt". För alla produkter som returnerar ett värde på FALSE, och därför inte har tillverkats fram till och med 26 juli markerar den produkterna som "Felfria".

inquarter

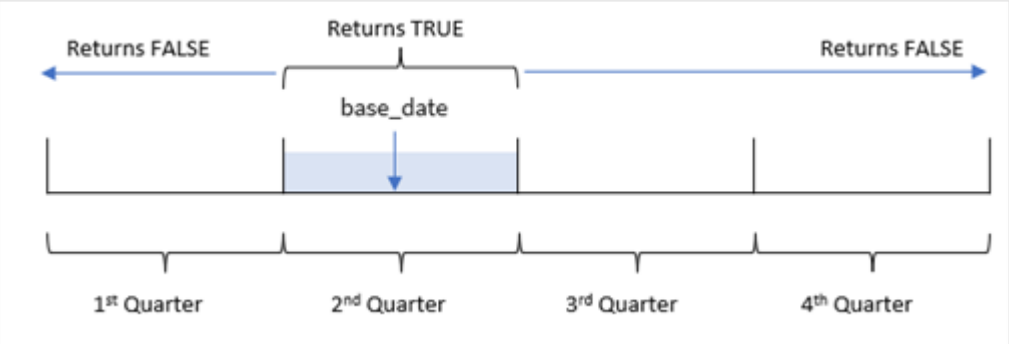
Denna funktion returnerar True om **timestamp** ligger inom det kvartal som innehåller **base_ date**.

```
InQuarter (timestamp, base_date, period_no[, first_month_of_year])
```

Returnerad datatyp: Booleansk

I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.

Diagram för `inquarter()`-funktionens intervall



Med andra ord delar `inquarter()`-funktionen in året i fyra kvartal mellan 1 januari och 31 december. Du kan använda `first_month_of_year`-argumentet för att ändra vilken månad som anges som första i din applikation. Kvartalen ändras då baserat på det argumentet. `base_date`-function identifierar vilket kvartal som ska användas som jämförelse för funktionen. Slutligen returnerar funktionen ett booleskt resultat när du jämför datumvärdena med det kvartalsegmentet.

Användning

Funktionen `inquarter()` returnerar ett booleskt resultat. Vanligtvis kommer denna typ av funktion att användas som ett villkor i en `if expression`. Detta returnerar en aggregering eller beräkning som beror på om ett datum inträffade under det valda kvartalet.

`inquarter()`-funktionen kan användas för att identifiera all utrustning som tillverkatas under ett kvartalssegment, baserat på de datum när utrustningen tillverkades.

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera kvartalet.
period_no	Kvartalets startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger det kvartal som innehåller **base_date**. Negativa värden i **period_no** anger föregående kvartal och positiva värden anger efterföljande kvartal.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Du kan använda följande värden för att ställa in den första månaden på året i argumentet `first_month_of_year`:

first_month_of_year values

Månad	värde
Februari	2
mars	3
april	4
Maj	5
juni	6
juli	7
augusti	8
september	9
oktober	10

Månad	värde
November	11
december	12

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
inquarter ('01/25/2013', '01/01/2013', 0)	Returnerar TRUE
inquarter ('01/25/2013', '04/01/2013', 0)	Returnerar FALSE
inquarter ('01/25/2013', '01/01/2013', -1)	Returnerar FALSE
inquarter ('12/25/2012', '01/01/2013', -1)	Returnerar TRUE
inquarter ('01/25/2013', '03/01/2013', 0, 3)	Returnerar FALSE
inquarter ('03/25/2013', '03/01/2013', 0, 3)	Returnerar TRUE

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2022 som läses in i en tabell som heter ‘`Transactions`’.
- En föregående laddning som innehåller `inquarter()`-funktionen som är inställd som ‘`in_quarter`’-fältet och avgör vilka transaktioner som ägde rum under samma kvartal som 15 maj 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inquarter (date,'05/15/2022', 0) as in_quarter
       ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_quarter`

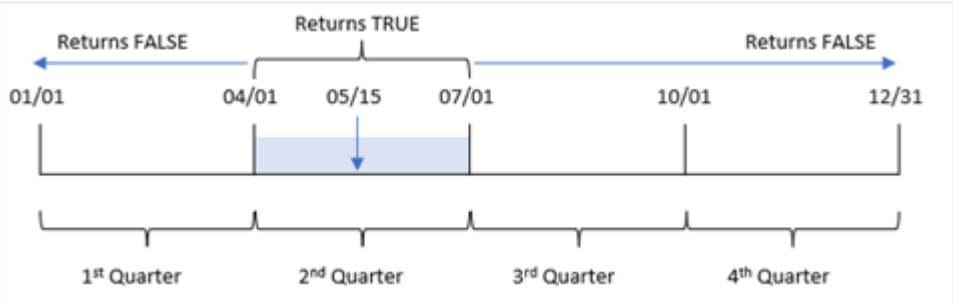
Resultattabell

Datum	in_quarter
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0

Datum	in_quarter
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	-1
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Fältet 'in_quarter' skapas i den föregående load-satsen genom att använda `inquarter()`-funktionen. Det första argumentet anger vilket fält som utvärderas. Det andra argumentet är ett hårdkodat datum för 15 maj som identifierar vilket kvartal som ska definieras som komparator. `period_no`-värdet `0` är det sista argumentet, och säkerställer att `inquarter()`-funktionen inte jämför kvartalen före eller efter det segmenterade kvartalet.

Diagram för `inquarter()`-funktionen med 15 maj som basdatum



Alla transaktioner som sker mellan 1 april och slutet av 30 juni returnerar det booleska resultatet TRUE.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner från 2022 som laddas i en tabell som heter 'Transactions'.
- En föregående laddning som innehåller inquarter()-funktionen som ställs in som previous_quarter-fältet och avgör vilka transaktioner som ägde rum under det kvartal som föregick kvartalet innehållande 15 maj 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inquarter (date,'05/15/2022', -1) as previous_qtr
       ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

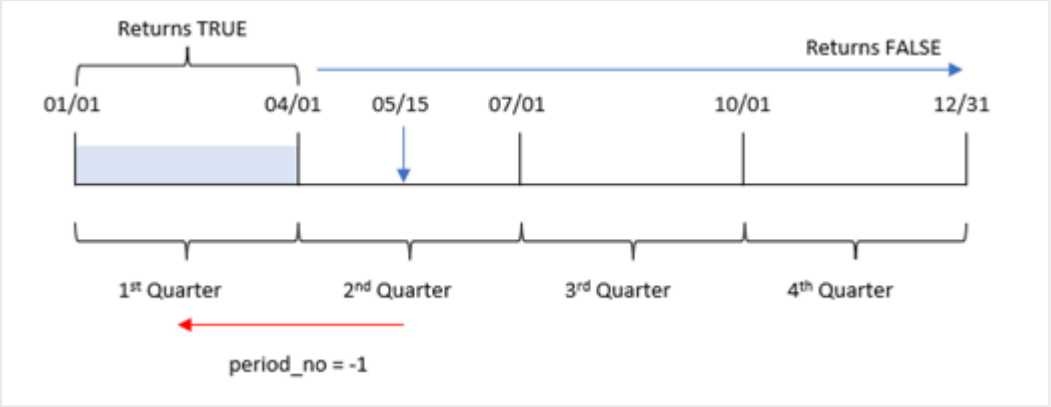
- `date`
- `previous_qtr`

Resultattabell

Datum	previous_qtr
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	-1
3/16/2022	-1
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Genom att använda `-1` som `period_no`-argument i `inquarter()`-funktionen skiftas gränserna för i jämförelsekvartalet bakåt med ett helt kvartal. 15 maj infaller under årets andra kvartal och därför motsvarar segmentet initialt kvartalet mellan 1 april och 30 juni. `period_no` förskjuter sedan detta segment tre månader bakåt, vilket medför att datumgränserna blir 1 januari till 30 mars.

Diagram för `inquarter()`-funktionen med 15 maj som basdatum



Därför kommer alla transaktioner som sker mellan 1 januari och 30 mars returnera det booleska resultatet TRUE.

Exempel 3 – first_month_of_year

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner från 2022 som laddas i en tabell som heter 'Transactions'.
- En föregående laddning som innehåller `inquarter()`-funktionen som ställs in som `in_quarter`-fältet och avgör vilka transaktioner som ägde rum under samma kvartal som 15 maj 2022.

I det här exemplet är dock organisationspolicyn att mars är den första månaden i räkenskapsåret.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inquarter (date,'05/15/2022', 0, 3) as in_quarter
       ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
```

```
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `previous_qtr`

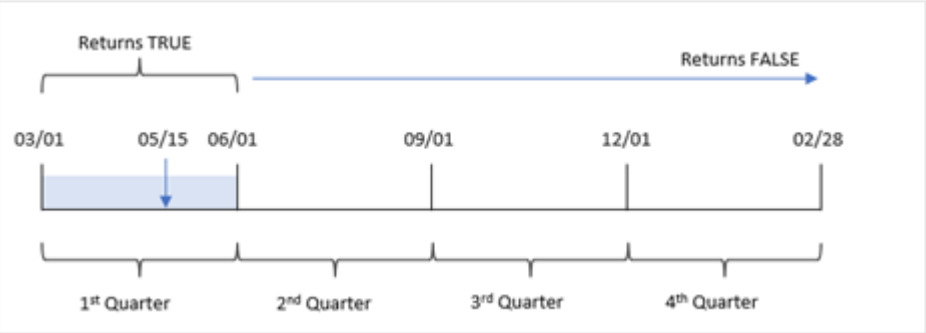
Resultattabell

Datum	previous_qtr
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	-1
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0

Datum	previous_qtr
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Genom att använda `3` som `first_month_of_year`-argument i `inquarter()`-funktionen ställs 1 mars in som årets början och delas sedan året upp i kvartal. Därför är kvartalssegmenten mar-maj, jun-aug, sep-nov och dec-feb. `base_date` som 15 maj ställer in kvartalet mar-maj som jämförande kvartal för funktionen.

Diagram för `inquarter()`-funktionen med mars inställt som årets första månad



Därför kommer alla transaktioner som sker mellan 1 mars och 31 maj returnera det booleska resultatet TRUE.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner från 2022 som laddas i en tabell som heter '`Transactions`'.
- En föregående laddning som innehåller `inquarter()`-funktionen som ställs in som `in_quarter`-fältet och avgör vilka transaktioner som ägde rum under samma kvartal som 15 maj 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
```

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `date`

Skapa följande mått för att beräkna om transaktioner ägde rum under samma kvartal som 15 maj:

```
=inquarter(date,'05/15/2022', 0)
```

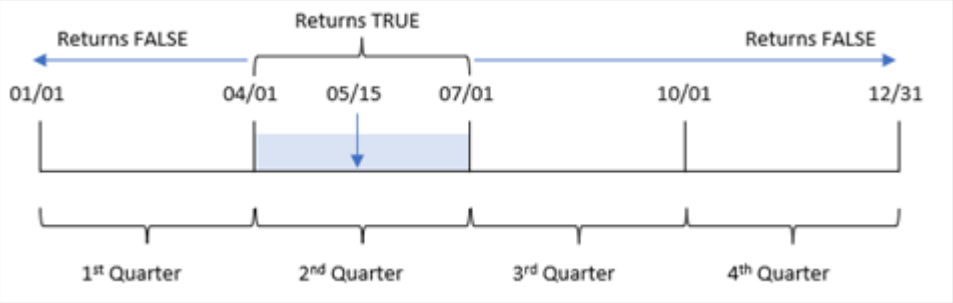
Resultattabell

Datum	in_quarter
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1

Datum	in_quarter
6/15/2022	-1
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

`in_quarter`-måttet skapas i diagrammet med hjälp av `inquarter()`-funktionen. Det första argumentet anger vilket fält som utvärderas. Det andra argumentet är ett hårdkodat datum för 15 maj som identifierar vilket kvartal som ska definieras som komparator. `period_no`-värdet `0` är det sista argumentet, och säkerställer att `inquarter()`-funktionen inte jämför kvartalen före eller efter det segmenterade kvartalet.

Diagram för `inquarter()`-funktionen med 15 maj som basdatum



Alla transaktioner som sker mellan 1 april och slutet av 30 juni returnerar det booleska resultatet TRUE.

Exempel 5 – Scenario

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning laddas i en tabell som heter `Products`.
- Tabellen innehåller följande fält:
    - produkt-ID
    - produkttyp
    - tillverkningsdatum
    - kostnadspris

I det här exemplet har man identifierat att utrustningsfel var orsaken till att produkter som tillverkades under kvartalet den 15 maj, 2022 var defekta. Slutanvändaren vill ha ett diagram som visar, efter kvartalsnamn, status för vilka produkter som tillverkades som var "defekta" eller "felfria", samt kostnaden för de produkter som tillverkades det kvartalet.

Laddningsskript

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

```
=quartername(manufacture_date)
```

Skapa följande mått:

- `=if(only(InQuarter(manufacture_date,makedate(2022,05,15),0)),'Defective','Faultless')`, för att identifiera vilka av produkterna som är defekta och vilka som är felfria med `inquarter()`-funktionen:
- `=sum(cost_price)`, för att visa den sammanlagda kostnaden för varje produkt.

Gör följande:

1. Ange måttens **Nummerformatering** till **Pengar**.
2. Under **Utseende**, stänger du av **Total**.

Resultattabell

quartername (manufacture_date)	=if(only(InQuarter(manufacture_date,makedate (2022,05,15),0)),'Defective','Faultless')	=sum(cost_ price)
Jan-mars 2022	Felfri	253,89
Apr-jun 2022	Defekt	351.48
Jul-sep 2022	Felfri	446.31
Okt-dec 2022	Felfri	163,91

Funktionen `inquarter()` returnerar ett booleskt värde vid utvärdering av tillverkningsdatumen för var och en av produkterna. För alla produkter som tillverkats under det kvartal där 15 maj ingår returnerar funktionen `inquarter()` det booleska värdet TRUE och markerar produkterna som "Defekta". För alla produkter som returnerar ett värde på FALSE, och därför inte tillverkats under detta kvartal, markerar den produkterna som "Felfria".

inquartertodate

Denna funktion returnerar True om **timestamp** ligger inom den del av det kvartal som innehåller **base_date** fram till och inklusive den sista millisekunden av **base_date**.

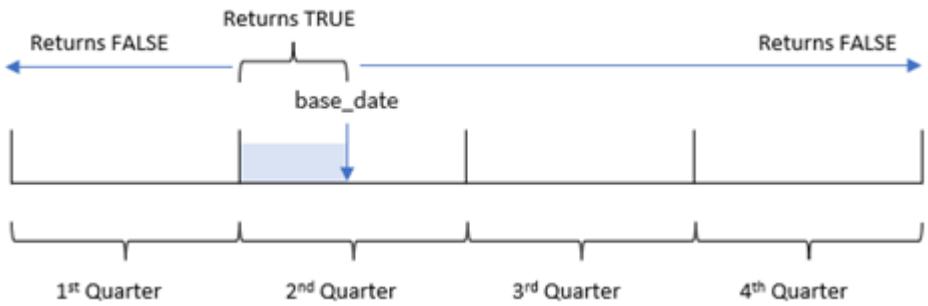
Syntax:

```
InQuarterToDate (timestamp, base_date, period_no [, first_month_of_year])
```

Returnerad datatyp: Booleansk

> *I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.*

Diagram för inquartertodate-funktionen



Funktionen `inquartertodate()` delar upp året i fyra lika stora kvartal mellan 1 januari och 31 december (eller den användardefinierade början på året och dess motsvarande slutdatum). Med hjälp av `base_date`kommer funktionen sedan att segmentera ett visst kvartal, med `base_date` som identifierar både kvartal och det högsta tillåtna datumet för det kvartalssegmentet. Slutligen returnerar funktionen ett booleskt resultat när du jämför de föreskrivna datumvärdena med det segmentet.

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera kvartalet.
period_no	Kvartalets startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger det kvartal som innehåller **base_date**. Negativa värden i **period_no** anger föregående kvartal och positiva värden anger efterföljande kvartal.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Användning

Funktionen `inquartertodate()` returnerar ett booleskt resultat. Vanligtvis kommer denna typ av funktion att användas som ett villkor i ett `if`-uttryck. `inquartertodate()`-funktionen returnerar en aggregering eller beräkning beroende på om ett utvärderat datum inträffade i kvartalet till och med datumet i fråga.

Till exempel kan `inquartertodate()`-funktionen användas för att identifiera all utrustning som tillverkats under ett kvartal fram till ett specifikt datum.

Exempel på funktioner

Exempel	Resultat
`inquartertodate ('01/25/2013', '03/25/2013', 0)`	Returnerar `TRUE`, eftersom värdet för `timestamp`, 01/25/2013, ligger inom tremånadersperioden från 01/01/2013 till 03/25/2013, där värdet för `base_date` 03/25/2013.

Exempel	Resultat
`inquartertodate ('04/26/2013', '03/25/2013', 0)`	Returnerar `FALSE` eftersom 04/26/2013 är utanför samma period som i förra exemplet.
`inquartertodate ('02/25/2013', '06/09/2013', -1)`	Returnerar`TRUE` eftersom värdet för `period_no`, som är -1, flyttar sökperioden en tremånadersperiod bakåt (ett kvartal av året. Detta innebär att sökperioden är 01/01/2013 till 03/09/2013.
`inquartertodate ('03/25/2006', '04/15/2006', 0, 2)`	Returnerar `TRUE` eftersom värdet för `first_month_of_year` är inställt på 2, vilket gör sökperioden till 2006-02-01 till 2006-04-15 istället för 2006- 04-01 till 2006-04-15.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2022 som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i `DateFormat`-systemvariabeln.
- Skapandet av ett fält, `in_quarter_to_date`, som bestämmer vilka transaktioner som ägde rum under kvartalet fram till den 15 maj 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
```

```
        *,
        inquartertodate(date,'05/15/2022', 0) as in_quarter_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_quarter_to_date`

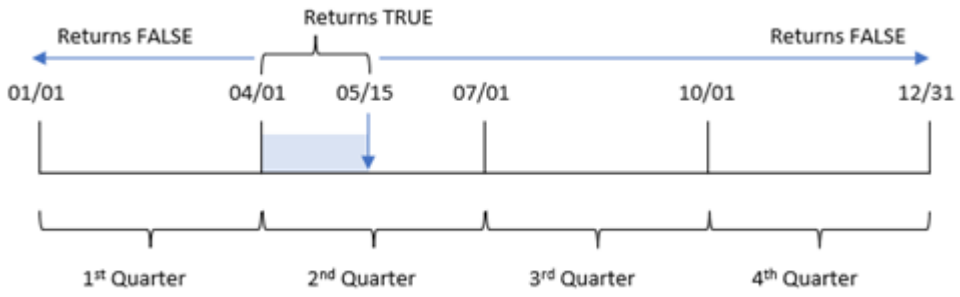
Resultattabell

datum	inquartertodate
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1

datum	inquartertodate
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

`in_quarter_to_date`-fältet skapas i föregående LOAD-sats med hjälp av funktionen `inquartertodate()`. Det första argumentet som anges identifierar vilket fält som utvärderas. Det andra argumentet är ett hårdkodat datum för den 15 maj, vilket är `base_date` som identifierar vilket kvartal som ska segmenteras och definierar slutgränsen för det segmentet. `period_no`-värdet 0 är det sista argumentet, vilket betyder att funktionen inte jämför kvartalen före eller efter det segmenterade kvartalet.

Diagram för inquartertodate-funktionen, inga ytterligare argument



Alla transaktioner som sker mellan 1 april och 15 maj returnerar det booleska resultatet `TRUE`. Transaktionsdatum som är 16 maj och senare kommer att returneras `FALSE`, liksom alla transaktioner före 1 april.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Skapandet av ett fält, `previous_qtr_to_date`, som bestämmer vilka transaktioner som ägde rum ett helt kvartal innan den 15 maj 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inquartertodate(date,'05/15/2022', -1) as previous_qtr_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

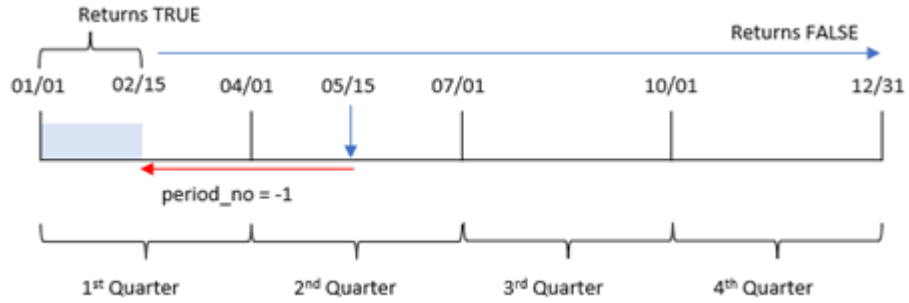
- `date`
- `previous_qtr_to_date`

Resultattabell

datum	previous_qtr_to_date
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

`period_no`-värdet på -1 indikerar att `inquartertodate ()`-funktionen jämför det ingående kvartalssegmentet med föregående kvartal. 15 maj infaller under årets andra kvartal, så segmentet motsvarar initialt mellan 1 april och 15 maj. `period_no` skjuter sedan bak detta segment till tre månader tidigare, vilket gör att datumgränserna blir 1 januari till 15 februari.

Exempel på diagram för inquartertodate-funktionen, period_no



Därför kommer alla transaktioner som sker mellan 1 januari och 15 februari att returnera ett booleskt resultat som är `TRUE`.

Exempel 3 – first_month_of_year

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Skapandet av ett fält, `in_quarter_to_date`, som bestämmer vilka transaktioner som ägde rum under kvartalet fram till den 15 maj 2022.

I det här exemplet anger vi mars som den första månaden av räkenskapsåret.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inquartertodate(date,'05/15/2022', 0,3) as in_quarter_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
```

```
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_quarter_to_date`

Resultattabell

datum	inquartertodate
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	-1
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0

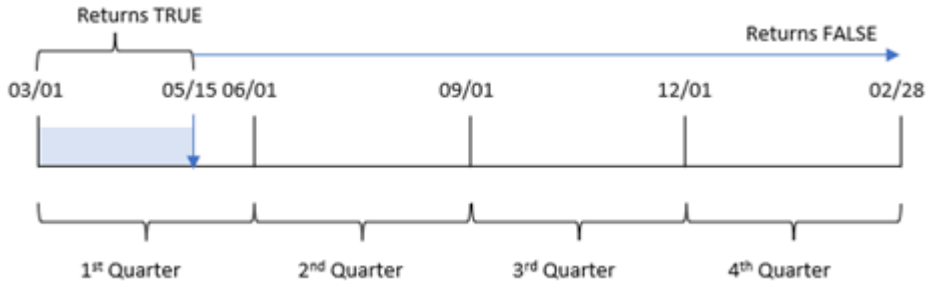
datum	inquartertodate
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Genom att använda 3 som `first_month_of_year`-argument i `inquartertodate()`-funktionen startar funktionen året den 1 mars och delar sedan upp året i kvartal. Därför är kvartalssegmenten:

- Mars till maj
- Juni till augusti
- September till november
- December till februari

`base_date` den 15 maj segmenterar då kvartalet mars till maj genom att ställa in slutdatum på 15 maj.

Exempel på diagram för inquartertodate-funktionen, first_month_of_year



Därför kommer alla transaktioner som sker mellan 1 mars och 15 maj att returnera det booleska resultatet `TRUE`, medan transaktioner med datum utanför dessa gränser kommer att returnera värdet `FALSE`.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet. I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som avgör om transaktioner har skett i samma kvartal som den 15 maj skapas som ett mått i diagramobjektet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
```

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:`date`.

Skapa följande mått:

```
=inquartertodate(date,'05/15/2022', 0)
```

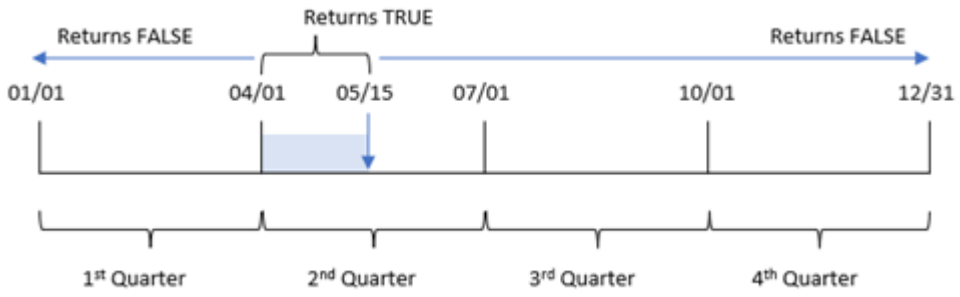
Resultattabell

datum	=inquartertodate(date,'05/15/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0

datum	=inquartertodate(date,'05/15/2022', 0)
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

`in_quarter_to_date`-måttet skapas i ett diagramobjekt med hjälp av `inquartertodate()`-funktionen. Det första argumentet är datumfältet som utvärderas. Det andra argumentet är ett hårdkodat datum för den 15 maj, vilket är `base_date` som identifierar vilket kvartal som ska segmenteras samt definierar slutgränsen för det segmentet. `period_no`-värdet 0 är det sista argumentet, vilket betyder att funktionen inte jämför kvartalen före eller efter det segmenterade kvartalet.

Exempel på diagram för inquartertodate-funktionen diagramobjekt



Alla transaktioner som sker mellan 1 april och 15 maj returnerar det boolesk resultatet `TRUE`. Transaktionsdatum som är 16 maj och senare kommer att returneras som `FALSE`, liksom alla transaktioner före 1 april.

Exempel 5 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning läses in i en tabell som heter `Products`.
- Information om produkt-ID, tillverkningsdatum och självkostnadspris.

Den 15 maj 2022 identifierades ett utrustningsfel i tillverkningsprocessen och åtgärdades. Produkter som tillverkats under det kvartalet fram till detta datum kommer att vara defekta. Slutanvändaren vill ha ett diagramobjekt som visar, efter kvartalsnamn, status för tillverkade produkter som var "defekta" eller "felfria", samt kostnaden för de produkter som tillverkades i kvartalet fram till det datumet.

Laddningsskript

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell. Skapa en dimension för att visa kvartalsnamnen:
   `=quartername(manufacture_date)`
2. Skapa sedan en dimension för att identifiera vilka av produkterna som är defekta och vilka som är felfria:
   `=if(inquartertodate(manufacture_date,makedate(2022,05,15),0),'Defective','Faultless')`
3. Skapa ett mått för att summera `cost_price` av produkterna:
   `=sum(cost_price)`
4. Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

quartername (manufacture_date)	if(inquartertodate(manufacture_date,makedate (2022,05,15),0),'Defective','Faultless')	=sum(cost_ price)
Jan-mars 2022	Felfri	$253.89
Apr-jun 2022	Felfri	$229.03
Apr-jun 2022	Defekt	$122.45
Jul-sep 2022	Felfri	$446.31
Okt-dec 2022	Felfri	$163.91

Funktionen `inquartertodate()` returnerar ett booleskt värde vid utvärdering av tillverkningsdatumen för var och en av produkterna. För de som returnerar ett booleskt värde på `TRUE`, markerar det produkterna som `'Defective'`. För alla produkter som returnerar värdet `FALSE`, och därför inte tillverkas under kvartalet fram till och med den 15 maj, markeras produkterna som `'Faultless'`.

inweek

Denna funktion returnerar True om **timestamp** ligger inom den vecka som innehåller **base_ date**.

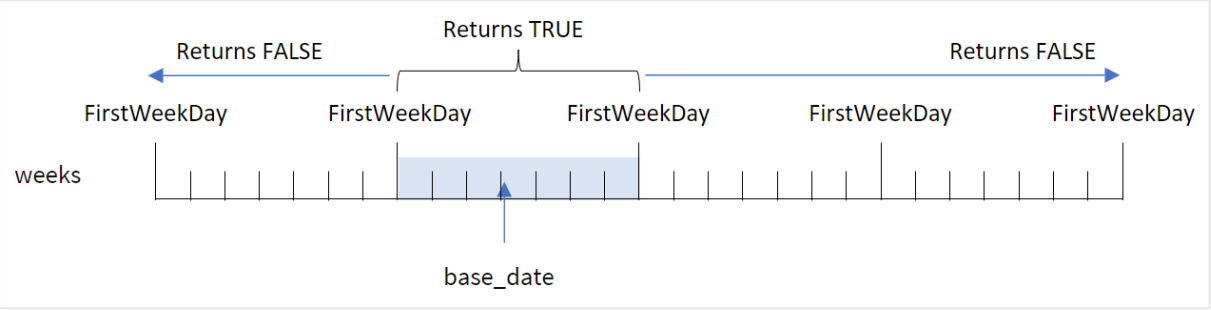
Syntax:

```
InWeek (timestamp, base_date, period_no[, first_week_day])
```

Returnerad datatyp: Booleansk

I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.

Diagram för `inweek()`-funktionens intervall



Funktionen `inweek()` använder argumentet `base_date` för att identifiera vilken sjudagarsperiod datumet ligger inom. Veckans startdag baseras på systemvariabeln `FirstWeekDay`. Du kan även ändra den första veckodagen genom att använda argumentet `first_week_day` i `inweek()`-funktionen.

När den valda veckan har definierats returnerar funktionen booleska resultat när du jämför de föreskrivna datumvärdena med det veckosegmentet.

Användning

Funktionen `InWeek` returnerar ett booleskt resultat. Vanligtvis kommer denna typ av funktion att användas som ett villkor i en `if expression`. `InWeek`-funktionen returnerar en aggregering eller beräkning som beror på om ett utvärderat datum ligger i veckan med det valda datumet i `base_date` argument.

Till exempel kan `InWeek`-funktionen användas för att identifiera all utrustning som tillverkas under en specifik vecka.

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera veckan.
period_no	Veckans startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger den vecka som innehåller **base_date**. Negativa värden i **period_no** anger föregående veckor och positiva värden anger efterföljande veckor.
first_week_day	Som standard är den första dagen i veckan söndag (vilket bestäms av systemvariabeln FirstWeekDay), med start vid midnatt mellan lördag och söndag. **first_week_day**-parametern ersätter **FirstWeekDay**-variabeln. Om du vill att veckan ska starta en annan dag anger du en flagga mellan 0 och 6.

first_week_day values

Dag	Värde
Måndag	0
Tisdag	1

Dag	Värde
Onsdag	2
Torsdag	3
Fredag	4
Lördag	5
Söndag	6

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`inweek ('01/12/2006', '01/14/2006', 0)`	Returnerar TRUE
`inweek ('01/12/2006', '01/20/2006', 0 )`	Returnerar FALSE
`inweek ('01/12/2006', '01/14/2006', -1 )`	Returnerar FALSE
`inweek ('01/07/2006', '01/14/2006', -1)`	Returnerar TRUE
`inweek ('01/12/2006', '01/09/2006', 0, 3)`	Returnerar FALSE eftersom `first_week_day` är angett som 3 (torsdag), vilket gör 2006-01-12 till den första dagen i veckan efter veckan som innehåller 2006-01-09.

Dessa ämnen kan hjälpa dig att arbeta med den här funktionen:

Relaterade ämnen

Avsnitt	Standardflagga/värde	Beskrivning
FirstWeekDay (page 257)	6/söndag	Definierar startdagen för varje vecka.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för månaden januari 2022 som laddas i en tabell som heter `Transactions`.
- Systemvariabeln `FirstWeekDay` som är inställd till `6` (söndag).
- En föregående laddning som innehåller följande:
  - `inweek()`-funktionen, inställd som fältet `in_week`, som fastställer vilka transaktioner som ägde rum under kvartalet fram till den 14 januari 2022.
  - `weekday()`-funktionen, inställd som fältet `week_day`, som visar vilken dag i veckan som motsvarar varje datum.

Laddningsskript

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       weekday(date) as week_day,
       inweek(date,'01/14/2022', 0) as in_week
       ;
Load
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
```

```
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `week_day`
- `in_week`

Resultattabell

datum	week_day	i veckan
01/02/2022	sön	0
01/05/2022	ons	0
01/06/2022	tors	0
2022-01-08	lör	0
09/01/2022	sön	-1
01/10/2022	mån	-1
11/01/2022	tis	-1
12/01/2022	ons	-1
13/01/2022	tors	-1
14/01/2022	fre	-1
15/01/2022	lör	-1
16/01/2022	sön	0
17/01/2022	mån	0
18/01/2022	tis	0
26/01/2022	ons	0
27/01/2022	tors	0
28/01/2022	fre	0
29/01/2022	lör	0
30/01/2022	sön	0
01/31/2022	mån	0

in_week-fältet skapas i föregående LOAD-sats med hjälp av inweek()-funktionen. Det första argumentet anger vilket fält som utvärderas. Det andra argumentet är ett hårdkodat datum för 14 januari, som är base_date. base_date-argumentet fungerar med FirstWeekDay-systemvariabeln för att identifiera jämförelseveckan. Ett period_no på 0 – vilket betyder att funktionen inte jämför veckorna före eller efter den segmenterade veckan – är det slutliga argumentet.

Systemvariabeln FirstWeekDay bestämmer att veckor börjar på en söndag och slutar på en lördag. Därför skulle januari delas upp i veckor enligt diagrammet nedan, där datumen mellan 9 och 15 januari ger den giltiga perioden för inweek()-beräkningen:

Diagram för kalender med inweek()-funktionens intervall framhävt

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Alla transaktioner som inträffar mellan 9 och 15 januari returnerar det booleska resultatet TRUE.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning som innehåller en uppsättning transaktioner för 2022 laddas i en tabell som heter `Transactions`.
- Systemvariabeln `FirstWeekDay` som är inställd till `6` (söndag).
- En föregående laddning som innehåller följande:
  - `inweek ()`-funktionen, inställd som fältet `prev_week`, som fastställer vilka transaktioner som ägde rum en hel vecka före veckan med 14 januari 2022.
  - `weekday()`-funktionen, inställd som fältet `week_day`, som visar vilken dag i veckan som motsvarar varje datum.

Laddningsskript

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       weekday(date) as week_day,
       inweek(date,'01/14/2022', -1) as prev_week
       ;
Load
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

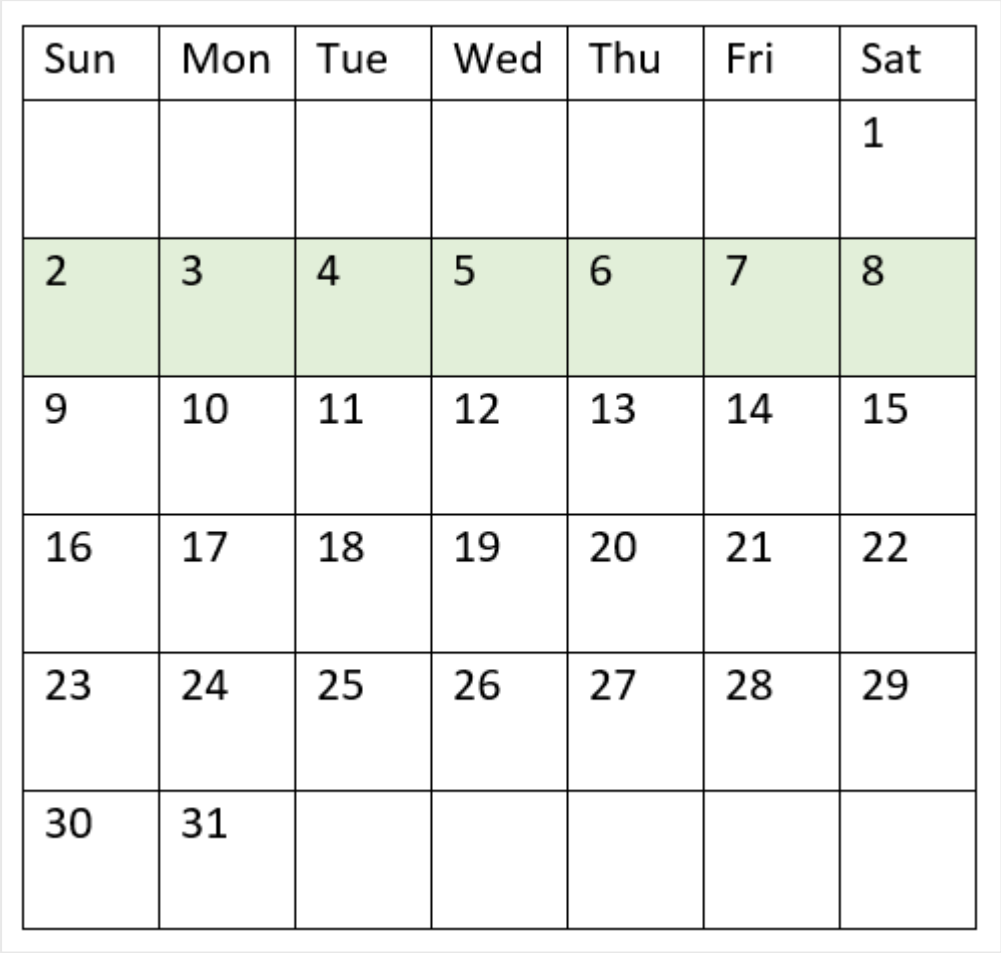
- `date`
- `week_day`
- `prev_week`

Resultattabell

datum	week_day	förra_veckan
01/02/2022	sön	-1
01/05/2022	ons	-1
01/06/2022	tors	-1
2022-01-08	lör	-1
2022-01-09	sön	0
10/01/2022	mån	0
11/01/2022	tis	0
12/01/2022	ons	0
13/01/2022	tors	0
14/01/2022	fre	0
15/01/2022	lör	0
2022-01-16	sön	0
17/01/2022	mån	0
18/01/2022	tis	0
26/01/2022	ons	0
27/01/2022	tors	0
28/01/2022	fre	0
29/01/2022	lör	0
30/01/2022	sön	0
01/31/2022	mån	0

Genom att använda `-1` som `period_no`-argument i `inweek()`-funktionen skiftas gränserna för i jämförelseveckan bakåt med sju hela dagar. Med ett `period_no` på `0` skulle veckan ligga mellan 9 och 15 januari. Men i det här exemplet skiftas `period_no` i `-1` start- och slutgränsen för det här segmentet bakåt med en vecka. Datumgränserna blir 2 januari till 8 januari.

Diagram för kalender med `inweek()`-funktionens intervall framhävt



Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Därför kommer alla transaktioner som sker mellan 2 och 8 januari att returnera det booleska resultatet TRUE.

Exempel 3 – first_week_day

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning som innehåller en uppsättning transaktioner för 2022 laddas i en tabell som heter `Transactions`.
- Systemvariabeln `FirstWeekDay` som är inställd till `6` (söndag).
- En föregående laddning som innehåller följande:
  - `inweek()`-funktionen, inställd som fältet `in_week`, som fastställer vilka transaktioner som ägde rum under kvartalet fram till den 14 januari 2022.

- `weekday()`-funktionen, inställd som fältet `week_day`, som visar vilken dag i veckan som motsvarar varje datum.

Laddningsskript

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       weekday(date) as week_day,
       inweek(date,'01/14/2022', 0, 0) as in_week
       ;
Load
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `week_day`
- `in_week`

Resultattabell

datum	week_day	i veckan
01/02/2022	sön	0
01/05/2022	ons	0
01/06/2022	tors	0
2022-01-08	lör	0
09/01/2022	sön	0
10/01/2022	mån	-1
11/01/2022	tis	-1
12/01/2022	ons	-1
13/01/2022	tors	-1
14/01/2022	fre	-1
15/01/2022	lör	-1
16/01/2022	sön	-1
17/01/2022	mån	0
18/01/2022	tis	0
26/01/2022	ons	0
27/01/2022	tors	0
28/01/2022	fre	0
29/01/2022	lör	0
30/01/2022	sön	0
01/31/2022	mån	0

Genom att använda `0` som `first_week_day`-argument i `inweek()`-funktionen ersätts systemvariabeln `FirstWeekDay` och anger måndag som den första dagen i veckan.

Diagram för kalender med `inweek()`-funktionens intervall framhävt

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Därför kommer alla transaktioner som sker mellan 10 och 16 januari att returnera det booleska resultatet TRUE

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är dock datauppsättningen oförändrad och har lästs in i programmet. Skapa ett mått i resultattabellen för att fastställa vilka transaktioner som ägde rum under veckan med 14 januari 2022.

Laddningsskript

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
```

```
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `date`

Skapa följande mått:

- `=inweek (date,'01/14/2022',0)`, för att beräkna huruvida transaktioner ägde rum under samma vecka som 14 januari.
- `=weekday(date)`, för att visa vilken veckodag som motsvarar varje datum.

Resultattabell

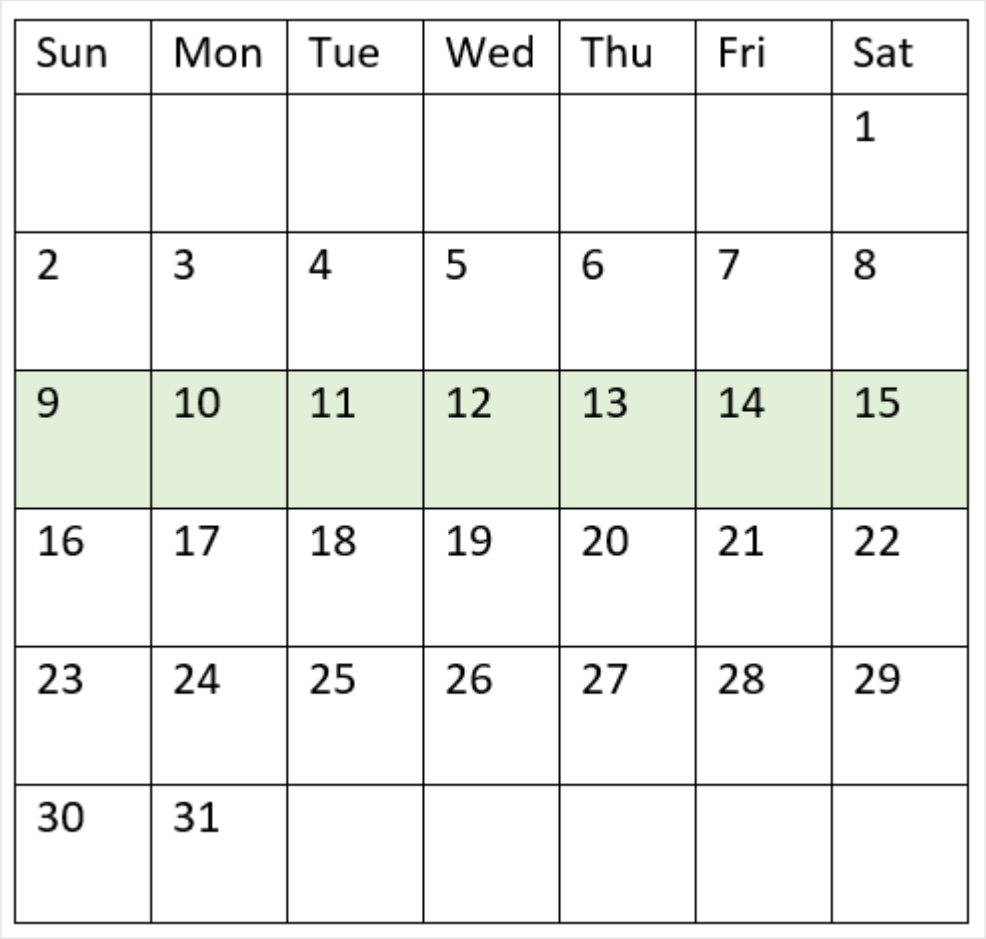
datum	week_day	=inweek (date,'01/14/2022',0)
01/02/2022	sön	0
01/05/2022	ons	0
01/06/2022	tors	0
2022-01-08	lör	0
09/01/2022	sön	-1
01/10/2022	mån	-1
11/01/2022	tis	-1

datum	week_day	=inweek (date,'01/14/2022',0)
12/01/2022	ons	-1
13/01/2022	tors	-1
14/01/2022	fre	-1
15/01/2022	lör	-1
16/01/2022	sön	0
17/01/2022	mån	0
18/01/2022	tis	0
26/01/2022	ons	0
27/01/2022	tors	0
28/01/2022	fre	0
29/01/2022	lör	0
30/01/2022	sön	0
01/31/2022	mån	0

`in_week`-måttet skapas i diagrammet med hjälp av `inweek()`-funktionen. Det första argumentet anger vilket fält som utvärderas. Det andra argumentet är ett hårdkodat datum för 14 januari, som är `base_date`. `base_date`-argumentet fungerar med `FirstWeekDay`-systemvariabeln för att identifiera jämförelseveckan. Ett `period_no` på `0` är det sista argumentet.

Systemvariabeln `FirstWeekDay` bestämmer att veckor börjar på en söndag och slutar på en lördag. Därför skulle januari delas upp i veckor enligt diagrammet nedan, där datumen mellan 9 och 15 januari ger den giltiga perioden för `inweek()`-beräkningen:

Diagram för kalender med `inweek()`-funktionens intervall framhävt



Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Alla transaktioner som inträffar mellan 9 och 15 januari returnerar det booleska resultatet `TRUE`.

Exempel 5 – Scenario

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning laddas i en tabell som heter `Products`.
- Tabellen innehåller följande fält:
  - produkt-ID
  - produkttyp
  - tillverkningsdatum
  - kostnadspris

I det här exemplet har man identifierat att utrustningsfel var orsaken till att produkter som tillverkades den 12 januari var defekta. Slutanvändaren vill ha en diagram som visar, efter vecka, status för vilka produkter som tillverkades som var "defekta" eller "felfria", samt kostnaden för de produkter som tillverkades den veckan.

Laddningsskript

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `=weekname(manufacture_date)`

Skapa följande mått:

- `=if(only(inweek(manufacture_date,makedate(2022,01,12),0)),'Defective','Faultless')`, för att identifiera vilka av produkterna som är defekta och vilka som är felfria med `inweek()`-funktionen:
- `=sum(cost_price)`, för att visa den sammanlagda kostnaden för varje produkt.

Gör följande:

1. Ange måttens **Nummerformatering** till **Pengar**.
2. Under **Utseende**, stänger du av **Total**.

Resultattabell

veckonamn (manufacture_date)	=om(endast(ivecka(manufacture_date,skapadatum (2022,01,12),0)), 'Defekt','Felfri')	=sum(cost_ price)
2022/02	Felfri	200.09
2022/03	Defekt	441.51
2022/04	Felfri	178.41
2022/05	Felfri	231.67
2022/06	Felfri	163,91

Funktionen `inweek()` returnerar ett booleskt värde vid utvärdering av tillverkningsdatumen för var och en av produkterna. För alla produkter som tillverkats under veckan med 12 januari returnerar funktionen `inweek()` det booleska värdet TRUE och markerar produkterna som "Defekta". För alla produkter som returnerar värdet FALSE och därför inte tillverkats under den veckan, markerar den produkterna som "Felfria".

inweektodate

Denna funktion returnerar True om **timestamp** ligger inom den del av veckan som innehåller **base_date** fram till och inklusive den sista millisekunden av **base_date**.

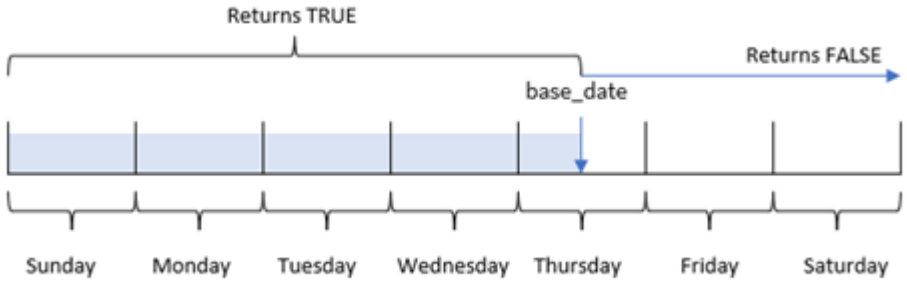
Syntax:

```
InWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Returnerad datatyp: Booleansk

> *I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.*

Diagram för inquartertodate-funktionen



`inweektodate()`-funktionen använder `base_date`-parametern för att identifiera ett maximalt gränsdatum för ett veckosegment, såväl som dess motsvarande datum för veckans början, vilket är baserat på

systemvariabeln `FirstWeekDay` (eller användardefinierad `first_week_day`-parameter). När detta veckosegment har definierats returnerar funktionen ett booleskt resultat när du jämför de föreskrivna datumvärdena med det segmentet.

Användning

Funktionen `inweektodate()` returnerar ett booleskt resultat. Vanligtvis kommer denna typ av funktion att användas som ett villkor i ett `if`-uttryck. Detta returnerar en aggregering eller beräkning beroende på om ett utvärderat datum inträffade i veckan till och med ett specifikt datum.

`inweektodate()`-funktionen kan till exempel användas för att beräkna alla försäljningar gjorda under en angiven vecka fram till ett visst datum.

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera veckan.
period_no	Veckans startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger den vecka som innehåller **base_date**. Negativa värden i **period_no** anger föregående veckor och positiva värden anger efterföljande veckor.
first_week_day	Som standard är den första dagen i veckan söndag (vilket bestäms av systemvariabeln FirstWeekDay), med start vid midnatt mellan lördag och söndag. **first_week_day**-parametern ersätter **FirstWeekDay**-variabeln. Om du vill att veckan ska starta en annan dag anger du en flagga mellan 0 och 6. För en vecka som börjar på måndag och slutar på söndag, använd flaggan 0 för måndag, 1 för tisdag, 2 för onsdag, 3 för torsdag, 4 för fredag, 5 för lördag och 6 för söndag.

Exempel på funktioner

Exempel	Interaktion
`inweektodate ('01/12/2006', '01/12/2006', 0)`	Returnerar `TRUE`.
`inweektodate ('01/12/2006', '01/11/2006', 0)`	Returnerar `FALSE`.
`inweektodate ('01/12/2006', '01/18/2006', -1)`	Returnerar `FALSE`. Eftersom `period_no` är angett som -1, är de faktiska data som `timestamp` jämförs med 01/11/2006.
`inweektodate ('01/11/2006', '01/12/2006', 0, 3 )`	Returnerar `FALSE`, eftersom `first_week_day` har angetts som `3` (torsdag), vilket gör 01/12/2006 till den första dagen i veckan efter veckan som innehåller 01/12/2006.

Dessa ämnen kan hjälpa dig att arbeta med den här funktionen:

Relaterade ämnen

Avsnitt	Standardflagga/värde	Beskrivning
FirstWeekDay (page 257)	6/söndag	Definierar startdagen för varje vecka.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för månaden januari 2022 som laddas i en tabell som heter `Transactions`.
- Datafältet tillhandahålls i `TimestampFormat='M/D/YYYY h:mm:ss[.fff]'`-formatet.
- Skapandet av ett fält, `in_week_to_date`, som bestämmer vilka transaktioner som ägde rum under kvartalet fram till den 14 januari 2022.
- Skapande av ett ytterligare fält benämnt `weekday`, med användning av `weekday()`-funktionen. Detta nya fält skapas för att visa vilken veckodag som motsvarar varje datum.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';
SET FirstWeekDay=6;
Transactions:
    Load
        *,
        weekday(date) as week_day,
        inweektodate(date,'01/14/2022', 0) as in_week_to_date
        ;
Load
```

```
*
Inline
[
id,date,amount
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `week_day`
- `in_week_to_date`

Resultattabell

datum	week_day	hittills under veckan
2022-01-02 12:22:06	sön	0
2022-01-05 01:02:30	ons	0
2022-01-06 15:36:20	tors	0
2022-01-08 10:58:35	lör	0
2022-01-09 08:53:32	sön	-1
2022-01-10 21:13:01	mån	-1
2022-01-11 00:57:13	tis	-1
2022-01-12 09:26:02	ons	-1
2022-01-13 15:05:09	tors	-1

datum	week_day	hittills under veckan
2022-01-14 18:44:57	fre	-1
2022-01-15 06:10:46	lör	0
2022-01-16 06:39:27	sön	0
2022-01-17 10:44:16	mån	0
2022-01-18 18:48:17	tis	0
2022-01-26 04:36:03	ons	0
2022-01-27 08:07:49	tors	0
2022-01-28 12:24:29	fre	0
2022-01-30 11:56:56	sön	0
2022-01-30 14:40:19	sön	0
2022-01-31 05:28:21	mån	0

`in_week_to_date`-fältet skapas i föregående LOAD-sats med hjälp av funktionen `inweektodate()`. Det första argumentet som anges identifierar vilket fält som utvärderas. Det andra argumentet är ett hårdkodat datum för den 14 januari, vilket är `base_date` som identifierar vilken vecka som ska segmenteras samt definierar slutgränsen för det segmentet. `period_no`-värdet 0 är det sista argumentet, vilket betyder att funktionen inte jämför veckorna före eller efter den segmenterade veckan.

Systemvariabeln `FirstWeekDay` bestämmer att veckor börjar på en söndag och slutar på en lördag. Därför skulle januari delas upp i veckor enligt diagrammet nedan, där datumen mellan 9 och 14 januari ger den giltiga perioden för `inweekdodate()`-beräkningen:

Kalenderdiagram som visar transaktionsdatum som skulle returnera det booleska resultatet TRUE

Sun	Mon	Tue	Wed	Thur	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Alla transaktioner som sker mellan 9 och 14 januari returnerar det booleska resultatet `TRUE`. Transaktioner före och efter datumen ger det booleska resultatet `FALSE`.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Skapandet av ett fält, `prev_week_to_date`, som bestämmer vilka transaktioner som ägde rum en hel vecka innan veckosegmentet som avslutas den 14 januari 2022.
- Skapande av ett ytterligare fält benämnt `weekday`, med användning av `weekday()`-funktionen. Detta nya fält skapas för att visa vilken veckodag som motsvarar varje datum.

Laddningsskript

```
SET FirstWeekDay=6;
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';
Transactions:
    Load
        *,
        weekday(date) as week_day,
        inweektodate(date,'01/14/2022', -1) as prev_week_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `week_day`
- `prev_week_to_date`

Resultattabell

datum	week_day	föregående_vecka_hittills
2022-01-02 12:22:06	sön	-1
2022-01-05 01:02:30	ons	-1
2022-01-06 15:36:20	tors	-1
2022-01-08 10:58:35	lör	0
2022-01-09 08:53:32	sön	0
2022-01-10 21:13:01	mån	0
2022-01-11 00:57:13	tis	0
2022-01-12 09:26:02	ons	0
2022-01-13 15:05:09	tors	0
2022-01-14 18:44:57	fre	0
2022-01-15 06:10:46	lör	0
2022-01-16 06:39:27	sön	0
2022-01-17 10:44:16	mån	0
2022-01-18 18:48:17	tis	0
2022-01-26 04:36:03	ons	0
2022-01-27 08:07:49	tors	0
2022-01-28 12:24:29	fre	0
2022-01-30 11:56:56	sön	0
2022-01-30 14:40:19	sön	0
2022-01-31 05:28:21	mån	0

`period_no`-värdet -1 indikerar att `inweektodate ()`-funktionen jämför det ingående kvartalssegmentet med föregående vecka. Veckosegmentet motsvarar initialt mellan 9 januari och 14 januari. `period_no` förskjuter sedan både start- och slutgränsen för detta segment till en vecka tidigare, vilket gör att datumgränserna blir 2 januari till 7 januari.

Kalenderdiagram som visar transaktionsdatum som skulle returnera det booleska resultatet TRUE

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Därför kommer alla transaktioner som sker mellan 2 och 8 januari (exklusive 8 janiari) att returnera det booleska resultatet är `TRUE`.

Exempel 3 – first_week_day

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Skapandet av ett fält, `in_week_to_date`, som bestämmer vilka transaktioner som ägde rum under kvartalet fram till den 14 januari 2022.
- Skapande av ett ytterligare fält benämnt `weekday`, med användning av `weekday()`-funktionen. Detta nya fält skapas för att visa vilken veckodag som motsvarar varje datum.

I det här exemplet har vi använt måndag som den första dagen i veckan.

Laddningsskript

```
SET FirstWeekDay=6;
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';

Transactions:
    Load
        *,
        weekday(date) as week_day,
        inweektodate(date,'01/14/2022', 0, 0) as in_week_to_date
        ;
Load
*
```

```
Inline
[
id,date,amount
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `week_day`
- `in_week_to_date`

Resultattabell

datum	week_day	hittills under veckan
2022-01-02 12:22:06	sön	0
2022-01-05 01:02:30	ons	0
2022-01-06 15:36:20	tors	0
2022-01-08 10:58:35	lör	0
2022-01-09 08:53:32	sön	0
2022-01-10 21:13:01	mån	-1
2022-01-11 00:57:13	tis	-1
2022-01-12 09:26:02	ons	-1
2022-01-13 15:05:09	tors	-1

datum	week_day	hittills under veckan
2022-01-14 18:44:57	fre	-1
2022-01-15 06:10:46	lör	0
2022-01-16 06:39:27	sön	0
2022-01-17 10:44:16	mån	0
2022-01-18 18:48:17	tis	0
2022-01-26 04:36:03	ons	0
2022-01-27 08:07:49	tors	0
2022-01-28 12:24:29	fre	0
2022-01-30 11:56:56	sön	0
2022-01-30 14:40:19	sön	0
2022-01-31 05:28:21	mån	0

Genom att använda 0 som `first_week_day`-argument i `inweektodate()`-funktionen, ersätter funktionsargumentet systemvariabeln `FirstWeekDay` och anger måndag som den första dagen i veckan.

Kalenderdiagram som visar transaktionsdatum som skulle returnera det booleska resultatet TRUE

Mon	Tue	Wed	Thu	Fri	Sat	Sun
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	17
24	25	26	27	28	29	30
31						

Därför kommer alla transaktioner som sker mellan 1o och 14 januari returnera det booleska resultatet `TRUE`, medan transaktioner med datum utanför dessa gränser kommer att returnera värdet `FALSE`.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet. I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som avgör om transaktioner har skett i veckan fram till den 14 januari 2022 skapas som ett mått i diagramobjektet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.
2. För att beräkna om transaktioner ägde rum under samma vecka fram till den 14 januari skapar du följande åtgärd:
   `=inweektodate(date,'01/14/2022',0)`
3. För att visa vilken veckodag som motsvarar varje datum skapar du ett ytterligare mått:
   `=weekday(date)`

Resultattabell

datum	week_day	hittills under veckan
2022-01-02 12:22:06	sön	0
2022-01-05 01:02:30	ons	0
2022-01-06 15:36:20	tors	0
2022-01-08 10:58:35	lör	0
2022-01-09 08:53:32	sön	-1
2022-01-10 21:13:01	mån	-1
2022-01-11 00:57:13	tis	-1
2022-01-12 09:26:02	ons	-1
2022-01-13 15:05:09	tors	-1
2022-01-14 18:44:57	fre	-1
2022-01-15 06:10:46	lör	0
2022-01-16 06:39:27	sön	0
2022-01-17 10:44:16	mån	0
2022-01-18 18:48:17	tis	0
2022-01-26 04:36:03	ons	0
2022-01-27 08:07:49	tors	0
2022-01-28 12:24:29	fre	0
2022-01-30 11:56:56	sön	0
2022-01-30 14:40:19	sön	0
2022-01-31 05:28:21	mån	0

`in_week_to_date`-fältet skapas som ett mått i ett diagramobjekt med hjälp av `inweektodate()`-funktionen. Det första argumentet som anges identifierar vilket fält som utvärderas. Det andra argumentet är ett hårdkodat datum för den 14 januari, vilket är `base_date` som identifierar vilken vecka som ska segmenteras samt definierar slutgränsen för det segmentet. `period_no`-värdet 0 är det sista argumentet, vilket betyder att funktionen inte jämför veckorna före eller efter den segmenterade veckan.

Systemvariabeln `FirstWeekDay` bestämmer att veckor börjar på en söndag och slutar på en lördag. Därför skulle januari delas upp i veckor enligt diagrammet nedan, där datumen mellan 9 och 14 januari ger den giltiga perioden för `inweekdodate()`-beräkningen:

Kalenderdiagram som visar transaktionsdatum som skulle returnera det booleska resultatet TRUE

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Alla transaktioner som sker mellan 9 och 14 januari returnerar det booleska resultatet `TRUE`. Transaktioner före och efter datumen ger det booleska resultatet `FALSE`.

Exempel 5 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning läses in i en tabell som heter `Products`.
- Information om produkt-ID, tillverkningsdatum och självkostnadspris.

I det här exemplet har man identifierat att utrustningsfel var orsaken till att produkter som tillverkades den 12 januari var defekta. Frågan löstes den 13 januari. Slutanvändaren vill ha ett diagramobjekt som visar, efter vecka, status för vilka produkter som tillverkades som var "defekta" eller "felfria", samt kostnaden för de produkter som tillverkades den veckan.

Laddningsskript

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell. Skapa en dimension för att visa veckonamnen.
   `=weekname(manufacture_date)`
2. Skapa sedan en dimension för att identifiera vilka av produkterna som är defekta och vilka som är felfria:
   `=if(inWeektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
3. Skapa ett mått för att summera `cost_price` av produkterna:
   `=sum(cost_price)`
4. Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

veckonamn (tillverkningsdatum)	if(inweektodate(manufacture_date,makedate (2022,01,12),0),'Defekt','Felfri')	=sum(cost_ price)
2022/02	Felfri	$200.09
2022/03	Defekt	$263.46
2022/03	Felfri	$178.05

veckonamn (tillverkningsdatum)	if(inweektodate(manufacture_date,makedate (2022,01,12),0),'Defekt','Felfri')	=sum(cost_ price)
2022/04	Felfri	$178.41
2022/05	Felfri	$147.46
2022/06	Felfri	$248.12

Funktionen `inweektodate()` returnerar ett booleskt värde vid utvärdering av tillverkningsdatumen för var och en av produkterna. För de som returnerar ett booleskt värde på `TRUE`, markerar det produkterna som `'Defective'`. För alla produkter som returnerar värdet `FALSE`, och därför inte tillverkas i veckan fram till och med den 12 januari, markeras produkterna som `'Faultless'`.

inyear

Denna funktion returnerar True om **timestamp** ligger inom det år som innehåller **base_date**.

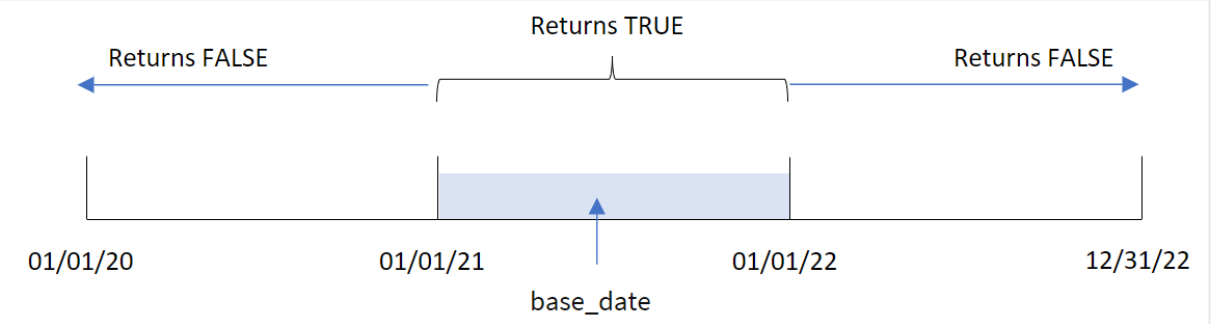
Syntax:

```
InYear (timestamp, base_date, period_no [, first_month_of_year])
```

Returnerad datatyp: Booleansk

I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.

Diagram för `inyear()`-funktionens intervall



`inyear()`-funktionen returnerar ett booleskt resultat när de valda datumvärdena jämförs med ett år som definieras av `base_date`.

Användning

Funktionen `inyear()` returnerar ett booleskt resultat. Vanligtvis kommer denna typ av funktion att användas som ett villkor i en `if expression`. Detta returnerar en aggregering eller beräkning beroende på om ett utvärderat datum inträffade under året i fråga. `inyear()`-funktionen kan exempelvis användas för att identifiera all försäljning som inträffade under ett definierat år.

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera året.
period_no	Årets startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger året som innehåller **base_date**. Negativa värden i **period_no** anger föregående år och positiva värden anger efterföljande år.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Du kan använda följande värden för att ställa in den första månaden på året i argumentet `first_month_of_year`:

first_month_of_year values

Månad	Värde
februari	2
mars	3
april	4
Maj	5
juni	6
juli	7
augusti	8
september	9
oktober	10
november	11
december	12

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`inyear ('01/25/2013', '01/01/2013', 0 )`	Returnerar TRUE
`inyear ('01/25/2012', '01/01/2013', 0)`	Returnerar FALSE
`inyear ('01/25/2013', '01/01/2013', -1)`	Returnerar FALSE
`inyear ('01/25/2012', '01/01/2013', -1 )`	Returnerar TRUE
`inyear ('01/25/2013', '01/01/2013', 0, 3)`	Returnerar TRUE Värdet för `base_date` och `first_month_of_year` anger att tidsmarkören måste infalla mellan 2012-01-03 och 2013-02-28
`inyear ('03/25/2013', '07/01/2013', 0, 3 )`	Returnerar TRUE

Exempel 1 – Grundläggande exempel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner mellan 2020 och 2022 som laddas i en tabell som heter `Transactions`.
- En föregående laddning som innehåller `inyear()`-funktionen som ställs in som `in_year`-fältet och avgör vilka transaktioner som ägde rum under samma år som 26 juli 2021.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
Transactions:
    Load
        *,
        inyear(date,'07/26/2021', 0) as in_year
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
```

```
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_year`

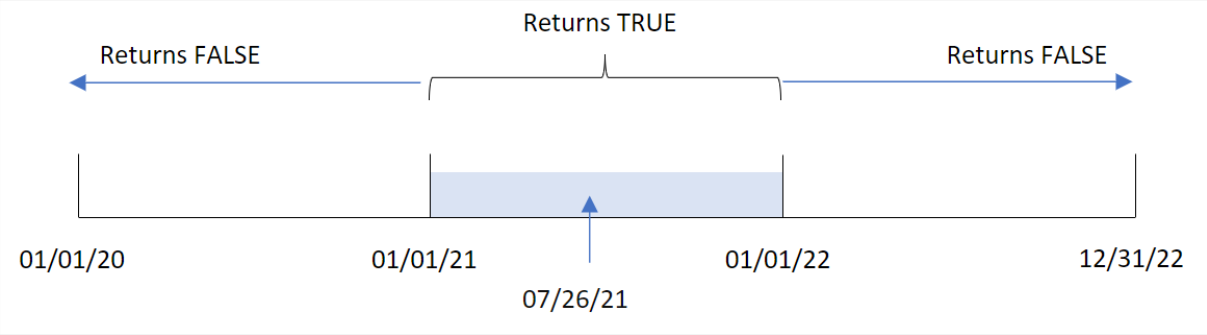
Resultattabell

datum	in_year
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1

datum	in_year
07/26/2021	-1
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

`in_year`-fältet skapas i föregående LOAD-sats med hjälp av `inyear()`-funktionen. Det första argumentet anger vilket fält som utvärderas. Det andra argumentet är ett hårdkodat datum för den 26 juli 2021, vilket är `base_date` som fastställer jämförelseåret. Ett `period_no` på `0` är det sista argumentet, vilket betyder att `inyear()`-funktionen inte jämför åren före eller efter året.

Diagram för `inyear()`-funktionens intervall med 26 juli som basdatum



Alla transaktioner som inträffar under 2021 returnerar det booleska värdet TRUE.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner mellan 2020 och 2022 som laddas i en tabell som heter `Transactions`.
- En föregående laddning som innehåller `inyear()`-funktionen som ställs in som `previous_year`-fältet och avgör vilka transaktioner som ägde rum under året före det år som innehåller 26 juli 2021.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
Transactions:
```

```
    Load
        *,
        inyear(date,'07/26/2021', -1) as previous_year
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `previous_year`

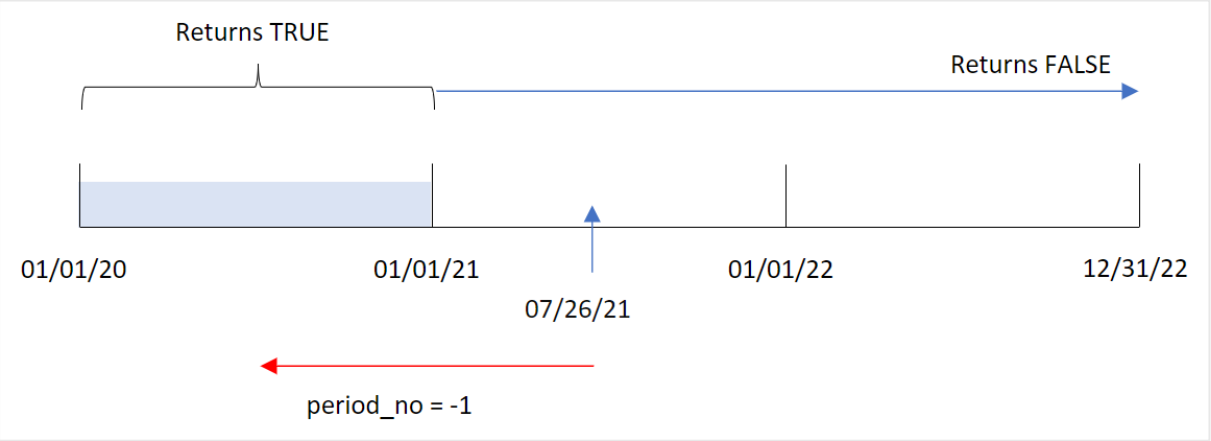
Resultattabell

datum	previous_year
01/13/2020	-1
02/26/2020	-1
03/27/2020	-1
04/16/2020	-1
05/21/2020	-1
08/14/2020	-1
10/07/2020	-1

datum	previous_year
12/05/2020	-1
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Genom att använda `-1` som `period_no`-argument i `inyear()`-funktionen skiftar gränserna för jämförelseåret bakåt ett helt år. 2021 identifieras initialt som jämförelseår. `period_no` förskjuter jämförelseåret ett år, så att 2020 blir jämförelseår.

Diagram för `inyear()`-funktionens intervall med argumentet för period_no inställt till `-1`



Alla transaktioner som inträffar under 2020 returnerar därför det booleska värdet TRUE.

Exempel 3 – first_month_of_year

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner mellan 2020 och 2022 som laddas i en tabell som heter `Transactions`.
- En föregående laddning som innehåller `inyear()`-funktionen som ställs in som `in_year`-fältet och avgör vilka transaktioner som ägde rum under samma år som 26 juli 2021.

I det här exemplet är dock organisationspolicyn att mars är den första månaden i räkenskapsåret.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
Transactions:
    Load
        *,
        inyear(date,'07/26/2021', 0, 3) as in_year
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

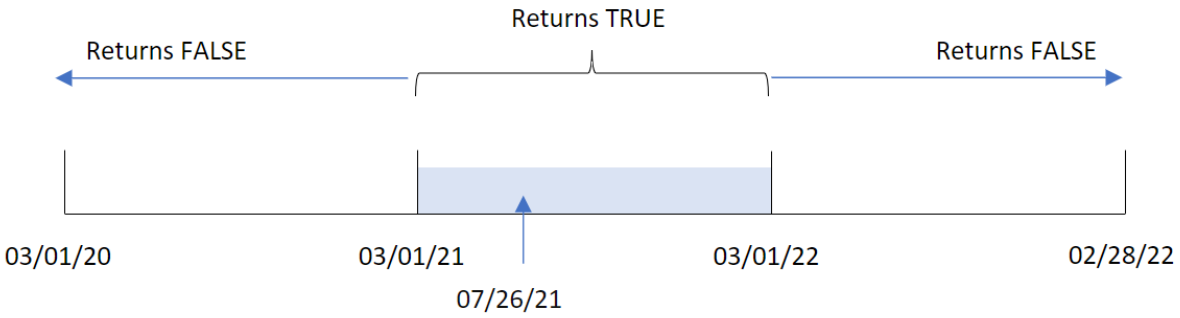
- `date`
- `in_year`

Resultattabell

datum	in_year
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Genom att använda `3` som `first_month_of_year`-argument i `inyear()`-funktionen börjar året 1 mars och slutar året i slutet av februari.

Diagram för `inyear()`-funktionens intervall med mars inställt som årets första månad.



Därför kommer alla transaktioner som sker mellan 1 mars 2021 och 1 mars 2022 returnera det booleska resultatet TRUE.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är dock datauppsättningen oförändrad och har lästs in i programmet. Beräkningen som avgör om transaktioner har skett under samma år som 26 juli 2021 skapas som ett mått i ett diagramobjekt i programmet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
```

```
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `date`

För att beräkna om transaktioner ägde rum under samma år som 26 juli 2021 skapar du följande mått:

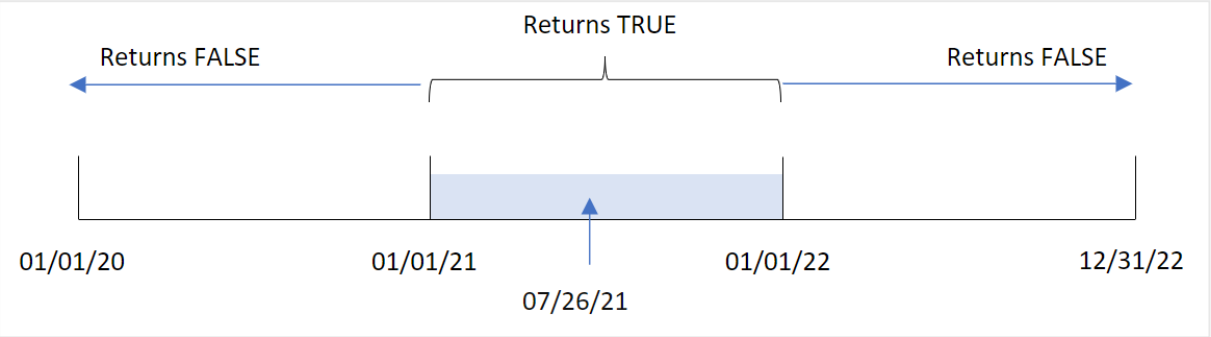
- `=inyear(date,'07/26/2021', 0)`

Resultattabell

datum	=inyear(date,'07/26/2021',0)
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

`in_year`-fältet skapas i diagrammet genom att använda `inyear()`-funktionen. Det första argumentet anger vilket fält som utvärderas. Det andra argumentet är ett hårdkodat datum för den 26 juli 2021, vilket är `base_date` som fastställer jämförelseåret. Ett `period_no` på `0` är det sista argumentet, vilket betyder att `inyear()`-funktionen inte jämför åren före eller efter året.

Diagram för `inyear()`-funktionens intervall med 27 juli som basdatum



Alla transaktioner som inträffar under 2021 returnerar det booleska värdet TRUE.

Exempel 5 – Scenario

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning laddas i en tabell som heter `Products`.
- Tabellen innehåller följande fält:
  - produkt-ID
  - produkttyp
  - tillverkningsdatum
  - kostnadspris

Slutanvändaren vill ha ett diagramobjekt som visar kostnaden för produkterna som tillverkades under 2021, per produkttyp.

Laddningsskript

```
Products:
Load
*
Inline
[
product_id,product_type,manufacture_date,cost_price
8188,product A,'01/13/2020',37.23
8189,product B,'02/26/2020',17.17
8190,product B,'03/27/2020',88.27
```

```
8191,product C,'04/16/2020',57.42
8192,product D,'05/21/2020',53.80
8193,product D,'08/14/2020',82.06
8194,product C,'10/07/2020',40.39
8195,product B,'12/05/2020',87.21
8196,product A,'01/22/2021',95.93
8197,product B,'02/03/2021',45.89
8198,product C,'03/17/2021',36.23
8199,product C,'04/23/2021',25.66
8200,product B,'05/04/2021',82.77
8201,product D,'06/30/2021',69.98
8202,product D,'07/26/2021',76.11
8203,product D,'12/27/2021',25.12
8204,product C,'06/06/2022',46.23
8205,product C,'07/18/2022',84.21
8206,product A,'11/14/2022',96.24
8207,product B,'12/12/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `product_type`

Skapa följande mått för att beräkna summan av varje produkt som tillverkades år 2021:

- `=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))`

Gör följande:

1. Ange måttens **Nummerformatering** till **Pengar**.
2. Under **Utseende**, stänger du av **Total**.

Resultattabell

product_type	=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))
produkt A	$95.93
produkt B	$128.66
produkt C	$61.89
produkt D	$171.21

Funktionen `inyear()` returnerar ett booleskt värde vid utvärdering av tillverkningsdatumen för var och en av produkterna. För alla produkter som tillverkats år 2021 returnerar `inyear()`-funktionen det booleska värdet TRUE och visar summan för `cost_price`.

inyeartodate

Denna funktion returnerar True om **timestamp** ligger inom den del av året som innehåller **base_date** fram till och inklusive den sista millisekunden av **base_date**.

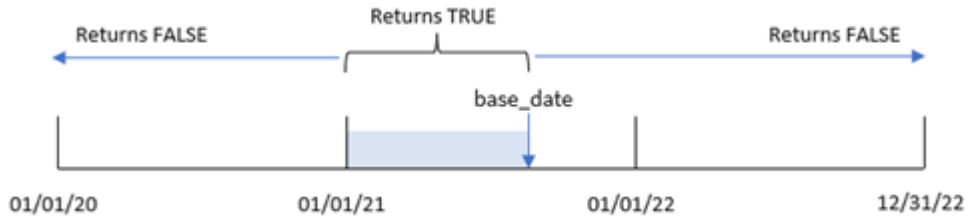
Syntax:

```
InYearToDate (timestamp, base_date, period_no[, first_month_of_year])
```

Returnerad datatyp: Booleansk

> *I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.*

Diagram för inyeartodate-funktionen



`inyeartodate()`-funktionen kommer sedan att segmentera en viss del av året med `base_date` som identifierar både kvartal och det högsta tillåtna datumet för det kvartalssegmentet. Funktionen utvärderar sedan om ett datumfält eller värde faller in i detta segment och returnerar ett booleskt resultat.

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera året.
period_no	Årets startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger året som innehåller **base_date**. Negativa värden i **period_no** anger föregående år och positiva värden anger efterföljande år.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Användning

Funktionen `inyeartodate()` returnerar ett booleskt resultat. Vanligtvis kommer denna typ av funktion att användas som ett villkor i ett `if`-uttryck. Detta returnerar en aggregering eller beräkning beroende på om ett utvärderat datum inträffade i kvartalet till och med datumet i fråga.

Till exempel kan `inyeartodate()`-funktionen användas för att identifiera all utrustning som tillverkats under ett år fram till ett specifikt datum.

I de här exemplen används datumformatet MM/DD/ÅÅÅÅ. Datumformatet anges i `SET DateFormat`-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Exempel på funktioner

Exempel	Resultat
`inyeartodate ('01/25/2013', '02/01/2013', 0)`	Returnerar `TRUE`.
`inyeartodate ('01/25/2012', '01/01/2013', 0)`	Returnerar `FALSE`.
`inyeartodate ('01/25/2012', '02/01/2013', -1)`	Returnerar `TRUE`.
`inyeartodate ('11/25/2012', '01/31/2013', 0, 4)`	Returnerar `TRUE`. Värdet för `timestamp` infaller under budgetåret som börjar i den fjärde månaden och innan värdet för `base_date`.
`inyeartodate ('3/31/2013', '01/31/2013', 0, 4 )`	Returnerar `FALSE`. Jämfört med det förra exemplet är värdet för `timestamp` fortfarande inom budgetåret, men efter datumet för `base_date`, så det infaller utanför delen av året.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner mellan 2020 och 2022 som laddas i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i systemvariabeln `DateFormat`.

- Skapandet av ett fält, `in_year_to_date`, som bestämmer vilka transaktioner som ägde rum under kvartalet fram till den 26 juli 2021.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inyeartodate(date,'07/26/2021', 0) as in_year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_year_to_date`

Resultattabell

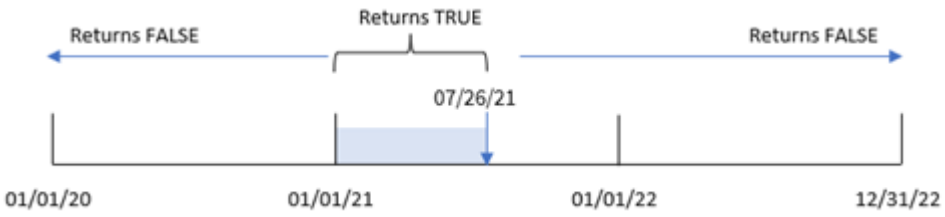
datum	in_year_to_date
01/13/2020	0
02/26/2020	0

datum	in_year_to_date
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

`in_year_to_date`-fältet skapas i föregående LOAD-sats med hjälp av funktionen `inyeartodate()`. Det första argumentet som anges identifierar vilket fält som utvärderas.

Det andra argumentet är ett hårdkodat datum för den 26 juli 2021, vilket är `base_date` som identifierar vilken vecka som ska segmenteras samt definierar slutgränsen för det segmentet. `period_no`-värdet 0 är det sista argumentet, vilket betyder att funktionen inte jämför åren före eller efter det segmenterade året.

Diagram för inyeartodate -funktionen, inga ytterligare argument



Alla transaktioner som sker mellan 1 januari och 26 juli returnerar det booleska resultatet `TRUE`. Transaktionsdatum före 2021 och efter den 26 juli 2021 returnerar `FALSE`.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Skapandet av ett fält, `previous_year_to_date`, som bestämmer vilka transaktioner som ägde rum ett helt år innan årssegmentet som avslutas den 26 juli 2021.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inyeartodate(date,'07/26/2021', -1) as previous_year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

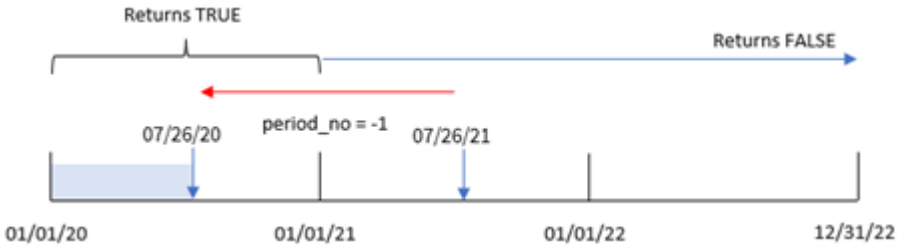
- `date`
- `previous_year_to_date`

Resultattabell

datum	previous_year_to_date
01/13/2020	-1
02/26/2020	-1
03/27/2020	-1
04/16/2020	-1
05/21/2020	-1
06/14/2020	-1
08/07/2020	0
09/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

`period_no`-värdet -1 indikerar att `inyeartodate ()`-funktionen jämför det ingående kvartalssegmentet med föregående året. Med ett inmatningsdatum 26 juli 2021 identifierades segmentet från 1 januari 2021 till 26 juli 2021 initialt som år till datum. `period_no` skjuter sedan bak detta segment till tre månader tidigare, vilket gör att datumgränserna blir 1 januari till 26 juli 2020.

Exempel på diagram för inyeartodate -funktionen, period_no



Därför kommer alla transaktioner som sker mellan 1 januari och 26 juli 2020 returnera ett booleskt resultat som är `TRUE`.

Exempel 3 – first_month_of_year

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Skapandet av ett fält, `in_year_to_date`, som bestämmer vilka transaktioner som ägde rum samma år fram till 26 juli.

I det här exemplet anger vi mars som den första månaden av räkenskapsåret.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inyeartodate(date,'07/26/2021', 0,3) as in_year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
```

```
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `in_year_to_date`

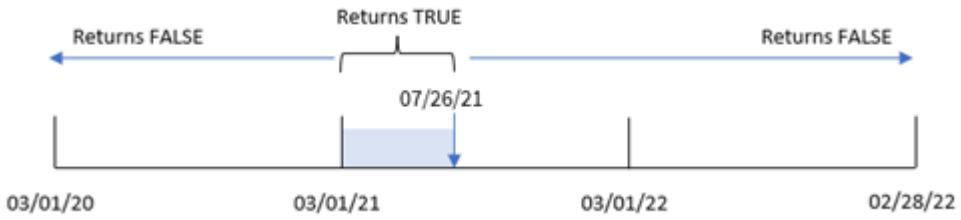
Resultattabell

datum	in_year_to_date
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
07/27/2021	0
06/06/2022	0

datum	in_year_to_date
07/18/2022	0
11/14/2022	0
12/12/2022	0

Genom att använda 3 som `first_month_of_year`-argument i `inyeartodate()`-funktionen, börjar funktionen året den 1 mars. `base_date` den 26 juli 2021 anger sedan slutdatumet för det årsegmentet.

Exempel på diagram för inyeartodate-funktionen, first_month_of_year



Därför kommer alla transaktioner som sker mellan 1 mars och 15 maj att returnera det booleska resultatet `TRUE`, medan transaktioner med datum utanför dessa gränser kommer att returnera värdet `FALSE`.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet. I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som avgör om transaktioner har skett mellan 1 juli och 26 juli skapas som ett mått i ett diagramobjekt i programmet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
```

```
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:`date`.

Skapa följande mått:

```
=inyeartodate(date,'07/26/2021', 0)
```

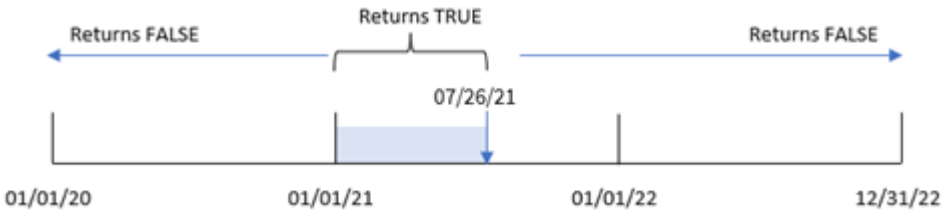
Resultattabell

datum	=inyeartodate(date,'07/26/2021', 0)
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1

datum	=inyeartodate(date,'07/26/2021', 0)
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

`in_year_to_date`-måttet skapas i ett diagramobjekt med hjälp av `inyeartodate()`-funktionen. Det första argumentet som anges identifierar vilket fält som utvärderas. Det andra argumentet är ett hårdkodat datum för den 26 juli 2021, vilket är `base_date` som identifierar vilken vecka som ska segmenteras samt definierar slutgränsen för det segmentet. `period_no`-värdet 0 är det sista argumentet, vilket betyder att funktionen inte jämför åren före eller efter det segmenterade året.

Exempel på diagram för inyeartodate-funktionen, diagramobjekt



Alla transaktioner som sker mellan 1 januari och 26 juli 2021 returnerar det booleska resultatet `TRUE`. Transaktionsdatum före 2021 och efter den 26 juli 2021 returnerar `FALSE`.

Exempel 5 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning läses in i en tabell som heter `Products`.
- Information om produkt-ID, produkttyp, tillverkningsdatum och självkostnadspris.

Slutanvändaren vill ha ett diagramobjekt som visar, per produkttyp, kostnaden för produkterna tillverkade år 2021 fram till den 26 juli.

Laddningsskript

```
Products:
Load
*
```

```
Inline
[
product_id,product_type,manufacture_date,cost_price
8188,product A,'01/13/2020',37.23
8189,product B,'02/26/2020',17.17
8190,product B,'03/27/2020',88.27
8191,product C,'04/16/2020',57.42
8192,product D,'05/21/2020',53.80
8193,product D,'08/14/2020',82.06
8194,product C,'10/07/2020',40.39
8195,product B,'12/05/2020',87.21
8196,product A,'01/22/2021',95.93
8197,product B,'02/03/2021',45.89
8198,product C,'03/17/2021',36.23
8199,product C,'04/23/2021',25.66
8200,product B,'05/04/2021',82.77
8201,product D,'06/30/2021',69.98
8202,product D,'07/26/2021',76.11
8203,product D,'12/27/2021',25.12
8204,product C,'06/06/2022',46.23
8205,product C,'07/18/2022',84.21
8206,product A,'11/14/2022',96.24
8207,product B,'12/12/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:`product_type`.

Skapa ett mått som beräknar summan av varje produkt som tillverkades år 2021 före den 27 juli:

```
=sum(if(inyeartodate(manufacture_date,makedate(2021,07,26),0),cost_price,0))
```

Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

product_type	=sum(if(inyeartodate(manufacture_date,makedate (2021,07,26),0),cost_price,0))
produkt A	$95.93
produkt B	$128.66
produkt C	$61.89
produkt D	$146.09

Funktionen `inyeartodate()` returnerar ett booleskt värde vid utvärdering av tillverkningsdatumen för var och en av produkterna. För alla produkter som tillverkats år 2021 före den 27 juli returnerar `inyeartodate()`-funktionen det booleska värdet `TRUE` och summerar `cost_price`.

Produkt D är den enda produkten som också tillverkades efter den 26 juli 2021. Posten med `product_ID` 8203 tillverkades den 27 december och kostade $25.12. Därför ingick inte denna kostnad i summan för Produkt D i diagramobjektet.

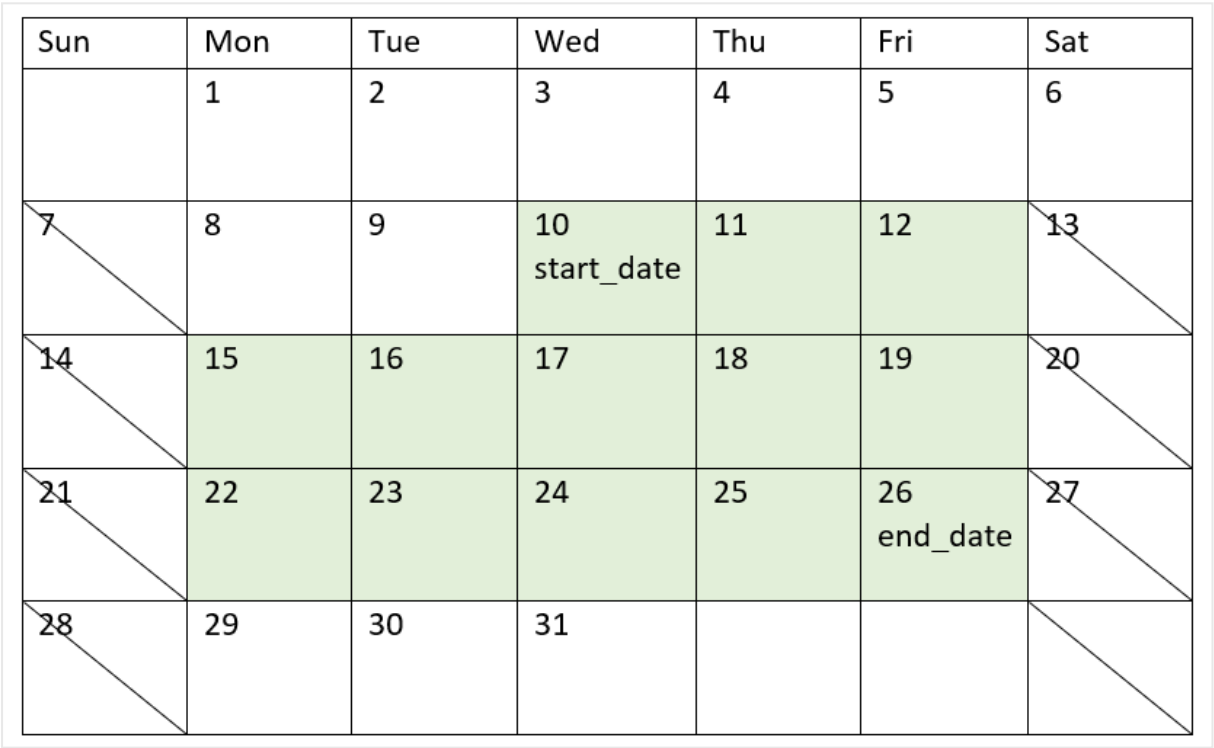
lastworkdate

Funktionen **lastworkdate** returnerar det tidigaste slutdatumet för att uppnå **no_of_workdays** (måndag-fredag) med början vid **start_date** med hänsyn till alla eventuella **holiday**. **start_date** och **holiday** ska vara giltiga datum eller tidsmarkörer.

```
lastworkdate(start_date, no_of_workdays {, holiday})
```

Returnerad datatyp: Heltal

En kalender som visar hur `lastworkdate()`-funktionen används.



Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10 start_date	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26 end_date	27
28	29	30	31			

Begränsningar

Det finns ingen metod för att ändra `lastworkdate()`-funktionen för regioner eller scenarier där någonting annat än en arbetsvecka som börjar på måndag och slutar på fredag ingår.

Semesterparametern måste vara en strängkonstant. Det accepterar inte ett uttryck.

Användning

`lastworkdate()`-funktionen används vanligen som del av ett uttryck när användaren vill beräkna det föreslagna slutet för ett projekt eller en uppgift, baserat på när projektet börjar och de helgdagar/semesterdagar som inträffar under denna period..

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Argument

Argument	Beskrivning
start_date	Startdatum som ska utvärderas.
no_of_ workdays	Antalet arbetsdagar som ska uppnås.
holiday	Ledighetsperioder som ska undantas från arbetsdagar. En helgdag anges som ett strängkonstant datum. Du kan ange flera semesterdatum, avgränsade med kommatecken. **Exempel:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Exempel 1 – Grundläggande exempel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller projekt-ID, projektens startdatum och beräknad arbetsinsats i dagar för projekten. Datauppsättningen laddas i en tabell som heter `Projects`.
- En föregående laddning som innehåller funktionen `lastworkdate()` som är inställd som fältet ‘`end_ date`’ och identifierar när varje projekt beräknas avslutas.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Projects:
    Load
       *,
       LastWorkDate(start_date,effort) as end_date
       ;
Load
id,
start_date,
effort
Inline
[
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- id
- start_date
- effort
- end_date

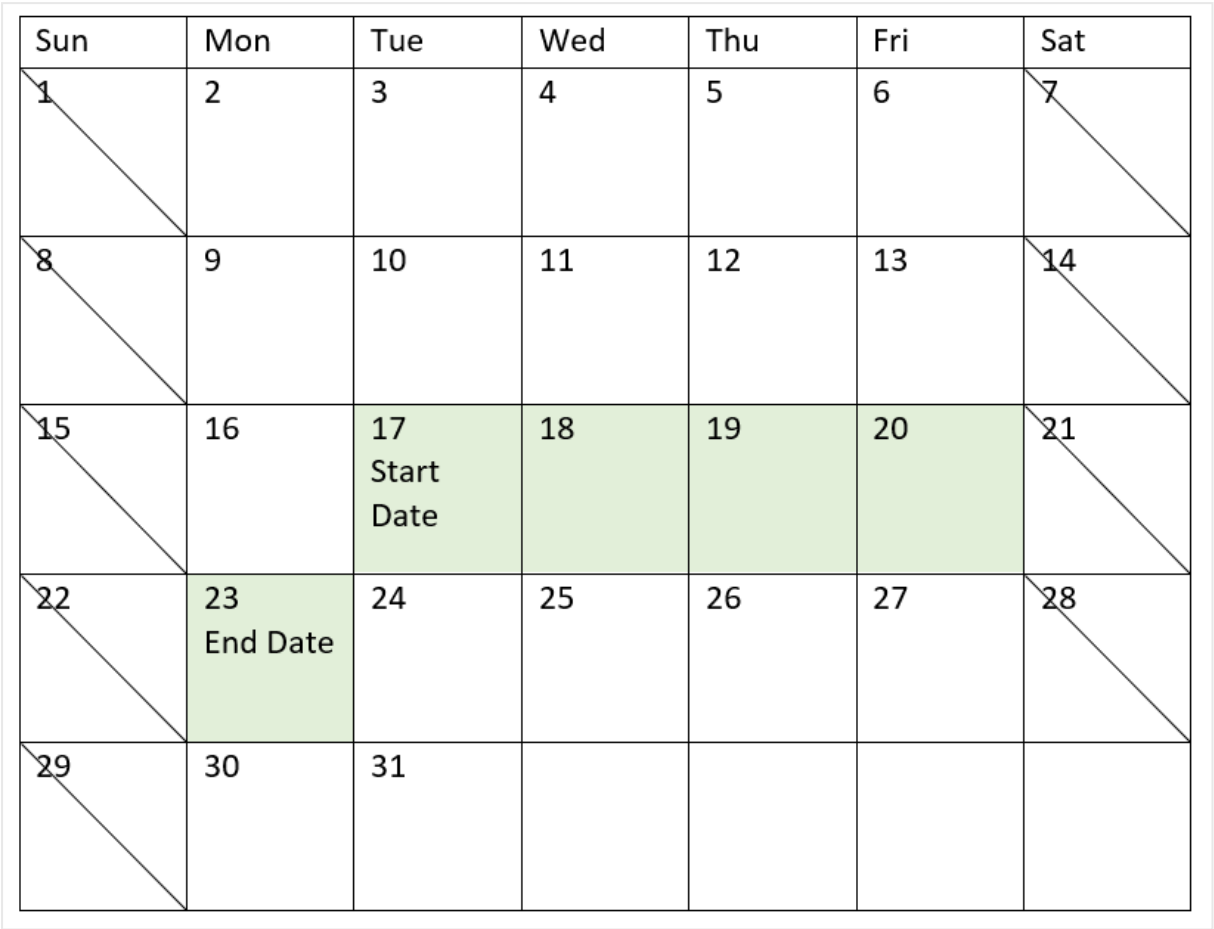
Resultattabell

Id	startdatum	effort	slutdatum
1	01/01/2022	14	01/20/2022
2	2022-02-10	17	2022-03-04
3	2022-05-17	5	05/23/2022
4	2022-06-01	12	06/16/2022
5	2022-08-10	26	09/14/2022

Eftersom det inte finns några schemalagda semesterdagar/helgdagar lägger funktionen till det definierade antalet arbetsdagar, måndag till fredag, till startdatumet för att hitta det tidigaste möjligaste slutdatumet.

I följande kalender visas start- och slutdatumet för projekt 3, med arbetsdagarna framhävda i grönt.

En kalender som visar start- och slutdatumet för projekt 3.



Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18	19	20	21
22	23 End Date	24	25	26	27	28
29	30	31				

Exempel 2 – enstaka semesterdag

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller projekt-ID, projektens startdatum och beräknad arbetsinsats i dagar för projekten. Datauppsättningen laddas i en tabell som heter `Projects`.
- En föregående laddning som innehåller funktionen `lastworkdate()` som är inställd som fältet `end_date` och identifierar när varje projekt beräknas avslutas.

Men det finns en schemalagd semesterdag 18 maj 2022. I `lastworkdate()`-funktionen i den föregående laddningen ingår semesterdagen i dess tredje argument för att identifiera när varje projekt är schemalagt att avslutas.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Projects:
    Load
       *,
       LastWorkDate(start_date,effort, '05/18/2022') as end_date
       ;
Load
id,
start_date,
effort
Inline
[
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `start_date`
- `effort`
- `end_date`

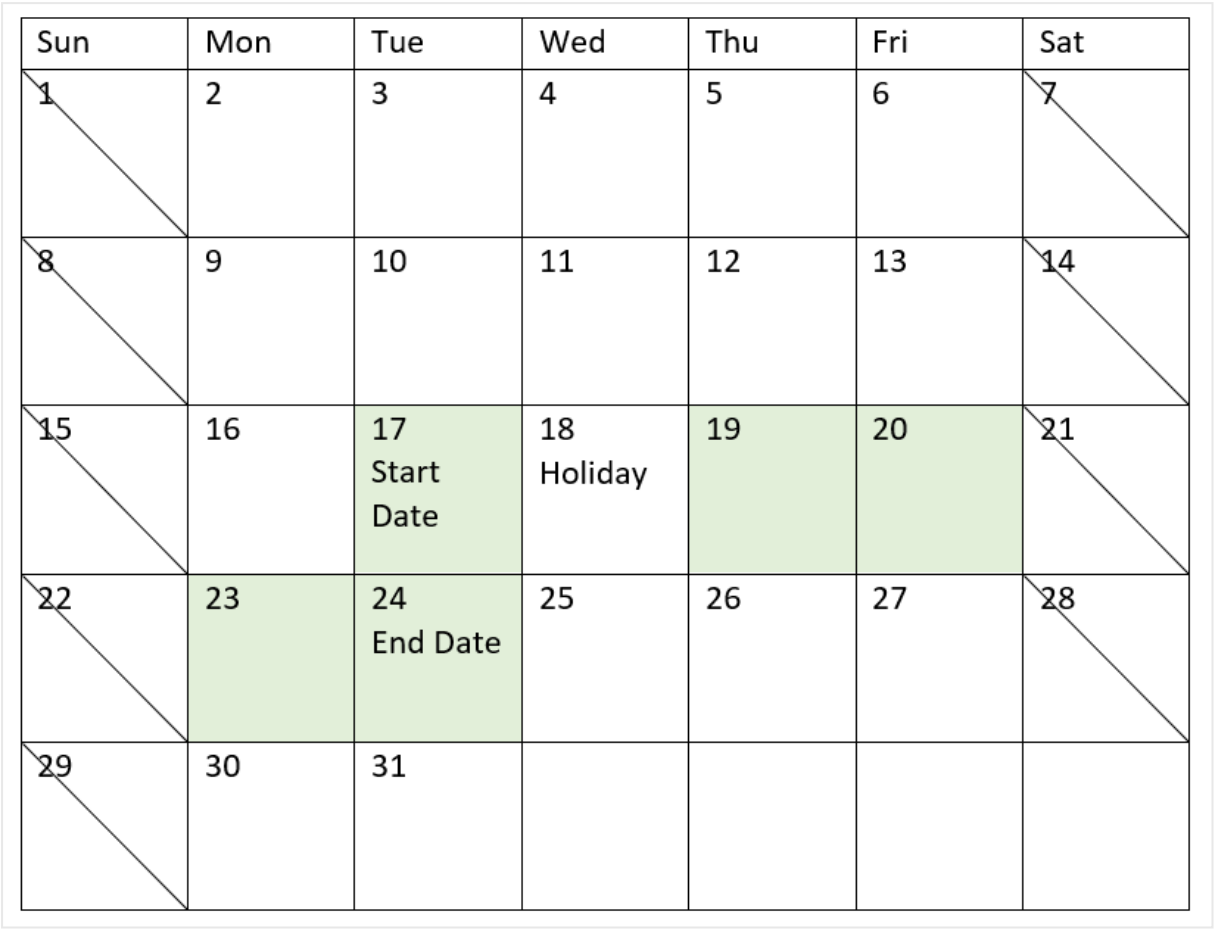
Resultattabell

Id	startdatum	effort	slutdatum
1	01/01/2022	14	01/20/2022
2	2022-02-10	17	2022-03-04
3	2022-05-17	5	05/24/2022
4	2022-06-01	12	06/16/2022
5	2022-08-10	26	09/14/2022

Den enda schemalagda semesterdagen anges som det tredje argumentet i `lastworkdate()`-funktionen. Till följd av detta skiftas slutdatumet för projekt 3 en dag framåt, eftersom semesterdagen äger rum på en av arbetsdagarna före slutdatumet.

Följande kalender visar start- och slutdatum för projekt 3 och visar att semesterdagen ändrar projektets slutdatum med en dag.

En kalender som visar start- och slutdatumet för projekt 3, med en semesterdag 18 maj



Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18 Holiday	19	20	21
22	23	24 End Date	25	26	27	28
29	30	31				

Exempel 3 – flera semesterdagar

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller projekt-ID, projektens startdatum och beräknad arbetsinsats i dagar för projekten. Datauppsättningen laddas i en tabell som heter `Projects`.
- En föregående laddning som innehåller funktionen `lastworkdate()` som är inställd som fältet `end_date` och identifierar när varje projekt beräknas avslutas.

Emellertid har fyra semesterdagar schemalagts för 19, 20, 21 och 22 maj. I `lastworkdate()`-funktionen i den föregående laddningen ingår var och en av semesterdagarna i dess tredje argument för att identifiera när varje projekt är schemalagt att avslutas.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Projects:
    Load
       *,
       LastWorkDate(start_date,effort, '05/19/2022','05/20/2022','05/21/2022','05/22/2022') as
end_date
       ;
Load
id,
start_date,
effort
Inline
[
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `start_date`
- `effort`
- `end_date`

Resultattabell

Id	startdatum	effort	slutdatum
1	01/01/2022	14	01/20/2022
2	2022-02-10	17	2022-03-04
3	2022-05-17	5	05/25/2022
4	2022-06-01	12	06/16/2022
5	2022-08-10	26	09/14/2022

De fyra semesterdagarna anges som en lista med argument i `lastworkdate()`-funktionen efter startdatumet och antalet arbetsdatagar.

Följande kalender visar start- och slutdatum för projekt 3 och visar att semesterdagarna ändrar projektets slutdatum med tre dagar.

En kalender som visar start- och slutdatumet för projekt 3 med semesterdagar 19 till 22 maj

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18	19 Holiday	20 Holiday	21 Holiday
22 Holiday	23	24	25 End Date	26	27	28
29	30	31				

Exempel 4 – enstaka semesterdag (diagram)

Laddningsskript och diagramuttryck

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är dock datauppsättningen oförändrad och har lästs in i applikationen. `end_date`-fältet beräknas via ett mått i ett diagramobjekt.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Projects:
Load
id,
start_date,
effort
Inline
[
```

```
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `start_date`
- `effort`

Skapa följande mått för att beräkna end_date:

- `=LastWorkDate(start_date,effort,'05/18/2022')`

Resultattabell

Id	startdatum	effort	=LastWorkDate(start_date,effort,'05/18/2022')
1	01/01/2022	14	01/20/2022
2	2022-02-10	17	2022-03-04
3	2022-05-17	5	05/23/2022
4	2022-06-01	12	06/16/2022
5	2022-08-10	26	09/14/2022

Den enda schemalagda semesterdagen anges som ett mått i diagrammet. Till följd av detta skiftas slutdatumet för projekt 3 en dag framåt, eftersom semesterdagen äger rum på en av arbetsdagarna före slutdatumet.

Följande kalender visar start- och slutdatum för projekt 3 och visar att semesterdagen ändrar projektets slutdatum med en dag.

En kalender som visar start- och slutdatumet för projekt 3, med en semesterdag 18 maj

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18 Holiday	19	20	21
22	23	24 End Date	25	26	27	28
29	30	31				

localtime

Denna funktion returnerar en tidsmarkör för aktuell tid från systemklockan för en angiven tidszon.

Syntax:

```
LocalTime([timezone [, ignoreDST ]])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
timezone	**timezone**anges som en sträng som innehåller någon av de geografiska platser som listas under **Time Zone** i **Windows Control Panel** för **Date and Time** eller som en sträng i formatet "GMT+tt:mm". En lista över godkända platser och tidszoner anges också i tabellen nedan. Om ingen tidszon anges returneras den lokala tiden. *Om du använder en DST-förskjutning (d.v.s. att du anger ett **ignoreDST**-argumentvärde som utvärderas till False) måste du ange ett ställe snarare än en GMT-förskjutning i **place**-argumentet. Det beror på att vid justering av sommartid krävs information om breddgrad i tillägg till informationen om längdgrad från en GMT-förskjutning. Mer information finns i Använder GMT-förskjutningar i kombination med DST (page 961).*
ignoreDST	Om det här argumentet utvärderas till True ignoreras DST (sommartid). Giltiga argumentvärden som utvärderas till True inkluderar `-1` och `True()`. Om det här argumentet utvärderas till False justeras tidsstämpeln för sommartid. Giltiga argumentvärden som utvärderas till False inkluderar `0` och `False()`. Om **ignoreDST**-argumentvärdet är ogiltigt utvärderar funktionen uttrycket som om **ignore_dst**-värdet utvärderas till True. Om **ignoreDST**-argumentvärdet inte har angetts utvärderar funktionen uttrycket som om **ignore_dst**-värdet utvärderas till False.

Giltiga platser och tidszoner

A-C	D-K	L-R	S-Z
Abu Dhabi	Darwin	La Paz	Samoa
Adelaide	Dhaka	Lima	Santiago
Alaska	Eastern Time (US & Canada)	Lisbon	Sapporo
Amsterdam	Edinburgh	Ljubljana	Sarajevo
Arizona	Ekaterinburg	London	Saskatchewan
Astana	Fiji	Madrid	Seoul
Athens	Georgetown	Magadan	Singapore
Atlantic Time (Canada)	Greenland	Mazatlan	Skopje

A-C	D-K	L-R	S-Z
Auckland	Greenwich Mean Time : Dublin	Melbourne	Sofia
Azores	Guadalajara	Mexico City	Solomon Is.
Baghdad	Guam	Mid-Atlantic	Sri Jayawardenepura
Baku	Hanoi	Minsk	St. Petersburg
Bangkok	Harare	Monrovia	Stockholm
Beijing	Hawaii	Monterrey	Sydney
Belgrade	Helsinki	Moscow	Taipei
Berlin	Hobart	Mountain Time (US & Canada)	Tallinn
Bern	Hong Kong	Mumbai	Tashkent
Bogota	Indiana (East)	Muscat	Tbilisi
Brasilia	International Date Line West	Nairobi	Tehran
Bratislava	Irkutsk	New Caledonia	Tokyo
Brisbane	Islamabad	New Delhi	Urumqi
Brussels	Istanbul	Newfoundland	Warsaw
Bucharest	Jakarta	Novosibirsk	Wellington
Budapest	Jerusalem	Nuku'alofa	West Central Africa
Buenos Aires	Kabul	Osaka	Vienna
Cairo	Kamchatka	Pacific Time (US & Canada)	Vilnius
Canberra	Karachi	Paris	Vladivostok
Cape Verde Is.	Kathmandu	Perth	Volgograd
Caracas	Kolkata	Port Moresby	Yakutsk
Casablanca	Krasnoyarsk	Prague	Yerevan
Central America	Kuala Lumpur	Pretoria	Zagreb
Central Time (US & Canada)	Kuwait	Quito	-
Chennai	Kyiv	Riga	-
Chihuahua	-	Riyadh	-

A-C	D-K	L-R	S-Z
Chongqing	-	Rome	-
Copenhagen	-	-	-

Exempel och resultat:

Nedanstående exempel baseras på den funktion som anropas 2023-08-14 08:39:47 lokal tid, med den lokala tidszonen på servern eller i skrivbordsmiljön inställd till GMT-05:00, och i en region som har implementerat sommartid vid det här listade datumet.

Skriptexempel

Exempel	Resultat
`localtime ()`	Returnerar den lokala tiden 2023-08-14 08:39:47.
`localtime ('London')`	Returnerar den lokala tiden i London, 2023-08-14 13:39:47.
`localtime ('GMT+02:00')`	Returnerar lokal tid i tidszonen GMT+02:00, 2023-08-14 14:39:47. Ingen justering görs för sommartid eftersom en GMT-förskjutning, snarare än en plats, anges.
`localtime ('Paris',-1)`	Returnerar lokal tid i Paris utan hänsyn till sommartid, 2023-08-14 13:39:47.
`localtime ('Paris',True())`	Returnerar lokal tid i Paris utan hänsyn till sommartid, 2023-08-14 13:39:47.
`localtime ('Paris',0)`	Returnerar lokal tid i Paris med hänsyn taget till sommartid, 2023-08-14 14:39:47.
`localtime ('Paris',False ())`	Returnerar lokal tid i Paris med hänsyn taget till sommartid, 2023-08-14 14:39:47.

Använder GMT-förskjutningar i kombination med DST

Enligt implementeringen av biblioteken för International Components for Unicode (ICU) i Qlik Sense krävs ytterligare information om breddgrad när förskjutningar av GMT (Greenwich Mean Time) i kombination med DST (Daylight Saving Time, sommartid) används.

GMT är en förskjutning av längdgraden (öst-västlig), medan DST är en förskjutning av breddgraden (nord-sydlig). Helsingfors (Finland) och Johannesburg (Sydafrika) har till exempel samma förskjutning GMT+02:00, men de har inte samma DST-förskjutning. Det innebär att förutom GMT-förskjutningen krävs för DST-förskjutningar information om breddgraden för den lokala tidszonen (inmatad geografisk tidszon) för att få fullständig information om lokala DST-förhållanden.

lunarweekend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden för den sista dagen i den sjudagarsperiod som innehåller **date**. Sjudagarsperioder i Qlik Sense definieras genom att räkna 1 januari som den första dagen i veckan. Bortsett från årets

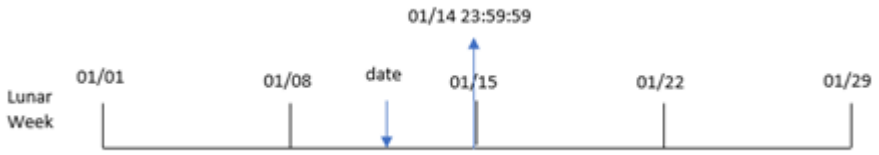
sista vecka kommer varje vecka att ha exakt sju dagar.

Syntax:

```
LunarweekEnd(date[, period_no[, first_week_day]])
```

Returnerad datatyp: dual

Exempeldiagram för `lunarweekend()`-funktionen



`lunarweekend()`-funktionen avgör vilken sjudagarsperiod `date` infaller i. Den returnerar sedan en tidsstämpel, i datumformat, för den sista millisekunden av den veckan.

Argument

Argument	Beskrivning
date	Det datum eller den tidsmarkör som ska utvärderas.
period_no	**period_no** är ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den sjudagarsperiod som innehåller **date**. Negativa värden i **period_no** anger föregående sjudagarsperioder och positiva värden anger efterföljande sjudagarsperioder.
first_week_day	En förflyttning av startpunkten som kan vara större eller mindre än noll. Detta flyttar början på året med det angivna antalet dagar och/eller delar av en dag.

Användning

`lunarweekend()`-funktionen används vanligtvis som en del av ett uttryck när användaren vill att beräkningen ska använda den del av veckan som ännu inte har inträffat. Till skillnad från `weekend()`-funktionen kommer den sista sjudagarsperioden i varje kalenderperiod att sluta 31 december. `lunarweekend()`-funktionen kan exempelvis användas för att beräkna ränta som fortfarande återstår under veckan.

Exempel på funktioner

Exempel	Resultat
`lunarweekend('01/12/2013')`	Returnerar 01/14/2013 23:59:59.
`lunarweekend('01/12/2013', -1)`	Returnerar 01/07/2013 23:59:59.
`lunarweekend('01/12/2013', 0, 1)`	Returnerar 01/15/2013 23:59:59.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2022 som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i `DateFormat`-systemvariabeln.
- Ett fält `end_of_week` skapas som returnerar en tidsmarkör för slutet av den sjudagarsperiod då transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekend(date) as end_of_week,
        timestamp(lunarweekend(date)) as end_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
```

```
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

Resultattabell

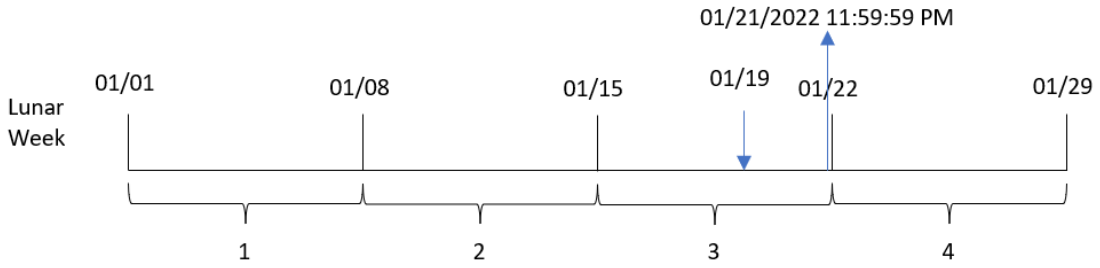
datum	slutet av veckan	tidsmarkör för slutet av veckan
1/7/2022	01/07/2022	2022-01-07 23:59:59
1/19/2022	2022-01-21	2022-01-21 23:59:59
2/5/2022	2022-02-11	2022-02-11 23:59:59
2/28/2022	2022-03-04	2022-03-04 23:59:59
3/16/2022	2022-03-18	2022-03-18 23:59:59
4/1/2022	04/01/2022	2022-04-01 23:59:59
5/7/2022	2022-05-13	2022-05-13 23:59:59
5/16/2022	2022-05-20	2022-05-20 23:59:59
6/15/2022	2022-06-17	2022-06-17 23:59:59
6/26/2022	2022-07-01	2022-07-01 23:59:59
7/9/2022	2022-07-15	2022-07-15 23:59:59
7/22/2022	2022-07-22	2022-07-22 23:59:59
7/23/2022	2022-07-29	2022-07-29 23:59:59
7/27/2022	2022-07-29	2022-07-29 23:59:59
8/2/2022	2022-08-05	2022-08-05 23:59:59

datum	slutet av veckan	tidsmarkör för slutet av veckan
8/8/2022	2022-08-12	2022-08-12 23:59:59
8/19/2022	2022-08-19	2022-08-19 23:59:59
9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
10/14/2022	10/14/2022	2022-10-14 23:59:59
10/29/2022	2022-11-04	2022-11-04 23:59:59

`end_of_week`-fältet skapades i den föregående load-satsen genom att använda `lunarweekend()`-funktionen och skicka `date`-fältet som funktionens argument.

`lunarweekend()`-funktionen identifierar vilken sjudagarsperiod datumvärdet infaller och returnerar en tidsstämpel för den sista millisekunden av den veckan.

Diagram för `lunarweekend()`-funktionen, exempel utan ytterligare argument



Transaktion 8189 ägde rum 19 januari. `lunarweekend()`-funktionen identifierar att sjudagarsperioden börjar 15 januari. Därför returnerar `end_of_week`-värdet för denna transaktion den sista millisekunden i sjudagarsperioden, vilket är 21 januari 23:59:59.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Ett fält `previous_lunar_week_end` skapas som returnerar tidsmarkören för slutet av sjudagarsperioden innan transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
```

```
Transactions:
    Load
        *,
        lunarweekend(date,-1) as previous_lunar_week_end,
        timestamp(lunarweekend(date,-1)) as previous_lunar_week_end_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `previous_lunar_week_end`
- `previous_lunar_week_end_timestamp`

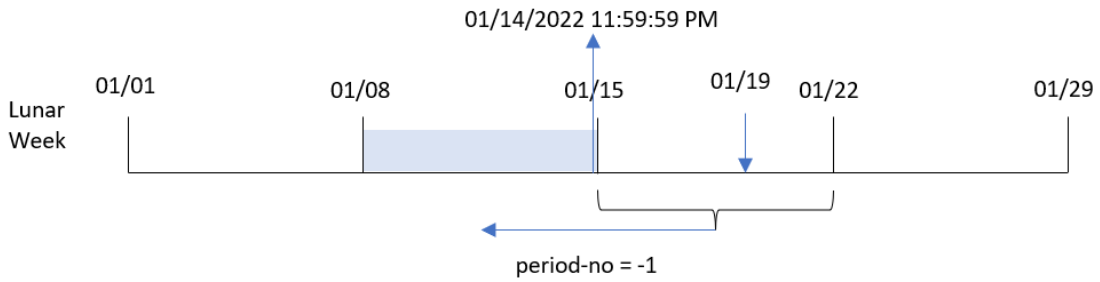
Resultattabell

datum	föregående_månvecka_slut	föregående_månvecka_slut_tidsmarkör
1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
1/19/2022	14/01/2022	2022-01-14 23:59:59
2/5/2022	02/04/2022	2022-02-04 23:59:59
2/28/2022	02/25/2022	2022-02-25 23:59:59
3/16/2022	03/11/2022	2022-03-18 23:59:59

datum	föregående_månvecka_slut	föregående_månvecka_slut_tidsmarkör
4/1/2022	03/25/2022	2022-03-25 23:59:59
5/7/2022	05/06/2022	2022-05-06 23:59:59
5/16/2022	2022-05-13	2022-05-13 23:59:59
6/15/2022	06/10/2022	2022-06-10 23:59:59
6/26/2022	06/24/2022	2022-06-24 23:59:59
7/9/2022	07/08/2022	2022-07-08 23:59:59
7/22/2022	2022-07-15	2022-07-15 23:59:59
7/23/2022	2022-07-22	2022-07-22 23:59:59
7/27/2022	2022-07-22	2022-07-22 23:59:59
8/2/2022	2022-07-29	2022-07-29 23:59:59
8/8/2022	2022-08-05	2022-08-05 23:59:59
8/19/2022	2022-08-12	2022-08-12 23:59:59
9/26/2022	09/23/2022	2022-09-23 23:59:59
10/14/2022	10/07/2022	2022-10-07 23:59:59
10/29/2022	10/28/2022	2022-10-28 23:59:59

Eftersom ett `period_no` på -1 användes som förskjutningsargument i `lunarweekend()`-funktionen i det här fallet identifierar funktionen först den sjudagarsperiod då transaktionerna ägde rum. Den skiftar sedan en vecka innan och identifierar den sista millisekunden i den sjudagarsperioden.

Diagram med exempel på `lunarweekend()`-funktionen, period_no



Transaktion 8189 ägde rum 19 januari. `lunarweekend()`-funktionen identifierar att sjudagarsperioden börjar 15 januari. Därför började den föregående sjudagarsperioden den 8 januari och slutade den 14 januari 23:59:59; detta är det värde som returneras för `previous_lunar_week_end`-fältet.

Exempel 3 – first_week_day

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet. I det här exemplet ställer vi in sjudagarsperioderna till att börja 5 januari.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekend(date,0,4) as end_of_week,
timestamp(lunarweekend(date,0,4)) as end_of_week_timestamp
;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

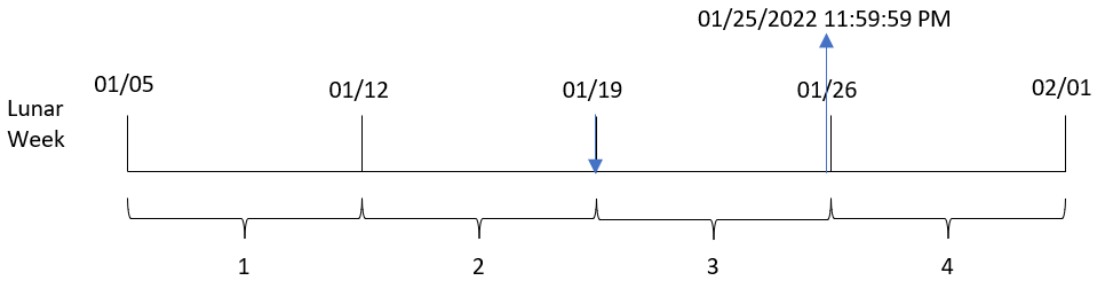
- `date`
- `end_of_week`
- `end_of_week_timestamp`

Resultattabell

datum	slutet av veckan	tidsmarkör för slutet av veckan
1/7/2022	11/01/2022	2022-01-11 23:59:59
1/19/2022	01/25/2022	2022-01-25 23:59:59
2/5/2022	02/08/2022	2022-02-08 23:59:59
2/28/2022	03/01/2022	2022-03-01 23:59:59
3/16/2022	03/22/2022	2022-03-22 23:59:59
4/1/2022	04/05/2022	2022-04-05 23:59:59
5/7/2022	05/10/2022	2022-05-10 23:59:59
5/16/2022	2022-05-17	2022-05-17 23:59:59
6/15/2022	06/21/2022	2022-06-21 23:59:59
6/26/2022	06/28/2022	2022-06-28 23:59:59
7/9/2022	07/12/2022	2022-07-12 23:59:59
7/22/2022	07/26/2022	2022-07-26 23:59:59
7/23/2022	2022-07-26	2022-07-26 23:59:59
7/27/2022	08/02/2022	2022-08-02 23:59:59
8/2/2022	2022-08-02	2022-08-02 23:59:59
8/8/2022	08/09/2022	2022-08-09 23:59:59
8/19/2022	08/23/2022	2022-08-23 23:59:59
9/26/2022	09/27/2022	2022-09-27 23:59:59
10/14/2022	10/18/2022	2022-10-18 23:59:59
10/29/2022	11/01/2022	2022-11-01 23:59:59

Eftersom `first_week_date`-argumentet 4 används i `lunarweekend()`-funktionen i det här fallet förskjuts årets början från 1 januari till 5 januari.

Diagram med exempel på `lunarweekend()`-funktionen, first_week_day



Transaktion 8189 ägde rum 19 januari. Eftersom sjudagarsperioderna börjar 5 januari identifierar `lunarweekend()`-funktionen att den sjudagarsperiod som innehåller 19 januari också börjar 19 januari. Därför inträffar slutet av den sjudagarsperioden 25 januari 23:59:59; detta är det värde som returneras för `end_of_week`-fältet.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som returnerar en tidsmarkör för slutet av sjudagarsperioden då transaktionerna ägde rum skapas som ett mått i ett diagramobjekt för programmet.

Laddningsskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
```

```
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

Lägg till följande mått:

```
=lunarweekend(date)
```

```
=timestamp(lunarweekend(date))
```

Resultattabell

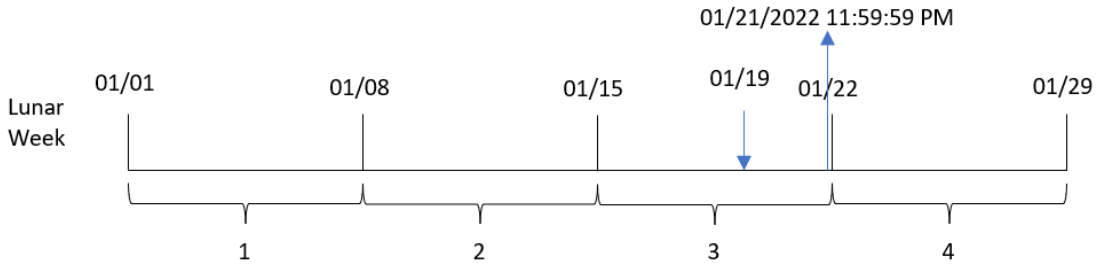
datum	=lunarweekend(date)	=tidsmarkör(lunarweekend(date))
1/7/2022	01/07/2022	2022-01-07 23:59:59
1/19/2022	2022-01-21	2022-01-21 23:59:59
2/5/2022	2022-02-11	2022-02-11 23:59:59
2/28/2022	2022-03-04	2022-03-04 23:59:59
3/16/2022	2022-03-18	2022-03-18 23:59:59
4/1/2022	04/01/2022	2022-04-01 23:59:59
5/7/2022	2022-05-13	2022-05-13 23:59:59
5/16/2022	2022-05-20	2022-05-20 23:59:59
6/15/2022	2022-06-17	2022-06-17 23:59:59
6/26/2022	2022-07-01	2022-07-01 23:59:59
7/9/2022	2022-07-15	2022-07-15 23:59:59
7/22/2022	2022-07-22	2022-07-22 23:59:59
7/23/2022	2022-07-29	2022-07-29 23:59:59
7/27/2022	2022-07-29	2022-07-29 23:59:59
8/2/2022	2022-08-05	2022-08-05 23:59:59
8/8/2022	2022-08-12	2022-08-12 23:59:59
8/19/2022	2022-08-19	2022-08-19 23:59:59
9/26/2022	09/30/2022	9/30/2022 11:59:59 PM

datum	=lunarweekend(date)	=tidsmarkör(lunarweekend(date))
10/14/2022	10/14/2022	2022-10-14 23:59:59
10/29/2022	2022-11-04	2022-11-04 23:59:59

`end_of_week`-måttet skapas i diagramobjektet genom att använda `lunarweekend()`-funktionen och skicka `date`-fältet som funktionens argument.

`lunarweekend()`-funktionen identifierar vilken sjudagarsperiod datumvärdet infaller och returnerar en tidsstämpel för den sista millisekunden av den veckan.

Diagram med exempel på `lunarweekend()`-funktionen, diagramobjekt



Transaktion 8189 ägde rum 19 januari. `lunarweekend()`-funktionen identifierar att sjudagarsperioden börjar 15 januari. Därför returnerar `end_of_week`-värdet för denna transaktion den sista millisekunden i sjudagarsperioden, vilket är 21 januari 23:59:59.

Exempel 5 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning läses in i en tabell som heter `Employee_Expenses`.
- Anställnings-ID, anställds namn och de genomsnittliga dagliga kostnadsanspråken för varje anställd.

Slutanvändaren vill ha ett diagramobjekt som visar de beräknade kostnadsanspråken som fortfarande återstår för resten av sjudagarsperioden, efter anställnings-ID och anställds namn.

Laddningsskript

```
Employee_Expenses:
Load
*
Inline
```

```
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell.
2. Lägg till följande fält som dimensioner:
   - `employee_id`
   - `employee_name`
3. Skapa därefter följande mått för att beräkna ackumulerad ränta:
   `=(lunarweekend(today(1))-today(1))*avg_daily_claim`
4. Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

employee_id	employee_name	=(lunarweekend(today(1))-today(1))*avg_daily_claim
182	Mark	$75.00
183	Deryck	$62,50
184	Dexter	$62,50
185	Sydney	$135.00
186	Agatha	$90.00

`lunarkweekend()`-funktionen returnerar slutdatumet för den aktuella sjudagarsperioden genom att använda dagens datum som enda argument. Sedan returnerar uttrycket antalet dagar som återstår av denna vecka genom att subtrahera dagens datum från årets slutdatum.

Detta värde multipliceras sedan med det genomsnittliga dagliga kostnadsanspråket från varje anställd för att beräkna det uppskattade värdet av anspråk som varje anställd förväntas göra under den återstående sjudagarsperioden.

lunarweekname

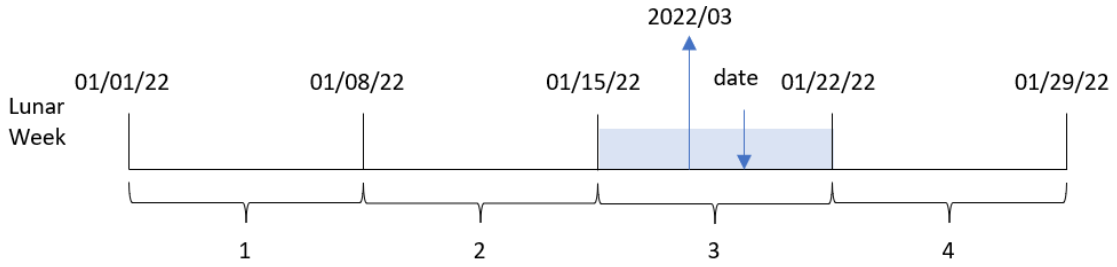
Denna funktion returnerar ett visningsvärde som visar året och sjudagarsperiodsnumret som motsvarar en tidsmarkör för den första millisekunden på den första dagen i sjudagarsperioden som innehåller **date**. Sjudagarsperioder i Qlik Sense definieras genom att räkna 1 januari som den första dagen i veckan. Bortsett från årets sista vecka kommer varje vecka att ha exakt sju dagar.

Syntax:

```
LunarWeekName(date [, period_no[, first_week_day]])
```

Returnerad datatyp: dual

Exempeldiagram för `lunarweekname()`-funktionen



`lunarweekname()`-funktionen fastställer vilken sjudagarsperiod som datumet infaller, med 1 januari som startdatum för veckoräkningen. Det returnerar sedan ett värde som består av `year/weekcount`.

Argument

Argument	Beskrivning
date	Det datum eller den tidsmarkör som ska utvärderas.
period_no	**period_no** är ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den sjudagarsperiod som innehåller **date**. Negativa värden i **period_no** anger föregående sjudagarsperioder och positiva värden anger efterföljande sjudagarsperioder.
first_week_day	En förflyttning av startpunkten som kan vara större eller mindre än noll. Detta flyttar början på året med det angivna antalet dagar och/eller delar av en dag.

Användning

Funktionen `lunarweekname()` är användbar när du vill jämföra sammanställningar efter sjudagarsperioder. Funktionen kan exempelvis användas för att fastställa den totala försäljningen av produkter per sjudagarsperiod. Sjudagarsperioder är användbara när du vill säkerställa att alla värden som finns i årets första vecka endast innehåller värden från tidigast 1 januari.

Dessa dimensioner kan skapas i laddningsskriptet genom att använda funktionen för att skapa ett fält i en Master Calendar-tabell. Funktionen kan också användas direkt i ett diagram som en beräknad dimension.

Exempel på funktioner

Exempel	Resultat
`lunarweekname('01/12/2013')`	Returnerar 2006/02.
`lunarweekname('01/12/2013', -1)`	Returnerar 2006/01.
`lunarweekname('01/12/2013', 0, 1)`	Returnerar 2006/02.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Datum utan ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2022 som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i systemvariabeln `DateFormat`.
- Ett fält `lunar_week_name` skapas som returnerar ett år och veckonummer för den sjudagarsperiod då transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekname(date) as lunar_week_name
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
```

```
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `lunar_week_name`

Resultattabell

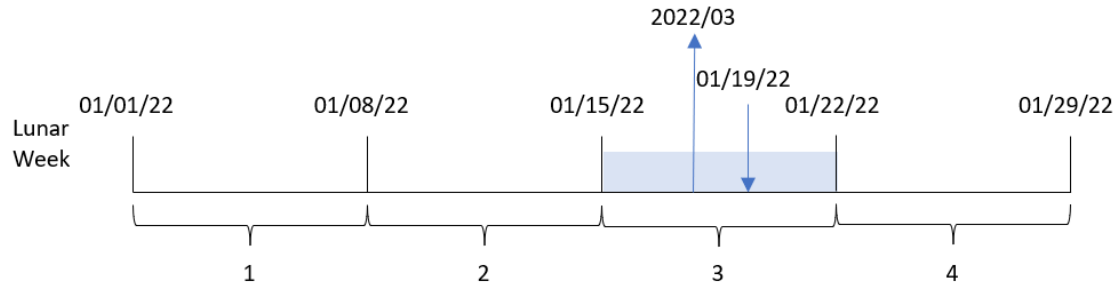
datum	lunar_week_name
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/06
2/28/2022	2022/09
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/19
5/16/2022	2022/20
6/15/2022	2022/24
6/26/2022	2022/26
7/9/2022	2022/28
7/22/2022	2022/29
7/23/2022	2022/30
7/27/2022	2022/30
8/2/2022	2022/31
8/8/2022	2022/32

datum	lunar_week_name
8/19/2022	2022/33
9/26/2022	2022/39
10/14/2022	2022/41
10/29/2022	2022/44

`lunar_week_name`-fältet skapades i den föregående load-satsen genom att använda `lunarweekname()`-funktionen och skicka `date`-fältet som funktionens argument.

`lunarweekname()`-funktionen identifierar vilken sjudagarsperiod datumvärdet infaller och returnerar år och veckonummer för detta datum.

Diagram för `lunarweekname()`-funktionen, exempel utan ytterligare argument



Transaktion 8189 ägde rum 19 januari. `lunarweekname()`-funktionen identifierar att detta datum infaller under den sjudagarsperiod som börjar 15 januari; detta är årets tredje sjudagarsperiod. Därför returnerar `lunar_week_name` värdet 2022/03 för transaktionen.

Exempel 2 – datum med argumentet period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Ett fält `previous_lunar_week_name` skapas som returnerar ett år och veckonummer för sjudagarsperioden innan transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
```

```
        *,
        lunarweekname(date,-1) as previous_lunar_week_name
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `previous_lunar_week_name`

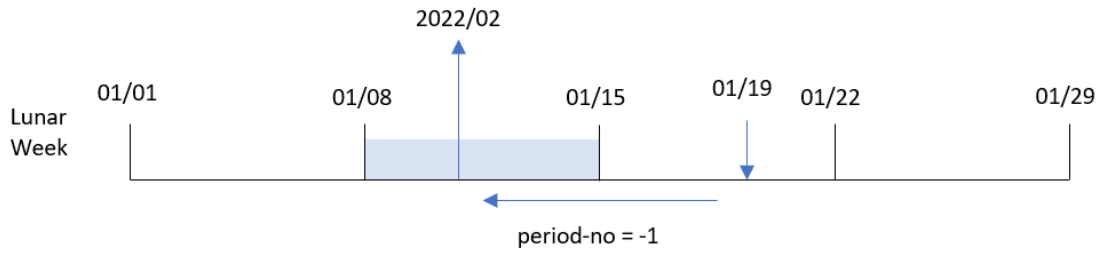
Resultattabell

datum	previous_lunar_week_name
1/7/2022	2021/52
1/19/2022	2022/02
2/5/2022	2022/05
2/28/2022	2022/08
3/16/2022	2022/10
4/1/2022	2022/12
5/7/2022	2022/18

datum	previous_lunar_week_name
5/16/2022	2022/19
6/15/2022	2022/23
6/26/2022	2022/25
7/9/2022	2022/27
7/22/2022	2022/28
7/23/2022	2022/29
7/27/2022	2022/29
8/2/2022	2022/30
8/8/2022	2022/31
8/19/2022	2022/32
9/26/2022	2022/38
10/14/2022	2022/40
10/29/2022	2022/43

Eftersom ett `period_no` på -1 användes som förskjutningsargument i `lunarweekname()`-funktionen i det här fallet identifierar funktionen först den sjudagarsperiod då transaktionerna ägde rum. Det returnerar då året och den föregående veckans nummer.

Diagram med exempel på `lunarweekname()`-funktionen, period_no



Transaktion 8189 ägde rum 19 januari. `lunarweekname()`-funktionen fastställer att denna transaktion ägde rum under årets tredje sjudagarsperiod och returnerar därför året och värdet för den föregående veckan, 2022/02, för `previous_lunar_week_name`-fältet.

Exempel 3 – datum med argumentet first_week_day

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet. I det här exemplet ställer vi in sjudagarsperioderna till att börja 5 januari.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekname(date,0,4) as lunar_week_name
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

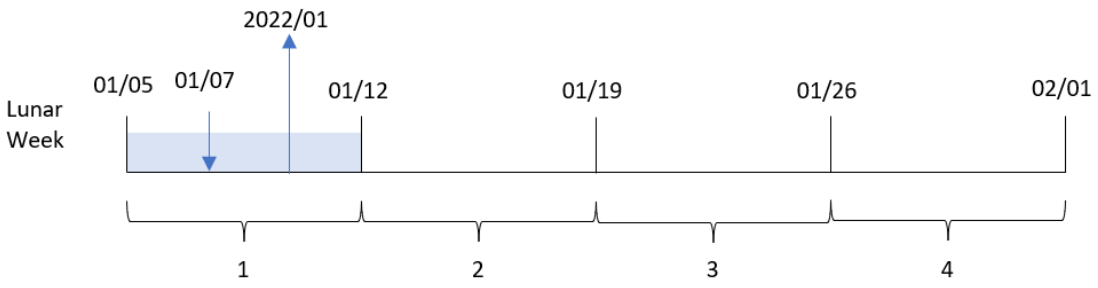
- `date`
- `lunar_week_name`

Resultattabell

datum	lunar_week_name
1/7/2022	2022/01
1/19/2022	2022/03

datum	lunar_week_name
2/5/2022	2022/05
2/28/2022	2022/08
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/18
5/16/2022	2022/19
6/15/2022	2022/24
6/26/2022	2022/25
7/9/2022	2022/27
7/22/2022	2022/29
7/23/2022	2022/29
7/27/2022	2022/30
8/2/2022	2022/30
8/8/2022	2022/31
8/19/2022	2022/33
9/26/2022	2022/38
10/14/2022	2022/41
10/29/2022	2022/43

Diagram med exempel på `lunarweekname()`-funktionen, first_week_day



Eftersom `first_week_date`-argumentet 4 används i `lunarweekname()`-funktionen i det här fallet förskjuts sjudagarsperiodens början från 1 januari till 5 januari.

Transaktionen 8188 ägde rum 7 januari. På grund av att sjudagarsperioderna börjar 5 januari identifierar `lunarweekname()`-funktionen att sjudagarsperioden där 7 januari ingår är årets första sjudagarsperiod. Därför är det returnerade `lunar_week_name`-värdet för den transaktoinen 2022/01.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som returnerar ett sjudagarsperiodsnummer och år då transaktionen ägde rum skapas som ett mått i ett diagramobjekt för programmet.

Laddningsskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

För att beräkna startdatum för den sjudagarsperiod då en transaktion äger rum skapar du följande mått:

```
=lunarweekname(date)
```

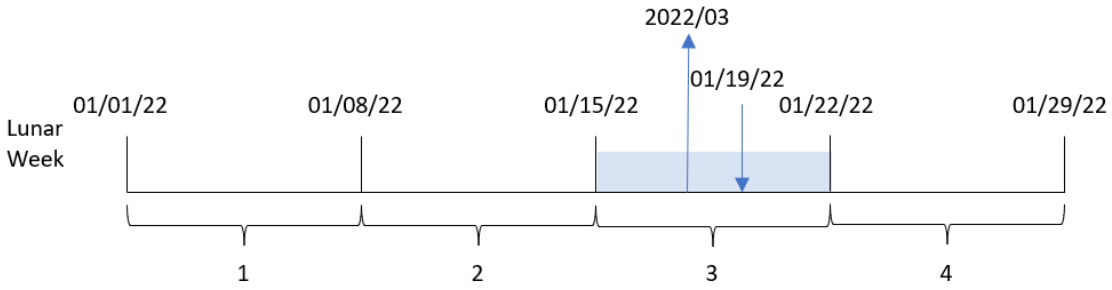
Resultattabell

datum	=lunarweekname(date)
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/06
2/28/2022	2022/09
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/19
5/16/2022	2022/20
6/15/2022	2022/24
6/26/2022	2022/26
7/9/2022	2022/28
7/22/2022	2022/29
7/23/2022	2022/30
7/27/2022	2022/30
8/2/2022	2022/31
8/8/2022	2022/32
8/19/2022	2022/33
9/26/2022	2022/39
10/14/2022	2022/41
10/29/2022	2022/44

`lunar_week_name`-måttet skapades i den föregående load-satsen genom att använda `lunarweekname()`-funktionen och skicka `date`-fältet som funktionens argument.

`lunarweekname()`-funktionen identifierar vilken sjudagarsperiod datumvärdet infaller och returnerar år och veckonummer för detta datum.

Diagram med exempel på `lunarweekname()`-funktionen, diagramobjekt



Transaktion 8189 ägde rum 19 januari. `lunarweekname()`-funktionen identifierar att detta datum infaller under den sjudagarsperiod som börjar 15 januari; detta är årets tredje sjudagarsperiod. Därför är `lunar_week_name` värdet för transaktionen 2022/03.

Exempel 5 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner efter tidsmarkör som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i systemvariabeln `DateFormat`.

Slutanvändaren vill ha ett diagramobjekt som visar den totala försäljningen per vecka för nuvarande år. Vecka 1, med en längd på sju dagar, ska börja 1 januari. Detta kan göras även när denna dimension inte är tillgänglig i datamodellen genom att använda `lunarweekname()`-funktionen som en beräknad dimension i diagrammet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
```

```
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell.
2. Skapa en beräknad dimension med följande uttryck:
   `=lunarweekname(date)`
3. Beräkna total försäljning med hjälp av följande aggregeringsmått.
   `=sum(amount)`
4. Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

=lunarweekname(date)	=sum(amount)
2022/01	$17.17
2022/03	$37.23
2022/06	$57.42
2022/09	$88.27
2022/11	$53.80
2022/13	$82.06
2022/19	$40.39
2022/20	$87.21
2022/24	$95.93
2022/26	$45.89
2022/28	$36.23
2022/29	$25.66
2022/30	$152.75

=lunarweekname(date)	=sum(amount)
2022/31	$76.11
2022/32	$25.12
2022/33	$46.23
2022/39	$84.21
2022/41	$96.24
2022/44	$67.67

lunarweekstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden för den första dagen i den sjudagarsperiod som innehåller **date**. Sjudagarsperioder i Qlik Sense definieras genom att räkna 1 januari som den första dagen i veckan. Bortsett från årets sista vecka kommer varje vecka att ha exakt sju dagar.

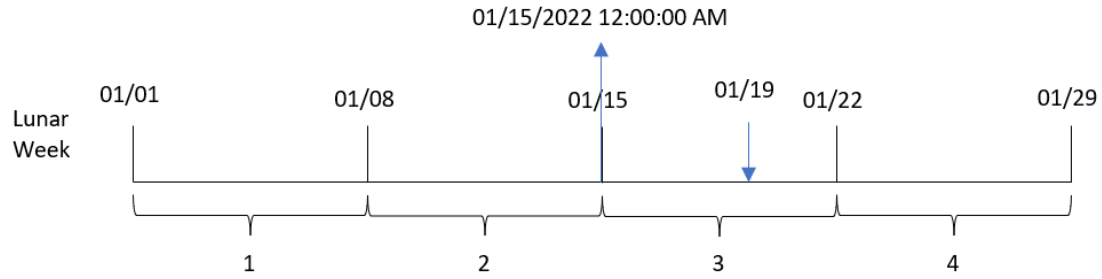
Syntax:

```
LunarweekStart(date[, period_no[, first_week_day]])
```

Returnerad datatyp: dual

`lunarweekstart()`-funktionen avgör vilken sjudagarsperiod `date` infaller i. Den returnerar sedan en tidsstämpel, i datumformat, för den första millisekunden av den veckan.

Exempeldiagram för `lunarweekstart()`-funktionen



Argument

Argument	Beskrivning
date	Det datum eller den tidsmarkör som ska utvärderas.
period_no	**period_no** är ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den sjudagarsperiod som innehåller **date**. Negativa värden i **period_no** anger föregående sjudagarsperioder och positiva värden anger efterföljande sjudagarsperioder.
first_week_day	En förflyttning av startpunkten som kan vara större eller mindre än noll. Detta flyttar början på året med det angivna antalet dagar och/eller delar av en dag.

Användning

`lunarweekstart()`-funktionen används vanligtvis som en del av ett uttryck när användaren vill att beräkningen ska använda den del av veckan som har förflutit hittills. Till skillnad från `weekstart()`-funktionen börjar veckan 1 januari varje nytt kalenderår och varje efterföljande vecka börjar sju dagar senare. `lunarweekstart()`-funktionen påverkas inte av systemvariabeln `FirstWeekDay`.

`lunarweekstart()` kan exempelvis användas för att beräkna den ränta som har ackumulerats under en vecka hittills.

Exempel på funktioner

Exempel	Resultat
`lunarweekstart ('01/12/2013')`	Returnerar 01/08/2013.
`lunarweekstart ('01/12/2013', -1)`	Returnerar 01/01/2013.
`lunarweekstart ('01/12/2013', 0, 1 )`	Returnerar 01/09/2013, eftersom inställningen first_week_day till 1 innebär att början av året ändras till 01/02/2013.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2022 som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i systemvariabeln `DateFormat`.

- Ett fält `start_of_week` skapas som returnerar en tidsmarkör för början av den sjudagarsperiod då transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekstart(date) as start_of_week,
        timestamp(lunarweekstart(date)) as start_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

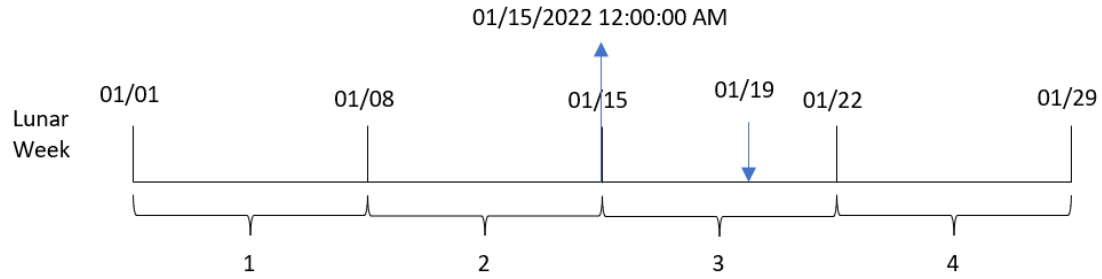
Resultattabell

datum	start_of_week	start_of_week_timestamp
1/7/2022	01/01/2022	2022-01-01 00:00:00
1/19/2022	2022-01-15	2022-01-15 00:00:00
2/5/2022	2022-02-05	2022-02-05 00:00:00
2/28/2022	02/26/2022	2022-02-26 00:00:00
3/16/2022	03/12/2022	2022-03-12 00:00:00
4/1/2022	2022-03-26	2022-03-26 00:00:00
5/7/2022	2022-05-07	2022-05-07 00:00:00
5/16/2022	2022-05-14	2022-05-14 00:00:00
6/15/2022	2022-06-11	2022-06-11 00:00:00
6/26/2022	2022-06-25	2022-06-25 00:00:00
7/9/2022	2022-07-09	2022-07-09 00:00:00
7/22/2022	2022-07-16	2022-07-16 00:00:00
7/23/2022	2022-07-23	2022-07-23 00:00:00
7/27/2022	2022-07-23	2022-07-23 00:00:00
8/2/2022	2022-07-30	2022-07-30 00:00:00
8/8/2022	06/08/2022	6/8/2022 00:00:00
8/19/2022	13/08/2022	13/8/2022 00:00:00
9/26/2022	24/09/2022	2022-09-24 00:00:00
10/14/2022	2022-10-08	2022-10-08 00:00:00
10/29/2022	10/29/2022	2022-10-29 00:00:00

`start_of_week`-fältet skapades i den föregående load-satsen genom att använda `lunarweekstart()`-funktionen och skicka `date`-fältet som funktionens argument.

`lunarweekstart()`-funktionen identifierar vilken sjudagarsperiod datumvärdet infaller och returnerar en tidsstämpel för den första millisekunden av den veckan.

Diagram för `lunarweekstart()`-funktionen, exempel utan ytterligare argument



Transaktion 8189 ägde rum 19 januari. `lunarweekstart()`-funktionen identifierar att sjudagarsperioden börjar 15 januari. Därför returnerar `start_of_week`-värdet för denna transaktion den första millisekunden denna dag, vilket är 15 januari 12:00:00 AM.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Ett fält `previous_lunar_week_start` skapas som returnerar tidsmarkören för början av sjudagarsperioden innan transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekstart(date,-1) as previous_lunar_week_start,
        timestamp(lunarweekstart(date,-1)) as previous_lunar_week_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
```

```
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

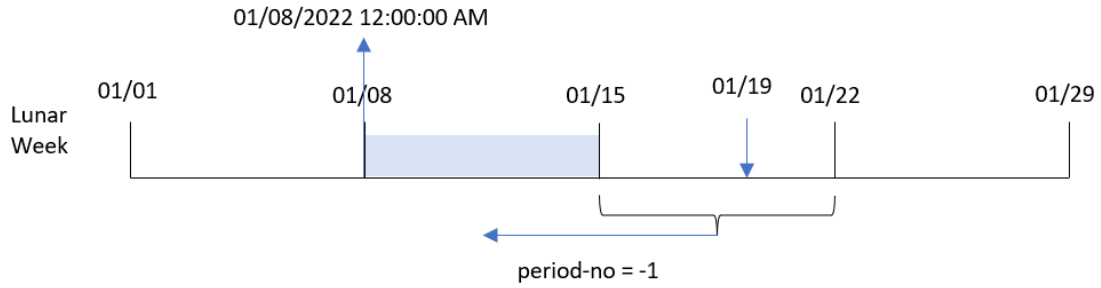
Resultat

Resultattabell

datum	föregående_månvecka_start	föregående_månvecka_start_tidsmarkör
1/7/2022	12/24/2021	12/24/2021 12:00:00 AM
1/19/2022	01/08/2022	2022-01-08 00:00:00
2/5/2022	29/01/2022	2022-01-29 00:00:00
2/28/2022	02/19/2022	2022-02-19 00:00:00
3/16/2022	03/05/2022	2022-03-05 00:00:00
4/1/2022	2022-03-19	2022-03-19 00:00:00
5/7/2022	04/30/2022	2022-04-30 00:00:00
5/16/2022	2022-05-07	2022-05-07 00:00:00
6/15/2022	06/04/2022	2022-06-04 00:00:00
6/26/2022	2022-06-18	2022-06-18 00:00:00
7/9/2022	2022-07-02	2022-07-02 00:00:00
7/22/2022	2022-07-09	2022-07-09 00:00:00
7/23/2022	2022-07-16	2022-07-16 00:00:00
7/27/2022	2022-07-16	2022-07-16 00:00:00
8/2/2022	2022-07-23	2022-07-23 00:00:00
8/8/2022	2022-07-30	2022-07-30 00:00:00
8/19/2022	2022-08-06	2022-08-06 00:00:00
9/26/2022	09/17/2022	2022-09-17 00:00:00
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/22/2022	10/22/2022 12:00:00 AM

Eftersom ett `period_no` på -1 användes som förskjutningsargument i `lunarweekstart()`-funktionen i det här fallet, identifierar funktionen först den sjudagarsperiod då transaktionerna äger rum. Den skiftar sedan en vecka innan och identifierar den första millisekunden i den sjudagarsperioden.

Diagram med exempel på `lunarweekstart()`-funktionen, period_no



Transaktion 8189 ägde rum 19 januari. `lunarweekstart()`-funktionen identifierar att sjudagarsperioden börjar 15 januari. Därför började den föregående sjudagarsperioden den 8 januari 12:00:00 AM; detta är det värde som returneras för `previous_lunar_week_start`-fältet.

Exempel 3 – first_week_day

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet. I det här exemplet ställer vi in sjudagarsperioderna till att börja 5 januari.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekstart(date,0,4) as start_of_week,
        timestamp(lunarweekstart(date,0,4)) as start_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
```

```
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

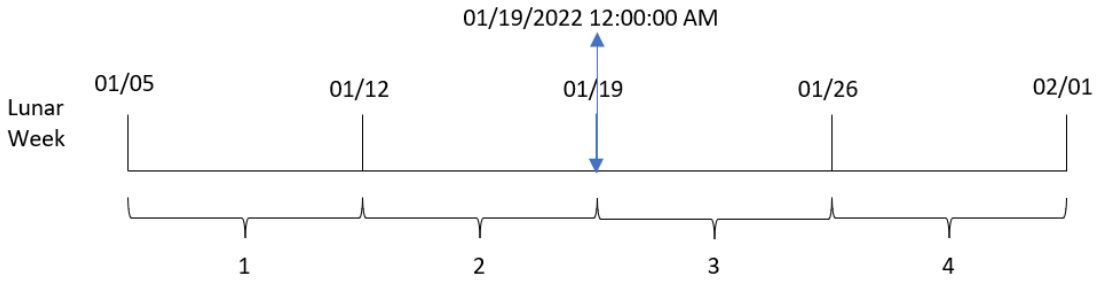
Resultattabell

datum	start_of_week	start_of_week_timestamp
1/7/2022	01/05/2022	1/5/2022 12:00:00 AM
1/19/2022	01/19/2022	1/19/2022 12:00:00 AM
2/5/2022	02/02/2022	2022-02-02 00:00:00
2/28/2022	02/23/2022	2022-02-23 00:00:00
3/16/2022	03/16/2022	2022-03-16 00:00:00
4/1/2022	03/30/2022	2022-03-30 00:00:00
5/7/2022	05/04/2022	2022-05-04 00:00:00
5/16/2022	2022-05-11	2022-05-11 00:00:00
6/15/2022	06/15/2022	2022-06-15 00:00:00
6/26/2022	06/22/2022	2022-06-22 00:00:00
7/9/2022	07/06/2022	2022-07-06 00:00:00
7/22/2022	2022-07-20	2022-07-20 00:00:00
7/23/2022	2022-07-20	2022-07-20 00:00:00
7/27/2022	2022-07-27	2022-07-27 00:00:00
8/2/2022	2022-07-27	2022-07-27 00:00:00

datum	start_of_week	start_of_week_timestamp
8/8/2022	08/03/2022	2022-08-03 00:00:00
8/19/2022	08/17/2022	2022-08-17 00:00:00
9/26/2022	09/21/2022	2022-09-21 00:00:00
10/14/2022	10/12/2022	2022-10-12 00:00:00
10/29/2022	10/26/2022	2022-10-26 00:00:00

Eftersom `first_week_date`-argumentet 4 används i `lunarweekstart()`-funktionen i det här fallet förskjuts årets början från 1 januari till 5 januari.

Diagram med exempel på `lunarweekstart()`-funktionen, first_week_day



Transaktion 8189 ägde rum 19 januari. Eftersom sjudagarsperioderna börjar 5 januari identifierar `lunarweekstart()`-funktionen att den sjudagarsperiod som innehåller 19 januari också börjar 19 januari 12:00:00 AM. Därför är detta det värde som returneras för `start_of_week`-fältet.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som returnerar en tidsmarkör för början av sjudagarsperioden då transaktionerna ägde rum skapas som ett mått i ett diagramobjekt för programmet.

Laddningsskript

```
Transactions:
Load
*
Inline
[
```

```
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

Lägg till följande mått:

```
=lunarweekstart(date)
```

```
=timestamp(lunarweekstart(date))
```

Resultattabell

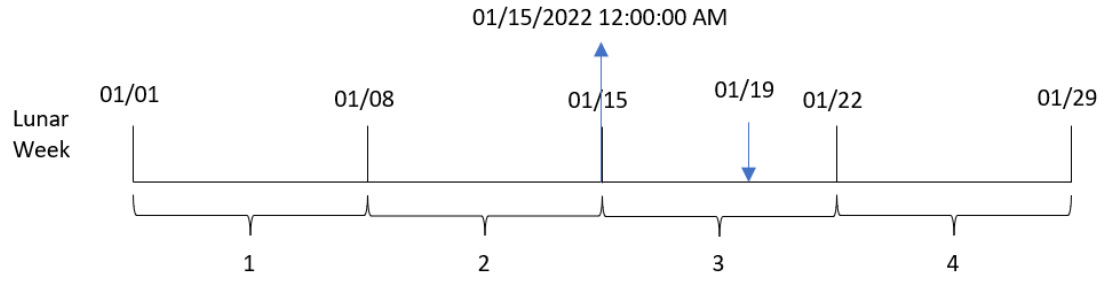
datum	=lunarweekstart(date)	=tidsmarkör(lunarweekstart(date))
1/7/2022	01/01/2022	2022-01-01 00:00:00
1/19/2022	2022-01-15	2022-01-15 00:00:00
2/5/2022	2022-02-05	2022-02-05 00:00:00
2/28/2022	02/26/2022	2022-02-26 00:00:00
3/16/2022	03/12/2022	2022-03-12 00:00:00
4/1/2022	2022-03-26	2022-03-26 00:00:00
5/7/2022	2022-05-07	2022-05-07 00:00:00
5/16/2022	2022-05-14	2022-05-14 00:00:00
6/15/2022	2022-06-11	2022-06-11 00:00:00
6/26/2022	2022-06-25	2022-06-25 00:00:00
7/9/2022	2022-07-09	2022-07-09 00:00:00

datum	=lunarweekstart(date)	=tidsmarkör(lunarweekstart(date))
7/22/2022	2022-07-16	2022-07-16 00:00:00
7/23/2022	2022-07-23	2022-07-23 00:00:00
7/27/2022	2022-07-23	2022-07-23 00:00:00
8/2/2022	2022-07-30	2022-07-30 00:00:00
8/8/2022	06/08/2022	6/8/2022 00:00:00
8/19/2022	13/08/2022	13/8/2022 00:00:00
9/26/2022	24/09/2022	2022-09-24 00:00:00
10/14/2022	2022-10-08	2022-10-08 00:00:00
10/29/2022	10/29/2022	2022-10-29 00:00:00

`start_of_week`-måttet skapades i ett diagramobjektet genom att använda `lunarweekstart()`-funktionen och skicka datumfältet som funktionens argument.

`lunarweekstart()`-funktionen identifierar vilken sjudagarsperiod datumvärdet infaller och returnerar en tidsstämpel för den sista millisekunden av den veckan.

Diagram med exempel på `lunarweekstart()`-funktionen, diagramobjekt



Transaktion 8189 ägde rum 19 januari. `lunarweekstart()`-funktionen identifierar att sjudagarsperioden börjar 15 januari. Därför är `start_of_week`-värdet för denna transaktion den första millisekunden denna dag, vilket är 15 januari 12:00:00 AM.

Exempel 5 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning lånesaldon som laddas i en tabell som heter `Loans`.
- Data som består av låne-ID:n, saldot vid veckans början och den enkla räntan som tas ut på varje lån per år.

Slutanvändaren vill ha ett diagramobjekt som visar, efter låne-ID, den aktuella räntan som har ackumulerats för varje lån hittills under veckan.

Laddningsskript

```
Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell.
2. Lägg till följande fält som dimensioner:
    - `loan_id`
    - `start_balance`
3. Skapa därefter följande mått för att beräkna ackumulerad ränta:
   `=start_balance*(rate*(today(1)-lunarweekstart(today(1)))/365)`
4. Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

loan_id	start_balance	=start_balance*(rate*(today(1)- lunarweekstart (today(1)))/365)
8188	USD 10000.00	$15.07
8189	$15000.00	$128.84
8190	$17500.00	$63.29
8191	$21000.00	$107.59
8192	$90000.00	$1139.18

Genom att använda dagens datum som enda argument returnerar `lunarweekstart()`-funktionen slutdatumet för det aktuella året. Genom att subtrahera resultatet från det aktuella datumet returnerar uttrycket antalet dagar som har förflutit hittills denna vecka.

Detta värde multipliceras sedan med räntan och divideras med 365 för att returnera den effektiva räntan för perioden. Resultatet multipliceras sedan med lånets utgångssaldo för att returnera den upplupna räntan hittills denna vecka.

makedate

Denna funktion returnerar ett datum beräknat utifrån året **YYYY**, månaden **MM** och dagen **DD**.

Syntax:

```
MakeDate(YYYY [ , MM [ , DD ] ])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
YYYY	Året i form av ett heltal.
MM	Månaden i form av ett heltal. Om inget månadsnummer anges, antas 1 (januari).
DD	Dagen i form av ett heltal. Om inget dagsnummer anges, antas 1 (den första).

Användning

`makedate()`-funktionen skulle i normalfallet användas i skriptet för datagenerering för att generera en kalender. Detta kan också användas när datumfältet inte är direkt tillgängligt som datum, utan måste transformeras för att extrahera års-, månads- och datumkomponenterna.

I de här exemplen används datumformatet MM/DD/ÅÅÅÅ. Datumformatet anges i `SET DateFormat`-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Exempel på funktioner

Exempel	Resultat
`makedate(2012)`	Returnerar 01/01/2012.
`makedate(12)`	Returnerar 01/01/2012.
`makedate(2012,12)`	Returnerar 12/01/2012.
`makedate(2012,2,14)`	Returnerar 02/14/2012.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra

formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Grundläggande exempel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2018 som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i systemvariabeln `DateFormat`.
- Ett fält `transaction_date` skapas som returnerar ett datum i formatet MM/DD/YYYY.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        makedate(transaction_year, transaction_month, transaction_day) as transaction_date
        ;
Load * Inline [
transaction_id, transaction_year, transaction_month, transaction_day, transaction_amount,
transaction_quantity, customer_id
3750, 2018, 08, 30, 12423.56, 23, 2038593
3751, 2018, 09, 07, 5356.31, 6, 203521
3752, 2018, 09, 16, 15.75, 1, 5646471
3753, 2018, 09, 22, 1251, 7, 3036491
3754, 2018, 09, 22, 21484.21, 1356, 049681
3756, 2018, 09, 22, -59.18, 2, 2038593
3757, 2018, 09, 23, 3177.4, 21, 203521
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- transaction_year
- transaction_month
- transaction_day
- transaction_date

Resultattabell

transaktionsår	transaktionsmånad	transaktionsdag	transaktionsdatum
2018	08	30	08/30/2018
2018	09	07	09/07/2018
2018	09	16	09/16/2018
2018	09	22	09/22/2018
2018	09	23	23/09/2018

transaction_date-fältet skapades i den föregående load-satsen genom att använda makedate()-funktionen och skicka års-, månads- och datumfälten som funktionens argument.

Funktionen kombinerar och konverterar sedan de här värdena till ett datumfält och returnerar resultatet i form av systemvariabeln DateFormat.

Exempel 2 – modifierat DateFormat

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Ett fält transaction_date skapas i formatet DD/MM/YYYY utan att systemvariabeln DateFormat ändras.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        date(makedate(transaction_year, transaction_month, transaction_day), 'DD/MM/YYYY') as
transaction_date
        ;
Load * Inline [
transaction_id, transaction_year, transaction_month, transaction_day, transaction_amount,
transaction_quantity, customer_id
3750, 2018, 08, 30, 12423.56, 23, 2038593
```

```
3751, 2018, 09, 07, 5356.31, 6, 203521
3752, 2018, 09, 16, 15.75, 1, 5646471
3753, 2018, 09, 22, 1251, 7, 3036491
3754, 2018, 09, 22, 21484.21, 1356, 049681
3756, 2018, 09, 22, -59.18, 2, 2038593
3757, 2018, 09, 23, 3177.4, 21, 203521
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `transaction_year`
- `transaction_month`
- `transaction_day`
- `transaction_date`

Resultattabell

transaktionsår	transaktionsmånad	transaktionsdag	transaktionsdatum
2018	08	30	30/08/2018
2018	09	07	07/09/2018
2018	09	16	16/09/2018
2018	09	22	22/09/2018
2018	09	23	23/09/2018

I det här fallet nästlas `makedate()`-funktion inuti `date()`-funktionen. Det andra argumentet i `date()`-funktionen ställer in formatet i `makedate()`-funktionens resultat till det erforderliga DD/MM/YYYY.

Exempel 3 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner för 2018 som laddas i en tabell som heter `Transactions`.
- Tillhandahållna transaktionsdatum över två fält: `year` och `month`.

Skapa ett diagramobjektmått, `transaction_date`, som returnerar ett datum i formatet MM/DD/YYYY.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load * Inline [
transaction_id, transaction_year, transaction_month, transaction_amount, transaction_quantity,
customer_id
3750, 2018, 08, 12423.56, 23, 2038593
3751, 2018, 09, 5356.31, 6, 203521
3752, 2018, 09, 15.75, 1, 5646471
3753, 2018, 09, 1251, 7, 3036491
3754, 2018, 09, 21484.21, 1356, 049681
3756, 2018, 09, -59.18, 2, 2038593
3757, 2018, 09, 3177.4, 21, 203521
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `year`
- `month`

Skapa detta mått för att fastställa `transaction_date`.

```
=makedate(transaction_year,transaction_month)
```

Resultattabell

transaktionsår	transaktionsmånad	transaktionsdatum
2018	08	08/01/2018
2018	09	09/01/2018

`transaction_date`-måttet skapades i den föregående load-satsen genom att använda `makedate()`-funktionen och skicka års- och månadsfälten som funktionens argument.

Funktionen kombinerar sedan dessa värden och det antagna dagsvärdet 01. De här värdena konverteras sedan till ett datumfält och returnerar resultatet i form av systemvariabeln `DateFormat`.

Exempel 4 – Scenario

Laddningsskript och diagramuttryck

Översikt

Skapa en kalenderdatauppsättning för kalenderåret 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
       load
                *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 400;
```

Resultat

Resultattabell

datum
01/01/2022
01/02/2022
01/03/2022
01/04/2022
01/05/2022
01/06/2022
01/07/2022
01/08/2022
01/09/2022
01/10/2022
11/01/2022
12/01/2022
13/01/2022
14/01/2022
15/01/2022
01/16/2022
17/01/2022
01/18/2022
01/19/2022
01/20/2022
2022-01-21

datum
2022-01-22
01/23/2022
01/24/2022
01/25/2022
+ ytterligare 340 rader

`makedate()`-funktionen skapar ett datumvärde för 31 december, 2021. `recno()`-funktionen tillhandahåller postnumret för den post som för tillfället laddas in i tabellen, med början från 1. Därför har den första posten datumet 1 januari 2022. Varje påföljande `recno()` ökar då detta datum med 1. Uttrycket är inneslutet i en `date()`-funktion för att konvertera värdet till ett datum. Denna process upprepas 400 gånger av `autogenerate`-funktionen. Slutligen kan, genom att använda en föregående laddning, ett `where`-villkor användas för att endast ladda datum från 2022. Skriptet genererar en kalender som innehåller alla datum under 2022.

maketime

Denna funktion returnerar en tid beräknat utifrån timmar **hh**, minuter **mm** och sekunder **ss**.

Syntax:

```
MakeTime(hh [ , mm [ , ss ] ])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
hh	Timmen i form av ett heltal.
mm	Minuten i form av ett heltal. Om ingen minut angivits, antas 00.
ss	Sekunden i form av ett heltal. Om ingen sekund angivits, antas 00.

Användning

`maketime()`-funktionen skulle i normalfallet användas i skriptet för datagenerering för att generera ett tidsfält. Ibland när tidsfältet härleds från inmatningstext kan denna funktion användas för att konstruera tiden med hjälp av dess komponenter.

I de här exemplen används tidsformatet `h:mm:ss`. Tidsformatet anges i `SET TimeFormat`-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Exempel på funktioner

Exempel	Resultat
`maketime(22)`	Returnerar 22:00:00.
`maketime(22, 17)`	Returnerar 22:17:00.
`maketime(22,17,52 )`	Returnerar 22:17:52.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – maketime()

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner som laddas i en tabell som heter `Transactions`.
- Tillhandahållna transaktionstider som tillhandahålls över tre fält: `hours`, `minutes`, och `seconds`.
- Ett fält `transaction_time` skapas som returnerar tiden i formatet för systemvariabeln `TimeFormat`.

Laddningsskript

```
SET TimeFormat='h:mm:ss TT';

Transactions:
    Load
        *,
        maketime(transaction_hour, transaction_minute, transaction_second) as transaction_time
        ;
Load * Inline [
transaction_id, transaction_hour, transaction_minute, transaction_second, transaction_amount,
transaction_quantity, customer_id
3750, 18, 43, 30, 12423.56, 23, 2038593
```

```
3751, 6, 32, 07, 5356.31, 6, 203521
3752, 12, 09, 16, 15.75, 1, 5646471
3753, 21, 43, 41, 7, 3036491
3754, 17, 55, 22, 21484.21, 1356, 049681
3756, 2, 52, 22, -59.18, 2, 2038593
3757, 9, 25, 23, 3177.4, 21, 203521
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `transaction_hour`
- `transaction_minute`
- `transaction_second`
- `transaction_time`

Resultattabell

transaction_hour	transaction_minute	transaction_second	transaction_time
2	52	22	2:52:22 AM
6	32	07	6:32:07 AM
9	25	23	9:25:23 AM
12	09	16	12:09:16 PM
17	55	22	5:55:22 PM
18	43	30	6:43:30 PM
21	43	41	9:43:41 PM

`transaction_time`-fältet skapades i den föregående load-satsen genom att använda `maketime()`-funktionen och skicka tim-, minut- och sekundfälten som funktionens argument.

Funktionen kombinerar och konverterar sedan de här värdena till ett tidsfält och returnerar resultatet i tidsformatet för systemvariabeln `TimeFormat`.

Exempel 2 – funktionen time()

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Ett fält `transaction_time` skapas som låter oss visa resultaten i 24-timmarsformat utan att ändra systemvariabeln `TimeFormat`.

Laddningsskript

```
SET TimeFormat='h:mm:ss TT';

Transactions:
    Load
        *,
        time(maketime(transaction_hour, transaction_minute, transaction_second),'h:mm:ss') as
transaction_time
        ;
Load * Inline [
transaction_id, transaction_hour, transaction_minute, transaction_second, transaction_amount,
transaction_quantity, customer_id
3750, 18, 43, 30, 12423.56, 23, 2038593
3751, 6, 32, 07, 5356.31, 6, 203521
3752, 12, 09, 16, 15.75, 1, 5646471
3753, 21, 43, 41, 7, 3036491
3754, 17, 55, 22, 21484.21, 1356, 049681
3756, 2, 52, 22, -59.18, 2, 2038593
3757, 9, 25, 23, 3177.4, 21, 203521
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `transaction_hour`
- `transaction_minute`
- `transaction_second`
- `transaction_time`

Resultattabell

transaction_hour	transaction_minute	transaction_second	transaction_time
2	52	22	2:52:22
6	32	07	6:32:07
9	25	23	9:25:23
12	09	16	12:09:16
17	55	22	17:55:22
18	43	30	18:43:30
21	43	41	21:43:41

I det här fallet nästlas `maketime()`-funktion inuti `time()`-funktionen. Det andra argumentet i `time()`-funktionen ställer in formatet i `maketime()`-funktionens resultat till det erforderliga `h:mm:ss`.

Exempel 3 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner som laddas i en tabell som heter `Transactions`.
- Tillhandahållna transaktionstider som tillhandahålls över två fält: `hours` och `minutes`.
- Ett fält `transaction_time` skapas som returnerar tiden i formatet för systemvariabeln `TimeFormat`.

Skapa ett diagramobjektmått, `transaction_time`, som returnerar en tid i formatet `h:mm:ss TT`.

Laddningsskript

```
SET TimeFormat='h:mm:ss TT';

Transactions:
Load * Inline [
transaction_id, transaction_hour, transaction_minute, transaction_amount, transaction_
quantity, customer_id
3750, 18, 43, 12423.56, 23, 2038593
3751, 6, 32, 5356.31, 6, 203521
3752, 12, 09, 15.75, 1, 5646471
3753, 21, 43, 7, 3036491
3754, 17, 55, 21484.21, 1356, 049681
3756, 2, 52, -59.18, 2, 2038593
3757, 9, 25, 3177.4, 21, 203521
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `transaction_hour`
- `transaction_minute`

För att beräkna `transaction_time` skapar du det här måttet:

```
=maketime(transaction_hour,transaction_minute)
```

Resultattabell

transaction_hour	transaction_minute	=maketime(transaction_hour, transaction_minute)
2	52	2:52:00 AM
6	32	6:32:00 AM
9	25	9:25:00 AM
12	09	12:09:00 PM
17	55	5:55:00 PM
18	43	6:43:00 PM
21	43	9:43:00 PM

`transaction_time`-måttet skapades i den föregående load-satsen genom att använda `maketime()`-funktionen och skicka tim- och minutfälten som funktionens argument.

Funktionen kombinerar sedan dessa värden, och sekunderna antas vara 00. De här värdena konverteras sedan till ett tidsfält och returnerar resultatet i form av systemvariabeln `TimeFormat`.

Exempel 4 – Scenario

Laddningsskript och diagramuttryck

Översikt

Skapa en kalenderdatauppsättning för januari månad 2022, uppbruten i åttatimmarssteg.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

tmpCalendar:
       load
               *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 31;

Left join(tmpCalendar)
load
       maketime((recno()-1)*8,00,00) as time
autogenerate 3;

Calendar:
load
       timestamp(date + time) as timestamp
resident tmpCalendar;

drop table tmpCalendar;
```

Resultat

Resultattabell

tidsmarkör
1/1/2022 12:00:00 AM
1/1/2022 8:00:00 AM
1/1/2022 4:00:00 PM
1/2/2022 12:00:00 AM
1/2/2022 8:00:00 AM
1/2/2022 4:00:00 PM
1/3/2022 12:00:00 AM
1/3/2022 8:00:00 AM
1/3/2022 4:00:00 PM
1/4/2022 12:00:00 AM
1/4/2022 8:00:00 AM
1/4/2022 4:00:00 PM
1/5/2022 12:00:00 AM
1/5/2022 8:00:00 AM
1/5/2022 4:00:00 PM
1/6/2022 12:00:00 AM
1/6/2022 8:00:00 AM
1/6/2022 4:00:00 PM
1/7/2022 12:00:00 AM
1/7/2022 8:00:00 AM
1/6/2022 4:00:00 PM
1/8/2022 12:00:00 AM
1/8/2022 8:00:00 AM
1/6/2022 4:00:00 PM
1/9/2022 12:00:00 AM
+ ytterligare 68 rader

Den initiala `autogenerate`-funktionen skapar en kalender som innehåller alla datum i januari i en tabell som heter `tmpCalendar`.

Ytterligare en tabell som innehåller tre poster skapas. För varje post tas `recno() - 1` (värdena 0, 1, 2) och resultatet multipliceras med 8. Detta ger värdena 0, 8 16 som resultat. De här värdena används som timparameter i en `maketime()`-funktion, med minut- och sekundvärden som 0. Detta resulterar i att tabellen innehåller tre tidsfält: 12:00:00 AM, 8:00:00 AM, and 4:00:00 PM.

Den här tabellen länkas till `tmpCalendar`-tabellen. Eftersom det inte finns några matchande fält mellan de två tabellerna för länkningen läggs tidsraderna till i varje datarad. Detta resulterar i att varje datarad nu repeteras tre gånger för varje tidsvärde.

Slutligen skapas kalendertabellen från en resident load i `tmpCalendar`-tabellen. Datum- och tidsfälten konkateneras och omsluts i `timestamp()`-funktionen för att skapa en tidsmarkörsfältet.

`tmpCalendar`-tabellen släpps sedan här.

makeweekdate

Funktionen returnerar ett datum som beräknats från året, veckonumret och veckodagen.

Syntax:

```
MakeWeekDate(weekyear [, week [, weekday [, first_week_day [, broken_weeks [, reference_day]]]]])
```

Returnerad datatyp: dual

`makeweekdate()`-funktionen är tillgänglig både som skript- och diagramfunktionen. Funktionen beräknar datumet baserat på parametrarna som

Argument

Argument	Beskrivning
weekyear	Det årtal som definieras av `WeekYear()`-funktionen för det specifika datumet, dvs. det årtal som veckonumret tillhör. *Veckoåret kan i vissa fall skilja sig från kalenderåret, till exempel om vecka 1 börjar redan i december året innan.*
week	Det veckonummer som definieras av `Week()`-funktionen för det specifika datumet. Om inget veckonummer anges antas 1.

Argument	Beskrivning
weekday	Veckodagen enligt definitionen i `WeekDay()`-funktionen för datumet i fråga. 0 är veckans första dag och 6 är veckans sista dag. Om ingen veckodag anges antas 0. *Även om 0 alltid är den första veckodagen och 6 alltid den sista, bestäms vilka veckodagar som motsvarar detta av parametern **first_week_day**. Om utelämnat används värdet för variabeln **FirstWeekDay**.* Om brutna veckor används tillsammans med en omöjlig kombination av parametrar kan detta leda till ett resultat som inte hör till det valda året. **Exempel:** `MakeWeekDate(2021,1,0,6,1)` Returnerar "27 dec 2020" eftersom denna dag är den första dagen (söndagen) i den angivna veckan. Den 1 januari 2021 var en fredag.
first_week_day	Anger dagen då veckan startar. Om utelämnat används värdet för variabeln **FirstWeekDay**. De möjliga värdena **first_week_day** är 0 för måndag, 1 för tisdag, 2 för onsdag, 3 för torsdag, 4 för fredag, 5 för lördag och 6 för söndag. Se *FirstWeekDay (page 257)* för mer information om systemvariabeln.
broken_weeks	Om du inte anger **broken_weeks** används värdet för variabeln **BrokenWeeks** till att definiera om veckor är brutna eller inte.
reference_day	Om du inte anger **reference_day** används värdet för variabeln **ReferenceDay** till att definiera vilken dag i januari som ska ställas in som referensdag för att definiera vecka 1.

Användning

`makeweekdate()`-funktionen would commonly be used in the script for data generation to generate a list of dates, or to construct dates when the year, week and day-of-week are provided in the input data.

Följande exempel antar:

```
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;
```

Exempel på funktioner

Exempel	Resultat
`makeweekdate(2014,6,6)`	returnerar 02/09/2014
`makeweekdate(2014,6,1)`	returnerar 02/04/2014
`makeweekdate(2014,6)`	returnerar 02/03/2014 (veckodag 0 antas)

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – dag inkluderad

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller total försäljning per vecka för 2022 i en tabell som heter `Sales`.
- Tillhandahållna transaktionsdatum över tre fält: `year`, `week` och `sales`.
- En föregående last som används för att skapa ett mått, `end_of_week`, med hjälp av `makeweekdate()`-funktionen för att returnera datumet för fredagen den veckan i formatet MM/DD/YYYY.

För att bewvisa att det returnerade datumet är en fredag är `end_of_week`-uttrycket också inneslutet i `weekday()`-funktionen så att veckodagen visas.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;

Transactions:
    Load
        *,
        makeweekdate(transaction_year, transaction_week,4) as end_of_week,
        weekday(makeweekdate(transaction_year, transaction_week,4)) as week_day
        ;
Load * Inline [
transaction_year, transaction_week, sales
2022, 01, 10000
2022, 02, 11250
2022, 03, 9830
```

```
2022, 04, 14010
2022, 05, 28402
2022, 06, 9992
2022, 07, 7292
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `transaction_year`
- `transaction_week`
- `end_of_week`
- `week_day`

Resultattabell

transaktionsår	transaktionsvecka	veckoslut	week_day
2022	1	01/07/2022	fre
2022	2	14/01/2022	fre
2022	3	21/01/2022	fre
2022	4	28/01/2022	fre
2022	5	02/04/2022	fre
2022	6	2022-02-11	fre
2022	7	2022-02-18	fre

`end_of_week`-fältet skapas i föregående LOAD-sats med hjälp av funktionen `makeweekdate()`. `transaction_year`- och `transaction_week`-fälten skickas genom funktionen som argument för år resp. vecka. Värdet 4 används för dagargumentet.

Funktionen kombinerar och konverterar sedan de här värdena till ett datumfält och returnerar resultatet i form av systemvariabeln `DateFormat`.

`makeweekdate()`-funktionen och dess argument är också omslutna av en `weekday()`-funktion för att returnera `week_day`-fältet och kan ses i tabellen ovan. `week_day`-fältet visar att de här datumen infaller på en fredag.

Exempel 2 – dag exkluderad

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller total försäljning per vecka för 2022 i en tabell som heter `Sales`.
- Tillhandahållna transaktionsdatum över tre fält: `year`, `week` och `sales`.
- En föregående laddning, som används för att skapa ett mått `first_day_of_week` med `makeweekdate()`. Då returneras datumet för måndagen i den veckan med formatet MM/DD/YYYY.

För att bevisa att det returnerade datumet är en måndag är `first_day_of_week`-uttrycket också inneslutet i `weekday()`-funktionen så att veckodagen visas.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;

Transactions:
    Load
        *,
        makeweekdate(transaction_year, transaction_week) as first_day_of_week,
        weekday(makeweekdate(transaction_year, transaction_week)) as week_day
        ;
Load * Inline [
transaction_year, transaction_week, sales
2022, 01, 10000
2022, 02, 11250
2022, 03, 9830
2022, 04, 14010
2022, 05, 28402
2022, 06, 9992
2022, 07, 7292
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `transaction_year`
- `transaction_week`
- `first_day_of_week`
- `week_day`

Resultattabell

transaktionsår	transaktionsvecka	veckans första dag	week_day
2022	1	01/03/2022	mån
2022	2	01/10/2022	mån
2022	3	17/01/2022	mån
2022	4	01/24/2022	mån

transaktionsår	transaktionsvecka	veckans första dag	week_day
2022	5	01/31/2022	mån
2022	6	02/07/2022	mån
2022	7	02/14/2022	mån

`first_day_of_week`-fältet skapas i föregående LOAD-sats med hjälp av funktionen `makeweekdate()`. Parametrarna `transaction_year` och `transaction_week` skickas som funktionsargument, och dagparametern lämnas blank.

Funktionen kombinerar och konverterar sedan de här värdena till ett datumfält och returnerar resultatet i form av systemvariabeln `DateFormat`.

`makeweekdate()`-funktionen och dess argument innesluts också i en `weekday()`-funktion som returnerar `week_day`-fältet. Som framgår av tabellen ovan returnerar fältet `week_day`måndag i alla fall eftersom den parametern lämnades tom i funktionen `makeweekdate()`, som är standardiserad till `0` (veckans första dag), och veckans första dag anges som måndag av `FirstWeekDay`-systemvariabeln.

Exempel 3 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller total försäljning per vecka för 2022 i en tabell som heter `Sales`.
- Tillhandahållna transaktionsdatum över tre fält: `year`, `week` och `sales`.

I det här exemplet kommer ett diagramobjekt att användas för att skapa ett mått som motsvarar `end_of_week`-beräkningen i det första exemplet. Det här måttet kommer att använda `makeweekdate()`-funktionen för att returnera datumet för fredagen i den veckan med formatet MM/DD/YYYY.

För att bevisa att datumet som returneras är en fredag skapas ytterligare ett mått som returnerar veckodagen.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;

Master_Calendar:
Load * Inline [
transaction_year, transaction_week, sales
2022, 01, 10000
2022, 02, 11250
```

```
2022, 03, 9830
2022, 04, 14010
2022, 05, 28402
2022, 06, 9992
2022, 07, 7292
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:
   - `transaction_year`
   - `transaction_week`
2. För att utföra beräkningen som motsvarar den i `end_of_week`-fältet från det första exemplet skapar du följande mått:
   `=makeweekdate(transaction_year,transaction_week,4)`
3. För att beräkna veckostart för den vecka för varje transaktion skapar du följande mått:
   `=weekday(makeweekdate(transaction_year,transaction_week,4))`

Resultattabell

transaktionsår	transaktionsvecka	=makeweekdate (transaktionsår,transaktionsvecka,4)	=weekday(makeweekdate (transaktionsår,transaktionsvecka,4))
2022	1	01/07/2022	fre
2022	2	14/01/2022	fre
2022	3	21/01/2022	fre
2022	4	28/01/2022	fre
2022	5	02/04/2022	fre
2022	6	2022-02-11	fre
2022	7	2022-02-18	fre

Ett fält som motsvarar `end_of_week` skapas i som ett mått i diagramobjektet med hjälp av `makeweekdate()`-funktionen. `transaction_year`- och `transaction_week`-fälten skickas som argument för år resp. vecka. Värdet 4 används för dagargumentet.

Funktionen kombinerar och konverterar sedan de här värdena till ett datumfält och returnerar resultatet i form av systemvariabeln `DateFormat`.

`makeweekdate()`-funktionen och dess argument innesluts också i en `weekday()`-funktion som returnerar en beräkningsmotsvarighet till den i `week_day`-fältet i det första exemplet. Som framgår av ovanstående tabell visar den sista kolumnen till höger att de här datumen inträffar på en fredag.

Exempel 4 – Scenario

Laddningsskript och diagramuttryck

Översikt

I det här exemplet skapas en lista över datum som innehåller alla fredagar under 2022.

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;

Calendar:
    load
           *,
        weekday(date) as weekday
    where year(date)=2022;
load
    makeweekdate(2022,recno()-2,4) as date
AutoGenerate 60;
```

Resultat

Resultattabell

datum	veckodag
01/07/2022	fre
2022-01-14	fre
2022-01-21	fre
2022-01-28	fre
02/04/2022	fre
2022-02-11	fre
2022-02-18	fre
02/25/2022	fre
2022-03-04	fre
03/11/2022	fre
2022-03-18	fre
03/25/2022	fre

datum	veckodag
04/01/2022	fre
04/08/2022	fre
04/15/2022	fre
04/22/2022	fre
04/29/2022	fre
05/06/2022	fre
2022-05-13	fre
2022-05-20	fre
05/27/2022	fre
06/03/2022	fre
06/10/2022	fre
2022-06-17	fre
+ ytterligare 27 rader	

`makeweekdate()`-funktionen hittar alla fredagar under 2022. Veckoparametern -2 säkerställer att inga datum missas. Slutligen skapar en föregående laddning ett extra `weekday`-fält för tydlighet, som visar att varje `date`-värde är en fredag.

minute

Denna funktion returnerar ett heltal som motsvarar minuten om decimaldelen av **expression** tolkas som tid enligt standardtolkningen av tal.

Syntax:

```
minute(expression)
```

Returnerad datatyp: heltal

Användning

Funktionen `minute()` är användbar när du vill jämföra sammanställningar efter minuter. Du kan till exempel använda funktionen om du vill visa aktivitetsräkningsfördelningen efter minuter.

Dessa dimensioner kan antingen skapas i laddningsskriptet genom att använda funktionen för att skapa ett fält i en Master Calendar-tabell. Eller användas direkt i ett diagram som en beräknad dimension.

Exempel på funktioner

Exempel	Resultat
`minute ( '09:14:36' )`	Returnerar 14.
`minute ( '0.5555' )`	Returnerar 19 (eftersom 0.5555 = 13:19:55)

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Variabel (skript)

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner efter tidsmarkör som läses in i en tabell som heter `Transactions`.
- `TimeStamp`-standardsystemvariabeln (`M/D/YYYY h:mm:ss[.fff] TT`) används.
- Ett fält `minute` skapas för att beräkna när transaktionerna ägde rum.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        minute(timestamp) as minute
        ;
Load
*
Inline
[
id,timestamp,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
9499,'2022-01-03 05:40:49',73.53,
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
```

```
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `timestamp`
- `minute`

Resultattabell

tidsmarkör	minut
2022-01-01 22:10:22	10
2022-01-02 08:35:54	35
2022-01-03 05:40:49	40
2022-01-03 14:21:53	21
2022-01-04 18:49:38	49
2022-01-04 22:58:34	58
2022-01-05 19:04:57	4
2022-01-05 19:34:46	34
2022-01-06 08:49:09	49
2022-01-06 11:29:38	29

Värdena i `minute`-fältet skapas genom att använda `minute()`-funktionen och skicka `timestamp` som uttrycket i föregående load-sats.

Exempel 2 – Diagramobjekt (diagram)

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- `TimeStamp`-standardsystemvariabeln (`M/D/YYYY h:mm:ss[.fff] TT`) används.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Värdena på `minute` beräknas via ett mått i ett diagramobjekt.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
Load
*
Inline
[
id,timestamp,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
9499,'2022-01-03 05:40:49',73.53,
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `timestamp`.

Skapa följande mått:

```
=minute(timestamp)
```

Resultattabell

tidsmarkör	minut
2022-01-01 22:10:22	10
2022-01-02 08:35:54	35
2022-01-03 05:40:49	40
2022-01-03 14:21:53	21
2022-01-04 18:49:38	49
2022-01-04 22:58:34	58
2022-01-05 19:04:57	4
2022-01-05 19:34:46	34
2022-01-06 08:49:09	49
2022-01-06 11:29:38	29

Värdena för `minute` skapas genom att använda funktionen `minute()` och skicka `timestamp` som ett uttryck i ett mått för diagramobjektet.

Exempel 3 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med tidsmarkörer som genereras för att representera poster vid en biljettspärr.
- Information med varje `timestamp` och dess motsvarande `id`, som laddas in i en tabell som heter `Ticket_Barrier_Tracker`.
- `TimeStamp`-standardsystemvariabeln (`M/D/YYYY h:mm:ss[.fff] TT`) används.

Användaren vill ha ett diagramobjekt som visar antalet spärrposter per minut.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

tmpTimeStampCreator:
       load
                *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 1;

join load
       maketime(floor(rand()*24),floor(rand()*59),floor(rand()*59)) as time
autogenerate 10000;

Ticket_Barrier_Tracker:
load
       recno() as id,
       timestamp(date + time) as timestamp
resident tmpTimeStampCreator;

drop table tmpTimeStampCreator;
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell.
2. Skapa en beräknad dimension med följande uttryck:
   `=minute(timestamp)`
3. Lägg till följande aggregeringsmått för att beräkna det totala antalet poster.

```
=count(id)
```

4. Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

minut(tidsmarkör)	=antal(id)
0	174
1	171
2	175
3	165
4	188
5	176
6	158
7	187
8	178
9	178
10	197
11	161
12	166
13	184
14	159
15	161
16	152
17	160
18	176
19	164
20	170
21	170
22	142
23	145
24	155
+ ytterligare 35 rader	

month

Denna funktion returnerar ett dualt värde: månadsnamnet som det är definierat i miljövariabeln **MonthNames** och ett heltal mellan 1 och 12. Månaden beräknas utifrån datumtolkningen av uttrycket, enligt standardtalformatet.

Funktionen returnerar månadens namn i formatet för systemvariabeln `MonthName` för ett visst datum. Den används vanligtvis för att skapa ett dagfält som en dimension i en Master Calendar.

Syntax:

```
month(expression)
```

Returnerad datatyp: heltal

Exempel på funktioner

Exempel	Resultat
`month( 2012-10-12 )`	returnerar Oct
`month( 35648 )`	returnerar Aug, eftersom 35648 = 1997-08-06

Exempel 1 – DateFormat datauppsättning (skript)

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- En datauppsättning av datum benämnd `Master_Calendar`. Systemvariabeln `DateFormat` är inställd på DD/MM/ÅÅÅÅ.
- En föregående laddning som skapar ett ytterligare fält benämnt `month_name`, med användning av `month()`-funktionen.
- Ett ytterligare fält benämnt `long_date` som använder `date()`-funktionen för att uttrycka hela datumet.

Laddningsskript

```
SET DateFormat='DD/MM/YYYY';

Master_Calendar:
Load
        date,
        date(date,'dd-MMMM-YYYY') as long_date,
        month(date) as month_name
Inline
```

```
[
date
03/01/2022
03/02/2022
03/03/2022
03/04/2022
03/05/2022
03/06/2022
03/07/2022
03/08/2022
03/09/2022
03/10/2022
03/11/2022
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `long_date`
- `month_name`

Resultattabell

datum	långt_datum	month_name
2022-03-01	03-januari-2022	Jan
03/02/2022	03-februari- 2022	Feb
03/03/2022	03-mars- 2022	Mar
2022-03-04	03-april- 2022	Apr
03/05/2022	03-maj- 2022	Maj
03/06/2022	03-juni- 2022	Jun
03/07/2022	03-juli- 2022	jul
03/08/2022	03-augusti- 2022	aug
03/09/2022	03-september- 2022	sep
03/10/2022	03-oktober- 2022	okt
03/11/2022	03-november- 2022	nov

Månadens namn är korrekt utvärderat av `month()`-funktionen i skriptet.

Exempel 2 – ANSI-datum (skript)

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- En datauppsättning av datum benämnd `Master_Calendar`. DateFormat-systemvariabeln DD/MM/ÅÅÅÅ används. Emellertid är de datum som är inkluderade i datauppsättningen i ANSI:s standarddatumformat.
- En föregående laddning som skapar ett ytterligare fält benämnt month_name med hjälp av month()-funktionen.
- Ett ytterligare fält benämnt long_date som använder date()-funktionen för att uttrycka hela datumet.

Laddningsskript

```
SET DateFormat='DD/MM/YYYY';
Master_Calendar:
Load
       date,
       date(date,'dd-MMMM-YYYY') as long_date,
       month(date) as month_name

Inline
[
date
2022-01-11
2022-02-12
2022-03-13
2022-04-14
2022-05-15
2022-06-16
2022-07-17
2022-08-18
2022-09-19
2022-10-20
2022-11-21
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `long_date`
- `month_name`

Resultattabell

datum	långt_datum	month_name
03/11/2022	11-mars- 2022	11
03/12/2022	12-mars- 2022	12
03/13/2022	13-mars- 2022	13
03/14/2022	14-mars- 2022	14
03/15/2022	15-mars- 2022	15
03/16/2022	16-mars- 2022	16
03/17/2022	17-mars- 2022	17
2022-03-18	18-mars- 2022	18
2022-03-19	19-mars- 2022	19
03/20/2022	20-mars- 2022	20
03/21/2022	21-mars- 2022	21

Månadens namn är korrekt utvärderat av `month()`-funktionen i skriptet.

Exempel 3 – Oformaterade datum (skript)

Laddningsskript och resultat

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- En datauppsättning av datum benämnd `Master_Calendar`. `DateFormat`-systemvariabeln DD/MM/ÅÅÅÅ används.
- En föregående laddning som skapar ett ytterligare fält benämnt `month_name`, med hjälp av `month()` -funktionen.
- Det ursprungliga oformaterade datumet, benämnt `unformatted_date`.
- Ett ytterligare fält benämnt `long_date`, som använder `date()`-funktionen för att uttrycka hela datumet.

Laddningsskript

```
SET DateFormat='DD/MM/YYYY';

Master_Calendar:
Load
      unformatted_date,
      date(unformatted_date,'dd-MMMM-YYYY') as long_date,
      month(unformatted_date) as month_name
```

```
Inline
[
unformatted_date
44868
44898
44928
44958
44988
45018
45048
45078
45008
45038
45068
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `unformatted_date`
- `long_date`
- `month_name`

Resultattabell

oformaterat_datum	långt_datum	month_name
44868	03-januari-2022	Jan
44898	03-februari- 2022	Feb
44928	03-mars- 2022	Mar
44958	03-april- 2022	Apr
44988	03-maj- 2022	Maj
45018	03-juni- 2022	Jun
45048	03-juli- 2022	jul
45078	03-augusti- 2022	aug
45008	03-september- 2022	sep
45038	03-oktober- 2022	okt
45068	03-november- 2022	nov

Månadens namn är korrekt utvärderat av `month()`-funktionen i skriptet.

Exempel 4 – Beräkna utgångsmånad

Laddningsskript och diagramuttryck

Översikt

Öppna Skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller:

- En datauppsättning av ordrar lagda i mars, benämnda `Subscriptions`. Tabellen innehåller tre fält:
  - id
  - order_date
  - belopp

Laddningsskript

```
Subscriptions:
Load
        id,
        order_date,
        amount
Inline
[
id,order_date,amount
1,03/01/2022,231.24
2,03/02/2022,567.28
3,03/03/2022,364.28
4,03/04/2022,575.76
5,03/05/2022,638.68
6,03/06/2022,785.38
7,03/07/2022,967.46
8,03/08/2022,287.67
9,03/09/2022,764.45
10,03/10/2022,875.43
11,03/11/2022,957.35
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `order_date`.

För att beräkna månaden när en beställning förfaller skapar du det här måttet: `=month(order_date+180)`.

Resultattabell

order_date	=month(order_date+180)
2022-03-01	jul

order_date	=month(order_date+180)
03/02/2022	aug
03/03/2022	aug
2022-03-04	sep
03/05/2022	okt
03/06/2022	nov
03/07/2022	dec
03/08/2022	Jan
03/09/2022	Mar
03/10/2022	Apr
03/11/2022	Maj

Funktionen `month()` utvärderar korrekt att en order som läggs den 11 mars går ut i juli.

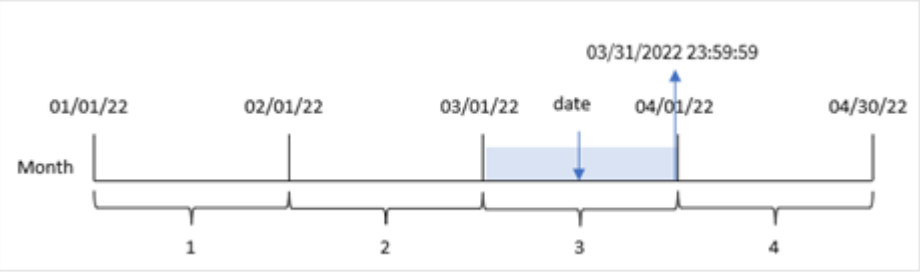
monthend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör med den sista millisekunden av den sista dagen i den månad som innehåller `date`. Det förvalda utdataformatet blir det `DateFormat` som har definierats i skriptet.

```
MonthEnd(date[, period_no])
```

Med andra ord bestämmer `monthend()`-funktionen vilket år datumet infaller. Den returnerar sedan en tidsmarkör, i datumformat, för den sista millisekunden av den månaden.

Diagram för `monthend`-funktionen.



Användning

`monthend()`-funktionen används som en del av ett uttryck när du vill att beräkningen ska använda den del av månaden som ännu inte har inträffat. Till exempel om du vill beräkna den totala ränta som ännu inte uppkommit under månaden.

Returnerad datatyp: Dual

Argument

Argument	Beskrivning
date	Det datum eller den tidsmarkör som ska utvärderas.
period_no	**period_no** är ett heltal som om det är 0 eller utelämnas anger månaden som innehåller **date**. Negativa värden i **period_no** anger föregående månader och positiva värden anger efterföljande månader.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`monthend('02/19/2012')`	Returnerar 02/29/2012 23:59:59.
`monthend('02/19/2001', -1)`	Returnerar 01/31/2001 23:59:59.

Exempel 1 – Grundläggande exempel

Ladda skript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner för 2022 som läses in i en tabell som heter ”`Transactions`”.
- Ett datumfält i systemvariabeln `DateFormat` i formatet (`MM/DD/YYYY`).
- En föregående LOAD-sats som innehåller:

- `monthend()`-funktionen som är inställt som ‘`end_of_month`’ fältet.
- `timestamp`-funktionen som är inställt som ‘`end_of_month_timestamp`’ fältet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    monthend(date) as end_of_month,
    timestamp(monthend(date)) as end_of_month_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `end_of_month`
- `end_of_month_timestamp`

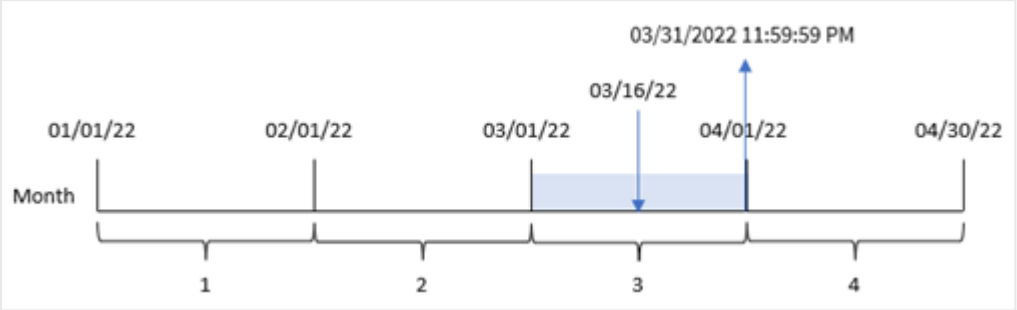
Resultattabell

id	datum	end_of_month	end_of_month_timestamp
8188	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8189	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8199	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8200	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8201	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

‘`end_of_month`’-fältet skapas i den föregående load-satsen genom att använda `monthend()`-funktionen och skicka datumfältet som funktionens argument.

`monthend()`-funktionen identifierar vilken månad datumvärdet infaller och returnerar en tidsmarkör för den sista millisekunden av den månaden.

Diagram för `monthend`-funktionen med mars som vald månad.



Transaktion 8192 ägde rum den 16 mars. `monthend()`-funktionen returnerar den sista millisekunden i den månaden, vilket är den 31 mars 23:59:59.

Exempel 2 – period_no

Ladda skript och resultat

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är uppgiften att skapa ett fält, `previous_month_end`, som returnerar tidsmarkören för månadsslutet innan transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    monthend(date,-1) as previous_month_end,
    timestamp(monthend(date,-1)) as previous_month_end_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
```

```
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `previous_month_end`
- `previous_month_end_timestamp`

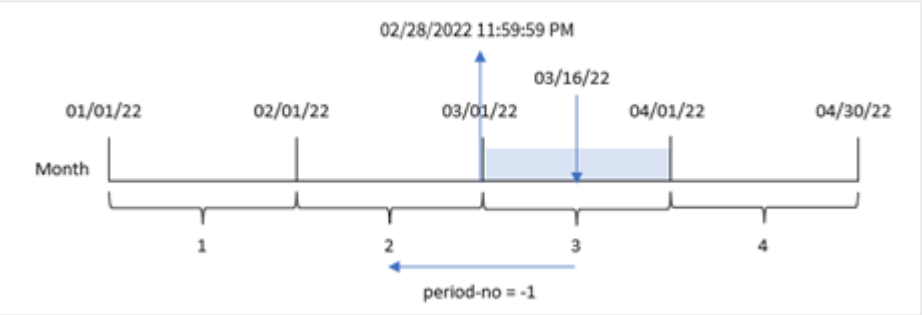
Resultattabell

id	datum	previous_month_end	previous_month_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	01/31/2022	1/31/2022 11:59:59 PM
8191	2/28/2022	01/31/2022	1/31/2022 11:59:59 PM
8192	3/16/2022	02/28/2022	2/28/2022 11:59:59 PM
8193	4/1/2022	03/31/2022	3/31/2022 11:59:59 PM
8194	5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
8195	5/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8196	6/15/2022	05/31/2022	5/31/2022 11:59:59 PM
8197	6/26/2022	05/31/2022	5/31/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	07/31/2022	7/31/2022 11:59:59 PM
8203	8/8/2022	07/31/2022	7/31/2022 11:59:59 PM
8204	8/19/2022	07/31/2022	7/31/2022 11:59:59 PM

id	datum	previous_month_end	previous_month_end_timestamp
8205	9/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8206	10/14/2022	09/30/2022	9/30/2022 11:59:59 PM
8207	10/29/2022	09/30/2022	9/30/2022 11:59:59 PM

Funktionen `monthend()` identifierar först den månad då transaktionerna äger rum eftersom ett `period_no` på `-1` används som förskjutningsargument. Den skiftar sedan till närmast föregående månad och identifierar den sista millisekunden i den månaden.

Diagram för `monthend`-funktionen med variabeln period_no.



Transaktion 8192 ägde rum den 16 mars. `monthend()`-funktionen identifierar att månaden innan transaktionen ägde rum var februari. Den returnerar sedan den sista millisekunden den månaden, 28 februari 23:59:59.

Exempel 3 – Diagramexempel

Laddningsskript och diagramuttryck

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är datauppsättningen oförändrad och har lästs in i applikationen. Uppgiften är att skapa en beräkning som returnerar en tidsmarkör för slutet av månaden då transaktionerna ägde rum som ett mått i ett diagram för applikationen.

Laddningsskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
```

```
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `id`

För att beräkna slutdatum för den månad då en transaktion äger rum skapar du följande mått:

- `=monthend(date)`
- `=timestamp(monthend(date))`

Resultattabell

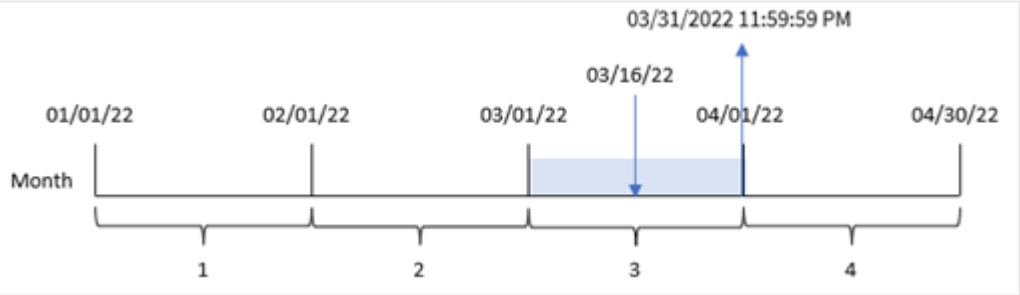
id	datum	=monthend(date)	=timestamp(monthend(date))
8188	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8189	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM
8190	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8191	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8192	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8193	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8194	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8195	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8196	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8197	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8198	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM

id	datum	=monthend(date)	=timestamp(monthend(date))
8199	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8201	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8202	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8203	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8204	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8205	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8206	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8207	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM

`end_of_month`-måttet skapades i diagrammet genom att använda `monthend()`-funktionen och skicka datumfältet som funktionens argument.

`monthend()`-funktionen identifierar vilken månad datumvärdet infaller och returnerar en tidsmarkör för den sista millisekunden av den månaden.

Diagram för `monthend`-funktionen med variabeln period_no.



Transaktion 8192 ägde rum den 16 mars. `monthend()`-funktionen returnerar den sista millisekunden i den månaden, vilket är den 31 mars 23:59:59.

Exempel 4 – Scenario

Laddningsskript och resultat

Översikt

I det här exemplet laddas en datauppsättning i en tabell som heter `Employee_Expenses`. Tabellen innehåller följande fält:

- Anställnings-ID
- Anställdas namn
- Genomsnittliga dagliga kostnadsanspråk för varje anställd.

Slutanvändaren vill ha ett diagram som visar, efter anställnings-id och anställds namn, de beräknade kostnadsanspråken för resten av månaden.

Laddningsskript

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `employee_id`
- `employee_name`

För att beräkna ackumulerad ränta skapar du detta mått:

```
=floor(monthend(today(1),0)-today(1))*avg_daily_claim
```

> *Det här måttet är dynamiskt och kommer att ge olika tabellresultat beroende på vilket datum du laddar uppgifterna.*

Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

employee_id	employee_name	=floor(monthend(today(1),0)-today(1))*avg_daily_claim
182	Mark	$30.00
183	Deryck	$25.00
184	Dexter	$25.00
185	Sydney	$54.00
186	Agatha	$36.00

`monthend()`-funktionen returnerar slutdatumet för den aktuella månaden genom att använda dagens datum som sitt enda argument. Uttrycket returnerar antalet dagar som återstår denna månad genom att subtrahera dagens datum från månadens slutdatum.

Detta värde multipliceras sedan med det genomsnittliga dagliga kostnadsanspråket från varje anställd för att beräkna det uppskattade värdet av anspråk som varje anställd förväntas göra under den återstående månaden.

monthname

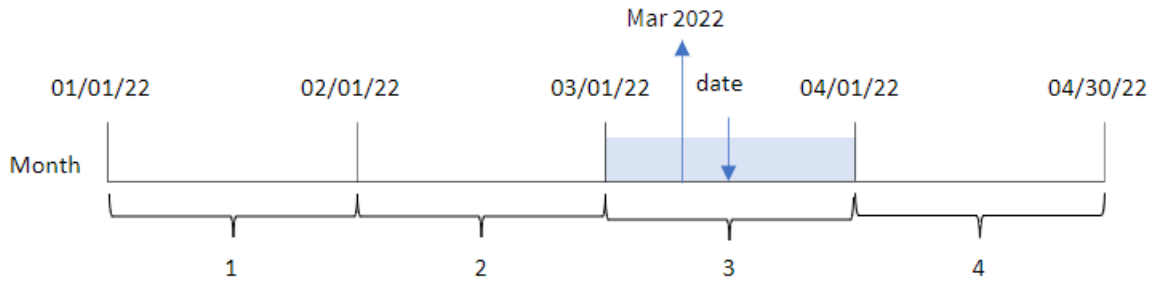
Denna funktion returnerar ett visningsvärde med månaden (formaterat enligt skriptvariabeln **MonthNames**) och året med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden av månadens första dag.

Syntax:

```
MonthName(date[, period_no])
```

Returnerad datatyp: dual

Diagram för monthname-funktionen



Argument

Argument	Beskrivning
date	Det datum eller den tidsmarkör som ska utvärderas.
period_no	**period_no** är ett heltal som om det är 0 eller utelämnas anger månaden som innehåller **date**. Negativa värden i **period_no** anger föregående månader och positiva värden anger efterföljande månader.

Exempel på funktioner

Exempel	Resultat
`monthname('10/19/2013')`	Returnerar Oct 2013
`monthname('10/19/2013', -1)`	Returnerar Sep 2013

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan

vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Grundläggande exempel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2022 som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i `DateFormat`-systemvariabeln.
- Skapandet av ett fält, `transaction_month`, som returnerar månaden då transaktionerna ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        monthname(date) as transaction_month
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
```

```
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

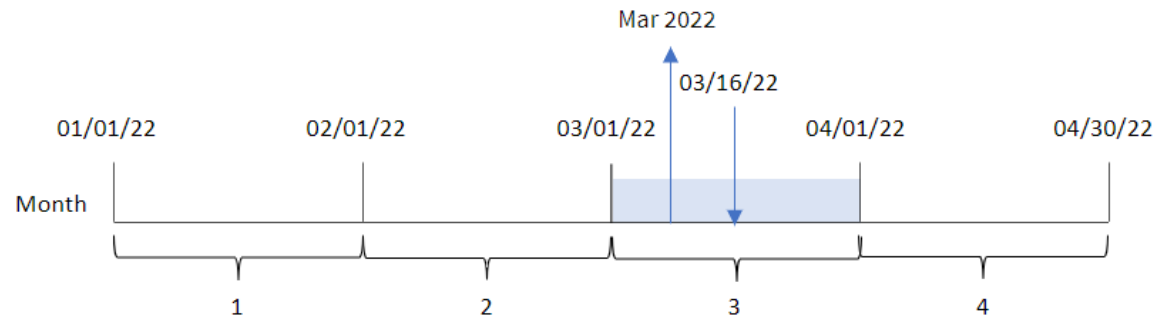
- `date`
- `transaction_month`

Resultattabell

datum	transaction_month
1/7/2022	Jan 2022
1/19/2022	Jan 2022
2/5/2022	Feb 2022
2/28/2022	Feb 2022
3/16/2022	Mar 2022
4/1/2022	Apr 2022
5/7/2022	Maj 2022
5/16/2022	Maj 2022
6/15/2022	Jun 2022
6/26/2022	Jun 2022
7/9/2022	Jul 2022
7/22/2022	Jul 2022
7/23/2022	Jul 2022
7/27/2022	Jul 2022
8/2/2022	Aug 2022
8/8/2022	Aug 2022
8/19/2022	Aug 2022
9/26/2022	Sep 2022
10/14/2022	Okt 2022
10/29/2022	Okt 2022

`transaction_month`-fältet skapades i den föregående load-satsen genom att använda `monthname()`-funktionen och skicka `date`-fältet som funktionens argument.

Diagram för monthname-funktionen



`monthname()`-funktionen identifierar att transaktion 8192 ägde rum i mars 2022 och returnerar detta värde med hjälp av systemvariabeln `MonthNames`.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma inline-datauppsättning och scenario som det första exemplet används.
- Skapandet av ett fält, `transaction_previous_month`, som returnerar tidsmarkören för månadsslutet före transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        monthname(date,-1) as transaction_previous_month
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
```

```
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `transaction_previous_month`

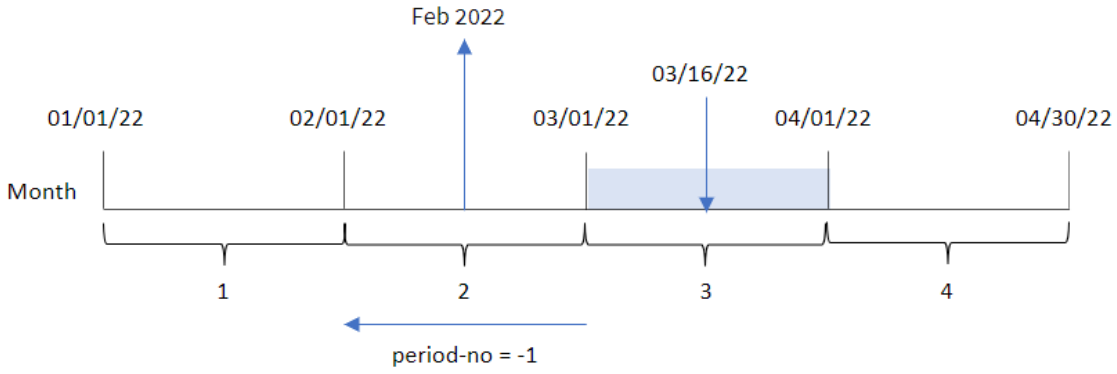
Resultattabell

datum	transaction_previous_month
1/7/2022	Dec 2021
1/19/2022	Dec 2021
2/5/2022	Jan 2022
2/28/2022	Jan 2022
3/16/2022	Feb 2022
4/1/2022	Mar 2022
5/7/2022	Apr 2022
5/16/2022	Apr 2022
6/15/2022	Maj 2022
6/26/2022	Maj 2022
7/9/2022	Jun 2022
7/22/2022	Jun 2022
7/23/2022	Jun 2022
7/27/2022	Jun 2022

datum	transaction_previous_month
8/2/2022	Jul 2022
8/8/2022	Jul 2022
8/19/2022	Jul 2022
9/26/2022	Aug 2022
10/14/2022	Sep 2022
10/29/2022	Sep 2022

I det här fallet, eftersom ett `period_no` på -1 användes som förskjutningsargument i `monthname()`-funktionen, identifierar funktionen först den månad då transaktionerna äger rum. Den skiftar sedan till en månad innan och returnerar månadens namn och år.

Exempel på diagram för monthname-funktionen, period_no



Transaktion 8192 ägde rum den 16 mars. `monthname()`-funktionen identifierar att månaden innan transaktionen ägde rum var februari och returnerar månaden, i systemvariabelformatet `MonthNames` tillsammans med år 2022.

Exempel 3 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet. I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som returnerar en tidsmarkör för slutet av månaden då transaktionerna ägde rum skapas som ett mått i ett diagramobjekt för programmet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:`date`.

Skapa följande mått:

```
=monthname(date)
```

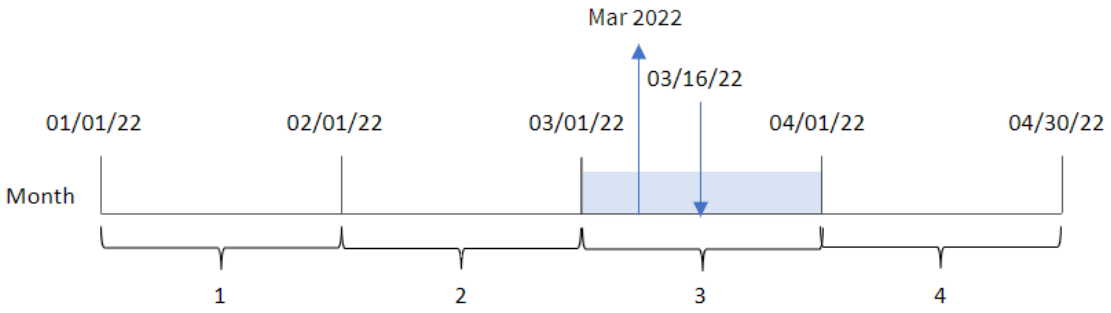
Resultattabell

datum	=monthname(date)
1/7/2022	Jan 2022
1/19/2022	Jan 2022
2/5/2022	Feb 2022
2/28/2022	Feb 2022
3/16/2022	Mar 2022

datum	=monthname(date)
4/1/2022	Apr 2022
5/7/2022	Maj 2022
5/16/2022	Maj 2022
6/15/2022	Jun 2022
6/26/2022	Jun 2022
7/9/2022	Jul 2022
7/22/2022	Jul 2022
7/23/2022	Jul 2022
7/27/2022	Jul 2022
8/2/2022	Aug 2022
8/8/2022	Aug 2022
8/19/2022	Aug 2022
9/26/2022	Sep 2022
10/14/2022	Okt 2022
10/29/2022	Okt 2022

`month_name`-måttet skapades i den föregående load-satsen genom att använda `monthname()`-funktionen och skicka `date`-fältet som funktionens argument.

Exempel på diagram för monthname-funktionen, diagramobjekt



`monthname()`-funktionen identifierar att transaktion 8192 ägde rum i mars 2022 och returnerar detta värde med hjälp av systemvariabeln `MonthNames`.

monthsend

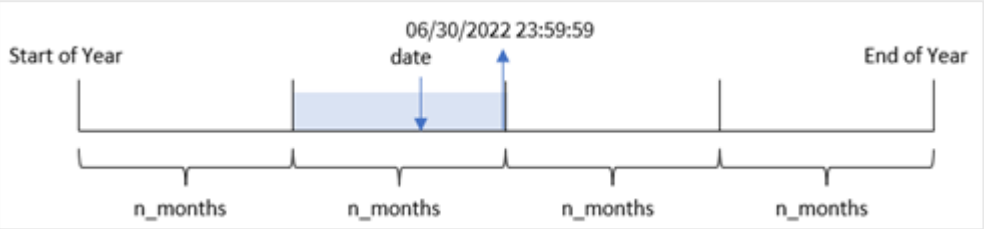
Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden i månaden, tvåmånadersperioden, kvartalet, fyramånadersperioden eller halvåret som innehåller ett basdatum. Det går även att hitta tidsmarkören för slutet av en föregående eller

senare tidsperiod. Det förvalda utdataformatet är det `DateFormat` som har definierats i skriptet.

```
MonthsEnd(n_months, date[, period_no [, first_month_of_year]])
```

Returnerad datatyp: dual

Diagram för `monthsend`-funktionen.



Argument

Argument	Beskrivning
n_months	Antalet månader som definierar perioden. Ett heltal eller uttryck vars resultat blir ett heltal som ska vara antingen: 1 (motsvaras av inmonth()-funktionen), 2 (tvåmånadersperiod), 3 (motsvaras av inquarter()-funktionen), 4 (fyramånadersperiod) eller 6 (halvår).
date	Det datum eller den tidsmarkör som ska utvärderas.
period_no	Periodens startpunkt kan flyttas med **period_no**, ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den period som innehåller **base_date**. Negativa värden i **period_no** anger föregående perioder och positiva värden anger efterföljande perioder.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

`monthsend()`-funktionen delar in året i segment baserat på det `n_months`-argument som tillhandahålls. Den utvärderar sedan vilket segment som varje tillhandahållet datum infaller och returnerar den sista millisekunden, i datumformat, för det segmentet. Funktionen kan returnera sluttidsmarkören från föregående eller följande segment och omdefiniera årets första månad.

Följande segment av året är tillgängliga i funktionen som `n_month`-argument.

`n_month` argument

Period	Antal månader
månad	1
två månader	2
kvartal	3
fyra månader	4
halvår	6

Användning

`monthsend()`-funktionen används som en del av ett uttryck när användaren vill att beräkningen ska använda den del av månaden som har förflutit hittills. Användaren har möjlighet att välja önskad period genom att använda en variabel. `monthsend()` kan exempelvis tillhandahålla en inmatningsvariabel som ger användaren möjlighet att beräkna den totala räntan som ännu inte uppkommit under månaden, kvartalet eller halvåret.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`monthsend(4, '07/19/2013')`	Returnerar 08/31/2013.
`monthsend(4, '10/19/2013', -1)`	Returnerar 08/31/2013.
`monthsend(4, '10/19/2013', 0, 2)`	Returnerar 01/31/2014. Eftersom årets start blir månad 2.

Exempel 1 – Grundläggande exempel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner för 2022 läses in i en tabell som heter '`Transactions`'.
- Ett datumfält som anges i `DateFormat`-systemvariabeln i formatet `(MM/DD/YYYY)`.
- En föregående LOAD-sats som innehåller följande:
    - `monthsend`-funktionen som är inställt som fältet '`bi_monthly_end`'. Detta grupperar

transaktioner i segment om två månader.

- `timestamp`-funktionen som returnerar starttidsmarkören för segmentet för varje transaktion.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    monthsend(2,date) as bi_monthly_end,
    timestamp(monthsend(2,date)) as bi_monthly_end_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

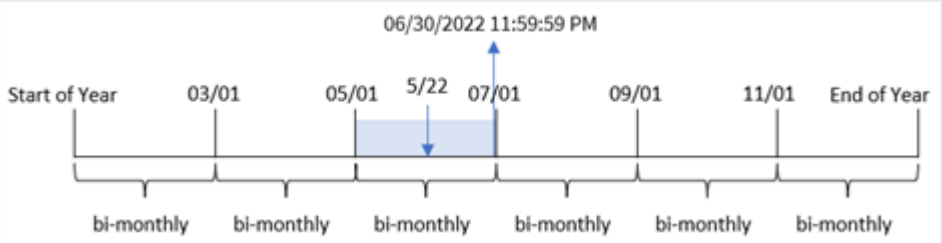
- `id`
- `date`
- `bi_monthly_end`
- `bi_monthly_end_timestamp`

Resultattabell

id	datum	bi_monthly_end	bi_monthly_end_timestamp
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM
8189	1/19/2022	02/28/2022	2/28/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	22/5/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	10/31/2022	10/31/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

'`bi_monthly_end`'-fältet skapas i den föregående load-satsen genom att använda `monthsend()`-funktionen. Det första argumentet som tillhandahålls är `2`, som delar upp året i tvåmånaderssegment. Det andra argumentet anger vilket fält som utvärderas.

Diagram för `monthsend`-funktionen med tvåmånaderssegment.



Transaktion 8195 äger rum den 22 maj. `monthsend()`-funktionen delar initialt upp året i tvåmånaderssegment. Transaktion 8195 infaller i segmentet mellan 1 maj och 30 juni. Resultatet blir att funktionen returnerar den sista millisekunden i det här segmentet, 2022-06-30, 23:59:59.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är uppgiften att skapa ett fält, `prev_bi_monthly_end`, som returnerar den första millisekunden i tvåmånaderssegmentet innan transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    monthsend(2,date,-1) as prev_bi_monthly_end,
    timestamp(monthsend(2,date,-1)) as prev_bi_monthly_end_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

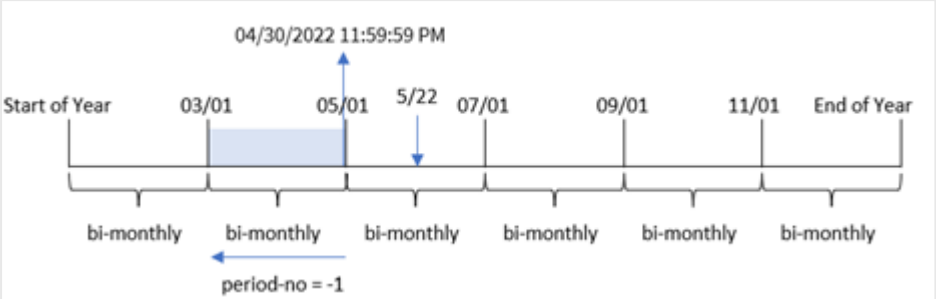
- `id`
- `date`
- `prev_bi_monthly_end`
- `prev_bi_monthly_end_timestamp`

Resultattabell

id	datum	prev_bi_monthly_end	prev_bi_monthly_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	12/31/2021	12/31/2021 11:59:59 PM
8191	2/28/2022	12/31/2021	12/31/2021 11:59:59 PM
8192	3/16/2022	02/28/2022	2/28/2022 11:59:59 PM
8193	4/1/2022	02/28/2022	2/28/2022 11:59:59 PM
8194	5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
8195	22/5/2022	04/30/2022	4/30/2022 11:59:59 PM
8196	6/15/2022	04/30/2022	4/30/2022 11:59:59 PM
8197	6/26/2022	04/30/2022	4/30/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	06/30/2022	6/30/2022 11:59:59 PM
8203	8/8/2022	06/30/2022	6/30/2022 11:59:59 PM
8204	8/19/2022	06/30/2022	6/30/2022 11:59:59 PM
8205	9/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8206	10/14/2022	08/31/2022	8/31/2022 11:59:59 PM
8207	10/29/2022	08/31/2022	8/31/2022 11:59:59 PM

Genom att använda `-1` som `period_no`-argument i `monthsend()`-funktionen, efter att initialt ha delat upp ett år i tvåmånaderssegment, returnerar funktionen den sista millisekunden i det föregående tvåmånaderssegmentet till när en transaktion äger rum.

Diagram för `monthsend`-funktion som returnerar det föregående tvåmånaderssegmentet.



Transaktion 8195 faller in i segmentet mellan maj och juni. Resultatet blir att det föregående tvåmånaderssegmentet var mellan 1 mars och 30 april och att funktionen returnerar den sista millisekunden i detta segment, 2022-04-30 23:59:59.

Exempel 3 – first_month_of_year

Laddningsskript och resultat

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är organisationspolicyn att april är den första månaden i räkenskapsåret.

Skapa ett fält, `bi_monthly_end`, som grupperar transaktioner i tvåmånaderssegment och returnerar tidsmarkören för den sista millisekunden i segmentet för varje transaktion.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    monthsend(2,date,0,4) as bi_monthly_end,
    timestamp(monthsend(2,date,0,4)) as bi_monthly_end_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
```

```
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `bi_monthly_end`
- `bi_monthly_end_timestamp`

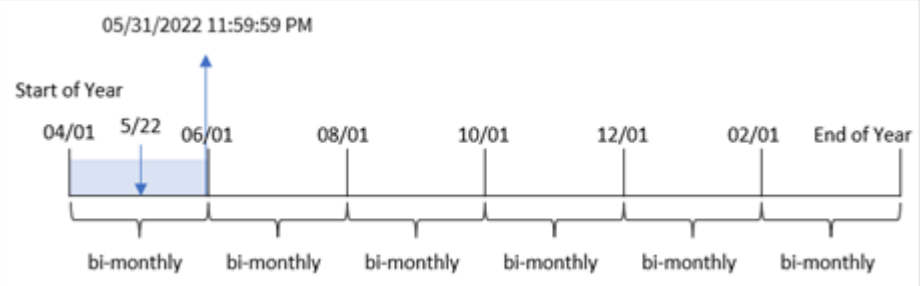
Resultattabell

id	datum	bi_monthly_end	bi_monthly_end_timestamp
8188	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8189	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	05/31/2022	5/31/2022 11:59:59 PM
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	22/5/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	07/31/2022	7/31/2022 11:59:59 PM
8197	6/26/2022	07/31/2022	7/31/2022 11:59:59 PM
8198	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8199	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8200	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8201	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8202	8/2/2022	09/30/2022	9/30/2022 11:59:59 PM
8203	8/8/2022	09/30/2022	9/30/2022 11:59:59 PM

id	datum	bi_monthly_end	bi_monthly_end_timestamp
8204	8/19/2022	09/30/2022	9/30/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	11/30/2022	11/30/2022 11:59:59 PM
8207	10/29/2022	11/30/2022	11/30/2022 11:59:59 PM

Genom att använda `4` som `first_month_of_year`-argumentet i `monthsend()`-funktionen, börjar funktionen året den 1 april. Den delar sedan upp året i tvåmånaderssegment: Apr-Maj, Jun-Jul, Aug-Sep, Okt-Nov, Dec-Jan, Feb-Mar.

Diagram för `monthsend`-funktionen med årets första månad inställd till april.



Transaktion 8195 ägde rum den 22 maj och faller in i segmentet mellan 1 april och 31 maj. Resultatet blir att funktionen returnerar den sista millisekunden i det här segmentet, 2022-05-31 23.59.59.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Samma datauppsättning och scenario som det första exemplet används. I det här exemplet är dock datauppsättningen oförändrad och har lästs in i applikationen.

I det här exemplet är uppgiften att skapa en beräkning som grupperar transaktioner i tvåmånaderssegment och returnerar tidsmarkören för den sista millisekunden i segmentet för varje transaktion som ett mått i ett diagramobjekt för en applikation.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
```

```
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

```
date
```

För att hämta tidsmarkören för den sista millisekunden i det tvåmånaderssegment då transaktionen tog plats skapar du följande mått:

- `=monthsEnd(2,date)`
- `=timestamp(monthsend(2,date))`

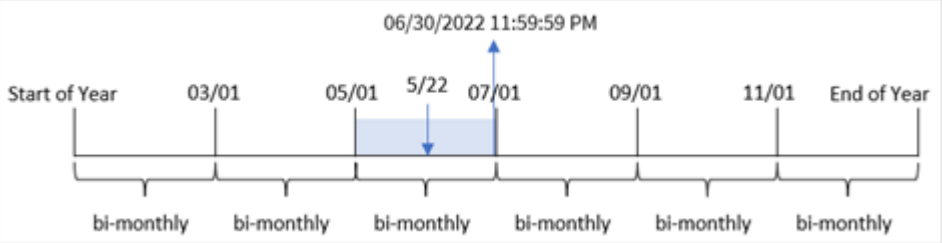
Resultattabell

id	datum	=monthsend(2,date)	=timestamp(monthsend(2,date))
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM
8189	1/19/2022	02/28/2022	2/28/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	22/5/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM

id	datum	=monthsend(2,date)	=timestamp(monthsend(2,date))
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	10/31/2022	10/31/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

`bi_monthly_end`-fältet skapas som ett mått i diagramobjektet med hjälp av `monthsend()`-funktionen. Det första argumentet som tillhandahålls är `2`, som delar upp året i tvåmånaderssegment. Det andra argumentet anger vilket fält som utvärderas.

Diagram för `monthsend`-funktionen med tvåmånaderssegment.



Transaktion 8195 äger rum den 22 maj. `monthsend()`-funktionen delar initialt upp året i tvåmånaderssegment. Transaktion 8195 infaller i segmentet mellan 1 maj och 30 juni. Resultatet blir att funktionen returnerar den första millisekunden i det här segmentet, 2022-06-30 23.59.59.

Exempel 5 – Scenario

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

I det här exemplet laddas en datauppsättning i en tabell som heter `Employee_Expenses`. Tabellen innehåller följande fält:

- Anställnings-ID
- Anställdas namn

- Genomsnittliga dagliga kostnadsanspråk för varje anställd.

Slutanvändaren vill ha ett diagram som visar, efter anställnings-id och anställds namn, de beräknade kostnadsanspråken för resten av en vald period. Räkenskapsåret börjar i januari.

Laddningsskript

```
SET vPeriod = 1;

Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Resultat

Ladda data och öppna ett nytt ark.

I början av laddningsskriptet har en variabel, `vPeriod`, skapats som är kopplad till variabelinmatningskontrollen.

Gör följande:

1. I resurspanelen klickar du på **Diagram**.
2. Under **Instrumentpanelsobjekt**, drar du en **Variabelinmatning** till arket.
3. Ange en titel för diagramobjektet.
4. Under **Variabel** väljer du **vPeriod** som namn och ställer in objektet så att det visas som en **rullgardinsmeny**.
5. Under **Värden** klickar du på **Dynamiska värden**. Ange följande:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`

Skapa en ny tabell och dessa fält som dimensioner:

- `employee_id`
- `employee_name`

För att beräkna ackumulerad ränta skapar du detta mått:

```
=floor(monthsend($(vPeriod),today(1))-today(1))*avg_daily_claim
```

Det här måttet är dynamiskt och kommer att ge olika tabellresultat beroende på vilket datum du laddar uppgifterna.

Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

employee_id	employee_ name	=floor(monthsend($(vPeriod),today(1))-today(1))*avg_daily_ claim
182	Mark	$1410.00
183	Deryck	$1175.00
184	Dexter	$1175.00
185	Sydney	$2538.00
186	Agatha	$1692.00

`monthsend()`-funktionen använder användarinmatning som sitt första argument och dagens datum som sitt andra argument. Detta returnerar slutdatumet för den tidsperiod som användaren har valt. Uttrycket returnerar sedan antalet dagar som återstår i den valda perioden genom att subtrahera dagens datum från detta slutdatum.

Detta värde multipliceras sedan med det genomsnittliga dagliga kostnadsanspråket från varje anställd för att beräkna det uppskattade värdet av anspråk som varje anställd förväntas göra under de återstående dagarna i den här perioden.

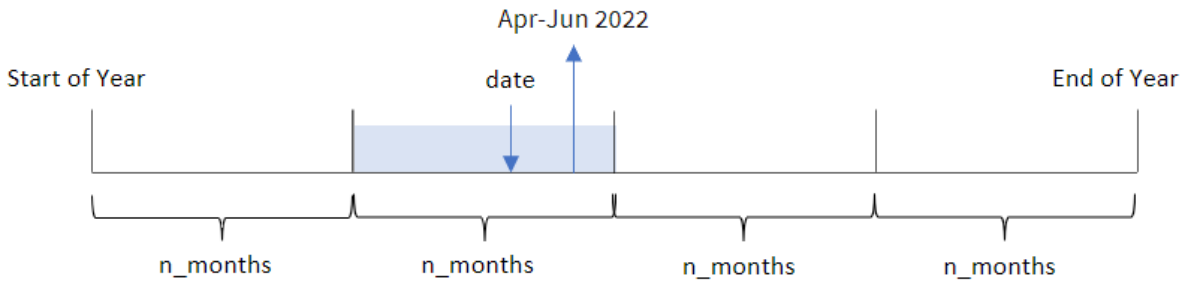
monthsname

Denna funktion returnerar ett visningsvärde som representerar månadsintervallet i perioden (formaterat enligt skriptvariabeln **MonthNames**) liksom året. Det underliggande numeriska värdet motsvarar en tidsmarkör för den första millisekunden i månaden, tvåmånadersperioden, kvartalet, fyramånadersperioden eller halvåret som innehåller ett basdatum.

```
MonthsName(n_months, date[, period_no[, first_month_of_year]])
```

Returnerad datatyp: Dual

Diagram för monthsname-funktionen



`monthsname()`-funktionen delar upp året i segment baserat på det angivna `n_months`-argumentet. Den utvärderar sedan segmentet som varje tillhandahållen `date` tillhör, och returnerar start- och slutmånadsnamnen för det segmentet, såväl som året. Funktionen ger också möjligheten att returnera dessa gränser från föregående eller efterföljande segment, samt omdefiniera vilket som är årets första månad.

Följande segment av året är tillgängliga i funktionen som `n_month`-argument:

Möjliga `n_month`-argument

Perioder	Antal månader
månad	1
två månader	2
kvartal	3
fyra månader	4
halvår	6

Argument

Argument	Beskrivning
n_months	Antalet månader som definierar perioden. Ett heltal eller uttryck vars resultat blir ett heltal som ska vara antingen: 1 (motsvaras av inmonth()-funktionen), 2 (tvåmånadersperiod), 3 (motsvaras av inquarter()-funktionen), 4 (fyramånadersperiod) eller 6 (halvår).
date	Det datum eller den tidsmarkör som ska utvärderas.
period_no	Periodens startpunkt kan flyttas med **period_no**, ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den period som innehåller **base_date**. Negativa värden i **period_no** anger föregående perioder och positiva värden anger efterföljande perioder.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Användning

`monthsname()`-funktionen är användbar när du vill ge användaren funktionalitet för att jämföra aggregationer efter en valfri period. Du kan till exempel ange en indatavariabel för att låta användaren se den totala försäljningen av produkter per månad, kvartal eller halvår.

Dessa dimensioner kan skapas antingen i laddningsskriptet genom att lägga till funktionen som ett fält i en Master Calendar-tabell, eller alternativt genom att skapa dimensionen direkt i ett diagram som en beräknad dimension.

Exempel på funktioner

Exempel	Resultat
`monthsname(4, '10/19/2013')`	Returnerar 'Sep-Dec 2013.' I detta och de andra exemplen är **SETMonthnames** uttrycket inställt på Jan;Feb;Mar, och så vidare.
`monthsname(4, '10/19/2013', -1)`	Returnerar 'May-Aug 2013'.
`monthsname(4, '10/19/2013', 0, 2)`	Returnerar Oct-Jan 2014, eftersom året är specificerat att börja i månaden 2. Därför slutar fyramånadersperioden med den första månaden av nästföljande år.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Grundläggande exempel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner för 2022, som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (MM/DD/YYYY) i `DateFormat` systemvariabeln.
- Skapandet av ett fält, `bi_monthly_range`, som grupperar transaktioner i tvåmånaderssegment och returnerar gränsnamnen för det segmentet för varje transaktion.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
```

```
        monthsname(2,date) as bi_monthly_range
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `bi_monthly_range`

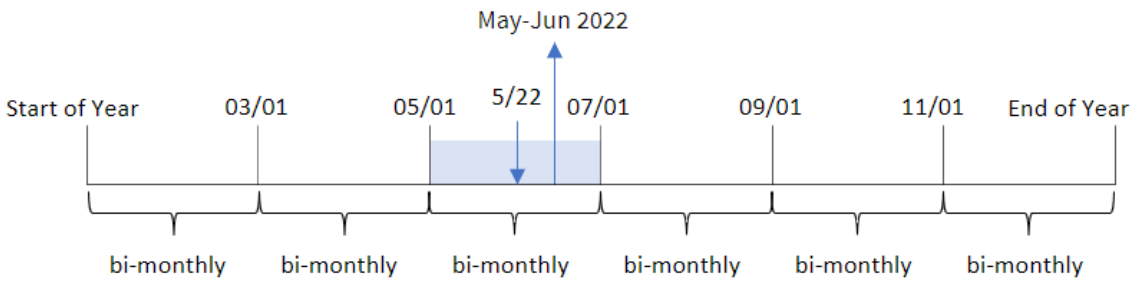
Resultattabell

datum	tvåmånadersintervall
2022-02-19	Jan-feb 2022
2022-03-07	Mar-apr 2022
2022-03-30	Mar-apr 2022
5/4/2022	Mar-apr 2022
16/4/2022	Mar-apr 2022
2022-05-01	Maj-jun 2022
5/7/2022	Maj-jun 2022
2022-05-22	Maj-jun 2022

datum	tvåmånadersintervall
6/15/2022	Maj-jun 2022
6/26/2022	Maj-jun 2022
7/9/2022	Jul-aug 2022
7/22/2022	Jul-aug 2022
7/23/2022	Jul-aug 2022
7/27/2022	Jul-aug 2022
8/2/2022	Jul-aug 2022
8/8/2022	Jul-aug 2022
8/19/2022	Jul-aug 2022
9/26/2022	Sep-okt 2022
10/14/2022	Sep-okt 2022
10/29/2022	Sep-okt 2022

`bi_monthly_range`-fältet skapas i föregående LOAD-sats med hjälp av funktionen `monthsname()`. Det första argumentet som tillhandahålls är 2, som delar upp året i tvåmånaderssegment. Det andra argumentet anger vilket fält som utvärderas.

Diagram för monthsname-funktionen, grundexempel



Transaktion 8195 äger rum den 22 maj. `monthsname()`-funktionen delar initialt upp året i tvåmånaderssegment. Transaktion 8195 infaller i segmentet mellan 1 maj och 30 juni. Därför returnerar funktionen dessa månader i systemvariabelformatet `MonthNames`, såväl som året, maj-juni 2022.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma inline-datauppsättning och scenario som det första exemplet används.
- Skapandet av ett fält, `prev_bi_monthly_range`, som grupperar transaktioner i tvåmånaderssegment och returnerar föregående segmentgränsnamn för varje transaktion.

Lägg till din andra text här, efter behov, med listor mm.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        MonthsName(2,date,-1) as prev_bi_monthly_range
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

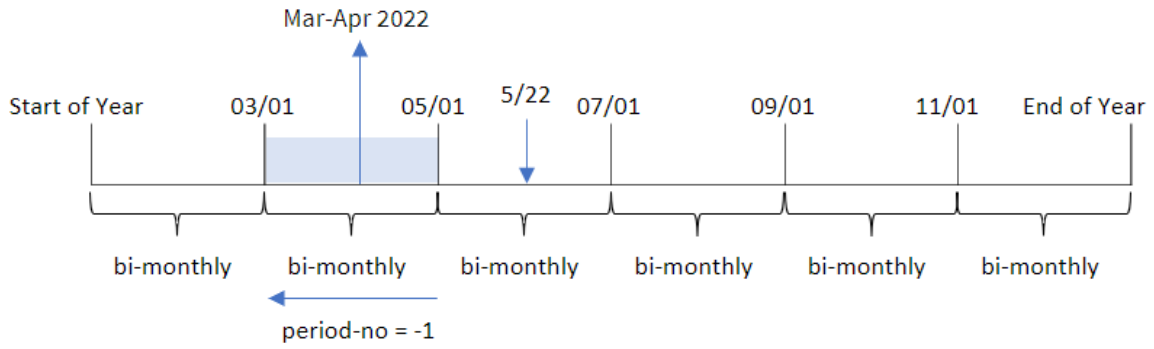
- `date`
- `prev_bi_monthly_range`

Resultattabell

datum	föregående_tvåmånadersintervall
2022-02-19	Nov-dec 2021
2022-03-07	Jan-feb 2022
2022-03-30	Jan-feb 2022
5/4/2022	Jan-feb 2022
16/4/2022	Jan-feb 2022
2022-05-01	Mar-apr 2022
5/7/2022	Mar-apr 2022
2022-05-22	Mar-apr 2022
6/15/2022	Mar-apr 2022
6/26/2022	Mar-apr 2022
7/9/2022	Maj-jun 2022
7/22/2022	Maj-jun 2022
7/23/2022	Maj-jun 2022
7/27/2022	Maj-jun 2022
8/2/2022	Maj-jun 2022
8/8/2022	Maj-jun 2022
8/19/2022	Maj-jun 2022
9/26/2022	Jul-aug 2022
10/14/2022	Jul-aug 2022
10/29/2022	Jul-aug 2022

I det här exemplet används -1 som `period_no`-argumentet i `monthsname()`-funktionen. Efter att initialt ha delat upp ett år i tvåmånaderssegment, returnerar funktionen sedan tidigare segmentgränser för när en transaktion äger rum.

Exempel på diagram för monthsname-funktionen, period_no



Transaktion 8195 faller in i segmentet mellan maj och juni. Detta innebär att det föregående tvåmånaderssegmentet var mellan 1 mars och 30 april, och därför returnerar funktionen mars-april 2022.

Exempel 3 – first_month_of_year

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma inline-datauppsättning och scenario som det första exemplet används.
- Skapandet av ett annat fält, `bi_monthly_range`, som grupperar transaktioner i tvåmånaderssegment och returnerar segmentgränsnamn för varje transaktion.

Men i det här exemplet måste vi också ställa in april som den första månaden i räkenskapsåret.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        MonthsName(2,date,0,4) as bi_monthly_range
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
```

```
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `bi_monthly_range`

Resultattabell

datum	tvåmånadersintervall
2022-02-19	Feb-mar 2021
2022-03-07	Feb-mar 2021
2022-03-30	Feb-mar 2021
5/4/2022	Apr-maj 2022
16/4/2022	Apr-maj 2022
2022-05-01	Apr-maj 2022
5/7/2022	Apr-maj 2022
2022-05-22	Apr-maj 2022
6/15/2022	Jun-jul 2022
6/26/2022	Jun-jul 2022
7/9/2022	Jun-jul 2022
7/22/2022	Jun-jul 2022
7/23/2022	Jun-jul 2022
7/27/2022	Jun-jul 2022

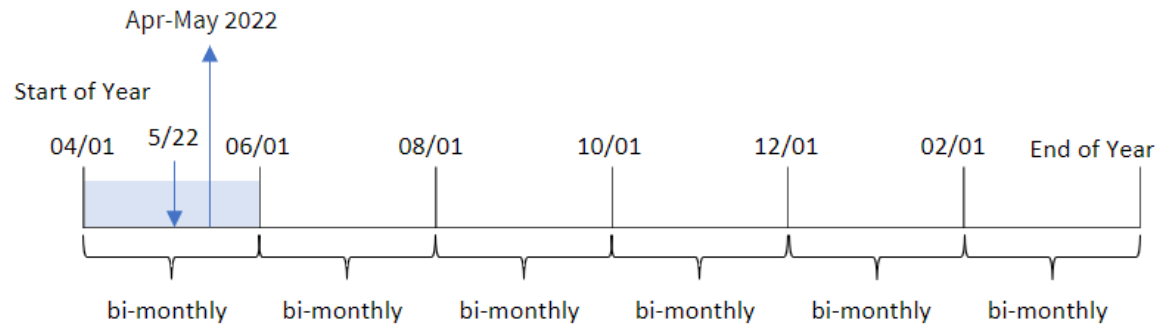
datum	tvåmånadersintervall
8/2/2022	Aug-sep 2022
8/8/2022	Aug-sep 2022
8/19/2022	Aug-sep 2022
9/26/2022	Aug-sep 2022
10/14/2022	Okt-nov 2022
10/29/2022	Okt-nov 2022

Genom att använda 4 som `first_month_of_year`-argument i `monthsname()`-funktionen, startar funktionen året den 1 april. Den delar sedan upp året i tvåmånaderssegment: Apr-Maj,Jun-Jul,Aug-Sep,Okt-Nov,Dec-Jan,Feb-Mar.

Stycketext för resultat.

Transaktion 8195 ägde rum den 22 maj och faller in i segmentet mellan 1 april och 31 maj. Därför returnerar funktionen april-maj 2022.

Exempel på diagram för monthsname-funktionen, first_month_of_year



Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna Dataladdningsredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet. I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som grupperar transaktioner i tvåmånaderssegment och returnerar segmentgränserna för varje transaktion skapas som ett mått i ett diagramobjekt för programmet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:`date`.

Skapa följande mått:

```
=monthsname(2,date)
```

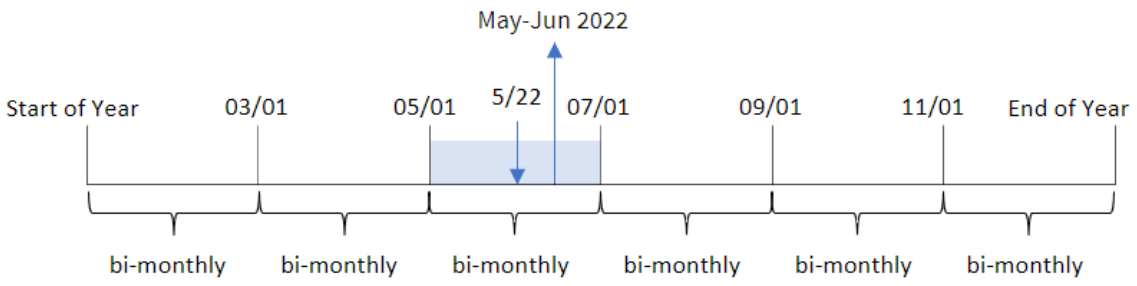
Resultattabell

datum	=månadsnamn(2,datum)
2022-02-19	Jan-feb 2022
2022-03-07	Mar-apr 2022
2022-03-30	Mar-apr 2022
5/4/2022	Mar-apr 2022
16/4/2022	Mar-apr 2022
2022-05-01	Maj-jun 2022

datum	=månadsnamn(2,datum)
5/7/2022	Maj-jun 2022
2022-05-22	Maj-jun 2022
6/15/2022	Maj-jun 2022
6/26/2022	Maj-jun 2022
7/9/2022	Jul-aug 2022
7/22/2022	Jul-aug 2022
7/23/2022	Jul-aug 2022
7/27/2022	Jul-aug 2022
8/2/2022	Jul-aug 2022
8/8/2022	Jul-aug 2022
8/19/2022	Jul-aug 2022
9/26/2022	Sep-okt 2022
10/14/2022	Sep-okt 2022
10/29/2022	Sep-okt 2022

`bi_monthly_range`-fältet skapas som ett mått i diagramobjektet med hjälp av `monthsname()`-funktionen. Det första argumentet som tillhandahålls är 2, som delar upp året i tvåmånaderssegment. Det andra argumentet anger vilket fält som utvärderas.

Exempel på diagram för monthsname-funktionen, diagramobjekt



Transaktion 8195 äger rum den 22 maj. `monthsname()`-funktionen delar initialt upp året i tvåmånaderssegment. Transaktion 8195 infaller i segmentet mellan 1 maj och 30 juni. Därför returnerar funktionen dessa månader i systemvariabelformatet `MonthNames`, såväl som året, maj-juni 2022.

Exempel 5 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner efter tidsmarkör som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (MM/DD/YYYY) i `DateFormat`-systemvariabeln.

Slutanvändaren vill ha ett diagramobjekt som visar den totala försäljningen efter en valfri period. Detta kan uppnås även när denna dimension inte är tillgänglig i datamodellen, med hjälp av `monthsname()`-funktionen som en beräknad dimension som modifieras dynamiskt med hjälp av variabelinmatningskontroll.

Laddningsskript

```
SET vPeriod = 1;
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/7/2022',17.17
8189,'1/19/2022',37.23
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark.

I början av laddningsskriptet har en variabel (`vPeriod`) skapats som kommer att kopplas till variabelinmatningskontrollen. Konfigurera sedan variabeln som ett anpassat objekt i arket.

Gör följande:

1. I resurspanelen klickar du på **Diagram**.
2. Under **Instrumentpanelsobjekt**, drar du en **Variabelinmatning** till arket.
3. Ange en titel för diagramobjektet.
4. Under **Variabel** väljer du **vPeriod** som namn och ställer in objektet så att det visas som en **rullgardinsmeny**.
5. Under **Värden** konfigurerar du objektet för att använda dynamiska värden. Ange följande:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

Skapa sedan resultattabellen.

Gör följande:

1. Skapa en ny tabell och lägg till följande beräknade dimension:
   `=monthsname($(vPeriod),date)`
2. Lägg till detta mått för att beräkna den totala försäljningen:
   `=sum(amount)`
3. Ange måttets **Nummerformatering** till **Pengar**. Klicka på ✔ **Redigering klar**. Du kan nu ändra data som visas i tabellen genom att justera tidssegmentet i variabelobjektet.

Så här kommer resultattabellen att se ut när `tertial`-alternativet är valt:

Resultattabell

monthsname($(vPeriod),datum)	=sum(amount)
Jan-apr 2022	253,89
Maj-aug 2022	713.58
Sep-dec 2022	248.12

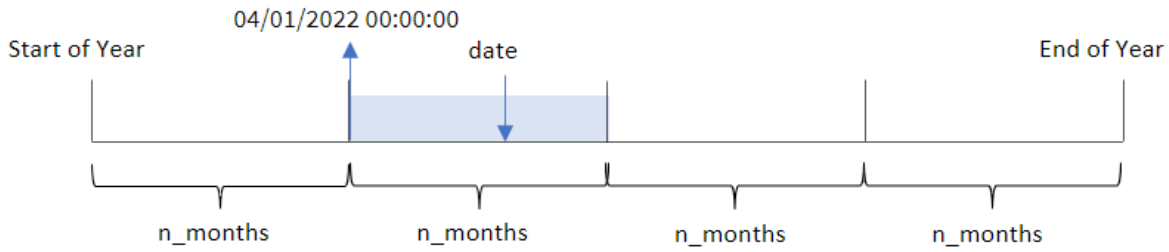
monthsstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden i månaden, tvåmånadersperioden, kvartalet, fyramånadersperioden eller halvåret som innehåller ett basdatum. Det går även att hitta tidsmarkören för en tidigare eller senare tidsperiod. Standardformatet för utdata är det **DateFormat** som har angetts i skriptet.

```
MonthsStart(n_months, date[, period_no [, first_month_of_year]])
```

Returnerad datatyp: Dual

Diagram för `monthsstart()`-funktionen.



`monthsstart()`-funktionen delar in året i segment baserat på det `n_months`-argument som tillhandahålls. Den utvärderar sedan vilket segment som varje tillhandahållet datum infaller och returnerar den första millisekunden, i datumformat, för det segmentet. Funktionen ger också möjlighet att returnera starttidsmarkören från föregående eller efterföljande segment, samt omdefiniera vilket som är årets första månad.

Följande segment av året är tillgängliga i funktionen som `n_month`-argument:

Möjliga `n_month`-argument

Perioder	Antal månader
månad	1
två månader	2
kvartal	3
fyra månader	4
halvår	6

Argument

Argument	Beskrivning
n_months	Antalet månader som definierar perioden. Ett heltal eller uttryck vars resultat blir ett heltal som ska vara antingen: 1 (motsvaras av inmonth()-funktionen), 2 (tvåmånadersperiod), 3 (motsvaras av inquarter()-funktionen), 4 (fyramånadersperiod) eller 6 (halvår).
date	Det datum eller den tidsmarkör som ska utvärderas.
period_no	Periodens startpunkt kan flyttas med **period_no**, ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den period som innehåller **base_date**. Negativa värden i **period_no** anger föregående perioder och positiva värden anger efterföljande perioder.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Användning

`monthsstart()`-funktionen används vanligtvis som en del av ett uttryck när användaren vill att beräkningen ska använda den del av en period som ännu inte har inträffat. Detta kan exempelvis användas för att tillhandahålla en inmatningsvariabel som ger användaren möjlighet att beräkna den totala räntan som har ackumulerats hittills under månaden, kvartalet eller halvåret.

Exempel på funktioner

Exempel	Resultat
`monthsstart(4, '10/19/2013')`	Returnerar 09/01/2013.
`monthsstart(4, '10/19/2013, -1)`	Returnerar 05/01/2013.
`monthsstart(4, '10/19/2013', 0, 2 )`	Returnerar 10/01/2013, eftersom årets början blir månad 2.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner för 2022, som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (MM/DD/YYYY) i `DateFormat` systemvariabeln.
- Ett fält `bi_monthly_start` skapas som grupperar transaktioner i tvåmånaderssegment och returnerar starttidsmarkören för det segmentet för varje transaktion.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
```

```
Transactions:
    Load
        *,
        monthsstart(2,date) as bi_monthly_start,
        timestamp(monthsstart(2,date)) as bi_monthly_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `bi_monthly_start`
- `bi_monthly_start_timestamp`

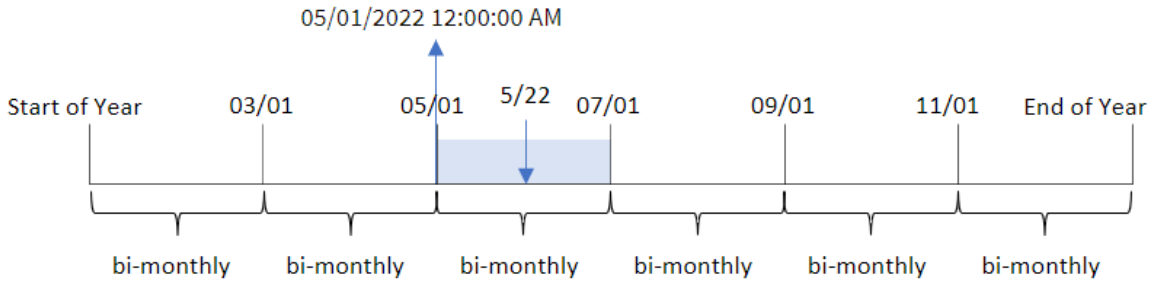
Resultattabell

datum	tvåmånaders_start	bi_monthly_start_tidsmarkör
2022-02-19	01/01/2022	2022-01-01 00:00:00
2022-03-07	2022-03-01	3/1/2022 12:00:00 AM
2022-03-30	2022-03-01	3/1/2022 12:00:00 AM
5/4/2022	2022-03-01	3/1/2022 12:00:00 AM
16/4/2022	2022-03-01	3/1/2022 12:00:00 AM

datum	tvåmånaders_start	bi_monthly_start_tidsmarkör
2022-05-01	2022-05-01	5/1/2022 12:00:00 AM
5/7/2022	2022-05-01	5/1/2022 12:00:00 AM
2022-05-22	2022-05-01	5/1/2022 12:00:00 AM
6/15/2022	2022-05-01	5/1/2022 12:00:00 AM
6/26/2022	2022-05-01	5/1/2022 12:00:00 AM
7/9/2022	2022-07-01	2022-07-01 00:00:00
7/22/2022	2022-07-01	2022-07-01 00:00:00
7/23/2022	2022-07-01	2022-07-01 00:00:00
7/27/2022	2022-07-01	2022-07-01 00:00:00
8/2/2022	2022-07-01	2022-07-01 00:00:00
8/8/2022	2022-07-01	2022-07-01 00:00:00
8/19/2022	2022-07-01	2022-07-01 00:00:00
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM

`bi_monthly_start`-fältet skapas i föregående LOAD-sats med hjälp av funktionen `monthsstart()`. Det första argumentet som tillhandahålls är 2, som delar upp året i tvåmånaderssegment. Det andra argumentet anger vilket fält som utvärderas.

Diagram för `monthsstart()`-funktionen, exempel utan ytterligare argument



Transaktion 8195 äger rum den 22 maj. `monthsstart()`-funktionen delar initialt upp året i tvåmånaderssegment. Transaktion 8195 infaller i segmentet mellan 1 maj och 30 juni. Därför returnerar funktionen den första millisekunden i det här segmentet, 1 maj 2022 0:00:00.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Ett fält `prev_bi_monthly_start` skapas som returnerar den första millisekunden i tvåmånaderssegmentet innan transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthsstart(2,date,-1) as prev_bi_monthly_start,
           timestamp(monthsstart(2,date,-1)) as prev_bi_monthly_start_timestamp
           ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

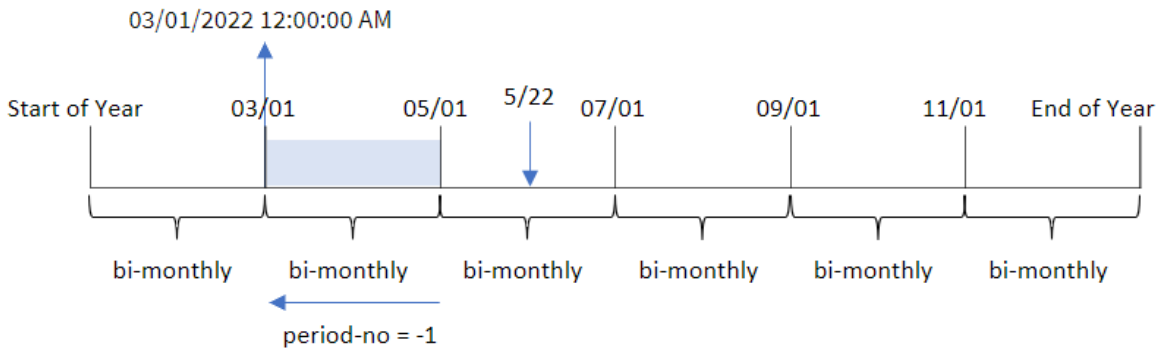
- `date`
- `prev_bi_monthly_start`
- `prev_bi_monthly_start_timestamp`

Resultattabell

datum	föregående_tvåmånaders_start	föregående_tvåmånaders_start_tidsmarkör
2022-02-19	11/01/2021	2021-11-01 00:00:00
2022-03-07	01/01/2022	2022-01-01 00:00:00
2022-03-30	01/01/2022	2022-01-01 00:00:00
5/4/2022	01/01/2022	2022-01-01 00:00:00
16/4/2022	01/01/2022	2022-01-01 00:00:00
2022-05-01	2022-03-01	3/1/2022 12:00:00 AM
5/7/2022	2022-03-01	3/1/2022 12:00:00 AM
2022-05-22	2022-03-01	3/1/2022 12:00:00 AM
6/15/2022	2022-03-01	3/1/2022 12:00:00 AM
6/26/2022	2022-03-01	3/1/2022 12:00:00 AM
7/9/2022	2022-05-01	5/1/2022 12:00:00 AM
7/22/2022	2022-05-01	5/1/2022 12:00:00 AM
7/23/2022	2022-05-01	5/1/2022 12:00:00 AM
7/27/2022	2022-05-01	5/1/2022 12:00:00 AM
8/2/2022	2022-05-01	5/1/2022 12:00:00 AM
8/8/2022	2022-05-01	5/1/2022 12:00:00 AM
8/19/2022	2022-05-01	5/1/2022 12:00:00 AM
9/26/2022	2022-07-01	2022-07-01 00:00:00
10/14/2022	2022-07-01	2022-07-01 00:00:00
10/29/2022	2022-07-01	2022-07-01 00:00:00

Genom att använda -1 som `period_no`-argument i `monthsstart()`-funktionen, efter att initialt ha delat upp ett år i tvåmånaderssegment, returnerar funktionen den första millisekunden i det föregående tvåmånaderssegmentet till när en transaktion äger rum.

Diagram med exempel på `monthsstart()`-funktionen, period_no



Transaktion 8195 faller in i segmentet mellan maj och juni. Därför var det föregående tvåmånaderssegmentet mellan 1 mars och 30 april, så funktionen returnerar funktionen den första millisekunden i det här segmentet, 1 mars 2022 0:00:00.

Exempel 3 – first_month_of_year

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Ett fält skapas `bi_monthly_start`, som grupperar transaktioner i tvåmånaderssegment och returnerar starttidsmarkören för uppsättningen för varje transaktion.

Men i det här exemplet måste vi också ställa in april som den första månaden i räkenskapsåret.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthsstart(2,date,0,4) as bi_monthly_start,
        timestamp(monthsstart(2,date,0,4)) as bi_monthly_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
```

```
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `bi_monthly_start`
- `bi_monthly_start_timestamp`

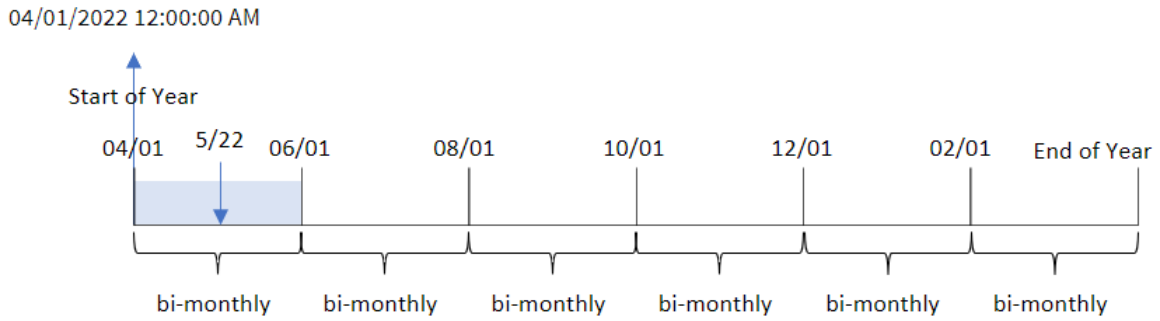
Resultattabell

datum	tvåmånaders_start	bi_monthly_start_tidsmarkör
2022-02-19	2022-02-01	2/1/2022 12:00:00 AM
2022-03-07	2022-02-01	2/1/2022 12:00:00 AM
2022-03-30	2022-02-01	2/1/2022 12:00:00 AM
5/4/2022	04/01/2022	2022-04-01 00:00:00
16/4/2022	04/01/2022	2022-04-01 00:00:00
2022-05-01	04/01/2022	2022-04-01 00:00:00
5/7/2022	04/01/2022	2022-04-01 00:00:00
2022-05-22	04/01/2022	2022-04-01 00:00:00
6/15/2022	2022-06-01	2022-06-01 00:00:00
6/26/2022	2022-06-01	2022-06-01 00:00:00
7/9/2022	2022-06-01	2022-06-01 00:00:00
7/22/2022	2022-06-01	2022-06-01 00:00:00

datum	tvåmånaders_start	bi_monthly_start_tidsmarkör
7/23/2022	2022-06-01	2022-06-01 00:00:00
7/27/2022	2022-06-01	2022-06-01 00:00:00
8/2/2022	2022-08-01	2022-08-01 00:00:00
8/8/2022	2022-08-01	2022-08-01 00:00:00
8/19/2022	2022-08-01	2022-08-01 00:00:00
9/26/2022	2022-08-01	2022-08-01 00:00:00
10/14/2022	10/01/2022	2022-10-01 00:00:00
10/29/2022	10/01/2022	2022-10-01 00:00:00

Genom att använda 4 som `first_month_of_year`-argument i `monthsstart()`-funktionen, startar funktionen året den 1 april. Den delar sedan upp året i tvåmånaderssegment: Apr-May,Jun-Jul,Aug-Sep,Oct-Nov,Dec-Jan,Feb-Mar.

Diagram med exempel på `monthsstart()`-funktionen, first_month_of_year



Transaktion 8195 ägde rum den 22 maj och faller in i segmentet mellan 1 april och 31 maj. Därför returnerar funktionen den första millisekunden i det här segmentet, 1 april 2022 0:00:00.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna Dataladdningsredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som grupperar transaktioner i tvåmånaderssegment och returnerar starttidsmarkören för uppsättningen för varje transaktion skapas som ett mått i ett diagramobjekt för programmet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date` .

Skapa följande mått:

```
=monthsstart(2,date)
```

```
=timestamp(monthsstart(2,date))
```

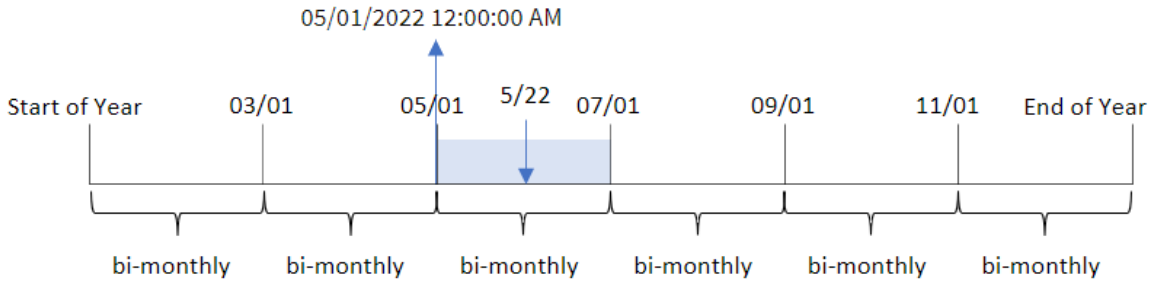
De här beräkningarna hämtar starttidsmarkören för det tvåmånaderssegment då varje transaktion ägde rum.

Resultattabell

datum	=månadsstart(2,datum)	=tidsmarkör(månadsstart(2,datum))
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM

datum	=månadsstart(2,datum)	=tidsmarkör(månadsstart(2,datum))
7/9/2022	2022-07-01	2022-07-01 00:00:00
7/22/2022	2022-07-01	2022-07-01 00:00:00
7/23/2022	2022-07-01	2022-07-01 00:00:00
7/27/2022	2022-07-01	2022-07-01 00:00:00
8/2/2022	2022-07-01	2022-07-01 00:00:00
8/8/2022	2022-07-01	2022-07-01 00:00:00
8/19/2022	2022-07-01	2022-07-01 00:00:00
2022-05-01	2022-05-01	5/1/2022 12:00:00 AM
5/7/2022	2022-05-01	5/1/2022 12:00:00 AM
2022-05-22	2022-05-01	5/1/2022 12:00:00 AM
6/15/2022	2022-05-01	5/1/2022 12:00:00 AM
6/26/2022	2022-05-01	5/1/2022 12:00:00 AM
2022-03-07	2022-03-01	3/1/2022 12:00:00 AM
2022-03-30	2022-03-01	3/1/2022 12:00:00 AM
5/4/2022	2022-03-01	3/1/2022 12:00:00 AM
16/4/2022	2022-03-01	3/1/2022 12:00:00 AM
2022-02-19	01/01/2022	2021-01-01 00:00:00

Diagram med exempel på `monthsstart()`-funktionen, diagramobjekt



Transaktion 8195 ägde rum den 22 maj. `monthsstart()`-funktionen delar initialt upp året i tvåmånaderssegment. Transaktion 8195 infaller i segmentet mellan 1 maj och 30 juni. Därför returnerar funktionen den första millisekunden i det här segmentet, 2022-05-01 0:00:00.

Exempel 5 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning lånesaldon som laddas i en tabell som heter `Loans`.
- Data som består av låne-ID:n, saldot vid månadens början och den enkla räntan som tas ut på varje lån per år.

Slutanvändaren vill ha ett diagramobjekt som visar, efter låne-id, den aktuella räntan som har ackumulerats för varje lån under den valda perioden. Räkenskapsåret börjar i januari.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Resultat

Ladda data och öppna ett ark.

I början av laddningsskriptet har en variabel (`vPeriod`) skapats som kommer att kopplas till variabelinmatningskontrollen. Konfigurera sedan variabeln som ett anpassat objekt i arket.

Gör följande:

1. I resurspanelen klickar du på **Diagram**.
2. Under **Instrumentpanelsobjekt**, drar du en **Variabelinmatning** till arket.
3. Ange en titel för diagramobjektet.
4. Under **Variabel** väljer du **vPeriod** som namn och ställer in objektet så att det visas som en **rullgardinsmeny**.

5. Under **Värden** konfigurerar du objektet för att använda dynamiska värden. Ange följande:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

Skapa sedan resultattabellen.

Gör följande:

1. Skapa en ny tabell. Lägg till följande fält som dimensioner:
   - `employee_id`
   - `employee_name`
2. Skapa ett mått för att beräkna ackumulerad ränta:
   `=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)`
3. Ange måttets **Nummerformatering** till **Pengar**. Klicka på ✔ **Redigering klar**. Du kan nu ändra data som visas i tabellen genom att justera tidssegmentet i variabelobjektet.

Så här kommer resultattabellen att se ut när `month`-periodalternativet är valt:

Resultattabell

loan_id	start_balance	=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)
8188	USD 10000.00	$7.95
8189	$15000.00	$67.93
8190	$17500.00	$33.37
8191	$21000.00	$56.73
8192	$90000.00	$600.66

`monthsstart()`-funktionen använder användarinmatning som sitt första argument och dagens datum som sitt andra argument, returnerar startdatumet i perioden som användaren har valt. Genom att subtrahera resultatet från det aktuella datumet returnerar uttrycket antalet dagar som har förflutit hittills denna period.

Detta värde multipliceras sedan med räntan och divideras med 365 för att returnera den effektiva räntan för perioden. Resultatet multipliceras sedan med lånets utgångssaldo för att returnera den upplupna räntan hittills denna period.

monthstart

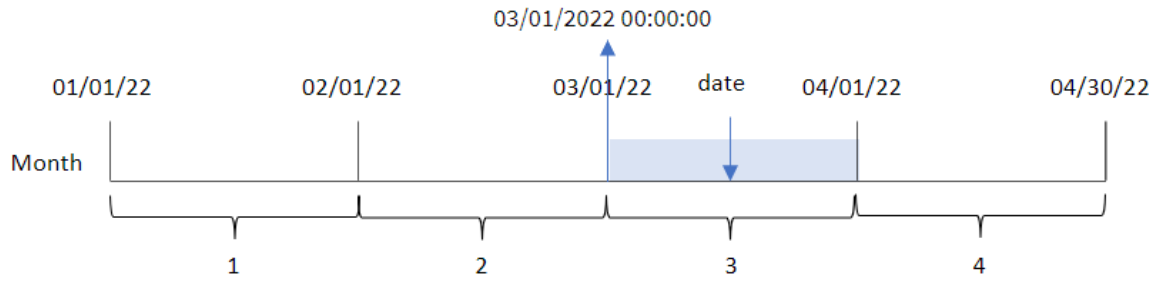
Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden av den första dagen i den månad som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

Syntax:

```
MonthStart(date[, period_no])
```

Returnerad datatyp: dual

Diagram för funktionen `monthstart()`



`monthstart()`-funktionen bestämmer vilken månad datumet infaller. Den returnerar sedan en tidsmarkör, i datumformat, för den första millisekunden av den månaden.

Argument

Argument	Beskrivning
date	Det datum eller den tidsmarkör som ska utvärderas.
period_no	**period_no** är ett heltal som om det är 0 eller utelämnas anger månaden som innehåller **date**. Negativa värden i **period_no** anger föregående månader och positiva värden anger efterföljande månader.

Användning

`monthstart()`-funktionen används vanligtvis som en del av ett uttryck när användaren vill att beräkningen ska använda den del av månaden som har förflutit hittills. Den kan exempelvis användas för att beräkna den ränta som har ackumulerats under en månad fram till ett visst datum.

Exempel på funktioner

Exempel	Resultat
`monthstart('10/19/2001')`	Returnerar 10/01/2001.
`monthstart('10/19/2001', -1)`	Returnerar 09/01/2001.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2022 som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i `DateFormat`-systemvariabeln.
- Ett fält `start_of_month` skapas som returnerar en tidsmarkör för början av den månad då transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthstart(date) as start_of_month,
        timestamp(monthstart(date)) as start_of_month_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
```

```
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `start_of_month`
- `start_of_month_timestamp`

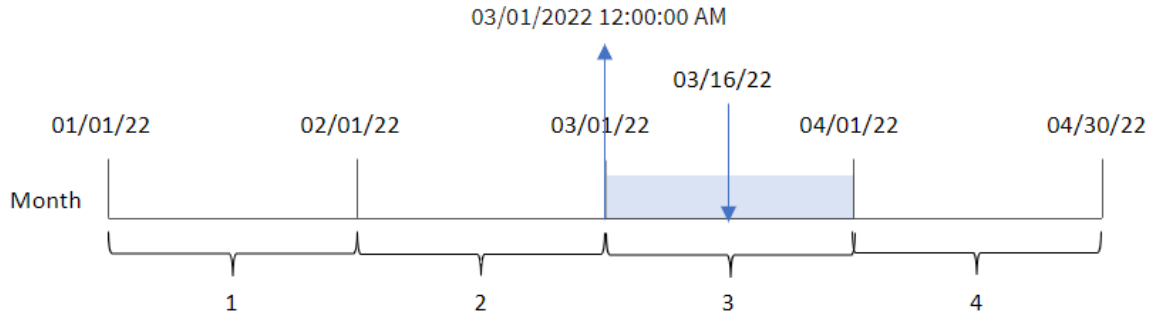
Resultattabell

datum	start_of_month	start_of_month_timestamp
1/7/2022	01/01/2022	2022-01-01 00:00:00
1/19/2022	01/01/2022	2022-01-01 00:00:00
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/01/2022	2/1/2022 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	04/01/2022	3/1/2022 12:00:00 AM
5/7/2022	2022-05-01	5/1/2022 12:00:00 AM
5/16/2022	05/01/2022	2022-05-01 00:00:00
6/15/2022	06/01/2022	2022-06-01 00:00:00
6/26/2022	07/01/2022	6/1/2022 12:00:00 AM
7/9/2022	07/01/2022	2022-07-01 00:00:00
7/22/2022	2022-07-01	2022-07-01 00:00:00
7/23/2022	2022-07-01	2022-07-01 00:00:00
7/27/2022	2022-07-01	2022-07-01 00:00:00
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	2022-08-01 00:00:00
8/19/2022	2022-08-01	2022-08-01 00:00:00
9/26/2022	01/09/2022	9/1/2022 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	2022-10-01 00:00:00

`start_of_month`-fältet skapades i den föregående load-satsen genom att använda `monthstart()`-funktionen och skicka datumfältet som funktionens argument.

`monthstart()`-funktionen identifierar vilken månad datumvärdet infaller och returnerar en tidsmarkör för den första millisekunden av den månaden.

Diagram för `monthstart()`-funktionen, exempel utan ytterligare argument



Transaktion 8192 ägde rum den 16 mars. `monthstart()`-funktionen returnerar den första millisekunden av den månaden, vilket är den 1 mars 0:00:00.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Ett fält `previous_month_start` skapas som returnerar tidsmarkören för början av månaden innan transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthstart(date,-1) as previous_month_start,
        timestamp(monthstart(date,-1)) as previous_month_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
```

```
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `previous_month_start`
- `previous_month_start_timestamp`

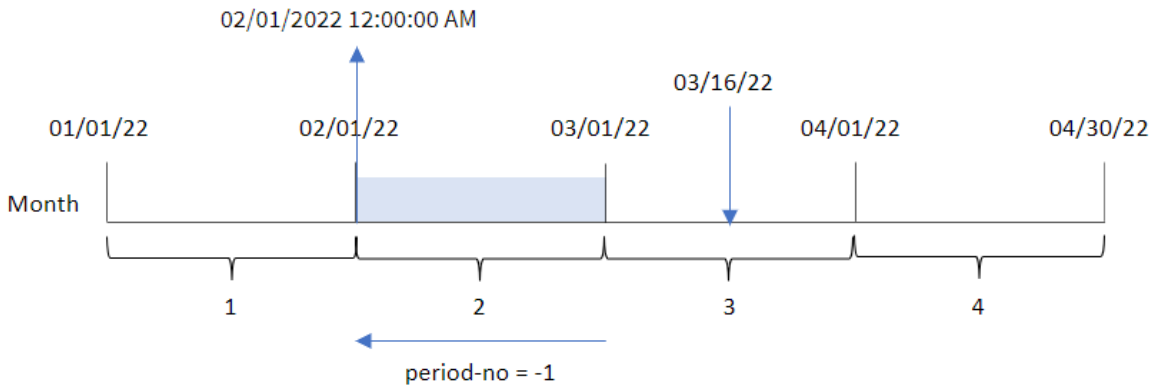
Resultattabell

datum	previous_month_start	previous_month_start_timestamp
1/7/2022	12/01/2021	12/1/2021 12:00:00 AM
1/19/2022	2021-12-01	2021-12-01 00:00:00
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	2022-01-01 00:00:00
3/16/2022	02/01/2022	2/1/2022 12:00:00 AM
4/1/2022	03/01/2022	2022-03-01 00:00:00
5/7/2022	04/01/2022	2022-04-01 00:00:00
5/16/2022	04/01/2022	2022-04-01 00:00:00
6/15/2022	2022-01-05	2022-01-05 00:00:00
6/26/2022	2022-01-05	2022-05-01 00:00:00
7/9/2022	06/01/2022	2022-06-01 00:00:00

datum	previous_month_start	previous_month_start_timestamp
7/22/2022	2022-06-01	2022-06-01 00:00:00
7/23/2022	2022-06-01	2022-06-01 00:00:00
7/27/2022	2022-06-01	2022-06-01 00:00:00
8/2/2022	2022-07-01	7/1/2022 12:00:00 AM
8/8/2022	2022-07-01	2022-07-01 00:00:00
8/19/2022	2022-07-01	2022-07-01 00:00:00
9/26/2022	2022-08-01	2022-08-01 00:00:00
10/14/2022	09/01/2022	2022-09-01 00:00:00
10/29/2022	2022-09-01	9/1/2022 12:00:00 AM

I det här fallet, eftersom ett `period_no` på -1 användes som förskjutningsargument i `monthstart()`-funktionen, identifierar funktionen först den månad då transaktionerna äger rum. Den skiftar sedan till närmast föregående månad och identifierar den första millisekunden i den månaden.

Diagram med exempel på `monthstart()`-funktionen, period_no



Transaktion 8192 ägde rum den 16 mars. `monthstart()`-funktionen identifierar att månaden innan transaktionen ägde rum var februari. Den returnerar sedan den första millisekunden den månaden, 1 februari 0:00:00.

Exempel 3 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som returnerar en tidsmarkör för början av månaden då transaktionerna ägde rum skapas som ett mått i ett diagramobjekt för programmet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

För att beräkna startdatum för den månad då en transaktion äger rum skapar du följande mått:

- `=monthstart(date)`
- `=timestamp(monthstart(date))`

Resultattabell

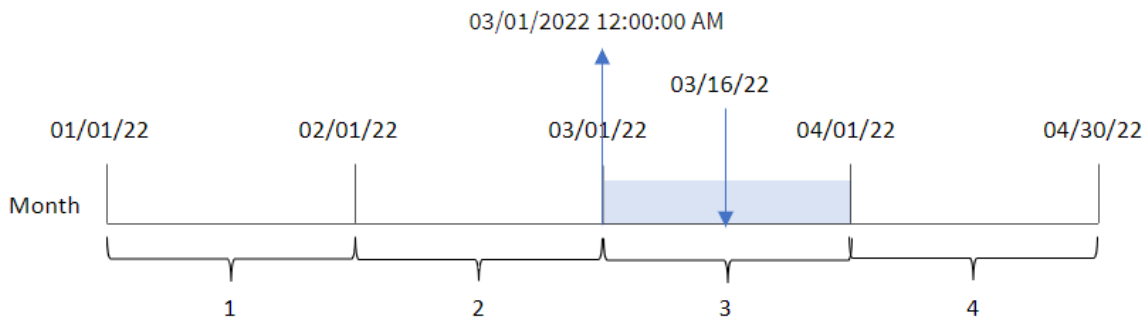
datum	=monthstart(date)	=timestamp(monthstart(date))
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

datum	=monthstart(date)	=timestamp(monthstart(date))
9/26/2022	01/09/2022	9/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	2022-08-01 00:00:00
8/19/2022	2022-08-01	8/1/2022 12:00:00 AM
7/9/2022	07/01/2022	2022-07-01 00:00:00
7/22/2022	2022-07-01	2022-07-01 00:00:00
7/23/2022	2022-07-01	2022-07-01 00:00:00
7/27/2022	2022-07-01	7/1/2022 12:00:00 AM
6/15/2022	06/01/2022	2022-06-01 00:00:00
6/26/2022	2022-06-01	6/1/2022 12:00:00 AM
5/7/2022	2022-05-01	5/1/2022 12:00:00 AM
5/16/2022	05/01/2022	5/1/2022 12:00:00 AM
4/1/2022	04/01/2022	3/1/2022 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/01/2022	2/1/2022 12:00:00 AM
1/7/2022	01/01/2022	2022-01-01 00:00:00
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM

`start_of_month`-måttet skapades i ett diagramobjektet genom att använda `monthstart()`-funktionen och skicka datumfältet som funktionens argument.

`monthstart()`-funktionen identifierar vilken månad datumvärdet infaller och returnerar en tidsmarkör för den första millisekunden av den månaden.

Diagram med exempel på `monthstart()`-funktionen, diagramobjekt



Transaktion 8192 ägde rum den 16 mars. `monthstart()`-funktionen identifierar att transaktionen ägde rum i mars och returnerar den första millisekunden av den månaden, vilket är den 1 mars 0:00:00.

Exempel 4 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning lånesaldon som laddas i en tabell som heter `Loans`.
- Data som består av låne-ID:n, saldot vid månadens början och den enkla räntan som tas ut på varje lån per år.

Slutanvändaren vill ha ett diagramobjekt som visar, efter låne-id, den aktuella räntan som har ackumulerats för varje lån under månaden hittills.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:
   - `loan_id`
   - `start_balance`
2. Skapa sedan ett mått för att beräkna den ackumulerade räntan:
   `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
3. Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

loan_id	start_balance	=start_balance*(rate*(today(1)-monthstart(today(1)))/365)
8188	$10000.00	$16.44
8189	$15000.00	$58.56
8190	$17500.00	$28.77
8191	$21000.00	$48.90
8192	$90000.00	$517.81

Genom att använda dagens datum som enda argument returnerar `monthstart()`-funktionen startdatumet för den aktuella månaden. Genom att subtrahera resultatet från det aktuella datumet returnerar uttrycket antalet dagar som har förflutit hittills denna månad.

Detta värde multipliceras sedan med räntan och divideras med 365 för att returnera den effektiva räntan för perioden. Resultatet multipliceras sedan med lånets utgångssaldo för att returnera den upplupna räntan hittills denna månad.

networkdays

Funktionen **networkdays** returnerar antalet arbetsdagar (måndag-fredag) mellan och inklusive **start_date** och **end_date** med hänsyn till eventuella **holiday**.

Syntax:

```
networkdays (start_date, end_date [, holiday])
```

Returnerad datatyp: heltal

Kalenderdiagram som visar det datumintervall som har returnerats av networkdays-funktionen.

`networkdays`-funktionen har följande begränsningar:

- Det finns inget sätt att ändra arbetsdagar. Med andra ord finns det inget sätt att ändra funktionen för regioner eller situationer som omfattar någonting annat än arbete mellan måndag och fredag.
- `holiday`-parametern måste vara en strängkonstant. Uttryck accepteras inte.

Argument

Argument	Beskrivning
start_date	Startdatum som ska utvärderas.
end_date	Slutdatumn som ska utvärderas.
holiday	Ledighetsperioder som ska undantas från arbetsdagar. En helgdag anges som ett strängkonstant datum. Du kan ange flera semesterdatum, avgränsade med kommatecken. **Exempel:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Användning

`networkdays()`-funktionen används vanligtvis som en del av ett uttryck när användaren vill att beräkningen ska använda det antal arbetsdagar som finns mellan två datum. Om en användare exempelvis vill beräkna den totala lönen som en anställd kommer tjäna på ett PAYE-avtal (pay as you earn).

Exempel på funktioner

Exempel	Resultat
`networkdays ('12/19/2013', '01/07/2014')`	Returnerar 14. Detta exempel tar inte hänsyn till semester.
`networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013')`	Returnerar 12. Detta exempel tar hänsyn till semester 12/25/2013 till 12/26/2013.
`networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014')`	Returnerar 10. I detta exempel beaktas två ledighetsperioder.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Grundläggande exempel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller projekt-ID, deras startdatum och slutdatum. Denna information laddas i en tabell som heter `Projects`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i `DateFormat`-systemvariabeln.
- Ytterligare ett fält, `net_work_days`, skapas för att beräkna antalet arbetsdagar som behövs i varje projekt.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Projects:
    Load
        *,
        networkdays(start_date,end_date) as net_work_days
```

```
        ;
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `start_date`
- `end_date`
- `net_work_days`

Resultattabell

id	startdatum	slutdatum	nettoarbetsdagar
1	01/01/2022	2022-01-18	12
2	2022-02-10	2022-02-17	6
3	2022-05-17	07/05/2022	36
4	2022-06-01	2022-06-12	8
5	2022-08-10	26/08/2022	13

Eftersom det inte finns några schemalagda semesterdagar (de hade funnits i det tredje argumentet i `networkdays()`-funktionen) subtraherar funktionen `start_date` från `end_date` samt alla helger för att beräkna antalet arbetsdagar mellan de två datumen.

Kalenderdiagram som framhäver arbetsdagarna för projekt 5 (inga semesterdagar)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Ovanstående kalender ger en visuell sammanfattning av projektet med `id` av 5. Projekt 5 börjar onsdag 10 augusti 2022 och slutar 26 augusti 2022. När alla lördagar och söndagar har ignorerats finns det 13 arbetsdagar mellan och inklusive de här två datumen.

Exempel 2 – enstaka semesterdag

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i föregående exempel.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i systemvariabeln `DateFormat`.
- Ytterligare ett fält, `net_work_days`, skapas för att beräkna antalet arbetsdagar som behövs i varje projekt.

I det här exemplet finns det en semesterdag som är schemalagd 19 augusti 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Projects:
```

```
    Load
        *,
        networkdays(start_date,end_date,'08/19/2022') as net_work_days
        ;
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `start_date`
- `end_date`
- `net_work_days`

Resultattabell

id	startdatum	slutdatum	nettoarbetsdagar
1	01/01/2022	2022-01-18	12
2	2022-02-10	2022-02-17	6
3	2022-05-17	07/05/2022	36
4	2022-06-01	2022-06-12	8
5	2022-08-10	26/08/2022	12

Den enda schemalagda semesterdagen anges som det tredje argumentet i `networkdays()`-funktionen.

Kalenderdiagram som framhäver arbetsdagarna för projekt 5 (en semesterdag)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19 Holiday	20
21	22	23	24	25	26	27
28	29	30	31			

Den ovanstående kalendern beskriver projekt 5 visuellt, så att modifieringen för att inkludera semesterdagen framgår. För projekt 5 inträffar den här semesterdagen fredagen 19 augusti 2022. Resultatet blir att det totala `net_work_days`-värdet för projekt 5 minskar med en dag från 13 till 12 dagar.

Exempel 3 – flera semesterdagar

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i systemvariabeln `DateFormat`.
- Ytterligare ett fält, `net_work_days`, skapas för att beräkna antalet arbetsdagar som behövs i varje projekt.

Men i det här exemplet finns det fyra schemalagda semesterdagar, från 18 till 21 augusti 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Projects:
    Load
        *,
        networkdays(start_date,end_date,'08/18/2022','08/19/2022','08/20/2022','08/21/2022')
as net_work_days
        ;
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `start_date`
- `end_date`
- `net_work_days`

Resultattabell

id	startdatum	slutdatum	nettoarbetsdagar
1	01/01/2022	2022-01-18	12
2	2022-02-10	2022-02-17	6
3	2022-05-17	07/05/2022	36
4	2022-06-01	2022-06-12	8
5	2022-08-10	26/08/2022	11

De fyra schemalagda semesterdagarna anges som en kommaseparerad lista från det tredje argumentet framåt i `networkdays()`-funktionen.

Kalenderdiagram som framhäver arbetsdagarna för projekt 5 (flera semesterdagar)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18 Holiday	19 Holiday	20
21	22	23	24	25	26	27
28	29	30	31			

Den ovanstående kalendern beskriver projekt 5 visuellt, så att modifieringen för att inkludera semesterdagarna framgår. Den här perioden med schemalagda semesterdagar inträffar under projekt 5, med två av dagarna på en torsdag och fredag. Resultatet blir att det totala `net_work_days`-värdet för projekt 5 minskar från 13 till 11 dagar.

Exempel 4 – enstaka semesterdag

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i systemvariabeln `DateFormat`.

Det finns en semesterdag som är schemalagd 19 augusti 2022.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. `net_work_days`-fältet beräknas via ett mått i ett diagramobjekt.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Projects:
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `start_date`
- `end_date`

Skapa följande mått:

```
= networkdays(start_date,end_date,'08/19/2022')
```

Resultattabell

id	startdatum	slutdatum	nettoarbetsdagar
1	01/01/2022	2022-01-18	12
2	2022-02-10	2022-02-17	6
3	2022-05-17	07/05/2022	36
4	2022-06-01	2022-06-12	8
5	2022-08-10	26/08/2022	12

Den enda schemalagda semesterdagen anges som det tredje argumentet i `networkdays()`-funktionen.

Kalenderdiagramet som visar nettoarbetsdagar med en semesterdag (diagramobjekt)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19 Holiday	20
21	22	23	24	25	26	27
28	29	30	31			

Den ovanstående kalendern beskriver projekt 5 visuellt, så att modifieringen för att inkludera semesterdagen framgår. För projekt 5 inträffar den här semesterdagen fredagen 19 augusti 2022. Resultatet blir att det totala `net_work_days`-värdet för projekt 5 minskar med en dag från 13 till 12 dagar.

now

I För klienthanterat Qlik Sense, returnerar denna funktion en tidsmarkör för aktuell tid. Funktionen returnerar värden i systemvariabelformatet **TimeStamp**. Som standard är värdet för **timer_mode** 1.

```
now([ timer_mode])
```

Returnerad datatyp: Dual

`now()`-funktionen kan användas antingen i laddningsskriptet eller i diagramobjekt.

Argument

Argument	Beskrivning
timer_mode	Kan ha följande värden: 0 (tid för senast slutförda dataladdning) 1 (tid då funktionen anropades) 2 (tid då applikationen öppnades) *Om du använder funktionen i ett dataladdningsskript, resulterar **timer_mode=0** i tiden för senast slutförda dataladdning, medan **timer_mode=1** visar tiden för funktionsanropet i den aktuella dataladdningen.*

`now()`-funktionen har stor påverkan på prestanda som kan ge rullningsproblem om funktionen används i tabelluttryck. Om det inte är absolut nödvändigt att använda den rekommenderar vi att använda funktionen `today()` i stället. Om det är nödvändigt att använda `now()` i en layout rekommenderar vi att använda icke-standardinställningarna `now(0)` eller `now(2)`, om det är möjligt, eftersom de inte kräver konstanta omberäkningar.

Användning

`now()`-funktionen används vanligen som en komponenten inom ett uttryck. Det kan exempelvis användas för att beräkna den återstående tiden i en produkts livscykel. `now()`-funktionen används i stället för `today()`-funktionen när uttrycket kräver att en del av en dag används.

I följande tabell finns en förklaring av resultatet som returneras av `now()`-funktionen, förutsatt att värdena för `timer_mode`-argumenten är olika:

Exempel på funktioner

Värde för timer_mode	Resultat om det används i laddningsskript	Resultat om det används i diagramobjekt
0	Returnerar en tidsmarkör, i formatet för systemvariabeln `TimeStamp`, för den sista slutförda dataladdningen före den senaste dataladdningen.	Returnerar en tidsmarkör, i formatet för systemvariabeln `TimeStamp`, för den senaste dataladdningen.
1	Returnerar en tidsmarkör, i formatet för systemvariabeln `TimeStamp`, för den senaste dataladdningen.	Returnerar tidsmarkören, i formatet för systemvariabeln `TimeStamp`, för funktionsanropet.

Värde för timer_mode	Resultat om det används i laddningsskript	Resultat om det används i diagramobjekt
2	Returnerar en tidsmarkör, i formatet för systemvariabeln `TimeStamp`, för när användarens session i programmet började. Detta uppdateras inte, såvida inte användaren laddar skriptet.	Returnerar en tidsmarkör, i formatet för systemvariabeln `TimeStamp`, för när användarens session i programmet började. Detta kommer att uppdateras när en ny session börjar eller data i applikationen laddas igen.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – generering av objekt med laddningsskript

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

I det här exemplet skapas tre variabler med `now()`-funktionen. För varje variabel används ett av `timer_mode`-alternativen för att visa hur det fungerar.

För att variablernas funktionalitet ska framgå laddar du skriptet en gång och, efter en kort stund, ytterligare en gång. Resultatet blir att `now(0)`- och `now(1)`-variablernas funktionalitet framgår genom att de visar olika värden.

Laddningsskript

```
LET vPreviousDataLoad = now(0);
LET vCurrentDataLoad = now(1);
LET vApplicationOpened = now(2);
```

Resultat

När data har lästs in en andra gång, skapar du tre textrutor enligt anvisningarna nedan.

Skapa först en textruta för data som tidigare har lästs in.

Gör följande:

1. Skapa en textruta genom att använda diagramobjektet **Text och bild**.
2. Lägg till följande mått till objektet:
   `=vPreviousDataLoad`
3. Under **Utseende**, väljer du **Show titles** och lägger till rubriken "Tidigare inläsningstid" till objektet.

Skapa sedan en textruta för de data som för närvarande läses in.

Gör följande:

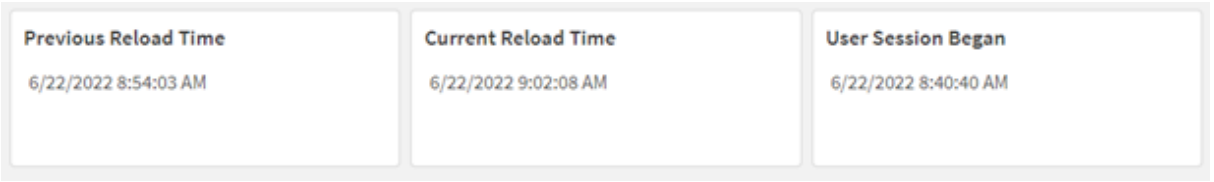
1. Skapa en textruta genom att använda diagramobjektet **Text och bild**.
2. Lägg till följande mått till objektet:
   `=vCurrentDataLoad`
3. Under **Utseende**, väljer du **Show titles** och lägger till rubriken "Aktuell inläsningstid" till objektet.

Skapa en sista textruta för att visa när användarens session i programmet startade.

Gör följande:

1. Skapa en textruta genom att använda diagramobjektet **Text och bild**.
2. Lägg till följande mått till objektet:
   `=vApplicationOpened`
3. Under **Utseende**, väljer du **Show titles** och lägger till rubriken "Användarsession startade" till objektet.

now() laddningsskriptvariabler



Bilden ovan visar exempelvärden för var och en av de skapade variablerna. Värdena skulle exempelvis kunna se ut som följer:

- Föregående laddningstid: 6/22/2022 8:54:03
- Aktuell laddningstid: 6/22/2022 9:02:08 AM
- Användarsessionen började: 6/22/2022 8:40:40 AM

Exempel 2 – generering av objekt utan laddningsskript

Laddningsskript och diagramuttryck

Översikt

I det här exemplet kommer du att skapa tre diagramobjekt med `now()`-funktionen utan att ladda några variabler eller data till programmet. För vart och ett av diagramobjekten används ett av `timer_mode`-alternativen för att visa hur det fungerar.

Det finns inget laddningsskript för det här exemplet.

Gör följande:

1. Öppna skriptredigeraren.
2. Klicka på **Läs in data** utan att ändra det befintliga laddningsskriptet.
3. Efter en kort stund laddar du skriptet på nytt.

Resultat

När data har lästs in en andra gång skapar du tre textrutor.

Först skapar du en textruta för den senaste dataladdningen.

Gör följande:

1. Skapa en textruta genom att använda diagramobjektet **Text och bild**.
2. Lägg till följande mått.
   `=now(0)`
3. Under **Utseende**, väljer du **Visa rubriker** och lägger till rubriken "Senaste datainläsning" till objektet.

Skapa sedan en textruta för att visa aktuell tid.

Gör följande:

1. Skapa en textruta genom att använda diagramobjektet **Text och bild**.
2. Lägg till följande mått:
   `=now(1)`
3. Under **Utseende**, väljer du **Visa rubriker** och lägger till rubriken "Aktuell tid" till objektet.

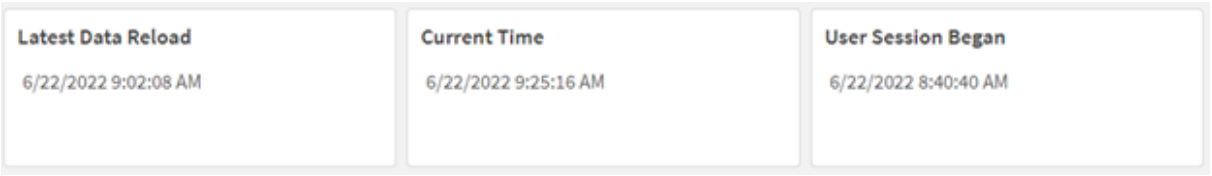
Skapa en sista textruta för att visa när användarens session i programmet startade.

Gör följande:

1. Skapa en textruta genom att använda diagramobjektet **Text och bild**.
2. Lägg till följande mått:
   `=now(2)`

3. Under **Utseende**, väljer du **Visa rubriker** och lägger till rubriken "Användarsession började" till objektet.

Exempel på `now()`-diagramobjekt



Bilden ovan visar exempelvärden för vart och ett av de skapade objekten. Värdena skulle exempelvis kunna se ut som följer:

- Data laddades senast: 6/22/2022 9:02:08
- Aktuell tid: 6/22/2022 9:25:16
- Användarsessionen började: 6/22/2022 8:40:40 AM

Diagramobjektet "Data laddades senast" använder ett `timer_mode`-värde på 0. Detta returnerar tidsmarkören för när data senast laddades om framgångsrikt.

Diagramobjektet "Aktuell tid" använder ett `timer_mode`-värde på 1. Detta returnerar aktuell tid enligt systemklockan. Om arket eller objektet uppdateras, kommer det här värdet att uppdateras.

Diagramobjektet "Användarsessionen började" använder ett `timer_mode`-värde på 2. Detta returnerar tidsmarkören för när programmet öppnades och användarens session började.

Exempel 3 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som består av ett lager för en åtgärd för mining av kryptovaluta, som laddas i en tabell som heter `Inventory`.
- Data med följande fält: `id`, `purchase_date` och `wph` (watt per timma).

Användaren behöver en tabell som visar den totala kostnaden för varje miningrigg hittills under månaden avseende strömförbrukning, sorterat på `id`.

Om allt stämmer ska det här värdet uppdateras när diagramobjektet uppdateras. Den aktuella kostnaden för el är 0,0678 USD per kWH.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Inventory:
Load
*
Inline
[
id,purchase_date,wph
8188,1/7/2022,1123
8189,1/19/2022,1432
8190,2/28/2022,1227
8191,2/5/2022,1322
8192,3/16/2022,1273
8193,4/1/2022,1123
8194,5/7/2022,1342
8195,5/16/2022,2342
8196,6/15/2022,1231
8197,6/26/2022,1231
8198,7/9/2022,1123
8199,7/22/2022,1212
8200,7/23/2022,1223
8201,7/27/2022,1232
8202,8/2/2022,1232
8203,8/8/2022,1211
8204,8/19/2022,1243
8205,9/26/2022,1322
8206,10/14/2022,1133
8207,10/29/2022,1231
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `id`.

Skapa följande mått:

```
=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
```

Om diagramobjektet var uppdaterades 22/06/2022 10:39:05 skulle det returnera följande resultat:

Resultattabell

id	=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
8188	$39.18
8189	$49.97
8190	$42.81
8191	46,13 USD
8192	$44.42

id	=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
8193	$39.18
8194	$46.83
8195	$81.72
8196	$42.95
8197	$42.95
8198	$39.18
8199	$42.29
8200	$42.67
8201	$42.99
8202	$42.99
8203	$42.25
8204	$43.37
8205	46,13 USD
8206	$39.53

Användaren vill att objektresultaten ska uppdateras varje gång objektet uppdateras. Därför finns `timer_mode`-argumentet med tillhandahållna instanser i `now()`-funktionen i uttrycket. Tidsmarkören för början av månaden, som identifieras genom att använda `now()`-funktionen som tidsmarkörargument i `monthstart()`-funktionen subtraheras från den aktuella tiden som identifieras med `now()`-funktionen. Detta ger den totala tidsmängden som har förflutit hittills under månaden, i dagar.

Det här värdet multipliceras med 24 (antalet timmar på ett dygn) och därefter med värdet i `wph`-fältet.

Resultatet divideras med 1000 så att det konverteras från watt per timma till kilowatt per timma innan det slutligen multipliceras med den angivna kWh-taxan.

quarter

Funktionen **quarter** returnerar ett heltal som representerar kvartalet för det angivna datumet. Det returnerade värdet är mellan 1 och 4. Funktionen kan användas i både laddningsskript och diagramuttryck.

Syntax:

```
quarter(date [, first_month_of_year])
```

Returnerad datatyp: heltal

quarter-funktionens argument

Argument	Beskrivning
date	Det datum som ska evalueras.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**. Om inget anges är standardvärdet 1.

Användning

Funktionen **quarter** är användbar när du vill jämföra sammanställningar efter kvartal. Funktionen kan till exempel användas om du vill se den totala försäljningen av produkter per kvartal. Den kan användas för att skapa nya fält i laddningsskriptet, samt i beräknade dimensioner och set-uttryck.

Exempel på funktioner

Exempel	Resultat
`quarter('10/29/2026')`	Returnerar 4.
`quartername('10/29/2026', 3)`	Returnerar 3.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel – laddningsskript

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2025 som läses in i en tabell som heter `Transactions`.

- Datumfältet tillhandahålls i formatet (M/D/YYYY) i systemvariabeln `DateFormat`.
- Skapandet av två fält:
    - `transaction_quarter` returnerar kvartalet då transaktionerna ägde rum.
    - `transaction_quarter_march_yearstart` returnerar transaktionernas kvartal, justerat för att sätta 1 mars som början på räkenskapsåret.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*,
quarter(date) as transaction_quarter,
quarter(date,3) as transaction_quarter_march_yearstart
;
Load
*
Inline
[
id,date,amount
8188,1/7/2025,17.17
8189,1/19/2025,37.23
8190,2/28/2025,88.27
8191,2/5/2025,57.42
8192,3/16/2025,53.80
8193,4/1/2025,82.06
8194,5/7/2025,40.39
8195,5/16/2025,87.21
8196,6/15/2025,95.93
8197,6/26/2025,45.89
8198,7/9/2025,36.23
8199,7/22/2025,25.66
8200,7/23/2025,82.77
8201,7/27/2025,69.98
8202,8/2/2025,76.11
8203,8/8/2025,25.12
8204,8/19/2025,46.23
8205,9/26/2025,84.21
8206,10/14/2025,96.24
8207,10/29/2025,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `transaction_quarter`
- `transaction_quarter_march_yearstart`

Resultattabell

date	transaction_quarter	transaction_quarter_march_yearstart
1/7/2025	1	4
1/19/2025	1	4
2/28/2025	1	4
2/5/2025	1	4
3/16/2025	1	1
4/1/2025	2	1
5/7/2025	2	1
5/16/2025	2	1
6/15/2025	2	2
6/26/2025	2	2
7/9/2025	3	2
7/22/2025	3	2
7/23/2025	3	2
7/27/2025	3	2
8/2/2025	3	2
8/8/2025	3	2
8/19/2025	3	2
9/26/2025	3	3
10/14/2025	4	3
10/29/2025	4	3

Exempel 2 – diagramuttryck

Diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningarna av transaktionskvartalen skapas som mått i ett diagramobjekt för programmet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2025,17.17
8189,1/19/2025,37.23
8190,2/28/2025,88.27
8191,2/5/2025,57.42
8192,3/16/2025,53.80
8193,4/1/2025,82.06
8194,5/7/2025,40.39
8195,5/16/2025,87.21
8196,6/15/2025,95.93
8197,6/26/2025,45.89
8198,7/9/2025,36.23
8199,7/22/2025,25.66
8200,7/23/2025,82.77
8201,7/27/2025,69.98
8202,8/2/2025,76.11
8203,8/8/2025,25.12
8204,8/19/2025,46.23
8205,9/26/2025,84.21
8206,10/14/2025,96.24
8207,10/29/2025,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `date`

Skapa följande mått:

- `=quarter(date)`
  Detta mått returnerar kvartalet då transaktionerna ägde rum.
- `=quarter(date,3)`
  Detta mått returnerar transaktionernas kvartal, justerat för att sätta 1 mars som början på räkenskapsåret.

Måtten skapas i diagramobjektet genom att använda `quarter()`-funktionen och skicka `date`-fältet som argument.

Resultattabell

date	=quarter(date)	=quarter(date,3)
1/7/2025	1	4
1/19/2025	1	4
2/28/2025	1	4
2/5/2025	1	4
3/16/2025	1	1
4/1/2025	2	1
5/7/2025	2	1
5/16/2025	2	1
6/15/2025	2	2
6/26/2025	2	2
7/9/2025	3	2
7/22/2025	3	2
7/23/2025	3	2
7/27/2025	3	2
8/2/2025	3	2
8/8/2025	3	2
8/19/2025	3	2
9/26/2025	3	3
10/14/2025	4	3
10/29/2025	4	3

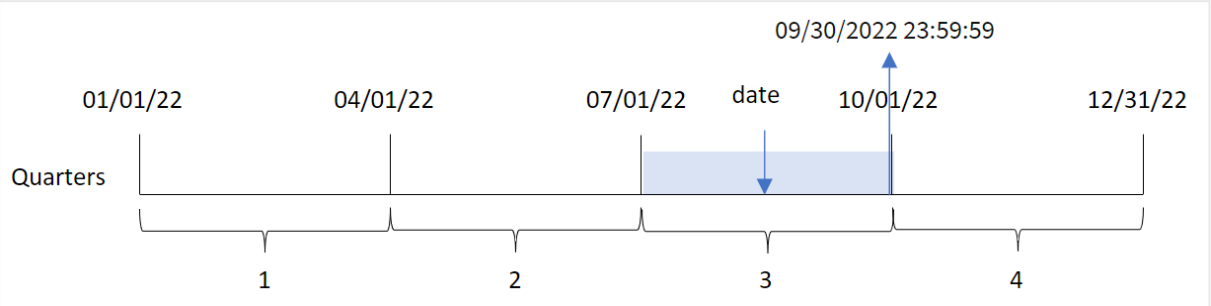
quarterend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden i det kvartal som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

```
QuarterEnd(date[, period_no[, first_month_of_year]])
```

Returnerad datatyp: Dual

Diagram för funktionen `quarterend()`



`quarterend()`-funktionen bestämmer vilket kvartal datumet infaller. Den returnerar sedan en tidsmarkör, i datumformat, för den sista millisekunden i den sista månaden i det kvartalet. Årets första månad är som standard januari. Du kan dock ändra vilken månad som anges som första månaden genom att använda `first_month_of_year`-argumentet i `quarterend()`-funktionen.

> *`quarterend()`-funktionen tar inte hänsyn till systemvariabeln `FirstMonthofYear`. Året börjar den 1 januari om inte `first_month_of_year`-argumentet används för att ändra det.*

Användning

`quarterend()`-funktionen används vanligen som en del av ett uttryck när du vill att beräkningen ska använda den del av kvartalet som ännu inte har inträffat. Till exempel om du vill beräkna den totala ränta som ännu inte uppkommit under kvartalet.

Argument

Argument	Beskrivning
date	Det datum eller den tidsmarkör som ska utvärderas.
period_no	**period_no** är ett heltal, där värdet 0 anger det kvartal som innehåller **date**. Negativa värden i **period_no** anger föregående kvartal och positiva värden anger efterföljande kvartal.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Du kan använda följande värden för att ställa in den första månaden på året i argumentet `first_month_of_year`:

first_month_of_year
values

Månad	värde
Februari	2
mars	3
april	4
Maj	5
juni	6
juli	7
augusti	8
september	9
oktober	10
November	11
december	12

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`quarterend('10/29/2005')`	Returnerar 12/31/2005 23:59:59.
`quarterend('10/29/2005', -1)`	Returns 09/30/2005 23:59:59.
`quarterend('10/29/2005', 0, 3)`	Returns 11/30/2005 23:59:59.

Exempel 1 – Grundläggande exempel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner under 2022 som läses in i en tabell som heter 'Transactions'.
- En föregående laddning som innehåller följande:
  - quarterend()-funktionen som har ställts in som end_of_quarter-fältet och returnerar en tidsmarkör för slutet av det kvartal när transaktionerna ägde rum.
  - timestamp()-funktionen som har ställts in som end_of_quarter_timestamp-fältet och returnerar den exakta tidsmarkören för slutet av det valda kvartalet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       quarterend(date) as end_of_quarter,
       timestamp(quarterend(date)) as end_of_quarter_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
```

```
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `end_of_quarter`
- `end_of_quarter_timestamp`

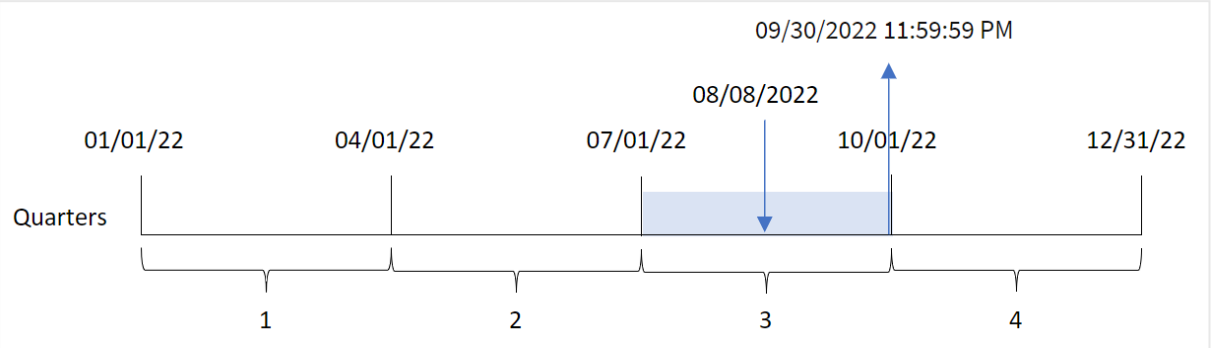
Resultattabell

Id	Datum	end_of_quarter	end_of_quarter_timestamp
8188	1/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8189	1/19/2022	03/31/2022	3/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	06/30/2022	6/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/16/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	09/30/2022	9/30/2022 11:59:59 PM
8199	7/22/2022	09/30/2022	9/30/2022 11:59:59 PM
8200	7/23/2022	09/30/2022	9/30/2022 11:59:59 PM
8201	7/27/2022	09/30/2022	9/30/2022 11:59:59 PM
8202	8/2/2022	09/30/2022	9/30/2022 11:59:59 PM
8203	8/8/2022	09/30/2022	9/30/2022 11:59:59 PM
8204	8/19/2022	09/30/2022	9/30/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	10/29/2022	12/31/2022	12/31/2022 11:59:59 PM

‘`end_of_quarter`’-fältet skapas i den föregående load-satsen genom att använda `quarterend()`-funktionen och skicka datumfältet som funktionens argument.

Funktionen `quarterend()` identifierar initialt vilket kvartal datumvärdet infaller och returnerar sedan en tidsmarkör för den sista millisekunden av det kvartalet.

Diagram för `quarterend()`-funktionen med kvartalsslutet för transaktion 8203 identifierat



Transaktion 8203 ägde rum 8 augusti. Funktionen `quarterend()` identifierar att transaktionen ägde rum under tredje kvartalet och returnerar den sista millisekunden av det kvartalet, vilket är 30 september kl. 23:59:59.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner från 2022 som laddas i en tabell som heter ‘`Transactions`’.
- En föregående laddning som innehåller följande:
  - `quarterend()`-funktionen som har ställts in som `previous_quarter_end`-fältet och returnerar en tidsmarkör för slutet av kvartalet innan transaktionerna ägde rum.
  - `timestamp()`-funktionen som har ställts in som `previous_end_of_quarter_timestamp`-fältet och returnerar den exakta tidsmarkören för slutet av kvartalet innan transaktionerna ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       quarterend(date, -1) as previous_quarter_end,
       timestamp(quarterend(date, -1)) as previous_quarter_end_timestamp
       ;
Load
*
```

```
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `previous_quarter_end`
- `previous_quarter_end_timestamp`

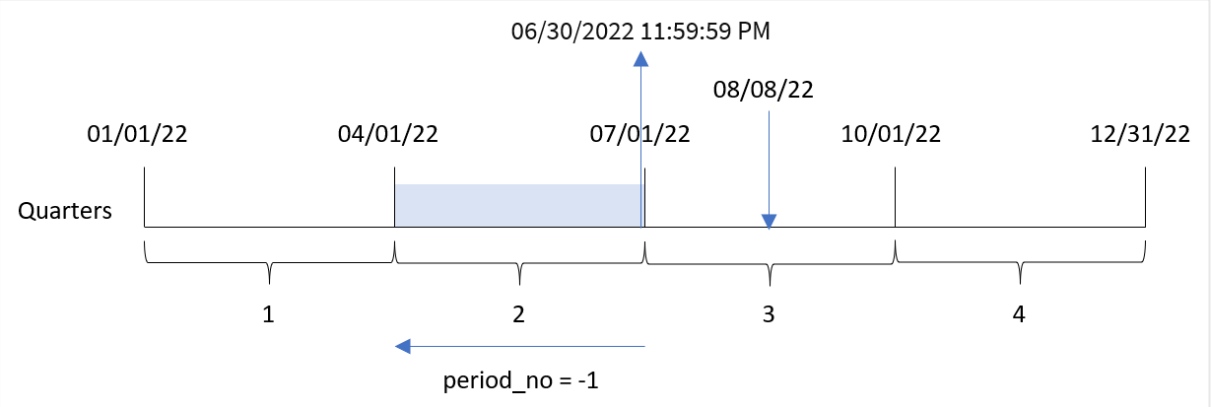
Resultattabell

Id	Datum	previous_quarter_end	previous_quarter_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	12/31/2021	12/31/2021 11:59:59 PM
8191	2/28/2022	12/31/2021	12/31/2021 11:59:59 PM
8192	3/16/2022	12/31/2021	12/31/2021 11:59:59 PM
8193	4/1/2022	03/31/2022	3/31/2022 11:59:59 PM
8194	5/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8195	5/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8196	6/15/2022	03/31/2022	3/31/2022 11:59:59 PM

Id	Datum	previous_quarter_end	previous_quarter_end_timestamp
8197	6/26/2022	03/31/2022	3/31/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	06/30/2022	6/30/2022 11:59:59 PM
8203	8/8/2022	06/30/2022	6/30/2022 11:59:59 PM
8204	8/19/2022	06/30/2022	6/30/2022 11:59:59 PM
8205	9/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8206	10/14/2022	09/30/2022	9/30/2022 11:59:59 PM
8207	10/29/2022	09/30/2022	9/30/2022 11:59:59 PM

Eftersom en `period_no` på `-1` används som förskjutningsargument i `quarterend()`-funktionen så identifierar funktionen först kvartalet då transaktionerna äger rum. Den skiftar sedan till närmast föregående kvartal och identifierar den sista millisekunden i det kvartalet.

Diagram för `quarterend()`-funktionen med en period_no på -1



Transaktion 8203 ägde rum 8 augusti. `quarterend()`-funktionen identifierar att kvartalet innan transaktionen ägde rum var mellan 1 april och 30 juni. Funktionen returnerar sedan den sista millisekunden i det kvartalet, 30 juni 23:59:59.

Exempel 3 – first_month_of_year

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner från 2022 som laddas i en tabell som heter 'Transactions'.
- En föregående laddning som innehåller följande:
  - quarterend()-funktionen som har ställts in som end_of_quarter-fältet och returnerar en tidsmarkör för slutet av det kvartal när transaktionerna ägde rum.
  - timestamp()-funktionen som har ställts in som end_of_quarter_timestamp-fältet och returnerar den exakta tidsmarkören för slutet av det valda kvartalet.

Men i det här exemplet är företagspolicyn att räkenskapsåret börjar 1 mars.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       quarterend(date, 0, 3) as end_of_quarter,
       timestamp(quarterend(date, 0, 3)) as end_of_quarter_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

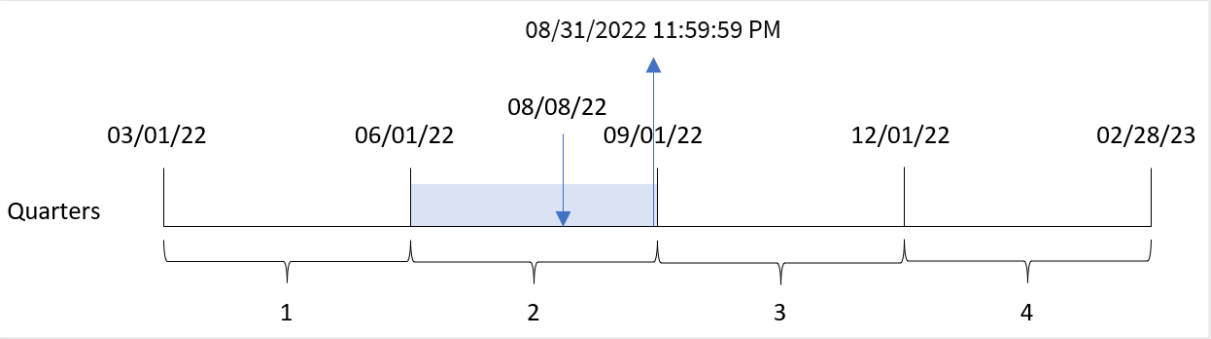
Resultat

Resultattabell

Id	Datum	end_of_quarter	end_of_quarter_timestamp
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM
8189	1/19/2022	02/28/2022	2/28/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8193	4/1/2022	05/31/2022	5/31/2022 11:59:59 PM
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	08/31/2022	8/31/2022 11:59:59 PM
8197	6/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	11/30/2022	11/30/2022 11:59:59 PM
8206	10/14/2022	11/30/2022	11/30/2022 11:59:59 PM
8207	10/29/2022	11/30/2022	11/30/2022 11:59:59 PM

Eftersom `first_month_of_year`-argumentet `3` används i `quarterend()`-funktionen flyttas årets början från 1 januari till 1 mars.

Diagram för `quarterend()`-funktionen med mars som årets första månad.



Transaktion 8203 ägde rum 8 augusti. Eftersom årets början är den 1 mars, inträffar årets kvartal mellan mar-maj, jun-aug, sep-nov och dec-feb.

`quarterend()`-funktionen identifierar att transaktionen ägde rum i kvartalet mellan början av juni och augusti och returnerar den sista millisekunden av det kvartalet, vilket är 31 augusti 23:59:59.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är dock datauppsättningen oförändrad och har lästs in i programmet. Beräkningen som returnerar en tidsmarkör för slutet av det kvartal då transaktionerna ägde rum skapas som ett mått i ett diagram i applikationen.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
```

```
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`

För att beräkna slutdatum för det kvartal då en transaktion äger rum skapar du följande mått:

- `=quarterend(date)`
- `=timestamp(quarterend(date))`

Resultattabell

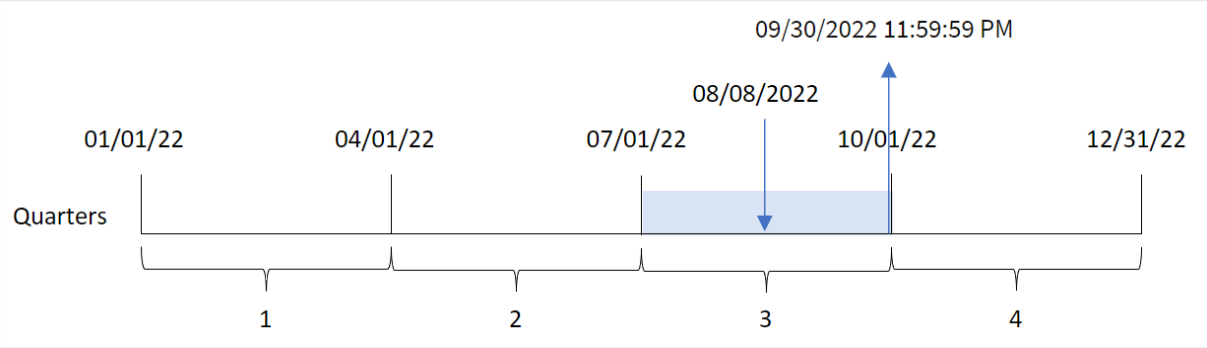
Id	Datum	=quarterend(date)	=timestamp(quarterend(date))
8188	1/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8189	1/19/2022	03/31/2022	3/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	06/30/2022	6/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/16/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	09/30/2022	9/30/2022 11:59:59 PM
8199	7/22/2022	09/30/2022	9/30/2022 11:59:59 PM
8200	7/23/2022	09/30/2022	9/30/2022 11:59:59 PM
8201	7/27/2022	09/30/2022	9/30/2022 11:59:59 PM
8202	8/2/2022	09/30/2022	9/30/2022 11:59:59 PM
8203	8/8/2022	09/30/2022	9/30/2022 11:59:59 PM

Id	Datum	=quarterend(date)	=timestamp(quarterend(date))
8204	8/19/2022	09/30/2022	9/30/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	10/29/2022	12/31/2022	12/31/2022 11:59:59 PM

'`end_of_quarter`'-fältet skapas i den föregående load-satsen genom att använda `quarterend()`-funktionen och skicka datumfältet som funktionens argument.

Funktionen `quarterend()` identifierar initialt vilket kvartal datumvärdet infaller och returnerar sedan en tidsmarkör för den sista millisekunden av det kvartalet.

Diagram för `quarterend()`-funktionen med kvartalsslutet för transaktion 8203 identifierat



Transaktion 8203 ägde rum 8 augusti. Funktionen `quarterend()` identifierar att transaktionen ägde rum under tredje kvartalet och returnerar den sista millisekunden av det kvartalet, vilket är 30 september kl. 23:59:59.

Exempel 5 – Scenario

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning läses in i en tabell som heter `Employee_Expenses`. Tabellen innehåller följande fält:
  - Anställnings-ID
  - Anställdas namn
  - Genomsnittliga dagliga kostnadsanspråk för varje anställd.

Slutanvändaren vill ha ett diagramobjekt som visar, efter anställnings-id och anställds namn, de beräknade kostnadsanspråken som fortfarande återstår för resten av kvartalet. Räkenskapsåret börjar i januari.

Laddningsskript

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `employee_id`
- `employee_name`

För att beräkna ackumulerad ränta skapar du följande mått:

- `=(quarterend(today(1))-today(1))*avg_daily_claim`

Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

employee_id	employee_name	=(quarterend(today(1))-today(1))*avg_daily_claim
182	Mark	$480.00
183	Deryck	$400.00
184	Dexter	$400.00
185	Sydney	$864.00
186	Agatha	$576.00

`quarterend()`-funktionen använder dagens datum som sitt enda argument och returnerar slutdatumet för den aktuella månaden. Därefter subtraherar den dagens datum från årets slutdatum, och uttrycket returnerar antalet dagar som återstår denna månad.

Detta värde multipliceras sedan med den genomsnittliga dagliga kostnadsanspråket för varje anställd för att beräkna det uppskattade värdet av anspråk som varje anställd förväntas göra under det återstående kvartalet.

quartername

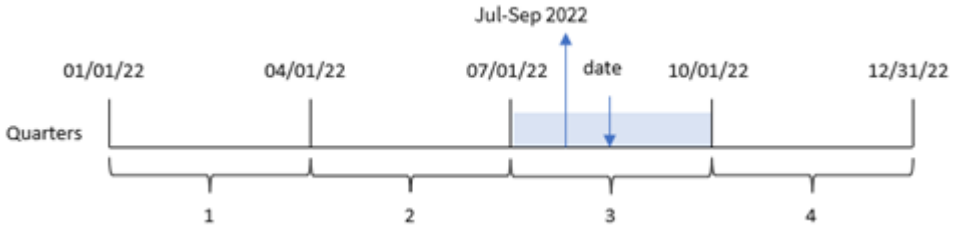
Denna funktion returnerar ett visningsvärde med kvartalets månader (formaterat enligt skriptvariabeln **MonthNames**) och år med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden av kvartalets första dag.

Syntax:

```
QuarterName(date[, period_no[, first_month_of_year]])
```

Returnerad datatyp: dual

Diagram för funktionen `quartername()`



`quartername()`-funktionen bestämmer vilket kvartal datumet infaller. Därefter returnerar det ett värde som visar start- och slutmånaderna i detta kvartal samt året. Det underliggande numeriska värdet för detta resultat är den första millisekunden i kvartalet.

Argument

Argument	Beskrivning
date	Det datum eller den tidsmarkör som ska utvärderas.
period_no	**period_no** är ett heltal, där värdet 0 anger det kvartal som innehåller **date**. Negativa värden i **period_no** anger föregående kvartal och positiva värden anger efterföljande kvartal.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Användning

Funktionen `quartername()` är användbar när du vill jämföra sammanställningar efter kvartal. Till exempel om du vill se den totala försäljningen av produkter per kvartal.

Denna funktion kan skapas i laddningsskriptet för att skapa ett fält i en Master Calendar-tabell. Eller användas direkt i ett diagram som en beräknad dimension.

I de här exemplen används datumformatet MM/DD/ÅÅÅÅ. Datumformatet anges i `SET DateFormat`-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Exempel på funktioner

Exempel	Resultat
`quartername('10/29/2013')`	Returnerar Oct-Dec 2013.
`quartername('10/29/2013', -1)`	Returnerar Jul-Sep 2013.
`quartername('10/29/2013', 0, 3)`	Returnerar Sep-Nov 2013.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Datum utan ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2022 som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i systemvariabeln `DateFormat`.
- Ett fält `transaction_quarter` skapas som returnerar kvartalet då transaktionerna ägde rum.

Lägg till din andra text här, efter behov, med listor mm.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        quartername(date) as transaction_quarter
        ;
Load
```

```
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `transaction_quarter`

Resultattabell

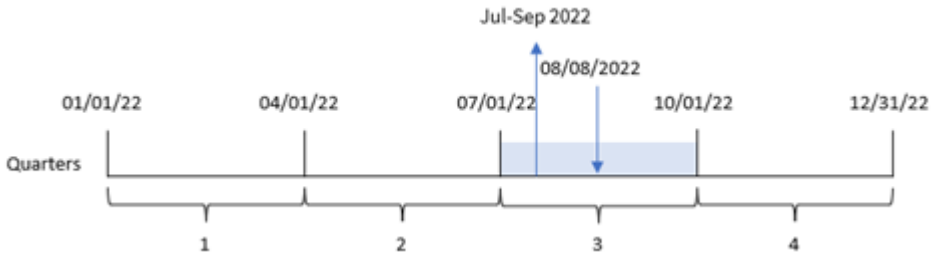
datum	transaction_quarter
1/7/2022	Jan-mars 2022
1/19/2022	Jan-mars 2022
2/5/2022	Jan-mars 2022
2/28/2022	Jan-mars 2022
3/16/2022	Jan-mars 2022
4/1/2022	Apr-jun 2022
5/7/2022	Apr-jun 2022
5/16/2022	Apr-jun 2022
6/15/2022	Apr-jun 2022
6/26/2022	Apr-jun 2022

datum	transaction_quarter
7/9/2022	Jul-sep 2022
7/22/2022	Jul-sep 2022
7/23/2022	Jul-sep 2022
7/27/2022	Jul-sep 2022
8/2/2022	Jul-sep 2022
8/8/2022	Jul-sep 2022
8/19/2022	Jul-sep 2022
9/26/2022	Jul-sep 2022
10/14/2022	Okt-dec 2022
10/29/2022	Okt-dec 2022

`transaction_quarter`-fältet skapades i den föregående load-satsen genom att använda `quartername()`-funktionen och skicka datumfältet som funktionens argument.

`quartername()`-funktionen identifierar initialt under vilket kvartal datumvärdet infaller. Därefter returnerar det ett värde som visar start- och slutmånaderna i detta kvartal samt året.

Diagram för `quartername()`-funktionen, exempel utan ytterligare argument



Transaktion 8203 ägde rum 8 augusti 2022. `quartername()`-funktionen identifierar att transaktionen ägde rum under det tredje kvartalet och returnerar därför jul-sep 2022. Månaderna visas i samma format som systemvariabeln `MonthNames`.

Exempel 2 – datum med argumentet period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Ett fält `previous_quarter` skapas som returnerar kvartalet innan transaktionerna ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        quartername(date,-1) as previous_quarter
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `previous_quarter`

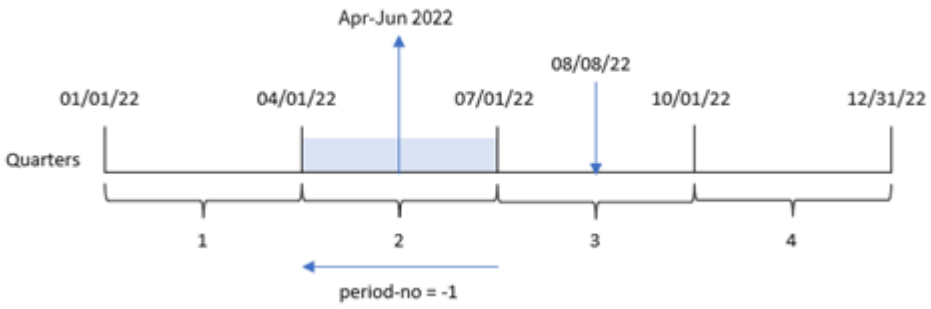
Resultattabell

datum	previous_quarter
1/7/2022	Okt-dec 2021

datum	previous_quarter
1/19/2022	Okt-dec 2021
2/5/2022	Okt-dec 2021
2/28/2022	Okt-dec 2021
3/16/2022	Okt-dec 2021
4/1/2022	Jan-mars 2022
5/7/2022	Jan-mars 2022
5/16/2022	Jan-mars 2022
6/15/2022	Jan-mars 2022
6/26/2022	Jan-mars 2022
7/9/2022	Apr-jun 2022
7/22/2022	Apr-jun 2022
7/23/2022	Apr-jun 2022
7/27/2022	Apr-jun 2022
8/2/2022	Apr-jun 2022
8/8/2022	Apr-jun 2022
8/19/2022	Apr-jun 2022
9/26/2022	Apr-jun 2022
10/14/2022	Jul-sep 2022
10/29/2022	Jul-sep 2022

I det här fallet, eftersom ett `period_no` på -1 användes som förskjutningsargument i `quartername()`-funktionen, identifierar funktionen först att transaktionerna ägde rum under det tredje kvartalet. Därefter skiftar den till närmast föregående kvartal och returnerar ett värde som visar start- och slutmånaderna i detta kvartal samt året.

Diagram med exempel på `quartername()`-funktionen, period_no



Transaktion 8203 ägde rum 8 augusti. `quartername()`-funktionen identifierar att kvartalet innan transaktionen ägde rum var mellan 1 april och 30 juni. Därför returnerar den apr-jun 2022.

Exempel 3 – datum med argumentet first_week_day

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet. Men i det här exemplet måste vi ställa in 1 mars som den första månaden i räkenskapsåret.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        quartername(date,0,3) as transaction_quarter
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

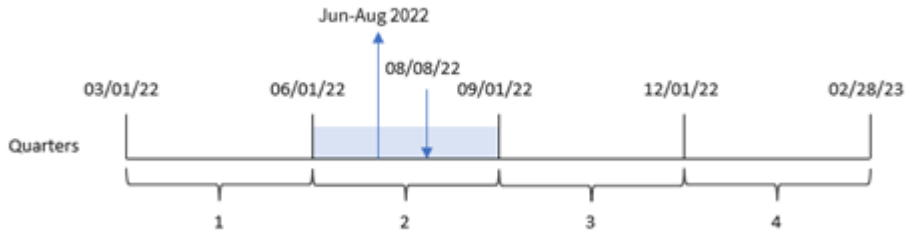
- `date`
- `transaction_quarter`

Resultattabell

datum	transaction_quarter
1/7/2022	dec-feb 2021
1/19/2022	dec-feb 2021
2/5/2022	dec-feb 2021
2/28/2022	dec-feb 2021
3/16/2022	mar-maj 2022
4/1/2022	mar-maj 2022
5/7/2022	mar-maj 2022
5/16/2022	mar-maj 2022
6/15/2022	jun-aug 2022
6/26/2022	jun-aug 2022
7/9/2022	jun-aug 2022
7/22/2022	jun-aug 2022
7/23/2022	jun-aug 2022
7/27/2022	jun-aug 2022
8/2/2022	jun-aug 2022
8/8/2022	jun-aug 2022
8/19/2022	jun-aug 2022
9/26/2022	sep-nov 2022
10/14/2022	sep-nov 2022
10/29/2022	sep-nov 2022

Eftersom `first_month_of_year`-argumentet 3 används i `quartername()`-funktionen i det här fallet flyttas årets början från 1 januari till 1 mars. Därför separeras årets kvartal till mars-maj, juni-augusti, september-november och december-februari.

Diagram med exempel på `quartername()`-funktionen, first_week_day



Transaktion 8203 ägde rum 8 augusti. `quartername()`-funktionen identifierar att transaktionen ägde rum under det andra kvartalet, mellan början av juni och slutet av augusti. Därför returnerar den jun-aug 2022.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som returnerar en tidsmarkör för slutet av kvartalet då transaktionerna ägde rum skapas som ett mått i ett diagramobjekt för programmet.

Laddningsskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
```

```
8205,9/26/022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

Skapa följande mått:

```
=quartername(date)
```

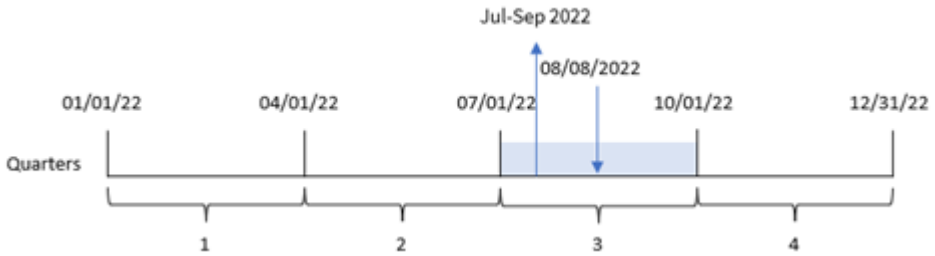
Resultattabell

datum	=quartername(date)
1/7/2022	Jan-mars 2022
1/19/2022	Jan-mars 2022
2/5/2022	Jan-mars 2022
2/28/2022	Jan-mars 2022
3/16/2022	Jan-mars 2022
4/1/2022	Apr-jun 2022
5/7/2022	Apr-jun 2022
5/16/2022	Apr-jun 2022
6/15/2022	Apr-jun 2022
6/26/2022	Apr-jun 2022
7/9/2022	Jul-sep 2022
7/22/2022	Jul-sep 2022
7/23/2022	Jul-sep 2022
7/27/2022	Jul-sep 2022
8/2/2022	Jul-sep 2022
8/8/2022	Jul-sep 2022
8/19/2022	Jul-sep 2022
9/26/2022	Jul-sep 2022
10/14/2022	Okt-dec 2022
10/29/2022	Okt-dec 2022

`transaction_quarter`-måttet skapades i den föregående load-satsen genom att använda `quartername()`-funktionen och skicka `date`-fältet som funktionens argument.

`quartername()`-funktionen identifierar initialt under vilket kvartal datumvärdet infaller. Därefter returnerar det ett värde som visar start- och slutmånaderna i detta kvartal samt året.

Diagram med exempel på `quartername()`-funktionen, diagramobjekt



Transaktion 8203 ägde rum 8 augusti 2022. `quartername()`-funktionen identifierar att transaktionen ägde rum under det tredje kvartalet och returnerar därför jul-sep 2022. Månaderna visas i samma format som systemvariabeln `MonthNames`.

Exempel 5 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner efter tidsmarkör som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i systemvariabeln `DateFormat`.

Slutanvändaren vill ha ett diagramobjekt som visar den totala försäljningen per kvartal för transaktionerna. Detta kan uppnås även när denna dimension inte är tillgänglig i datamodellen, med hjälp av `quartername()`-funktionen som en beräknad dimension i diagrammet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/7/2022',17.17
8189,'1/19/2022',37.23
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
```

```
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell.
2. Skapa en beräknad dimension med följande uttryck:
   `=quartername(date)`
3. Beräkna sedan total försäljning med hjälp av följande aggregeringsmått.
   `=sum(amount)`
4. Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

=quartername(date)	=sum(amount)
Jul-sep 2022	$446.31
Apr-jun 2022	$351.48
Jan-mars 2022	$253.89
Okt-dec 2022	$163.91

quarterstart

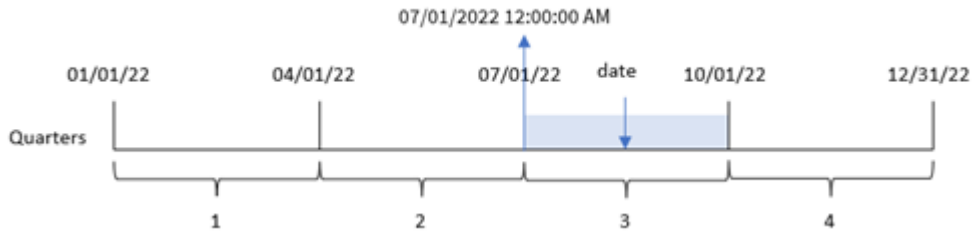
Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden i det kvartal som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

Syntax:

```
QuarterStart(date[, period_no[, first_month_of_year]])
```

Returnerad datatyp: dual

Diagram för funktionen `quarterstart()`



`quarterstart()`-funktionen bestämmer vilket kvartal `date` infaller. Den returnerar sedan en tidsmarkör, i datumformat, för den första millisekunden i den första månaden i det kvartalet.

Argument

Argument	Beskrivning
date	Det datum eller den tidsmarkör som ska utvärderas.
period_no	**period_no** är ett heltal, där värdet 0 anger det kvartal som innehåller **date**. Negativa värden i **period_no** anger föregående kvartal och positiva värden anger efterföljande kvartal.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Användning

`quarterstart()`-funktionen används vanligtvis som en del av ett uttryck när användaren vill att beräkningen ska använda den del av kvartalet som har förflutit hittills. Den kan exempelvis användas om en användare vill beräkna den ränta som har ackumulerats under ett kvartal fram till dagens datum.

Exempel på funktioner

Exempel	Resultat
`quarterstart('10/29/2005')`	Returnerar 10/01/2005.
`quarterstart('10/29/2005', -1 )`	Returnerar 07/01/2005.
`quarterstart('10/29/2005', 0, 3)`	Returnerar 09/01/2005.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2022 som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i `DateFormat`-systemvariabeln.
- Ett fält `start_of_quarter` skapas som returnerar en tidsmarkör för början av det kvartal då transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        quarterstart(date) as start_of_quarter,
        timestamp(quarterstart(date)) as start_of_quarter_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
```

```
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

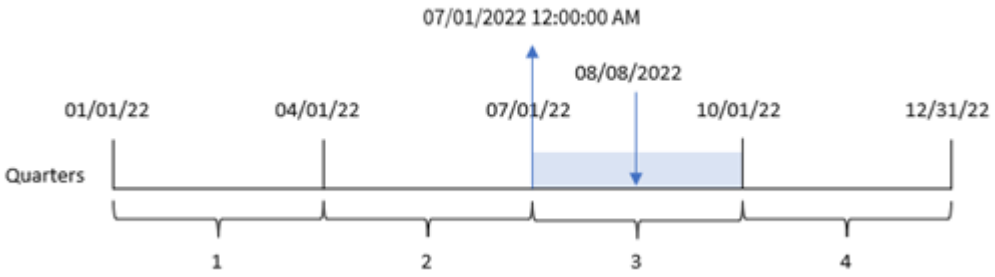
- `date`
- `start_of_quarter`
- `start_of_quarter_timestamp`

Resultattabell

datum	kvartalets_början	kvartalets_början_tidsmarkör
1/7/2022	01/01/2022	2022-01-01 00:00:00
1/19/2022	01/01/2022	2022-01-01 00:00:00
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	2022-01-01 00:00:00
3/16/2022	01/01/2022	2022-01-01 00:00:00
4/1/2022	04/01/2022	2021-04-01 00:00:00
5/7/2022	04/01/2022	2021-04-01 00:00:00
5/16/2022	04/01/2022	2021-04-01 00:00:00
6/15/2022	04/01/2022	2021-04-01 00:00:00
6/26/2022	04/01/2022	2021-04-01 00:00:00
7/9/2022	07/01/2022	2021-07-01 00:00:00
7/22/2022	2022-07-01	2021-07-01 00:00:00
7/23/2022	2022-07-01	2021-07-01 00:00:00
7/27/2022	2022-07-01	2021-07-01 00:00:00
8/2/2022	2022-07-01	2021-07-01 00:00:00
8/8/2022	2022-07-01	2021-07-01 00:00:00
8/19/2022	2022-07-01	2021-07-01 00:00:00
9/26/2022	2022-07-01	2021-07-01 00:00:00
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	2022-10-01 00:00:00

`start_of_quarter`-fältet skapades i den föregående load-satsen genom att använda `quarterstart()`-funktionen och skicka datumfältet som funktionens argument. `quarterstart()`-funktionen identifierar initialt under vilket kvartal datumvärdet infaller. Den returnerar sedan en tidsmarkör för den första millisekunden i det kvartalet.

Diagram för `quarterstart()`-funktionen, exempel utan ytterligare argument



Transaktion 8203 ägde rum 8 augusti. `quarterstart()`-funktionen identifierar att transaktionen ägde rum under det tredje kvartalet och returnerar den första millisekunden av det kvartalet, vilket är 1 juli 0:00:00.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Ett fält `previous_quarter_start` skapas som returnerar tidsmarkören för början av kvartalet innan transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        quarterstart(date,-1) as previous_quarter_start,
        timestamp(quarterstart(date,-1)) as previous_quarter_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
```

```
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `previous_quarter_start`
- `previous_quarter_start_timestamp`

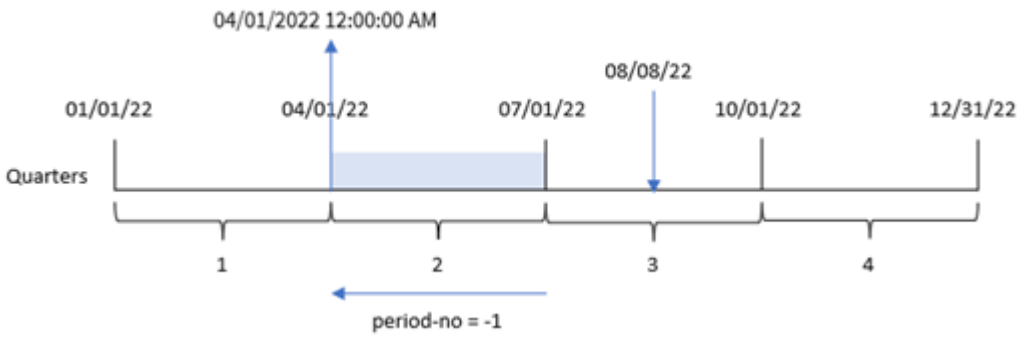
Resultattabell

datum	föregående_kvartals_start	föregående_kvartals_start_tidsmarkör
1/7/2022	2021-10-01	2021-10-01 00:00:00
1/19/2022	2021-10-01	2021-10-01 00:00:00
2/5/2022	2021-10-01	2021-10-01 00:00:00
2/28/2022	2021-10-01	2021-10-01 00:00:00
3/16/2022	2021-10-01	2021-10-01 00:00:00
4/1/2022	01/01/2022	2022-01-01 00:00:00
5/7/2022	01/01/2022	2022-01-01 00:00:00
5/16/2022	01/01/2022	2022-01-01 00:00:00
6/15/2022	01/01/2022	2022-01-01 00:00:00
6/26/2022	01/01/2022	2022-01-01 00:00:00
7/9/2022	04/01/2022	2021-04-01 00:00:00
7/22/2022	04/01/2022	2021-04-01 00:00:00
7/23/2022	04/01/2022	2021-04-01 00:00:00

datum	föregående_kvartals_start	föregående_kvartals_start_tidsmarkör
7/27/2022	04/01/2022	2021-04-01 00:00:00
8/2/2022	04/01/2022	2021-04-01 00:00:00
8/8/2022	04/01/2022	2021-01-04 00:00:00
8/19/2022	04/01/2022	2021-01-04 00:00:00
9/26/2022	04/01/2022	2021-01-04 00:00:00
10/14/2022	2022-01-07	2022-01-07 00:00:00
10/29/2022	2022-07-01	2022-07-01 00:00:00

I det här fallet, eftersom ett `period_no` på -1 användes som förskjutningsargument i `quarterstart()`-funktionen, identifierar funktionen först det kvartal då transaktionerna äger rum. Den skiftar sedan till närmast föregående kvartal och identifierar den första millisekunden i det kvartalet.

Diagram med exempel på `quarterstart()`-funktionen, period_no



Transaktion 8203 ägde rum 8 augusti. `quarterstart()`-funktionen identifierar att kvartalet innan transaktionen ägde rum var mellan 1 april och 30 juni. Den returnerar sedan den första millisekunden det kvartalet, 1 april 0:00:00.

Exempel 3 – first_month_of_year

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet. Men i det här exemplet måste vi ställa in 1 mars som den första månaden i räkenskapsåret.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
```

```
    Load
        *,
        quarterstart(date,0,3) as start_of_quarter,
        timestamp(quarterstart(date,0,3)) as start_of_quarter_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `start_of_quarter`
- `start_of_quarter_timestamp`

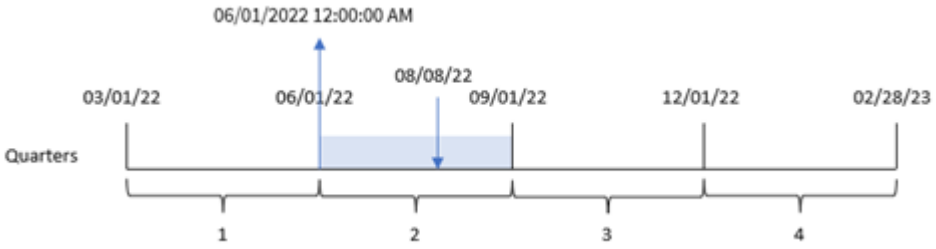
Resultattabell

datum	kvartalets_början	kvartalets_början_tidsmarkör
1/7/2022	12/01/2021	12/1/2021 12:00:00 AM
1/19/2022	2021-12-01	2021-12-01 00:00:00
2/5/2022	2021-12-01	2021-12-01 00:00:00
2/28/2022	2021-12-01	2021-12-01 00:00:00
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM

datum	kvartalets_början	kvartalets_början_tidsmarkör
4/1/2022	03/01/2022	2022-03-01 00:00:00
5/7/2022	2022-03-01	2022-03-01 00:00:00
5/16/2022	2022-03-01	2022-03-01 00:00:00
6/15/2022	06/01/2022	2022-06-01 00:00:00
6/26/2022	2022-06-01	2022-06-01 00:00:00
7/9/2022	06/01/2022	2022-06-01 00:00:00
7/22/2022	2022-06-01	2022-06-01 00:00:00
7/23/2022	2022-06-01	2022-06-01 00:00:00
7/27/2022	2022-06-01	2022-06-01 00:00:00
8/2/2022	2022-06-01	2022-06-01 00:00:00
8/8/2022	2022-06-01	2022-06-01 00:00:00
8/19/2022	2022-06-01	2022-06-01 00:00:00
9/26/2022	01/09/2022	9/1/2022 12:00:00 AM
10/14/2022	09/01/2022	2022-09-01 00:00:00
10/29/2022	2022-09-01	2022-09-01 00:00:00

Eftersom `first_month_of_year`-argumentet 3 används i `quarterstart()`-funktionen flyttas årets början från 1 januari till 1 mars.

Diagram med exempel på `quarterstart()`-funktionen, first_month_of_year



Transaktion 8203 ägde rum 8 augusti. Eftersom årets början är 1 mars inträffar kvartalen under året i mars-maj, juni-augusti, september-november och december-februari. `quarterstart()`-funktionen identifierar att transaktionen ägde rum i kvartalet mellan början av juni och augusti och returnerar den första millisekunden i det kvartalet, vilket är 1 juni 0:00:00.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som returnerar en tidsmarkör för slutet av kvartalet då transaktionerna ägde rum skapas som ett mått i ett diagramobjekt för programmet.

Laddningsskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

Lägg till följande mått:

- `=quarterstart(date)`
- `=timestamp(quarterstart(date))`

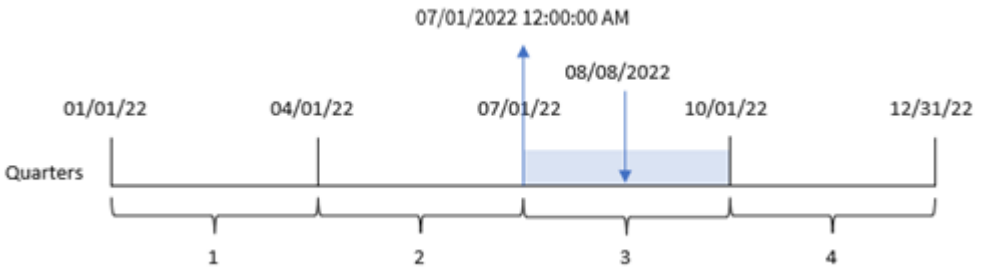
Resultattabell

datum	=kvartalsstart(datum)	=tidsmarkör(kvartalsstart(datum))
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	2022-10-01 00:00:00
7/9/2022	07/01/2022	2022-07-01 00:00:00
7/22/2022	2022-07-01	2022-07-01 00:00:00
7/23/2022	2022-07-01	2022-07-01 00:00:00
7/27/2022	2022-07-01	2022-07-01 00:00:00
8/2/2022	2022-07-01	7/1/2022 12:00:00 AM
8/8/2022	2022-07-01	2022-07-01 00:00:00
8/19/2022	2022-07-01	2022-07-01 00:00:00
9/26/2022	2022-07-01	2022-07-01 00:00:00
4/1/2022	04/01/2022	3/1/2022 12:00:00 AM
5/7/2022	04/01/2022	2022-04-01 00:00:00
5/16/2022	04/01/2022	2022-04-01 00:00:00
6/15/2022	04/01/2022	2022-04-01 00:00:00
6/26/2022	04/01/2022	2022-04-01 00:00:00
1/7/2022	01/01/2022	2022-01-01 00:00:00
1/19/2022	01/01/2022	2022-01-01 00:00:00
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	2022-01-01 00:00:00
3/16/2022	01/01/2022	2022-01-01 00:00:00

`start_of_quarter`-måttet skapades i den föregående load-satsen genom att använda `quarterstart()`-funktionen och skicka `date`-fältet som funktionens argument.

`quarterstart()`-funktionen identifierar vilket kvartal datumvärdet infaller och returnerar en tidsmarkör för den första millisekunden i det kvartalet.

Diagram med exempel på `quarterstart()`-funktionen, diagramobjekt



Transaktion 8203 ägde rum 8 augusti. `quarterstart()`-funktionen identifierar att transaktionen ägde rum under det tredje kvartalet och returnerar den första millisekunden i det kvartalet. Det returnerade värdet är 1 juli 0:00:00.

Exempel 5 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning lånesaldon som laddas i en tabell som heter `Loans`.
- Data som består av låne-ID:n, saldot vid kvartalets början och den enkla räntan som tas ut på varje lån per år.

Slutanvändaren vill ha ett diagramobjekt som visar, efter låne-id, den aktuella räntan som har ackumulerats för varje lån under kvartalet hittills.

Laddningsskript

```
Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:
   - `loan_id`
   - `start_balance`
2. Skapa sedan detta mått för att beräkna ackumulerad ränta:
   `=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)`
3. Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

loan_id	start_balance	=start_balance*(ränta*(idag(1)-kvartalsstart(idag(1)))/365)
8188	USD 10000.00	$15.07
8189	$15000.00	$128.84
8190	$17500.00	$63.29
8191	$21000.00	$107.59
8192	$90000.00	$1139.18

Genom att använda dagens datum som enda argument returnerar `quarterstart()`-funktionen slutdatumet för det aktuella året. Genom att subtrahera resultatet från det aktuella datumet returnerar uttrycket antalet dagar som har förflutit hittills under detta kvartal.

Detta värde multipliceras sedan med räntan och divideras med 365 för att returnera den effektiva räntan för perioden. Resultatet multipliceras sedan med lånets utgångssaldo för att returnera den upplupna räntan hittills detta kvartal.

second

Denna funktion returnerar ett heltal som motsvarar sekunden om decimaldelen av **expression** tolkas som en tidpunkt enligt standardtolkningen av tal.

Syntax:

```
second (expression)
```

Returnerad datatyp: heltal

Användning

Funktionen `second()` är användbar när du vill jämföra sammanställningar per sekund. Funktionen kan till exempel användas om du vill visa aktivitetsräkningsfördelningen per sekund.

Dessa dimensioner kan skapas antingen i laddningsskriptet genom att använda funktionen för att skapa ett fält i en Master Calendar-tabell, eller använda den direkt i ett diagram som en beräknad dimension.

Exempel på funktioner

Exempel	Resultat
`second( '09:14:36' )`	returnerar 36
`second( '0.5555' )`	returnerar 55 (eftersom 0.5555 = 13:19:55)

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Variabel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner efter tidsmarkör som läses in i en tabell som heter `Transactions`.
- `TimeStamp`-standardsystemvariabeln (`M/D/YYYY h:mm:ss[.fff] TT`) används.
- Ett fält skapas, `second`, för att beräkna när transaktionerna ägde rum.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        second(date) as second
        ;
Load
*
Inline
[
id,date,amount
9497,'01/05/2022 7:04:57 PM',47.25
```

```
9498,'01/03/2022 2:21:53 PM',51.75
9499,'01/03/2022 5:40:49 AM',73.53
9500,'01/04/2022 6:49:38 PM',15.35
9501,'01/01/2022 10:10:22 PM',31.43
9502,'01/05/2022 7:34:46 PM',13.24
9503,'01/06/2022 10:58:34 PM',74.34
9504,'01/06/2022 11:29:38 AM',50.00
9505,'01/02/2022 8:35:54 AM',36.34
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `second`

Resultattabell

datum	andra
01/01/2022 10:10:22 PM	22
01/02/2022 8:35:54 AM	54
01/03/2022 5:40:49 AM	49
01/03/2022 2:21:53 PM	53
01/04/2022 6:49:38 PM	38
01/05/2022 7:04:57 PM	57
01/05/2022 7:34:46 PM	46
01/06/2022 8:49:09 AM	9
01/05/2022 11:29:38 AM	38
01/06/2022 10:58:34 PM	34

Värdena i `second` f-fältet skapas genom att använda `second()`-funktionen och skicka datumet som uttrycket i föregående load-sats.

Exempel 2 – Diagramobjekt

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet. I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Värdena på `second` beräknas via ett mått i ett diagramobjekt.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
Load
*
Inline
[
id,date,amount
9497,'01/05/2022 7:04:57 PM',47.25
9498,'01/03/2022 2:21:53 PM',51.75
9499,'01/03/2022 5:40:49 AM',73.53
9500,'01/04/2022 6:49:38 PM',15.35
9501,'01/01/2022 10:10:22 PM',31.43
9502,'01/05/2022 7:34:46 PM',13.24
9503,'01/06/2022 10:58:34 PM',74.34
9504,'01/06/2022 11:29:38 AM',50.00
9505,'01/02/2022 8:35:54 AM',36.34
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:`date`.

Skapa följande mått:

```
=second(date)
```

Resultattabell

datum	=sekund(datum)
01/01/2022 10:10:22 PM	22
01/02/2022 8:35:54 AM	54
01/03/2022 5:40:49 AM	49
01/03/2022 2:21:53 PM	53
01/04/2022 6:49:38 PM	38
01/05/2022 7:04:57 PM	57
01/05/2022 7:34:46 PM	46
01/06/2022 8:49:09 AM	9
01/05/2022 11:29:38 AM	38
01/06/2022 10:58:34 PM	34

Värdena för `second` skapas genom att använda funktionen `second()` och skicka datumet som ett uttryck i ett mått för diagramobjektet.

Exempel 3 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med tidsmarkörer, som genereras för att representera trafiken till en viss festivals webbplats för biljettförsäljning. De här tidsmarkörerna och ett motsvarande `id` laddas i en tabell som heter `Web_Traffic`.
- `TimeStamp`-systemvariabeln `M/D/YYYY h:mm:ss[.fff] TT` används.

I det här scenariot fanns det 10 000 biljetter, som började säljas 20 maj 2021 9:00. En minut senare var biljetterna slutsålda.

Användaren vill ha ett diagramobjekt som visar antalet besök på webbplatsen per sekund.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

tmpTimeStampCreator:
load
    makedate(2022,05,20) as date
AutoGenerate 1;

join load
    maketime(9+floor(rand()*2),0,floor(rand()*59)) as time
autogenerate 10000;

Web_Traffic:
load
    recno() as id,
    timestamp(date + time) as timestamp
resident tmpTimeStampCreator;

drop table tmpTimeStampCreator;
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell.
2. Skapa sedan en beräknad dimension med följande uttryck:

```
=second(timestamp)
```

3. Skapa ett aggregeringsmått för att beräkna det totala antalet poster:
```
=count(id)
```

Resultattabellen kommer att se ut som nedanstående tabell, men med andra värden för aggregeringsmåttet:

Resultattabell

sekund(tidsmarkör)	=antal(id)
0	150
1	184
2	163
3	178
4	179
5	158
6	177
7	169
8	149
9	186
10	169
11	179
12	186
13	182
14	180
15	153
16	191
17	203
18	158
19	159
20	163
+ ytterligare 39 rader	

setdateyear

Den här funktionen tar som indata en **timestamp** och ett **year** och uppdaterar **timestamp** med det **year** som har angetts i indata.

Syntax:

```
setdateyear (timestamp, year)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
timestamp	En tidsangivelse i standard-Qlik Sense-format (ofta endast ett datum).
year	Ett fyrsiffrigt årtal.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`setdateyear ('29/10/2005', 2013)`	Returnerar '29/10/2013
`setdateyear ('29/10/2005 04:26:14', 2013)`	Returnerar '29/10/2013 04:26:14' För att se tidsdelen av tidsmarkören i en visualisering måste du ange talformatet som Datum och välja ett värde för Formatering som visar tidsvärden.

Exempel:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
SetYear:

Load *,

SetDateYear(testdates, 2013) as NewYear

Inline [

testdates

1/11/2012

10/12/2012

1/5/2013
```

```
2/1/2013

19/5/2013

15/9/2013

11/12/2013

2/3/2014

14/5/2014

13/6/2014

7/7/2014

4/8/2014

];
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn där året har angetts som 2013.

Resultattabell

testdatum	Nyår
1/11/2012	1/11/2013
10/12/2012	10/12/2013
2/1/2012	2/1/2013
1/5/2013	1/5/2013
19/5/2013	19/5/2013
15/9/2013	15/9/2013
11/12/2013	11/12/2013
2/3/2014	2/3/2013
14/5/2014	14/5/2013
13/6/2014	13/6/2013
7/7/2014	7/7/2013
4/8/2014	4/8/2013

setdateyearmonth

Den här funktionen tar som indata en **timestamp**, en **month** och ett **year** och uppdaterar **timestamp** med det **year** och den **month** som har angetts i indata. .

Syntax:

```
SetDateYearMonth (timestamp, year, month)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
timestamp	En tidsangivelse i standard-Qlik Sense-format (ofta endast ett datum).
year	Ett fyrsiffrigt årtal.
month	En en- eller tvåsiffrig månad.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`setdateyearmonth ('29/10/2005', 2013, 3)`	Returnerar '29/03/2013
`setdateyearmonth ('29/10/2005 04:26:14', 2013, 3)`	Returnerar '29/03/2013 04:26:14' För att se tidsdelen av tidsmarkören i en visualisering måste du ange talformatet som Datum och välja ett värde för Formatering som visar tidsvärden.

Exempel:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
SetYearMonth:

Load *,

SetDateYearMonth(testdates, 2013,3) as NewYearMonth

Inline [

testdates

1/11/2012

10/12/2012

2/1/2013
```

```
19/5/2013

15/9/2013

11/12/2013

14/5/2014

13/6/2014

7/7/2014

4/8/2014

];
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn där året har angetts som 2013.

Resultattabell

testdates	NewYearMonth
1/11/2012	1/3/2013
10/12/2012	10/3/2013
2/1/2012	2/3/2013
19/5/2013	19/3/2013
15/9/2013	15/3/2013
11/12/2013	11/3/2013
14/5/2014	14/3/2013
13/6/2014	13/3/2013
7/7/2014	7/3/2013
4/8/2014	4/3/2013

timezone

I För klienthanterat Qlik Sense returnerar den här funktionen tidszonen, enligt definitionen för den dator som kör Qlik-motorn.

Syntax:

```
TimeZone( )
```

Returnerad datatyp: dual

Exempel:

```
timezone( )
```

Om du vill se en annan tidszon i en mätning i din applikation, kan du använda funktionen `localtime()` i en mätning.

today

Denna funktion returnerar aktuell tid. Funktionen returnerar värden i systemvariabelformatet `DateFormat`.

```
today([ timer_mode])
```

Returnerad datatyp: Dual

`today()`-funktionen kan användas antingen i laddningsskriptet eller i diagramobjekt.

Standard `timer_mode` värdet är 1.

Argument

Argument	Beskrivning
timer_mode	Kan ha följande värden: 0 (dag för senast slutförda dataladdning) 1 (dag då funktionen anropades) 2 (dag då applikationen öppnades) *Om du använder funktionen i ett laddningsskript, resulterar **timer_mode=0** i dagen för senast slutförda dataladdning, medan **timer_mode=1** visar dagen för den aktuella dataladdningen.*

Exempel på funktioner

Värde för timer_mode	Resultat om det används i laddningsskript	Resultat om det används i diagramobjekt
0	Returnerar ett datum, i formatet för systemvariabeln `DateFormat`, för den sista slutförda dataladdningen före den senaste dataladdningen.	Returnerar ett datum, i formatet för systemvariabeln `DateFormat`, för den senaste dataladdningen.
1	Returnerar ett datum, i formatet för systemvariabeln `DateFormat`, för den senaste dataladdningen.	Returnerar ett datum, i formatet för systemvariabeln `DateFormat`, för funktionsanropet.
2	Returnerar ett datum, i formatet för systemvariabeln `DateFormat`, för när användarens session i programmet började. Detta uppdateras inte, såvida inte användaren laddar skriptet.	Returnerar datumet, i formatet för systemvariabeln `DateFormat`, för när användarens session i programmet började. Detta kommer att uppdateras när en ny session börjar eller data i applikationen laddas igen.

Användning

`today()`-funktionen används vanligen som en komponenten inom ett uttryck. Den kan exempelvis användas för att beräkna den ränta som har ackumulerats under en månad fram till dagens datum.

I följande tabell finns en förklaring av resultatet som returneras av `today()`-funktionen, förutsatt att värdena för `timer_mode`-argumenten är olika:

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – generering av objekt med laddningsskript

Laddningsskript och resultat

Översikt

I det här exemplet skapas tre variabler med `today()`-funktionen. För varje variabel används ett av `timer_mode`-alternativen för att visa hur det fungerar.

För att variablernas funktionalitet ska framgå laddar du skriptet en gång, och därefter efter 24 timmar ytterligare en gång. Resultatet blir att `today(0)`- och `today(1)`-variablernas funktionalitet framgår genom att de visar olika värden.

Laddningsskript

```
LET vPreviousDataLoad = today(0);
LET vCurrentDataLoad = today(1);
LET vApplicationOpened = today(2);
```

Resultat

När data har lästs in en andra gång, skapar du tre textrutor enligt anvisningarna nedan.

Skapa först en textruta för data som tidigare har lästs in.

Gör följande:

1. Skapa en textruta genom att använda diagramobjektet **Text och bild**.
2. Lägg till följande mått till objektet:
   `=vPreviousDataLoad`
3. Under **Utseende**, väljer du **Show titles** och lägger till rubriken "Tidigare inläsningstid" till objektet.

Skapa sedan en textruta för de data som för närvarande läses in.

Gör följande:

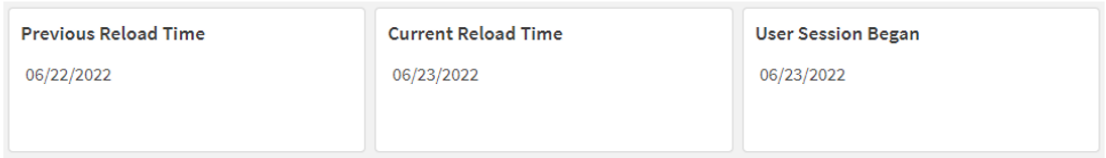
1. Skapa en textruta genom att använda diagramobjektet **Text och bild**.
2. Lägg till följande mått till objektet:
   `=vCurrentDataLoad`
3. Under **Utseende**, väljer du **Show titles** och lägger till rubriken "Aktuell inläsningstid" till objektet.

Skapa en sista textruta för att visa när användarens session i programmet startade.

Gör följande:

1. Skapa en textruta genom att använda diagramobjektet **Text och bild**.
2. Lägg till följande mått till objektet:
   `=vApplicationOpened`
3. Under **Utseende**, väljer du **Show titles** och lägger till rubriken "Användarsession startade" till objektet.

Diagram för variabler som skapats med `today()`-funktionen i laddningsskript



Bilden ovan visar exempelvärden för var och en av de skapade variablerna. Värdena skulle exempelvis kunna se ut som följer:

- Föregående laddningstid: 2022-06-22
- Aktuell laddningstid: 2022-06-23
- Användarsessionen började: 2022-06-23

Exempel 2 – generering av objekt utan laddningsskript

Laddningsskript och diagramuttryck

Översikt

I det följande exemplet skapas tre diagramobjekt med `today()`-funktionen. För vart och ett av diagramobjekten används ett av `timer_mode`-alternativen för att visa hur det fungerar.

Det finns inget laddningsskript för det här exemplet.

Resultat

När data har lästs in en andra gång skapar du tre textrutor.

Först skapar du en textruta för den senaste dataladdningen.

Gör följande:

1. Skapa en textruta genom att använda diagramobjektet **Text och bild**.
2. Lägg till följande mått.
   `=today(0)`
3. Under **Utseende**, väljer du **Visa rubriker** och lägger till rubriken "Senaste datainläsning" till objektet.

Skapa sedan en textruta för att visa aktuell tid.

Gör följande:

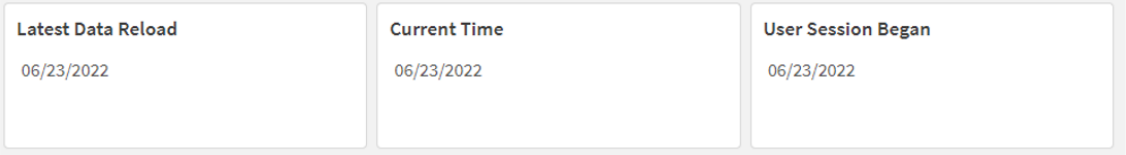
1. Skapa en textruta genom att använda diagramobjektet **Text och bild**.
2. Lägg till följande mått:
   `=today(1)`
3. Under **Utseende**, väljer du **Visa rubriker** och lägger till rubriken "Aktuell tid" till objektet.

Skapa en sista textruta för att visa när användarens session i programmet startade.

Gör följande:

1. Skapa en textruta genom att använda diagramobjektet **Text och bild**.
2. Lägg till följande mått:
   `=today(2)`
3. Under **Utseende**, väljer du **Visa rubriker** och lägger till rubriken "Användarsession började" till objektet.

Diagram för objekt som skapats med `today()`-funktionen utan laddningsskript



Bilden ovan visar exempelvärden för vart och ett av de skapade objekten. Värdena skulle exempelvis kunna se ut som följer:

- Senaste dataladdning: 2022-06-23
- Aktuell tid: 2022-06-23
- Användarsessionen började: 2022-06-23

Diagramobjektet "Data laddades senast" använder ett `timer_mode`-värde på 0. Detta returnerar tidsmarkören för när data senast laddades om framgångsrikt.

Diagramobjektet "Aktuell tid" använder ett `timer_mode`-värde på 1. Detta returnerar aktuell tid enligt systemklockan. Om arket eller objektet uppdateras, kommer det här värdet att uppdateras.

Diagramobjektet "Användarsessionen började" använder ett `timer_mode`-värde på 2. Detta returnerar tidsmarkören för när programmet öppnades och användarens session började.

Exempel 3 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning lånesaldon, som laddas i en tabell som heter `Loans`.
- Tabelldata med fält för låne-ID, saldot vid månadens början och den enkla räntan som tas ut på varje lån per år.

Slutanvändaren vill ha ett diagramobjekt som visar, efter låne-id, den aktuella räntan som har ackumulerats för varje lån under månaden hittills. Programmet laddas bara en gång i veckan, men användaren vill att resultaten ska uppdateras varje gång objektet eller applikationen uppdateras.

Laddningsskript

```
Loans:
Load
*
Inline
```

```
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell.
2. Lägg till följande fält som dimensioner:
   - `loan_id`
   - `start_balance`
3. Skapa sedan ett mått för att beräkna den ackumulerade räntan:
   `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
4. Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

loan_id	start_balance	=start_balance*(rate*(today(1)-monthstart(today(1)))/365)
8188	USD 10000.00	$16.44
8189	$15000.00	$58.56
8190	$17500.00	$28.77
8191	$21000.00	$48.90
8192	$90000.00	$517.81

Genom att använda `monthstart()`-funktionen för att returnera dagens datum som enda argument returnerar `today()`-funktionen startdatumet för den aktuella månaden. Genom att använda `today()`-funktionen igen, denna gång för att subtrahera resultatet från det aktuella datumet, returnerar uttrycket antalet dagar som har förflutit hittills denna månad.

Detta värde multipliceras sedan med räntan och divideras med 365 för att returnera den effektiva räntan för perioden. Resultatet multipliceras sedan med lånets utgångssaldo för att returnera den upplupna räntan hittills denna månad.

Eftersom värdet 1 används som `timer_mode`-argument i `today()`-funktionen i uttrycket, kommer det returnerade datumet vara dagens datum och resultaten uppdateras på motsvarande sätt varje gång diagramobjektet uppdateras (genom att öppna programmet, uppdatera sidan, flytta sig mellan ark osv.).

UTC

Returnerar aktuell Coordinated Universal Time.

Syntax:

```
UTC( )
```

Returnerad datatyp: dual

Exempel:

```
utc( )
```

week

Denna funktion returnerar ett heltal som motsvarar veckonumret som motsvarar angivet datum.

```
week(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

Returnerad datatyp: Heltal

Argument

Argument	Beskrivning
timestamp	Det datum eller den tidsmarkör som ska utvärderas.
first_week_day	Anger dagen då veckan startar. Om utelämnat används värdet för variabeln**FirstWeekDay**. De möjliga värdena **first_week_day** är 0 för måndag, 1 för tisdag, 2 för onsdag, 3 för torsdag, 4 för fredag, 5 för lördag och 6 för söndag. Se *FirstWeekDay (page 257)* för mer information om systemvariabeln.
broken_weeks	Om du inte anger **broken_weeks** används värdet för variabeln **BrokenWeeks** till att definiera om veckor är brutna eller inte.
reference_day	Om du inte anger **reference_day**, används värdet för variabeln **ReferenceDay** till att definiera vilken dag i januari som ska ställas in som referensdag för att definiera vecka 1. Som standard använder Qlik Sense-funktioner 4 som referensdag. Det betyder att vecka 1 måste innehålla 4 januari, eller med andra ord, vecka 1 måste alltid bestå av minst 4 dagar i januari.

`week()`-funktionen bestämmer i vilken vecka datumet infaller och returnerar veckans nummer.

I Qlik Sense, hämtas de regionala inställningarna när applikationen skapas, och motsvarande inställningar lagras i skriptet som miljövariabler. Dessa används för att fastställa veckonumret.

Detta innebär att de flesta europeiska applikationsutvecklare får följande miljövariabler, vilket motsvarar ISO 8601-definitionen:

```
Set FirstWeekDay =0;    // Monday as first week day
Set BrokenWeeks  =0;    // Use unbroken weeks
Set ReferenceDay =4;    // Jan 4th is always in week 1
```

En nordamerikansk applikationsutvecklare får ofta följande miljövariabler:

```
Set FirstWeekDay =6;    // Sunday as first week day
Set BrokenWeeks  =1;    // Use broken weeks
Set ReferenceDay =1;    // Jan 1st is always in week 1
```

Veckans första dag bestäms av systemvariabeln `FirstWeekDay`. Du kan även ändra den första veckodagen genom att använda argumentet `first_week_day` i `week()`-funktionen.

Om ditt program använder brutna veckor, börjar veckoräkningen den 1 januari och slutar dagen före systemvariabeln `FirstWeekDay` oavsett hur många dagar som har förlöpt.

Om ditt program använder obrutna veckor kan vecka 1 börja föregående år eller under de första dagarna i januari. Detta beror på hur du använder miljövariablerna `FirstWeekDay` och `ReferenceDay`.

Användning

Funktionen `The week()` är användbar när du vill jämföra sammanställningar efter veckor. Den kan till exempel användas om du vill se den totala försäljningen av produkter per vecka. `week()`-funktionen väljs före `weekname()` när användaren inte nödvändigtvis vill att beräkningen ska använda programmets systemvariabler `BrokenWeeks`, `FirstWeekDay` eller `ReferenceDay`.

Till exempel om du vill se den totala försäljningen av produkter per vecka.

Om programmet använder obrutna veckor kan vecka 1 innehålla datum från december föregående år eller exkludera datum i januari innevarande år. Om programmet använder brutna veckor kan vecka 1 innehålla färre än sju dagar.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exemplen nedan antar

```
Set DateFormat= 'MM/DD/YYYY';
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

Exempel på funktioner

Exempel	Resultat
`week('12/28/2021')`	Returnerar 52.
`week(44614)`	Returnerar 8, eftersom detta är serienumret för 2022-02-22.

Exempel	Resultat
`week('01/03/2021')`	Returnerar 53.
`week('01/03/2021',6)`	Returnerar 1.

Exempel 1 – Standardsystemvariabler

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för sista veckan år 2021 och de två första veckorna år 2022, som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i systemvariabeln `DateFormat`.
- Ett fält `week_number` skapas som returnerar det år och veckonummer då transaktionen ägde rum.
- Ett fält `week_day` skapas som visar veckodagsvärdet för varje transaktionsdatum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
        *,
        WeekDay(date) as week_day,
        Week(date) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
```

```
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `week_day`
- `week_number`

Resultattabell

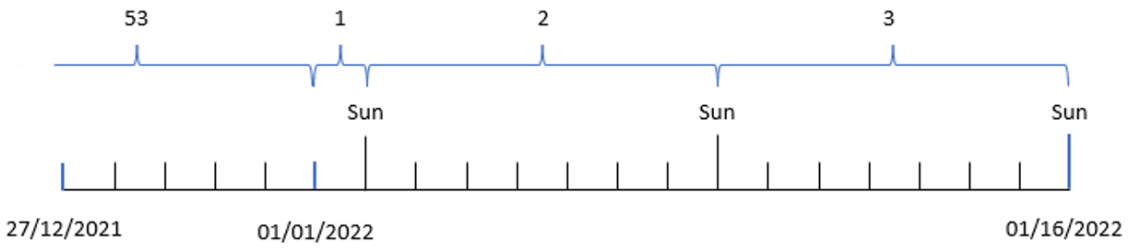
id	datum	week_day	veckonummer
8183	12/27/2021	mån	53
8184	2021-12-28	tis	53
8185	29/12/2021	ons	53
8186	30/12/2021	tors	53
8187	12/31/2021	fre	53
8188	01/01/2022	lör	1
8189	01/02/2022	sön	2
8190	01/03/2022	mån	2
8191	01/04/2022	tis	2
8192	01/05/2022	ons	2
8193	01/06/2022	tors	2
8194	01/07/2022	fre	2
8195	2022-01-08	lör	2
8196	2022-01-09	sön	3
8197	10/01/2022	mån	3
8198	11/01/2022	tis	3
8199	12/01/2022	ons	3
8200	13/01/2022	tors	3
8201	14/01/2022	fre	3

week_number-fältet skapas i den föregående load-satsen genom att använda week()-funktionen och skicka date-fältet som funktionens argument.

Inga andra parametrar skickas till funktoinen och därför gäller följande standardvariabler som påverkar week()-funktionen:

- BrokenWeeks: Veckoräkningen börjar 1 januari
- FirstWeekDay: Veckans första dag är söndag

Diagram över week()-funktionen med standardssystemvariabler



Eftersom programmet använder standardsystemvariabeln BrokenWeeks börjar vecka 1 den 1 januari, en lördag.

På grund av standardsystemvariabeln FirstWeekDay börjar veckorna på en söndag. Den första söndagen efter 1 januari inträffar 2 januari. Därför börjar vecka 2 då.

Exempel 2 – first_week_day

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Ett fält week_number skapas som returnerar det år och veckonummer då transaktionen ägde rum.
- Ett fält week_day skapas som visar veckodagsvärdet för varje transaktionsdatum.

I det här exemplet vill vi ställa in att arbetsveckan börjar på tisdagar.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
```

```
        *,
        WeekDay(date) as week_day,
        Week(date,1) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `week_day`
- `week_number`

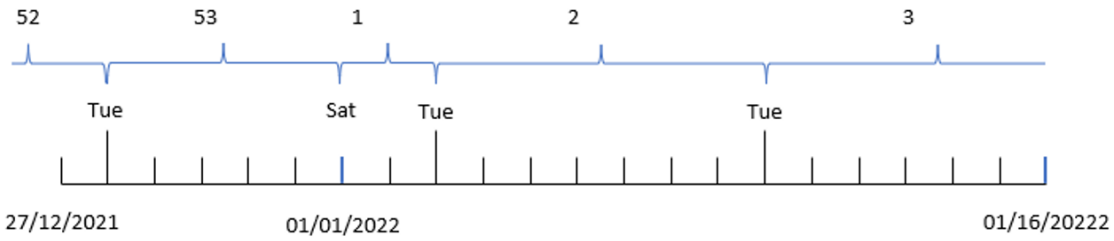
Resultattabell

id	datum	week_day	veckonummer
8183	12/27/2021	mån	52
8184	2021-12-28	tis	53
8185	29/12/2021	ons	53
8186	30/12/2021	tors	53
8187	12/31/2021	fre	53
8188	01/01/2022	lör	1

id	datum	week_day	veckonummer
8189	01/02/2022	sön	1
8190	01/03/2022	mån	1
8191	01/04/2022	tis	2
8192	01/05/2022	ons	2
8193	01/06/2022	tors	2
8194	01/07/2022	fre	2
8195	2022-01-08	lör	2
8196	2022-01-09	sön	2
8197	10/01/2022	mån	2
8198	11/01/2022	tis	3
8199	12/01/2022	ons	3
8200	13/01/2022	tors	3
8201	14/01/2022	fre	3

Programmet använder fortfarande brutna veckor. Men argumentet `first_week_day` har ställts in till 1 i `week()`-funktionen. Detta ställer in veckans första dag till en tisdag.

Diagram med exempel på `week()`-funktionen, first_week_day



Programmet använder standardsystemvariabeln `BrokenWeeks`, så vecka 1 börjar 1 januari, en lördag.

`first_week_day`-argumentet `week()`-funktionen ställer in veckans första dag till en tisdag. Därför börjar vecka 53 den 28 december 2021.

Men eftersom funktionen fortfarande använder brutna veckor kommer vecka 1 bara att vara två dagar lång, eftersom den första tisdagen efter 1 januari inträffar 3 januari.

Exempel 3 – unbroken_weeks

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet.

I det här exemplet använder vi obrutna veckor.

Laddningsskript

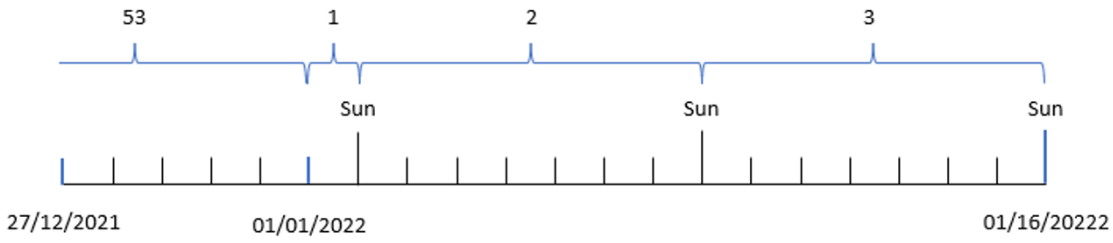
```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
        *,
        WeekDay(date) as week_day,
        Week(date,6,0) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `week_day`
- `week_number`

Diagram med exempel på `week()`-funktionen, diagramobjekt



Resultattabell

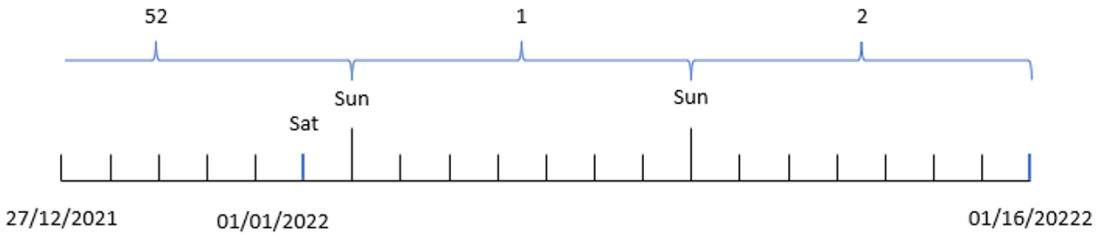
id	datum	week_day	veckonummer
8183	12/27/2021	mån	52
8184	2021-12-28	tis	52
8185	29/12/2021	ons	52
8186	30/12/2021	tors	52
8187	12/31/2021	fre	52
8188	01/01/2022	lör	52
8189	01/02/2022	sön	1
8190	01/03/2022	mån	1
8191	01/04/2022	tis	1
8192	01/05/2022	ons	1
8193	01/06/2022	tors	1
8194	01/07/2022	fre	1
8195	2022-01-08	lör	1
8196	2022-01-09	sön	2
8197	10/01/2022	mån	2
8198	11/01/2022	tis	2

id	datum	week_day	veckonummer
8199	12/01/2022	ons	2
8200	13/01/2022	tors	2
8201	14/01/2022	fre	2

Parametern first_week_date sätts till 1, vilket innebär att tisdag blir första dagen i veckan. Parametern broken_weeks sätts till 0, vilket tvingar funktionen att använda obrutna veckor. Slutligen sätter den tredje parametern reference_day till 2.

Parametern `first_week_date` sätts till 6, vilket innebär att söndag blir första dagen i veckan. Parametern `broken_weeks` sätts till 0, vilket tvingar funktionen att använda obrutna veckor.

Diagram för week()-funktionen, med obrutna veckor i exemplet



Eftersom obrutna veckor används börjar inte vecka 1 nödvändigtvis 1 januari, men den måste i stället ha minst fyra dagar. I datauppsättningen slutar vecka 52 därför på lördag 1 januari 2022. Vecka 1 börjar sedan vid `FirstWeekDay` systemvariabeln, som är söndag 2 januari. Den här veckan slutar följande lördag, 8 januari.

Exempel 4 – reference_day

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det tredje exemplet.
- Ett fält `week_number` skapas som returnerar det år och veckonummer då transaktionen ägde rum.
- Ett fält `week_day` skapas som visar veckodagsvärdet för varje transaktionsdatum.

Dessutom måste följande villkor vara uppfyllda:

- Arbetsveckan börjar på en tisdag.
- Företaget använder obrutna veckor.

- Det `reference_day` värdet är 2. Med andra ord, det minsta möjliga antalet dagar i januari i vecka 1 kommer vara 2.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
        *,
        WeekDay(date) as week_day,
        Week(date,1,0,2) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

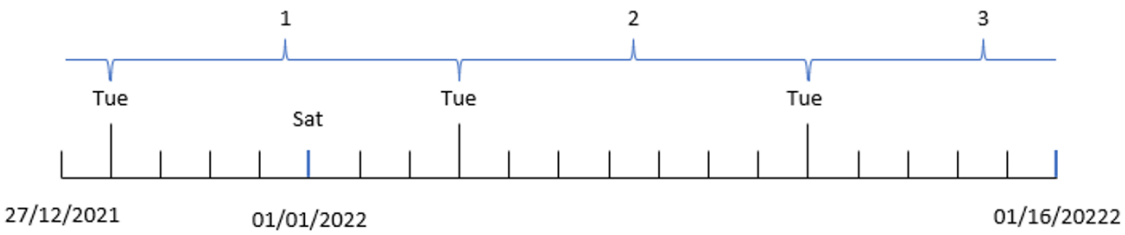
- `id`
- `date`
- `week_day`
- `week_number`

Resultattabell

id	datum	week_day	veckonummer
8183	12/27/2021	mån	52
8184	2021-12-28	tis	1
8185	29/12/2021	ons	1
8186	30/12/2021	tors	1
8187	12/31/2021	fre	1
8188	01/01/2022	lör	1
8189	01/02/2022	sön	1
8190	01/03/2022	mån	1
8191	01/04/2022	tis	2
8192	01/05/2022	ons	2
8193	01/06/2022	tors	2
8194	01/07/2022	fre	2
8195	2022-01-08	lör	2
8196	2022-01-09	sön	2
8197	10/01/2022	mån	2
8198	11/01/2022	tis	3
8199	12/01/2022	ons	3
8200	13/01/2022	tors	3
8201	14/01/2022	fre	3

Parametern `first_week_date` sätts till 1, vilket innebär att tisdag blir första dagen i veckan. Parametern `broken_weeks` sätts till 0, vilket tvingar funktionen att använda obrutna veckor. Slutligen sätter den tredje parametern `reference_day`-parametern till 2.

Diagram med exempel på `week()`-funktionen, reference_day



Eftersom funktionen använder obrutna veckor och ett `reference_day`-värde på 2 som parameter behöver det bara ingå två dagar i januari i vecka 1. Eftersom den första dagen i veckan är tisdag börjar vecka 1 den 28 december 2021 och slutar måndag 3 januari 2022.

Exempel 5 – exempel på diagramobjekt

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som returnerar veckonumret skapas som ett mått i ett diagramobjekt.

Laddningsskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell.
2. Lägg till följande fält som dimensioner:
   - `id`
   - `date`
3. Skapa sedan följande mått:
   `=week (date)`
4. Skapa ett mått, `week_day` som visar veckodagsvärdet vör varje transaktionsdatum:
   `=weekday(date)`

Resultattabell

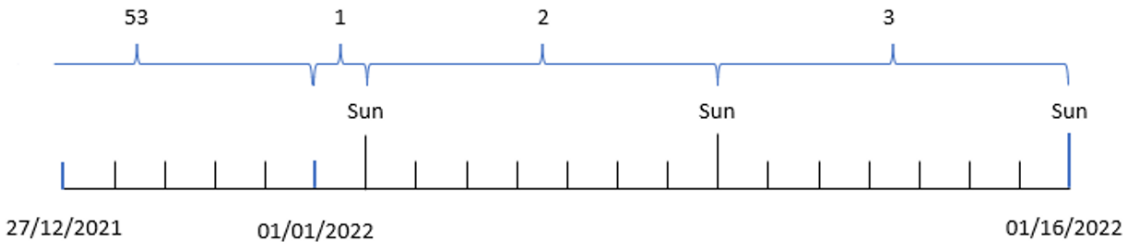
id	datum	=vecka(datum)	=weekday(date)
8183	12/27/2021	53	mån
8184	2021-12-28	53	tis
8185	29/12/2021	53	ons
8186	30/12/2021	53	tors
8187	12/31/2021	53	fre
8188	01/01/2022	1	lör
8189	01/02/2022	2	sön
8190	01/03/2022	2	mån
8191	01/04/2022	2	tis
8192	01/05/2022	2	ons
8193	01/06/2022	2	tors
8194	01/07/2022	2	fre
8195	2022-01-08	2	lör
8196	2022-01-09	3	sön
8197	10/01/2022	3	mån
8198	11/01/2022	3	tis
8199	12/01/2022	3	ons
8200	13/01/2022	3	tors
8201	14/01/2022	3	fre

`week_number`-fältet skapas i den föregående load-satsen genom att använda `week()`-funktionen och skicka `date`-fältet som funktionens argument.

Inga andra parametrar skickas till funktoinen och därför gäller följande standardvariabler som påverkar `week()`-funktionen:

- `BrokenWeeks`: Veckoräkningen börjar 1 januari
- `FirstWeekDay`: Veckans första dag är söndag

Diagram med exempel på `week()`-funktionen, diagramobjekt



Eftersom programmet använder standardsystemvariabeln `BrokenWeeks` börjar vecka 1 den 1 januari, en lördag.

På grund av standardsystemvariabeln `FirstWeekDay` börjar veckorna på en söndag. Den första söndagen efter 1 januari inträffar 2 januari. Därför börjar vecka 2 då.

Exempel 6 – scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för sista veckan år 2019 och de två första veckorna år 2020 som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (ÅÅÅÅ-MM-DD) i systemvariabeln `DateFormat`.

Programmet använder främst brutna veckor i sin instrumentpanel. Slutanvändaren vill emellertid ha ett diagramobjekt som visar den totala försäljningen per vecka med obrutna veckor. Referensdagen ska vara 2 januari, med veckans början på tisdag. Detta kan uppnås även när denna dimension inte är tillgänglig i datamodellen, med hjälp av `week()`-funktionen som en beräknad dimension i diagrammet.

Laddningsskript

```
SET BrokenWeeks=1;
SET ReferenceDay=0;
SET DateFormat='MM/DD/YYYY';

Transactions:
```

```
Load
*
Inline
[
id,date,amount
8183,12/27/2019,58.27
8184,12/28/2019,67.42
8185,12/29/2019,23.80
8186,12/30/2019,82.06
8187,12/31/2019,40.56
8188,01/01/2020,37.23
8189,01/02/2020,17.17
8190,01/03/2020,88.27
8191,01/04/2020,57.42
8192,01/05/2020,53.80
8193,01/06/2020,82.06
8194,01/07/2020,40.56
8195,01/08/2020,53.67
8196,01/09/2020,26.63
8197,01/10/2020,72.48
8198,01/11/2020,18.37
8199,01/12/2020,45.26
8200,01/13/2020,58.23
8201,01/14/2020,18.52
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell.
2. Skapa följande beräknade dimension:
   `=week(date)`
3. Skapa sedan följande aggregeringsmått:
   `=sum(amount)`
4. Ange måttens **Nummerformatering** till **Pengar**.
5. Välj **sortering**smenyn och ta bort anpassad sortering för den beräknade dimensionen.
6. Avmarkera alternativen **Sortera numeriskt** och **Sortera alfabetiskt**.

Resultattabell

vecka(datum)	summa(belopp)
52	$125.69
53	$146.42
1	$200.09
2	$347.57
3	$122.01

weekday

Denna funktion returnerar ett dualt värde med:

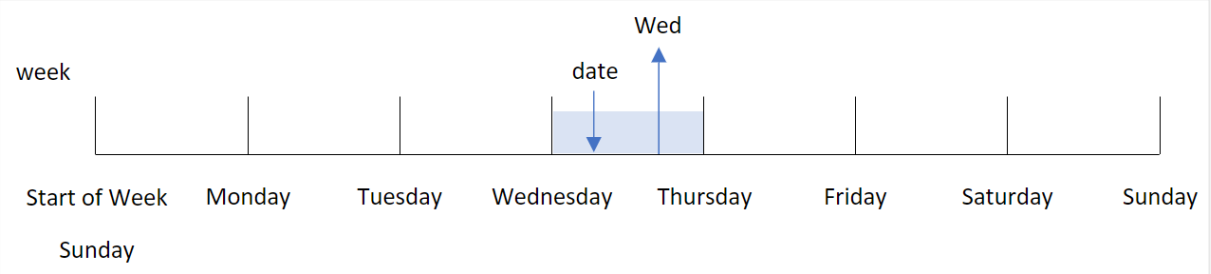
- Ett namn på en dag som definierat i miljövariabeln **DayNames**.
- Ett heltal mellan 0 och 6 som motsvarar den nominella veckodagen (0-6).

```
weekday(date [,first_week_day=0])
```

Returnerad datatyp: Dual

`weekday()`-funktionen avgör vilken veckodag ett datum inträffar. Det returnerar sedan ett strängvräde som representerar den dagen.

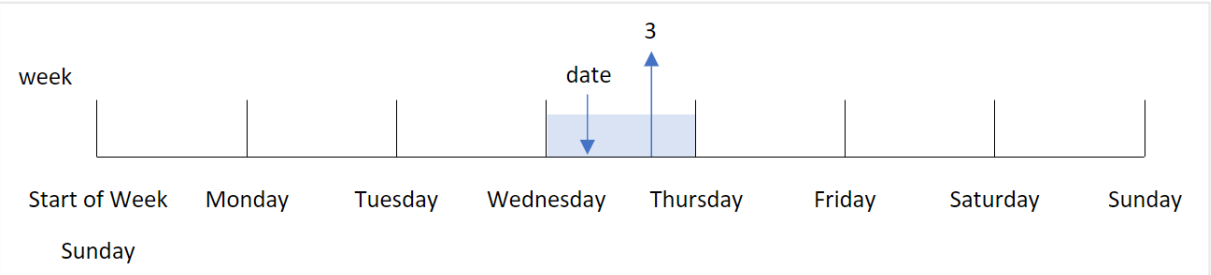
Diagram för `weekday()`-funktionen som returnerar namnet på dagen då ett datum infaller



Resultatet returnerar nummervärdet som motsvarar den veckodagen (0-6), baserat på veckans första dag. Om den första dagen i veckan har satts till söndag returnerar en onsdag nummervärdet `3`. Den här startdagen bestäms antingen av systemvariabeln `FirstWeekDay` eller av funktionsparametern `first_week_day`.

Du kan använda det här talvärdet som en del av ett aritmetiskt uttryck. Till exempel, multiplicera det med `1` för att returnera själva värdet.

Diagram för `weekday()`-funktionen med nummervärdet för den dag som visas i stället för dagens namn



Användning

Funktionen `weekday()` är användbar när du vill jämföra sammanställningar efter veckodagar. Om du exempelvis vill jämföra genomsnittlig försäljning av produkter efter veckodag.

Dessa dimensioner kan skapas i laddningsskriptet genom att använda funktionen för att skapa ett fält i en **Master Calendar**-tabell, eller skapas direkt i ett diagram som ett beräknat mått.

Relaterade ämnen

Ämnen	Interaktion
FirstWeekDay (page 257)	Definierar startdagen för varje vecka.

Argument

Argument	Beskrivning
date	Det datum eller den tidsmarkör som ska utvärderas.
first_week_day	Anger dagen då veckan startar. Om utelämnat används värdet för variabeln **FirstWeekDay**. *FirstWeekDay (page 257)*

Du kan använda följande värden för att ställa in vilken dag veckan börjar i argumentet `first_week_day`:

first_week_day values

Dag	värde
Måndag	0
Tisdag	1
Onsdag	2
Torsdag	3
Fredag	4
Lördag	5
Söndag	6

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Om inget annat sägs,är `FirstWeekDay` inställd till `0` i dessa exempel.

Exempel på funktioner

Exempel	Resultat
`weekday('10/12/1971')`	Returnerar ' Tue' och 1.
`weekday('10/12/1971' , 6)`	Returnerar 'Tue' och 2. I det här exemplet är söndag (`6`) den första dagen i veckan.

`SET FirstWeekDay=6;`
`...`
 	
`weekday('10/12/1971')`
	Returnerar 'Tue' och 2.

Exempel 1 – veckodagssträng

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner för 2022 som laddas in i en tabell som heter '`Transactions`'.
- Systemvariabeln `FirstWeekDay` som är inställd till 6 (söndag).
- `DayNames`-variabeln som är inställd till att använda standardnamnen för dagar.
- En föregående laddning som innehåller `weekday()`-funktionen, som är satt som '`week_day`'-fältet och returnerar den veckodag då transaktionerna ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
    Load
       *,
       WeekDay(date) as week_day
       ;
Load
*
Inline
[
id,date,amount
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
```

```
8193,01/06/2022,82.06
8194,01/07/2022,40.39
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- id
- date
- week_day

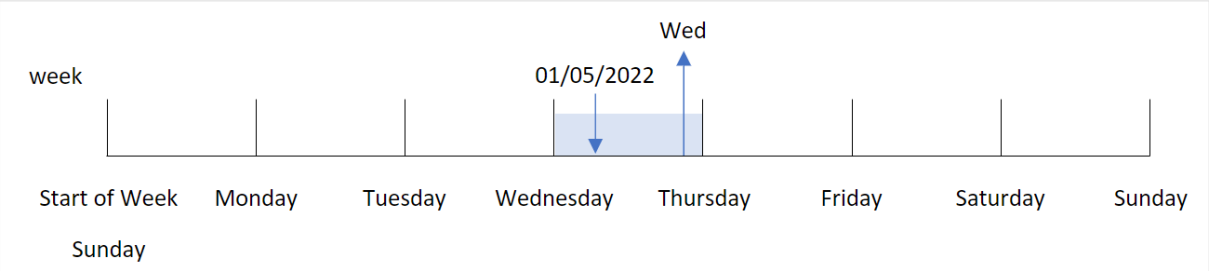
Resultattabell

Id	Datum	week_day
8188	01/01/2022	lör
8189	01/02/2022	sön
8190	01/03/2022	mån
8191	01/04/2022	tis
8192	01/05/2022	ons
8193	01/06/2022	tors
8194	01/07/2022	fre

'week_day'-fältet skapades i den föregående load-satsen genom att använda weekday()-funktionen och skicka datumfältet som funktionens argument.

weekday()-funktionen returnerar veckodagens strängvärde, det vill säga, returnerar namnet på veckodagen som ställs in med systemvariabeln DayNames.

Diagram för weekday()-funktionen som returnerar onsdag som veckodag för transaktion 8192



Transaktion 8192 ägde rum den 5 januari. Systemvariabeln FirstWeekDay anger söndag som den första dagen i veckan. Funktionstransaktionen weekday() ägde rum på en onsdag och returnerar detta värde, i den förkortade formen från systemvariabeln DayNames i week_day-fältet.

Värdena i week_day-fältet är högerjusterade i kolumnen eftersom det finns ett dualt tal och textresultat för fältet (onsdagen den 3:e). För att konvertera fältvärdet till dess numeriska motsvarighet kan fältet inneslutas i num()-funktionen. I transaktion 8192 exempelvis skulle onsdagsvärdet konverteras till talet 3.

Exempel 2 – first_week_day

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2022 som laddas i en tabell som heter 'Transactions'.
- Systemvariabeln FirstWeekDay som är inställd till 6 (söndag).
- DayNames-variabeln som är inställd till att använda standardnamnen för dagar.
- En föregående laddning som innehåller weekday()-funktionen, som är satt till week_day-fältet och returnerar den veckodag då transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
    Load
       *,
       WeekDay(date,1) as week_day
       ;
Load
*
Inline
[
id,date,amount
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.39
];
```

Resultat

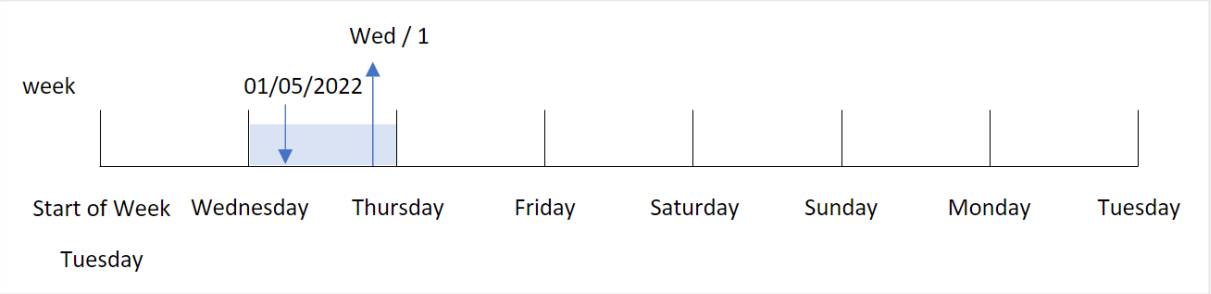
Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- id
- date
- week_day

Resultattabell

Id	Datum	week_day
8188	01/01/2022	lör
8189	01/02/2022	sön
8190	01/03/2022	mån
8191	01/04/2022	tis
8192	01/05/2022	ons
8193	01/06/2022	tors
8194	01/07/2022	fre

Diagram för `weekday()`-funktionen som visar att onsdag har det duala värdet 1



Eftersom `first_week_day`-argumentet är satt till `1` i `weekday()`-funktionen är tisdag den första dagen i veckan. Därför kommer alla transaktioner som äger rum på en tisdag ha ett dualt numeriskt värde på `0`.

Transaktion 8192 ägde rum den 5 januari. `weekday()`-funktionen identifierar att detta är en onsdag, och uttrycket skulle alltså returnera det duala numeriska värdet `1`.

Exempel 3 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2022 som laddas i en tabell som heter ‘`Transactions`’.
- Systemvariabeln `FirstWeekDay` som är inställd till `6` (söndag).
- `DayNames`-variabeln som är inställd till att använda standardnamnen för dagar.

I det här exemplet är dock datauppsättningen oförändrad och har lästs in i programmet. Beräkningen som identifierar veckodagsvärdet skapas som ett mått i ett diagram i applikationen.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
Load
*
Inline
[
id,date,amount
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.39
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`

För att beräkna veckodagsvärdet skapar du följande mått:

- `=weekday(date)`

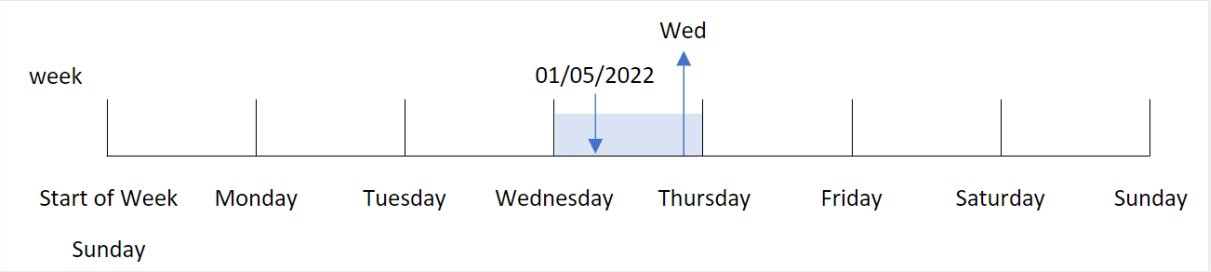
Resultattabell

Id	Datum	=weekday(date)
8188	01/01/2022	lör
8189	01/02/2022	sön
8190	01/03/2022	mån
8191	01/04/2022	tis
8192	01/05/2022	ons
8193	01/06/2022	tors
8194	01/07/2022	fre

`=weekday(date)`-fältet skapades i diagrammet genom att använda `weekday()`-funktionen och skicka datumfältet som funktionens argument.

`weekday()`-funktionen returnerar veckodagens strängvärde, det vill säga, returnerar namnet på veckodagen som ställs in med systemvariabeln `DayNames`.

Diagram för `weekday()`-funktionen som returnerar onsdag som veckodag för transaktion 8192



Transaktion 8192 ägde rum den 5 januari. Systemvariabeln `FirstWeekDay` anger söndag som den första dagen i veckan. Funktionstransaktionen `weekday()` ägde rum på en onsdag och returnerar detta värde, i den förkortade formen från systemvariabeln `DayNames` i `=weekday(date)`-fältet.

Exempel 4 – Scenario

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2022 som laddas i en tabell som heter 'Transactions'.
- Systemvariabeln `FirstWeekDay` som är inställd till `6` (söndag).
- `DayNames`-variabeln som är inställd till att använda standardnamnen för dagar.

Slutanvändaren vill ha ett diagram som visar den genomsnittliga försäljningen per veckodag för transaktionerna.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
LOAD
  RecNo() AS id,
  MakeDate(2022, 1, Ceil(Rand() * 31)) as date,
  Rand() * 1000 AS amount

Autogenerate(1000);
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `=weekday(date)`
- `=avg(amount)`

Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

weekday(date)	Avg(amount)
sön	$536.96
mån	$500.80
tis	$515.63
ons	$509.21
tors	$482.70
fre	$441.33
lör	$505.22

weekend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden på den sista dagen i den kalendervecka som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

Syntax:

```
WeekEnd(timestamp [, period_no [, first_week_day ]])
```

Returnerad datatyp: dual

`weekend()`-funktionen bestämmer vilken vecka datumet infaller. Den returnerar sedan en tidsstämpel, i datumformat, för den sista millisekunden av den veckan. Veckans första dag bestäms av miljövariabeln `FirstWeekDay`. Men den kan ersättas av `first_week_day`-argumentet i `weekend()`-funktionen.

Argument

Argument	Beskrivning
timestamp	Det datum eller den tidsmarkör som ska utvärderas.
period_no	**shift** är ett heltal, där värdet 0 anger den vecka som innehåller **date**. Negativa värden i skift anger föregående veckor och positiva värden anger efterföljande veckor.
first_week_day	Anger dagen då veckan startar. Om utelämnat används värdet för variabeln **FirstWeekDay**. De möjliga värdena för **first_week_day** är 0 för måndag, 1 för tisdag, 2 för onsdag, 3 för torsdag, 4 för fredag, 5 för lördag och 6 för söndag. Se *FirstWeekDay (page 257)* för mer information om systemvariabeln

Användning

`weekend()`-funktionen används vanligtvis som en del av ett uttryck när användaren vill att beräkningen ska använda det återstående antalet dagar i veckan för det angivna datumet. Det kan till exempel användas om en användare vill beräkna den totala ränta som ännu inte uppkommit under veckan.

Följande exempel antar:

```
SET FirstWeekDay=0;
```

Exempel	Resultat
`weekend('01/10/2013')`	Returnerar 01/12/2013 23:59:59.
`weekend('01/10/2013', -1)`	Returnerar 01/05/2013 23:59:59..
`weekend('01/10/2013', 0, 1)`	Returnerar 01/14/2013 23:59:59.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel:

Om du vill ha ISO-inställningar för veckor och veckonummer, ska du se till att ha följande i skriptet:

```
Set DateFormat   ='YYYY-MM-DD';
Set FirstWeekDay =0;    // Monday as first week day
Set BrokenWeeks  =0;    //(use unbroken weeks)
Set ReferenceDay =4;    // Jan 4th is always in week 1
```

Om du vill ha USA-inställningar, ska du se till att ha följande i skriptet:

```
Set DateFormat   ='M/D/YYYY';
Set FirstWeekDay =6;    // Sunday as first week day
Set BrokenWeeks  =1;    //(use broken weeks)
Set ReferenceDay =1;    // Jan 1st is always in week 1
```

Exemplen ovan resulterar i följande från `weekend()`-funktionen:

Exempel på veckoslutsfunktion

Datum	ISO-veckoslut	USA-veckoslut
Lör 2020 Dec 26	2020-12-27	12/26/2020
Sön 2020 Dec 27	2020-12-27	2021-02-01
Mån 2020 Dec 28	2021-01-03	2021-01-02
Tis 2020 Dec 29	2021-01-03	2021-01-02
Ons 2020 Dec 30	2021-01-03	2021-02-01
Tors 2020 Dec 31	2021-01-03	2021-02-01
Fre 2021 Jan 1	2021-01-03	2021-02-01
Lör 2021 Jan 2	2021-01-03	2021-01-02
Sön 2021 Jan 3	2021-01-03	2021-01-09
Mån 2021 Jan 4	2021-01-10	2021-01-09
Tis 2021 Jan 5	2021-01-10	2021-01-09

Veckosluten är på söndagar i ISO-kolumnen och på lördagar i USA-kolumnen.

Exempel 1 – Grundläggande exempel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2022 som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (MM/DD/YYYY) i systemvariabeln `DateFormat`.
- Ett fält `end_of_week` skapas som returnerar en tidsmarkör för slutet av den vecka då transaktionerna ägde rum.

Laddningsskript

```
SET FirstWeekDay=6;

Transactions:
    Load
        *,
        weekend(date) as end_of_week,
        timestamp(weekend(date)) as end_of_week_timestamp
        ;
```

```
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

Resultattabell

datum	slutet av veckan	tidsmarkör för slutet av veckan
1/7/2022	2022-01-08	2022-01-08 23:59:59
1/19/2022	2022-01-22	2022-01-22 23:59:59
2/5/2022	2022-02-05	2022-02-05 23:59:59
2/28/2022	03/05/2022	2022-03-05 23:59:59
3/16/2022	2022-03-19	2022-03-19 23:59:59
4/1/2022	2022-04-02	2022-04-02 23:59:59
5/7/2022	2022-05-07	2022-05-07 23:59:59
5/16/2022	2022-05-21	2022-05-21 23:59:59

datum	slutet av veckan	tidsmarkör för slutet av veckan
6/15/2022	2022-06-18	2022-06-18 23:59:59
6/26/2022	2022-07-02	2022-07-02 23:59:59
7/9/2022	2022-07-09	2022-07-09 23:59:59
7/22/2022	2022-07-23	2022-07-23 23:59:59
7/23/2022	2022-07-23	2022-07-23 23:59:59
7/27/2022	2022-07-30	2022-07-30 23:59:59
8/2/2022	2022-08-06	2022-08-06 23:59:59
8/8/2022	2022-08-13	2022-08-13 23:59:59
8/19/2022	2022-08-20	2022-08-20 23:59:59
9/26/2022	2022-10-01	2022-10-01 23:59:59
10/14/2022	2022-10-15	2022-10-15 23:59:59
10/29/2022	10/29/2022	2022-10-29 23:59:59

`end_of_week`-fältet skapades i den föregående load-satsen genom att använda `weekend()`-funktionen och skicka datumfältet som funktionens argument.

`weekend()`-funktionen identifierar vilken vecka datumvärdet infaller och returnerar en tidsstämpel för den sista millisekunden av den veckan.

Diagram för `weekend()`-funktionen, grundläggande exempel

Transaktion 8191 ägde rum 5 februari. Systemvariabeln `FirstWeekDay` sätter söndag som första dag i veckan. `weekend()`-funktionen identifierar att den första lördagen efter 5 februari – och därmed slutet av veckan – inföll 5 februari. Därför returnerar `end_of_week`-värdet för den transaktionen den sista millisekunden den dagen, vilket är 5 februari 23:59:59.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Ett fält `previous_week_end` skapas som returnerar tidsmarkören för början av veckan innan transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        weekend(date,-1) as previous_week_end,
        timestamp(weekend(date,-1)) as previous_week_end_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

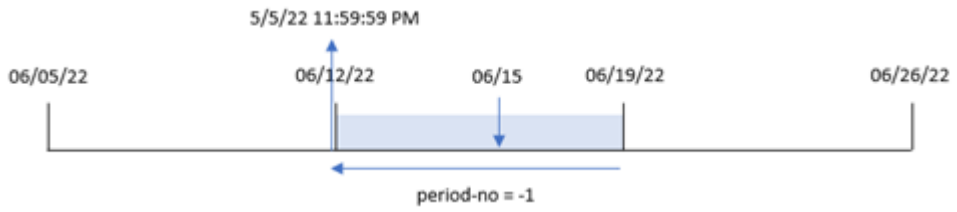
- `date`
- `previous_week_end`
- `previous_week_end_timestamp`

Resultattabell

datum	slutet av veckan	tidsmarkör för slutet av veckan
1/7/2022	01/01/2022	2022-01-01 23:59:59
1/19/2022	15/01/2022	2022-01-15 23:59:59
2/5/2022	29/01/2022	2022-01-29 23:59:59
2/28/2022	02/26/2022	2022-02-26 23:59:59
3/16/2022	03/12/2022	2022-03-12 23:59:59
4/1/2022	2022-03-26	2022-03-26 23:59:59
5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
5/16/2022	2022-05-14	2022-05-14 23:59:59
6/15/2022	2022-06-11	2022-06-11 23:59:59
6/26/2022	2022-06-25	2022-06-25 23:59:59
7/9/2022	2022-07-02	2022-07-02 23:59:59
7/22/2022	2022-07-16	2022-07-16 23:59:59
7/23/2022	2022-07-16	2022-07-16 23:59:59
7/27/2022	2022-07-23	2022-07-23 23:59:59
8/2/2022	2022-07-30	2022-07-30 23:59:59
8/8/2022	2022-08-06	2022-08-06 23:59:59
8/19/2022	2022-08-13	2022-08-13 23:59:59
9/26/2022	24/09/2022	2022-09-24 23:59:59
10/14/2022	2022-10-08	2022-10-08 23:59:59
10/29/2022	10/22/2022	2022-10-22 23:59:59

Eftersom ett `period_no` på -1 användes som förskjutningsargument i `weekend()`-funktionen i det här fallet identifierar funktionen först den vecka då transaktionerna äger rum. Den tittar sedan en vecka innan och identifierar den sista millisekunden i den veckan.

Diagram med exempel på `weekend()`-funktionen, period_no



Transaktion 8196 ägde rum den 15 juni. `weekend()`-funktionen identifierar att veckan börjar 12 juni. Därför slutar den föregående veckan 11 juni 11:59:59; detta är det värde som returneras för `previous_week_end`-fältet.

Exempel 3 – first_week_day

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet. Men i det här exemplet behöver vi sätta tisdag som första dag i arbetsveckan.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        weekend(date,0,1) as end_of_week,
        timestamp(weekend(date,0,1)) as end_of_week_timestamp,
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

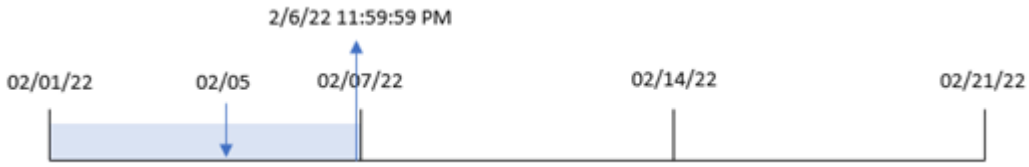
- `date`
- `end_of_week`
- `end_of_week_timestamp`

Resultattabell

datum	slutet av veckan	tidsmarkör för slutet av veckan
1/7/2022	01/10/2022	2022-01-10 23:59:59
1/19/2022	01/24/2022	2022-01-24 23:59:59
2/5/2022	02/07/2022	2022-02-07 23:59:59
2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
3/16/2022	03/21/2022	2022-03-21 23:59:59
4/1/2022	04/04/2022	2022-04-04 23:59:59
5/7/2022	05/09/2022	2022-05-09 23:59:59
5/16/2022	05/16/2022	2022-05-16 23:59:59
6/15/2022	06/20/2022	2022-06-20 23:59:59
6/26/2022	06/27/2022	2022-06-27 23:59:59
7/9/2022	07/11/2022	2022-07-11 23:59:59
7/22/2022	2022-07-25	2022-07-25 23:59:59
7/23/2022	2022-07-25	2022-07-25 23:59:59
7/27/2022	08/01/2022	2022-08-01 23:59:59
8/2/2022	2022-08-08	2022-08-08 23:59:59
8/8/2022	2022-08-08	2022-08-08 23:59:59
8/19/2022	08/22/2022	2022-08-22 23:59:59
9/26/2022	09/26/2022	2022-09-26 23:59:59
10/14/2022	10/17/2022	2022-10-17 23:59:59
10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

Eftersom `first_week_date`-argumentet på 1 används i `weekend()`-funktionen i det här fallet, sätts tisdag som första dag i veckan.

Diagram med exempel på `weekend()`-funktionen, first_week_day



Transaktion 8191 ägde rum 5 februari. `weekend()`-funktionen identifierar att den första måndagen efter det här datumet – och därför slutet av veckan och det returnerade värdet – var 6 februari 23:59:59.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet. I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som returnerar en tidsmarkör för slutet av veckan då transaktionerna ägde rum skapas som ett mått i ett diagramobjekt för programmet.

Laddningsskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
```

```
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

För att beräkna veckostart för den vecka då en transaktion äger rum lägger du till följande mått:

- `=weekend(date)`
- `=timestamp(weekend(date))`

Resultattabell

datum	=weekend(date)	=tidsmarkör(helg(datum))
1/7/2022	2022-01-08	2022-01-08 23:59:59
1/19/2022	2022-01-22	2022-01-22 23:59:59
2/5/2022	2022-02-05	2022-02-05 23:59:59
2/28/2022	03/05/2022	2022-03-05 23:59:59
3/16/2022	2022-03-19	2022-03-19 23:59:59
4/1/2022	2022-04-02	2022-04-02 23:59:59
5/7/2022	2022-05-07	2022-05-07 23:59:59
5/16/2022	2022-05-21	2022-05-21 23:59:59
6/15/2022	2022-06-18	2022-06-18 23:59:59
6/26/2022	2022-07-02	2022-07-02 23:59:59
7/9/2022	2022-07-09	2022-07-09 23:59:59
7/22/2022	2022-07-23	2022-07-23 23:59:59
7/23/2022	2022-07-23	2022-07-23 23:59:59
7/27/2022	2022-07-30	2022-07-30 23:59:59
8/2/2022	2022-08-06	2022-08-06 23:59:59
8/8/2022	2022-08-13	2022-08-13 23:59:59
8/19/2022	2022-08-20	2022-08-20 23:59:59
9/26/2022	2022-10-01	2022-10-01 23:59:59
10/14/2022	2022-10-15	2022-10-15 23:59:59
10/29/2022	10/29/2022	2022-10-29 23:59:59

`end_of_week`-måttet skapades i ett diagramobjektet genom att använda `weekend()`-funktionen och skicka datumfältet som funktionens argument. `weekend()`-funktionen identifierar vilken vecka datumvärdet infaller och returnerar en tidsstämpel för den sista millisekunden av den veckan.

Diagram med exempel på `weekend()`-funktionen, diagramobjekt

Transaktion 8191 ägde rum 5 februari. Systemvariabeln `FirstWeekDay` sätter söndag som första dag i veckan. `weekend()`-funktionen identifierar att den första lördagen efter 5 februari – och därmed slutet av veckan – inföll 5 februari. Därför returnerar `end_of_week`-värdet för den transaktionen den sista millisekunden den dagen, vilket är 5 februari 23:59:59.

Exempel 5 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning läses in i en tabell som heter `Employee_Expenses`.
- Data bestående av anställnings-ID, anställdas namn och de genomsnittliga dagliga kostnadsanspråken för varje anställd.

Slutanvändaren vill ha ett diagramobjekt som visar, efter anställnings-id och anställds namn, de beräknade kostnadsanspråken som fortfarande återstår för resten av veckan.

Laddningsskript

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:
   - `employee_id`
   - `employee_name`
2. Skapa sedan ett mått för att beräkna den ackumulerade räntan:
   `=(weekend(today(1))-today(1))*avg_daily_claim`
3. Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

employee_id	employee_name	=(weekend(today(1))-today(1))*avg_daily_claim
182	Mark	$90.00
183	Deryck	$75.00
184	Dexter	$75.00
185	Sydney	$162.00
186	Agatha	$108.00

`weekend()`-funktionen returnerar slutdatumet för den aktuella veckan genom att använda dagens datum som enda argument. Sedan returnerar uttrycket antalet dagar som återstår av denna vecka genom att subtrahera dagens datum från årets slutdatum.

Detta värde multipliceras sedan med det genomsnittliga dagliga kostnadsanspråket från varje anställd för att beräkna det uppskattade värdet av anspråk som varje anställd förväntas göra under den återstående veckan.

weekname

Denna funktion returnerar ett värde som visar år och veckonummer med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden på den första dagen i veckan som innehåller **date**.

```
WeekName(date[, period_no [, first_week_day [, broken_weeks [, reference_
day]]]])
```

`weekname()`-funktionen bestämmer i vilken vecka datumet infaller och returnerar veckans veckonummer och år. Veckans första dag bestäms av systemvariabeln `FirstWeekDay`variabel. Du kan även ändra den första veckodagen genom att använda argumentet `first_week_day` i `weekname()`-funktionen.

I Qlik Sense hämtas de regionala inställningarna när applikationen skapas, och de motsvarande inställningarna lagras i skriptet som miljövariabler.

En nordamerikansk applikationsutvecklare får ofta `Set BrokenWeeks=1;` i skriptet, vilket motsvarar brutna veckor. En europeisk applikationsutvecklare får ofta `Set BrokenWeeks=0;` i skriptet, vilket motsvarar obrutna veckor.

Om ditt program använder brutna veckor, börjar veckoräkningen den 1 januari och slutar dagen före systemvariabeln `FirstWeekDay` oavsett hur många dagar som har förlöpt.

Om din ansökan använder obrutna veckor kan vecka 1 börja föregående år eller under de första dagarna i januari. Detta beror på hur du använder systemvariablerna `ReferenceDay` och `FirstWeekDay`.

Exempel på veckonamnsfunktion

Datum	ISO-veckonamn	USA-veckonamn
Lör 2020 Dec 26	2020/52	2020/52
Sön 2020 Dec 27	2020/52	2020/53
Mån 2020 Dec 28	2020/53	2020/53
Tis 2020 Dec 29	2020/53	2020/53
Ons 2020 Dec 30	2020/53	2020/53
Tors 2020 Dec 31	2020/53	2020/53
Fre 2021 Jan 1	2020/53	2021/01
Lör 2021 Jan 2	2020/53	2021/01
Sön 2021 Jan 3	2020/53	2021/02
Mån 2021 Jan 4	2021/01	2021/02
Tis 2021 Jan 5	2021/01	2021/02

Användning

Funktionen `weekname()` är användbar när du vill jämföra sammanställningar efter veckor.

Till exempel om du vill se den totala försäljningen av produkter per vecka. För att upprätthålla samstämmighet med `BrokenWeeks`-miljövariabeln i applikationen använder du `weekname()` i stället för `lunarweekname()`. Om programmet använder obrutna veckor kan vecka 1 innehålla datum från december föregående år eller exkludera datum i januari innevarande år. Om programmet använder brutna veckor kan vecka 1 innehålla mindre än sju dagar.

Returnerad datatyp: Dual

Argument

Argument	Beskrivning
timestamp	Det datum eller den tidsmarkör som ska utvärderas.
period_no	**shift** är ett heltal, där värdet 0 anger den vecka som innehåller **date**. Negativa värden i skift anger föregående veckor och positiva värden anger efterföljande veckor.

Argument	Beskrivning
first_week_day	Anger dagen då veckan startar. Om utelämnat används värdet för variabeln **FirstWeekDay**. De möjliga värdena **first_week_day** är 0 för måndag, 1 för tisdag, 2 för onsdag, 3 för torsdag, 4 för fredag, 5 för lördag och 6 för söndag. Se *FirstWeekDay (page 257)* för mer information om systemvariabeln.
broken_weeks	Om du inte anger **broken_weeks** används värdet för variabeln **BrokenWeeks** till att definiera om veckor är brutna eller inte.
reference_day	Om du inte anger **reference_day**, används värdet för variabeln **ReferenceDay** till att definiera vilken dag i januari som ska ställas in som referensdag för att definiera vecka 1. Som standard använder Qlik Sense-funktioner 4 som referensdag. Det betyder att vecka 1 måste innehålla 4 januari, eller med andra ord, vecka 1 måste alltid bestå av minst 4 dagar i januari.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exemplen nedan antar:

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

Exempel på funktioner

Exempel	Resultat
weekname('01/12/2013')	Returnerar 2013/02.
weekname('01/12/2013', -1)	Returnerar 2013/01.
weekname('01/12/2013', 0, 1)	Returnerar 2013/02.

Exempel 1 – Datum utan ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för sista veckan år 2021 och de två första veckorna år 2022 läses in i en tabell som heter ”`Transactions`”.
- Systemvariabeln `DateFormat` som är inställd på `MM/DD/YYYY`-formatet.
- Systemvariabeln `BrokenWeeks` som är inställd på `1`.
- Systemvariabeln `FirstWeekDay` som är inställd på `6`.
- En föregående laddning som innehåller följande:
  - `weekday()`-funktionen som är inställd som fältet ‘`week_number`’, som returnerar år och veckonummer när transaktionerna ägde rum.
  - `weekname()`-funktionen som är inställd som fältet som heter `week_day`, för att visa veckodagsvärdet för varje transaktionsdatum.

Laddningsskript

```
SET BrokenWeeks=1;
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;

Transactions:
   Load
     *,
     WeekDay(date) as week_day,
     Weekname(date) as week_number
     ;
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
```

```
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `week_day`
- `week_number`

Resultattabell

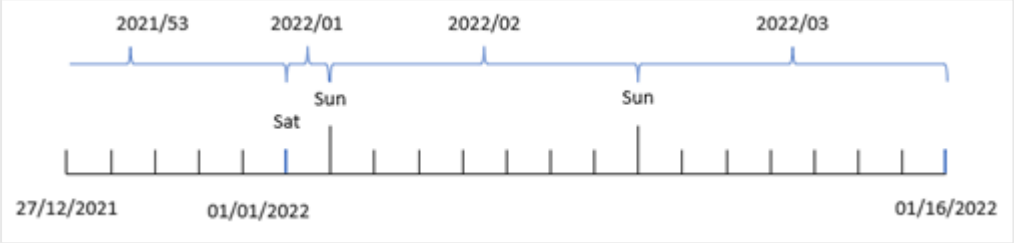
Id	Datum	week_day	veckonummer
8183	12/27/2021	mån	2021/53
8184	28/12/2021	tis	2021/53
8185	29/12/2021	ons	2021/53
8186	30/12/2021	tors	2021/53
8187	12/31/2021	fre	2021/53
8188	01/01/2022	lör	2022/01
8189	01/02/2022	sön	2022/02
8190	01/03/2022	mån	2022/02
8191	01/04/2022	tis	2022/02
8192	01/05/2022	ons	2022/02
8193	01/06/2022	tors	2022/02
8194	01/07/2022	fre	2022/02
8195	2022-01-08	lör	2022/02
8196	09/01/2022	sön	2022/03
8197	10/01/2022	mån	2022/03
8198	11/01/2022	tis	2022/03
8199	12/01/2022	ons	2022/03
8200	13/01/2022	tors	2022/03
8201	14/01/2022	fre	2022/03

'week_number'-fältet skapas i den föregående load-satsen genom att använda weekname()-funktionen och skicka datumfältet som funktionens argument.

weekname()-funktionen identifierar initialt vilken vecka datumvärdet infaller och returnerar veckotalet och året då transaktionen äger rum.

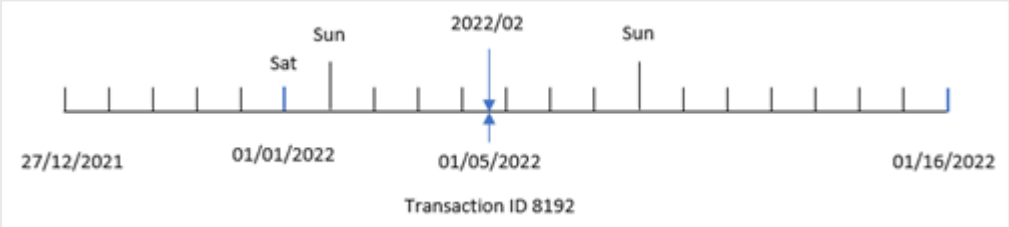
Systemvariabeln FirstWeekDay anger söndag som den första dagen i veckan. Systemvariabeln BrokenWeeks ställer in programmet för att använda brutna veckor, vilket innebär att vecka 1 börjar den 1 januari.

Diagram för weekname()-funktionen med standardvariablerna.



Vecka 1 börjar den 1 januari, vilket är en lördag, och därför returnerar transaktioner som sker på detta datum värdet 2022/01 (år och veckonummer).

Diagram över weekname() funktion som identifierar veckonummer för transaktion 8192.



Eftersom programmet använder brutna veckor och den första veckodagen är söndag, returnerar transaktioner från 2 till 8 januari värdet 2022/02 (veckonummer 2 för år 2022.) Ett exempel på detta skulle vara transaktion 8192 som ägde rum den 5 januari och returnerar värdet 2022/02 för week_number-fältet.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Samma datauppsättning och scenario som det första exemplet används.

Men i det här exemplet är uppgiften att skapa ett fält, previous_week_number, som returnerar år och veckonummer innan transaktionerna ägde rum.

Öppna Skriptredigeraren och lägg till följande laddningsskript till ett nytt delavsnitt.

Laddningsskript

```
SET BrokenWeeks=1;
SET FirstWeekDay=6;

Transactions:
   Load
     *,
     weekname(date,-1) as previous_week_number
     ;
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `week_day`
- `week_number`

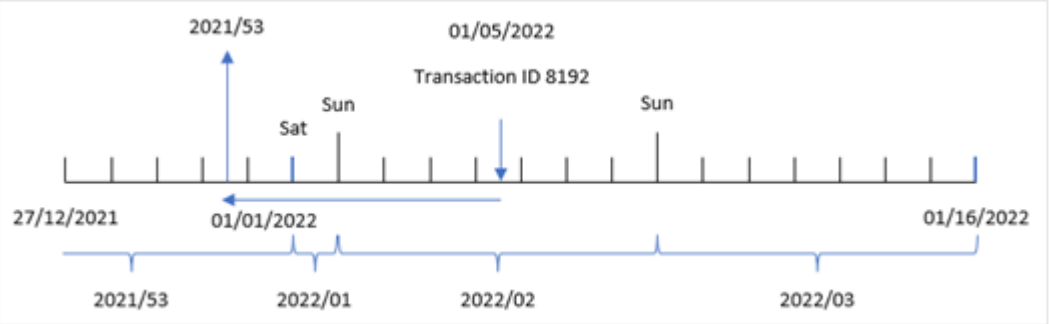
Resultattabell

Id	Datum	week_day	veckonummer
8183	12/27/2021	mån	2021/52
8184	28/12/2021	tis	2021/52

Id	Datum	week_day	veckonummer
8185	29/12/2021	ons	2021/52
8186	30/12/2021	tors	2021/52
8187	12/31/2021	fre	2021/52
8188	01/01/2022	lör	2021/52
8189	01/02/2022	sön	2021/53
8190	01/03/2022	mån	2021/53
8191	01/04/2022	tis	2021/53
8192	01/05/2022	ons	2021/53
8193	01/06/2022	tors	2021/53
8194	01/07/2022	fre	2021/53
8195	2022-01-08	lör	2022/01
8196	09/01/2022	sön	2022/02
8197	10/01/2022	mån	2022/02
8198	11/01/2022	tis	2022/02
8199	12/01/2022	ons	2022/02
8200	13/01/2022	tors	2022/02
8201	14/01/2022	fre	2022/02

Eftersom en `period_no` på `-1` användes som förskjutningsargument i `weekname()`-funktionen så identifierar funktionen först veckan då transaktionerna äger rum. Den tittar sedan en vecka innan och identifierar den första millisekunden i den veckan.

Diagram för `weekname()`-funktionen med en `period_no`-förskjutning på -1.



Transaktion 8192 ägde rum den 5 januari 2022. `weekname()`-funktionen tittar en vecka före den 30 december 2021 och returnerar veckonummer och år för det datumet - `2021/53`.

Exempel 3 – first_week_day

Laddningsskript och resultat

Översikt

Samma datauppsättning och scenario som det första exemplet används.

Men i det här exemplet är företagets policy att arbetsveckan ska börja på tisdag.

Öppna Skriptredigeraren och lägg till följande laddningsskript till ett nytt delavsnitt.

Laddningsskript

```
SET BrokenWeeks=1;
SET DateFormat='MM/DD/YYYY';

Transactions:
   Load
     *,
     weekday(date) as week_day,
     weekname(date,0,1) as week_number
     ;
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Resultat

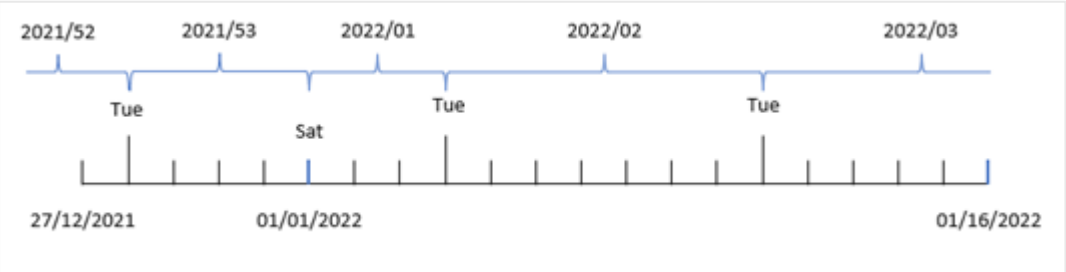
Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- id
- date
- week_day
- week_number

Resultattabell

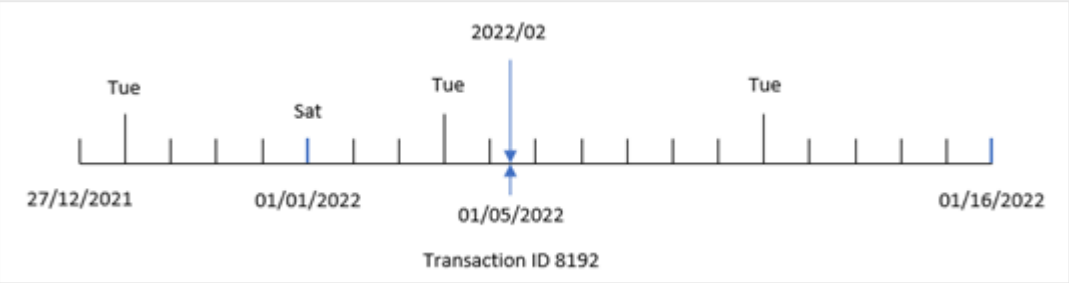
Id	Datum	week_day	veckonummer
8183	12/27/2021	mån	2021/52
8184	28/12/2021	tis	2021/53
8185	29/12/2021	ons	2021/53
8186	30/12/2021	tors	2021/53
8187	12/31/2021	fre	2021/53
8188	01/01/2022	lör	2022/01
8189	01/02/2022	sön	2022/01
8190	01/03/2022	mån	2022/01
8191	01/04/2022	tis	2022/02
8192	01/05/2022	ons	2022/02
8193	01/06/2022	tors	2022/02
8194	01/07/2022	fre	2022/02
8195	2022-01-08	lör	2022/02
8196	09/01/2022	sön	2022/02
8197	10/01/2022	mån	2022/02
8198	11/01/2022	tis	2022/03
8199	12/01/2022	ons	2022/03
8200	13/01/2022	tors	2022/03
8201	14/01/2022	fre	2022/03

Diagram över `weekname()`-funktionen med tisdag som första dag i veckan.



Eftersom `first_week_date`-argumentet för `1` används i `weekname()`-funktionen, använder det tisdag som den första dagen i veckan. Funktionen bestämmer därför att vecka 53 2021 börjar tisdagen den 28 december. Då programmet använder brutna veckor, börjar vecka 1 den 1 januari 2022 och slutar den sista millisekunden för måndagen den 3 januari 2022.

Diagram som visar veckonummer för transaktion 8192 med tisdag som första veckodag.



Transaktion 8192 ägde rum den 5 januari 2022. Således, om en `first_week_day`-parameter används för tisdag returnerar `weekname()`-funktionen `2022/02`-värdet för `week_number`-fältet.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är dock datauppsättningen oförändrad och har lästs in i programmet. Beräkningen som returnerar ett årsnummer för veckan då transaktionerna ägde rum skapas som ett mått i ett diagramobjekt för programmet.

Laddningsskript

```
SET BrokenWeeks=1;
Transactions:
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
```

```
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `=week_day (date)`

För att beräkna veckostart för den vecka då en transaktion äger rum skapar du följande mått:

```
=weekname(date)
```

Resultattabell

Id	Datum	=weekday(date)	=weekname(date)
8183	12/27/2021	mån	2021/53
8184	28/12/2021	tis	2021/53
8185	29/12/2021	ons	2021/53
8186	30/12/2021	tors	2021/53
8187	12/31/2021	fre	2021/53
8188	01/01/2022	lör	2022/01
8189	01/02/2022	sön	2022/02
8190	01/03/2022	mån	2022/02
8191	01/04/2022	tis	2022/02
8192	01/05/2022	ons	2022/02
8193	01/06/2022	tors	2022/02
8194	01/07/2022	fre	2022/02
8195	2022-01-08	lör	2022/02
8196	09/01/2022	sön	2022/03
8197	10/01/2022	mån	2022/03
8198	11/01/2022	tis	2022/03
8199	12/01/2022	ons	2022/03

Id	Datum	=weekday(date)	=weekname(date)
8200	13/01/2022	tors	2022/03
8201	14/01/2022	fre	2022/03

`week_number`-fältet skapades som ett mått i diagramobjektet genom att använda `weekname()`-funktionen och skicka datumfältet som funktionens argument.

`weekname()`-funktionen identifierar initialt vilken vecka datumvärdet infaller och returnerar veckotalet och året då transaktionen äger rum.

Systemvariabeln `FirstWeekDay` anger söndag som den första dagen i veckan. Systemvariabeln `BrokenWeeks` ställer in programmet för att använda brutna veckor, vilket innebär att vecka 1 börjar den 1 januari.

Diagram som visar veckonummer med söndag som första veckodag.

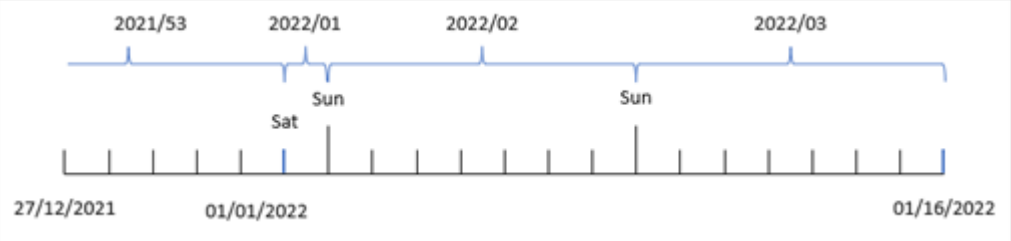
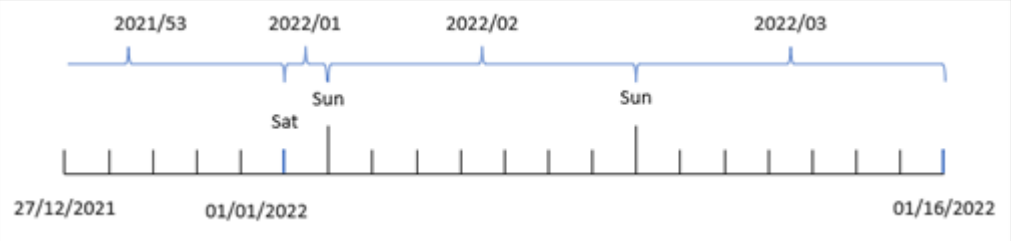


Diagram som visar att transaktion 8192 ägde rum i vecka 2.



Eftersom programmet använder brutna veckor och den första veckodagen är söndag, returnerar transaktioner från 2 till 8 januari värdet 2022/02, veckonummer 2 för år 2022. Observera att transaktion 8192 ägde rum den 5 januari och returnerar värdet 2022/02 för `week_number`-fältet.

Exempel 5 – Scenario

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för sista veckan år 2019 och de två första veckorna år 2020 läses in i en tabell som heter `Transactions`.
- Systemvariabeln `BrokenWeeks` som är inställd på `0`.
- Systemvariabeln `ReferenceDay` som är inställd på `2`.
- Systemvariabeln `DateFormat` som är inställd på `MM/DD/YYYY`-formatet.

Laddningsskript

```
SET BrokenWeeks=0;
SET ReferenceDay=2;
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8183,12/27/2019,58.27
8184,12/28/2019,67.42
8185,12/29/2019,23.80
8186,12/30/2019,82.06
8187,12/31/2019,40.56
8188,01/01/2020,37.23
8189,01/02/2020,17.17
8190,01/03/2020,88.27
8191,01/04/2020,57.42
8192,01/05/2020,53.80
8193,01/06/2020,82.06
8194,01/07/2020,40.56
8195,01/08/2020,53.67
8196,01/09/2020,26.63
8197,01/10/2020,72.48
8198,01/11/2020,18.37
8199,01/12/2020,45.26
8200,01/13/2020,58.23
8201,01/14/2020,18.52
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell.

Skapa en beräknad dimension med följande uttryck:

```
=weekname(date)
```

För att beräkna total försäljning skapar du följande aggregeringsmått:

```
=sum(amount)
```

Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

weekname(date)	=sum(amount)
2019/52	$125.69
2020/01	$346.51
2020/02	$347.57
2020/03	$122.01

För att visa resultaten av att använda funktionen weekname() i det här scenariot lägger du till följande fält som en dimension:

`date`

Resultattabell med datumfält

weekname(date)	Datum	=sum(amount)
2019/52	12/27/2019	$58.27
2019/52	12/28/2019	$67.42
2020/01	12/29/2019	$23.80
2020/01	12/30/2019	$82.06
2020/01	12/31/2019	$40.56
2020/01	01/01/2020	$37.23
2020/01	01/02/2020	$17.17
2020/01	01/03/2020	$88.27
2020/01	01/04/2020	$57.42
2020/02	01/05/2020	$53.80
2020/02	01/06/2020	$82.06
2020/02	01/07/2020	$40.56
2020/02	01/08/2020	$53.67
2020/02	01/09/2020	$26.63
2020/02	01/10/2020	$72.48
2020/02	01/11/2020	$18.37
2020/03	01/12/2020	$45.26
2020/03	01/13/2020	$58.23
2020/03	01/14/2020	$18.52

Eftersom programmet använder obrutna veckor och vecka 1 kräver minst två dagar i januari på grund av systemvariabeln `ReferenceDay`, inkluderar vecka 1 av 2020 transaktioner från 29 december 2019.

weekstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden på den första dagen (måndag) i den kalendervecka som innehåller **date**. Det förvalda utdataformatet är det **DateFormat** som har definierats i skriptet.

Syntax:

```
WeekStart(timestamp [, period_no [, first_week_day ]])
```

Returnerad datatyp: dual

`weekstart()`-funktionen bestämmer vilken vecka datumet infaller. Den returnerar sedan en tidsstämpel, i datumformat, för den första millisekunden av den veckan. Veckans första dag bestäms av miljövariabeln `FirstWeekDay`. Men den kan ersättas av `first_week_day`-argumentet i `weekstart()`-funktionen.

Argument

Argument	Beskrivning
timestamp	Det datum eller den tidsmarkör som ska utvärderas.
period_no	**shift** är ett heltal, där värdet 0 anger den vecka som innehåller **date**. Negativa värden i skift anger föregående veckor och positiva värden anger efterföljande veckor.
first_week_day	Anger dagen då veckan startar. Om utelämnat används värdet för variabeln **FirstWeekDay**. De möjliga värdena **first_week_day** är 0 för måndag, 1 för tisdag, 2 för onsdag, 3 för torsdag, 4 för fredag, 5 för lördag och 6 för söndag. Se *FirstWeekDay (page 257)* för mer information om systemvariabeln.

Användning

`weekstart()`-funktionen används vanligtvis som en del av ett uttryck när användaren vill att beräkningen ska använda den del av veckan som har förflutit hittills. Det kan exempelvis användas om en användare vill beräkna de totala lönerna som de anställda har tjänat hittills under veckan.

Följande exempel antar:

```
SET FirstWeekDay=0;
```

Exempel på funktioner

Exempel	Resultat
`weekstart('01/12/2013')`	Returnerar 01/07/2013.
`weekstart('01/12/2013', -1 )`	Returnerar 11/31/2012.
`weekstart('01/12/2013', 0, 1)`	Returnerar 01/08/2013.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel:

Om du vill ha ISO-inställningar för veckor och veckonummer, ska du se till att ha följande i skriptet:

```
Set DateFormat   ='YYYY-MM-DD';
Set FirstWeekDay =0;    // Monday as first week day
Set BrokenWeeks  =0;    //(use unbroken weeks)
Set ReferenceDay =4;    // Jan 4th is always in week 1
```

Om du vill ha USA-inställningar, ska du se till att ha följande i skriptet:

```
Set DateFormat   ='M/D/YYYY';
Set FirstWeekDay =6;    // Sunday as first week day
Set BrokenWeeks  =1;    //(use broken weeks)
Set ReferenceDay =1;    // Jan 1st is always in week 1
```

Exemplen ovan resulterar i följande från `weekstart()`-funktionen:

Exempel på veckostartsfunktion

Datum	ISO-veckostart	USA-veckostart
Lör 2020 Dec 26	2020-12-21	12/20/2020
Sön 2020 Dec 27	2020-12-21	12/27/2020
Mån 2020 Dec 28	2020-12-28	12/27/2020
Tis 2020 Dec 29	2020-12-28	12/27/2020
Ons 2020 Dec 30	2020-12-28	12/27/2020
Tors 2020 Dec 31	2020-12-28	12/27/2020
Fre 2021 Jan 1	2020-12-28	12/27/2020
Lör 2021 Jan 2	2020-12-28	12/27/2020
Sön 2021 Jan 3	2020-12-28	1/3/2021
Mån 2021 Jan 4	2021-01-04	1/3/2021
Tis 2021 Jan 5	2021-01-04	1/3/2021

Veckan börjar på måndagar i ISO-kolumnen och på söndagar i USA-kolumnen.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för 2022 som läses in i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (MM/DD/YYYY) i systemvariabeln `DateFormat`.
- Ett fält `start_of_week` skapas som returnerar en tidsmarkör för början av den vecka då transaktionerna ägde rum.

Laddningsskript

```
SET FirstWeekDay=6;

Transactions:
    Load
        *,
        weekstart(date) as start_of_week,
        timestamp(weekstart(date)) as start_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
```

```
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

Resultattabell

datum	start_of_week	start_of_week_timestamp
1/7/2022	01/02/2022	1/2/2022 12:00:00 AM
1/19/2022	01/16/2022	1/16/2022 12:00:00 AM
2/5/2022	01/30/2022	1/30/2022 12:00:00 AM
2/28/2022	02/27/2022	2/27/2022 12:00:00 AM
3/16/2022	03/13/2022	3/13/2022 12:00:00 AM
4/1/2022	03/27/2022	3/27/2022 12:00:00 AM
5/7/2022	2022-05-01	5/1/2022 12:00:00 AM
5/16/2022	05/15/2022	5/15/2022 12:00:00 AM
6/15/2022	06/12/2022	6/12/2022 12:00:00 AM
6/26/2022	06/26/2022	6/26/2022 12:00:00 AM
7/9/2022	07/03/2022	7/3/2022 12:00:00 AM
7/22/2022	07/17/2022	7/17/2022 12:00:00 AM
7/23/2022	07/17/2022	7/17/2022 12:00:00 AM
7/27/2022	07/24/2022	7/24/2022 12:00:00 AM
8/2/2022	07/31/2022	7/31/2022 12:00:00 AM
8/8/2022	08/07/2022	8/1/2022 12:00:00 AM
8/19/2022	08/14/2022	8/14/2022 12:00:00 AM
9/26/2022	09/25/2022	9/25/2022 12:00:00 AM
10/14/2022	10/09/2022	10/9/2022 12:00:00 AM
10/29/2022	10/23/2022	10/23/2022 12:00:00 AM

`start_of_week`-fältet skapades i den föregående load-satsen genom att använda `weekstart()`-funktionen och skicka datumfältet som funktionens argument.

`weekstart()`-funktionen identifierar initialt vilken vecka datumvärdet infaller och returnerar en tidsstämpel för den första millisekunden av den veckan.

Diagram för `weekstart()`-funktionen, exempel utan ytterligare argument

Transaktion 8191 ägde rum 5 februari. Systemvariabeln `FirstWeekDay` sätter söndag som första dag i veckan. `weekstart()`-funktionen identifierar att den första söndagen före 5 februari – och därmed början av veckan – inföll 30 januari. Därför returnerar `start_of_week`-värdet för den transaktionen den första millisekunden den dagen, vilket är 30 januari 0:00:00.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Ett fält `previous_week_start` skapas som returnerar tidsmarkören för början av kvartalet innan transaktionen ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        weekstart(date,-1) as previous_week_start,
        timestamp(weekstart(date,-1)) as previous_week_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
```

```
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `previous_week_start`
- `previous_week_start_timestamp`

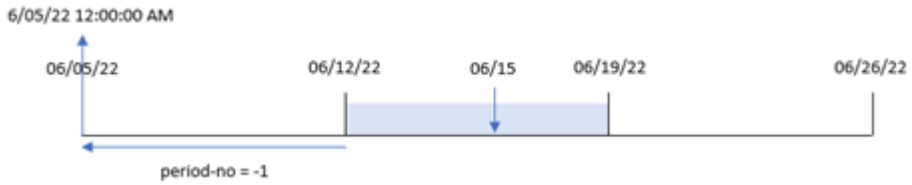
Resultattabell

datum	previous_week_start	previous_week_start_timestamp
1/7/2022	12/26/2021	12/26/2021 12:00:00 AM
1/19/2022	01/09/2022	1/9/2022 12:00:00 AM
2/5/2022	01/23/2022	1/23/2022 12:00:00 AM
2/28/2022	02/20/2022	2/20/2022 12:00:00 AM
3/16/2022	03/06/2022	3/6/2022 12:00:00 AM
4/1/2022	03/20/2022	3/20/2022 12:00:00 AM
5/7/2022	04/24/2022	4/24/2022 12:00:00 AM
5/16/2022	05/08/2022	5/8/2022 12:00:00 AM
6/15/2022	06/05/2022	6/5/2022 12:00:00 AM
6/26/2022	06/19/2022	6/19/2022 12:00:00 AM
7/9/2022	06/26/2022	6/26/2022 12:00:00 AM
7/22/2022	07/10/2022	7/10/2022 12:00:00 AM
7/23/2022	07/10/2022	7/10/2022 12:00:00 AM
7/27/2022	07/17/2022	7/17/2022 12:00:00 AM
8/2/2022	07/24/2022	7/24/2022 12:00:00 AM

datum	previous_week_start	previous_week_start_timestamp
8/8/2022	07/31/2022	7/31/2022 12:00:00 AM
8/19/2022	08/07/2022	8/1/2022 12:00:00 AM
9/26/2022	09/18/2022	9/18/2022 12:00:00 AM
10/14/2022	10/02/2022	10/2/2022 12:00:00 AM
10/29/2022	10/16/2022	10/16/2022 12:00:00 AM

I det här fallet, eftersom ett `period_no` på -1 användes som förskjutningsargument i `weekstart()`-funktionen, identifierar funktionen först den vecka då transaktionerna äger rum. Den tittar sedan en vecka innan och identifierar den första millisekunden i den veckan.

Diagram med exempel på `weekstart()`-funktionen, period_no



Transaktion 8196 ägde rum den 15 juni. `weekstart()`-funktionen identifierar att veckan börjar 12 juni. Därför började den föregående veckan 5 juni 0:00:00; detta är det värde som returneras för `previous_week_start`-fältet.

Exempel 3 – first_week_day

Laddningsskript och resultat

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan till en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet. Men i det här exemplet behöver vi sätta tisdag som första dag i arbetsveckan.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        weekstart(date,0,1) as start_of_week,
        timestamp(weekstart(date,0,1)) as start_of_week_timestamp
        ;
Load
*
```

```
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

Resultattabell

datum	start_of_week	start_of_week_timestamp
1/7/2022	01/04/2022	1/4/2022 12:00:00 AM
1/19/2022	01/18/2022	1/18/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/22/2022	2/22/2022 12:00:00 AM
3/16/2022	03/15/2022	3/15/2022 12:00:00 AM
4/1/2022	03/29/2022	3/29/2022 12:00:00 AM
5/7/2022	05/03/2022	5/3/2022 12:00:00 AM
5/16/2022	05/10/2022	5/10/2022 12:00:00 AM
6/15/2022	06/14/2022	6/14/2022 12:00:00 AM

datum	start_of_week	start_of_week_timestamp
6/26/2022	06/21/2022	6/21/2022 12:00:00 AM
7/9/2022	07/05/2022	7/5/2022 12:00:00 AM
7/22/2022	07/19/2022	7/19/2022 12:00:00 AM
7/23/2022	07/19/2022	7/19/2022 12:00:00 AM
7/27/2022	07/26/2022	7/26/2022 12:00:00 AM
8/2/2022	08/02/2022	8/2/2022 12:00:00 AM
8/8/2022	08/02/2022	8/2/2022 12:00:00 AM
8/19/2022	08/16/2022	8/16/2022 12:00:00 AM
9/26/2022	09/20/2022	9/20/2022 12:00:00 AM
10/14/2022	10/11/2022	10/11/2022 12:00:00 AM
10/29/2022	10/25/2022	10/25/2022 12:00:00 AM

Eftersom `first_week_date`-argumentet på 1 används i `weekstart()`-funktionen i det här fallet, sätts tisdag som första dag i veckan.

Diagram med exempel på `weekstart()`-funktionen, first_week_day

Transaktion 8191 ägde rum 5 februari. `weekstart()`-funktionen identifierar att den första tisdagen före det här datumet – och därför början av veckan och det returnerade värdet – var 1 februari 0:00:00.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som returnerar en tidsmarkör för början veckan då transaktionerna ägde rum skapas som ett mått i ett diagramobjekt för programmet.

Laddningsskript

```
Transactions:

Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

För att beräkna veckostart för den vecka då en transaktion äger rum lägger du till följande mått:

- `=weekstart(date)`
- `=timestamp(weekstart(date))`

Resultattabell

datum	start_of_week	start_of_week_timestamp
1/7/2022	01/02/2022	1/2/2022 12:00:00 AM
1/19/2022	01/16/2022	1/16/2022 12:00:00 AM
2/5/2022	01/30/2022	1/30/2022 12:00:00 AM
2/28/2022	02/27/2022	2/27/2022 12:00:00 AM
3/16/2022	03/13/2022	3/13/2022 12:00:00 AM

datum	start_of_week	start_of_week_timestamp
4/1/2022	03/27/2022	3/27/2022 12:00:00 AM
5/7/2022	2022-05-01	5/1/2022 12:00:00 AM
5/16/2022	05/15/2022	5/15/2022 12:00:00 AM
6/15/2022	06/12/2022	6/12/2022 12:00:00 AM
6/26/2022	06/26/2022	6/26/2022 12:00:00 AM
7/9/2022	07/03/2022	7/3/2022 12:00:00 AM
7/22/2022	07/17/2022	7/17/2022 12:00:00 AM
7/23/2022	07/17/2022	7/17/2022 12:00:00 AM
7/27/2022	07/24/2022	7/24/2022 12:00:00 AM
8/2/2022	07/31/2022	7/31/2022 12:00:00 AM
8/8/2022	08/07/2022	8/1/2022 12:00:00 AM
8/19/2022	08/14/2022	8/14/2022 12:00:00 AM
9/26/2022	09/25/2022	9/25/2022 12:00:00 AM
10/14/2022	10/09/2022	10/9/2022 12:00:00 AM
10/29/2022	10/23/2022	10/23/2022 12:00:00 AM

`start_of_week`-måttet skapades i den föregående load-satsen genom att använda `weekstart()`-funktionen och skicka `date`-fältet som funktionens argument.

`weekstart()`-funktionen identifierar initialt vilken vecka datumvärdet infaller och returnerar en tidsstämpel för den första millisekunden av den veckan.

Diagram med exempel på `weekstart()`-funktionen, diagramobjekt

Transaktion 8191 ägde rum 5 februari. Systemvariabeln `FirstWeekDay` sätter söndag som första dag i veckan. `weekstart()`-funktionen identifierar att den första söndagen före 5 februari – och därför början av veckan – var 30 januari. Därför returnerar `start_of_week`-värdet för denna transaktion den första millisekunden denna dag, vilket är 30 januari 0:00:00.

Exempel 5 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning läses in i en tabell som heter `Payroll`.
- Data bestående av anställnings-ID, anställdas namn och den dagliga lönen som varje anställd tjänar.

Anställda börjar arbeta på måndag och arbetar sex dagar per vecka. Systemvariabeln `FirstWeekDay` får inte ändras.

Slutanvändaren vill ha ett diagramobjekt som visar intjänade löner under veckan fram till dagens datum efter anställnings-ID och anställdas namn.

Laddningsskript

```
Payroll:
Load
*
Inline
[
employee_id,employee_name,day_rate
182,Mark, $150
183,Deryck, $125
184,Dexter, $125
185,Sydney,$270
186,Agatha,$128
];
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:
   - `employee_id`
   - `employee_name`
2. Skapa sedan ett mått för att beräkna de intjänade lönerna under veckan till dagens datum: `=if(today(1)-weekstart(today(1),0,0)<7,(today(1)-weekstart(today(1),0,0))*day_rate,day_rate*6)`
3. Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

employee_id	employee_name	=if(today(1)-weekstart(today(1),0,0)<7,(today(1)-weekstart(today(1),0,0))*day_rate,day_rate*6)
182	Mark	$600.00
183	Deryck	$500.00
184	Dexter	$500.00
185	Sydney	$1080.00
186	Agatha	$512.00

`weekstart()`-funktionen sätter måndagen till första dag i veckan genom att använda dagens datum som sitt första argument och 0 som sitt tredje argument, sätter måndag till första dag i veckan och returnerar startdatumet i den aktuella veckan. Genom att subtrahera resultatet från det aktuella datumet returnerar uttrycket sedan antalet dagar som har förflutit hittills denna vecka.

Villkoret utvärderar sedan om det har varit mer än sex dagar den här veckan. I så fall multipliceras den anställdes `day_rate` med 6 dagar. Om inte, så multipliceras `day_rate` med antalet dagar som har inträffat så här långt i veckan.

weekyear

Funktionen returnerar det år som veckonumret hör till enligt miljövariablerna. Veckonummer går från 1 till cirka 52.

```
weekyear(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

Returnerad datatyp: Heltal

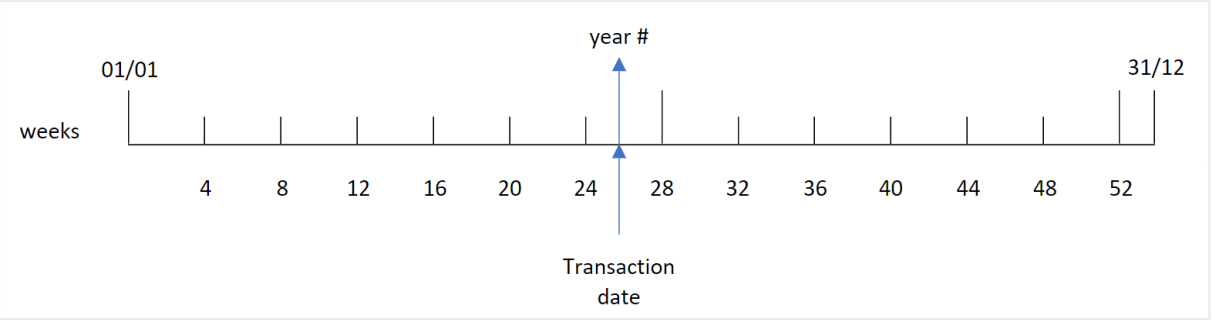
Argument

Argument	Beskrivning
timestamp	Det datum eller den tidsmarkör som ska utvärderas.
first_week_day	Anger dagen då veckan startar. Om utelämnat används värdet för variabeln**FirstWeekDay**. De möjliga värdena **first_week_day** är 0 för måndag, 1 för tisdag, 2 för onsdag, 3 för torsdag, 4 för fredag, 5 för lördag och 6 för söndag. Se *FirstWeekDay (page 257)* för mer information om systemvariabeln.
broken_weeks	Om du inte anger **broken_weeks** används värdet för variabeln **BrokenWeeks** till att definiera om veckor är brutna eller inte.
reference_day	Om du inte anger **reference_day** används värdet för variabeln **ReferenceDay** till att definiera vilken dag i januari som ska ställas in som referensdag för att definiera vecka 1. Som standard använder Qlik Sense-funktioner 4 som referensdag. Det betyder att vecka 1 måste innehålla 4 januari, eller med andra ord, vecka 1 måste alltid bestå av minst 4 dagar i januari.

`weekyear()`-funktionen bestämmer vilken vecka under året datumet infaller. Den returnerar sedan det år som motsvarar det veckonumret.

Om `BrokenWeeks` är inställt på `0` (false), kommer `weekyear()` att returnera samma som `year()`.

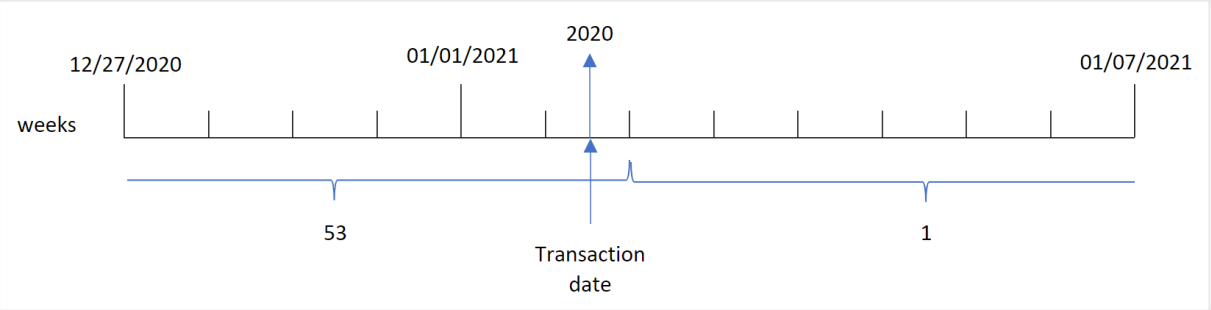
Diagram för `weekyear()`-funktionens intervall



Men om systemvariabeln `BrokenWeeks` sätts till att använda obrutna veckor får vecka 1 bara innehålla ett visst antal dagar i januari baserat på det specificerade värdet i systemvariabeln `ReferenceDay`.

Om exempelvis ett `ReferenceDay`-värde på `4` används måste vecka 1 ha minst fyra dagar i januari. Vecka 1 kan innehålla datum i december föregående år och det sista veckonumret under ett år kan innehålla datum i januari i följande år. I situationer som denna kommer `weekyear()`-funktionen att returnera ett annat värde än `year()`-funktionen.

Diagram för `weekyear()`-funktionens intervall när obrutna veckor används



Användning

Funktionen `weekyear()` är användbar när du vill jämföra sammanställningar per år. Till exempel om du vill se den totala försäljningen av produkter per år. `weekyear()`-funktionen är att föredra framför `year()` när användaren vill ha överensstämmelse med systemvariabeln `BrokenWeeks` i applikationen.

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`weekyear('12/30/1996',0,0,4)`	Returnerar 1997, eftersom vecka 1 i 1997 börjar 1996-12-30
`weekyear('01/02/1997',0,0,4)`	Returnerar 1997
`weekyear('12/28/1997',0,0,4)`	Returnerar 1997
`weekyear('12/30/1997',0,0,4)`	Returnerar 1998, eftersom vecka 1 i 1998 börjar 1997-12-29
`weekyear('01/02/1999',0,0,4)`	Returnerar 1998, eftersom vecka 53 i 1998 slutar 1999-01-03

Relaterade ämnen

Avsnitt	Interaktion
week (page 1172)	Returnerar ett heltal som motsvarar veckonumret enligt ISO 8601.
year (page 1244)	Returnerar ett heltal som motsvarar året om uttrycket tolkas som ett datum enligt standardtolkningen av tal.

Exempel 1 – brutna veckor

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för sista veckan år 2020 och den första veckan år 2021 som läses in i en tabell som heter `Transactions`.
- `BrokenWeeks`-variabeln som är inställd till 1.
- En föregående laddning som innehåller följande:
  - `weekyear()`-funktionen, inställd som fältet `week_year`, som returnerar det år när transaktionerna ägde rum.
  - `week()`-funktionen, inställd som fältet `week`, som visar veckonumret för varje transaktionsdatum.

Laddningsskript

```
SET BrokenWeeks=1;

Transactions:
       Load
       *,
       week(date) as week,
       weekyear(date) as week_year
       ;
Load
*
Inline
[
id,date,amount
8176,12/28/2020,19.42
8177,12/29/2020,23.80
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `week`
- `week_year`

Resultattabell

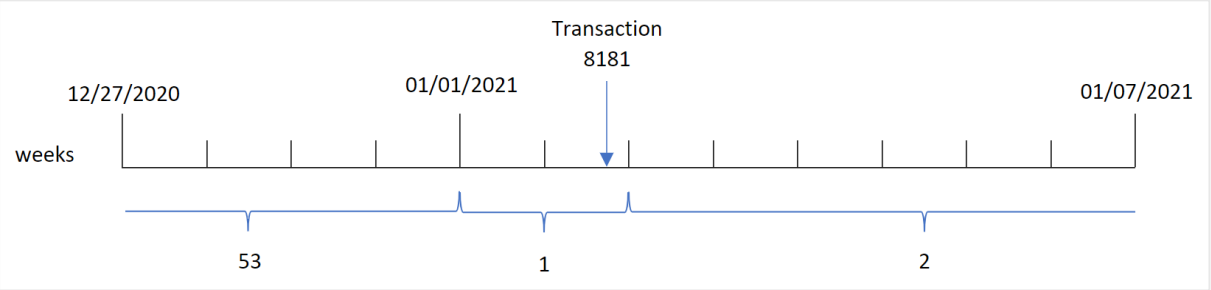
Id	Datum	vecka	vecka_år
8176	2020-12-28	53	2020
8177	29/12/2020	53	2020
8178	2020-12-30	53	2020
8179	12/31/2020	53	2020
8180	2021-01-01	1	2021
8181	02/01/2021	1	2021
8182	03/01/2021	2	2021

Id	Datum	vecka	vecka_år
8183	2021-01-04	2	2021
8184	05/01/2021	2	2021
8185	06/01/2021	2	2021
8186	07/01/2021	2	2021

'week_year'-fältet skapades i den föregående load-satsen genom att använda weekyear()-funktionen och skicka datumfältet som funktionens argument.

Systemvariabeln BrokenWeeks sätts till 1, vilket innebär att applikationen använder brutna veckor. Vecka 1 börjar 1 januari.

Diagram för weekyear()-funktionens intervall när brutna veckor används



Transaktion 8181 äger rum 2 januari, vilket ingår i vecka 1. Därför returneras värdet 2021 för week_year-fältet.

Exempel 2 – obrutna veckor

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för sista veckan år 2020 och första veckan år 2021 som läses in i en tabell som heter Transactions.
- BrokenWeeks-variabeln som är inställd till 0.
- En föregående laddning som innehåller följande:
  - Funktionen weekyear(), inställd som fältet week_year, som returnerar det år när transaktionerna ägde rum.
  - week()-funktionen, inställd som fältet week, som visar veckonumret för varje transaktionsdatum.

Men i det här exemplet är företagspolicyn att använda obrutna veckor.

Laddningsskript

```
SET BrokenWeeks=0;

Transactions:
       Load
       *,
       week(date) as week,
       weekyear(date) as week_year
       ;
Load
*
Inline
[
id,date,amount
8176,12/28/2020,19.42
8177,12/29/2020,23.80
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `week`
- `week_year`

Resultattabell

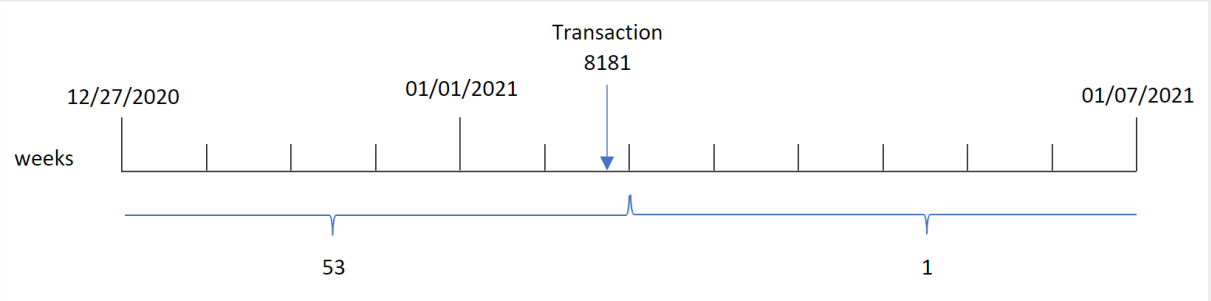
Id	Datum	vecka	vecka_år
8176	2020-12-28	53	2020
8177	29/12/2020	53	2020
8178	2020-12-30	53	2020
8179	12/31/2020	53	2020
8180	2021-01-01	53	2020
8181	02/01/2021	53	2020
8182	03/01/2021	1	2021

Id	Datum	vecka	vecka_år
8183	2021-01-04	1	2021
8184	05/01/2021	1	2021
8185	06/01/2021	1	2021
8186	07/01/2021	1	2021

Systemvariabeln `BrokenWeeks` sätts till `0`, vilket innebär att programmet använder obrutna veckor. Därför behöver inte vecka 1 börja 1 januari.

Vecka 53 2020 fortsätter till slutet av 2 januari 2021 och vecka 1 2021 börjar söndag 3 januari 2021.

Diagram för `weekyear()`-funktionens intervall när obrutna veckor används



Transaktion 8181 äger rum 2 januari, vilket ingår i vecka 1. Därför returnerar den värdet 2021 för fältet 'week_year'.

Exempel 3 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är dock datauppsättningen oförändrad och har lästs in i programmet. Beräkningen som returnerar veckonumret för det år då transaktionerna ägde rum skapas som ett mått i ett diagram i applikationen.

Laddningsskript

```
SET BrokenWeeks=1;

Transactions:
Load
*
Inline
[
id,date,amount
8176,12/28/2020,19.42
```

```
8177,12/29/2020,23.80
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`

För att beräkna veckan då en transaktion äger rum skapar du följande mått:

- `=week(date)`

För att beräkna året då en transaktion äger rum baserat på veckonumret skapar du följande mått:

- `=weekyear(date)`

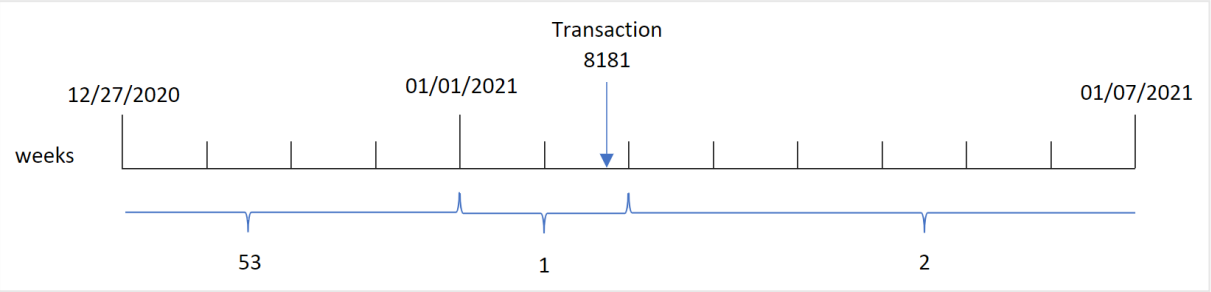
Resultattabell

Id	Datum	vecka	vecka_år
8176	2020-12-28	53	2020
8177	29/12/2020	53	2020
8178	2020-12-30	53	2020
8179	12/31/2020	53	2020
8180	2021-01-01	1	2021
8181	02/01/2021	1	2021
8182	03/01/2021	2	2021
8183	2021-01-04	2	2021
8184	05/01/2021	2	2021
8185	06/01/2021	2	2021
8186	07/01/2021	2	2021

‘`week_year`’-fältet skapades i den föregående load-satsen genom att använda `weekyear()`-funktionen och skicka datumfältet som funktionens argument.

Systemvariabeln `BrokenWeeks` sätts till `1`, vilket innebär att applikationen använder brutna veckor. Vecka 1 börjar 1 januari.

Diagram för `weekyear()`-funktionens intervall när brutna veckor används



Transaktion 8181 äger rum 2 januari, vilket ingår i vecka 1. Därför returneras värdet 2021 för `week_year`-fältet.

Exempel 4 – Scenario

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner för sista veckan år 2020 och första veckan år 2021 som läses in i en tabell som heter `Transactions`.
- `BrokenWeeks`-variabeln som är inställd till `0`. Detta betyder att applikationen kommer att använda obrutna veckor.
- `ReferenceDay`-variabeln som är inställd till `2`. Detta betyder att året kommer att börja 2 januari och innehålla minst två dagar i januari.
- `FirstWeekDay`-variabeln som är inställd till `1`. Detta innebär att veckans första dag blir en tisdag.

Företagspolicyn är att använda brutna vewckor. Slutanvändaren vill ha ett diagram som visar den totala försäljningen per år. Applikationen använder obrutna veckor och vecka 1 innehåller minst två dagar i januari.

Laddningsskript

```
SET BrokenWeeks=0;
SET ReferenceDay=2;
SET FirstWeekDay=1;

Transactions:
Load
*
Inline
[
id,date,amount
```

```
8176,12/28/2020,19.42
8177,12/29/2020,23.80
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell.

För att beräkna året då en transaktion äger rum baserat på veckonumret skapar du följande mått:

- `=weekyear(date)`

För att beräkna total försäljning skapar du följande mått:

- `sum(amount)`

Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

weekyear(date)	=sum(amount)
2020	19.42
2021	373.37

year

Denna funktion returnerar ett heltal som motsvarar året om **expression** tolkas som ett datum enligt standardtolkningen av tal.

Syntax:

```
year(expression)
```

Returnerad datatyp: heltal

`year()`-funktionen är tillgänglig både som skript- och diagramfunktionen. Den här funktionen returnerar året för ett visst datum. Den används vanligtvis för att skapa ett årsfält som en dimension i en Master Calendar.

Användning

Funktionen `year()` är användbar när du vill jämföra sammanställningar per år. Funktionen kan till exempel användas om du vill se den totala försäljningen av produkter per år.

Dessa dimensioner kan antingen skapas i laddningsskriptet genom att använda funktionen för att skapa ett fält i en Master Calendar-tabell. Eller användas direkt i ett diagram som en beräknad dimension.

Exempel på funktioner

Exempel	Resultat
`year( '2012-10-12' )`	returnerar 2012
`year( '35648' )`	returnerar 1997, eftersom 35648 = 1997-08-06

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – DateFormat datauppsättning (skript)

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum som läses in i en tabell med namnet `Master Calendar`.
- `DateFormat`-standardsystemvariabeln MM/DD/YYYY används.
- En föregående laddning som används för att skapa ytterligare ett fält, `year`, med hjälp av `year()`-funktionen.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Master_Calendar:

    Load
        date,
        year(date) as year
        ;
Load
date
```

```
Inline
[
date
12/28/2020
12/29/2020
12/30/2020
12/31/2020
01/01/2021
01/02/2021
01/03/2021
01/04/2021
01/05/2021
01/06/2021
01/07/2021
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `year`

Resultattabell

datum	år
2020-12-28	2020
29/12/2020	2020
30/12/2020	2020
12/31/2020	2020
2021-01-01	2021
02/01/2021	2021
03/01/2021	2021
04/01/2021	2021
05/01/2021	2021
06/01/2021	2021
07/01/2021	2021

Exempel 2 – ANSI-datum

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum som läses in i en tabell med namnet `Master Calendar`.
- Standardsystemvariabeln `DateFormat` (MM/DD/YYYY) används. Emellertid är de datum som är inkluderade i datauppsättningen i ANSI:s standarddatumformat.
- En föregående laddning som används för att skapa ytterligare ett fält benämnt `year` med hjälp av `year()`-funktionen.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Master_Calendar:
    Load
        date,
        year(date) as year
        ;
Load
date
Inline
[
date
2020-12-28
2020-12-29
2020-12-30
2020-12-31
2021-01-01
2021-01-02
2021-01-03
2021-01-04
2021-01-05
2021-01-06
2021-01-07
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `year`

Resultattabell

datum	år
2020-12-28	2020
2020-12-29	2020
2020-12-30	2020

datum	år
2020-12-31	2020
2021-01-01	2021
2021-01-02	2021
2021-01-03	2021
2021-01-04	2021
2021-01-05	2021
2021-01-06	2021
2021-01-07	2021

Exempel 3 – Oformaterade datum

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning med datum i numeriskt format som läses in i en tabell med namnet `Master Calendar`.
- Standardsystemvariabeln `DateFormat` (MM/DD/YYYY) används.
- En föregående laddning som används för att skapa ytterligare ett fält, `year`, med hjälp av `year()`-funktionen.

Det ursprungliga oformaterade datumet, benämnt `unformatted_date`, laddas och för tydlighets skull används ytterligare ett fält benämnt `long_date` för att konvertera det numeriska datumet till ett formaterat datumfält med `date()`-funktionen.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Master_Calendar:
    Load
        unformatted_date,
        date(unformatted_date) as long_date,
        year(unformatted_date) as year
        ;
Load
unformatted_date
Inline
[
unformatted_date
```

```
44868
44898
44928
44958
44988
45018
45048
45078
45008
45038
45068
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `unformatted_date`
- `long_date`
- `year`

Resultattabell

oformaterat_datum	långt_datum	år
44868	11/03/2022	2022
44898	12/03/2022	2022
44928	01/02/2023	2023
44958	02/01/2023	2023
44988	03/03/2023	2023
45008	03/23/2023	2023
45018	04/02/2023	2023
45038	04/22/2023	2023
45048	05/02/2023	2023
45068	05/22/2023	2023
45078	06/01/2023	2023

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna skriptredigeraren och lägg till laddningsskriptet nedan i en ny flik.

I det här exemplet laddas en datauppsättning med gjorda beställningar i en tabell som heter Sales. Tabellen innehåller tre fält:

- `id`
- `sales_date`
- `amount`

Garantier på försäljning av produkter de senaste två åren från datum för försäljningen. Uppgiften är att skapa ett mått i ett diagram för att bestämma vilket år varje garanti kommer att sluta gälla.

Laddningsskript

```
Sales:
Load
id,
sales_date,
amount
Inline
[
id,sales_date,amount
1,12/28/2020,231.24,
2,12/29/2020,567.28,
3,12/30/2020,364.28,
4,12/31/2020,575.76,
5,01/01/2021,638.68,
6,01/02/2021,785.38,
7,01/03/2021,967.46,
8,01/04/2021,287.67
9,01/05/2021,764.45,
10,01/06/2021,875.43,
11,01/07/2021,957.35
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `sales_date`.

Skapa följande mått:

```
=year(sales_date+365*2)
```

Resultattabell

försäljningsdatum	=år(försäljningsdatum+365*2)
2020-12-28	2022
29/12/2020	2022
30/12/2020	2022
12/31/2020	2022

försäljningsdatum	=år(försäljningsdatum+365*2)
2021-01-01	2023
02/01/2021	2023
03/01/2021	2023
04/01/2021	2023
05/01/2021	2023
06/01/2021	2023
07/01/2021	2023

Resultatet av det här måttet återfinns i ovanstående tabell. För att lägga till ett år till ett datum multiplicerar du 365 med 2 och lägger till resultatet i försäljningsdatumet. Därför har försäljning som ägde rum under 2020 utgångsåret 2022.

yearend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör med den sista millisekunden av den sista dagen i det år som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

Syntax:

```
YearEnd( date[, period_no[, first_month_of_year = 1]])
```

Med andra ord bestämmer `yearend()`-funktionen vilket år datumet infaller. Den returnerar sedan en tidsstämpel, i datumformat, för den sista millisekunden av det året. Årets första månad är som standard januari. Du kan dock ändra vilken månad som anges som första månaden genom att använda `first_month_of_year`-argumentet i `yearend()`-funktionen.

> *`yearend()`-funktionen tar inte hänsyn till systemvariabeln `FirstMonthOfYear`. Året börjar den 1 januari om inte `first_month_of_year`-argumentet används för att ändra det.*

Diagram för `yearend()`-funktionen.

Användning

`yearend()`-funktionen används som en del av ett uttryck när du vill att beräkningen ska använda den del av året som ännu inte har inträffat. Till exempel om du vill beräkna den totala ränta som ännu inte uppkommit under året.

Returnerad datatyp: dual

Argument

Argument	Beskrivning
date	Det datum eller den tidsmarkör som ska utvärderas.
period_no	**period_no** är ett heltal, där värdet 0 anger det år som innehåller **date**. Negativa värden i **period_no** anger föregående år och positiva värden anger efterföljande år.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Du kan använda följande värden för att ställa in den första månaden på året i argumentet `first_month_of_year`:

first_month_of_year values

Månad	Värde
februari	2
mars	3
april	4
Maj	5
juni	6
juli	7
augusti	8
september	9
oktober	10
november	11
december	12

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så

kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`yearend('10/19/2001')`	Returnerar 12/31/2001 23:59:59.
`yearend('10/19/2001', -1)`	Returnerar 12/31/2000 23:59:59.
`yearend('10/19/2001', 0, 4)`	Returnerar 03/31/2002 23:59:59.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner mellan år 2020 och 2022 som läses in i en tabell som heter `Transactions`.
- Datumfältet har tillhandahållits i `(MM/DD/YYYY)`-formatet med systemvariabeln `DateFormat`.
- En föregående LOAD-sats som innehåller följande:
  - `yearend()`-funktion som är inställd som `year_end`-fält.
  - `Timestamp()`-funktion som är inställd som `year_end_timestamp`-fält.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearend(date) as year_end,
       timestamp(yearend(date)) as year_end_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
```

```
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `year_end`
- `year_end_timestamp`

Resultattabell

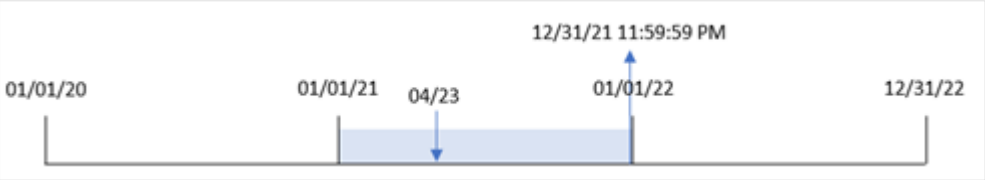
id	datum	year_end	year_end_timestamp
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM
8192	05/21/2020	12/31/2020	12/31/2020 11:59:59 PM
8193	08/14/2020	12/31/2020	12/31/2020 11:59:59 PM
8194	10/07/2020	12/31/2020	12/31/2020 11:59:59 PM
8195	12/05/2020	12/31/2020	12/31/2020 11:59:59 PM
8196	01/22/2021	12/31/2021	12/31/2021 11:59:59 PM
8197	02/03/2021	12/31/2021	12/31/2021 11:59:59 PM
8198	03/17/2021	12/31/2021	12/31/2021 11:59:59 PM
8199	04/23/2021	12/31/2021	12/31/2021 11:59:59 PM
8200	05/04/2021	12/31/2021	12/31/2021 11:59:59 PM

id	datum	year_end	year_end_timestamp
8201	06/30/2021	12/31/2021	12/31/2021 11:59:59 PM
8202	07/26/2021	12/31/2021	12/31/2021 11:59:59 PM
8203	12/27/2021	12/31/2021	12/31/2021 11:59:59 PM
8204	06/06/2022	12/31/2022	12/31/2022 11:59:59 PM
8205	07/18/2022	12/31/2022	12/31/2022 11:59:59 PM
8206	11/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	12/12/2022	12/31/2022	12/31/2022 11:59:59 PM

`year_end`-fältet skapades i den föregående load-satsen genom att använda `yearend()`-funktionen och skicka datumfältet som funktionens argument.

Funktionen `yearend()` identifierar initialt vilket år datumvärdet infaller och returnerar en tidsstämpel för den sista millisekunden av det året.

Diagram över `yearend()`-funktion med transaktion 8199 vald.



Transaktion 8199 ägde rum den 23 april 2021. `yearend()`-funktionen returnerar den sista millisekunden av det året, vilket är den 31 december kl 11:59:59 PM.

Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Samma datauppsättning och scenario som det första exemplet används.

Men i det här exemplet är uppgiften att skapa ett fält, `previous_year_end`, som returnerar slutdatum och tidsstämpel för året innan det år då en transaktion ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearend(date,-1) as previous_year_end,
       timestamp(yearend(date,-1)) as previous_year_end_timestamp
       ;
```

```
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `previous_year_end`
- `previous_ year_end_timestamp`

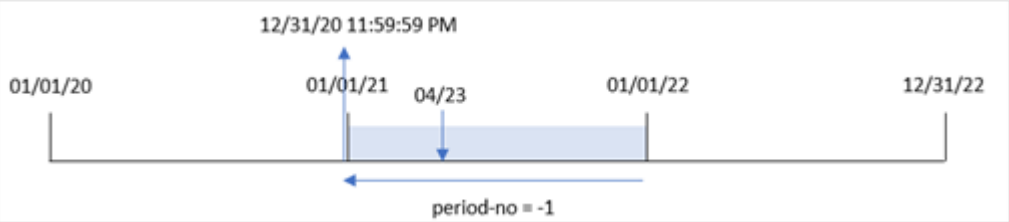
Resultattabell

id	datum	previous_year_end	previous_year_end_timestamp
8188	01/13/2020	12/31/2019	12/31/2019 11:59:59 PM
8189	02/26/2020	12/31/2019	12/31/2019 11:59:59 PM
8190	03/27/2020	12/31/2019	12/31/2019 11:59:59 PM
8191	04/16/2020	12/31/2019	12/31/2019 11:59:59 PM
8192	05/21/2020	12/31/2019	12/31/2019 11:59:59 PM
8193	08/14/2020	12/31/2019	12/31/2019 11:59:59 PM
8194	10/07/2020	12/31/2019	12/31/2019 11:59:59 PM

id	datum	previous_year_end	previous_year_end_timestamp
8195	12/05/2020	12/31/2019	12/31/2019 11:59:59 PM
8196	01/22/2021	12/31/2020	12/31/2020 11:59:59 PM
8197	02/03/2021	12/31/2020	12/31/2020 11:59:59 PM
8198	03/17/2021	12/31/2020	12/31/2020 11:59:59 PM
8199	04/23/2021	12/31/2020	12/31/2020 11:59:59 PM
8200	05/04/2021	12/31/2020	12/31/2020 11:59:59 PM
8201	06/30/2021	12/31/2020	12/31/2020 11:59:59 PM
8202	07/26/2021	12/31/2020	12/31/2020 11:59:59 PM
8203	12/27/2021	12/31/2020	12/31/2020 11:59:59 PM
8204	06/06/2022	12/31/2021	12/31/2021 11:59:59 PM
8205	07/18/2022	12/31/2021	12/31/2021 11:59:59 PM
8206	11/14/2022	12/31/2021	12/31/2021 11:59:59 PM
8207	12/12/2022	12/31/2021	12/31/2021 11:59:59 PM

Eftersom en `period_no` på `-1` användes som förskjutningsargument i `yearend()`-funktionen så identifierar funktionen först året då transaktionerna äger rum. Den tittar sedan en vecka innan och identifierar den första millisekunden i den veckan.

Diagram för `yearend()`-funktionen med en period_no på -1.



Transaktion 8199 ägde rum den 23 april 2021. `yearend()`-funktionen returnerar den sista millisekunden av det året, vilket är den 31 december 2020, kl. 23:59:59 för `previous_year_end`-fältet

Exempel 3 – first_month_of_year

Laddningsskript och resultat

Översikt

Samma datauppsättning och scenario som det första exemplet används.

Men i det här exemplet är företagets policy att året ska börja den 1 april.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearend(date,0,4) as year_end,
       timestamp(yearend(date,0,4)) as year_end_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- id
- date
- year_end
- year_end_timestamp

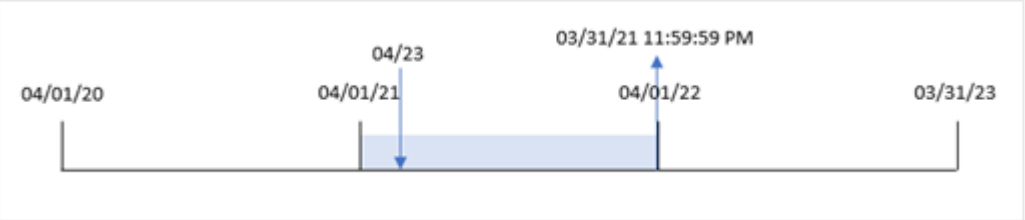
Resultattabell

id	datum	year_end	year_end_timestamp
8188	01/13/2020	03/31/2020	3/31/2020 11:59:59 PM

id	datum	year_end	year_end_timestamp
8189	02/26/2020	03/31/2020	3/31/2020 11:59:59 PM
8190	03/27/2020	03/31/2020	3/31/2020 11:59:59 PM
8191	04/16/2020	03/31/2021	3/31/2021 11:59:59 PM
8192	05/21/2020	03/31/2021	3/31/2021 11:59:59 PM
8193	08/14/2020	03/31/2021	3/31/2021 11:59:59 PM
8194	10/07/2020	03/31/2021	3/31/2021 11:59:59 PM
8195	12/05/2020	03/31/2021	3/31/2021 11:59:59 PM
8196	01/22/2021	03/31/2021	3/31/2021 11:59:59 PM
8197	02/03/2021	03/31/2021	3/31/2021 11:59:59 PM
8198	03/17/2021	03/31/2021	3/31/2021 11:59:59 PM
8199	04/23/2021	03/31/2022	3/31/2022 11:59:59 PM
8200	05/04/2021	03/31/2022	3/31/2022 11:59:59 PM
8201	06/30/2021	03/31/2022	3/31/2022 11:59:59 PM
8202	07/26/2021	03/31/2022	3/31/2022 11:59:59 PM
8203	12/27/2021	03/31/2022	3/31/2022 11:59:59 PM
8204	06/06/2022	03/31/2023	3/31/2023 11:59:59 PM
8205	07/18/2022	03/31/2023	3/31/2023 11:59:59 PM
8206	11/14/2022	03/31/2023	3/31/2023 11:59:59 PM
8207	12/12/2022	03/31/2023	3/31/2023 11:59:59 PM

Eftersom `first_month_of_year`-argumentet för `4` används i `yearend()`-funktionen, ställer det in årets första dag till 1 april och årets sista dag till 31 mars.

Diagram för `yearend()`-funktionen med april som årets första månad.



Transaktion 8199 ägde rum den 23 april 2021. Eftersom `yearend()`-funktionen ställer in början av året till 1 april, returnerar den 31 mars 2022 som `year_end`-värdet för transaktionen.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är dock datauppsättningen oförändrad och har lästs in i programmet. Beräkningen som returnerar en tidsmarkör för slutdatum för året då en transaktionen ägde rum skapas som ett mått i ett diagramobjekt för programmet.

Laddningsskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`

För att beräkna vilket år en transaktion ägde rum skapar du följande mått:

- `=yearend(date)`
- `=timestamp(yearend(date))`

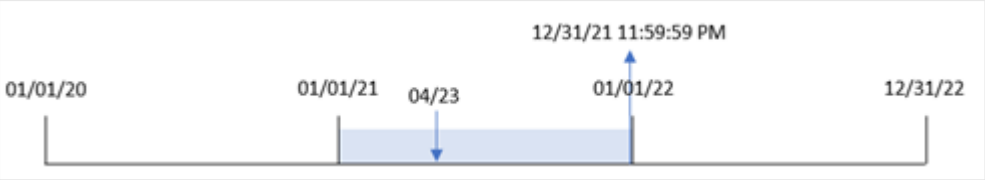
Resultattabell

id	datum	=yearend(date)	=timestamp(yearend(date))
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM
8192	05/21/2020	12/31/2020	12/31/2020 11:59:59 PM
8193	08/14/2020	12/31/2020	12/31/2020 11:59:59 PM
8194	10/07/2020	12/31/2020	12/31/2020 11:59:59 PM
8195	12/05/2020	12/31/2020	12/31/2020 11:59:59 PM
8196	01/22/2021	12/31/2021	12/31/2021 11:59:59 PM
8197	02/03/2021	12/31/2021	12/31/2021 11:59:59 PM
8198	03/17/2021	12/31/2021	12/31/2021 11:59:59 PM
8199	04/23/2021	12/31/2021	12/31/2021 11:59:59 PM
8200	05/04/2021	12/31/2021	12/31/2021 11:59:59 PM
8201	06/30/2021	12/31/2021	12/31/2021 11:59:59 PM
8202	07/26/2021	12/31/2021	12/31/2021 11:59:59 PM
8203	12/27/2021	12/31/2021	12/31/2021 11:59:59 PM
8204	06/06/2022	12/31/2022	12/31/2022 11:59:59 PM
8205	07/18/2022	12/31/2022	12/31/2022 11:59:59 PM
8206	11/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	12/12/2022	12/31/2022	12/31/2022 11:59:59 PM

`end_of_year`-måttet skapades i ett diagramobjektet genom att använda `yearend()`-funktionen och skicka datumfältet som funktionens argument.

`yearend()`-funktionen identifierar initialt vilket år datumvärdet infaller och returnerar en tidsstämpel för den sista millisekunden av det året.

Diagram för `yearend()`-funktion som visar att Transaktion 8199 ägde rum i april.



Transaktion 8199 ägde rum den 23 april 2021. `yearend()`-funktionen returnerar den sista millisekunden av det året, vilket är den 31 december kl 11:59:59 PM.

Exempel 5 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning läses in i en tabell som heter `Employee_Expenses`. Tabellen innehåller följande fält:
    - Anställnings-ID
    - Namn för den anställde
    - genomsnittliga dagliga kostnadsanspråk för varje anställd

Slutanvändaren vill ha ett diagramobjekt som visar, efter anställnings-id och anställds namn, de beräknade kostnadsanspråken som fortfarande återstår för resten av året. Räkenskapsåret börjar i januari.

Laddningsskript

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- employee_id
- employee_name

För att beräkna kostnadsanspråken skapar du följande mått:

```
=(yearend(today(1))-today(1))*avg_daily_claim
```

Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

employee_id	employee_name	=(yearend(today(1))-today(1))*avg_daily_claim
182	Mark	$3240.00
183	Deryck	$2700.00
184	Dexter	$2700.00
185	Sydney	$5832.00
186	Agatha	$3888.00

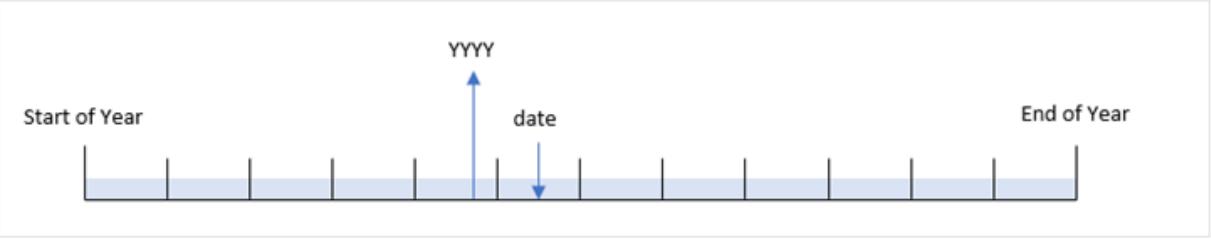
Genom att använda dagens datum som enda argument returnerar `yearend()`-funktionen slutdatumet för det aktuella året. Sedan, genom att subtrahera dagens datum från årets slutdatum, returnerar uttrycket antalet dagar som återstår av detta år.

Detta värde multipliceras sedan med den genomsnittliga dagliga kostnadsanspråket från varje anställd för att beräkna det uppskattade värdet av anspråk som varje anställd förväntas göra under det återstående året.

yearname

Denna funktion returnerar ett fyrsiffrigt år som visningsvärde med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden av den första dagen på det år som innehåller **date**.

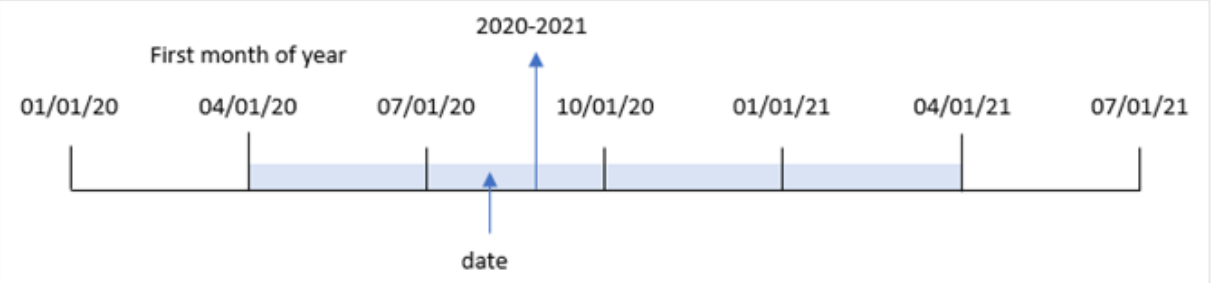
Diagram över tidsintervall för `yearname()` -funktionen.



`yearname()`-funktionen skiljer sig från `year()`-funktionen eftersom den låter dig förskjuta det datum du vill utvärdera och låter dig ställa in årets första månad.

Om årets första månad inte är januari kommer funktionen att returnera de två fyrsiffriga åren över den tolvmånadersperiod som innehåller datumet. Till exempel, om början av året är april och datumet som utvärderas är 06/30/2020 så skulle det returnerade resultatet vara `2020-2021`.

Diagram för `yearname()`-funktionen med april inställt som årets första månad.



Syntax:

```
YearName(date[, period_no[, first_month_of_year]] )
```

Returnerad datatyp: dual

Argument	Beskrivning
date	Det datum eller den tidsmarkör som ska utvärderas.
period_no	**period_no** är ett heltal, där värdet 0 anger det år som innehåller **date**. Negativa värden i **period_no** anger föregående år och positiva värden anger efterföljande år.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**. Visningsvärdet blir då en sträng som visar två år.

Du kan använda följande värden för att ställa in den första månaden på året i argumentet `first_month_of_year`:

first_month_of_year values

Månad	Värde
februari	2
mars	3
april	4
Maj	5
juni	6
juli	7
augusti	8
september	9
oktober	10
november	11
december	12

Användning

`yearname()`-funktionen är användbar för att jämföra aggregeringar per år. Till exempel om du vill se den totala försäljningen av produkter per vecka.

Dessa dimensioner kan skapas i laddningsskriptet genom att använda funktionen för att skapa ett fält i en Master Calendar-tabell. De kan också skapas i ett diagram som beräknade dimensioner

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`yearname('10/19/2001')`	Returnerar '2001.'
`yearname('10/19/2001',-1)`	Returnerar "2000".
`yearname('10/19/2001',0,4)`	Returnerar "2000-2001".

Relaterade ämnen

Avsnitt	Beskrivning
year (page 1244)	Denna funktion returnerar ett heltal som motsvarar året om uttrycket tolkas som ett datum enligt standardtolkningen av tal.

Exempel 1 – inga ytterligare argument

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner mellan år 2020 och 2022 som läses in i en tabell som heter `Transactions`.
- Systemvariabeln `DateFormat` som är inställd på `MM/DD/YYYY`.
- En föregående laddning som innehåller `yearname()`-funktionen som är inställd på `year_name`-fältet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
      *,
      yearname(date) as year_name
      ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `year_name`

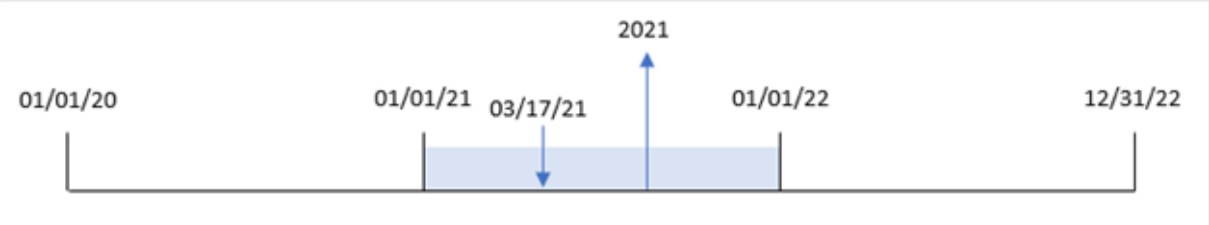
Resultattabell

datum	year_name
01/13/2020	2020
02/26/2020	2020
03/27/2020	2020
04/16/2020	2020
05/21/2020	2020
08/14/2020	2020
10/07/2020	2020
12/05/2020	2020
01/22/2021	2021
02/03/2021	2021
03/17/2021	2021
04/23/2021	2021
05/04/2021	2021
06/30/2021	2021
07/26/2021	2021
12/27/2021	2021
06/06/2022	2022
07/18/2022	2022
11/14/2022	2022
12/12/2022	2022

`year_name`-fältet skapades i den föregående load-satsen genom att använda `yearname()`-funktionen och skicka datumfältet som funktionens argument.

`yearname()`-funktionen identifierar vilket år datumvärdet infaller och returnerar detta som ett fyrsiffrigt årtal.

Diagram över `yearname()`-funktionen som visar 2021 som årsvärde.



Exempel 2 – period_no

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner mellan år 2020 och 2022 som läses in i en tabell som kallas transaktioner.
- Systemvariabeln DateFormat som är inställd på `MM/DD/YYYY`.
- En föregående inläsning som innehåller `yearname()`-funktionen som är inställd på `year_name`-fältet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
      *,
      yearname(date,-1) as prior_year_name
      ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

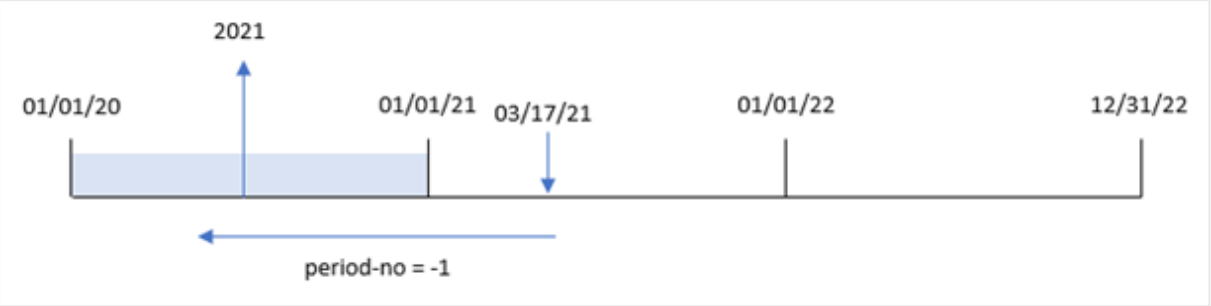
- `date`
- `prior_year_name`

Resultattabell

datum	prior_year_name
01/13/2020	2019
02/26/2020	2019
03/27/2020	2019
04/16/2020	2019
05/21/2020	2019
08/14/2020	2019
10/07/2020	2019
12/05/2020	2019
01/22/2021	2020
02/03/2021	2020
03/17/2021	2020
04/23/2021	2020
05/04/2021	2020
06/30/2021	2020
07/26/2021	2020
12/27/2021	2020
06/06/2022	2021
07/18/2022	2021
11/14/2022	2021
12/12/2022	2021

Eftersom en `period_no` på `-1` används som förskjutningsargument i `yearname()`-funktionen så identifierar funktionen först året då transaktionerna äger rum. Funktionen flyttas sedan till ett år tidigare och returnerar det resulterande året.

Diagram för `yearname()`-funktionen med `period_no` inställt på `-1`.



Exempel 3 – first_month_of_year

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning som i det första exemplet.
- Systemvariabeln `DateFormat` som är inställd på `MM/DD/YYYY`.
- En föregående inläsning som innehåller `yearname()`-funktionen som är inställd på `year_name`-fältet.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
      *,
      yearname(date,0,4) as year_name
      ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
```

```
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `year_name`

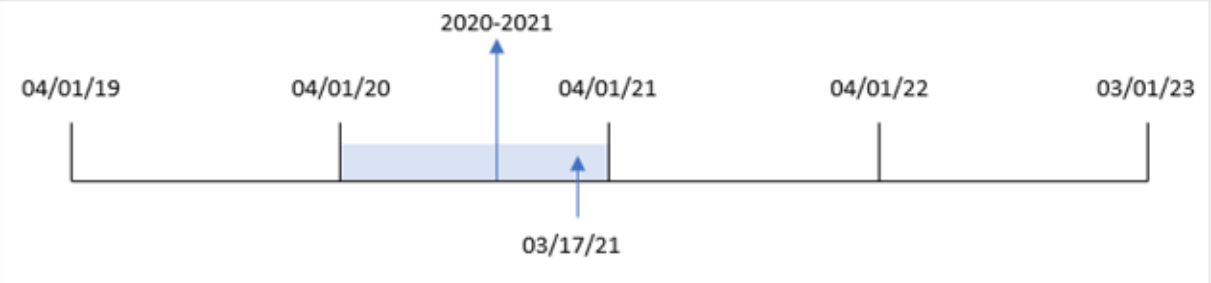
Resultattabell

datum	year_name
01/13/2020	2019-2020
02/26/2020	2019-2020
03/27/2020	2019-2020
04/16/2020	2020-2021
05/21/2020	2020-2021
08/14/2020	2020-2021
10/07/2020	2020-2021
12/05/2020	2020-2021
01/22/2021	2020-2021
02/03/2021	2020-2021
03/17/2021	2020-2021
04/23/2021	2021-2022
05/04/2021	2021-2022
06/30/2021	2021-2022
07/26/2021	2021-2022
12/27/2021	2021-2022
06/06/2022	2022-2023
07/18/2022	2022-2023
11/14/2022	2022-2023
12/12/2022	2022-2023

Eftersom `first_month_of_year`-argumentet för `4` används i `yearname()`-funktionen, flyttas början av året från 1 januari till 1 april. Därför innefattar varje tolvmånadersperiod två kalenderår och funktionen `yearname()` returnerar de två fyrsiffriga åren för utvärderade datum.

Transaktion 8199 ägde rum den 17 mars 2021. `yearname()`-funktionen ställer in början av året på den 1 april och slutet den 30 mars. Därför inträffade transaktion 8198 under årsperioden från 1 april 2020 till 30 mars 2021. På grund av detta returnerar `yearname()`-funktionen värdet 2020-2021.

Diagram för `yearname()`-funktionen med mars inställt som årets första månad.



Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning som i det första exemplet.
- Systemvariabeln `DateFormat` som är inställd på `MM/DD/YYYY`.

Fältet som returnerar året då transaktionen skedde i skapas dock som ett mått i ett diagramobjekt.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
```

```
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

```
date
```

För att beräkna fältet `year_name` skapar du det här måttet:

```
=yearname(date)
```

Resultattabell

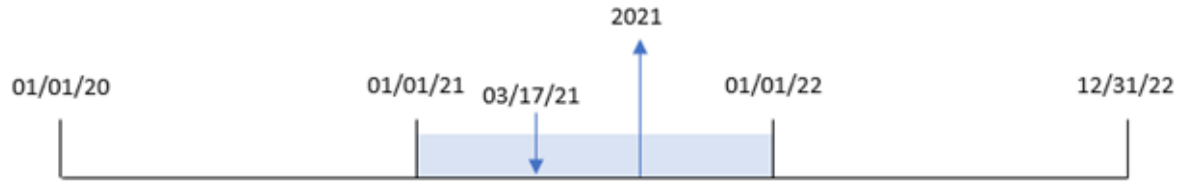
datum	=yearname(date)
01/13/2020	2020
02/26/2020	2020
03/27/2020	2020
04/16/2020	2020
05/21/2020	2020
08/14/2020	2020
10/07/2020	2020
12/05/2020	2020
01/22/2021	2021
02/03/2021	2021
03/17/2021	2021
04/23/2021	2021
05/04/2021	2021
06/30/2021	2021
07/26/2021	2021
12/27/2021	2021

datum	=yearname(date)
06/06/2022	2022
07/18/2022	2022
11/14/2022	2022
12/12/2022	2022

`year_name`-måttet skapades i ett diagramobjektet genom att använda `yearname()`-funktionen och skicka datumfältet som funktionens argument.

`yearname()`-funktionen identifierar vilket år datumvärdet infaller och returnerar detta som ett fyrsiffrigt årtal.

Diagram över `yearname()`-funktionen med 2021 som årsvärde.



Exempel 5 – Scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning som i det första exemplet.
- Systemvariabeln `DateFormat` som är inställd på `MM/DD/YYYY`.

Slutanvändaren vill ha ett diagram som visar den totala försäljningen per kvartal för transaktionerna. Använd `yearname()`-funktionen som en beräknad dimension för att skapa detta diagram när dimensionen `yearname()` inte är tillgänglig i datamodellen.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
```

```
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell.

För att jämföra aggregationer efter år, skapa den här beräknade dimensionen:

```
=yearname(date)
```

Skapa det här måttet:

```
=sum(amount)
```

Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

=yearname(date)	=sum(amount)
2020	$463.55
2021	$457.69
2022	$294.35

yearstart

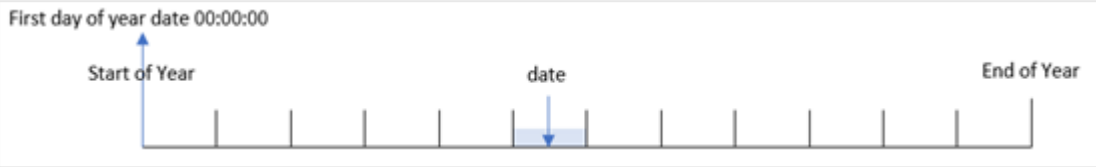
Denna funktion returnerar en tidsmarkör som motsvarar starten av den första dagen i det år som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

Syntax:

```
YearStart(date[, period_no[, first_month_of_year]])
```

Med andra ord bestämmer `yearstart()`-funktionen vilket år datumet infaller. Den returnerar sedan en tidsstämpel, i datumformat, för den sista millisekunden av det året. Den första månaden för året är som standard januari. Du kan dock ändra vilken månad som anges som första månaden genom att använda `first_month_of_year`-argumentet i `yearstart()`-funktionen.

Diagram över `yearstart()`-funktionen som visar tidsintervallet som funktionen kan täcka.



Användning

`yearstart()`-funktionen används som en del av ett uttryck när du vill att beräkningen ska använda den del av året som hittills har förflutit. Till exempel om du vill beräkna den ränta som har ackumulerats under ett år hittills.

Returnerad datatyp: dual

Argument

Argument	Beskrivning
date	Det datum eller den tidsmarkör som ska utvärderas.
period_no	**period_no** är ett heltal, där värdet 0 anger det år som innehåller **date**. Negativa värden i **period_no** anger föregående år och positiva värden anger efterföljande år.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Följande månader kan användas i `first_month_of_year argument`:

first_month_of_year values

Månad	Värde
Februari	2
mars	3
april	4
Maj	5
juni	6
juli	7
augusti	8
september	9

Månad	Värde
oktober	10
November	11
december	12

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel på funktioner

Exempel	Resultat
`yearstart('10/19/2001')`	Returns 01/01/2001 00:00:00.
`yearstart('10/19/2001',-1)`	Returns 01/01/2000 00:00:00.
`yearstart('10/19/2001',0,4)`	Returns 04/01/2001 00:00:00.

Exempel 1 – Grundläggande exempel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller transaktioner mellan år 2020 och 2022 som läses in i en tabell som heter `Transactions`.
- Datumfältet har tillhandahållits i `MM/DD/YYYY`-formatet med systemvariabeln `DateFormat`.
- En föregående LOAD-sats som innehåller följande:
  - `yearstart()`-funktion som är inställd som `year_start`-fält.
  - `Timestamp()`-funktion som är inställd som `year_start_timestamp`-fält.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearstart(date) as year_start,
       timestamp(yearstart(date)) as year_start_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`
- `year_start`
- `year_start_timestamp`

Resultattabell

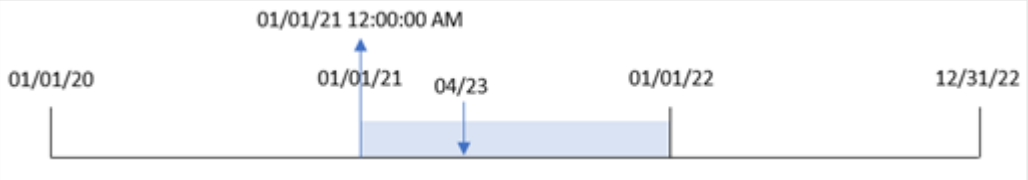
id	datum	year_start	year_start_timestamp
8188	01/13/2020	01/01/2020	1/1/2020 12:00:00 AM

id	datum	year_start	year_start_timestamp
8189	02/26/2020	01/01/2020	1/1/2020 12:00:00 AM
8190	03/27/2020	01/01/2020	1/1/2020 12:00:00 AM
8191	04/16/2020	01/01/2020	1/1/2020 12:00:00 AM
8192	05/21/2020	01/01/2020	1/1/2020 12:00:00 AM
8193	08/14/2020	01/01/2020	1/1/2020 12:00:00 AM
8194	10/07/2020	01/01/2020	1/1/2020 12:00:00 AM
8195	12/05/2020	01/01/2020	1/1/2020 12:00:00 AM
8196	01/22/2021	2021-01-01	2021-01-01 00:00:00
8197	02/03/2021	2021-01-01	2021-01-01 00:00:00
8198	03/17/2021	2021-01-01	2021-01-01 00:00:00
8199	04/23/2021	2021-01-01	2021-01-01 00:00:00
8200	05/04/2021	2021-01-01	2021-01-01 00:00:00
8201	06/30/2021	2021-01-01	2021-01-01 00:00:00
8202	07/26/2021	2021-01-01	2021-01-01 00:00:00
8203	12/27/2021	2021-01-01	2021-01-01 00:00:00
8204	06/06/2022	01/01/2022	1/1/2022 12:00:00 AM
8205	07/18/2022	01/01/2022	1/1/2022 12:00:00 AM
8206	11/14/2022	01/01/2022	1/1/2022 12:00:00 AM
8207	12/12/2022	01/01/2022	1/1/2022 12:00:00 AM

`year_start`-fältet skapades i den föregående load-satsen genom att använda `yearstart()`-funktionen och skicka datumfältet som funktionens argument.

`yearstart()`-funktionen identifierar initialt vilket år datumvärdet infaller och returnerar en tidsstämpel för den första millisekunden av det året.

Diagram över `yearstart()`-funktionen och transaktionen 8199.



Transaktion 8199 ägde rum den 23 april 2021. `yearstart()`-funktionen returnerar den sista millisekunden av det året, vilket är den 1 januari 12:00:00 AM.

Exampel 2 – period_no

Laddningsskript och resultat

Översikt

Samma datauppsättning och scenario som det första exemplet används.

Men i det här exemplet är uppgiften att skapa ett fält, `previous_year_start`, som returnerar tidsstämpel för startdatum för året innan det år då en transaktion ägde rum.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearstart(date,-1) as previous_year_start,
       timestamp(yearstart(date,-1)) as previous_year_start_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

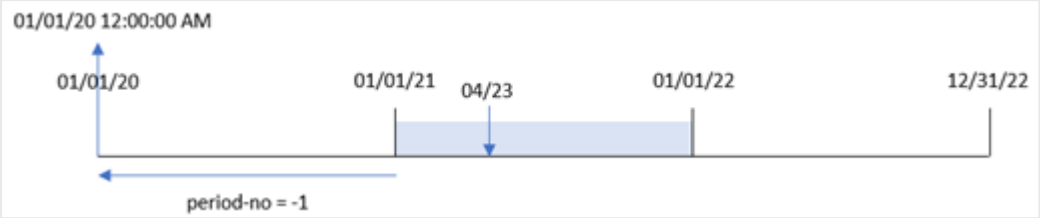
- id
- date
- previous_year_start
- previous_ year_start_timestamp

Resultattabell

id	datum	previous_year_start	previous_year_start_timestamp
8188	01/13/2020	01/01/2019	1/1/2019 12:00:00 AM
8189	02/26/2020	01/01/2019	1/1/2019 12:00:00 AM
8190	03/27/2020	01/01/2019	1/1/2019 12:00:00 AM
8191	04/16/2020	01/01/2019	1/1/2019 12:00:00 AM
8192	05/21/2020	01/01/2019	1/1/2019 12:00:00 AM
8193	08/14/2020	01/01/2019	1/1/2019 12:00:00 AM
8194	10/07/2020	01/01/2019	1/1/2019 12:00:00 AM
8195	12/05/2020	01/01/2019	1/1/2019 12:00:00 AM
8196	01/22/2021	01/01/2020	1/1/2020 12:00:00 AM
8197	02/03/2021	01/01/2020	1/1/2020 12:00:00 AM
8198	03/17/2021	01/01/2020	1/1/2020 12:00:00 AM
8199	04/23/2021	01/01/2020	1/1/2020 12:00:00 AM
8200	05/04/2021	01/01/2020	1/1/2020 12:00:00 AM
8201	06/30/2021	01/01/2020	1/1/2020 12:00:00 AM
8202	07/26/2021	01/01/2020	1/1/2020 12:00:00 AM
8203	12/27/2021	01/01/2020	1/1/2020 12:00:00 AM
8204	06/06/2022	2021-01-01	2021-01-01 00:00:00
8205	07/18/2022	2021-01-01	2021-01-01 00:00:00
8206	11/14/2022	2021-01-01	2021-01-01 00:00:00
8207	12/12/2022	2021-01-01	2021-01-01 00:00:00

I det här fallet, eftersom ett `period_no` på `-1` användes som förskjuningsargument i `yearstart()`-funktionen, identifierar funktionen först den månad då transaktionerna äger rum. Den tittar sedan ett år innan och identifierar den första millisekunden för det året.

Diagram för `yearstart()`-funktionen med en `period_no` på -1.



Transaktion 8199 ägde rum den 23 april 2021. `yearstart()`-funktionen returnerar den första millisekunden av det föregående året, vilket är den 1 januari 2020 kl 12:00:00 för `previous_year_start`-fältet

Exempel 3 – first_month_of_year

Laddningsskript och resultat

Översikt

Samma datauppsättning och scenario som det första exemplet används.

Men i det här exemplet är företagets policy att året ska börja den 1 april.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearstart(date,0,4) as year_start,
       timestamp(yearstart(date,0,4)) as year_start_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
```

```
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

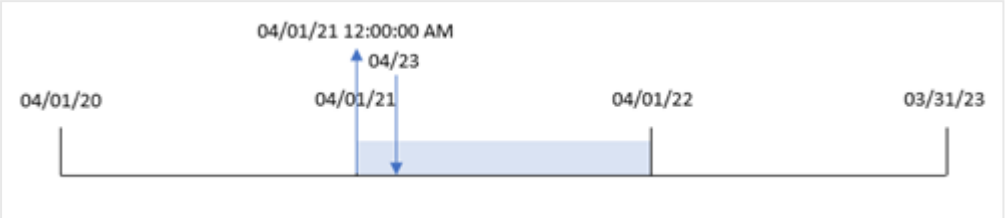
- id
- date
- year_start
- year_start_timestamp

Resultattabell

id	datum	year_start	year_start_timestamp
8188	01/13/2020	04/01/2019	2022-04-01 00:00:00
8189	02/26/2020	04/01/2019	4/1/2019 12:00:00 AM
8190	03/27/2020	04/01/2019	4/1/2019 12:00:00 AM
8191	04/16/2020	04/01/2020	4/1/2020 12:00:00 AM
8192	05/21/2020	04/01/2020	4/1/2020 12:00:00 AM
8193	08/14/2020	04/01/2020	4/1/2020 12:00:00 AM
8194	10/07/2020	04/01/2020	4/1/2020 12:00:00 AM
8195	12/05/2020	04/01/2020	4/1/2020 12:00:00 AM
8196	01/22/2021	04/01/2020	4/1/2020 12:00:00 AM
8197	02/03/2021	04/01/2020	4/1/2020 12:00:00 AM
8198	03/17/2021	04/01/2020	4/1/2020 12:00:00 AM
8199	04/23/2021	04/01/2021	2021-01-04 00:00:00
8200	05/04/2021	04/01/2021	2021-01-04 00:00:00
8201	06/30/2021	04/01/2021	2021-01-04 00:00:00
8202	07/26/2021	04/01/2021	2021-01-04 00:00:00
8203	12/27/2021	04/01/2021	2021-01-04 00:00:00
8204	06/06/2022	04/01/2022	3/1/2022 12:00:00 AM
8205	07/18/2022	04/01/2022	3/1/2022 12:00:00 AM
8206	11/14/2022	04/01/2022	3/1/2022 12:00:00 AM
8207	12/12/2022	04/01/2022	3/1/2022 12:00:00 AM

I det här fallet, eftersom `first_month_of_year`-argumentet för `4` används i `yearstart()`-funktionen, ställer det in årets första dag till 1 april och årets sista dag till 31 mars.

Diagram för `yearstart()`-funktionen med april inställt som den första månaden.



Transaktion 8199 ägde rum den 23 april 2021. Eftersom `yearstart()`-funktionen ställer in början av året till 1 april, returnerar den 31 mars 2022 som `year_start`-värdet för transaktionen.

Exempel 4 – Diagramobjektexempel

Laddningsskript och diagramuttryck

Översikt

Samma datauppsättning och scenario som det första exemplet används.

I det här exemplet är dock datauppsättningen oförändrad och har lästs in i programmet. Beräkningen som returnerar en tidsmarkör för startdatum för året då en transaktionen ägde rum skapas som ett mått i ett diagramobjekt för programmet.

Laddningsskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
```

```
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `id`
- `date`

För att beräkna vilket år en transaktion ägde rum skapar du följande mått:

- `=yearstart(date)`
- `=timestamp(yearstart(date))`

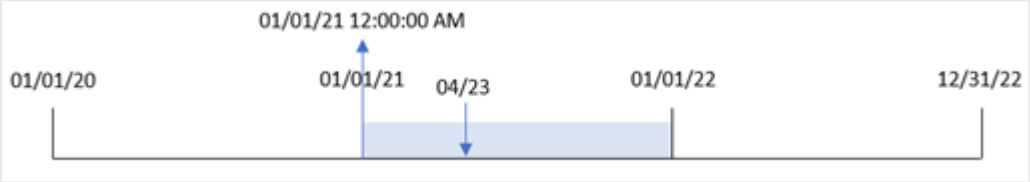
Resultattabell

id	datum	=yearstart(date)	=timestamp(yearstart(date))
8188	06/06/2022	01/01/2022	1/1/2022 12:00:00 AM
8189	07/18/2022	01/01/2022	1/1/2022 12:00:00 AM
8190	11/14/2022	01/01/2022	1/1/2022 12:00:00 AM
8191	12/12/2022	01/01/2022	1/1/2022 12:00:00 AM
8192	01/22/2021	2021-01-01	2021-01-01 00:00:00
8193	02/03/2021	2021-01-01	2021-01-01 00:00:00
8194	03/17/2021	2021-01-01	2021-01-01 00:00:00
8195	04/23/2021	2021-01-01	2021-01-01 00:00:00
8196	05/04/2021	2021-01-01	2021-01-01 00:00:00
8197	06/30/2021	2021-01-01	2021-01-01 00:00:00
8198	07/26/2021	2021-01-01	2021-01-01 00:00:00
8199	12/27/2021	2021-01-01	2021-01-01 00:00:00
8200	01/13/2020	01/01/2020	1/1/2020 12:00:00 AM
8201	02/26/2020	01/01/2020	1/1/2020 12:00:00 AM
8202	03/27/2020	01/01/2020	1/1/2020 12:00:00 AM
8203	04/16/2020	01/01/2020	1/1/2020 12:00:00 AM
8204	05/21/2020	01/01/2020	1/1/2020 12:00:00 AM
8205	08/14/2020	01/01/2020	1/1/2020 12:00:00 AM
8206	10/07/2020	01/01/2020	1/1/2020 12:00:00 AM
8207	12/05/2020	01/01/2020	1/1/2020 12:00:00 AM

`start_of_year`-måttet skapades i ett diagramobjektet genom att använda `yearstart()`-funktionen och skicka datumfältet som funktionens argument.

`yearstart()`-funktionen identifierar initialt vilket år datumvärdet infaller och returnerar en tidsstämpel för den första millisekunden av det året.

Diagram över `yearstart()`-funktionen och transaktionen 8199.



Transaktion 8199 ägde rum den 23 april 2021. `yearstart()`-funktionen returnerar den sista millisekunden av det året, vilket är den 1 januari 12:00:00 AM.

Exempel 5 – Scenario

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning läses in i en tabell som heter `Loans`. Tabellen innehåller följande fält:
  - Låne-ID:n.
  - Saldo i början av året.
  - Den enkla räntan som tas ut på varje lån per år.

Slutanvändaren vill ha ett diagramobjekt som visar, efter låne-id, den aktuella räntan som har ackumulerats för varje lån under året hittills.

Laddningsskript

```
Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `loan_id`
- `start_balance`

För att beräkna ackumulerad ränta skapar du följande mått:

```
=start_balance*(rate*(today(1)-yearstart(today(1)))/365)
```

Ange måttens **Nummerformatering** till **Pengar**.

Resultattabell

loan_id	start_balance	=start_balance*(rate*(today(1)-yearstart(today(1)))/365)
8188	$10000.00	$39.73
8189	$15000.00	$339.66
8190	$17500.00	$166.85
8191	$21000.00	$283.64
8192	$90000.00	$3003.29

Genom att använda dagens datum som enda argument returnerar `yearstart()`-funktionen slutdatumet för det aktuella året. Genom att subtrahera resultatet från det aktuella datumet returnerar uttrycket antalet dagar som har förflutit hittills i år.

Detta värde multipliceras sedan med räntan och divideras med 365 för att returnera den effektiva räntan för perioden. Den effektiva räntan för perioden multipliceras sedan med lånets utgångssaldo för att återbetala den upplupna räntan hittills i år.

yeartodate

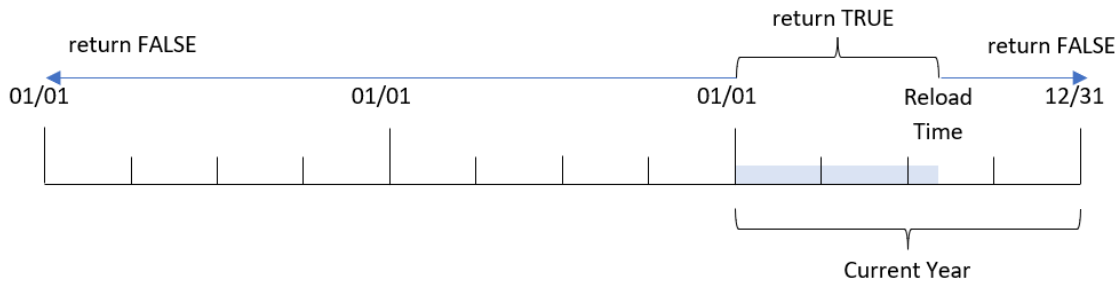
Denna funktion räknar ut om inmatningstidsmarkören hamnar inom året för datumet då skriptet senast lästes in och returnerar True om så är fallet, False om så inte är fallet.

```
YearToDate(timestamp[ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Returnerad datatyp: Booleansk

> *I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.*

Exempeldiagram för `yeartodate()`-funktionen



Om ingen av de valfria parametrarna används, omfattar definitionen alla datum inom ett kalenderår från den 1 januari t.o.m. datumet för senaste skriptexekvering.

Med andra ord används `yeartodate()`-funktionen, när den utlöses utan ytterligare parametrar, för att utvärdera en tidsmarkör och returnerar ett booleskt resultat baserat på huruvida datumet inträffade inom kalenderåret fram till och inklusive det datum då laddningen ägde rum.

Men det är också möjligt att ersätta årets startdatum med `firstmonth`-argumentet, och att göra jämförelser med föregående eller efterföljande år med `yearoffset`-argumentet.

Slutligen har `yeartodate()`-funktionen en parameter för att ställa in `todaydate` för historiska datauppsättningar. Den kommer då i stället att jämföra tidsmarkörer för kalenderåret fram till och med det datum som anges i `todaydate`-argumentet.

Argument

Argument	Beskrivning
timestamp	Tidsmarkören som ska utvärderas, exempelvis 2012-10-12.
yearoffset	Genom att ange en **yearoffset**, **yeartodate** returneras True för samma period ett annat år. Ett negativt värde på **yearoffset** indikerar ett tidigare år, ett positivt värde ett framtida år. Det senaste året uppnås genom att ange yearoffset = -1. Om det utelämnas, antas 0.
firstmonth	Genom att ange en **firstmonth** mellan 1 och 12 (1 om parametern utelämnas), kan årets början flyttas till första dagen på valfri månad. Om du exempelvis vill börja ett budgetår den 1 maj, sätter du **firstmonth** till 5. Ett värde på 1 anger att budgetåret börjar 1 januari och ett värde på 12 anger att budget året börjar 1 december.
todaydate	Genom att ange **todaydate** (tidsmarkör för senaste skriptexekveringen om parametern utelämnas), kan du flytta dagen som används som periodens övre gräns.

Användning

Funktionen `yeartodate()` returnerar ett booleskt resultat. Vanligtvis kommer denna typ av funktion att användas som ett villkor i ett if-uttryck. Detta returnerar en aggregering eller beräkning beroende på om det utvärderade datumet inträffade i året fram till och med det senaste laddningsdatumet för programmet.

Till exempel kan YearToDate()-funktionen användas för att identifiera all utrustning som har tillverkats hittills under det nuvarande året.

I följande exempel antas att senaste laddningstillfället är 2011-11-18.

Exempel på funktioner

Exempel	Resultat
`yeartodate( '11/18/2010')`	returnerar False
`yeartodate( '02/01/2011')`	returnerar True
`yeartodate( '11/18/2011')`	returnerar True
`yeartodate( '11/19/2011')`	returnerar False
`yeartodate( '11/19/2011', 0, 1, '12/31/2011')`	returnerar True
`yeartodate( '11/18/2010', -1)`	returnerar True
`yeartodate( '11/18/2011', -1)`	returnerar False
`yeartodate( '04/30/2011', 0, 5)`	returnerar False
`yeartodate( '05/01/2011', 0, 5)`	returnerar True

Lokala inställningar

Om inget annat anges använder exemplen i detta ämne följande datumformat: MM/DD/YYYY. Datumformatet anges i `SET DateFormat`-satsen i datainläsningsskriptet. Förvald datumformatering kan vara annorlunda i ditt system, på grund av dina regionala inställningar och andra faktorer. Du kan ändra formaten i exemplen nedan så att det passar dina krav. Eller så kan du ändra formaten i ditt laddningsskript så att de matchar dessa exempel.

Standardregionalinställningar i program baseras på de regionala systeminställningarna för datorn eller servern där Qlik Sense är installerad. Om Qlik Senseservern du ansluter till är inställd på Sverige så kommer Dataladdningsredigeraren att använda svenska regionala inställningar för datum, tid och valuta. Dessa regionala formatinställningar är inte relaterade till språket som visas i Qlik Sense användargränssnittet. Qlik Sense kommer att visas på samma språk som webbläsaren du använder.

Exempel 1 – Grundläggande exempel

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner mellan 2020 och 2022 som laddas i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (MM/DD/YYYY) i `DateFormat`-systemvariabeln.
- Skapandet av ett fält `year_to_date` som bestämmer vilka transaktioner som ägde rum under kalenderåret fram till datumet för den senaste omladdningen.

När det här skrivs är datumet 26 april 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        yeartodate(date) as year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

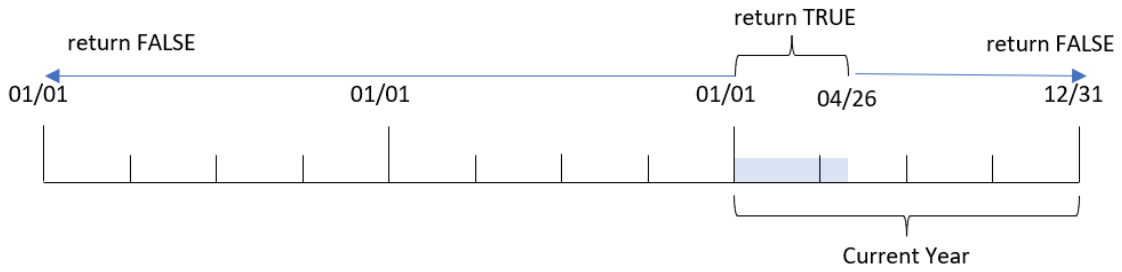
- `date`
- `year_to_date`

Resultattabell

datum	year_to_date
01/10/2020	0
02/28/2020	0

datum	year_to_date
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

Diagram för `yeartodate()`-funktionen, grundläggande exempel



`year_to_date`-fältet skapas i den föregående load-satsen genom att använda `yeartodate()`-funktionen och skicka `date`-fältet som funktionens argument.

Eftersom inga ytterligare parametrar skickas till funktionenen identifierar `yeartodate()`-funktionen först laddningsdatumet och därmed gränserna för det aktuella kalenderåret (med början 1 januari) som returnerar ett booleskt resultat på `TRUE`.

Därför kommer alla transaktioner som sker mellan 1 januari och 26 april, laddningsdatumet, att returnera ett booleskt resultat som är TRUE. Alla transaktioner som inträffar innan 2022 börjar returnerar det booleska värdet FALSE.

Exempel 2 – yearoffset

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Ett fält `two_years_prior` skapas, som bestämmer vilka transaktioner som ägde rum två hela år innan nuvarande kalenderår.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        yeartodate(date,-2) as two_years_prior
        ;
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

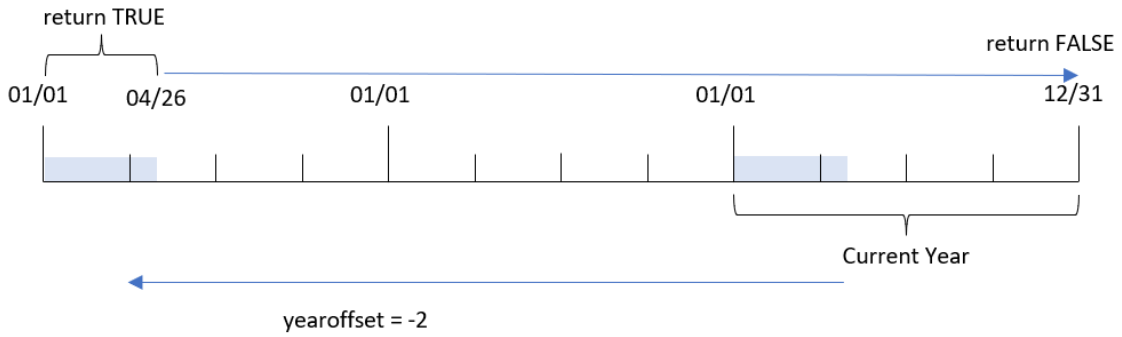
- `date`
- `two_years_prior`

Resultattabell

datum	two_years_prior
01/10/2020	-1
02/28/2020	-1
04/09/2020	-1
04/16/2020	-1
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	0
02/26/2022	0
03/07/2022	0
03/11/2022	0

Genom att använda -2 som `yearoffset`-argument i `yeartodate()`-funktionen skiftar funktionerna gränserna för segmentet i jämförelsekalenderåret med två hela år. Initialt motsvarar årssegmentet 1 januari till 26 april 2022. `yearoffset`-argumentet förskjuter sedan det här segmentet till två år tidigare. Datumgränserna kommer då att inträffa mellan 1 januari och 26 april 2020.

Diagram med exempel på `yeartodate()`-funktionen, yearoffset



Därför kommer alla transaktioner som sker mellan 1 januari och 26 april 2020 returnera ett booleskt resultat som är `TRUE`. Alla transaktioner som inträffar före eller efter det här segmentet returnerar `FALSE`.

Exempel 3 – firstmonth

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Ett fält `year_to_date` skapas som bestämmer vilka transaktioner som ägde rum samma kalenderår fram datumet för den senaste laddningen.

I det här exempel sätter vi budgetårets början till 1 juli.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        yeartodate(date,0,7) as year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
```

```
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `year_to_date`

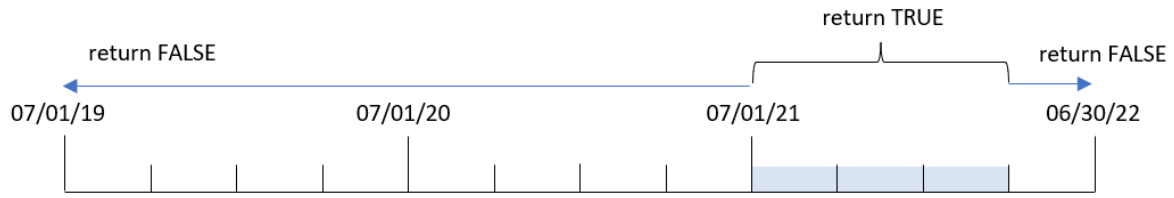
Resultattabell

datum	year_to_date
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0

datum	year_to_date
07/26/2021	-1
12/27/2021	-1
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

Eftersom `firstmonth`-argumentet på 7 används i `yeartodate()`-funktionen, ställer den in årets första dag till 1 juli och årets sista dag till 30 juni.

Diagram med exempel på `yeartodate()`-funktionen, firstmonth



Därför kommer alla transaktioner som sker mellan 1 juli och 26 april, laddningsdatumet, att returnera ett booleskt resultat som är `TRUE`. Alla transaktioner som inträffar före 1 juli 2021 returnerar det booleska värdet `FALSE`.

Exempel 4 – todaydate

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Samma datauppsättning och scenario som i det första exemplet.
- Ett fält `year_to_date` skapas som bestämmer vilka transaktioner som ägde rum samma kalenderår fram datumet för den senaste laddningen.

Men i det här exemplet behöver vi identifiera alla transaktioner som ägde rum under kalenderåret fram till och med 1 mars 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
```

```
Transactions:
    Load
        *,
        yeartodate(date, 0, 1, '03/01/2022') as year_to_date
;
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `date`
- `year_to_date`

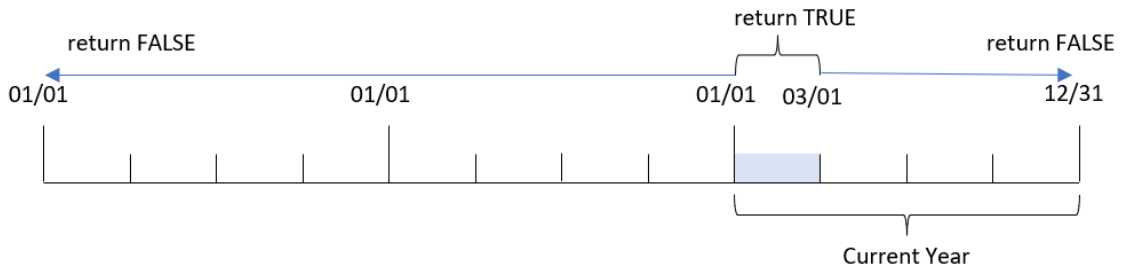
Resultattabell

datum	year_to_date
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0

datum	year_to_date
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	0
03/11/2022	0

Eftersom `todaydate`-argumentet 2022-03-01 används i `yeartodate()`-funktionen i det här fallet sätter den slutgränsen för det jämförande kalenderårssegmentet till 1 mars 2022. Det är mycket viktigt att `firstmonth`-parametern (mellan 1 och 12) tillhandahålls; i annat fall returnerar funktionen resultatet null.

Diagram av `yeartodate()`-funktionen med exempel som använder todaydate-argument



Därför kommer alla transaktioner som sker mellan 1 januari 2022 och 1 mars 2022, `todaydate`-parametern, att returnera det booleska resultatet `TRUE`. Alla transaktioner som inträffar före 1 januari 2022 eller efter 1 mars 2022 returnerar det booleska värdet `FALSE`.

Exempel 5 – exempel på diagramobjekt

Laddningsskript och diagramuttryck

Översikt

Öppna Dataladdningsredigeraren och lägg till laddningsskriptet nedan i en ny flik.

Laddningsskriptet innehåller samma datauppsättning och scenario som i det första exemplet.

I det här exemplet har dock den oförändrade datauppsättningen laddats till programmet. Beräkningen som avgör vilka transaktioner som har skett under kalenderåret fram till datumet för den senaste laddningen skapas som ett mått i ett diagramobjekt i programmet.

Laddningsskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension: `date`.

Lägg till följande mått:

```
=yeartodate(date)
```

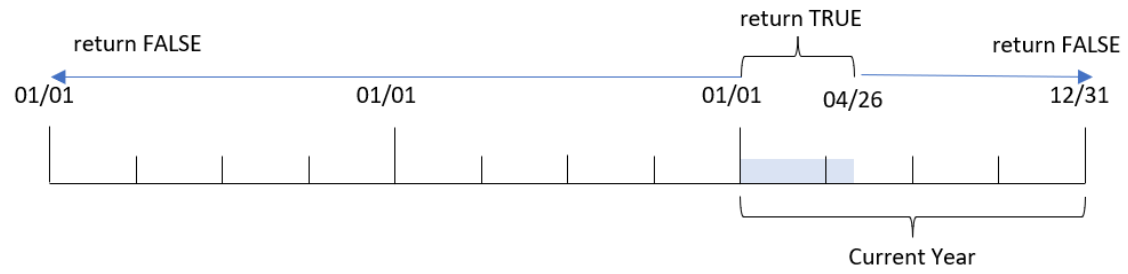
Resultattabell

datum	=yeartodate(date)
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

`year_to_date`-måttet skapas i diagramobjektet med hjälp av `yeartodate()`-funktionen och skickar `date`-fältet som funktionens argument.

Eftersom inga ytterligare parametrar skickas till funktionenen identifierar `yeartodate()`-funktionen först laddningsdatumet och därmed gränserna för det aktuella kalenderåret (med början 1 januari) som returnerar ett booleskt resultat på `TRUE`.

Diagram med exempel på `yeartodate()`-funktionen som använder diagramobjekt



Alla transaktioner som sker mellan 1 januari och 26 april, laddningsdatumet, att returnera ett booleskt resultat som är `TRUE`. Alla transaktioner som inträffar innan 2022 börjar returnerar det booleska värdet `FALSE`.

Exempel 6 – scenario

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som innehåller en uppsättning transaktioner mellan 2020 och 2022, som laddas i en tabell som heter `Transactions`.
- Datumfältet tillhandahålls i formatet (MM/DD/YYYY) i `DateFormat`-systemvariabeln.

Användaren vill ha ett KPI-objekt som presenterar den totala försäljningen för motsvarande period under 2021 som aktuellt år och vid senaste laddningstiden.

När det här skrivs är datumet 16 juni 2022.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
```

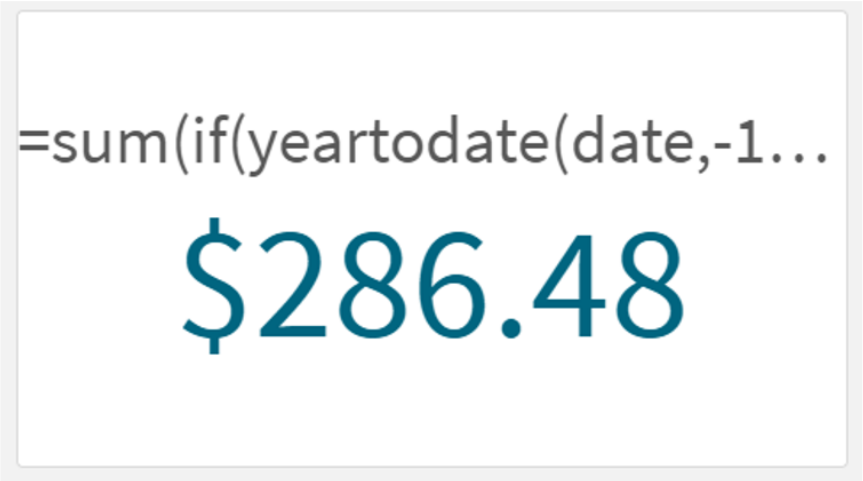
```
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Resultat

Gör följande:

1. Skapa ett KPI-objekt.
2. Skapa följande aggregeringsmått för att beräkna total försäljning:
   `=sum(if(yeartodate(date,-1),amount,0))`
3. Ange måttens **Nummerformatering** till **Pengar**.

KPI `yeartodate()`-diagram för 2021



Funktionen `yeartodate()` returnerar ett booleskt värde vid utvärdering av datum för varje transaktions-ID. Eftersom laddningen skedde 16 juni 2022 segmenterar `yeartodate`-funktionen årsperioden till mellan 2022-01-01 och 2022-06-16. Men eftersom ett `period_no`-värde på -1 användes i funktionen skiftas de här gränserna sedan till föregående år. Därför returnerar `yeartodate()`-funktionen ett booleskt värde på `TRUE` och summerar beloppet för alla transaktioner mellan 2021-01-01 och 2021-06-16.

8.8 Exponentiella och logaritmiska funktioner

Den här delen beskriver funktioner som är relaterade till exponential- och logaritmberäkningar. Alla funktioner kan användas både i dataladdningsskriptet och diagramuttryck.

I funktionerna nedan är parametrarna uttryck där **x** och **y** ska tolkas som reella tal.

exp

Den naturliga exponentiella funktionen, e^x, med den naturliga logaritmen **e** som bas. Resultatet blir ett positivt tal.

```
exp(x )
```

Exempel och resultat:

`exp(3)` returnerar 20,085.

log

Den naturliga logaritmen av **x**. Funktionen är bara definierad om **x**> 0. Resulterar i ett tal.

```
log(x )
```

Exempel och resultat:

`log(3)` returnerar 1,0986.

log10

Den vanliga logaritmen (bas 10) av **x**. Funktionen är bara definierad om **x**> 0. Resulterar i ett tal.

```
log10(x )
```

Exempel och resultat:

`log10(3)` returnerar 0,4771.

pow

Returnerar **x** upphöjt till **y**. Resulterar i ett tal.

```
pow(x,y )
```

Exempel och resultat:

`pow(3, 3)` returnerar 27.

sqr

x i kvadrat (**x** upphöjt till 2). Resulterar i ett tal.

```
sqr (x )
```

Exempel och resultat:

`sqr(3)` returnerar 9.

sqrt
Kvadratroten av **x**. Funktionen är bara definierad om **x** >= 0. Resultatet blir ett positivt tal.

```
sqrt(x )
```

Exempel och resultat:

`sqrt(3)` returnerar 1,732.

8.9 Fältfunktioner

Dessa funktioner kan endast användas i diagramuttryck.

Fältfunktioner returnerar antingen heltal eller strängar som identifierar olika aspekter av fälturval.

Räknefunktioner

GetAlternativeCount
GetAlternativeCount() används för att hitta antalet alternativa (ljusgrå) värden i det identifierade fältet.

```
GetAlternativeCount - diagramfunktion (field_name)
```

GetExcludedCount
GetExcludedCount() finner antalet uteslutna distinkta värden i det identifierade fältet. Enbart uteslutna (mörkgrå) fält räknas. Alternativa (ljusgrå) och valda uteslutna (mörkgrå med markering) värden räknas inte.

```
GetExcludedCount - diagramfunktion(field_name)
```

GetPossibleCount
GetPossibleCount() används för att hitta antalet möjliga värden i det identifierade fältet. Om det identifierade fältet innehåller urval räknas de markerade (gröna) fälten. Värden som är associerade på annat sätt (vita värden) räknas.

```
GetPossibleCount - diagramfunktion(field_name)
```

GetSelectedCount
GetSelectedCount() finner antalet valda (gröna) värden i ett fält.

```
GetSelectedCount - diagramfunktion (field_name [, include_excluded])
```

GetStateCounts
Diagramfunktionen **GetStateCounts()** används för att beräkna det totala antalet unika värden som matchar de angivna urvalstillstånden.

```
GetStateCounts - diagramfunktion (field_name, state_name [, state_
type1,...state_typeN])
```

Fält- och urvalsfunktioner

GetCurrentSelections

GetCurrentSelections() returnerar en lista över de aktuella valen i applikationen. Om valen istället görs med hjälp av en sökträng i en sökruta, **GetCurrentSelections()** returnerar söksträngen.

```
GetCurrentSelections - diagramfunktion([record_sep [,tag_sep [,value_sep
[,max_values]]]])
```

GetFieldSelections

GetFieldSelections() returnerar en **sträng** med de aktuella urvalen i ett fält.

```
GetFieldSelections - diagramfunktion ( field_name [, value_sep [, max_
values]])
```

GetObjectDimension

GetObjectDimension() returnerar namnet på dimensionen. **Index** är ett valfritt heltal som anger dimensionen som ska returneras.

```
GetObjectDimension - diagramfunktion ([index])
```

GetObjectField

GetObjectField() returnerar -fältuttrycket för dimensionen. **Index** är ett valfritt heltal som anger den dimension som ska returneras.

```
GetObjectField - diagramfunktion ([index])
```

GetObjectMeasure

GetObjectMeasure() returnerar namnet på måttet. **Index** är ett valfritt heltal som anger måttet som ska returneras.

```
GetObjectMeasure - diagramfunktion ([index])
```

GetAlternativeCount - diagramfunktion

GetAlternativeCount() används för att hitta antalet alternativa (ljusgrå) värden i det identifierade fältet.

Syntax:

```
GetAlternativeCount (field_name)
```

Returnerad datatyp: heltal

> *Färgerna som används i urvalsfältet och för varje urvaltillstånd kan ändras med ett anpassat tema. Om du arbetar med en applikation som använder ett anpassat tema, kanske du märker att dina val inte visas med de färger som beskrivs i det här hjälpämnet.*

Argument

Argument	Beskrivning
field_name	Fältet som innehåller dataintervallet som ska mätas.

I följande tabell listas andra funktioner som är relaterade till denna funktion.

Relaterade funktioner

Funktion	Interaktion
GetStateCounts - diagramfunktion (page 1342)	Med hjälp av **GetStateCounts()** kan du kombinera beräkningen av följande antal med ett enda funktionsanrop: • Antal valda inkluderade värden. • Antal möjliga värden. • Antal alternativa värden. • Antal uteslutna värden, exklusive alternativa och valda uteslutna värden. • Antal valda uteslutna värden.
GetSelectedCount - diagramfunktion (page 1338)	Returnerar antalet valda inkluderade värden.
GetPossibleCount - diagramfunktion (page 1334)	Returnerar antalet möjliga värden.
GetAlternativeCount - diagramfunktion (page 1305)	Returnerar antalet uteslutna värden, exklusive alternativa och valda uteslutna värden.

Exempel: diagramuttryck

Exempel	Resultat
`GetAlternativeCount ([First name])`	Returnerar `0` om inget val har gjorts i fältet `First Name`. Om val har gjorts är det värde som returneras det totala antalet distinkta tillgängliga värden minus antalet valda värden.

Exempel – Grundprinciper för GetAlternativeCount

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:

- FirstName
- LastName
- Initials
- HasCellphone

Laddningsskript

```
Example:
LOAD * inline [
FirstName|LastName|Initials|HasCellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny filterruta och lägg till detta fält som en dimension:

- FirstName

Lägg till ett KPI och skapa sedan följande mått:

- =GetAlternativeCount (FirstName), för att hitta antalet alternativa värden i fältet FirstName.

KPI:n med funktionen GetAlternativeCount visas med ett värde på 0. Om du väljer Jane i filterrutan FirstName returnerar KPI:n värdet 4 eftersom det finns fyra unika och uteslutna (grå) värden. Välj nu Mark, så att både Jane och Mark väljs. KPI-värdet för måttet GetAlternativeCount returnerar 3 eftersom det finns 3 unika och uteslutna värden, John, Peter och Sue.

KPI-resultat som visar 3 unika och uteslutna värden efter att ha valt Jane och Mark från filterrutan FirstName

Du kan se hur de värden som returneras av funktionen `GetAlternativeCount` ändras när ytterligare val görs.

Exempel – scenario för GetAlternativeCount

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `CustomerTypeData`.
- Följande fält i datatabellen:
    - `CustomerID`
    - `CustomerType`

Laddningsskript

```
CustomerTypeData:
LOAD * INLINE [
CustomerID, CustomerType
C001, Retail
C002, Wholesale
C003, Distributor
C004, Retail
C005, Wholesale
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny filterruta och lägg till detta fält som en dimension:

- `CustomerType`

Lägg till ett KPI och skapa sedan följande mått:

- `=GetAlternativeCount (CustomerType)`, för att hitta antalet alternativa värden i fältet `CustomerType`. KPI:n med funktionen `GetAlternativeCount` visas med ett värde på `0` eftersom inga värden har valts i fältet `CustomerType`.

Välj `Retail` från filterrutan `CustomerType`. `GetAlternativeCount` returnerar värdet `2` eftersom det finns två unika och uteslutna (ljusgrå) värden i fältet `CustomerType`.

KPI-resultat som visar 2 unika och uteslutna värden efter att ha valt Retail från filterrutan CustomerType

Resultatet visar att även om det finns flera värden, visar funktionen `GetAlternativeCount` endast det unika antalet objekt som inte har valts.

Exempel – scenario för GetAlternativeCount med tillgängliga värden

Diagramuttryck

Översikt

Detta exempel bygger vidare på det föregående scenariot. Datauppsättningen innehåller en andra datatabell för att visa hur val i varje fält ändrar de resultat som returneras av funktionen `GetAlternativeCount`.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som laddas i följande 2 inlinedatatabeller och motsvarande fält:
  - `CustomerTypeData`: Innehåller fälten `CustomerID` och `CustomerType`.
  - `CustomerSegment`: Innehåller fälten `CustomerType` och `Segment`.

Laddningsskript

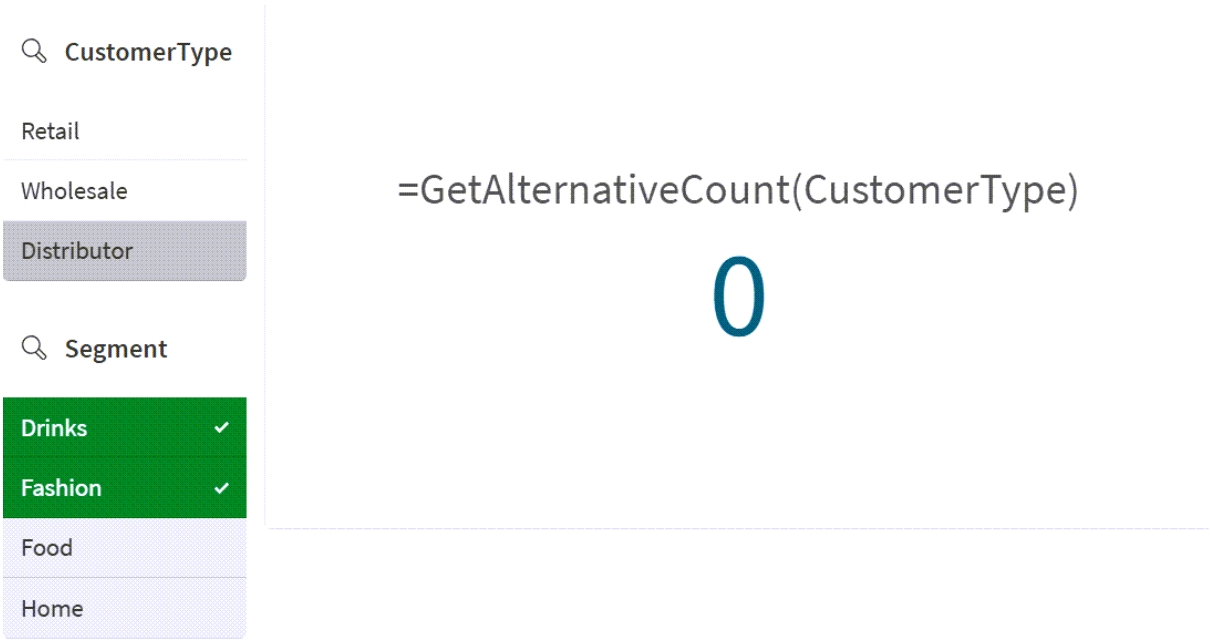
```
CustomerTypeData:
LOAD * INLINE [
CustomerID, CustomerType
C001, Retail
C002, Wholesale
C003, Distributor
C004, Retail
C005, Wholesale
];

CustomerSegment:
Load * Inline [
```

```
CustomerType, Segment
Retail, Fashion
Retail, Home
Wholesale, Fashion
Wholesale, Drinks
Distributor, Food
];
```

Resultat

1. Ladda data och öppna ett ark. Skapa en ny filterruta och lägg till dessa fält som dimensioner:
   - `CustomerType`
   - `Segment`
2. Lägg till ett KPI och skapa sedan följande mått:
   - `=GetAlternativeCount (CustomerType)`, för att hitta antalet alternativa värden i fältet `CustomerType`.
     KPI:n med funktionen `GetAlternativeCount` visas med ett värde på `0` eftersom inga värden har valts i fältet `CustomerType` eller `Segment`.
3. Välj `Drinks` och `Fashion` från `Segment`-filtret. KPI:n returnerar fortfarande `0` eftersom inget värde har valts i fältet `CustomerType`.

   KPI-resultat som visar 0 unika och uteslutna värden efter valen i `Segment`-filtret



4. Välj `Retail` i `CustomerType`-fältet och behåll urvalen i `Segment`-filtret.
   `GetAlternativeCount`-funktionen returnerar nu `1` eftersom det bara finns ett unikt och uteslutet (ljusgrå) värde i fältet `CustomerType`, `Wholesale`.

KPI-resultat som visar att det finns 1 unikt och uteslutet värde efter valen i filtren för `CustomerType` och `Segment`

Även om två värden, `Wholesale` och `Distributor`, inte är valda i fältet `CustomerType` (dessa visas som ljusgrå och mörkgrå), är endast ett värde, `Wholesale`, *tillgängligt* för val i `CustomerType` på grund av val i filtret `Segment`.

GetCurrentSelections - diagramfunktion

GetCurrentSelections() returnerar en lista över de aktuella valen i applikationen. Om valen istället görs med hjälp av en söksträng i en sökruta, **GetCurrentSelections()** returnerar söksträngen.

Om alternativ används måste du ange record_sep. Om du vill ange en ny rad ställer du in **record_sep** som **chr(13)&chr(10)**.

Om alla utom två värden, eller alla utom ett värde, är valda används formatet "NOT x,y" respektive "NOT y". Om du väljer alla värden och antalet värden blir högre än max_values, returneras texten ALL.

Syntax:

```
GetCurrentSelections ([record_sep [, tag_sep [, value_sep [, max_values [, state_name]]]]])
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
record_sep	Avgränsaren som avgränsar fältposterna. Standardinställningen är <CR><LF> vilket innebär en ny rad.
tag_sep	Avgränsaren som ska finnas mellan fältnamnets tagg och fältvärdena. Standard är ": ".
value_sep	Avgränsaren som avgränsar fältvärdena. ',' är standard.
max_values	Det maximala antalet fältvärden som ska visas individuellt i en lista. När ett större antal värden är valt används i stället formatet 'x av y värden'. 6 är standardvärde.
state_name	Namnet på ett alternativt tillstånd som har valts till den specifika visualiseringen. Om argumentet **state_name** används tas endast hänsyn till de urval som är förknippade med det angivna tillståndsnamnet.

Exempel: diagramuttryck

Exempel	Resultat
`GetCurrentSelections ()`	Returnerar (-) om inget val har gjorts i något av fälten. Om urval har gjorts är det värde som returneras fältnamnet och värdet, t.ex. `First Name : John`.

Exempel – grunder i GetCurrentSelections

Diagramuttryck

Översikt

I det här exemplet visas hur de värden som returneras av funktionen `GetCurrentSelections` ändras beroende på vilka val du gör.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `FirstName`
  - `LastName`
  - `Initials`
  - `HasCellphone`

Laddningsskript

```
Example:
LOAD * inline [
```

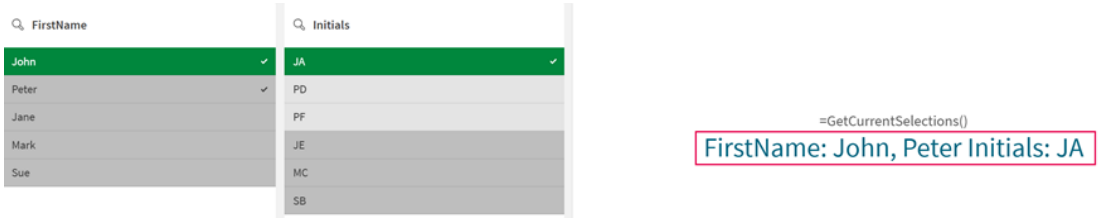
```
FirstName|LastName|Initials|HasCellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

Resultat

1. Ladda data och öppna ett ark. Skapa en ny filterruta och lägg till dessa fält som dimensioner:
   - `FirstName`
   - `Initials`
2. Lägg till ett KPI-objekt och skapa sedan följande mått:
   - `=GetCurrentSelections ()`, för att hitta aktuella urval.
     KPI:n returneras (-) eftersom inga värden har valts.

   > *I det här exemplet har utformningen av KPI:n justerats för att använda ett responsiv layoutbeteende som underlättar visning av resultatet. .*

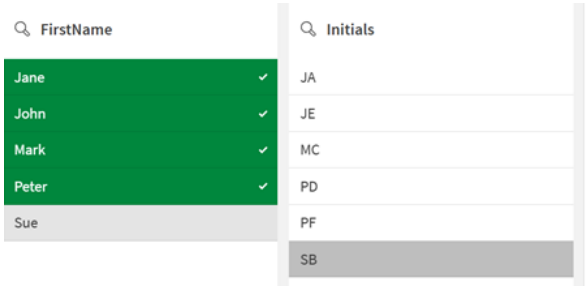
3. Klicka på **Redigera ark** för att växla till analysläge och välj sedan `John` i fältet `FirstName`. KPI returnerar följande: `FirstName: John`.
   - Välj nu `Peter`, så att både `John` och `Peter` väljs. KPI returnerar följande: `FirstName: John, Peter`.
   - Lägg till i urvalet genom att välja `JA` från fältet `Initials`. KPI returnerar följande: `First name: John, Peter Initials: JA`.



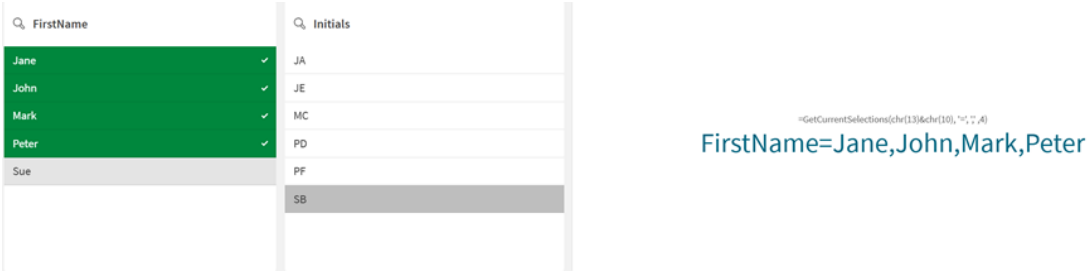
4. Töm urvalet och gör följande:
   - Välj `John` från fältet `FirstName` och `JA` från fältet `Initials`.
   - Redigera KPI-uttrycket till följande: `=GetCurrentSelections ( chr(13)&chr(10) , ' = ' )`, för att ändra taggseparatorn till ett likhetstecken (=).

   > *KPI stöder inte radbrytningar och lägger till ett mellanslag som postavgränsare.*

5. Töm urvalet och gör följande:
   - Markera alla namn utom `Sue` i fältet `FirstName`.
   - Redigera KPI-uttrycket till följande: `=GetCurrentSelections (chr(13)&chr(10), '=', ',' ,3)`, för att lägga till en taggavgränsare (=), en värdeavgränsare (,) och det maximala antalet värden som ska listas (3).



KPI:n returnerar värdet : `FirstName=NOT Sue` eftersom fyra namn valdes, men argumentet `max_values` listar bara tre. Om du ändrar argumentet `max_values` till `4` kommer alla fyra värdena att listas separat med samma urval.



Exempel – Anpassa rubriken på ett diagram för att visa urval

Diagramuttryck

Översikt

I det här exemplet visas hur du använder funktionen `GetCurrentSelections` för att skapa dynamiska rubriker för diagram baserat på användarens val.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `SalesID`
  - `Product`
  - `Year`
  - `Region`
  - `SalesAmount`

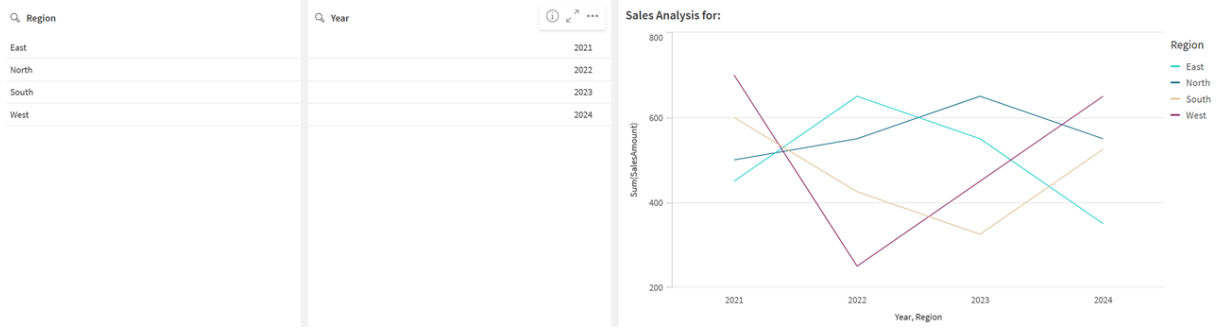
Laddningsskript

```
Example:
LOAD * INLINE [
SalesID, Product, Year, Region, SalesAmount
```

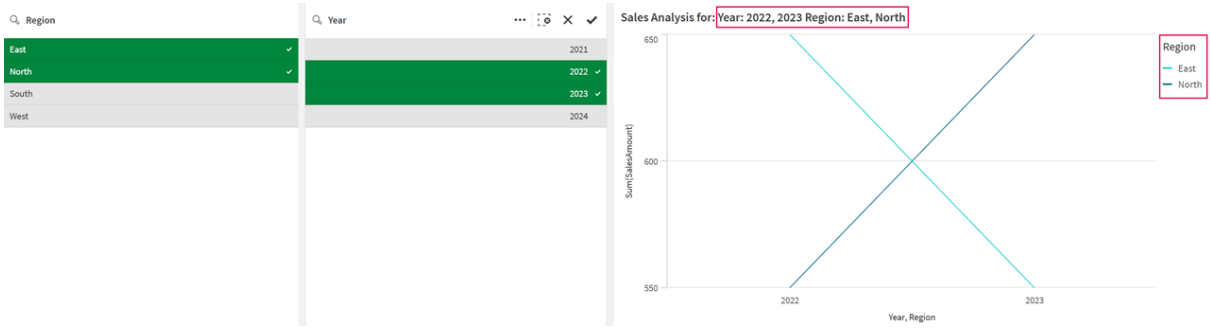
```
1, WidgetA, 2021, North, 500
2, WidgetB, 2021, South, 600
3, WidgetC, 2021, East, 450
4, WidgetA, 2021, West, 700
5, WidgetB, 2022, North, 550
6, WidgetC, 2022, South, 425
7, WidgetB, 2022, East, 650
8, WidgetC, 2022, West, 250
9, WidgetB, 2023, North, 650
10, WidgetC, 2023, South, 325
11, WidgetB, 2023, East, 550
12, WidgetC, 2023, West, 450
13, WidgetB, 2024, North, 550
14, WidgetC, 2024, South, 525
15, WidgetB, 2024, East, 350
16, WidgetC, 2024, West, 650
];
```

Resultat

1. Ladda data och öppna ett ark. Skapa en ny filterruta och lägg till dessa fält som dimensioner:
   - `Region`
   - `Year`
2. Skapa ett linjediagram och lägg till följande dimensioner:
   - `Year`
   - `Region`
3. Skapa följande mått:
   - `=Sum(SalesAmount)`, för att visa den totala försäljningen.
4. I egenskapspanelen för åtgärden, under **Utseende** > **Allmänt** > **Rubrik**, anger du följande uttryck: `='Sales Analysis for: ' & GetCurrentSelections().`



   Resultatet visar summan av försäljningen för alla regioner och år. Diagramrubriken visas som `Sales Analysis for:`.
5. Klicka på **Redigera ark** för att växla till analysläge och gör sedan följande val:
   - I fältet **Region** väljer du `East` och `North`.
   - I fältet **År** väljer du `2022` och `2023`.



Diagramrubriken ändras så att filtervalen för `Region` och `Year` visas.

GetExcludedCount - diagramfunktion

GetExcludedCount() finner antalet uteslutna distinkta värden i det identifierade fältet. Enbart uteslutna (mörkgrå) fält räknas. Alternativa (ljusgrå) och valda uteslutna (mörkgrå med markering) värden räknas inte.

Syntax:

```
GetExcludedCount (field_name)
```

Returnerad datatyp: heltal

> *Färgerna som används i urvalsfältet och för varje urvaltillstånd kan ändras med ett anpassat tema. Om du arbetar med en applikation som använder ett anpassat tema, kanske du märker att dina val inte visas med de färger som beskrivs i det här hjälpämnet.*

Argument

Argument	Beskrivning
field_name	Fältet som innehåller dataintervallet som ska mätas.

I följande tabell listas andra funktioner som är relaterade till denna funktion.

Relaterade funktioner

Funktion	Interaktion
GetStateCounts - diagramfunktion (page 1342)	Med hjälp av **GetStateCounts()** kan du kombinera beräkningen av följande antal med ett enda funktionsanrop: • Antal valda inkluderade värden. • Antal möjliga värden. • Antal alternativa värden. • Antal uteslutna värden, exklusive alternativa och valda uteslutna värden. • Antal valda uteslutna värden.
GetSelectedCount - diagramfunktion (page 1338)	Returnerar antalet valda inkluderade värden.
GetPossibleCount - diagramfunktion (page 1334)	Returnerar antalet möjliga värden.
GetAlternativeCount - diagramfunktion (page 1305)	Returnerar antalet alternativa värden.

Exempel: diagramuttryck

Exempel	Resultat
`GetExcludedCount( Initials )`	Returnerar `0` om inget val har gjorts. Om ett urval har gjorts är det värde som returneras antalet uteslutna värden i `Initials` med mörkgrå färg.

Exempel – grunderna i GetExcludedCount

Diagramuttryck

Översikt

Det här exemplet visar hur värden som returneras av funktionen `GetExcludedCount` ändras när ytterligare val görs.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `FirstName`
  - `LastName`
  - `Initials`
  - `HasCellphone`

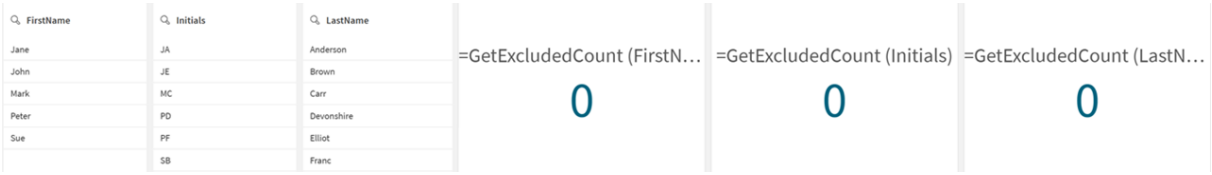
Laddningsskript

```
Example:
LOAD * inline [
FirstName|LastName|Initials|HasCellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

Resultat

1. Ladda data och öppna ett ark. Skapa en ny filterruta och lägg till dessa fält som dimensioner:
   - `FirstName`
   - `LastName`
   - `Initials`
2. Skapa tre KPI-objekt och lägg till följande uttryck som mått, ett per KPI-objekt:
   a. `=GetExcludedCount(FirstName)`, för att räkna antalet uteslutna värden i fältet `FirstName`.
   b. `=GetExcludedCount(Initials)`, för att räkna antalet uteslutna värden i fältet `Initials`.
   c. `=GetExcludedCount(LastName)`, för att räkna antalet uteslutna värden i fältet `LastName`.

   Arket kommer att se ut som följande bild eftersom inga värden har valts. Funktionen `GetExcludedCount` returnerar `0` för varje KPI.



3. Klicka på **Redigera ark** för att växla till analysläge och gör sedan följande val i turordning:

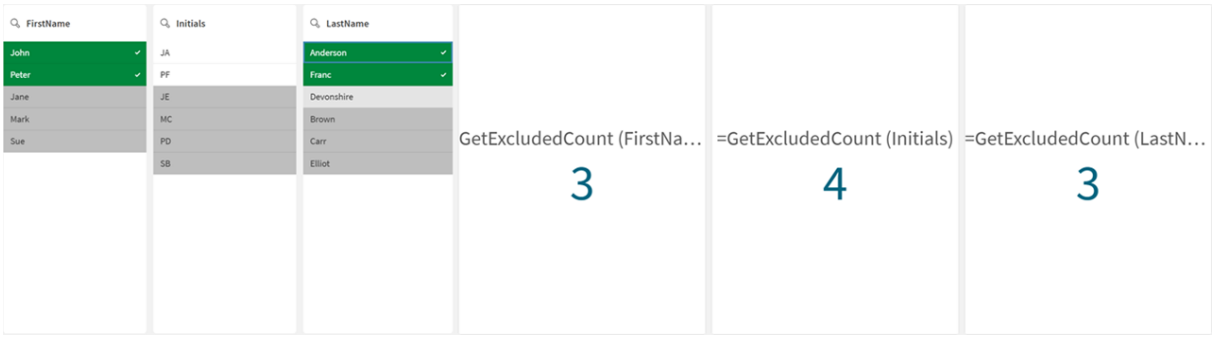
Filtrera urval och resultat i analysläget

Val av filterruta	KPI-resultat
Välj `John` från `FirstName`-filtret.	• `GetExcludedCount (Initials)` returnerar ett värde på `5` eftersom det finns `5` uteslutna värden i `Initials` med mörkgrå färg. Värdet `JA` förblir vitt eftersom det är kopplat till urvalet `John` i `FirstName`. • `GetExcludedCount (LastName)` returnerar ett värde på `5` eftersom det finns `5` uteslutna värden i `LastName` med mörkgrå färg. Värdet `Anderson` förblir vitt eftersom det är kopplat till urvalet `John` i `FirstName`.
Välj `John` och `Peter` från `FirstName`-	• `GetExcludedCount (Initials)` returnerar ett värde på `3` eftersom det finns `3` uteslutna värden i `Initials` med mörkgrå färg. • `GetExcludedCount (LastName)` returnerar ett värde på `3` eftersom det

Val av filterruta	KPI-resultat
filtret.	finns 3 uteslutna värden i LastName med mörkgrå färg.
När du har valt John och Peter väljer du Franc i filtret LastName.	• GetExcludedCount (FirstName) returnerar ett värde på 3 eftersom det finns 3 utesluten värden i FirstName med mörkgrå färg och ingen markering för val. Funktionen The GetExcludedCount räknar endast uteslutna värden. Alternativa och valda uteslutna värden ingår inte i antalsräkningen.John är ett utvalt, uteslutet värde. • GetExcludedCount (Initials) returnerar ett värde på 5 eftersom det finns 5 uteslutna värden i Initials med mörkgrå färg. • GetExcludedCount (LastName) returnerar ett värde på 3 eftersom det finns 3 uteslutna värden i LastName med mörkgrå färg. De 2 ljusgrå färgvärdena, Anderson och Devonshire, är alternativa värden.
Behåll de tidigare valen och välj sedan Anderson i filtret LastName.	• GetExcludedCount (FirstName) returnerar ett värde på 3 eftersom det finns 3 uteslutna värden i FirstName med mörkgrå färg. • GetExcludedCount (Initials) returnerar ett värde på 4 eftersom det finns 4 uteslutna värden i Initials med mörkgrå färg. • GetExcludedCount (LastName) returnerar ett värde på 3 eftersom det finns 3 uteslutna värden i LastName med mörkgrå färg.

Du kan se det slutliga resultatet av valen i filterrutan på följande bild.

KPI-resultat av valet av förnamn, John och Peter, och efternamn, Franc och Anderson



Exempel – GetExcludedCount-scenario

Diagramuttryck

Översikt

En försäljningschef använder funktionen GetExcludedCount för att skapa ett KPI-diagram för att analysera sålda produkter per säljare.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Salesperson`
  - `Product`
  - `SalesAmount`

Laddningsskript

```
Example:
LOAD * INLINE [
Salesperson, Product, SalesAmount
John, Laptops, 1000
Alice, Smartphones, 800
Bob, Laptops, 1200
Carol, Tablets, 700
Dave, Tablets, 900
];
```

Resultat

1. Ladda data och öppna ett ark. Skapa en ny filterruta och lägg till dessa fält som dimensioner:
   - `Salesperson`
   - `Product`
2. Skapa ett nytt KPI-objekt och lägg till följande mått:
   - `=GetExcludedCount(Product)`, för att hitta antalet uteslutna värden i fältet `Product`.
3. Klicka på **Redigera ark** för att växla till analysläge och välj sedan `Alice` i `Salesperson`-filtret. Funktionen `GetExcludedCount` returnerar värdet `2` eftersom det finns `2` unika och uteslutna (ljusgrå) värden i fältet `Product`.

Resultaten visar hur du kan använda funktionen `GetExcludedCount` för att analysera data. Säljaren `Alice` har inte sålt några produkter i 2 av de 3 produktgrupperna `Laptop`- och `Tablet`-produkter.

GetFieldSelections - diagramfunktion

GetFieldSelections() returnerar en **sträng** med de aktuella urvalen i ett fält.

Om alla utom två värden, eller alla utom ett värde, är valda används formatet ”NOT x,y” respektive ”NOT y”. Om du väljer alla värden och antalet värden blir högre än max_values, returneras texten ALL.

Syntax:

```
GetFieldSelections ( field_name [, value_sep [, max_values [, state_name]]])
```

Returnerad datatyp: sträng

Retursträngformat

Format	Beskrivning
'a, b, c'	Om antalet valda värden är max_values eller färre, returneras en sträng med en lista över de valda värdena. Värdena separeras med value_sep som avgränsare.
'NOT a, b, c'	Om antalet ej valda värden är max_values eller färre, returneras en sträng med en lista över de ej valda värdena med NOT som prefix. Värdena separeras med value_sep som avgränsare.
'x of y'	x = antalet valda värden y = det totala antalet värden Det här returneras när max_values < x < (y - max_values).
'ALL'	Returneras om alla värden har valts.
'-'	Returneras om inget värde har valts.
<search string>	Om du har valt med sökning returneras söksträngen.

Argument

Argument	Beskrivning
field_name	Fältet som innehåller dataintervallet som ska mätas.
value_sep	Avgränsaren som avgränsar fältvärdena. ',' är standard.
max_values	Det maximala antalet fältvärden som ska visas individuellt i en lista. När ett större antal värden är valt används i stället formatet 'x av y värden'. 6 är standardvärde.
state_name	Namnet på ett alternativt tillstånd som har valts till den specifika visualiseringen. Om argumentet **state_name** används tas endast hänsyn till de urval som är förknippade med det angivna tillståndsnamnet.

Exempel: diagramuttryck

Exempel	Resultat
`GetFieldSelections( FirstName )`	Returnerar "-" (NULL) om inget val har gjorts i fältet `FirstName`. Annars returneras värdet på urvalen, till exempel `John`.

Exempel – grunderna i GetFieldSelections

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `FirstName`
  - `LastName`
  - `Initials`
  - `HasCellphone`

Laddningsskript

```
Example:
LOAD * inline [
FirstName|LastName|Initials|HasCellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

Resultat

1. Ladda data och öppna ett ark. Skapa en ny filterruta och lägg till detta fält som en dimension:
   - `FirstName`
2. Skapa ett KPI-objekt och lägg till följande mått:
   - `=GetFieldSelections(FirstName)`, för att returnera värdena för det urval som gjorts i fältet `FirstName`.

   I det här exemplet har utformningen av KPI:n justerats för att använda ett responsiv layoutbeteende som underlättar visning av resultatet.
3. Klicka på **Redigera ark** för att växla till analysläge och utför sedan följande åtgärder i den ordning som de visas i tabellen:

Åtgärder och resultat i analysläget

Åtgärd	KPI-resultat
Välj `John` från `FirstName`-filtret.	Åtgärden returnerar värdet: `John`.

Åtgärd	KPI-resultat
Välj `John` och `Peter` från `FirstName`-filtret.	Åtgärden returnerar värdena: `John, Peter`.
Klicka på **Redigera ark** för att växla till redigeringsläge och redigera sedan KPI-måttets uttryck enligt följande: `=GetFieldSelections(FirstName,'; ')`	Åtgärden returnerar värdena med det nya `value_sep`-formatet: `John; Peter`.
Redigera KPI-måttuttrycket till följande: `=GetFieldSelections(FirstName,';', 2)`	Åtgärden returnerar värdena med det nya formatutrymmet `value_sep`: `John;Peter`.
Klicka på **Redigera ark** för att växla tillbaka till analysläge. Radera urvalen från filtret `FirstName` och välj sedan `John`, `Sue` och `Mark`.	Åtgärden returnerar `NOT Jane;Peter`, eftersom antalet val överstiger `max_value`-argumentet på `2`. Om ingen `max_value` anges kommer alla valda namn att visas: `John; Sue; Mark`.

Följande bild visar resultatet av den sista åtgärden i tabellen där antalet val överstiger argumentet `max_values`.

KPI-resultat vid val av förnamnen John, Sue, och Mark

I det här exemplet kan du se hur de värden som returneras av funktionen `GetFieldSelections` ändras när val görs.

Exempel – scenariot GetFieldSelections

Diagramuttryck

Översikt

Funktionen `GetFieldSelections` används för att markera kundnamn när de väljs.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:

- OrderID
- CustomerName
- Product
- Quantity

Laddningsskript

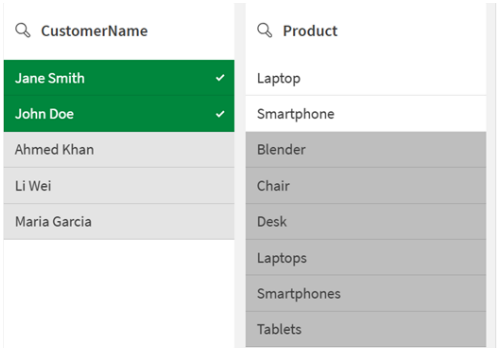
```
Example:
LOAD * INLINE [
OrderID, CustomerName, Product, Quantity
1, John Doe, Laptop, 2
2, Jane Smith, Smartphone, 1
3, Ahmed Khan, Desk, 3
4, Maria Garcia, Chair, 1
5, Li Wei, Blender, 2
];
```

Resultat

1. Ladda data och öppna ett ark. Skapa en ny filterruta och lägg till dessa fält som dimensioner:
   - CustomerName
   - Product
2. Skapa ett KPI-objekt och lägg till följande mått:
   - =GetFieldSelections(CustomerName).

   > *I det här exemplet har utformningen av KPI:n justerats för att använda ett responsiv layoutbeteende som underlättar visning av resultatet. .*

3. Klicka på **Redigera ark** för att växla till analysläge och välj sedan Jane Smith och John Doe i fältet CustomerName.
   Funktionen GetFieldSelections returnerar värdena: Jane Smith, John Doe.



Du kan se de kunder som valts i filterrutan i KPI:n.

GetObjectDimension - diagramfunktion

GetObjectDimension() returnerar namnet på dimensionen. **Index** är ett valfritt heltal som anger dimensionen som ska returneras.

Du kan inte använda den här funktionen i ett diagram på följande platser: titel, underrubrik, fotnot, referenslinjeuttryck och min/max-uttryck.

Du kan inte referera till namnet på en dimension eller ett mått i ett annat objekt med Object ID.

Syntax:

```
GetObjectDimension ([index])
```

Returnerad datatyp: Sträng

Exempel: diagramuttryck

Exempel	Resultat
`GetObjectDimension ()` `GetObjectDimension (0)`	Returnerar namnet på den första dimensionen i diagrammet.
`GetObjectDimension (1)`	Returnerar namnet på den andra dimensionen i diagrammet.

Exempel – grunderna i GetObjectDimension

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `TransactionDate`
  - `CustomerID`
  - `TransactionQuantity`

Laddningsskript

```
Example:
LOAD * INLINE [
TransactionDate, CustomerID, TransactionQuantity
2018/08/30, 049681, 13
2018/08/30, 203521, 6
```

```
2018/08/30, 203521, 21
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `TransactionDate`
- `CustomerID`
- `TransactionQuantity`

Skapa följande mått:

- `=GetObjectDimension ()`, för att hitta den första dimensionen i tabellen.
- `=GetObjectDimension (0)`, för att hitta den första dimensionen i tabellen.
- `=GetObjectDimension (1)`, för att hitta den andra dimensionen i tabellen.

Resultattabell

TransactionDate	CustomerID	TransactionQuantity	GetObjectDimension ()	GetObjectDimension (0)	GetObjectDimension (1)
2018/08/30	049681	13	TransactionDate	TransactionDate	CustomerID
2018/08/30	203521	6	TransactionDate	TransactionDate	CustomerID
2018/08/30	203521	21	TransactionDate	TransactionDate	CustomerID

När du tittar på resultaten kan du se hur funktionen `GetObjectDimension` returnerar namnet på den dimension som anges i funktionsparametrarna.

Exempel – GetObjectDimension avancerat scenario

Diagramuttryck

Översikt

I detta exempel används funktionen `GetObjectDimension` i kombination med filterval. Beräkningen av dimension och mått som visas i diagramobjektet ändras baserat på vilket filter du väljer.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i två inlinedatatabeller: `Sales` och `Filter`. Den första tabellen, `Sales`, innehåller försäljningsdata. Den andra tabellen, `Filter`, innehåller en lista med fältnamn som kommer att användas för att växla dimension. Med växlingsfunktionen kan du växla mellan dimensioner i ett diagram och anpassa den måttberäkning som visas beroende på vilken dimension

som valts.

- Följande fält i `Sales`-tabellen:
  - `Country`
  - `Salesperson`
  - `SalesValue`
- Följande fält i `Filter`-tabellen: `FilterField`

Laddningsskript

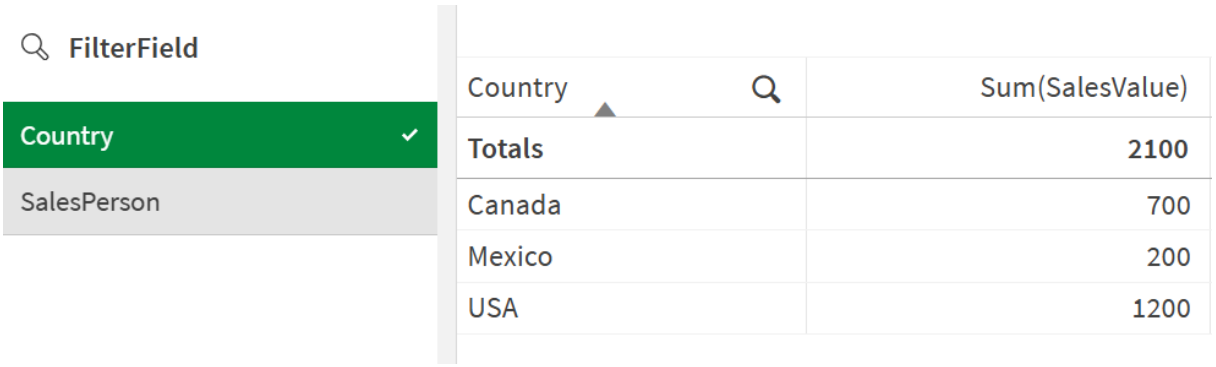
```
Sales:
LOAD * INLINE [
Country, SalesPerson, SalesValue
USA, John, 500
USA, Alice, 700
Canada, Bob, 300
Canada, Carol, 400
Mexico, Dave, 200
];

Filter:
LOAD * INLINE [
FilterField
Country
SalesPerson
];
```

Resultat

1. Ladda data och öppna ett ark. Skapa en ny filterruta och lägg till detta fält som en dimension:
   - `FilterField`
2. Högerklicka på **FilterField** under **Fält** i resurspanelen och välj **Fältinställningar**. Markera kryssrutan **Alltid ett värde valt** och klicka **Spara**.
3. Skapa en tabell och lägg till en dimension genom att lägga till följande uttryck:
   - `=$(= FilterField)`
   - Under **Etikett** anger du följande uttryck för att definiera kolumnnamnet för dimensionen: `='$(= FilterField)'`
4. Skapa följande mått:
   - `=If(GetObjectDimension() = 'Country', Sum(SalesValue), Avg(SalesValue))`, för att summera försäljningsvärdena om den valda dimensionen är `Country`, annars för att returnera genomsnittet av försäljningsvärdet när den valda dimensionen är `SalesPerson`.
   - Under `Label` anger du följande uttryck för att definiera kolumnnamnet för mätvärdet enligt filterurvalet: `=If(GetObjectDimension() = 'Country', 'Sum(SalesValue)', 'Avg(SalesValue)')`

I analysläget, när du växlar mellan `SalesPerson` och `Country` i `FieldFilter`-filtret, ändras tabellen så att den innehåller den valda dimensionen och motsvarande mätvärde för den dimensionen. Om du t.ex. väljer `Country` i filtret visas `Country` som dimension i den första kolumnen i tabellen. Mätvärdet använder sedan funktionen `GetObjectDimension`, likställer den med `Country` och returnerar `Sum(SalesValue)`.

Filterruta och tabellresultat för Country



Följande tabell visar resultatet när du väljer `Country` i filtret `FilterField`.

Resultattabell för Country

Country	Sum(SalesValue)
Totals	**2100**
Canada	700
Mexico	200
USA	1200

Följande tabell visar resultatet när du väljer `SalesPerson` i filtret `FilterField`.

Resultattabell för SalesPerson

SalesPerson	Avg(SalesValue)
Totals	**420**
Alice	700
Bob	300
Carol	400
Dave	200
John	500

GetObjectField - diagramfunktion

GetObjectField() returnerar -fältuttrycket för dimensionen. **Index** är ett valfritt heltal som anger den dimension som ska returneras.

Du kan inte använda den här funktionen i ett diagram på följande platser: titel, underrubrik, fotnot, referenslinjeuttryck och min/max-uttryck.

Du kan inte referera till namnet på en dimension eller ett mått i ett annat objekt med Object ID.

Syntax:

```
GetObjectField ([index])
```

Returnerad datatyp: sträng

Exempel: diagramuttryck

Exempel	Resultat
`GetObjectField ()` `GetObjectField (0)`	Returnerar fältuttrycket för den första dimensionen i diagrammet.
`GetObjectField (1)`	Returnerar fältuttrycket för den andra dimensionen i diagrammet.

Om du vill returnera namnet på ett **mått** använder du funktionen GetObjectMeasure istället.

Exempel – grunderna i GetObjectField

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `TransactionDate`
  - `CustomerID`
  - `TransactionQuantity`

Laddningsskript

```
Example:
LOAD * INLINE [
TransactionDate, CustomerID, TransactionQuantity
2018/08/30, 049681, 13
2018/08/30, 203521, 6
2018/08/30, 203521, 21
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `TransactionDate`
- `CustomerID`
- `TransactionQuantity`

Skapa följande mått:

- `=GetObjectField()`, för att hitta den första dimensionen i tabellen.
- `=GetObjectField(0)`, för att hitta den första dimensionen i tabellen.
- `=GetObjectField(1)`, för att hitta den andra dimensionen i tabellen.

Resultattabell

TransactionDate	CustomerID	TransactionQuantity	GetObjectField ()	GetObjectField (0)	GetObjectField (1)
2018/08/30	049681	13	TransactionDate	TransactionDate	CustomerID
2018/08/30	203521	6	TransactionDate	TransactionDate	CustomerID
2018/08/30	203521	21	TransactionDate	TransactionDate	CustomerID

När du tittar på resultaten kan du se hur funktionen `GetObjectField` returnerar namnet på det fält som anges i funktionsparametrarna.

Exempel – GetObjectField avancerat scenario

Diagramuttryck

Översikt

I detta exempel används funktionen `GetObjectField` i kombination med filterval. Beräkningen av dimension och mått som visas i diagramobjektet ändras baserat på vilket filter du väljer.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i två inlinedatatabeller: `Sales` och `Filter`. Den första tabellen, `Sales`, innehåller försäljningsdata. Den andra tabellen, `Filter`, innehåller en lista med fältnamn som kommer att användas för att växla dimension. Med växlingsfunktionen kan du växla mellan dimensioner i ett diagram och anpassa den måttberäkning som visas beroende på vilken dimension som valts.
- Följande fält i `Sales`-tabellen:

    - Country
    - Salesperson
    - SalesValue
- Följande fält i Filter-tabellen: FilterField

Laddningsskript

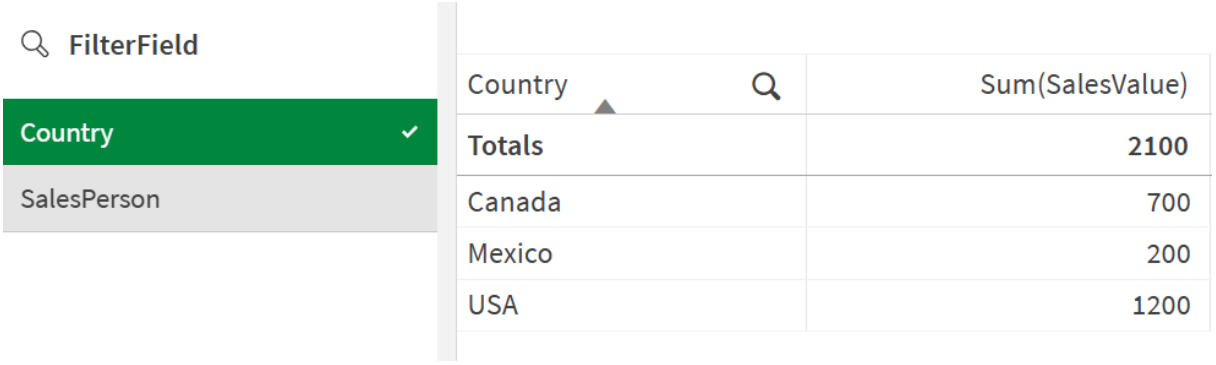
```
Sales:
LOAD * INLINE [
Country, SalesPerson, SalesValue
USA, John, 500
USA, Alice, 700
Canada, Bob, 300
Canada, Carol, 400
Mexico, Dave, 200
];

Filter:
LOAD * INLINE [
FilterField
Country
SalesPerson
];
```

Resultat

1. Ladda data och öppna ett ark. Skapa en ny filterruta och lägg till detta fält som en dimension:
    - FilterField
2. Högerklicka på **FilterField** under **Fält** i resurspanelen och välj **Fältinställningar**. Markera kryssrutan **Alltid ett värde valt** och klicka **Spara**.
3. Skapa en tabell och lägg till en dimension genom att lägga till följande uttryck:
    - =$(= FilterField)
    - Under **Etikett** anger du följande uttryck för att definiera kolumnnamnet för dimensionen: ='$(= FilterField)'
4. Skapa följande mått:
    - =If(GetObjectField() = 'Country', Sum(SalesValue), Avg(SalesValue)), för att summera försäljningsvärdena om den valda dimensionen är Country, annars för att returnera genomsnittet av försäljningsvärdet när den valda dimensionen är SalesPerson.
    - Under Label anger du följande uttryck för att definiera kolumnnamnet som kommer att visas för mätvärdet enligt filterurvalet: =If(GetObjectField() = 'Country', 'Sum(SalesValue)', 'Avg(SalesValue)') Om t.ex. Country väljs i filtret, kommer namnet på åtgärden att returnera värdet Sum(SalesValue).

I analysläget, när du växlar mellan SalesPerson och Country i FieldFilter-filtret, ändras tabellen så att den innehåller den valda dimensionen och motsvarande mätvärde för den dimensionen. Om du t.ex. väljer Country i filtret visas Country som dimension i den första kolumnen i tabellen. Mätvärdet använder sedan funktionen GetObjectField, likställer den med Country och returnerar Sum(SalesValue).

Filterruta och tabellresultat för Country



Följande tabell visar resultatet när du väljer `Country` i filtret `FilterField`.

Resultattabell för `Country`

Country	Sum(SalesValue)
Canada	700
Mexico	200
USA	1200

Följande tabell visar resultatet när du väljer `SalesPerson` i filtret `FilterField`.

Resultattabell för `SalesPerson`

SalesPerson	Avg(SalesValue)
Alice	700
Bob	300
Carol	400
Dave	200
John	500

GetObjectMeasure - diagramfunktion

GetObjectMeasure() returnerar namnet på måttet. **Index** är ett valfritt heltal som anger måttet som ska returneras.

Du kan inte använda den här funktionen i ett diagram på följande platser: titel, underrubrik, fotnot, referenslinjeuttryck och min/max-uttryck.

Du kan inte referera till namnet på en dimension eller ett mått i ett annat objekt med Object ID.

Syntax:

```
GetObjectMeasure ([index])
```

Returnerad datatyp: Sträng

Exempel: diagramuttryck

Exempel	Resultat
`GetObjectMeasure ()` `GetObjectMeasure (0)`	Returnerar namnet på det första mätvärdet i diagrammet.
`GetObjectMeasure (1)`	Returnerar namnet på det andra mätvärdet i diagrammet.

Exempel – grunderna i GetObjectMeasure

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `CustomerID`
  - `TransactionQuantity`

Laddningsskript

```
Example:
LOAD * INLINE [
CustomerID, TransactionQuantity
049681, 13
203521, 6
203521, 21
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `CustomerID`

Skapa följande mått:

- `Sum(TransactionQuantity)` - första mätvärdet.
- `Avg(TransactionQuantity)` - andra mätvärdet
- `=GetObjectMeasure()`, för att hitta det första mätvärdet i tabellen.
- `=GetObjectMeasure(0)`, för att hitta det första mätvärdet i tabellen.
- `=GetObjectMeasure(1)`, för att hitta det andra mätvärdet i tabellen.

Resultattabell

CustomerID	Sum (TransactionQuantity)	Avg (TransactionQuantity)	GetObjectMeasure ()	GetObjectMeasure (0)	GetObjectMeasure (1)
Totals	**40**	**13.333333**	**Sum (TransactionQuantity)**	**Sum (TransactionQuantity)**	**Avg (TransactionQuantity)**
049681	13	13	=Sum (TransactionQuantity)	=Sum (TransactionQuantity)	=Avg (TransactionQuantity)
203521	27	13.5	=Sum (TransactionQuantity)	=Sum (TransactionQuantity)	=Avg (TransactionQuantity)

När du tittar på resultaten kan du se hur funktionen `GetObjectMeasure` returnerar namnet på det -som anges i funktionsparametrarna.

GetPossibleCount - diagramfunktion

GetPossibleCount() används för att hitta antalet möjliga värden i det identifierade fältet. Om det identifierade fältet innehåller urval räknas de markerade (gröna) fälten. Värden som är associerade på annat sätt (vita värden) räknas.

För fält med urval returnerar **GetPossibleCount()** antalet valda (gröna) fält.

Syntax:

```
GetPossibleCount (field_name)
```

Returnerad datatyp: heltal

> *Färgerna som används i urvalsfältet och för varje urvaltillstånd kan ändras med ett anpassat tema. Om du arbetar med en applikation som använder ett anpassat tema, kanske du märker att dina val inte visas med de färger som beskrivs i det här hjälpämnet.*

Argument

Argument	Beskrivning
field_name	Fältet som innehåller dataintervallet som ska mätas.

I följande tabell listas andra funktioner som är relaterade till denna funktion.

Relaterade funktioner

Funktion	Interaktion
GetStateCounts - diagramfunktion (page 1342)	Med hjälp av **GetStateCounts()** kan du kombinera beräkningen av följande antal med ett enda funktionsanrop: • Antal valda inkluderade värden. • Antal möjliga värden. • Antal alternativa värden. • Antal uteslutna värden, exklusive alternativa och valda uteslutna värden. • Antal valda uteslutna värden.
GetSelectedCount - diagramfunktion (page 1338)	Returnerar antalet valda inkluderade värden.
GetAlternativeCount - diagramfunktion (page 1305)	Returnerar antalet alternativa värden.
GetPossibleCount - diagramfunktion (page 1334)	Returnerar antalet uteslutna värden, exklusive alternativa och valda uteslutna värden.

Exempel: diagramuttryck

Exempel	Resultat
`GetPossibleCount( Initials )`	Om inga val görs returneras det totala antalet unika värden i fältet `Initials`. Om ett värde i det här fältet väljs returneras 1. Om ett val görs i ett annat fält är det värde som returneras antalet unika värden i fältet `Initials` som relaterar till de värden som valts i det andra fältet.

Exempel – grunderna i GetPossibleCount

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `FirstName`
  - `LastName`
  - `Initials`
  - `HasCellphone`

Laddningsskript

```
Example:
LOAD * inline [
FirstName|LastName|Initials|HasCellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

Resultat

1. Ladda data och öppna ett ark. Skapa en ny filterruta och lägg till dessa fält som dimensioner:
   - `FirstName`
   - `Initials`
2. Skapa två KPI-objekt, ett för vart och ett av följande mätuttryck:
   - KPI 1: `=GetPossibleCount(FirstName)` ska returnera antalet möjliga unika värden i fältet `FirstName`.
   - KPI 2: `=GetPossibleCount(Initials)` ska returnera antalet möjliga unika värden i fältet `Initials`.

   Varje KPI returnerar antalet möjliga unika värden. Utan några val i filtret returnerar KPI 1 värdet `5` och KPI 2 returnerar värdet `6`.

   > *I det här exemplet har utformningen av KPI:n justerats för att använda ett responsiv layoutbeteende som underlättar visning av resultatet.*

3. Klicka på **Redigera ark** för att växla till analysläge och välj `John` i `FirstName`-filtret.
   Båda KPI:erna returnerar värdet `1` eftersom det finns ett valt värde i fältet `FirstName` och `JA` är det enda associerade värdet i fältet `Initials`.
4. Töm urvalet i `FirstName`-filtret och välj sedan `Peter`.
   KPI 1 returnerar värdet `1` eftersom det finns one möjligt unikt värde tillgängligt i fältet `FirstName`. KPI 2 returnerar `2` eftersom det finns two möjliga unika värden tillgängliga i fältet `Initials`.

Exempel – GetPossibleCount-scenario

Diagramuttryck

Översikt

En chef vill filtrera medarbetardata för att avgöra vilka medarbetare som har erfarenhet inom olika kompetensområden.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Project`
  - `Employee`
  - `Skill`

Laddningsskript

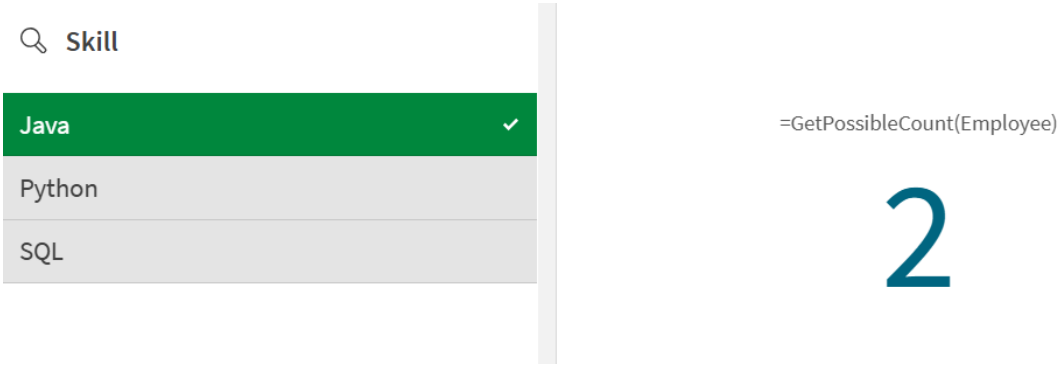
```
Example:
LOAD * INLINE [
Project, Employee, Skill
Project X, John, Java
Project X, Mary, SQL
Project Y, James, Python
Project Y, Linda, Java
Project Z, Robert, SQL
];
```

Resultat

1. Ladda data och öppna ett ark. Skapa en ny filterruta och lägg till detta fält som en dimension:
   - `Skill`
2. Skapa ett KPI-objekt och lägg till följande mått:
   - `=GetPossibleCount(Employee)` för att fastställa antalet anställda.

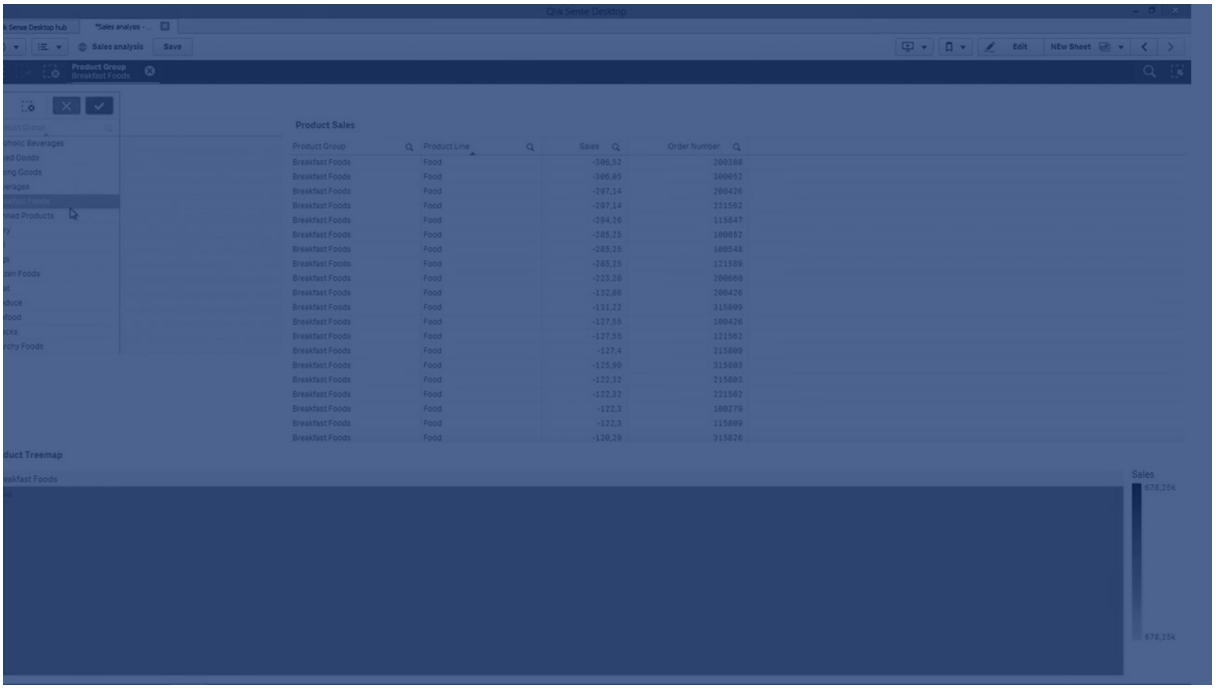
   Utan några val i filterrutan returnerar KPI:n värdet `5` eftersom det finns five anställda.

   I det här exemplet har utformningen av KPI:n justerats för att använda ett responsiv layoutbeteende som underlättar visning av resultatet.
3. Klicka på **Redigera ark** för att växla till analysläge och välj `Java` i `Skill`-filtret.
   KPI:n ger värdet `2` eftersom det finns two anställda som har kompetensen `Java`.



GetSelectedCount - diagramfunktion

GetSelectedCount() finner antalet valda (gröna) värden i ett fält.



Syntax:

```
GetSelectedCount (field_name [, include_excluded [, state_name]])
```

Returnerad datatyp: heltal

> *Färgerna som används i urvalsfältet och för varje urvaltillstånd kan ändras med ett anpassat tema. Om du arbetar med en applikation som använder ett anpassat tema, kanske du märker att dina val inte visas med de färger som beskrivs i det här hjälpämnet.*

Argument

Argument	Beskrivning
field_name	Fältet som innehåller dataintervallet som ska mätas.
include_excluded	Om det anges som **True()**, inkluderar räkningen valda värden som just nu är exkluderade av urval i andra fält. Om de är False eller utelämnade inkluderas dessa värden inte.
state_name	Namnet på ett alternativt tillstånd som har valts till den specifika visualiseringen. Om argumentet **state_name** används tas endast hänsyn till de urval som är förknippade med det angivna tillståndsnamnet.

I följande tabell listas andra funktioner som är relaterade till denna funktion.

Relaterade funktioner

Funktion	Interaktion
GetStateCounts - diagramfunktion (page 1342)	Med hjälp av **GetStateCounts()** kan du kombinera beräkningen av följande antal med ett enda funktionsanrop: • Antal valda inkluderade värden. • Antal möjliga värden. • Antal alternativa värden. • Antal uteslutna värden, exklusive alternativa och valda uteslutna värden. • Antal valda uteslutna värden.
GetPossibleCount - diagramfunktion (page 1334)	Returnerar antalet möjliga värden.
GetAlternativeCount - diagramfunktion (page 1305)	Returnerar antalet alternativa värden.
GetSelectedCount - diagramfunktion (page 1338)	Returnerar antalet uteslutna värden, exklusive alternativa och valda uteslutna värden.

Exempel: diagramuttryck

Exempel	Resultat
`GetSelectedCount (FirstName)`	Om inga val görs returneras `0`. Om något val har gjorts i fältet `FirstName` returneras antalet valda artiklar.
`GetSelectedCount (FirstName, True ())`	Om inga val görs returneras `0`. Om några val har gjorts i fältet `FirstName` returneras det antal artiklar som valts, oberoende av eventuella efterföljande val i andra fält som kan minska de tillgängliga värdena i fältet `FirstName`.

Exempel – grunderna i GetSelectedCount

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `FirstName`
  - `LastName`
  - `Initials`
  - `HasCellphone`

Laddningsskript

```
Example:
LOAD * inline [
FirstName|LastName|Initials|HasCellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

Resultat

1. Ladda data och öppna ett ark. Skapa en ny filterruta och lägg till dessa fält som dimensioner:
   - `FirstName`
   - `Initials`
2. Skapa två KPI-objekt, ett för vart och ett av följande mätuttryck:
   - KPI 1: `=GetSelectedCount(FirstName)` för att returnera antalet artiklar som valts i `FirstName`-fältet.
   - KPI 2: `=GetSelectedCount(Initials)` för att returnera antalet artiklar som valts i `Initials`-fältet.

   Varje KPI returnerar antalet valda objekt i respektive fält. Utan några val i filtren returnerar båda KPI:erna värdet `0`.

   I det här exemplet har utformningen av KPI:n justerats för att använda ett responsiv layoutbeteende som underlättar visning av resultatet.
3. Klicka på **Redigera ark** för att växla till analysläge och välj `John` i `FirstName`-filtret.

KPI 1 returnerar värdet `1` eftersom det finns ett objekt som valts i `FirstName`-filtret. KPI 2 returnerar fortfarande `0` eftersom filtret `Initials` inte har några valda objekt.

Exempel – GetSelectedCount med parametern include_excluded

Diagramuttryck

Översikt

I det här exemplet används samma datauppsättning som i det föregående exemplet, men här visas skillnaden i resultat när parametern `include_excluded` ställs in på `True()`.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `FirstName`
  - `LastName`
  - `Initials`
  - `HasCellphone`

Laddningsskript

```
Example:
LOAD * inline [
FirstName|LastName|Initials|HasCellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

Resultat

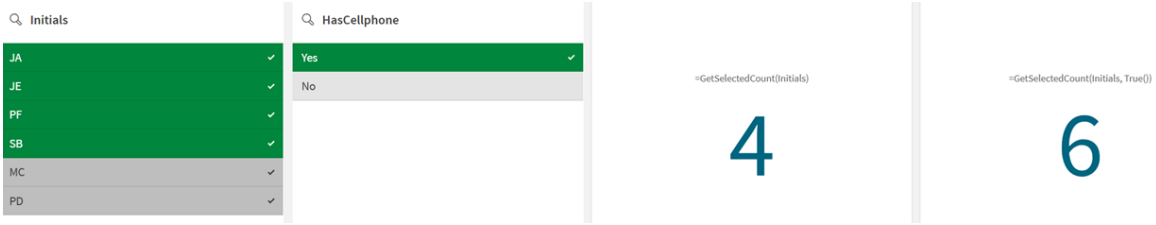
1. Ladda data och öppna ett ark. Skapa en ny filterruta och lägg till dessa fält som dimensioner:
   - `Initials`
   - `HasCellPhone`

2. Skapa två KPI-objekt, ett för vart och ett av följande mätuttryck:
   - KPI 1: `=GetselectedCount(Initials)` för att fastställa antalet värden som har valts i fältet `Initials`.
   - KPI 2: `=GetSelectedCount([Initials], True())` för att inkludera valda värden som just nu exkluderas av urval i andra fält.

   Utan några val i filterrutan returnerar båda KPI:erna värdet `0`.

   > *I det här exemplet har utformningen av KPI:n justerats för att använda ett responsiv layoutbeteende som underlättar visning av resultatet.*

3. Klicka på **Redigera ark** för att växla till analysläge. Välj alla värden i `Initials`-filtret och välj sedan `Yes` i `HasCellphone`-filtret.



När du jämför resultaten för varje KPI kan du se hur var och en returnerar ett annat värde beroende på om parametern `include_excluded` är inställd på `True()` eller inte. KPI 2 returnerar värdet `6` eftersom det inkluderar alla valda värden i `Initials`-filtret, även om valet i `HasCellphone`-filtret har reducerat de tillgängliga objekten till `4`. KPI 1 returnerar `4` eftersom den inte räknar de utvalda uteslutna värdena i `Initials`.

GetStateCounts - diagramfunktion

Diagramfunktionen **GetStateCounts()** används för att beräkna det totala antalet unika värden som matchar de angivna urvalstillstånden.

Med **GetStateCounts()** kan du kombinera beräkningarna från följande funktioner till ett enda funktionsanrop: **GetSelectedCount()**, **GetNotSelectedCount()**, **GetAlternativeCount()**, **GetPossibleCount()** och **GetExcludedCount()**. Antalet valda uteslutna värden är också tillgängligt för att lägga till i beräkningen. Du kan ange om varje funktionsberäkning lägger till eller subtraherar från de returnerade totalvärdena.

Syntax:

```
GetStateCounts (field_name, state_name [, state_type1,...state_typeN])
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
field_name	Det fält för vilket du beräknar urvalstillståndet. Ett icke-existerande fältnamn resulterar i ett nullresultat.
state_name	Namnet på det parallella tillståndet. Om argumentet är tomt (' ') eller null, används det ärvda parallella tillståndet. Använd $ för att uttryckligen använda standardtillståndet. Ett namngivet (ej tomt) namn på ett tillstånd som inte matchar ett befintligt tillstånd resulterar i ett nullresultat.
state_type	En lista med en eller flera tillståndstyper för fältvärdet. Dessa tillståndstyper kommer att aggregeras till ett antal. En tillståndstyp anges med hjälp av en nyckel. Skriv in varje nyckel med enkla citattecken. När detta argument utelämnas returnerar funktionen en sträng med alla tillgängliga tillståndsantal för fältet i samma ordning som uppräkningen. En lista över de tillstånd som du kan använda finns i tabellen nedanför.

Tillståndstyper refereras med hjälp av specifika nycklar. Du kan använda den numeriska eller textversionen av nyckeln. Kombinera fler än en nyckel i samma uttryck för att ytterligare anpassa resultatet. Du kan subtrahera antalet tillstånd från det totala antalet i stället för att lägga till det. För att göra det använder du textnyckeln och anger en minussymbol (-) som prefix för tillståndstypen.

Tillståndstypen, och i förekommande fall minussymbolen, måste omges av en uppsättning enkla citattecken.

Nycklar för varje fälttillståndstyp

Fälttillståndstyp	Beskrivning	Numerisk nyckel	Textnyckel
Valda	Inkluderar valda värden i beräkningen. För motsvarande funktion, se *GetSelectedCount - diagramfunktion (page 1338)*.	1	S
Valfritt	Inkluderar valfria (ej valda, men möjliga att välja) värden i beräkningen. För motsvarande funktion, se *GetPossibleCount - diagramfunktion (page 1334)*.	2	O
Avmarkerad	Inkluderar ej valda värden i beräkningen. Denna tillståndstyp är endast tillgänglig när fältet är i AND-läge.	3	D

Fälttillståndstyp	Beskrivning	Numerisk nyckel	Textnyckel
Alternativ	Inkluderar alternativa värden i beräkningen. För motsvarande funktion, se *GetAlternativeCount - diagramfunktion (page 1305)*.	4	A
Utesluten	Inkluderar uteslutna (ej valda) värden i beräkningen. För motsvarande funktion, se *GetExcludedCount - diagramfunktion (page 1316)*.	5	X
Urval uteslutet	Inkluderar valda uteslutna värden i beräkningen.	6	XS

Användning

Med **GetStateCounts()** kan du beräkna ett anpassat urvalstillstånd. Funktionen gör att du kan konsolidera flera funktionsanrop till ett enda funktionsanrop, vilket förenklar processen med att skriva ditt uttryck.

Du kan exempelvis behöva beräkna det totala antalet uteslutna, alternativa och valda uteslutna värden för ett fält. Du kan använda **GetStateCounts()** för att beräkna detta antal.

Exempel: diagramuttryck

Exempel	Resultat
`=GetStateCounts( ProductName, Null(), 'S' )`	Returnerar det valda antalet för `ProductName`, i det ärvda alternativa tillståndet.
`=GetStateCounts( ProductName, '', 'X', 'A', 'XS' )`	Returnerar det totala antalet uteslutna, valda uteslutna och alternativa värden för `ProductName`. Det ärvda parallella tillståndet används.
`=GetStateCounts( ProductName, '', 'S', 'XS' )`	Returnerar det totala antalet användarurval `ProductName`, i det ärvda alternativa tillståndet.
Förutsatt att fältet *ProductName* är i och-läge. `=GetStateCounts (ProductName, '', 'D', '-O')`	Returnerar antalet värden som inte valts, subtraherat med antalet möjliga värden, för `ProductName`. Det ärvda parallella tillståndet används.
`=GetStateCounts (ProductName, '', 'X', , 'A', 'XS')`	Returnerar det totala antalet uteslutna, valda uteslutna och alternativa värden för `ProductName`. Det ärvda parallella tillståndet används.
`=GetStateCounts (ProductName, '$', 'O')`	Returnerar det möjliga antalet för `ProductName`, i det parallella standardtillståndet.
`=GetStateCounts (ProductName, 'StateA', 'S')`	Returnerar det valda antalet för `ProductName`, i det alternativa tillståndet benämnt `StateA`.

Exempel 1 – Antal totala användarurval (inklusive valda uteslutna värden)

Diagramuttryck

Översikt

Funktionen **GetSelectedCount()** returnerar det totala antalet valda värden, exklusive det totala antalet valda uteslutna värden. Med **GetExcludedCount()** kan du inkludera antalet utvalda uteslutna värden i diagrammets beräkning.

Öppna Skriptredigeraren och lägg till följande laddningsskript till ett nytt delavsnitt.

Laddningsskript

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark.
2. Skapa två filterrutor.
3. Lägg till **First name** som dimension i den första filterrutan. Lägg till **Last name** som dimension i den andra filterrutan.
4. Skapa därefter ett KPI med detta mått:
   `=GetStateCounts([First Name],'', 'S','XS')`
5. I filterrutan **First name** väljer du värdena **Mark** och **Peter**. I filterrutan **Last name** väljer du värdena **Carr**.
6. Observera att KPI visar ett värde på 2. Detta återspeglar att alla värden som du har valt, inklusive det valda uteslutna värdet **Peter**, räknas.

Exempel 2 – Kombination av uteslutna, valda uteslutna och alternativa antal

Diagramuttryck

Översikt

Funktionen **GetExcludedCount()** returnerar det totala antalet unika värden som inte är valda och uteslutna. Om du vill inkludera valda uteslutna och alternativa antal i din visualisering kan du använda funktionen **GetStateCounts()** på följande sätt.

Öppna Skriptredigeraren och lägg till följande laddningsskript till ett nytt delavsnitt.

Laddningsskript

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark.
2. Skapa två filterrutor.
3. Lägg till **First name** som dimension i den första filterrutan. Lägg till **Last name** som dimension i den andra filterrutan.
4. Skapa därefter ett KPI med detta mått:
   `=GetStateCounts([First name], '', 'X', 'A', 'XS')`
5. I filterrutan **First name** väljer du värdena **John** och **Peter**. I filterrutan **Last name** väljer du värdena **Franc**.
6. Observera att KPI visar ett värde på 4. Detta återspeglar att det valda uteslutna värdet **John** ingår i antalet, utöver antalet uteslutna värden.
7. Redigera KPI:t och ersätt det befintliga måttet med följande:
   `=GetStateCounts([Last name], '', 'X', 'A', 'XS')`
8. Behåll de befintliga urvalen och välj värdet **Anderson** i filterrutan **Last name**.
9. Observera att KPI fortfarande visar ett värde på 4. Detta återspeglar att det valda uteslutna värdet **Devonshire** ingår i antalet, utöver antalet uteslutna värden.

8.10 Filfunktioner

Filfunktionerna (finns endast i skriptuttryck) returnerar information om tabellfilen som för närvarande läses. Funktionerna returnerar NULL för alla datakällor utom tabellfiler (undantag: **ConnectString( )**).

Filfunktioner – en översikt

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Attribute

Denna skriptfunktion returnerar metataggarnas värde från olika filformat som text. Följande filformat stöds: MP3, WMA, WMV, PNG och JPG. Om filen **filename** inte finns, inte är en filtyp som stöds eller inte innehåller en metatagg som heter **attributename**, returneras NULL.

```
Attribute (filename, attributename)
```

ConnectString

ConnectString()-funktionen returnerar namnet på den aktiva datakopplingen för ODBC- eller OLE DB -kopplingar. Funktionen returnerar en tom sträng om ingen **connect**-sats har exekverats eller efter en **disconnect**-sats.

```
ConnectString ()
```

FileBaseName

FileBaseName-funktionen returnerar en textsträng som anger namnet på den tabellfil som håller på att läsas in, dock utan sökväg och filtillägg.

```
FileBaseName ()
```

FileDir

FileDir-funktionen returnerar en textsträng som anger sökvägen till katalogen som innehåller den tabellfil som håller på att läsas in.

```
FileDir ()
```

FileExtension

FileExtension-funktionen returnerar en textsträng som anger filtillägget till den tabellfil som håller på att läsas in.

```
FileExtension ()
```

FileName

FileName-funktionen returnerar en textsträng som anger namnet på den tabellfil som håller på att läsas in, utan sökväg men med filtillägget.

```
FileName ()
```

FilePath

FilePath-funktionen returnerar en textsträng som anger den kompletta sökvägen till den tabellfil som håller på att läsas in.

```
FilePath ()
```

FileSize

FileSize-funktionen returnerar ett heltal som anger storleken i byte för filen filename eller, om inget filename angetts, för den tabellfil som håller på att läsas in.

```
FileSize ()
```

FileTime

FileTime-funktionen returnerar en tidsmarkör i UTC-format för den senaste ändringen av en angiven fil. Om ingen fil anges returnerar funktionen en tidsmarkör i UTC för den senaste ändringen av den tabellfil som för närvarande är inläst.

```
FileTime ([ filename ])
```

GetFolderPath

Funktionen **GetFolderPath** returnerar värdet av funktionen Microsoft Windows *SHGetFolderPath*. Den här funktionen tar som indata namnet på en Microsoft Windows -mapp och returnerar den fullständiga sökvägen till mappen.

```
GetFolderPath ()
```

QvdCreateTime

Denna skriptfunktion returnerar XML-huvudets tidsstämpel från en QVD-fil, om någon sådan finns, annars returneras NULL. I tidsmarkören anges tiden i UTC.

```
QvdCreateTime (filename)
```

QvdFieldName

Denna skriptfunktion returnerar namnet på fält nummer **fieldno** i en QVD-fil. Om fältet inte finns returneras NULL.

```
QvdFieldName (filename , fieldno)
```

QvdNoOfFields

Denna skriptfunktion returnerar antalet fält i en QVD-fil.

```
QvdNoOfFields (filename)
```

QvdNoOfRecords

Denna skriptfunktion returnerar aktuellt antal poster i en QVD-fil.

```
QvdNoOfRecords (filename)
```

QvdTableName

Denna skriptfunktion returnerar namnet på tabellen i en QVD-fil.

```
QvdTableName (filename)
```

Attribute

Denna skriptfunktion returnerar metataggarnas värde från olika filformat som text. Följande filformat stöds: MP3, WMA, WMV, PNG och JPG. Om filen **filename** inte finns, inte är en filtyp som stöds eller inte innehåller en metatagg som heter **attributename**, returneras NULL.

Syntax:

```
Attribute(filename, attributename)
```

Ett stort antal metataggar kan läsas. Exemplen i det här avsnittet visar vilka taggar som kan läsas för respektive filtyp som stöds.

Du kan bara läsa metataggar som har sparats i filen enligt den relevanta specifikationen, exempelvis ID2v3 för MP3-filer eller EXIF för JPG-filer, inte metainformation som har sparats i ***Utforskaren*** *i Windows.*

Argument

Argument	Beskrivning
filename	Namnet på en mediefil, inklusive sökväg om så krävs, som en mapp datakoppling **Exempel:** ***'lib://Table Files/'*** I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense applikationens arbetskatalog. **Exempel:** *data*
attributename	Namnet på en metatagg.

I exemplen används **GetFolderPath**-funktionen för att hitta sökvägarna till mediefiler. Eftersom **GetFolderPath** endast stöds i bakåtkompatibelt läge måste du byta ut referenserna till **GetFolderPath** mot en lib://-datakopplingssökväg när du använder funktionen i standardläge eller i Qlik Sense SaaS.

Behörighetskontroll för filsystem (page 1907)

Example 1: MP3-filer

Det här skriptet läser alla möjliga MP3-metataggar i mappen *MyMusic*.

```
// Script to read MP3 meta tags
for each vExt in 'mp3'
for each vFoundFile in filelist( GetFolderPath('MyMusic') & '\*.'& vExt )
FileList:
LOAD FileLongName,
    subfield(FileLongName,'\',-1) as FileShortName,
    num(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,
    FileTime(FileLongName) as FileTime,
    // ID3v1.0 and ID3v1.1 tags
    Attribute(FileLongName, 'Title') as Title,
    Attribute(FileLongName, 'Artist') as Artist,
    Attribute(FileLongName, 'Album') as Album,
    Attribute(FileLongName, 'Year') as Year,
    Attribute(FileLongName, 'Comment') as Comment,
    Attribute(FileLongName, 'Track') as Track,
    Attribute(FileLongName, 'Genre') as Genre,
    // ID3v2.3 tags
    Attribute(FileLongName, 'AENC') as AENC, // Audio encryption
```

```
Attribute(FileLongName, 'APIC') as APIC, // Attached picture
Attribute(FileLongName, 'COMM') as COMM, // Comments
Attribute(FileLongName, 'COMR') as COMR, // Commercial frame
Attribute(FileLongName, 'ENCR') as ENCR, // Encryption method registration
Attribute(FileLongName, 'EQUA') as EQUA, // Equalization
Attribute(FileLongName, 'ETCO') as ETCO, // Event timing codes
Attribute(FileLongName, 'GEOB') as GEOB, // General encapsulated object
Attribute(FileLongName, 'GRID') as GRID, // Group identification registration
Attribute(FileLongName, 'IPLS') as IPLS, // Involved people list
Attribute(FileLongName, 'LINK') as LINK, // Linked information
Attribute(FileLongName, 'MCDI') as MCDI, // Music CD identifier
Attribute(FileLongName, 'MLLT') as MLLT, // MPEG location lookup table
Attribute(FileLongName, 'OWNE') as OWNE, // Ownership frame
Attribute(FileLongName, 'PRIV') as PRIV, // Private frame
Attribute(FileLongName, 'PCNT') as PCNT, // Play counter
Attribute(FileLongName, 'POPM') as POPM, // Popularimeter
Attribute(FileLongName, 'POSS') as POSS, // Position synchronisation frame
Attribute(FileLongName, 'RBUF') as RBUF, // Recommended buffer size
Attribute(FileLongName, 'RVAD') as RVAD, // Relative volume adjustment
Attribute(FileLongName, 'RVRB') as RVRB, // Reverb
Attribute(FileLongName, 'SYLT') as SYLT, // Synchronized lyric/text
Attribute(FileLongName, 'SYTC') as SYTC, // Synchronized tempo codes
Attribute(FileLongName, 'TALB') as TALB, // Album/Movie/Show title
Attribute(FileLongName, 'TBPM') as TBPM, // BPM (beats per minute)
Attribute(FileLongName, 'TCOM') as TCOM, // Composer
Attribute(FileLongName, 'TCON') as TCON, // Content type
Attribute(FileLongName, 'TCOP') as TCOP, // Copyright message
Attribute(FileLongName, 'TDAT') as TDAT, // Date
Attribute(FileLongName, 'TDLY') as TDLY, // Playlist delay
Attribute(FileLongName, 'TENC') as TENC, // Encoded by
Attribute(FileLongName, 'TEXT') as TEXT, // Lyricist/Text writer
Attribute(FileLongName, 'TFLT') as TFLT, // File type
Attribute(FileLongName, 'TIME') as TIME, // Time
Attribute(FileLongName, 'TIT1') as TIT1, // Content group description
Attribute(FileLongName, 'TIT2') as TIT2, // Title/songname/content description
Attribute(FileLongName, 'TIT3') as TIT3, // Subtitle/Description refinement
Attribute(FileLongName, 'TKEY') as TKEY, // Initial key
Attribute(FileLongName, 'TLAN') as TLAN, // Language(s)
Attribute(FileLongName, 'TLEN') as TLEN, // Length
Attribute(FileLongName, 'TMED') as TMED, // Media type
Attribute(FileLongName, 'TOAL') as TOAL, // Original album/movie/show title
Attribute(FileLongName, 'TOFN') as TOFN, // Original filename
Attribute(FileLongName, 'TOLY') as TOLY, // Original lyricist(s)/text writer(s)
Attribute(FileLongName, 'TOPE') as TOPE, // Original artist(s)/performer(s)
Attribute(FileLongName, 'TORY') as TORY, // Original release year
Attribute(FileLongName, 'TOWN') as TOWN, // File owner/licensee
Attribute(FileLongName, 'TPE1') as TPE1, // Lead performer(s)/Soloist(s)
Attribute(FileLongName, 'TPE2') as TPE2, // Band/orchestra/accompaniment
Attribute(FileLongName, 'TPE3') as TPE3, // Conductor/performer refinement
Attribute(FileLongName, 'TPE4') as TPE4, // Interpreted, remixed, or otherwise modified by
Attribute(FileLongName, 'TPOS') as TPOS, // Part of a set
Attribute(FileLongName, 'TPUB') as TPUB, // Publisher
Attribute(FileLongName, 'TRCK') as TRCK, // Track number/Position in set
Attribute(FileLongName, 'TRDA') as TRDA, // Recording dates
Attribute(FileLongName, 'TRSN') as TRSN, // Internet radio station name
Attribute(FileLongName, 'TRSO') as TRSO, // Internet radio station owner
```

```
    Attribute(FileLongName, 'TSIZ') as TSIZ, // Size
    Attribute(FileLongName, 'TSRC') as TSRC, // ISRC (international standard recording code)
    Attribute(FileLongName, 'TSSE') as TSSE, // Software/Hardware and settings used for
encoding
    Attribute(FileLongName, 'TYER') as TYER, // Year
    Attribute(FileLongName, 'TXXX') as TXXX, // User defined text information frame
    Attribute(FileLongName, 'UFID') as UFID, // Unique file identifier
    Attribute(FileLongName, 'USER') as USER, // Terms of use
    Attribute(FileLongName, 'USLT') as USLT, // Unsychronized lyric/text transcription
    Attribute(FileLongName, 'WCOM') as WCOM, // Commercial information
    Attribute(FileLongName, 'WCOP') as WCOP, // Copyright/Legal information
    Attribute(FileLongName, 'WOAF') as WOAF, // Official audio file webpage
    Attribute(FileLongName, 'WOAR') as WOAR, // Official artist/performer webpage
    Attribute(FileLongName, 'WOAS') as WOAS, // Official audio source webpage
    Attribute(FileLongName, 'WORS') as WORS, // Official internet radio station homepage
    Attribute(FileLongName, 'WPAY') as WPAY, // Payment
    Attribute(FileLongName, 'WPUB') as WPUB, // Publishers official webpage
    Attribute(FileLongName, 'WXXX') as WXXX; // User defined URL link frame
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels);
Next vFoundFile
Next vExt
```

Example 2: JPEG

Det här skriptet läser alla möjliga EXIF-metataggar från JPG-filer i mappen *MyPictures*.

```
// Script to read Jpeg Exif meta tags
for each vExt in 'jpg', 'jpeg', 'jpe', 'jfif', 'jif', 'jfi'
for each vFoundFile in filelist( GetFolderPath('MyPictures') & '\*.'& vExt )
FileList:
LOAD FileLongName,
    subfield(FileLongName,'\',-1) as FileShortName,
    num(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,
    FileTime(FileLongName) as FileTime,
    // ************   Exif Main (IFD0) Attributes   ************
    Attribute(FileLongName, 'ImageWidth') as ImageWidth,
    Attribute(FileLongName, 'ImageLength') as ImageLength,
    Attribute(FileLongName, 'BitsPerSample') as BitsPerSample,
    Attribute(FileLongName, 'Compression') as Compression,
    //  examples: 1=uncompressed, 2=CCITT, 3=CCITT 3, 4=CCITT 4,
    //5=LZW, 6=JPEG (old style), 7=JPEG, 8=Deflate, 32773=PackBits RLE,
    Attribute(FileLongName, 'PhotometricInterpretation') as PhotometricInterpretation,
    //  examples: 0=WhiteIsZero, 1=BlackIsZero, 2=RGB, 3=Palette, 5=CMYK, 6=YCbCr,
    Attribute(FileLongName, 'ImageDescription') as ImageDescription,
    Attribute(FileLongName, 'Make') as Make,
    Attribute(FileLongName, 'Model') as Model,
    Attribute(FileLongName, 'StripOffsets') as StripOffsets,
    Attribute(FileLongName, 'Orientation') as Orientation,
    //  examples: 1=TopLeft, 2=TopRight, 3=BottomRight, 4=BottomLeft,
    // 5=LeftTop, 6=RightTop, 7=RightBottom, 8=LeftBottom,
    Attribute(FileLongName, 'SamplesPerPixel') as SamplesPerPixel,
    Attribute(FileLongName, 'RowsPerStrip') as RowsPerStrip,
    Attribute(FileLongName, 'StripByteCounts') as StripByteCounts,
    Attribute(FileLongName, 'XResolution') as XResolution,
    Attribute(FileLongName, 'YResolution') as YResolution,
    Attribute(FileLongName, 'PlanarConfiguration') as PlanarConfiguration,
```

```
    //  examples: 1=chunky format, 2=planar format,
    Attribute(FileLongName, 'ResolutionUnit') as ResolutionUnit,
    //  examples: 1=none, 2=inches, 3=centimeters,
    Attribute(FileLongName, 'TransferFunction') as TransferFunction,
    Attribute(FileLongName, 'Software') as Software,
    Attribute(FileLongName, 'DateTime') as DateTime,
    Attribute(FileLongName, 'Artist') as Artist,
    Attribute(FileLongName, 'HostComputer') as HostComputer,
    Attribute(FileLongName, 'WhitePoint') as WhitePoint,
    Attribute(FileLongName, 'PrimaryChromaticities') as PrimaryChromaticities,
    Attribute(FileLongName, 'YCbCrCoefficients') as YCbCrCoefficients,
    Attribute(FileLongName, 'YCbCrSubSampling') as YCbCrSubSampling,
    Attribute(FileLongName, 'YCbCrPositioning') as YCbCrPositioning,
    //  examples: 1=centered, 2=co-sited,
    Attribute(FileLongName, 'ReferenceBlackWhite') as ReferenceBlackWhite,
    Attribute(FileLongName, 'Rating') as Rating,
    Attribute(FileLongName, 'RatingPercent') as RatingPercent,
    Attribute(FileLongName, 'ThumbnailFormat') as ThumbnailFormat,
    //  examples: 0=Raw Rgb, 1=Jpeg,
    Attribute(FileLongName, 'Copyright') as Copyright,
    Attribute(FileLongName, 'ExposureTime') as ExposureTime,
    Attribute(FileLongName, 'FNumber') as FNumber,
    Attribute(FileLongName, 'ExposureProgram') as ExposureProgram,
    //  examples: 0=Not defined, 1=Manual, 2=Normal program, 3=Aperture priority, 4=Shutter
priority,
    // 5=Creative program, 6=Action program, 7=Portrait mode, 8=Landscape mode, 9=Bulb,
    Attribute(FileLongName, 'ISOSpeedRatings') as ISOSpeedRatings,
    Attribute(FileLongName, 'TimeZoneOffset') as TimeZoneOffset,
    Attribute(FileLongName, 'SensitivityType') as SensitivityType,
    //  examples: 0=Unknown, 1=Standard output sensitivity (SOS), 2=Recommended exposure index
(REI),
    // 3=ISO speed, 4=Standard output sensitivity (SOS) and Recommended exposure index (REI),
    //5=Standard output sensitivity (SOS) and ISO Speed, 6=Recommended exposure index (REI)
and ISO Speed,
    // 7=Standard output sensitivity (SOS) and Recommended exposure index (REI) and ISO speed,
    Attribute(FileLongName, 'ExifVersion') as ExifVersion,
    Attribute(FileLongName, 'DateTimeOriginal') as DateTimeOriginal,
    Attribute(FileLongName, 'DateTimeDigitized') as DateTimeDigitized,
    Attribute(FileLongName, 'ComponentsConfiguration') as ComponentsConfiguration,
    //  examples: 1=Y, 2=Cb, 3=Cr, 4=R, 5=G, 6=B,
    Attribute(FileLongName, 'CompressedBitsPerPixel') as CompressedBitsPerPixel,
    Attribute(FileLongName, 'ShutterSpeedValue') as ShutterSpeedValue,
    Attribute(FileLongName, 'ApertureValue') as ApertureValue,
    Attribute(FileLongName, 'BrightnessValue') as BrightnessValue, //  examples: -1=Unknown,
    Attribute(FileLongName, 'ExposureBiasValue') as ExposureBiasValue,
    Attribute(FileLongName, 'MaxApertureValue') as MaxApertureValue,
    Attribute(FileLongName, 'SubjectDistance') as SubjectDistance,
    //  examples: 0=Unknown, -1=Infinity,
    Attribute(FileLongName, 'MeteringMode') as MeteringMode,
    //  examples: 0=Unknown, 1=Average, 2=CenterWeightedAverage, 3=Spot,
    // 4=MultiSpot, 5=Pattern, 6=Partial, 255=Other,
    Attribute(FileLongName, 'LightSource') as LightSource,
    //  examples: 0=Unknown, 1=Daylight, 2=Fluorescent, 3=Tungsten, 4=Flash, 9=Fine weather,
    // 10=Cloudy weather, 11=Shade, 12=Daylight fluorescent,
    // 13=Day white fluorescent, 14=Cool white fluorescent,
    // 15=White fluorescent, 17=Standard light A, 18=Standard light B, 19=Standard light C,
```

```
    // 20=D55, 21=D65, 22=D75, 23=D50, 24=ISO studio tungsten, 255=other light source,
    Attribute(FileLongName, 'Flash') as Flash,
    Attribute(FileLongName, 'FocalLength') as FocalLength,
    Attribute(FileLongName, 'SubjectArea') as SubjectArea,
    Attribute(FileLongName, 'MakerNote') as MakerNote,
    Attribute(FileLongName, 'UserComment') as UserComment,
    Attribute(FileLongName, 'SubSecTime') as SubSecTime,
    Attribute(FileLongName, 'SubsecTimeOriginal') as SubsecTimeOriginal,
    Attribute(FileLongName, 'SubsecTimeDigitized') as SubsecTimeDigitized,
    Attribute(FileLongName, 'XPTitle') as XPTitle,
    Attribute(FileLongName, 'XPComment') as XPComment,
    Attribute(FileLongName, 'XPAuthor') as XPAuthor,
    Attribute(FileLongName, 'XPKeywords') as XPKeywords,
    Attribute(FileLongName, 'XPSubject') as XPSubject,
    Attribute(FileLongName, 'FlashpixVersion') as FlashpixVersion,
    Attribute(FileLongName, 'ColorSpace') as ColorSpace, //  examples: 1=sRGB,
65535=Uncalibrated,
    Attribute(FileLongName, 'PixelXDimension') as PixelXDimension,
    Attribute(FileLongName, 'PixelYDimension') as PixelYDimension,
    Attribute(FileLongName, 'RelatedSoundFile') as RelatedSoundFile,
    Attribute(FileLongName, 'FocalPlaneXResolution') as FocalPlaneXResolution,
    Attribute(FileLongName, 'FocalPlaneYResolution') as FocalPlaneYResolution,
    Attribute(FileLongName, 'FocalPlaneResolutionUnit') as FocalPlaneResolutionUnit,
    //  examples: 1=None, 2=Inch, 3=Centimeter,
    Attribute(FileLongName, 'ExposureIndex') as ExposureIndex,
    Attribute(FileLongName, 'SensingMethod') as SensingMethod,
    //  examples: 1=Not defined, 2=One-chip color area sensor, 3=Two-chip color area sensor,
    // 4=Three-chip color area sensor, 5=Color sequential area sensor,
    // 7=Trilinear sensor, 8=Color sequential linear sensor,
    Attribute(FileLongName, 'FileSource') as FileSource,
    //  examples: 0=Other, 1=Scanner of transparent type,
    // 2=Scanner of reflex type, 3=Digital still camera,
    Attribute(FileLongName, 'SceneType') as SceneType,
    //  examples: 1=A directly photographed image,
    Attribute(FileLongName, 'CFAPattern') as CFAPattern,
    Attribute(FileLongName, 'CustomRendered') as CustomRendered,
    //  examples: 0=Normal process, 1=Custom process,
    Attribute(FileLongName, 'ExposureMode') as ExposureMode,
    //  examples: 0=Auto exposure, 1=Manual exposure, 2=Auto bracket,
    Attribute(FileLongName, 'WhiteBalance') as WhiteBalance,
    //  examples: 0=Auto white balance, 1=Manual white balance,
    Attribute(FileLongName, 'DigitalZoomRatio') as DigitalZoomRatio,
    Attribute(FileLongName, 'FocalLengthIn35mmFilm') as FocalLengthIn35mmFilm,
    Attribute(FileLongName, 'SceneCaptureType') as SceneCaptureType,
    //  examples: 0=Standard, 1=Landscape, 2=Portrait, 3=Night scene,
    Attribute(FileLongName, 'GainControl') as GainControl,
    //  examples: 0=None, 1=Low gain up, 2=High gain up, 3=Low gain down, 4=High gain down,
    Attribute(FileLongName, 'Contrast') as Contrast,
    //  examples: 0=Normal, 1=Soft, 2=Hard,
    Attribute(FileLongName, 'Saturation') as Saturation,
    //  examples: 0=Normal, 1=Low saturation, 2=High saturation,
    Attribute(FileLongName, 'Sharpness') as Sharpness,
    //  examples: 0=Normal, 1=Soft, 2=Hard,
    Attribute(FileLongName, 'SubjectDistanceRange') as SubjectDistanceRange,
    //  examples: 0=Unknown, 1=Macro, 2=Close view, 3=Distant view,
    Attribute(FileLongName, 'ImageUniqueID') as ImageUniqueID,
    Attribute(FileLongName, 'BodySerialNumber') as BodySerialNumber,
```

```
    Attribute(FileLongName, 'CMNT_GAMMA') as CMNT_GAMMA,
    Attribute(FileLongName, 'PrintImageMatching') as PrintImageMatching,
    Attribute(FileLongName, 'OffsetSchema') as OffsetSchema,
    // ************   Interoperability Attributes   ************
    Attribute(FileLongName, 'InteroperabilityIndex') as InteroperabilityIndex,
    Attribute(FileLongName, 'InteroperabilityVersion') as InteroperabilityVersion,
    Attribute(FileLongName, 'InteroperabilityRelatedImageFileFormat') as
InteroperabilityRelatedImageFileFormat,
    Attribute(FileLongName, 'InteroperabilityRelatedImageWidth') as
InteroperabilityRelatedImageWidth,
    Attribute(FileLongName, 'InteroperabilityRelatedImageLength') as
InteroperabilityRelatedImageLength,
    Attribute(FileLongName, 'InteroperabilityColorSpace') as InteroperabilityColorSpace,
    //  examples: 1=sRGB, 65535=Uncalibrated,
    Attribute(FileLongName, 'InteroperabilityPrintImageMatching') as
InteroperabilityPrintImageMatching,
    // ************   GPS Attributes   ************
    Attribute(FileLongName, 'GPSVersionID') as GPSVersionID,
    Attribute(FileLongName, 'GPSLatitudeRef') as GPSLatitudeRef,
    Attribute(FileLongName, 'GPSLatitude') as GPSLatitude,
    Attribute(FileLongName, 'GPSLongitudeRef') as GPSLongitudeRef,
    Attribute(FileLongName, 'GPSLongitude') as GPSLongitude,
    Attribute(FileLongName, 'GPSAltitudeRef') as GPSAltitudeRef,
    //  examples: 0=Above sea level, 1=Below sea level,
    Attribute(FileLongName, 'GPSAltitude') as GPSAltitude,
    Attribute(FileLongName, 'GPSTimeStamp') as GPSTimeStamp,
    Attribute(FileLongName, 'GPSSatellites') as GPSSatellites,
    Attribute(FileLongName, 'GPSStatus') as GPSStatus,
    Attribute(FileLongName, 'GPSMeasureMode') as GPSMeasureMode,
    Attribute(FileLongName, 'GPSDOP') as GPSDOP,
    Attribute(FileLongName, 'GPSSpeedRef') as GPSSpeedRef,
    Attribute(FileLongName, 'GPSSpeed') as GPSSpeed,
    Attribute(FileLongName, 'GPSTrackRef') as GPSTrackRef,
    Attribute(FileLongName, 'GPSTrack') as GPSTrack,
    Attribute(FileLongName, 'GPSImgDirectionRef') as GPSImgDirectionRef,
    Attribute(FileLongName, 'GPSImgDirection') as GPSImgDirection,
    Attribute(FileLongName, 'GPSMapDatum') as GPSMapDatum,
    Attribute(FileLongName, 'GPSDestLatitudeRef') as GPSDestLatitudeRef,
    Attribute(FileLongName, 'GPSDestLatitude') as GPSDestLatitude,
    Attribute(FileLongName, 'GPSDestLongitudeRef') as GPSDestLongitudeRef,
    Attribute(FileLongName, 'GPSDestLongitude') as GPSDestLongitude,
    Attribute(FileLongName, 'GPSDestBearingRef') as GPSDestBearingRef,
    Attribute(FileLongName, 'GPSDestBearing') as GPSDestBearing,
    Attribute(FileLongName, 'GPSDestDistanceRef') as GPSDestDistanceRef,
    Attribute(FileLongName, 'GPSDestDistance') as GPSDestDistance,
    Attribute(FileLongName, 'GPSProcessingMethod') as GPSProcessingMethod,
    Attribute(FileLongName, 'GPSAreaInformation') as GPSAreaInformation,
    Attribute(FileLongName, 'GPSDateStamp') as GPSDateStamp,
    Attribute(FileLongName, 'GPSDifferential') as GPSDifferential;
    //  examples: 0=No correction, 1=Differential correction,
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels);
Next vFoundFile
Next vExt
```

Example 3: Windows-mediefiler

Det här skriptet läser alla möjliga WMA/WMV ASF-metataggar i mappen *MyMusic*.

```
/ Script to read WMA/WMV ASF meta tags
for each vExt in 'asf', 'wma', 'wmv'
for each vFoundFile in filelist( GetFolderPath('MyMusic') & '\*.'& vExt )
FileList:
LOAD FileLongName,
    subfield(FileLongName,'\',-1) as FileShortName,
    num(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,
    FileTime(FileLongName) as FileTime,
    Attribute(FileLongName, 'Title') as Title,
    Attribute(FileLongName, 'Author') as Author,
    Attribute(FileLongName, 'Copyright') as Copyright,
    Attribute(FileLongName, 'Description') as Description,
    Attribute(FileLongName, 'Rating') as Rating,
    Attribute(FileLongName, 'PlayDuration') as PlayDuration,
    Attribute(FileLongName, 'MaximumBitrate') as MaximumBitrate,
    Attribute(FileLongName, 'WMFSDKVersion') as WMFSDKVersion,
    Attribute(FileLongName, 'WMFSDKNeeded') as WMFSDKNeeded,
    Attribute(FileLongName, 'IsVBR') as IsVBR,
    Attribute(FileLongName, 'ASFLeakyBucketPairs') as ASFLeakyBucketPairs,
    Attribute(FileLongName, 'PeakValue') as PeakValue,
    Attribute(FileLongName, 'AverageLevel') as AverageLevel;
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels);
Next vFoundFile
Next vExt
```

Example 4: PNG

Det här skriptet läser alla möjliga PNG-metataggar i mappen *MyPictures*.

```
// Script to read PNG meta tags
for each vExt in 'png'
for each vFoundFile in filelist( GetFolderPath('MyPictures') & '\*.'& vExt )
FileList:
LOAD FileLongName,
    subfield(FileLongName,'\',-1) as FileShortName,
    num(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,
    FileTime(FileLongName) as FileTime,
    Attribute(FileLongName, 'Comment') as Comment,
    Attribute(FileLongName, 'Creation Time') as Creation_Time,
    Attribute(FileLongName, 'Source') as Source,
    Attribute(FileLongName, 'Title') as Title,
    Attribute(FileLongName, 'Software') as Software,
    Attribute(FileLongName, 'Author') as Author,
    Attribute(FileLongName, 'Description') as Description,
    Attribute(FileLongName, 'Copyright') as Copyright;
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels);
Next vFoundFile
Next vExt
```

ConnectString

ConnectString()-funktionen returnerar namnet på den aktiva datakopplingen för ODBC- eller OLE DB -kopplingar. Funktionen returnerar en tom sträng om ingen **connect**-sats har exekverats eller efter en **disconnect**-sats.

Syntax:

```
ConnectString()
```

Exempel och resultat:

Skriptexempel

Exempel	Resultat

`LIB CONNECT TO 'Tutorial ODBC';`
 	
`ConnectString:`
 	
`Load ConnectString() as ConnectString AutoGenerate 1;`
	Returnerar Tutorial ODBC i fältet ConnectString.
	
	I det här exemplet antas att du har en tillgänglig datakoppling som heter Tutorial ODBC.

FileBaseName

FileBaseName-funktionen returnerar en textsträng som anger namnet på den tabellfil som håller på att läsas in, dock utan sökväg och filtillägg.

Syntax:

```
FileBaseName()
```

Exempel och resultat:

Skriptexempel

Exempel	Resultat

`LOAD *, filebasename( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Returnerar abc i fält X i varje post som läses in.

FileDir

FileDir-funktionen returnerar en textsträng som anger sökvägen till katalogen som innehåller den tabellfil som håller på att läsas in.

Syntax:

```
FileDir()
```

Funktionen har enbart stöd för mappdatakopplingar i standardläget.

Exempel och resultat:

Skriptexempel

Exempel	Resultat

`Load *, filedir( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Returnerar 'C:\UserFiles' i fält X i varje post som läses in.

FileExtension

FileExtension-funktionen returnerar en textsträng som anger filtillägget till den tabellfil som håller på att läsas in.

Syntax:

```
FileExtension()
```

Exempel och resultat:

Skriptexempel

Exempel	Resultat

`LOAD *, FileExtension( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Returnerar 'txt' i fält X i varje post som läses in.

FileName

FileName-funktionen returnerar en textsträng som anger namnet på den tabellfil som håller på att läsas in, utan sökväg men med filtillägget.

Syntax:

```
FileName()
```

Exempel och resultat:

Skriptexempel

Exempel	Resultat

`LOAD *, FileName( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Returnerar 'abc.txt' i fält X i varje post som läses in.

FilePath

FilePath-funktionen returnerar en textsträng som anger den kompletta sökvägen till den tabellfil som håller på att läsas in.

Syntax:

```
FilePath()
```

Funktionen har enbart stöd för mappdatakopplingar i standardläget.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`Load *, FilePath( ) as X from C:\UserFiles\abc.txt`	Returnerar 'C:\UserFiles\abc.txt' i fält X i varje post som läses in.

FileSize

FileSize-funktionen returnerar ett heltal som anger storleken i byte för filen filename eller, om inget filename angetts, för den tabellfil som håller på att läsas in.

FileSize([filename]**)**

Argument

Argument	Beskrivning
filename	Namnet på en fil, vid behov inklusive sökväg, som en datakoppliing för en mapp eller webbfil. Om du inte anger ett filnamn används den tabellfil som läses in för tillfället. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense applikationens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`LOAD *, FileSize( ) as X from abc.txt;`	Returnerar den angivna filens (abc.txt) storlek som ett heltal i fält X i varje post som läses in.
`FileSize( 'lib://DataFiles/xyz.xls' )`	Returnerar storleken på filen xyz.xls.

FileTime

FileTime-funktionen returnerar en tidsmarkör i UTC-format för den senaste ändringen av en angiven fil. Om ingen fil anges returnerar funktionen en tidsmarkör i UTC för den senaste ändringen av den tabellfil som för närvarande är inläst.

```
FileTime([ filename ])
```

Argument

Argument	Beskrivning
filename	Namnet på en fil, vid behov inklusive sökväg, som en mapp eller webbfildatakoppling. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense applikationens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`LOAD *, FileTime( ) as X from abc.txt;`	Returnerar tidsmarkören för den senaste ändringen av filen (abc.txt) i fältet X i varje läst post.
`FileTime( 'xyz.xls' )`	Returnerar datum och tid för den senaste ändringen av filen xyz.xls.

GetFolderPath

Funktionen **GetFolderPath** returnerar värdet av funktionen Microsoft Windows *SHGetFolderPath*. Den här funktionen tar som indata namnet på en Microsoft Windows -mapp och returnerar den fullständiga sökvägen till mappen.

Funktionen stöds inte i standardläget. .

```
GetFolderPath(foldername)
```

Argument

Argument	Beskrivning
foldername	Namnet på mappen Microsoft Windows. Mappnamnet får inte innehålla några blanksteg. Eventuellt blanksteg i mappnamnet som syns i Windows Explorer ska tas bort från mappnamnet. Exempel: *MyMusic* *MyDocuments*

Målet med det här exemplet är att få fram sökvägarna till Microsoft Windows följande *MyMusic*, *MyPictures* och *Windows*-mappar: Lägg till exempelskriptet i din applikation och ladda om den.

```
LOAD
 GetFolderPath('MyMusic') as MyMusic,
 GetFolderPath('MyPictures') as MyPictures,
 GetFolderPath('Windows') as Windows
AutoGenerate 1;
```

När applikationen laddas om läggs fälten *MyMusic*, *MyPictures* och *Windows* till i datamodellen. Varje fält innehåller sökvägen till den mapp som har definierats i indata. Exempel:

- *C:\Users\smu\Music* for the folder *MyMusic*
- *C:\Users\smu\Pictures* for the folder *MyPictures*
- *C:\Windows* for the folder *Windows*

QvdCreateTime

Denna skriptfunktion returnerar XML-huvudets tidsstämpel från en QVD-fil, om någon sådan finns, annars returneras NULL. I tidsmarkören anges tiden i UTC.

```
QvdCreateTime(filename)
```

Argument

Argument	Beskrivning
filename	Namnet på en QVD-fil, vid behov inklusive sökväg, exempelvis en mapp eller webbdatakoppling. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense applikationens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*

```
QvdCreateTime('MyFile.qvd')
QvdCreateTime('C:\MyDir\MyFile.qvd')
QvdCreateTime('lib://DataFiles/MyFile.qvd')
```

QvdFieldName

Denna skriptfunktion returnerar namnet på fält nummer **fieldno** i en QVD-fil. Om fältet inte finns returneras NULL.

```
QvdFieldName(filename , fieldno)
```

Argument

Argument	Beskrivning
filename	Namnet på en QVD-fil, vid behov inklusive sökväg, exempelvis en mapp eller webbdatakoppling. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense applikationens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*
fieldno	Fält-numret i den tabell som finns i QVD-filen.

```
QvdFieldName ('MyFile.qvd', 5)
QvdFieldName ('C:\MyDir\MyFile.qvd', 5)
QvdFieldName ('lib://DataFiles/MyFile.qvd', 5)
```

Alla tre exemplen returnerar namnet på det femte fältet i tabellen som finns i QVD-filen.

QvdNoOfFields

Denna skriptfunktion returnerar antalet fält i en QVD-fil.

```
QvdNoOfFields(filename)
```

Argument

Argument	Beskrivning
filename	Namnet på en QVD-fil, vid behov inklusive sökväg, exempelvis en mapp eller webbdatakoppling. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense applikationens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*

```
QvdNoOfFields ('MyFile.qvd')
QvdNoOfFields ('C:\MyDir\MyFile.qvd')
QvdNoOfFields ('lib://DataFiles/MyFile.qvd')
```

QvdNoOfRecords

Exempel: Denna skriptfunktion returnerar aktuellt antal poster i en QVD-fil.

```
QvdNoOfRecords(filename)
```

Argument

Argument	Beskrivning
filename	Namnet på en QVD-fil, vid behov inklusive sökväg, exempelvis en mapp eller webbdatakoppling. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense applikationens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*

```
QvdNoOfRecords ('MyFile.qvd')
QvdNoOfRecords ('C:\MyDir\MyFile.qvd')
QvdNoOfRecords ('lib://DataFiles/MyFile.qvd')
```

QvdTableName

Denna skriptfunktion returnerar namnet på tabellen i en QVD-fil.

```
QvdTableName(filename)
```

Argument

Argument	Beskrivning
filename	Namnet på en QVD-fil, vid behov inklusive sökväg, exempelvis en mapp eller webbdatakoppling. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense applikationens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*

```
QvdTableName ('MyFile.qvd')
QvdTableName ('C:\MyDir\MyFile.qvd')
QvdTableName ('lib://data\MyFile.qvd')
```

8.11 Finansiella funktioner

Finansiella funktioner kan användas i dataladdningsskriptet och diagramuttryck för att beräkna betalningar och räntesatser.
I alla parametrar betecknar negativa tal pengar som man betalar ut. Positiva tal betecknar pengar som erhålls.
I listan visas de parametrar som används i de finansiella funktionerna (förutom dem som börjar med **range**-).

> *För alla finansiella funktioner är det mycket viktigt att vara konsekvent vid angivande av enheter för **rate** och **nper**. Om man gör en månatlig avbetalning på ett femårigt lån med en årlig räntesats på 6%, bör man använda 0.005 (6%/12) för **rate** och 60 (5*12) för **nper**. Om man gör en årlig avbetalning på samma lån, bör man använda 6% för **rate** och 5 för **nper**.*

Finansiella funktioner – en översikt

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

FV

Denna funktion returnerar det framtida värdet av en investering baserat på periodiska, återkommande inbetalningar och en enkel årsränta.

```
FV (rate, nper, pmt [ ,pv [ , type ] ])
```

nPer

Denna funktion returnerar antalet perioder för en investering baserad på periodiska, konstanta inbetalningar och en konstant räntesats.

```
nPer (rate, pmt, pv [ ,fv [ , type ] ])
```

Pmt

Denna funktion returnerar inbetalningsbeloppet för ett lån baserat på periodiska, konstanta inbetalningar och en konstant räntesats. Den kan inte ändras under annuitetens löptid. En betalning anges som ett negativt tal, till exempel -20.

```
Pmt (rate, nper, pv [ ,fv [ , type ] ] )
```

PV

Denna funktion returnerar det aktuella värdet av en investering.

```
PV (rate, nper, pmt [ ,fv [ , type ] ])
```

Rate

Denna funktion returnerar räntesatsen per period i en annuitet. Resultatet anges i ett fördefinierat talformat: **Fix**, två decimaler och %.

```
Rate (nper, pmt , pv [ ,fv [ , type ] ])
```

BlackAndSchole

Black and Scholes-modellen är en matematisk modell för finansmarknadsanalyser. Formeln beräknar det teoretiska värdet av en option. I Qlik Sense returnerar **BlackAndSchole**-funktionen värdet enligt den omodifierade Black and Scholes-formeln (europeisk optionsmodell).

```
BlackAndSchole(strike , time_left , underlying_price , vol , risk_free_rate , type)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
strike	Aktiens framtida inköpspris.
time_left	Antalet återstående perioder.
underlying_ price	Aktiens aktuella värde.
vol	Volatiliteten (för aktiekurser) uttryckt som procentandel i decimalform, per tidsperiod.
risk_free_rate	Den riskfria räntan uttryckt som procentandel i decimalform, per tidsperiod.
call_or_put	Typ av option: 'c', 'call' eller vilket numeriskt värde som helst, utom noll, för köpoptioner 'p', 'put' eller 0 för säljoptioner.

Begränsningar:

Värdet på strike, time_left och underlying_price måste vara >0.

Värdet på vol och risk_free_rate måste vara: <0 eller >0.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`BlackAndSchole(130, 4, 68.5, 0.4, 0.04, 'call')` Detta beräknar det teoretiska priset på en option som köps på 4 år till ett värde av 130 per aktie, vilket idag motsvarar 68,5. Formeln använder volatiliteten 0,4 (40 %) per år och en riskfri ränta på 0,04 (4 %).	Returnerar 11,245

FV

Denna funktion returnerar det framtida värdet av en investering baserat på periodiska, återkommande inbetalningar och en enkel årsränta.

Syntax:

```
FV(rate, nper, pmt [ ,pv [ , type ] ])
```

Returnerad datatyp: numeriska. Som standard formateras resultatet som valuta..

Argument:

Argument

Argument	Beskrivning
rate	Räntesatsen per period.
nper	Det totala antalet inbetalningsperioder i en annuitet.
pmt	Den inbetalning som görs varje period. Den kan inte ändras under annuitetens löptid. En betalning anges som ett negativt tal, till exempel -20.
pv	Det aktuella värdet (present value), eller den klumpsumma som en serie framtida inbetalningar för närvarande är värda. Om **pv** utelämnas, förutsätts 0 (noll).
type	Ska sättas till 0 om inbetalningarna ska göras i slutet av perioden och 1 om inbetalningarna ska göras i början av perioden. Om **type** utelämnas, förutsätts 0.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
Du betalar av en ny hushållsmaskin med 36 månadsavbetalningar på 20 $. Räntesatsen är 6 % per år. Räkningen kommer i slutet av månaden. Hur mycket har du investerat totalt när den sista räkningen betalats? `FV(0.005,36,-20)`	Returnerar $786.72

nPer

Denna funktion returnerar antalet perioder för en investering baserad på periodiska, konstanta inbetalningar och en konstant räntesats.

Syntax:

```
nPer(rate, pmt, pv [ ,fv [ , type ] ])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
rate	Räntesatsen per period.
nper	Det totala antalet inbetalningsperioder i en annuitet.
pmt	Den inbetalning som görs varje period. Den kan inte ändras under annuitetens löptid. En betalning anges som ett negativt tal, till exempel -20.
pv	Det aktuella värdet (present value), eller den klumpsumma som en serie framtida inbetalningar för närvarande är värda. Om **pv** utelämnas, förutsätts 0 (noll).
fv	Det framtida värdet, eller den kassabehållning man vill uppnå efter att den sista räkningen betalats. Om **fv** utelämnas, förutsätts 0.
type	Ska sättas till 0 om inbetalningarna ska göras i slutet av perioden och 1 om inbetalningarna ska göras i början av perioden. Om **type** utelämnas, förutsätts 0.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
Du vill sälja en hushållsmaskin med månadsavbetalningar på 200 kronor. Räntesatsen är 6 % per år. Räkningen kommer i slutet av månaden. Hur många perioder krävs för att värdet på pengarna du erhållit efter att den sista räkningen betalats ska vara 800 $? `nPer(0.005,-20,0,800)`	Returnerar 36.56

Pmt

Denna funktion returnerar inbetalningsbeloppet för ett lån baserat på periodiska, konstanta inbetalningar och en konstant räntesats. Den kan inte ändras under annuitetens löptid. En betalning anges som ett negativt tal, till exempel -20.

```
Pmt(rate, nper, pv [ ,fv [ , type ] ] )
```

Returnerad datatyp: numeriska. Som standard formateras resultatet som valuta..

För att räkna ut det totala beloppet som betalats in under lånets löptid multiplicerar du det returnerade **pmt**-värdet med **nper**.

Argument:

Argument

Argument	Beskrivning
rate	Räntesatsen per period.
nper	Det totala antalet inbetalningsperioder i en annuitet.
pv	Det aktuella värdet (present value), eller den klumpsumma som en serie framtida inbetalningar för närvarande är värda. Om **pv** utelämnas, förutsätts 0 (noll).
fv	Det framtida värdet, eller den kassabehållning man vill uppnå efter att den sista räkningen betalats. Om **fv** utelämnas, förutsätts 0.
type	Ska sättas till 0 om inbetalningarna ska göras i slutet av perioden och 1 om inbetalningarna ska göras i början av perioden. Om **type** utelämnas, förutsätts 0.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
Följande formel returnerar det månatliga inbetalningsbeloppet för ett lån på 20 000 kronor som ska vara avbetalat på 8 månader och där räntesatsen är 10%: `Pmt(0.1/12,8,20000)`	Returnerar -$2,594.66
Om inbetalningarna för samma lån ska göras i början av perioden blir inbetalningsbeloppet följande: `Pmt(0.1/12,8,20000,0,1)`	Returnerar -$2,573.21

PV

Denna funktion returnerar det aktuella värdet av en investering.

```
PV(rate, nper, pmt [ ,fv [ , type ] ])
```

Returnerad datatyp: numeriska. Som standard formateras resultatet som valuta..

Det aktuella värdet är det totala belopp som en serie framtida inbetalningar för närvarande är värda. Om man exempelvis lånar pengar, är lånebeloppet det aktuella värdet för den som lånar ut.

Argument:

Argument

Argument	Beskrivning
rate	Räntesatsen per period.
nper	Det totala antalet inbetalningsperioder i en annuitet.
pmt	Den inbetalning som görs varje period. Den kan inte ändras under annuitetens löptid. En betalning anges som ett negativt tal, till exempel -20.
fv	Det framtida värdet, eller den kassabehållning man vill uppnå efter att den sista räkningen betalats. Om **fv** utelämnas, förutsätts 0.
type	Ska sättas till 0 om inbetalningarna ska göras i slutet av perioden och 1 om inbetalningarna ska göras i början av perioden. Om **type** utelämnas, förutsätts 0.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
Vilket är det aktuella värdet av en skuld som du betalar av med 100 $ i slutet av varje månad under en femårsperiod, om räntan är sju procent? `PV(0.07/12,12*5,-100,0,0)`	Returnerar $5,050.20

Rate

Denna funktion returnerar räntesatsen per period i en annuitet. Resultatet anges i ett fördefinierat talformat: **Fix**, två decimaler och %.

Syntax:

```
Rate(nper, pmt , pv [ ,fv [ , type ] ])
```

Returnerad datatyp: numeriska.

rate beräknas med hjälp av iteration och kan ha noll eller fler lösningar. Om de successiva resultaten av **rate** inte konvergerar, returneras ett NULL-värde.

Argument:

Argument

Argument	Beskrivning
nper	Det totala antalet inbetalningsperioder i en annuitet.
pmt	Den inbetalning som görs varje period. Den kan inte ändras under annuitetens löptid. En betalning anges som ett negativt tal, till exempel -20.

Argument	Beskrivning
pv	Det aktuella värdet (present value), eller den klumpsumma som en serie framtida inbetalningar för närvarande är värda. Om **pv** utelämnas, förutsätts 0 (noll).
fv	Det framtida värdet, eller den kassabehållning man vill uppnå efter att den sista räkningen betalats. Om **fv** utelämnas, förutsätts 0.
type	Ska sättas till 0 om inbetalningarna ska göras i slutet av perioden och 1 om inbetalningarna ska göras i början av perioden. Om **type** utelämnas, förutsätts 0.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
Vilken är räntesatsen på ett femårigt annuitetslån på 10 000 SEK med månatliga inbetalningar på 300 SEK? `Rate(60,-300,10000)`	Returnerar 2.00%

8.12 Formateringsfunktioner

Formateringsfunktionerna skapar ett visningsformat för de numeriska fält eller uttryck som matas in. Beroende på datatyp kan du ange tecken för decimalavgränsare, tusenavgränsare och så vidare.

Funktionerna returnerar ett dualt värde med både strängen och det numeriska värdet, men det kan också ses som att de gör en konvertering från tal till sträng. **Dual()** är ett specialfall, men de andra formateringsfunktionerna tar det numeriska värdet av indatauttrycken och genererar en sträng som representerar numret.

Tolkningsfunktionerna däremot gör det motsatta: de tar stränguttryck och utvärderar dem som tal och specificerar sedan format för det resulterande talet.

Funktionerna kan användas både i dataladdningsskript och diagramuttryck.

Alla numeriska värden anges med en decimalpunkt som decimalavgränsare.

Formateringsfunktioner - en översikt

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

ApplyCodepage

ApplyCodepage() tillämpar en annan teckenuppsättning för kodsidan på fältet eller texten som angetts i uttrycket. Argumentet **codepage** måste skrivas i talformat.

```
ApplyCodepage (text, codepage)
```

Date

Date() formaterar ett uttryck som ett datum med hjälp av formatet som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, eller i en formatsträng, om så är tillämpligt.

```
Date (number[, format])
```

Dual

Dual() kombinerar ett tal och en sträng till en enda post, så att talåtergivningen av posten kan användas för sortering och beräkning, samtidigt som strängvärdet kan användas för visning.

```
Dual (text, number)
```

Interval

Interval() formaterar ett tal som ett tidsintervall med hjälp av formatet som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, eller i en formatsträng, om så är tillämpligt.

```
Interval (number[, format])
```

Money

Money() formaterar ett uttryck numeriskt som ett valutavärde i det format som har ställts in i systemvariablerna i dataladdningsskriptet, eller i operativsystemet, om inte en formatsträng finns. Den formaterar även valbara decimal- och tusentalsavgränsare.

```
Money (number[, format[, dec_sep [, thou_sep]]])
```

Num

Num() formaterar ett tal, vilket innebär att det numeriska värdet i indata konverteras och visas med det format som specificerats i den andra parametern. Om den andra parametern utelämnas använder funktionen de decimal- och tusentalsavgränsare som anges i dataladdningsskriptet. Egna symboler för decimaler och tusentalsseparatorer är valbara parametrar.

```
Num (number[, format[, dec_sep [, thou_sep]]])
```

Time

Time() formaterar ett uttryck som ett tidsvärde, i det tidsformat som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns.

```
Time (number[, format])
```

Timestamp

TimeStamp() formaterar ett uttryck som ett datum och tid-värde, i det format för tidsmarkörer som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns.

```
Timestamp (number[, format])
```

Se även:

p *Tolkningsfunktioner (page 1430)*

ApplyCodepage

ApplyCodepage() tillämpar en annan teckenuppsättning för kodsidan på fältet eller texten som angetts i uttrycket. Argumentet **codepage** måste skrivas i talformat.

> *ApplyCodepage kan användas i diagramuttryck men är oftast en skriptfunktion i Skriptredigeraren. När du exempelvis laddar filer som kan ha sparats med andra teckenuppsättningar kan du tillämpa kodsidan med den teckenuppsättning som du behöver.*

Syntax:

```
ApplyCodepage(text, codepage)
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Fält eller text som du vill använda en annan kodsida för, som fås av argumentet **codepage**.
codepage	Tal som representerar kodsidan som ska tillämpas på fältet eller uttrycket som fås med **text**.

Exempel: laddningsskript

Exempel

```
LOAD
ApplyCodepage(ROWX,1253) as GreekProduct,
ApplyCodepage (ROWY, 1255) as HebrewProduct,
ApplyCodepage (ROWZ, 65001) as EnglishProduct;
SQL SELECT ROWX, ROWY, ROWZ From Products;
```

Resultat

Vid inläsning från SQL kan källan ha en blandning av olika teckenuppsättningar: Kyrilliska, hebreiska med mera från UTF-8-formatet. Dessa måste läsas in rad för rad och olika kodsidor måste tillämpas för varje rad. **codepage**-värdet 1253 motsvarar Windows grekiska teckenuppsättning, värdet 1255 representerar hebreiska och värdet 65001 representerar vanliga latinska UTF-8-tecken.

Se även: *Teckenuppsättning (page 191)*

Date

Date() formaterar ett uttryck som ett datum med hjälp av formatet som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, eller i en formatsträng, om så är tillämpligt.

Syntax:

```
Date(number[, format])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
number	Talet som ska formateras.
format	En sträng som beskriver formatet på den sträng som blir resultatet. Om ingen formatsträng anges används det datumformat som har definierats i systemvariablerna i det dataladdningsskript eller operativsystem som används. Funktionen `Date` har stöd för många formatalternativ för olika datumstrukturer, t.ex: `DD/MM/YYYY`: Dag/Månad/År (t.ex. 15/02/2022) `MM-DD-YYYY`: Månad-Dag-År med bindestreck (t.ex. 02-15-2022) `YYYY-MM-DD`: År-Månad-Dag, ISO-format (t.ex. 2022-02-15) `D-MMM-YYYY`: Dag, kort månadsnamn, År, med bindestreck (t.ex. 15-Feb-2022) `MMMM YYYY`: Långt månadsnamn och År (t.ex. February 2022) `MM/YYYY`: Månad och År (t.ex. 02/2022) `DD MMMM YYYY`: Dag, långt månadsnamn och År (t.ex. 15 February 2022)

Exempel: diagramuttryck

Exempel	Resultat
`Date( 35648 )`	Returnerar `97-08-06` när inställningen för `DateFormat` är `YY-MM-DD`.
`Date( 35648, 'YY.MM.DD' )`	Returnerar `97.08.06` när inställningen för `DateFormat` är `YY-MM-DD`.
`Date( '8/6/97', 'YY.MM.DD' )`	Returnerar `NULL` när inställningen för `DateFormat` är `YY-MM-DD`.

Exempel – grunderna i datum

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `RowNo`
  - `InputDate`
- Variabeln `DateFormat`, som använder formatet `YY-MM-DD`.

Laddningsskript

```
SET DateFormat='YY-MM-DD';
Example:
Load * inline [
RowNo, InputDate
1, 35648
2, 35648.375
3, 8/6/97
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `RowNo`
- `InputDate`

Skapa följande mått:

- `=Date(InputDate)`, för att visa värdet som ett datum som formateras med hjälp av inställningarna för variabeln `DateFormat`.
- `=Date(InputDate, 'DD.MM.YYYY')`om du vill visa värdet som ett datum som är formaterat med det format som anges i funktionen.
- `=Date(InputDate, 'YY.MM.DD')`om du vill visa värdet som ett datum som är formaterat med det format som anges i funktionen.
- `=Num(InputDate)`, för att returnera det numeriska värdet för det första tecknet.

Results table

RowNo	InputDate	Date (InputDate)	Date(InputDate, 'DD.MM.YYYY')	Date(InputDate, 'YY.MM.DD')	Num (InputDate)
1	35648	97-08-06	06.08.1997	97.08.06	35648
2	35648.375	97-08-06	06.08.1997	97.08.06	35648.375
3	8/6/97	-	-	-	-

Resultaten visar hur funktionen `Date` kan användas för att visa och formatera datum. I det här exemplet anges `DateFormat` som `YY-MM-DD` i laddningsskriptet. Systemet förväntar sig att uttrycket för datumet ska använda detta format. Du kan se att värdet för `InputDate` i rad 3 (`8/6/97`) inte följer detta format, därför känner systemet inte igen det som ett datum och returnerar `NULL`.

Ändra inställningen `DateFormat` i laddningsskriptet till `SET DateFormat='M/D/YY';` och ladda om data. Värdet på rad 3 kommer nu att överensstämma med det förväntade formatet. Systemet känner igen det som ett giltigt datum och formaterar det i enlighet med detta. Följande tabell visar resultaten med den nya inställningen `DateFormat`.

Results table with variable statement `Set DateFormat='M/D/YY'`

RowNo	InputDate	Date (InputDate)	Date(InputDate, 'DD.MM.YYYY')	Date(InputDate, 'YY.MM.DD')	Num (InputDate)
1	35648	97-08-06	06.08.1997	97.08.06	35648
2	35648.375	97-08-06	06.08.1997	97.08.06	35648.375
3	35648	97-08-06	06.08.1997	97.08.06	35648.375

Exempel – Datumscenario

Diagramuttryck

Översikt

En datauppsättning innehåller försäljningsdata som använder datumformatet `MM-DD-YYYY`. Du vill använda datumformatet `DD/MM/YYYY`.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Customer`
  - `SalesDate`
  - `Amount`
- Variabeln `DateFormat`, som använder formatet `MM/DD/YYYY`.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';
Example:
Load * inline [
Customer, SalesDate, Amount
Customer A, 2/15/2024, 100008
Customer B, 9/30/2024, 12500
Customer C, 12/28/2024, 15000
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=Date(SalesDate ,'DD/MM/YYYY')` om du vill visa värdet som ett datum som är formaterat med det format som anges i funktionen.
- `=Sum(Amount)`, för att beräkna det totala värdet av kostnaden för sålda varor.

Results table

Customer	Date(SalesDate ,'DD/MM/YYYY')	Sum(Amount)
Customer A	15/02/2024	100008
Customer B	30/09/2024	12500
Customer C	28/12/2024	15000

När du tittar på resultaten kan du se att även om datumen i datauppsättningen hade ett visst format, kan du med funktionen `Date` presentera datumen i ett annat format.

Dual

Dual() kombinerar ett tal och en sträng till en enda post, så att talåtergivningen av posten kan användas för sortering och beräkning, samtidigt som strängvärdet kan användas för visning.

Syntax:

```
Dual(text, number)
```

Returnerad datatyp: dual

Alla duala returvärden är högerjusteradee.

Argument

Argument	Beskrivning
text	Strängvärdet som ska användas i kombination med talargumentet.
number	Talet som ska användas i kombination med strängen i strängargumentet.

I Qlik Sense är alla fältvärden potentiellt duala värden. Detta innebär att fältvärdena både kan ha ett numeriskt värde och ett textvärde. Exempelvis kan ett datum ha det numeriska värdet 40908 och textrepresentationen '2011-12-31'.

När flera dataelement som laddas i ett fält har olika textsträngar men samma numeriska värde, delar de alla den första påträffade textsträngen.

***dual**-funktionen används vanligen tidigt i skriptet, innan andra data lästs in i det berörda fältet, för att skapa denna första strängrepresentation som kommer att visas i filterrutor.*

Exempel: diagramuttryck

Exempel	Beskrivning
`Dual(IF(DayOfWeek = 0, 'Monday', IF(DayOfWeek = 1, 'Tuesday', If(DayOfWeek = 2 ,'Wednesday', If(DayOfWeek = 3 ,'Thursday', If(DayOfWeek = 4 ,'Friday', If (DayOfWeek = 5 ,'Saturday', If (DayOfWeek = 6 ,'Sunday' ))))))), DayOfWeek)`	Fältet `DayOfWeek` kan användas i en visualisering som en dimension. I en tabell sorteras veckodagarna automatiskt i rätt nummerordning, istället för i alfabetisk ordning.
`Dual(Pick( Priority , 'Low','Medium','High'), Priority)`	Fältet `Priority` kan användas i en visualisering som en dimension. I en tabell sorteras fältet `Priority` automatiskt i rätt nummerordning, istället för i alfabetisk ordning.
`Dual('Q' & Ceil(Month(Date)/3), Ceil(Month(Date)/3))`	Fältet `Date` kan användas i en visualisering som en dimension. Textvärdena `Q1` till `Q4` skapas och tilldelas de numeriska värdena `1` till `4`.
`Dual(WeekYear(Date) & '-W' & Week(Date), WeekStart(Date))`	Fältet `Date` kan användas i en visualisering som en dimension. Det här exemplet skapar textvärden med formen `'2012-W22'` och tilldelar samtidigt ett numeriskt värde som motsvarar datumtal för veckans första dag, exempelvis: `41057`

Exempel: Laddningsskript

Exempel	Beskrivning

Lägg till följande exempel i skriptet och kör det.
`Load dual ( NameDay,NumDay ) as DayOfWeek inline`
`[ NameDay,NumDay`
`Monday,0`
`Tuesday,1`
`Wednesday,2`
`Thursday,3`
`Friday,4`
`Saturday,5`
`Sunday,6 ];`
	Fältet `DayOfWeek` kan t.ex. användas i en visualisering som en dimension. I en tabell sorteras veckodagarna automatiskt i rätt nummerordning, istället för i alfabetisk ordning.

`Load Dual('Q' & Ceil (Month(Now())/3), Ceil (Month(Now())/3)) as Quarter AutoGenerate 1;`	Det här exemplet letar rätt på aktuellt kvartal. Det visas som `Q1` när funktionen **Nu** körs under de första tre månaderna på året, `Q2` för de följande tre månaderna och så vidare. Om du använder sortering kommer dock fältet `Quarter` att fungera som sitt numeriska värde 1 till 4:
`Dual('Q' & Ceil(Month (Date)/3), Ceil(Month (Date)/3)) as Quarter`	Som i tidigare exempel skapas fältet `Quarter` med textvärdena `Q1` till `Q4`, och tilldelas de numeriska värdena `1` till `4`. För att kunna använda detta i skriptet måste värdena för `Date` laddas.
`Dual(WeekYear(Date) & '-W' & Week(Date), WeekStart(Date)) as YearWeek`	Det här exemplet skapar ett fält `YearWeek` med textvärden i formen `2012-W22` och tilldelar samtidigt ett numeriskt värde som motsvarar datumnumret för veckans första dag, exempelvis: `41057`. För att kunna använda detta i skriptet måste värdena för `Date` laddas.

Exempel – grunderna i dual

Diagramuttryck

Översikt

En datauppsättning innehåller kundtjänstdata. Du vill granska antalet samtal per veckodag.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `DayOfWeek`
    - `NoOfCalls`

Laddningsskript

```
Example:
Load * inline [
DayOfWeek, NoOfCalls
0, 25000
1, 30000
2, 27500
3, 32000
4, 23250
5, 28950
6, 32350
];
```

Resultat

Ladda data och öppna ett ark. Skapa ett nytt stapeldiagram och lägg sedan till en dimension med hjälp av någon av följande metoder:

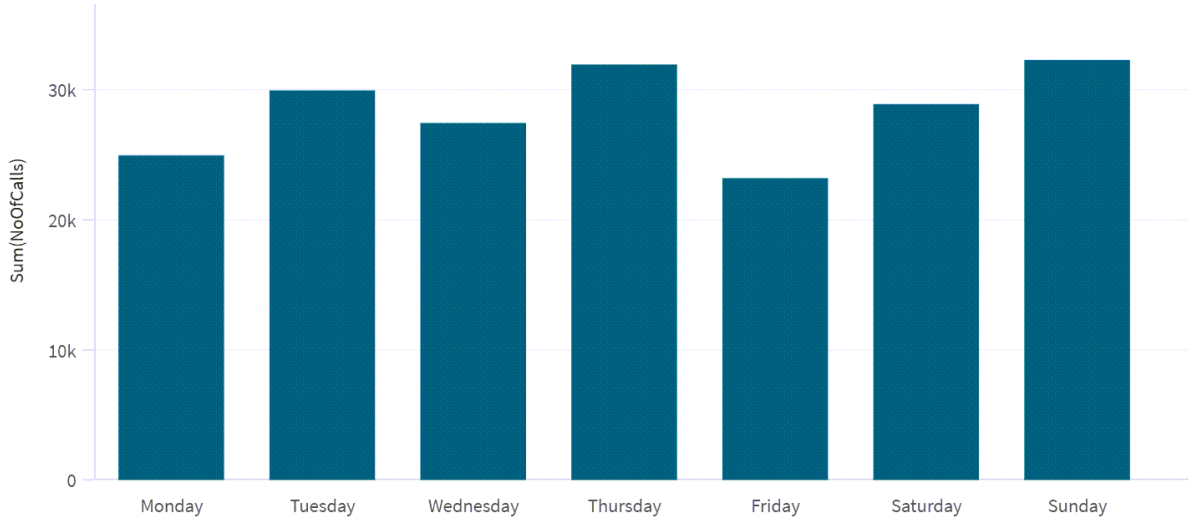
a. I uttrycket för dimensionen lägger du till följande:
```
=Dual(IF(DayOfWeek = 0, 'Monday',
IF(DayOfWeek = 1, 'Tuesday',
If(DayOfWeek = 2 ,'Wednesday',
If(DayOfWeek = 3 ,'Thursday',
If(DayOfWeek = 4 ,'Friday',
If(DayOfWeek = 5 ,'Saturday',
If(DayOfWeek = 6 ,'Sunday'
)))))), DayOfWeek)
```

b. Denna metod är mer effektiv. I uttrycket för dimensionen lägger du till följande:
```
=Dual(pick( DayOfWeek +1 , 'Monday','Tuesday','Wednesday', 'Thursday','Friday',
'Saturday','Sunday' ), DayOfWeek)
```

Skapa följande mått:

- `=Sum(NoOfCalls)`, för att beräkna antalet samtal.

Resultat: stapeldiagram som visar antalet samtal per veckodag



Stapeldiagrammet visar antalet samtal per veckodag. Det sorterar automatiskt veckodagarna i rätt ordning.

Exempel – Dual-scenario

Diagramuttryck

Översikt

En datauppsättning innehåller feedback från kundtjänst. Du vill sortera och visa feedbacken så att de negativa svaren visas först, följt av de positiva svaren.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ResponseType`
  - `NoOfResponses`

Laddningsskript

```
Example:
Load * inline [
ResponseType, NoOfResponses
1, 350
2, 560
3, 4350
4, 2350
5, 5350
];
```

Resultat

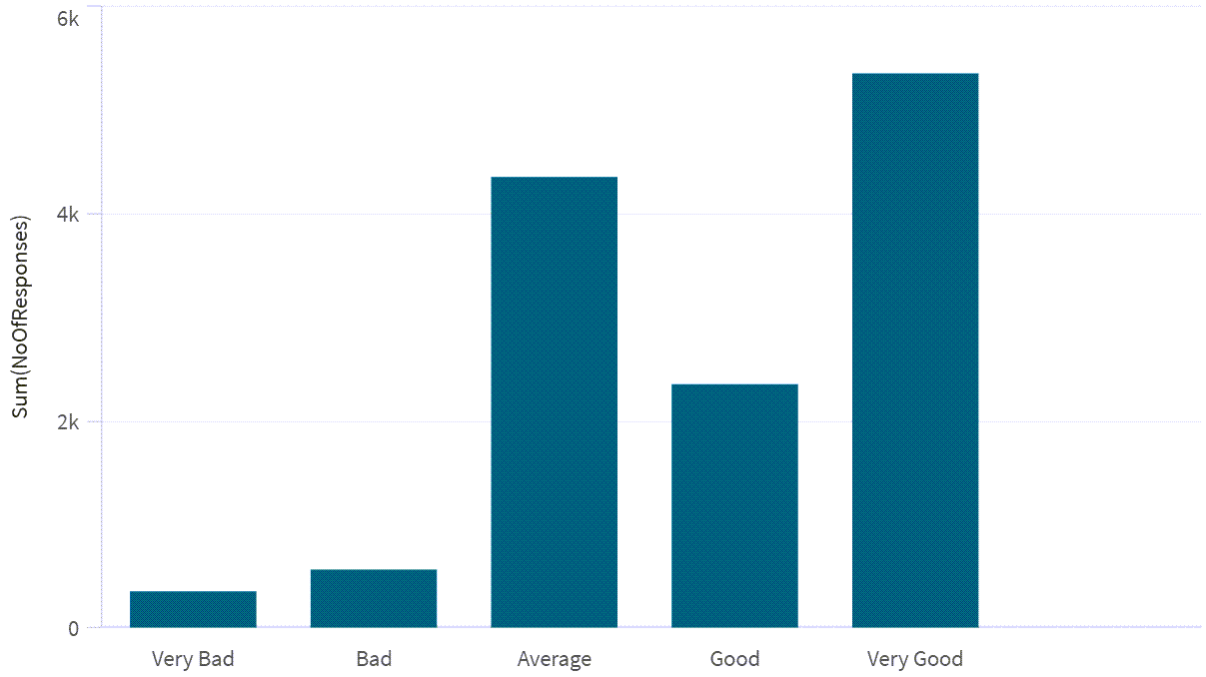
Ladda data och öppna ett ark. Skapa ett nytt stapeldiagram och lägg till följande uttryck i en dimension.

- `=Dual(pick( ResponseType , 'Very Bad','Bad','Average', 'Good','Very Good'), ResponseType)`

Skapa följande mått:

- `=Sum(NoOfResponses)`, för att beräkna antalet svar.

Resultat: stapeldiagram som visar antalet svar per svarstyp



Stapeldiagrammet visar antalet svar per svarstyp. Observera att staplarna automatiskt sorteras från negativt till positivt. Svarstypen, som är ett numeriskt värde, har tilldelats en textbeskrivning.

Interval

Interval() formaterar ett tal som ett tidsintervall med hjälp av formatet som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, eller i en formatsträng, om så är tillämpligt.

Intervall kan formateras som tid, dagar eller som en kombination av dagar, timmar, minuter, sekunder och bråkdels sekunder.

Syntax:

```
Interval(number[, format])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
number	Talet som ska formateras.
format	Sträng som beskriver hur den resulterande intervallsträngen ska formateras. Om den utelämnas används operativsystemets datumformat och decimalavgränsare. Om den utelämnas används operativsystemets datumformat och decimalavgränsare. Funktionen `Interval` har stöd för många formatalternativ för olika tidsstrukturer, t.ex: `hh:mm` – bara timmar och minuter `hh:mm:ss` – timmar, minuter och sekunder `d hh:mm` – dagar, timmar och minuter

Exempel: diagramuttryck

Exempel	Resultat
`Interval( 0.375 )`	Returnerar `9:00:00` när `TimeFormat` är `h:mm:ss`.
`Interval(.375, 'hh:mm' )`	Returnerar `9:00` när formatet i funktionen är `hh.mm`.
`Interval( 0.375, 'd hh' )`	Returnerar `0 09` när formatet i funktionen är `d hh`.

Exempel – grunderna i intervall

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i en datatabell:
  - `IntervalValue`
- Variabeln `TimeFormat`, som använder formatet `h:mm:ss`.

Laddningsskript

```
SET TimeFormat='h:mm:ss';
Example:
Load * inline [
IntervalValue
0.375
1.375
```

```
35.648
36.875
368.75
468.95
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `IntervalValue`

Skapa följande mått:

- `=Interval(IntervalValue)` för att visa värdet som ett intervallformat med hjälp av systeminställningen.
- `=Interval(IntervalValue, 'd hh')` för att visa värdet som ett intervallformat med det format som anges i funktionen.
- `=Interval(IntervalValue, 'hh:mm')` för att visa värdet som ett intervallformat med det format som fås i funktionen.
- `=Interval(IntervalValue, 'd hh:mm')` för att visa värdet som ett intervallformat med det format som fås i funktionen.

Resultattabell

IntervalValue	Interval (IntervalValue)	Interval (IntervalValue, 'd hh')	Interval (IntervalValue, 'hh:mm'	Interval (IntervalValue, 'd hh:mm')
0.375	9:00:00	0 09	09:00	0 09:00
1.375	33:00:00	1 09	33:00	1 09:00
35.648	855:33:07	35 15	855:33	35 15:33
36.875	885:00:00	36 21	885:00	36 21:00
368,75	8850:00:00	368 18	8850:00	368 18:00
468.95	11254:48:00	468 22	11254:48	468 22:48

Resultaten visar hur `Interval`-funktionen visar värden enligt formatparametern i funktionen. Till exempel, när värdet `468.95`, i den sista raden, formateras med hjälp av funktionen `Interval` och formatparametern, `d hh:mm`, returnerar värdet `468` dagar, `22` timmar och `48` minuter.

Exempel – Intervallscenario

Diagramuttryck

Översikt

Supportärendedata beräknar lösningstiden per ärende i sekunder. Du vill omvandla detta värde till ett annat format.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `TicketID`
  - `ResolutionTime`

Laddningsskript

```
Example:
Load * inline [
TicketID, ResolutionTime
Ticket A, 4350
Ticket B, 560
Ticket C, 86400
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `TicketID`

Skapa följande mått:

- `=Interval(ResolutionTime / 86400, 'hh:mm:ss')` för att konvertera värdena till dagar, genom att dividera `ResolutionTime` med `86,400` och sedan formatera det med hjälp av det format som tillhandahålls i funktionen.

Resultattabell

TicketID	Intervall(ResolutionTime / 86400, 'hh:mm:ss')
Ärende A	01:12:30
Ärende B	00:09:20
Ärende C	24:00:00

De tidsvärden för upplösning som ursprungligen angavs i sekunder konverteras nu till en bråkdel av en dag och formateras sedan i timmar, minuter och sekunder. Tabellen visar t.ex. att `Ticket B` tog `9` timmar och `20` minuter att lösa.

Money

Money() formaterar ett uttryck numeriskt som ett valutavärde i det format som har ställts in i systemvariablerna i dataladdningsskriptet, eller i operativsystemet, om inte en formatsträng finns. Den formaterar även valbara decimal- och tusentalsavgränsare.

Syntax:

```
Money(number[, format[, dec_sep[, thou_sep]]])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
number	Talet som ska formateras.
format	Sträng som beskriver hur den resulterande valutasträngen ska formateras. Om det utelämnas används det värde som angetts i dataladdningsskriptet för `MoneyFormat`. Funktionen `Money` har stöd för många formatalternativ för olika datumstrukturer, t.ex: `$#,##0.00`: Lägger till ett dollartecken och två decimaler. `€ #,##0`: Lägger till ett eurotecken, inga decimaler och ett blanksteg efter eurotecknet. `¥#,##0`: Lägger till ett tecken för japanska yen utan decimaler. `$#,##0;($#,##0)`: Formaterar negativa värden inom parentes. `₣ #.##0,00`: Lägger till ett tecken för schweizerfranc med punkter som tusentalsavgränsare och ett kommatecken för decimalavgränsare (till exempel `₣ 1.000,00`). `R #,##0.00`: Sydafrikanskt randtecken med två decimaler och blanksteg efter randtecknet. `$ #,##0.000`: Lägger till tre decimaler i dollarformatet.
dec_sep	Sträng som specificerar decimalavgränsare. Om det utelämnas används det `MoneyDecimalSep`-värde som angetts i dataladdningsskriptet. Funktionen `Money` stöder många format för decimalavgränsare, t.ex: Komma ' , ': Till exempel, `$1,000,00` för ettusen dollar. Punkt ' . ': Till exempel `€ 1,000.00`. Bindestreck ' - ': När det används i stället för den traditionella punkten eller kommatecknet som decimalavgränsare visar det värden som `¥1,000-00`.

Argument	Beskrivning
thou_sep	Sträng som specificerar tusendelsavgränsare. Om det utelämnas används det `MoneyThousandSep`-värde som angetts i dataladdningsskriptet. Funktionen `Money` stöder många alternativ för tusentalsavgränsaren: Punkt `'.'`: Använder en punkt som tusentalsavgränsare (vanligt i vissa europeiska format), visas som `$1.000,00` om den kombineras med en decimalavgränsare med kommatecken. Mellanslag `' '`: Använder ett mellanslag som tusentalsavgränsare, vilket är vanligt i vissa europeiska länder. Visas som `€ 1 000.00`. Understreck `'_'`:Använder ett understreck för tusentalsavgränsaren, som ibland används i tekniska eller specifikt regionala format. Visar värden som `$1_000.00`.

Om argumenten **format**, **dec_sep** och **thou_sep** utelämnas används det valutaformat som är inställt i operativsystemet vid formateringen av varje specifik del av utskriften. Operativsysteminställningarna inkluderas automatiskt som systemvariabler när skriptet skapas, vilket säkerställer att dessa är konsekventa i hela programmet.

Dessa systemvariabler är **MoneyFormat**, **MoneyDecimalSep**, och **MoneyThousandSep**, och mappar till funktionsargumenten **format**, **dec_sep** och **thou_sep**.

Exempel: diagramuttryck

Exempel	Resultat
`Money( 35648 )`	Returnerar `kr 35 648,00` när `MoneyFormat`-inställningen är `kr ##0,00` och `MoneyThousandSep` är en punkt `' '`.
`Money( 35648, '#,##0 ¥', '.' , ',' )`	Returnerar `3,564,800 ¥` genom att ignorera inställningen för `MoneyFormat` och `MoneyThousandSep` eftersom de valfria parametrarna `Format`, `dec_sep` och `thou_sep` anges i funktionen.
`Money(35648)`	Returnerar `$ 35,648.00` när `MoneyFormat`-inställningen är `$ #,##0.00` och `MoneyThousandSep` är ett komma `','`.

Exempel – grunderna i Money

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `InputValue`.

- Följande variabelformat:
  - Variabeln `MoneyFormat`, som använder formatet: `kr ##0,00`
  - Variabeln `MoneyThousandSep` är inställd på att använda ett blanksteg `' '`

Laddningsskript

```
SET MoneyFormat='kr ##0,00';
SET MoneyThousandSep=' ';
Example:
Load * inline [
InputValue
35648
3564800
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `InputValue`

Skapa följande mått:

- `=Money(InputValue)` för att visa värdet som en valuta formaterad med hjälp av systeminställningen.
- `=Money(InputValue, '#,##0 ¥', '.' , ',')` för att visa värdet som en valuta som är formaterat med det format som anges i funktionen.
- `=Num(InputValue)`, för att returnera det numeriska värdet.

Results table

InputValue	Money(InputValue)	Money(InputValue, '#,##0 ¥', '.' , ',')	Num(InputValue)
35648	kr35 648,00	35,648 ¥	35648
3564800	kr3 564 800,00	3,564,800 ¥	3564800

I det här exemplet kan du se hur funktionen `Money` visar värden och hur värdena i parametern format kan ändra utdata.

Exempel – grunderna i valuta med en annan formatinställning

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `InputValue`.

- Följande variabelformat:
  - Variabeln `MoneyFormat`, som använder formatet: `$ #,##0.00`
  - Variabeln `MoneyThousandSep` är inställd på att använda ett komma `','`

Laddningsskript

```
SET MoneyFormat='$ #,##0.00';
SET MoneyThousandSep=',';
Example:
Load * inline [
InputValue
35648
3564800
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `InputValue`

Skapa följande mått:

- `=Money(InputValue)` för att visa värdet som en valuta formaterad med hjälp av systeminställningen.
- `=Money(InputValue, '#,##0 ¥', '.' , ', )` för att visa värdet som en valuta som är formaterat med det format som anges i funktionen.
- `=Num(InputValue)`, för att returnera det numeriska värdet.

Results table

InputValue	Money(InputValue)	Money(InputValue, '#,##0 ¥', '.' , ',')	Num(InputValue)
35648	$ 35,648.00	35,648 ¥	35648
3564800	$ 3,564,800.00	3,564,800 ¥	3564800

I det här exemplet kan du se hur funktionen `Money` visar värden och hur värdena i parametern format kan ändra utdata.

Exempel – Money-scenario

Diagramuttryck

Översikt

Försäljningsdata från kunder tas emot i olika valutor. Du vill visa data i en tabell med hjälp av relevant valutaformat.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Customer`
  - `CurrencyCode`
  - `Sales`
- Följande variabelformat:
  - Variabeln `MoneyFormat`, som använder formatet: `$ ###0.00;-$ ###0.00`
  - Variabeln `MoneyThousandSep` är inställd på att använda ett komma `','`
  - Variabeln `MoneyDecimalSep` är inställd på att använda en punkt `'.'`

Laddningsskript

```
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat='$ ###0.00;-$ ###0.00';
Example:
Load * inline [
Customer, CurrencyCode, Sales
Customer A, USD, 100008
Customer B, EUR, 12500
Customer C, YEN, 15000
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=Money(Sales, If(CurrencyCode = 'USD', '$#,##0.00', If(CurrencyCode = 'EUR', '€ #,##0.00', '¥#,##0')))`, för att visa valutan i rätt valutaformat.

Results table

Customer	Money(Sales, If(CurrencyCode = 'USD', '$#,##0.00', If(CurrencyCode = 'EUR', '€ #,##0.00', '¥#,##0')))
Customer A	$100,008.00
Customer B	€ 12,500.00
Customer C	¥15,000

Försäljningsdata använder nu det valutaformat som definierats i mätuttrycket. Parametrarna decimal och tusen använder inställningarna i laddningsskriptet.

Num

Num() formaterar ett tal, vilket innebär att det numeriska värdet i indata konverteras och visas med det format som specificerats i den andra parametern. Om den andra parametern utelämnas använder funktionen de decimal- och tusentalsavgränsare som anges i dataladdningsskriptet. Egna symboler för decimaler och tusentalsseparatorer är valbara parametrar.

Syntax:

```
Num(number[, format[, dec_sep [, thou_sep]]])
```

Returnerad datatyp: dual

Num-funktionen returnerar ett dualt värde med både strängvärdet och det numeriska värdet. Funktionen tar det numeriska värdet i indatauttrycket och genererar en sträng som representerar talet.

Argument

Argument	Beskrivning
number	Textsträngen som ska utvärderas.
format	Sträng som anger hur den resulterande strängen ska formateras. Om den utelämnas används de decimal- och tusentalsavgränsare som anges i dataladdningsskriptet. Funktionen `Num` stöder många formatalternativ för olika datumstrukturer, t.ex: `#,##0`: Visar ett tal utan decimaler. `#,##0.00`: Visar ett tal med två decimaler. `##0.00%`: Visar ett tal som ett procenttal med två decimaler. `#,##0;(#,##0)`: Formaterar negativa tal till att visas inom parentes. `$#,##0.00`: Formaterar tal som valuta med en $-symbol. `0.00E+0`: Visar tal i vetenskaplig notation.
dec_sep	Sträng som specificerar decimalavgränsare. Om det utelämnas används värdet för variabeln DecimalSep som angetts i dataladdningsskriptet. Funktionen `Num` stöder många format för decimalavgränsare, t.ex: Komma ' , ': Till exempel, `$1,000,00` för ettusen dollar. Punkt ' . ': Till exempel `€ 1,000.00`. Bindestreck ' - ': När det används i stället för den traditionella punkten eller kommatecknet som decimalavgränsare visar det värden som `¥1,000-00`. Mer information finns i *DecimalSep (page 255)*.

Argument	Beskrivning
thou_sep	Sträng som specificerar tusendelsavgränsare. Om det utelämnas används värdet för variabeln ThousandSep som angetts i dataladdningsskriptet. Funktionen `Num` stöder många alternativ för tusentalsavgränsaren: Punkt `'.'`: Använder en punkt som tusentalsavgränsare (vanligt i vissa europeiska format), visas som `$1.000,00` om den kombineras med en decimalavgränsare med kommatecken. Mellanslag `' '`: Använder ett mellanslag som tusentalsavgränsare, vilket är vanligt i vissa europeiska länder. Visas som `€ 1 000.00`. Understreck `'_'`:Använder ett understreck för tusentalsavgränsaren, som ibland används i tekniska eller specifikt regionala format. Visar värden som `$1_000.00`. Mer information finns i *ThousandSep (page 291)*.

Exempel: diagramuttryck

Exempel	Resultat (returnerat strängvärde)
`Num(35648.312)`	Returnerar `35648.312` (beror på miljövariablerna i skriptet)
`Num(35648.312, '0.0', '.')`	Returnerar `35648.3`
`Num(35648.312, '0,00', ',')`	Returnerar `3,56,48`
`Num(35648.312, '#,##0.0', '.', ',')`	Returnerar `35,648.3`
`Num(35648.312, '# ##0', ',', ' ')`	Returnerar `35 648`

Exempel – grunderna i Num

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i en datatabell:
  - `InputValue`

Laddningsskript

```
Example:
Load * inline [
InputValue
35648
3564800
```

```
356.48
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `InputValue`

Skapa följande mått:

- `=Num(InputDate)` för att visa värdet som ett tal formaterat med hjälp av systeminställningarna.
- `=Num(InputValue, '0.0', '.')` för att visa värdet som ett tal som är formaterat med det format som anges i funktionen.
- `=Num(InputValue, '0,00', ',')` för att visa värdet som ett tal som är formaterat med det format som anges i funktionen.
- `=Num(InputValue, '#,##0.0', '.', ',')` för att visa värdet som ett tal som är formaterat med det format som anges i funktionen.
- `=Num(InputValue, '# ##0', ',', ' ')` för att visa värdet som ett tal som är formaterat med det format som anges i funktionen.

Results table

InputValue	Num (InputDate)	Num (InputValue, '0.0', '.')	Num (InputValue, '0,00', ',')	Num (InputValue, '#,##0.0', '.', ',')	Num (InputValue, '# ##0', ',', ' ')
356.48	356.48	356.5	3,56	356.5	356
35648	35648	35648.0	3,56,48	35,648.0	35 648
3564800	3564800	3564800.0	3,56,48,00	3,564,800.0	3 564 800

Resultaten visar hur funktionen `Num` kan visa värdena för fältet `InputValue` baserat på det format som anges i funktionsuttrycket.

Laddningsskript

Funktionen `Num` kan användas i laddningsskriptet för att formatera ett tal, även om tusentals- och decimalavgränsare redan har ställts in i skriptet. I laddningsskriptet nedan ingår specifika tusentals- och decimalavgränsare, men `Num` används för att formatera data på olika sätt.

Laddningsskript

Skapa ett nytt delavsnitt i Skriptredigeraren och lägg sedan till exempelskriptet och kör det. Lägg sedan till åtminstone de fält som listas i resultattabellen till ett ark i din applikation för att se resultatet.

```
SET ThousandSep=',';
SET DecimalSep='.';
Transactions:
Load
*,
Num(transaction_amount) as [No formatting],
```

```
Num(transaction_amount,'0') as [0],
Num(transaction_amount,'#,##0') as [#,##0],
Num(transaction_amount,'# ###,00') as [# ###,00],
Num(transaction_amount,'# ###,00',',',' ') as [# ###,00 , ',' , ' '],
Num(transaction_amount,'#,###.00','.',',') as [#,###.00 , '.' , ','],
Num(transaction_amount,'$#,###.00') as [$#,###.00],
;
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 20180830, 12423.56, 23, 0,2038593, L, Red
3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue
3753, 20180922, 1251, 7, 0, 3036491, l, Black
3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red
3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue
3757, 20180923, 3177.4, 21, .14, 203521, XL, Black
];
```

Resultat

Qlik Sense-tabell som visar resultaten man får när *Num*-funktionen används på olika sätt i laddningsskriptet. Tabellens fjärde kolumn innehåller exempel på när formatering inte används korrekt.

No formatting	0	#,##0	# ###,00	# ###,00 , ',' , ''	#,###.00 , '.' , ','	$#,###.00
-59.18	-59	-59	-59###,00	-59,18	-59.18	$-59,18
15.75	16	16	16###,00	15,75	15.75	$15,75
1251	1251	1,251	1251###,00	1 251,00	1,251.00	$1,251.00
3177.4	3177	3,177	3177###,00	3 177,40	3,177.40	$3,177.40
5356.31	5356	5,356	5356###,00	5 356,31	5,356.31	$5,356.31
12423.56	12424	12,424	12424###,00	12 423,56	12,423.56	$12,423.56
21484.21	21484	21,484	21484###,00	21 484,21	21,484.21	$21,484.21

Exempel – Num-scenario

Diagramuttryck

Översikt

En datauppsättning innehåller försäljningsdata som du vill presentera i ett enklare numeriskt format.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `Customer`
    - `Sales`

Laddningsskript

```
Example:
Load * inline [
Customer, Sales
Customer A, 2550000
Customer B, 12500
Customer C, 15000
Customer D, 250
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=If(Sales > 1000000, Num(Sales/1000000, '#,##0.00M'), If(Sales > 1000, Num(Sales/1000, '#,##0.00K'), Num(Sales, '#,##0.00')))`, för att ställa in den numeriska representationen av värdena.

Results table

Customer	If(Sales > 1000000, Num(Sales/1000000, '#,##0.00M'), If(Sales > 1000, Num (Sales/1000, '#,##0.00K'), Num(Sales, '#,##0.00')))
Customer A	2.55M
Customer B	12.50K
Customer C	15.00K
Customer D	250.00

Måttuttrycket i kolumn 2 använder funktionen `Num` för att presentera data i ett förenklat numeriskt format.

Laddningsskript

`Num` kan användas i laddningsskript för att formatera ett tal som en procentandel.

Laddningsskript

Gå till Skriptredigeraren och skapa ett nytt delavsnitt. Lägg sedan till exempelskriptet och kör det. Lägg sedan till de fält som listas i resultatkolumnen till ett ark i din applikation för att se resultatet.

```
SET ThousandSep=',';
SET DecimalSep='.';
```

```
Transactions:
Load
*,
Num(discount,'#,##0%') as [Discount #,##0%]
;
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 20180830, 12423.56, 23, 0,2038593, L, Red
3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue
3753, 20180922, 1251, 7, 0, 3036491, l, Black
3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red
3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue
3757, 20180923, 3177.4, 21, .14, 203521, XL, Black
];
```

Qlik Sense-tabell som visar resultaten man får när `Num`-funktionen används i laddningsskriptet för att formatera procentandelar.

Discount	Discount #,##0%
0	0%
0.1	10%
0.22	22%
0	0%
75	7,500%
0.333333333333333	33%
.14	14%

Time

Time() formaterar ett uttryck som ett tidsvärde, i det tidsformat som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns.

Syntax:

```
Time(number[, format])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
number	Talet som ska formateras.

Argument	Beskrivning
format	Sträng som beskriver hur den resulterande tidsträngen ska formateras. Om den utelämnas används operativsystemets tidsformat och decimalavgränsare. Funktionen `Time` har stöd för många formatalternativ för olika tidsstrukturer, t.ex: `hh:mm:ss` - formaterar ett tidsvärde i ett standardtidsformat och visar timmar, minuter och sekunder med inledande nollor. `hh:mm TT` - Använder ett 12-timmarsformat med suffixet AM/PM. `hh:mm` - Visar endast timmar och minuter och utelämnar sekunder. `hh.mm.ss` - tidsformat med olika avdelare, t.ex. en punkt (`.`) i stället för ett kolon. `hh TT` - Inkluderar endast timme och AM/PM. `hh "Hrs" mm "Mins"` - visar tiden med anpassad text, t.ex. Hrs.

Exempel: diagramuttryck

Exempel	Resultat
`Time( 0.375 )`	Returnerar `09:00:00` när inställningen för `TimeFormat` är `'hh:mm:ss'`.
`Time( 35648.375 )`	Returnerar `09:00:00` när `Timeformat` är `'hh:mm:ss'`.
`Time( 0.99999, 'hh-mm' )`	Returnerar `23-59` genom att ignorera inställningen `TimeFormat` eftersom formatet anges i måttet.

Exempel – grunderna i Tid

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i en datatabell:
  - `TimeValue`
- Variabeln `TimeFormat`, som använder formatet `hh:mm:ss`.

Laddningsskript

```
SET TimeFormat='hh:mm:ss';
Example:
Load * inline [
TimeValue
0.375
```

```
35648.375
0.99999
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `TimeValue`

Skapa följande mått:

- `=Time(TimeValue )` för att visa värdet som en tid formaterad med hjälp av systeminställningen.
- `=Time(TimeValue, 'hh-mm' )` för att visa värdet som en tid med det format som fås i funktionen.
- `=Time(TimeValue, 'hh:mm:ss' )` för att visa värdet som en tid med det format som fås i funktionen.
- `=Time(TimeValue, 'hh:mm TT' )` för att visa värdet som en tid med det format som fås i funktionen.
- `=Time(TimeValue, 'hh:mm' )` för att visa värdet som en tid med det format som fås i funktionen.
- `=Time(TimeValue, 'hh.mm.ss' )` för att visa värdet som en tid med det format som fås i funktionen.
- `=Time(TimeValue, 'hh TT' )` för att visa värdet som en tid med det format som fås i funktionen.
- `=Time(TimeValue, 'hh "Hrs" mm "Mins"' )` för att visa värdet som en tid med det format som fås i funktionen.

Resultattabell

Tidsvärde	Tid (Tidsvärde)	Tid (Tidsvärde, 'hh-mm')	Tid (Tidsvärde, 'hh:mm:ss')	Tid (Tidsvärde, 'hh:mm TT')	Time (TimeValue, 'hh:mm')	Time (TimeValue, 'hh.mm.ss')	Time (TimeValue, 'hh TT')	Time (TimeValue, 'hh "Tim" mm "Min"')
0.375	09:00:00	09-00	09:00:00	09:00 AM	09:00	09.00.00	09 AM	09 tim 00 min
0.99999	23:59:59	23-59	23:59:59	11:59 PM	23:59	23.59.59	11 PM	23 timmar 59 minuter
35648.375	09:00:00	09-00	09:00:00	09:00	09:00	09:00:00	09:00	09 timmar 00 minuter

Resultaten visar hur funktionen `Time` kan visa värdena baserat på det format som anges i funktionsuttrycket.

Exempel – grunderna i tid med en annan TimeFormat-inställning

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i en datatabell:
  - `TimeValue`
- Variabeln `TimeFormat`, som använder formatet `hh.mm.ss`.

Laddningsskript

```
SET TimeFormat='hh.mm.ss';
Example:
Load * inline [
TimeValue
0.375
35648.375
0.99999
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `TimeValue`

Skapa följande mått:

- `=Time(TimeValue )` för att visa värdet som en tid formaterad med hjälp av systeminställningen.
- `=Time(TimeValue, 'hh-mm' )` för att visa värdet som en tid med det format som fås i funktionen.
- `=Time(TimeValue, 'hh:mm:ss' )` för att visa värdet som en tid med det format som fås i funktionen.
- `=Time(TimeValue, 'hh:mm TT' )` för att visa värdet som en tid med det format som fås i funktionen.
- `=Time(TimeValue, 'hh:mm' )` för att visa värdet som en tid med det format som fås i funktionen.
- `=Time(TimeValue, 'hh.mm.ss' )` för att visa värdet som en tid med det format som fås i funktionen.
- `=Time(TimeValue, 'hh TT' )` för att visa värdet som en tid med det format som fås i funktionen.
- `=Time(TimeValue, 'hh "Hrs" mm "Mins"' )` för att visa värdet som en tid med det format som fås i funktionen.

Resultattabell

Tidsvärde	Tid (Tidsvärde)	Tid (Tidsvärde, 'hh-mm')	Tid (Tidsvärde, 'hh:mm:ss')	Tid (Tidsvärde, 'hh:mm TT')	Time (TimeValue, 'hh:mm')	Time (TimeValue, 'hh.mm.ss')	Time (TimeValue, 'hh TT')	Time (TimeValue, 'hh "Timmar" mm "Minuter"')
0.375	09:00:00	09:00	09:00:00	09:00	09:00	09:00:00	09.00	09 timmar 00 minuter
0,99999	23:59:59	23:59	23:59:59	23:59	23:59	23:59:59	23.00	23 tim 59 min
35648,375	09:00:00	09:00	09:00:00	09:00	09:00	09.00.00	09:00	09 tim 00 min

Resultaten visar hur funktionen `Time` kan visa värdena baserat på det format som anges i funktionsuttrycket.

Timestamp

TimeStamp() formaterar ett uttryck som ett datum och tid-värde, i det format för tidsmarkörer som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns.

Syntax:

```
Timestamp(number[, format])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
number	Talet som ska formateras.

Argument	Beskrivning
format	Sträng som beskriver hur den resulterande tidsmarkörsträngen ska formateras. Om den utelämnas används det korta datumformat, tidsformat och decimalavgränsare som har ställts in för operativsystemet. Funktionen `Timestamp` har stöd för många formatalternativ för olika tidsstrukturer, t.ex: `DD/MM/YYYY`: Fokuserar bara på datumet (t.ex.: `15/11/2024`). `YYYY-MM-DD hh:mm:ss`: Visar hela tidsmarkören (t.ex.: `2024-11-15 13:30:45`). `MMM DD, YYYY`: Visar en kortversion av månaden och hela året (t.ex.: `Nov 15, 2024`). `hh:mm:ss TT`: Visar endast tiden i 12-timmarsformat (t.ex.: `01:30:45 PM`) `HH:mm:ss`: Visar tiden i 24-timmarsformat (t.ex.: `13:30:45`) `wwww, MMM DD YYYY`: Visar hela veckodagen och datumet (t.ex.: `Friday, Nov 15 2024`) `YYYY-MM-DD hh:mm:ss.fff`: Inkluderar millisekunder (t.ex.: `2024-11-15 13:30:45.123`).

Exempel: diagramuttryck

Exempel	Resultat
`Timestamp( 35648 )`	Returnerar `97-08-06 00:00:00` när `TimestampFormat` är `YY-MM-DD hh:mm:ss`.
`Timestamp(35648,'YYYY-MM-DD hh.mm')`	Returnerar `1997-08-06 00.00` när medföljande format är `YYYY-MM-DD hh.mm`.
`Timestamp( 35648,'M/D/YYYY h:mm:ss[.fff] TT')`	Returnerar `8/6/1997 12:00:00 AM` när medföljande format är `M/D/YYYY h:mm:ss[.fff] TT`.
`Timestamp( 35648,'www, DD MMM YYYY hh:mm TT')`	Returnerar `Wed, 06 Aug 1997 12:00 AM` när medföljande format är `www, DD MMM YYYY hh:mm TT`.

Exempel – grunderna i Timestamp

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i en datatabell:
  - `TimeStampValue`
- Variabeln `TimestampFormat`, som använder formatet `YY-MM-DD hh:mm:ss`.

Laddningsskript

```
SET TimestampFormat='YY-MM-DD hh:mm:ss';
Example:
Load * inline [
TimeStampValue
35648.375
35648
36648.875
36648.8756789
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `TimeStampValue`

Skapa följande mått:

- `=Timestamp(TimeStampValue)` för att visa värdet som en tidsmarkör formaterad med hjälp av systeminställningen.
- `=Timestamp(TimeStampValue,'YYYY-MM-DD hh.mm')` för att visa värdet som en tidsmarkör med det format som fås i funktionen.
- `=Timestamp(TimeStampValue,'M/D/YYYY h:mm:ss[.fff] TT')` för att visa värdet som en tidsmarkör med det format som fås i funktionen.
- `=Timestamp(TimeStampValue,'YYYY-MM-DD hh:mm:ss.fff')` för att visa värdet som en tidsmarkör med det format som fås i funktionen.
- `=Timestamp(TimeStampValue,'www, DD MMM YYYY hh:mm TT')` för att visa värdet som en tidsmarkör med det format som fås i funktionen.
- `=Timestamp(TimeStampValue,'wwww, MMM DD YYYY')` för att visa värdet som en tidsmarkör med det format som fås i funktionen.

Resultattabell

TimeStampValue	Timestamp (TimeStampValue)	Timestamp (TimeStampValue,'YYYY-MM-DD hh.mm'	Timestamp (TimeStampValue,'M/D/YYYY h:mm:ss[.fff] TT')	Timestamp (TimeStampValue,'YYYY-MM-DD hh:mm:ss.fff')	Timestamp (TimeStampValue,'www, DD MMM YYYY hh:mm TT')	Timestamp (TimeStampValue,'wwww, MMM DD YYYY')
35648	97-08-06 00:00:00	1997-08-06 00.00	1997-08-06 00:00:00	1997-08-06 00:00:00	Wed, 06 Aug 1997 12:00 AM	Wednesday, Aug 06 1997
35648.375	1997-08-06 09:00:00	1997-08-06 09:00	1997-08-06 09:00:00	1997-08-06 09:00:00.000	Wed, 06 Aug 1997 09:00 AM	Wednesday, Aug 06 1997

TimeStampValue	Timestamp (TimeStampValue)	Timestamp (TimeStampValue,'YYYY-MM-DD hh.mm'	Timestamp (TimeStampValue,'M/D/YYYY h:mm:ss[.fff] TT')	Timestamp (TimeStampValue,'YYYY-MM-DD hh:mm:ss.fff')	Timestamp (TimeStampValue,'www, DD MMM YYYY hh:mm TT')	Timestamp (TimeStampValue,'www, MMM DD YYYY')
36648.875	00-05-02 21:00:00	2000-05-02 21.00	5/2/2000 9:00:00 PM	2000-05-02 21:00:00.000	Tue, 02 May 2000 09:00 PM	Tuesday, May 02 2000
36648.8756789	2000-05-02 21:00:58	2000-05-02 21:00	2000-05-02 21:00:58	2000-05-02 21:00:58.657	Tue, 02 May 2000 09:00 PM	Tuesday, May 02 2000

Resultaten visar hur funktionen `TimestampFormat` kan visa värden med hjälp av olika formatparametrar.

Exempel – grunderna i tidsmarkörer med en annan TimestampFormat-inställning

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i en datatabell:
  - `TimeStampValue`
- Variabeln `TimestampFormat`, som använder formatet `YY-MM-DD hh:mm:ss`.

Laddningsskript

```
SET TimestampFormat='M/D/YY hh:mm:ss';
Example:
Load * inline [
TimeStampValue
35648.375
35648
36648.875
36648.8756789
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `TimeStampValue`

Skapa följande mått:

- `=Timestamp(TimeStampValue)` för att visa värdet som en tidsmarkör formaterad med hjälp av systeminställningen.
- `=Timestamp(TimeStampValue,'YYYY-MM-DD hh.mm')` för att visa värdet som en tidsmarkör med det format som fås i funktionen.
- `=Timestamp(TimeStampValue,'M/D/YYYY h:mm:ss[.fff] TT')` för att visa värdet som en tidsmarkör med det format som fås i funktionen.
- `=Timestamp(TimeStampValue,'YYYY-MM-DD hh:mm:ss.fff')` för att visa värdet som en tidsmarkör med det format som fås i funktionen.
- `=Timestamp(TimeStampValue,'www, DD MMM YYYY hh:mm TT')` för att visa värdet som en tidsmarkör med det format som fås i funktionen.
- `=Timestamp(TimeStampValue,'wwww, MMM DD YYYY')` för att visa värdet som en tidsmarkör med det format som fås i funktionen.

Resultattabell

TimeStampValue	Timestamp (TimeStampValue)	Timestamp (TimeStampValue,'YYYY-MM-DD hh.mm'	Timestamp (TimeStampValue,'M/D/YYYY h:mm:ss[.fff] TT')	Timestamp (TimeStampValue,'YYYY-MM-DD hh:mm:ss.fff')	Timestamp (TimeStampValue,'www, DD MMM YYYY hh:mm TT')	Timestamp (TimeStampValue,'wwww, MMM DD YYYY')
35648	8/6/97 00:00:00	1997-08-06 00:00	1997-08-06 00:00:00	1997-08-06 00:00:00.000	Wed, 06 Aug 1997 12:00 AM	Wednesday, Aug 06 1997
1997-08-06 09:00:00	1997-08-06 09:00:00	1997-08-06 09:00	1997-08-06 09:00:00	1997-08-06 09:00:00	Wed, 06 Aug 1997 09:00 AM	Wednesday, Aug 06 1997
1997-08-06 21:00:00	5/2/00 21:00:00	2000-05-02 21:00	2000-05-02 21:00:00	2000-05-02 21:00:00.000	Tue, 02 May 2000 09:00 PM	Tuesday, May 02 2000
2000-05-02 21:00:58	2000-05-02 21:00:58	2000-05-02 21.00	2000-05-02 21:00:58	2000-05-02 21:00:58.657	Tue, 02 May 2000 09:00 PM	Tuesday, May 02 2000

Resultaten visar hur funktionen `TimestampFormat` kan visa värden med hjälp av olika formatparametrar.

Exempel – ändra tidsmarkörens format för kundorder

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `Customer`
    - `CustomerName`
    - `OrderTimestamp`

Laddningsskript

```
Example:
Load * inline [
OrderID, CustomerName, OrderTimestamp
1, Alice, 2024-02-20 11:00:00
2, Bob, 2024-03-01 15:30:00
3, Charlie, 2024-04-12 09:45:00
4, David, 2024-05-21 21:10:00
5, Eve, 2024-06-30 14:25:00
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `OrderID`
- `CustomerName`

Skapa följande mått:

- `=Timestamp(OrderTimestamp, 'WWWW, DD MMM YYYY hh:mm TT')`, för att visa orderns tidsmarkör i ett format som visar veckodag, datum och klockslag.

Resultattabell

Order-ID	Kundnamn	Timestamp(OrderTimestamp, 'WWWW, DD MMM YYYY hh:mm TT')
1	Alice	Tuesday, 20 Feb 2024 11:00 AM
2	Bob	Friday, 01 Mar 2024 03:30 PM
3	Charlie	Friday, 12 Apr 2024 09:45 AM
4	David	Tuesday, 21 May 2024 09:10 PM
5	Eve	Sunday, 30 Jun 2024 02:25 PM

Med hjälp av funktionen `Timestamp` formateras nu tidsmarkören så att den innehåller veckodag och det nya formatet för datum och tid.

8.13 Allmänna numeriska funktioner

I dessa allmänna numeriska funktioner är argumenten uttryck där **x** ska tolkas som ett reellt tal. Alla funktioner kan användas både i dataladdningsskript och diagramuttryck.

Allmänna numeriska funktioner – en översikt

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

bitcount

BitCount() returnerar hur många bitar i den binära motsvarigheten till ett decimaltal som är inställda på 1. Denna funktion returnerar alltså antalet inställda bitar i **integer_number**, där **integer_number** tolkas som ett signerat 32-bitars heltal.

```
BitCount (integer_number)
```

div

Div() returnerar heltalsdelen av den aritmetiska divisionen av det första argumentet med det andra argumentet. Båda parametrarna tolkas som reella tal, de behöver alltså inte nödvändigtvis vara heltal.

```
Div (integer_number1, integer_number2)
```

fabs

Fabs() returnerar absolutvärdet för **x**. Resultatet blir ett positivt tal.

```
Fabs (x)
```

fact

Fact() returnerar faktorn av ett positivt heltal **x**.

```
Fact (x)
```

frac

Frac() returnerar decimaldelen av **x**.

```
Frac (x)
```

sign

Sign() returnerar 1, 0 eller -1 beroende på om **x** är ett positivt tal, 0 eller ett negativt tal.

```
Sign (x)
```

Kombinations- och permutationsfunktioner

combin

Combin() returnerar antalet kombinationer av **q**-element som kan väljas från en uppsättning **p**-element. Motsvaras av formeln: `Combin(p,q) = p! / q!(p-q)!` Ordningen som komponenterna väljs i är inte signifikant.

```
Combin (p, q)
```

permut

Permut() returnerar antalet permutationer av **q** element som kan väljas från en uppsättning med **p** element. Motsvaras av formeln: `Permut(p,q) = (p)! / (p - q)!` Ordningen som komponenterna väljs i är signifikant.

```
Permut (p, q)
```

Modulofunktioner

fmod

fmod() är en generaliserad modulofunktion som returnerar den återstående delen av heltalsdivisionen av det första argumentet (dividenden) med det andra argumentet (divisorn). Resultatet blir ett reellt tal. Båda argumenten tolkas som reella tal, de behöver alltså inte nödvändigtvis vara heltal.

```
Fmod (a, b)
```

mod

Mod() är en matematisk modulofunktion som returnerar den icke-negativa återstoden av en heltalsdivison. Det första argumentet är dividenden, det andra argumentet är divisorn. Båda argumenten måste vara heltalsvärden.

```
Mod (integer_number1, integer_number2)
```

Paritetsfunktioner

even

Even() returnerar True (-1) om **integer_number** är ett jämnt heltal eller noll. Den returnerar False (0) om **integer_number** är ett udda heltal och NULL om **integer_number** inte är ett heltal.

```
Even (integer_number)
```

odd

Odd() returnerar True (-1) om **integer_number** är ett udda heltal eller noll. Det returnerar False (0) om **integer_number** är ett jämnt heltal och NULL om **integer_number** inte är ett heltal.

```
Odd (integer_number)
```

Avrundningsfunktioner

ceil

Ceil() rundar av ett tal uppåt till närmaste multipel av det **step** som förskjuts av **offset** -talet.

```
Ceil (x[, step[, offset]])
```

floor

Floor() rundar av ett tal nedåt till närmaste multipel av det **step** som förskjuts av **offset** -talet.

```
Floor (x[, step[, offset]])
```

round

Round() returnerar resultatet av avrundning av ett tal uppåt eller nedåt till närmaste multipel av **step** som förskjuts av **offset** -talet.

```
Round ( x [ , steg [ , förskjutning ]])
```

BitCount

BitCount() returnerar hur många bitar i den binära motsvarigheten till ett decimaltal som är inställda på 1. Denna funktion returnerar alltså antalet inställda bitar i **integer_number**, där **integer_number** tolkas som ett signerat 32-bitars heltal.

Syntax:

```
BitCount(integer_number)
```

Returnerad datatyp: heltal

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`BitCount ( 3 )`	3 är 11 binärt, detta returnerar därför 2
`BitCount ( -1 )`	-1 är 64 ettor binärt, detta returnerar därför 64

Ceil

Ceil() rundar av ett tal uppåt till närmaste multipel av det **step** som förskjuts av **offset** -talet.

Jämför med funktionen **floor** som rundar av indata nedåt.

Syntax:

```
Ceil(x[, step[, offset]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
x	Indatatal.
step	Intervallsteg. Standardvärdet är 1.
offset	Definierar basen för stegintervallet. Standardvärdet är 0.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Ceil(2.4 )`	Returnerar 3 I det här exemplet är storleken på steget 1 och basen på stegintervallet 0. Intervallen är ...0 < x <=1, 1 < x <= 2, **2< x <=3**, 3< x <=4...
`Ceil(4.2 )`	Returnerar 5
`Ceil(3.88 ,0.1)`	Returnerar 3.9 I det här exemplet är storleken på intervallet 0,1 och basen på intervallet 0. Intervallen är ... 3.7 < x <= 3.8, **3.8 < x <= 3.9**, 3.9 < x <= 4.0...
`Ceil(3.88 ,5)`	Returnerar 5
`Ceil(1.1 ,1)`	Returnerar 2
`Ceil(1.1 ,1,0.5)`	Returnerar 1,5 I det här exemplet är storleken på steget 1 och förskjutningen 0,5. Det innebär att basen på stegintervallet är 0,5 och inte 0. Intervallen är ...**0.5 < x <=1.5**, 1.5 < x <= 2.5, 2.5< x <=3.5, 3.5< x <=4.5...
`Ceil(1.1 ,1,-0.01)`	Returnerar 1.99 Intervallen är ...-0.01< x <= 0.99, **0.99< x <= 1.99**, 1.99 < x <=2.99...

Combin

Combin() returnerar antalet kombinationer av **q**-element som kan väljas från en uppsättning **p**-element. Motsvaras av formeln: `Combin(p,q) = p! / q!(p-q)!` Ordningen som komponenterna väljs i är inte signifikant.

Syntax:

```
Combin(p, q)
```

Returnerad datatyp: heltal

Begränsningar:

Icke-heltal kommer att trunkeras.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
Hur många kombinationer av 7 nummer kan göras bland totalt 35 lotterinummer? `Combin( 35,7 )`	Returnerar 6 724 520

Div

Div() returnerar heltalsdelen av den aritmetiska divisionen av det första argumentet med det andra argumentet. Båda parametrarna tolkas som reella tal, de behöver alltså inte nödvändigtvis vara heltal.

Syntax:

```
Div(integer_number1, integer_number2)
```

Returnerad datatyp: heltal

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Div( 7,2 )`	Returnerar 3
`Div( 7.1,2.3 )`	Returnerar 3
`Div( 9,3 )`	Returnerar 3
`Div( -4,3 )`	Returnerar -1
`Div( 4,-3 )`	Returnerar -1
`Div( -4,-3 )`	Returnerar 1

Even

Even() returnerar True (-1) om **integer_number** är ett jämnt heltal eller noll. Den returnerar False (0) om **integer_number** är ett udda heltal och NULL om **integer_number** inte är ett heltal.

Syntax:

```
Even(integer_number)
```

Returnerad datatyp: Booleansk

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Even( 3 )`	Returnerar 0, False
`Even( 2 * 10 )`	Returnerar -1, True
`Even( 3.14 )`	Returnerar NULL

Fabs

Fabs() returnerar absolutvärdet för **x**. Resultatet blir ett positivt tal.

Syntax:

```
fabs(x)
```

Returnerad datatyp: numeriska

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`fabs( 2.4 )`	Returnerar 2.4
`fabs( -3.8 )`	Returnerar 3.8

Fact

Fact() returnerar faktorn av ett positivt heltal **x**.

Syntax:

```
Fact(x)
```

Returnerad datatyp: heltal

Begränsningar:

Om talet **x** inte är ett heltal trunkeras det. Icke-positiva tal returnerar NULL.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Fact( 1 )`	Returnerar 1
`Fact( 5 )`	Returnerar 120 (1 * 2 * 3 * 4 * 5 = 120)
`Fact( -5 )`	Returnerar NULL

Floor

Floor() rundar av ett tal nedåt till närmaste multipel av det **step** som förskjuts av **offset** -talet.

Jämför med **ceil** funktionen, som rundar av indatatal uppåt.

Syntax:

```
Floor(x[, step[, offset]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
x	Indatatal.
step	Intervallsteg. Standardvärdet är 1.
offset	Definierar basen för stegintervallet. Standardvärdet är 0.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Floor(2.4)`	Returnerar 2 In this example, the size of the step is 1 and the base of the step interval is 0. The intervals are ...0 <= x <1, 1 <= x < 2, **2<= x <3**, 3<= x <4....
`Floor(4.2)`	Returnerar 4

Exempel	Resultat
`Floor(3.88 ,0.1)`	Returnerar 3.8 I det här exemplet är storleken på intervallet 0,1 och basen på intervallet 0. Intervallen är ... 3.7 <= x < 3.8, **3.8 <= x < 3.9**, 3.9 <= x < 4.0...
`Floor(3.88 ,5)`	Returnerar 0
`Floor(1.1 ,1)`	Returnerar 1
`Floor(1.1 ,1,0.5)`	Returnerar 0.5 I det här exemplet är storleken på steget 1 och förskjutningen 0,5. Det innebär att basen på stegintervallet är 0,5 och inte 0. Intervallen är ...**0.5 <= x <1.5**, 1.5 <= x < 2.5, 2.5<= x <3.5,...

Fmod

fmod() är en generaliserad modulofunktion som returnerar den återstående delen av heltalsdivisionen av det första argumentet (dividenden) med det andra argumentet (divisorn). Resultatet blir ett reellt tal. Båda argumenten tolkas som reella tal, de behöver alltså inte nödvändigtvis vara heltal.

Syntax:

```
fmod(a, b)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
a	Dividend
b	Divisor

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`fmod( 7,2 )`	Returnerar 1
`fmod( 7.5,2 )`	Returnerar 1,5
`fmod( 9,3 )`	Returnerar 0
`fmod( -4,3 )`	Returnerar -1
`fmod( 4,-3 )`	Returnerar 1
`fmod( -4,-3 )`	Returnerar -1

Frac

Frac() returnerar decimaldelen av **x**.

Fraktionens definieras så att `Frac(x ) + Floor(x ) = x`. Förenklat innebär detta att fraktionen av ett positivt tal är skillnaden mellan talet (x) och heltalet som föregår decimaldelen.

Till exempel: Fraktionen av 11,43 = 11,43 - 11 = 0,43

För ett negativt tal, säg -1,4, Floor(-1.4) = -2, som ger följande resultat:

Fraktionen av -1,4 = -1,4 - (-2) = -1,4 + 2 = 0,6

Syntax:

```
Frac(x)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
x	Tal som decimaldel ska returneras för.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Frac( 11.43 )`	Returnerar 0,43
`Frac( -1.4 )`	Returnerar 0,6
Extrahera tidskomponenten från den numeriska representationen av en tidsmarkör, vilket därmed utelämnar datumet. `Time(Frac(44518.663888889))`	Returnerar 15.56.00

Mod

Mod() är en matematisk modulofunktion som returnerar den icke-negativa återstoden av en heltalsdivison. Det första argumentet är dividenden, det andra argumentet är divisorn. Båda argumenten måste vara heltalsvärden.

Syntax:

```
Mod(integer_number1, integer_number2)
```

Returnerad datatyp: heltal

Begränsningar:

integer_number2 måste vara större än 0.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Mod( 7,2 )`	Returnerar 1
`Mod( 7.5,2 )`	Returnerar NULL
`Mod( 9,3 )`	Returnerar 0
`Mod( -4,3 )`	Returnerar 2
`Mod( 4,-3 )`	Returnerar NULL
`Mod( -4,-3 )`	Returnerar NULL

Odd

Odd() returnerar True (-1) om **integer_number** är ett udda heltal eller noll. Det returnerar False (0) om **integer_number** är ett jämnt heltal och NULL om **integer_number** inte är ett heltal.

Syntax:

```
Odd(integer_number)
```

Returnerad datatyp: Booleansk

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Odd( 3 )`	Returnerar -1, True
`Odd( 2 * 10 )`	Returnerar 0, False
`Odd( 3.14 )`	Returnerar NULL

Permut

Permut() returnerar antalet permutationer av **q** element som kan väljas från en uppsättning med **p** element. Motsvaras av formeln: `Permut(p,q) = (p)! / (p - q)!` Ordningen som komponenterna väljs i är signifikant.

Syntax:

```
Permut(p, q)
```

Returnerad datatyp: heltal

Begränsningar:

Om parametrarna inte är heltal kommer de att trunkeras.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
På hur många sätt kan medaljerna guld, silver och brons fördelas vid en 100-meters final med 8 deltagare? `Permut( 8,3 )`	Returnerar 336

Round

Round() returnerar resultatet av avrundning av ett tal uppåt eller nedåt till närmaste multipel av **step** som förskjuts av **offset** -talet.

Om det tal som ska rundas av ligger exakt mitt i ett intervall sker avrundningen uppåt.

Syntax:

```
Round(x[, step[, offset]])
```

Returnerad datatyp: numeriska

Om du rundar av ett flyttalsnummer kan felaktiga resultat visas. De här avrundningsfelen uppstår eftersom flyttalsnummer återges med ett ändligt antal binära siffror. Resultaten beräknas därför med hjälp av ett tal som redan är avrundat. Om dessa avrundningsfel påverkar ditt arbete multiplicerar du talen för att omvandla dem till heltal innan du avrundar.

Argument:

Argument

Argument	Beskrivning
x	Indatatal.
step	Intervallsteg. Standardvärdet är 1.
offset	Definierar basen för stegintervallet. Standardvärdet är 0.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Round(3.8 )`	Returnerar 4 I det här exemplet är storleken på steget 1 och basen på stegintervallet 0. Intervallen är ...0 <= x <1, 1 <= x < 2, 2<= x <3, **3<= x <4**...
`Round(3.8,4 )`	Returnerar 4
`Round(2.5 )`	Returnerar 3. I det här exemplet är storleken på steget 1 och basen på stegintervallet 0. Intervallen är ...0 <= x <1, 1 <= x <2, **2<= x <3**...
`Round(2,4 )`	Returnerar 4. Avrundat uppåt eftersom 2 är exakt hälften av stegintervallet 4. I det här exemplet är storleken på steget 4 och basen på stegintervallet 0. Intervallen är ...**0 <= x <4**, 4 <= x <8, 8<= x <12...
`Round(2,6 )`	Returnerar 0. Avrundat nedåt eftersom 2 är mindre än hälften av stegintervallet 6. I det här exemplet är storleken på steget 6 och basen på stegintervallet 0. Intervallen är ...**0 <= x <6**, 6 <= x <12, 12<= x <18...
`Round(3.88 ,0.1)`	Returnerar 3.9 I det här exemplet är storleken på steget 0,1 och basen på stegintervallet 0. Intervallen är ... 3.7 <= x <3.8, **3.8 <= x <3.9**, 3.9 <= x < 4.0...
`Round (3.88875,1/1000)`	Returnerar 3,889 I det här exemplet är storleken på steget 0,001, vilket avrundar talet uppåt och begränsar det till tre decimalpunkter.
`Round(3.88 ,5)`	Returnerar 5
`Round(1.1 ,1,0.5)`	Returnerar 1,5 I det här exemplet är storleken på steget 1 och basen på stegintervallet 0,5. Intervallen är ...**0.5 <= x <1.5**, 1.5 <= x <2.5, 2.5<= x <3.5...

Sign

Sign() returnerar 1, 0 eller -1 beroende på om **x** är ett positivt tal, 0 eller ett negativt tal.

Syntax:

```
Sign(x)
```

Returnerad datatyp: numeriska

Begränsningar:

Om inget numeriskt värde påträffas returneras NULL.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Sign( 66 )`	Returnerar 1
`Sign( 0 )`	Returnerar 0
`Sign( - 234 )`	Returnerar -1

8.14 Geospatiala funktioner

De här funktionerna används för att hantera geospatiala data i kartvisualiseringar. Qlik Sense följer GeoJSON-specifikationer för geospatiala data och stöder följande:

- Punkt
- Linjesträng
- Polygon
- Multipolygon

För mer information om GeoJSON-specifikationer, se:
≤ GeoJSON.org

Översikt över geospatiala funktioner

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Det finns två kategorier av geospatiala funktioner: aggregering och icke-aggregering.

Aggregeringsfunktioner tar en geometriuppsättning (områdspunkter) som indata och returnerar en enhetlig geometri. Till exempel kan flera områden slås ihop och en enda gräns för det aggregerade området kan ritas ut på kartan.

Icke-aggregerande funktioner tar en enda geometri och returnerar en geometri. Till exempel för funktionen GeoGetPolygonCenter(), om gränsgeometrin för ett område används som indata så returneras punktgeometri (longitud och latitud) för mitten av det området.

Följande är aggregeringsfunktioner:

GeoAggrGeometry

GeoAggrGeometry() används för att aggregera ett antal områden till ett större område, exempelvis aggregera ett antal underregioner till en region.

```
GeoAggrGeometry (field_name)
```

GeoBoundingBox

GeoBoundingBox() används i skript för att aggregera geometri på en yta och beräkna den minsta omskrivande rektangeln som innehåller alla koordinater.

```
GeoBoundingBox (field_name)
```

GeoCountVertex

GeoCountVertex() används för att hitta antal hörn som en polygongeometri innehåller.

```
GeoCountVertex(field_name)
```

GeoInvProjectGeometry

GeoInvProjectGeometry() används för att aggregera geometri på en yta och tillämpa inversionen av en projektion.

```
GeoInvProjectGeometry(type, field_name)
```

GeoProjectGeometry

GeoProjectGeometry() används för att aggregera geometri på en yta och tillämpa en projektion.

```
GeoProjectGeometry(type, field_name)
```

GeoReduceGeometry

GeoReduceGeometry() används för att minska antalet hörn i en geometri och för att aggregera ett antal områden till ett område, samtidigt som gränslinjerna från de enskilda områdena fortfarande visas.

```
GeoReduceGeometry (geometry)
```

Följande är icke-aggregerande funktioner:

GeoGetBoundingBox

GeoGetBoundingBox() används i skript- och diagramuttryck för att beräkna den minsta geospatiala omskrivande rektangeln som innehåller alla koordinaterna i en geometri.

```
GeoGetBoundingBox (geometry)
```

GeoGetPolygonCenter

GeoGetPolygonCenter() används i skript och diagramuttryck för att beräkna och returnera mittpunkten i en geometri.

```
GeoGetPolygonCenter (geometry)
```

GeoMakePoint

GeoMakePoint() används i skript och diagramuttryck för att skapa och tagga en punkt med latitud och longitud.

```
GeoMakePoint (lat_field_name, lon_field_name)
```

GeoProject

GeoProject() används i skript och diagramuttryck för att tillämpa en projektion på en geometri.

```
GeoProject (geo_data)
```

MakeGeoLine

MakeGeoLine() används för att validera geografiska data i JSON och ställa in lämpliga taggar på fältet. **MakeGeoLine()** bekräftar att data är en array med punkter.

```
MakeGeoLine (geo_data)
```

MakeGeoPolygon

MakeGeoPolygon() används för att validera geografisk data i JSON och ställa in lämpliga taggar på fältet. **MakeGeoPolygon()** bekräftar att datan är en array med linjer.

```
MakeGeoPolygon (geo_data)
```

MakeGeoPoint

MakeGeoPoint() används för att validera geografiska data i JSON och ställa in lämpliga taggar på fältet. **MakeGeoPoint()** bekräftar att data är en array med två nummer.

```
MakeGeoPoint (geo_data)
```

MakeGeoRegion

MakeGeoRegion() används för att validera geografisk data i JSON och ställa in lämpliga taggar på fältet. **MakeGeoRegion()** bekräftar att datan är en array med polygoner.

```
MakeGeoRegion (geo_data)
```

GeoAggrGeometry

GeoAggrGeometry() används för att aggregera ett antal områden till ett större område, exempelvis aggregera ett antal underregioner till en region.

Syntax:

```
GeoAggrGeometry(field_name)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

GeoAggrGeometry() kan vanligtvis användas för att kombinera geospatiala gränsdata. Till exempel så kanske du har postnummerområden för förorter i en stad och försäljningsintäkter för varje områden. Om en säljares område täcker flera postnummer kan det vara bra att presentera total försäljning per område, snarare än individuella områden, och visa resultaten på en färgkarta.

GeoAggrGeometry() kan beräkna aggregerade data för de enskilda förortsområdena och generera en sammanslagen områdesgeometri i datamodellen. Om säljområdenas gränser sedan justerar kan de nya sammanslagna områdena och intäkterna visas på kartan när data laddas igen.

Eftersom **GeoAggrGeometry()** är en aggregerande funktion, krävs en **LOAD**-sats med en **Group by**-klausul om du använder den i skriptet.

> *Gränserna på kartan som skapats med **GeoAggrGeometry()** gäller för de sammanslagna områdena. Om du vill visa de enskilda gränslinjerna för de föraggregerade områdena, använd **GeoReduceGeometry()**.*

Exempel:
I detta exempel läses en KML-fil med områdesdata in och sedan läses en tabell med aggregerade områdesdata in.

```
[MapSource]:
LOAD [world.Name],
       [world.Point],
       [world.Area]
 FROM [lib://Downloads/world.kml]
(kml, Table is [World.shp/Features]);

Map:
LOAD world.Name,
     GeoAggrGeometry(world.Area) as [AggrArea]
resident MapSource Group By world.Name;

Drop Table MapSource;
```

GeoBoundingBox

GeoBoundingBox() används i skript för att aggregera geometri på en yta och beräkna den minsta omskrivande rektangeln som innehåller alla koordinater.

En GeoBoundingBoxvisas som en lista med fyra värden: vänster, höger, överst, nederst.

Syntax:

```
GeoBoundingBox(field_name)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

GeoBoundingBox() aggregerar en uppsättning geometrier och returnerar fyra koordinater för den minsta rektangeln som innehåller alla koordinater för den aggregerade geometrin.

Om du vill visualisera resultaten på en karta ska du överföra resultatsträngen med fyra koordinater i polygonformat, tagga det överförda fältet med ett geopolygonformat och dra och släppa det fältet på kartobjektet. De rektangulära rutorna kommer sedan att visas i kartvisualiseringen.

GeoCountVertex

GeoCountVertex() används för att hitta antal hörn som en polygongeometri innehåller.

Syntax:

```
GeoCountVertex(field_name)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

GeoGetBoundingBox

GeoGetBoundingBox() används i skript- och diagramuttryck för att beräkna den minsta geospatiala omskrivande rektangeln som innehåller alla koordinaterna i en geometri.

En geospatial omskrivande rektangel, som skapats av funktionen GeoBoundingBox(), visas som en lista med fyra värden: vänster, höger, överst, nederst.

Syntax:

```
GeoGetBoundingBox(field_name)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

*Använd inte satsen **Group by** i Skriptredigeraren med denna och andra icke-aggregerade geospatiala funktioner, eftersom detta kommer att leda till ett fel vid laddning.*

GeoGetPolygonCenter

GeoGetPolygonCenter() används i skript och diagramuttryck för att beräkna och returnera mittpunkten i en geometri.

I vissa fall är det ett krav att plotta små punkter istället för färgifyllning på en karta. Om de befintliga geospatiala data bara är tillgängliga i form av områdesgeometri (till exempel en gräns) ska du använda **GeoGetPolygonCenter()** för att hämta ett par med longitud och latitud för mittenområdet.

Syntax:

```
GeoGetPolygonCenter(field_name)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

*Använd inte satsen **Group by** i Skriptredigeraren med denna och andra icke-aggregerade geospatiala funktioner, eftersom detta kommer att leda till ett fel vid laddning.*

GeoInvProjectGeometry

GeoInvProjectGeometry() används för att aggregera geometri på en yta och tillämpa inversionen av en projektion.

Syntax:

```
GeoInvProjectGeometry(type, field_name)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
type	Projektionstyp som används för att omvandla kartans geometri. Den kan ta ett av två värden: ”enhet” (standard), som ger en 1:1-projektion, eller ”mercator”, som ger standardprojektionen Mercator.
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

Exempel:

Skriptexempel

Exempel	Resultat
I en `Load`-sats: `GeoInvProjectGeometry ('mercator',AreaPolygon) as InvProjectGeometry`	Geometrin laddas som **AreaPolygon** omvandlas med hjälp av inverterad omvandling av Mercator-projektionen och sparas som **InvProjectGeometry** för att användas i visualiseringar.

GeoMakePoint

GeoMakePoint() används i skript och diagramuttryck för att skapa och tagga en punkt med latitud och longitud. GeoMakePoint returnerar punkter i ordningen longitud-latitud.

Syntax:

```
GeoMakePoint(lat_field_name, lon_field_name)
```

Returnerad datatyp: sträng, formaterad [longitud, latitud]

Argument:

Argument

Argument	Beskrivning
lat_field_name	Ett fält eller ett uttryck som hänvisar till ett fält som representerar punktens latitud.
lon_field_name	Ett fält eller ett uttryck som hänvisar till ett fält som representerar punktens longitud.

*Använd inte satsen **Group by** i Skriptredigeraren med denna och andra icke-aggregerade geospatiala funktioner, eftersom detta kommer att leda till ett fel vid laddning.*

GeoProject

GeoProject() används i skript och diagramuttryck för att tillämpa en projektion på en geometri.

Syntax:

```
GeoProject(type, field_name)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
type	Projektionstyp som används för att omvandla kartans geometri. Den kan ta ett av två värden: ”enhet” (standard), som ger en 1:1-projektion, eller ”mercator”, som ger webbprojektionen Mercator.
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

*Använd inte satsen **Group by** i Skriptredigeraren med denna och andra icke-aggregerade geospatiala funktioner, eftersom detta kommer att leda till ett fel vid laddning.*

Exempel:

Skriptexempel

Exempel	Resultat
I en `Load`-sats: `GeoProject ('mercator',Area) as GetProject`	Mercator-projektionen tillämpas på geometrin som laddats som **Area**, och resultatet sparas som **GetProject**.

GeoProjectGeometry

GeoProjectGeometry() används för att aggregera geometri på en yta och tillämpa en projektion.

Syntax:

```
GeoProjectGeometry(type, field_name)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
type	Projektionstyp som används för att omvandla kartans geometri. Den kan ta ett av två värden: ”enhet” (standard), som ger en 1:1-projektion, eller ”mercator”, som ger webbprojektionen Mercator.
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

Exempel:

Exempel	Resultat
I en `Load`-sats: `GeoProjectGeometry ('mercator',AreaPolygon) as ProjectGeometry`	Geometrin som laddats som **AreaPolygon** omvandlas med hjälp Mercator-projektion och sparas som **ProjectGeometry** för att användas i visualiseringar.

GeoReduceGeometry

GeoReduceGeometry() används för att minska antalet hörn i en geometri och för att aggregera ett antal områden till ett område, samtidigt som gränslinjerna från de enskilda områdena fortfarande visas.

Syntax:

```
GeoReduceGeometry(field_name[, value])
```

Returnerad datatyp: JSON

Argument:

Argument

Argument	Beskrivning
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.
value	Mängden reduktion som ska tillämpas på geometrin. Intervallet är från 0 till 1, där 0 representerar ingen reduktion och 1 representerar maximal reduktion av hörn. *Användning av ett value på 0,9 eller högre med en komplex datauppsättning kan minska antalet hörn till en nivå där den visuella återgivningen är felaktig.*

GeoReduceGeometry() utför även en funktion som liknar **GeoAggrGeometry()** genom att det aggregerar ett antal områden till ett enda. Skillnaden är att enskilda gränslinjer från förhandsaggregerade data visas på mappan om du använder **GeoReduceGeometry()**.

Eftersom **GeoReduceGeometry()** är en aggregerande funktion, om du använder den i skriptet krävs en **LOAD**-sats med en **Group by**-klausul.

Exempel:
I detta exempel läses en KML-fil med områdesdata in och sedan läses en tabell in med minskade och aggregerade områdesdata.

```
[MapSource]:
LOAD [world.Name],
       [world.Point],
       [world.Area]
 FROM [lib://Downloads/world.kml]
(kml, Table is [World.shp/Features]);

Map:
LOAD world.Name,
     GeoReduceGeometry(world.Area,0.5) as [ReducedArea]
resident MapSource Group By world.Name;

Drop Table MapSource;
```

MakeGeoLine

MakeGeoLine() används för att validera geografiska data i JSON och ställa in lämpliga taggar på fältet. **MakeGeoLine()** bekräftar att data är en array med punkter.

Syntax:

```
MakeGeoLine(geo_data)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
geo_data	Källan till geografisk data. Detta kan konstrueras med andra geospatiala funktioner, strängliteraler, JSON-funktioner, KML-data eller andra källor. **geo_data** valideras och returnerar endast giltig data av den typen. Ogiltig data returnerar istället ett NULL-värde.

*Använd inte satsen **Group by** i Skriptredigeraren med denna och andra icke-aggregerade geospatiala funktioner, eftersom detta kommer att leda till ett fel vid laddning.*

MakeGeoPoint

MakeGeoPoint() används för att validera geografiska data i JSON och ställa in lämpliga taggar på fältet. **MakeGeoPoint()** bekräftar att data är en array med två nummer.

Syntax:

```
MakeGeoPoint(geo_data)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
geo_data	Källan till geografisk data. Detta kan konstrueras med andra geospatiala funktioner, strängliteraler, JSON-funktioner, KML-data eller andra källor. **geo_data** valideras och returnerar endast giltig data av den typen. Ogiltig data returnerar istället ett NULL-värde.

*Använd inte satsen **Group by** i Skriptredigeraren med denna och andra icke-aggregerade geospatiala funktioner, eftersom detta kommer att leda till ett fel vid laddning.*

MakeGeoPolygon

MakeGeoPolygon() används för att validera geografisk data i JSON och ställa in lämpliga taggar på fältet. **MakeGeoPolygon()** bekräftar att datan är en array med linjer.

Syntax:

```
MakeGeoPolygon(geo_data)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
geo_data	Källan till geografisk data. Detta kan konstrueras med andra geospatiala funktioner, strängliteraler, JSON-funktioner, KML-data eller andra källor. **geo_data** valideras och returnerar endast giltig data av den typen. Ogiltig data returnerar istället ett NULL-värde.

*Använd inte satsen **Group by** i Skriptredigeraren med denna och andra icke-aggregerade geospatiala funktioner, eftersom detta kommer att leda till ett fel vid laddning.*

MakeGeoRegion

MakeGeoRegion() används för att validera geografisk data i JSON och ställa in lämpliga taggar på fältet. **MakeGeoRegion()** bekräftar att datan är en array med polygoner.

Syntax:

```
MakeGeoRegion(geo_data)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
geo_data	Källan till geografisk data. Detta kan konstrueras med andra geospatiala funktioner, strängliteraler, JSON-funktioner, KML-data eller andra källor. **geo_data** valideras och returnerar endast giltig data av den typen. Ogiltig data returnerar istället ett NULL-värde.

*Använd inte satsen **Group by** i Skriptredigeraren med denna och andra icke-aggregerade geospatiala funktioner, eftersom detta kommer att leda till ett fel vid laddning.*

8.15 Tolkningsfunktioner

Tolknignsfuntionerna utvärderar innehållet i textdatafält eller uttryck och fastställer ett angivet dataformat för det resulterande numeriska värdet. Med de här funktionerna kan du ange format för talet, i enlighet med dess datatyp, vilket kan vara sådana attribut som t.ex. decimalavgränsare, tusendelsavgränsare och datumformat.

Tolkningsfunktionerna returnerar ett dualt värde med både strängen och det numeriska värdet, men det kan också ses som att de gör en konvertering från sträng till tal. Funktionerna tar textvärdet i indatauttrycken och generera ett tal som representerar strängen.

Formateringsfunktionerna däremot gör det motsatta: de tar taluttryck och utvärderar dem som strängar och specificerar sedan format för den resulterande texten.

Om inga tolkningsfunktioner används, tolkar Qlik Sense data som en blandning av tal, datum, tider, tidsmarkörer och strängar och använder standardinställningar för talformat, datumformat och tidsformat, som har definierats av skriptvariabler och av operativsystemet.

Alla tolkningsfunktioner kan användas både i dataladdningsskript och diagramuttryck.

> *Alla numeriska värden anges med en decimalpunkt som decimalavgränsare.*

Tolkningsfunktioner - en översikt

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Date#

Date# utvärderar ett uttryck som ett datum i det format som har angetts i det andra argumentet, om så är tillämpligt. Om formatkoden utelämnas används det standarddatumformat som definierats i operativsystemet.

```
Date#(text[, format])
```

Interval#

Interval#() utvärderar ett textuttryck som ett tidsintervall i formatet som har ställts in för operativsystemet, som standard eller i formatet som anges i det andra argumentet, om så är tillämpligt.

```
Interval#(text[, format])
```

Money#

Money#() konverterar en textsträng till ett monetärt värde i det format som har ställts in i laddningsskriptet eller operativsystemet, om inte en formatsträng tillhandahålls. Egna symboler för decimaler och tusentalsseparatorer är valbara parametrar.

```
Money#(text[, format[, dec_sep[, thou_sep ] ] ])
```

Num#

Num#() tolkar en textsträng som ett numeriskt värde, vilket innebär att indatasträngen konverteras till ett tal med det format som specificerats i den andra parametern. Om den andra parametern utelämnas använder funktionen de decimal- och tusentalsavgränsare som anges i dataladdningsskriptet. Egna symboler för decimaler och tusentalsseparatorer är valbara parametrar.

```
Num#(text[ , format[, dec_sep[ , thou_sep]]])
```

Text

Text() tvingar en tolkning av uttrycket som text, även om en numerisk tolkning är möjlig.

```
Text(expr)
```

Time#

Time#() utvärderar ett uttryck som ett tidsvärde, i det tidsformat som har ställts in i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns..

```
Time#(text[, format])
```

Timestamp#

Timestamp#() utvärderar ett uttryck som ett datum- och tidsvärde, i det tidsmarkörformat som har ställts in i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns.

```
Timestamp# (text[, format])
```

Se även:

p *Formateringsfunktioner (page 1371)*

Date#

Date# utvärderar ett uttryck som ett datum i det format som har angetts i det andra argumentet, om så är tillämpligt.

Syntax:

```
Date#(text[, format])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
text	Textsträngen som ska utvärderas.

Argument	Beskrivning
format	Sträng som beskriver formatet som ska användas för att tolka textsträngen. Om datumformatet utelämnas används det datumformat som är inställt i systemvariablerna i dataladdningsskriptet eller operativsystemet. Funktionen `Date#` har stöd för många formatalternativ för olika datumstrukturer, t.ex: `DD/MM/YYYY`: Dag/Månad/År (t.ex. 15/02/2022) `MM-DD-YYYY`: Månad-Dag-År med bindestreck (t.ex. 02-15-2022) `YYYY-MM-DD`: År-Månad-Dag, ISO-format (t.ex. 2022-02-15) `D-MMM-YYYY`: Dag, kort månadsnamn, År, med bindestreck (t.ex. 15-Feb-2022) `MMMM YYYY`: Långt månadsnamn och År (t.ex. February 2022) `MM/YYYY`: Månad och År (t.ex. 02/2022) `DD MMMM YYYY`: Dag, långt månadsnamn och År (t.ex. 15 February 2022)

Exempel: diagramuttryck

Exempel	Resultat
`Date#('8/7/97')`	Returnerar ett giltigt datum för `8/7/97` när `DateFormat`-inställningen är `M/D/YY`.
`Date#('21.10.2023', 'DD.MM.YYYY' )`	Returnerar ett giltigt datum för `21.10.2023` när datuminställningen är `DD.MM.YYYY`.
`Date#('21.10.2023')`	Returnerar `NULL` när inställningen för `DateFormat` är `M/D/YY`.

Exempel – grunderna i datum#

Laddningsskript

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i en datatabell:
  - `StringDate`
- Variabeln `DateFormat`, som använder formatet `M/D/YY`.
- En föregående last som skapar ett nytt fält som heter `Date`. Funktionerna `Date#` och `Num` används för att konvertera strängen till ett datum och sedan returnera värdet som ett tal.

Laddningsskript

```
SET DateFormat='M/D/YY';
Example:
Load *,
Num(Date#(StringDate, 'D/M/YYYY')) as Date
;
LOAD * INLINE [
StringDate
13/12/2024
31/1/23
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `StringDate`
- `Date`

Skapa följande mått:

- `=Date(Date)` för att visa värdet som ett datum som är formaterat med den datumformatsvariabel som anges i laddningsskriptet.
- `=IsNum(StringDate)`, för att kontrollera att värdet är en sträng och inte utvärderas som ett datum eftersom datumformatet för fältvärdena inte stämmer överens med systeminställningen.

Results table

StringDate	Date	Date(Date)	IsNum(StringDate)
13/12/2024	45639	12/13/24	0
31/1/23	44957	1/31/23	0

Resultatet visar att fältet `Date` i kolumn 2, som skapades av den föregående laddningen, har konverterat `StringDate`-värdena till ett datum som uttrycks som ett tal. Formatfunktionen `Date` i kolumn 3 känner igen `Date`-värdena som datum och formaterar dem enligt variabeln `DateFormat` i laddningsskriptet.

Exempel – Datum#-scenario

Laddningsskript

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `Customer`
    - `BirthDate`
- En föregående last som skapar ett nytt fält som heter `ParsedBirthDate`.

Laddningsskript

```
Example:
Load *,
Date#(BirthDate, 'MM.DD.YYYY') as ParsedBirthDate;
Load * inline [
Customer, BirthDate
Customer A, 08.25.2022
Customer B, 08.25.2022
Customer C, 08.25.2022
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Customer`
- `BirthDate`
- `ParsedBirthDate`

Skapa följande mått:

- `=Date(ParsedBirthDate,'DD/MM/YYYY')` för att visa värdet som ett formaterat datum.

Results table

Customer	BirthDate	ParsedBirthDate	Date(ParsedBirthDate, 'DD/MM/YYYY')
Customer A	08.25.2022	08.25.2022	25/08/2022
Customer B	06.29.2022	06.29.2022	29/06/2022
Customer C	05.31.2022	05.31.2022	31/05/2022

I det här exemplet kan vi se att i kolumn 2 behandlas fältet `BirthDate` som text av systemet. Funktionen `Date#` som användes i föregående laddning konverterar `BirthDate` till ett igenkännbart datum i kolumn 3, `ParsedBirthDate`. Formatfunktionen `Date` i kolumn 4 kan sedan visa `ParsedBirthDate`--värdena i formatet `DD/MM/YYYY`.

Interval#

Interval#() utvärderar ett textuttryck som ett tidsintervall i formatet som har ställts in för operativsystemet, som standard eller i formatet som anges i det andra argumentet, om så är tillämpligt.

Syntax:

```
Interval#(text[, format])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
text	Textsträngen som ska utvärderas.
format	En sträng som beskriver det förväntade indataformat som ska användas när man omvandlar strängen till ett numeriskt intervall. Om den utelämnas används operativsystemets datumformat och decimalavgränsare. Funktionen `Interval#` har stöd för många formatalternativ för olika tidsstrukturer, t.ex: `hh:mm` – bara timmar och minuter `hh:mm:ss` – timmar, minuter och sekunder `d hh:mm` – dagar, timmar och minuter

Funktionen **Interval#** omvandlar ett texttidsintervall till en numerisk motsvarighet.

Exempel: diagramuttryck

Exempel	Resultat
`Interval#( '12:30', 'hh:mm' )`	Returnerar det numeriska värdet `0.5208`, som är 12.5 timmar när det omvandlas till en del av en dag.
`Interval#( '1 05', 'd hh' )`	Returnerar en numerisk representation som motsvarar `1.2083` dagar.

Exempel – grunderna i intervall#

Laddningsskript

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i en datatabell:
  - `IntervalValue`
- En tidigare laddning som använder funktionen `Interval#` för att konvertera texten i fältet `IntervalValue` till ett tal, som kan identifieras som en tid.

Laddningsskript

```
Example:
Load *, Interval#(IntervalValue, 'D hh') as FormattedIntervalValue;
Load * inline [
IntervalValue
1 05
2 15
1 09
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `IntervalValue`

Skapa följande mått:

- `=Num(FormattedIntervalValue)`, för att visa värdet som ett tal efter att det har bearbetats i laddningsskriptet med hjälp av funktionen `Interval#`.
- `=Interval(FormattedIntervalValue, 'dd:hh:mm')` för att visa värdet som ett intervallformat med det format som anges i funktionen.

Resultattabell

IntervalValue	Num(FormattedIntervalValue)	Interval(FormattedIntervalValue, 'dd:hh:mm'
1 05	1.2083333333333	01:05:00
1 09	1.375	01:09:00
2 15	2.625	02:15:00

Resultaten visar hur funktionen `Interval#` omvandlar värden till ett tal som sedan kan visas med hjälp av en angiven formatparameter. I kolumnen `IntervalValue` omvandlas t.ex. värdet `2 15` till ett tal i den föregående laddningen (fältet `FormattedIntervalValue`). I den andra kolumnen används funktionen `Num` för att visa det konverterade värdet som ett tal, `2.625`. I den tredje kolumnen används funktionen `Interval` för att formatera detta tal i dagar, timmar och minuter och visa det som `02:15:00`, eller 2 dagar 15 timmar.

Exempel – Intervall#-scenario

Laddningsskript

Översikt

En datauppsättning med supportärende innehåller ett fält som blandar text och tidsdata, till exempel `2 days 04 hours`. Vi vill konvertera dessa värden till normaltidsvärden.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `TicketID`
  - `ResolutionTime`
- En tidigare laddning som använder funktionen `Interval#` för att konvertera texten i fältet `ResolutionTime` till ett tal som kan identifieras som dagar och timmar.

Laddningsskript

```
Example:
Load *, Interval#(ResolutionTime, 'd "days" hh "hours"') as FormattedResolutionTime;
Load * inline [
TicketID, ResolutionTime
Ticket A, 2 days 04 hours
Ticket B, 2 days 06 hours
Ticket C, 3 days 10 hours
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `TicketID`
- `ResolutionTime`

Skapa följande mått:

- `=Interval(FormattedResolutionTime, 'dd:hh')`, för att konvertera värdena till önskat format med ett dubbelt värde.
- `=Num(FormattedResolutionTime)`, för att visa fältets numeriska värden.

Resultattabell

TicketID	ResolutionTime	Interval (FormattedResolutionTime, 'dd hh')	Num (FormattedResolutionTime)
Ärende A	2 dagar 04 timmar	02 04	2.1666666666667
Ärende B	2 dagar 06 timmar	02 06	2.25
Ärende C	3 dagar 10 timmar	03 10	3.4166666666667

Resultaten visar hur du kan använda funktionen `Interval#` för att konvertera värden till ett tal och sedan visa dem i ett mer typiskt tidsformat. Till exempel har `Ticket B` en upplösningstid på `2 days 06 hours`. Värdet konverteras till ett tal i den föregående laddningen (fältet `FormattedResolutionTime`). Det nya värdet visas som `02 06` (2 dagar och 6 timmar) när det formateras med funktionen `Interval` i kolumn 3. I kolumn 4 visar funktionen `Num` sitt numeriska värde som `2.25`.

Money#

Money#() konverterar en textsträng till ett monetärt värde i det format som har ställts in i laddningsskriptet eller operativsystemet, om inte en formatsträng tillhandahålls. Egna symboler för decimaler och tusentalsseparatorer är valbara parametrar.

Syntax:

```
Money#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
text	Textsträngen som ska utvärderas.
format	En sträng som beskriver det förväntade indataformat som ska användas när man omvandlar strängen till ett numeriskt intervall. Om det utelämnas används det värde som angetts i dataladdningsskriptet för `MoneyFormat`. Funktionen `Money#` har stöd för många formatalternativ för olika money-strukturer, t.ex: `$#,##0.00`: Lägger till ett dollartecken och två decimaler. `€ #,##0`: Lägger till ett eurotecken, inga decimaler och ett blanksteg efter eurotecknet. `¥#,##0`: Lägger till ett tecken för japanska yen utan decimaler. `$#,##0;($#,##0)`: Formaterar negativa värden inom parentes. `₣ #.##0,00`: Lägger till ett tecken för schweizerfranc med punkter som tusentalsavgränsare och ett kommatecken för decimalavgränsare (till exempel `₣ 1.000,00`). `R #,##0.00`: Sydafrikanskt randtecken med två decimaler och blanksteg efter randtecknet. `$ #,##0.000`: Lägger till tre decimaler i dollarformatet.
dec_sep	Sträng som specificerar decimalavgränsare. Om det utelämnas används det `MoneyDecimalSep`-värde som angetts i dataladdningsskriptet. Funktionen `Money#` stöder många format för decimalavgränsare, t.ex: Komma ' , ': Till exempel, `$1,000,00` för ettusen dollar. Punkt ' . ': Till exempel `€ 1,000.00`. Bindestreck ' - ': När det används i stället för den traditionella punkten eller kommatecknet som decimalavgränsare visar det värden som `¥1,000-00`.

Argument	Beskrivning
thou_sep	Sträng som specificerar tusendelsavgränsare. Om det utelämnas används det MoneyThousandSep-värde som angetts i dataladdningsskriptet. Funktionen Money# stöder många alternativ för tusentalsavgränsaren: Punkt '.': Använder en punkt som tusentalsavgränsare (vanligt i vissa europeiska format), visas som $1.000,00 om den kombineras med en decimalavgränsare med kommatecken. Mellanslag ' ': Använder ett mellanslag som tusentalsavgränsare, vilket är vanligt i vissa europeiska länder. Visas som € 1 000.00. Understreck '_':Använder ett understreck för tusentalsavgränsaren, som ibland används i tekniska eller specifikt regionala format. Visar värden som $1_000.00.

Funktionen Money# fungerar vanligtvis som funktionen Num#, men hämtar standardvärdena för decimal- och tusenavgränsare från skript-variablerna för valutaformat eller från systemets inställningar för valuta.

Exempel: diagramuttryck

Exempel	Resultat
Money#('35 648,37 kr' , '# ##0,00 kr',',',' ')	Returnerar 35 648,37 kr när inställningen för MoneyFormat är # ##0,00 kr. Följande måste ställas in i ditt laddningsskript för att detta ska utvärderas korrekt som ett tal: • SET MoneyDecimalSep=','; • SET MoneyThousandSep=' ';
Money#('$35,648.37', ' $#', '.', ',')	Returnerar $35,648.37 med följande inställning: • MoneyFormatär $# • SET MoneyDecimalSep=','; • SET MoneyThousandSep=' ';

Exempel – grunderna i Money#

Laddningsskript

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter Example.
- Ett fält i datatabellen heter MoneyValue.

- En föregående laddning som använder funktionen `Money#` för att konvertera textvärdena i data till ett värde som Qlik Sense kommer att känna igen som ett giltigt värde.

Laddningsskript

```
SET MoneyFormat='$ #,##0.00';
SET MoneyThousandSep=',';
Example:
Load *, Money#(MoneyValue, '# ##0,00 kr',',',' ') as FormattedMoneyValue;
Load * inline [
MoneyValue
35 648,37 kr
56 735,56 kr
] (delimiter is ';');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `MoneyValue`
- `FormattedMoneyValue`

Skapa följande mått:

- `=Money( FormattedMoneyValue, 'kr #,##0', '.' , ',' )`, för att se värdet som pengar, formaterat med de argument som anges i funktionen `Money`.
- `=Num(FormattedMoneyValue)` för att returnera det numeriska värdet efter att det har bearbetats av funktionen `Money#`.

Results table

MoneyValue	FormattedMoneyValue	Money(FormattedMoneyValue, 'kr #,##0', '.' , ',')	Num (FormattedMoneyValue)
56 735,56 kr	56 735,56 kr	kr 56,736	56735.56
35 648,37 kr	35 648,37 kr	kr 35,648	35648.37

Resultattabellen visar hur värden kan visas med hjälp av följande funktioner och format: `Money#`, `Money` och `Num`. I den andra kolumnen visas dimensionen `FormattedMoneyValue`, som beräknades med hjälp av funktionen `Money#` för att ändra värdet till pengar med formatet `'# ##0,00 kr',',',' '`. I den tredje kolumnen används funktionen `Money` för att ändra valutaformatet från `FormattedMoneyValue` till `'kr #,##0', '.' , ','`. Den fjärde kolumnen använder funktionen `Num` för att visa det numeriska värdet för `FormattedMoneyValue`-dimensionen med hjälp av formatet i systemvariabeln.

Exempel – Money#-scenario

Laddningsskript

Översikt

En datauppsättning med försäljningsorder innehåller fält som använder olika valutor. I det här exemplet visas hur du separerar och summerar försäljningsvärdena baserat på valuta.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `TransactionID`
  - `Currency`
  - `Amount`
- Systemvariablerna `MoneyDecimalSep`, `MoneyFormat` och `MoneyThousandSep` är inställda på att använda ett annat format än de som används i datan i demonstrationssyfte.
- En föregående laddning som laddar fälten `TransactionID`, `Currency` och `Amount`, och skapar följande fält:
  - `AmountValue`, detta fält använder funktionen `Money#` för att konvertera den text som det tar emot till ett värde som systemet förstår.

Laddningsskript

```
SET MoneyDecimalSep='-';
SET MoneyFormat='###0-00';
SET MoneyThousandSep=';';
Example:
LOAD
TransactionID,
Currency,
Amount,
IF(Currency = 'USD', Money#(Amount, '$#,##0.00','.',','),
IF(Currency = 'EUR', Money#(Amount, '€ #.##0,00',',','.'))) AS AmountValue
INLINE [
TransactionID, Currency, Amount
"TXN001", "USD", "$ 1,500.00"
"TXN002", "EUR", "€ 1.200,75"
"TXN003", "USD", "$ 1,750.50"
"TXN004", "EUR", "€ 950,00"
"TXN005", "USD", "$ 2,600.25"
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `TransactionID`
- `Currency`
- `Amount`
- `AmountValue`

Skapa följande mått:

- `Num(AmountValue)`

Resultattabell

TransactionID	Currency	Amount	AmountValue	Num(AmountValue)
TXN001	USD	$1,500.00	$ 1,500.00	1500
TXN002	EUR	€ 1.200,75	€ 1.200,75	1200.75
TXN003	USD	$ 1,750.50	$ 1,750.50	1750.5
TXN004	EUR	€ 950,00	€ 950,00	950
TXN005	USD	$ 2,600.25	$ 2,600.25	2600.25

Resultatet visar hur funktionen `Money#` omvandlar textvärden till monetära numeriska värden som sedan kan utvärderas. Värdena i fältet `Amount` är t.ex. text och visas med vänsterjustering. Men när dessa har bearbetats av funktionen `Money#` för att skapa det nya fältet `AmountValue` känner systemet nu igen dem som monetära numeriska värden. Funktionen `Num`, som används i kolumnen längst till höger, kan nu utvärdera de monetära värdena i `AmountValue` och ändra formatet från pengar till ett tal.

Num#

Num#() tolkar en textsträng som ett numeriskt värde, vilket innebär att indatasträngen konverteras till ett tal med det format som specificerats i den andra parametern. Om den andra parametern utelämnas använder funktionen de decimal- och tusentalsavgränsare som anges i dataladdningsskriptet. Egna symboler för decimaler och tusentalsseparatorer är valbara parametrar.

Syntax:

```
Num#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Returnerad datatyp: dual

Funktionen **Num#()** returnerar ett dualt värde med både strängvärdet och det numeriska värdet. Funktionen tar indatauttryckets textrepresentation och genererar ett tal. Talets format ändras inte: utdata formateras på samma sätt som indata.

Argument

Argument	Beskrivning
text	Textsträngen som ska utvärderas.

Argument	Beskrivning
format	Sträng som specificerar talformatet som används i den första parametern. Om den utelämnas används de decimal- och tusentalsavgränsare som anges i dataladdningsskriptet. Funktionen `Num#` stöder många formatalternativ för olika datumstrukturer, t.ex: `#,##0`: Visar ett tal utan decimaler. `#,##0.00`: Visar ett tal med två decimaler. `##0.00%`: Visar ett tal som ett procenttal med två decimaler. `#,##0;(#,##0)`: Formaterar negativa tal till att visas inom parentes. `$#,##0.00`: Formaterar tal som valuta med en $-symbol. `0.00E+0`: Visar tal i vetenskaplig notation.
dec_sep	Sträng som specificerar decimalavgränsare. Om det utelämnas används värdet för variabeln DecimalSep som angetts i dataladdningsskriptet. Funktionen `Num#` stöder många format för decimalavgränsare, t.ex: Komma ' , ': Till exempel, `$1,000,00` för ettusen dollar. Punkt ' . ': Till exempel `€ 1,000.00`. Bindestreck ' - ': När det används i stället för den traditionella punkten eller kommatecknet som decimalavgränsare visar det värden som `¥1,000-00`. Mer information finns i *DecimalSep (page 255)*.
thou_sep	Sträng som specificerar tusendelsavgränsare. Om det utelämnas används värdet för variabeln ThousandSep som angetts i dataladdningsskriptet. Funktionen `Num#` stöder många alternativ för tusentalsavgränsaren: Punkt ' . ': Använder en punkt som tusentalsavgränsare (vanligt i vissa europeiska format), visas som `$1.000,00` om den kombineras med en decimalavgränsare med kommatecken. Mellanslag ' ': Använder ett mellanslag som tusentalsavgränsare, vilket är vanligt i vissa europeiska länder. Visas som `€ 1 000.00`. Understreck '_':Använder ett understreck för tusentalsavgränsaren, som ibland används i tekniska eller specifikt regionala format. Visar värden som `$1_000.00`. Mer information finns i *ThousandSep (page 291)*.

Exempel: diagramuttryck

Exempel	Resultat (returnerat strängvärde)
`Num#( '35,648.31', '#', '.' , ',' )`	Returnerar strängen `35,648.31` och det numeriska värdet `35648.31`.

Exempel	Resultat (returnerat strängvärde)
`Num#( ‘35 648.312’, '#', '.' , ',' )`	Returnerar strängen `35 648.312` och det numeriska värdet `35648.312`.
`Num#( '35.648,3123', '#', '.' , ',' )`	Returnerar strängen `35.648,3123` och det numeriska värdet `NULL` (–).
`Num#( '35 648,31234', '#', '.' , ',' )`	Returnerar strängen `35 648,31234` och det numeriska värdet `NULL` (–).
`Num#('1,234.56', '#,##0.00')`	Returnerar strängen `1,234.56` och det numeriska värdet `1230.45`.
`Num#('$1,230.45', '$#,##0.00')`	Returnerar strängen `$1,230.45` och det numeriska värdet `1234.56`.
`Num#('20220315', 'YYYYMMDD')`	Returnerar strängen `20220315` och det numeriska värdet `20220315`.

Exempel – grunderna i Num#

Laddningsskript

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ProductID`
  - `ProductName`
  - `SalesAmount`
- Standardinställningarna för `ThousandSep` och `DecimalSep` för att illustrera detta exempel.
- En föregående laddning som använder funktionen `Num#`. När den körs skapas ett extra fält som heter `FormattedSalesAmount`.

Laddningsskript

```
SET ThousandSep=',';
SET DecimalSep='.';
Example:
LOAD
ProductID,
ProductName,
```

```
SalesAmount,
Num#(SalesAmount, '#.##0,00',',','.') AS FormattedSalesAmount
INLINE [
ProductID, ProductName, SalesAmount
1, Product A, '1.234,56'
2, Product B, '2.456,78'
3, Product C, '345,67'
4, Product D, '987,12'
5, Product E, '1.200,45'
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ProductID`
- `ProductName`
- `SalesAmount`
- `FormattedSalesAmount`

Skapa följande mått:

- `=Num(FormattedSalesAmount)`, för att visa fältets numeriska värde enligt systeminställningarna för talformatet.

Results table

ProductID	ProductName	SalesAmount	FormattedSalesAmount	Num (FormattedSalesAmount)
1	Product A	1.234,56	1.234,56	1234.56
2	Product B	2.456,78	2.456,78	2456.78
3	Product C	345,67	345,67	345.67
4	Product D	987,12	987,12	987.12
5	Product E	1.200,45	1.200,45	1200.45

I det här exemplet kan du se hur funktionen `Num#` omvandlar `SalesAmount`-data från sitt ursprungliga textformat till ett numeriskt värde.

Exempel – Num#-scenario

Laddningsskript

Översikt

En datauppsättning innehåller försäljningsdata från flera länder i olika textformat. Vi vill standardisera data och visa dem i en tabell.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Country`
  - `Product`
  - `Amount`
- En tidigare laddning som använder funktionen `Num#` för att konvertera värdena i fältet `Amount`.

Laddningsskript

```
Example:
LOAD
Country,
Product,
IF(Country = 'Germany', Num#(Amount, '#.##0,00', ',', '.'), Num#(Amount, '#,##0.00')) AS
Amount
INLINE [
Country, Product, Amount
Germany, Product A, '1.234,56'
USA, Product B, '2,345.78'
Germany, Product C, '987,12'
USA, Product D, '1,200.45'
Germany, Product E, '345,67'
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Product`
- `Country`
- `Amount`

Skapa följande mått:

- `Num(Amount)` för att visa `Amount`-fältets numeriska värde enligt systeminställningarna för talformatet.

Results table

Product	Country	Amount	Num(Amount)
Customer A	Germany	1.234,56	1234.56
Customer B	USA	2,345.78	2345.78
Customer C	Germany	987,12	987.12
Customer D	USA	1,200.45	1200.45
Customer E	Germany	345,67	345.67

När man jämför kolumnen `Amount` med kolumnen `Num(Amount)` ser man att de data som presenteras med funktionen `Num#` ändrar de underliggande numeriska värdena men behåller strängformatet som det är. Till exempel använder alla värden för `Germany` i kolumnen `Amount` ett kommatecken som decimalavgränsare, men det underliggande formatet i `Num(Amount)` använder en punkt för alla värden.

Text

Text() tvingar en tolkning av uttrycket som text, även om en numerisk tolkning är möjlig.

Syntax:

```
Text (expr)
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
expr	Ett uttryck som ska behandlas som text.

Exempel: diagramuttryck

Exempel	Resultat
`Text( 1234 )`	Returnerar `1234` som en sträng. Det finns inget numeriskt värde.
`Text( pi( ) )`	Returnerar `3.1415926535898` som en sträng. Det finns inget numeriskt värde.

Exempel – grunderna i Text

Laddningsskript

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ID`
  - `PostalCode`
- En föregående laddning som använder funktionen `Text` för att skapa ett nytt fält med namnet `PostalCode_Text` för att säkerställa att postnumren förblir som text och inte konverteras till siffror.

Laddningsskript

```
Example:
LOAD
ID,
PostalCode,
```

```
Text(PostalCode) AS PostalCode_Text;
LOAD * INLINE [
ID, PostalCode
1, 12345
2, 54321
3, 67890
4, 11223
5, 33445
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ID`
- `PostalCode`
- `PostalCode_Text`

Results table

ID	PostalCode	PostalCode_Text
1	12345	12345
2	54321	54321
3	67890	67890
4	11223	11223
5	33445	33445

Utdata från föregående laddning använder funktionen `Text` för att säkerställa att alla numeriska värden behandlas som text. Du kan se att värdena i kolumnen `PostalCode_Text` behandlas som text och visas vänsterjusterade som standard, medan värdena i kolumnen `PostalCode` är siffror och visas högerjusterade som standard.

Exempel – Text-scenario

Laddningsskript

Översikt

En datauppsättning innehåller kundfakturadata som anges som numeriska värden, men du vill behålla dessa värden som text.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `InvoiceNumber`
  - `CustomerName`
- En föregående laddning som använder funktionen `Text` för att skapa ett nytt fält med namnet `InvoiceNumber_Text` för att säkerställa att fakturanumret behandlas som text.

Laddningsskript

```
Example:
LOAD
InvoiceNumber,
CustomerName,
Text(InvoiceNumber) AS InvoiceNumber_Text;
LOAD * INLINE [
InvoiceNumber, CustomerName
10001, John Doe
10002, Jane Smith
10003, Bob Johnson
10004, Alice Brown
10005, Michael Green
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `InvoiceNumber`
- `CustomerName`
- `InvoiceNumber_Text`

Results table

InvoiceNumber	CustomerName	InvoiceNumber_Text
10001	John Doe	10001
10002	Jane Smith	10002
10003	Bob Johnson	10003
10004	Alice Brown	10004
10005	Michael Green	10005

Resultatet visar att de numeriska värdena i kolumnen `InvoiceNumber` nu har konverterats till textvärden i kolumnen `InvoiceNumber_Text` och att de är vänsterjusterade.

Time#

Time#() utvärderar ett uttryck som ett tidsvärde, i det tidsformat som har ställts in i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns.

Syntax:

```
time#(text[, format])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
text	Textsträngen som ska utvärderas.
format	Sträng som beskriver det format som ska användas för att tolka textsträngen. Om den utelämnas används operativsystemets tidsformat och decimalavgränsare. Funktionen `Time#` har stöd för många formatalternativ för olika tidsstrukturer, t.ex: `hh:mm:ss` – formaterar ett tidsvärde i ett standardtidsformat och visar timmar, minuter och sekunder med inledande nollor. `hh:mm TT` – Använder ett 12-timmarsformat med suffixet AM/PM. `hh:mm` – Visar endast timmar och minuter och utelämnar sekunder. `hh.mm.ss` – tidsformat med olika avdelare, t.ex. en punkt (`.`) i stället för ett kolon. `hh TT` – Inkluderar endast timme och AM/PM. `hh "Hrs" mm "Mins"` – visar tiden med anpassad text, t.ex. Hrs.

Exempel: diagramuttryck

Exempel	Resultat
`Time#('09:00:00')`	Returnerar `09:00:00` när inställningen för `TimeFormat` är `'hh:mm:ss'`.
`Time#( '09:00:00','hh:mm:ss' )`	Returnerar `09:00:00` när medföljande format är `'hh:mm:ss'`.
`Time#('03:25 PM', 'hh:mm TT')`	Returnerar `03:25 PM` när medföljande format är `'hh:mm TT'`.

Exempel – grunderna i Tid#

Laddningsskript

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i en datatabell:
  - `InputTime`
- Variabeln `TimeFormat`, som använder formatet `h:mm:ss TT`.
- En föregående laddning som använder funktionen `Time#` för att konvertera textvärdet på `InputTime` till ett tids/numeriskt fält som heter `ParsedTime` som systemet kan använda.

Laddningsskript

```
SET TimeFormat='h:mm:ss TT';
Example:
Load
InputTime,
Time#(InputTime, 'hh-mm-ss') as ParsedTime
Inline [
InputTime
12-30-45
05-15-00
22-10-55
08-20-35
19-45-15
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `InputTime`
- `ParsedTime`

Skapa följande mått:

- `=Time(ParsedTime)`, för att visa tiden med hjälp av det format som anges i laddningsskriptet som standardinställning.

Resultattabell

InputTime	ParsedTime	Time(ParsedTime)
05-15-00	05-15-00	5:15:00 AM
08-20-35	08-20-35	8:20:35 AM
12-30-45	12-30-45	12:30:45 PM
19-45-15	19-45-15	7:45:15 PM
22-10-55	22-10-55	10:10:55 PM

Värdena i fältet `InputTime` visas som text och är vänsterjusterade. När värdena i `InputTime` har analyserats genom `Time#`-funktionen skapas det resulterande `ParsedTime`-fältet. Värdena i det nya fältet är högerjusterade eftersom de nu behandlas som ett giltigt tidsvärde. Den tredje kolumnen använder funktionen `Time` för att bedöma `ParsedTime`-värdena och formatera dem enligt `TimeFormat` i laddningsskriptet.

Exempel – Time#-scenario

Laddningsskript

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i en datatabell:
  - `NoDelimiterTime`
- Variabeln `TimeFormat`, som använder formatet `hh.mm.ss`.
- En föregående laddning som använder funktionen `Time#` för att konvertera textvärdet på `NoDelimiterTime` till ett tids/numeriskt fält som heter `ParsedNoDelimiterTime`.

Laddningsskript

```
SET TimeFormat='hh.mm.ss';
Example:
Load
NoDelimiterTime,
Time#(NoDelimiterTime, 'hhmmss') as ParsedNoDelimiterTime
Inline [
NoDelimiterTime
143015
091045
235959
060530
184010
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `NoDelimiterTime`

Skapa följande mått:

- `=Time(ParsedNoDelimiterTime,'hh:mm:ss')`, för att visa tiden med hjälp av det format som anges i laddningsskriptet som standardinställning.

Resultattabell

NoDelimiterTime	Time(ParsedNoDelimiterTime,'hh:mm:ss')
060530	06:05:30
091045	09:10:45
143015	14:30:15
184010	18:40:10
235959	23:59:59

Resultaten visar hur du kan använda funktionen `Time#` för att analysera och konvertera värden från text till ett tids-/numeriskt värde.

Timestamp#

Timestamp#() utvärderar ett uttryck som ett datum- och tidsvärde, i det tidsmarkörformat som har ställts in i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns.

Syntax:

```
timestamp#(text[, format])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
text	Textsträngen som ska utvärderas.
format	Sträng som beskriver det format som ska användas för att tolka textsträngen. Om den utelämnas används det korta datumformat, tidsformat och decimalavgränsare som har ställts in för operativsystemet. ISO 8601 stöds för tidsmarkörer. Funktionen `Timestamp#` har stöd för många formatalternativ för olika tidsstrukturer, t.ex: `DD/MM/YYYY`: Fokuserar bara på datumet (t.ex.: `15/11/2024`). `YYYY-MM-DD hh:mm:ss`: Visar hela tidsmarkören (t.ex.: `2024-11-15 13:30:45`). `MMM DD, YYYY`: Visar en kortversion av månaden och hela året (t.ex.: `Nov 15, 2024`). `hh:mm:ss TT`: Visar endast tiden i 12-timmarsformat (t.ex.: `01:30:45 PM`) `HH:mm:ss`: Visar tiden i 24-timmarsformat (t.ex.: `13:30:45`) `wwww, MMM DD YYYY`: Visar hela veckodagen och datumet (t.ex.: `Friday, Nov 15 2024`) `YYYY-MM-DD hh:mm:ss.fff`: Inkluderar millisekunder (t.ex.: `2024-11-15 13:30:45.123`).

Exempel: diagramuttryck

Exempel	Resultat
`Timestamp#('12-03-2023 10:25:30', 'DD-MM-YYYY hh:mm:ss')`	Returnerar `12-03-2023 10:25:30`
`Timestamp#('2023/03/15 13:45', 'YYYY/MM/DD hh:mm')`	Returnerar `2023/03/15 13:45`

Exempel – grunderna i Timestamp#

Laddningsskript

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i en datatabell:
  - `InputString`
- En föregående laddning som använder funktionen `Timestamp#` för att behandla uttrycket som ett datum- och tidsvärde. När den körs skapas ett extra fält som heter `TimeStamp_InputString`.

Laddningsskript

```
SET DateFormat='M/D/YYYY'
Example:
Load *,
Timestamp(Timestamp#(InputString)) as TimeStamp_InputString;
LOAD * INLINE [
InputString
2015-09-15T12:13:14
1952-10-16T13:14:00+0200
1109-03-01T14:15
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `InputString`
- `TimeStamp_InputString`

Resultattabell

Indatasträng	TimeStamp_InputString
1109-03-01T14:15	3/1/1109 2:15:00 PM
1952-10-16T13:14:00+0200	10/16/1952 11:14:00 AM
2015-09-15T12:13:14	9/15/2015 12:13:14 PM

I det här exemplet kan du se hur strängvärden kan utvärderas som en tidsmarkör med hjälp av funktionen `Timestamp#`.

Exempel – Timestamp#-scenario

Laddningsskript

Översikt

Data från en händelselogg innehåller ett format för tidsmarkör som du vill använda.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Event`
  - `EventDateTime`
- En tidigare laddning som använder funktionen `Timestamp#` för att konvertera värdena i fältet `EventDateTime` till ett datum- och tidsformat som systemet kan förstå. När den körs skapas ett extra fält som heter `EventTimestamp`.

Laddningsskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';
Example:
LOAD
Event,
EventDateTime,
Timestamp#(EventDateTime, 'DD-MM-YYYY hh:mm:ss') AS EventTimestamp
INLINE [
Event, EventDateTime
"Login", "12-03-2023 10:25:30"
"Logout", "12-03-2023 12:45:10"
"Login", "13-03-2023 09:10:15"
"Logout", "13-03-2023 17:30:20"
"Login", "14-03-2023 08:00:00"
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Event`
- `EventDateTime`
- `EventTimestamp`

Skapa följande mått:

- `=Timestamp(EventTimestamp)`, för att visa tidsmarkören formaterad med hjälp av det format som anges i laddningsskriptet som standardinställning.

Resultattabell

Händelse	EventDateTime	EventTimestamp	Timestamp(EventTimestamp)
Logga in	12-03-2023 10:25:30	2023-03-12 10:25:30	3/12/2023 10:25:30 AM
Logga in	13-03-2023 09:10:15	2023-03-13 09:10:15	3/13/2023 9:10:15 AM
Logga in	14-03-2023 08:00:00	2023-03-14 08:00:00	2023-03-14 08:00:00
Logga ut	12-03-2023 12:45:10	12-03-2023 12:45:10	3/12/2023 12:45:10 PM
Logga ut	13-03-2023 17:30:20	13-03-2023 17:30:20	3/13/2023 5:30:20 PM

I det här exemplet kan du se hur värden kan konverteras med hjälp av funktionen `Timestamp#`.

8.16 Postöverskridande funktioner

Postöverskridande funktioner används i följande fall:

- I dataladdningsskriptet när ett värde från tidigare laddade dataposter behövs för utvärdering av den aktuella posten.
- I ett diagramuttryck, om ett annat värde från datauppsättningen till en visualisering behövs.

Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när en postöverskridande diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder en postöverskridande diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den postöverskridande funktionen. Den här begränsningen gäller inte funktionen ekvivalent skript, om en sådan funktion finns.

Självrefererande uttrycksdefinitioner kan endast göras på ett pålitligt sätt i tabeller med färre än 100 rader, men detta kan variera beroende på maskinvaran som Qlik-motorn körs på.

Radfunktioner

Dessa funktioner kan endast användas i diagramuttryck.

Above

Above() utvärderar ett uttryck på en rad ovanför den aktuella raden inom ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden direkt ovanför. För diagram som inte är tabeller utvärderar **Above()** för raden ovanför den aktuella raden i diagrammets raka tabellmotsvarighet.

```
Above - diagramfunktion([TOTAL [<fld{,fld}>]] expr [ , offset [,count]])
```

Below

Below() utvärderar ett uttryck på en rad under den aktuella raden inom ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden direkt nedanför. För diagram som inte är tabeller utvärderar **Below()** för raden nedanför den aktuella kolumnen i diagrammets raka tabellmotsvarighet.

```
Below - diagramfunktion([TOTAL[<fld{,fld}>]] expression [ , offset [,count
 ]])
```

Bottom

Bottom() utvärderar ett uttryck på den sista (nedersta) raden i ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden nederst. För diagram som inte är tabeller görs utvärderingen på den sista raden i den aktuella kolumnen i diagrammets raka tabellmotsvarighet.

```
Bottom - diagramfunktion([TOTAL[<fld{,fld}>]] expr [ , offset [,count ]])
```

Top

Top() utvärderar ett uttryck på den första (översta) raden i ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden högst upp. För diagram som inte är tabeller görs **Top()**-utvärderingen på den första raden i den aktuella kolumnen i diagrammets raka tabellmotsvarighet.

```
Top - diagramfunktion([TOTAL [<fld{,fld}>]] expr [ , offset [,count ]])
```

NoOfRows

NoOfRows() returnerar antalet rader i det aktuella kolumnsegmentet i en tabell. För bitmappsdiagram returnerar **NoOfRows()** antalet rader i diagrammets raka tabellmotsvarighet.

```
NoOfRows ([TOTAL])
```

Kolumnfunktioner

Dessa funktioner kan endast användas i diagramuttryck.

Column

Column() returnerar det värde som återfinns i kolumnen som motsvarar **ColumnNo** i en rak tabell om dimensionerna ignoreras. Exempelvis returnerar **Column(2)** värdet för den andra måttkolumnen.

```
Column - diagramfunktion(ColumnNo)
```

Dimensionality

Dimensionality() returnerar antalet dimensioner för den aktuella raden. Vad gäller pivottabeller returnerar funktionen det totala antalet dimensionskolumner som har icke-aggregerat innehåll, d.v.s. inte innehåller delsummor eller dolda aggregeringar.

```
Dimensionality - diagramfunktion ( )
```

Secondarydimensionality

SecondaryDimensionality() returnerar antalet dimensionspivottabellrader som har icke-aggregerat innehåll, d.v.s. inte innehåller delsummor eller dolda aggregeringar. Denna funktion motsvarar funktionen **dimensionality()** för horisontella pivottabelldimensioner.

```
SecondaryDimensionality - diagramfunktion ( )
```

Fältfunktioner

FieldIndex

FieldIndex() returnerar den placering som fältvärdet **value** har i fältet **field_name** (i laddningsordning).

```
FieldIndex (field_name , value)
```

FieldValue

FieldValue() returnerar det värde som återfinns i position **elem_no** i fältet **field_name** (i inläsningsordning).

```
FieldValue(field_name , elem_no)
```

FieldValueCount

FieldValueCount() är en **heltal**sfunktion som returnerar antalet distinkta värden i ett fält.

```
FieldValueCount (field_name)
```

Pivottabellfunktioner

Dessa funktioner kan endast användas i diagramuttryck.

After

After() returnerar värdet för ett uttryck utvärderat med en pivottabells dimensionsvärden som de visas i kolumnen efter den aktuella kolumnen inom ett radsegment i pivottabellen.

```
After - diagramfunktion([TOTAL] expression [ , offset [,n]])
```

Before

Before()returnerar värdet av ett uttryck utvärderat men en pivottabells dimensionsvärden som de visas i kolumnen framför den aktuella kolumnen inom ett radsegment i pivottabellen.

```
Before - diagramfunktion([TOTAL] expression [ , offset [,n]])
```

First

First() returnerar värdet för ett uttryck utvärderat med en pivottabells dimensionsvärden som de visas i den första kolumnen i det aktuella radsegmentet i pivottabellen. Funktionen returnerar NULL i alla diagramtyper utom pivottabeller.

```
First - diagramfunktion([TOTAL] expression [ , offset [,n]])
```

Last

Last() returnerar värdet för ett uttryck utvärderat med en pivottabells dimensionsvärden som de visas i den sista kolumnen i det aktuella radsegmentet i pivottabellen. Funktionen returnerar NULL i alla diagramtyper utom pivottabeller.

```
Last - diagramfunktion([TOTAL] expression [ , offset [,n]])
```

ColumnNo

ColumnNo() returnerar numret på den aktuella kolumnen inom det aktuella radsegmentet i en pivottabell. Första kolumnen är nummer 1.

```
ColumnNo - diagramfunktion([TOTAL])
```

NoOfColumns

NoOfColumns() returnerar antalet kolumner i det aktuella radsegmentet i en pivottabell.

```
NoOfColumns - diagramfunktion([TOTAL])
```

Postöverskridande funktioner i dataladdningsskriptet

Exists

Exists() avgör om ett specifikt fältvärde redan har laddats in i fältet i dataladdningsskriptet. Funktionen returnerar TRUE eller FALSE, så att det kan användas i **where**-satsen för en **LOAD**-sats eller en **IF**-sats.

```
Exists (field_name [, expr])
```

LookUp

LookUp() letar in i en tabell som redan är laddad och returnerar värdet av **field_name** vilket motsvarande den första förekomsten av värdet **match_field_value** i fältet **match_field_name**. Tabellen kan vara den aktuella tabell eller en annan tabell som har laddats.

```
LookUp (field_name, match_field_name, match_field_value [, table_name])
```

Peek

Peek() returnerar värdet för ett fält i en tabell för en rad som redan har laddats. Radnumret kan anges, liksom tabellen. Om inget radnummer anges används posten som laddades senast.

```
Peek (field_name[, row_no[, table_name ] ])
```

Previous

Previous() returnerar värdet av uttrycket **expr** genom att använda data från en tidigare indatapost som inte uteslutits till följd av en **where**-sats. För den första posten i en intern tabell kommer funktionen att returnera NULL.

```
Previous (expr)
```

Se även:

p *Intervallfunktioner (page 1576)*

Above - diagramfunktion

Above() utvärderar ett uttryck på en rad ovanför den aktuella raden inom ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden direkt ovanför. För diagram som inte är tabeller utvärderar **Above()** för raden ovanför den aktuella raden i diagrammets raka tabellmotsvarighet.

Syntax:

```
Above([TOTAL] expr [ , offset [,count]])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Om en offset n, större än 0 anges, flyttas utvärderingen av uttrycket n antal rader ovanför den aktuella raden. Om startpunkt anges till 0 utvärderas uttrycket på den aktuella raden. Anges ett negativt värde för startpunkten gör det att Above-funktionen fungerar likadant som Below-funktionen med motsvarande positivt värde för startpunkten.
count	Om ett tredje uttryck, **count**, anges till större än 1, returnerar funktionen ett intervall av **count**-värden, ett för varje **count**-tabellrad. Raderna räknas uppåt från den ursprungliga cellen. I denna form kan funktionen användas som argument i någon av de speciella intervallfunktionerna. *Intervallfunktioner (page 1576)*
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

På kolumnsegmentets första rad returneras värdet NULL, eftersom det inte finns någon rad ovanför denna.

Ett kolumnsegment definieras som en konsekutiv underuppsättning med celler som har samma värden för dimensionerna i den aktuella sorteringsordningen. Postöverskridande diagramfunktioner beräknas i kolumnsegmentet exklusive dimensionen längst till höger i motsvarande raka tabelldiagram. Om det enbart finns en dimension i diagrammet, eller om kvalificeraren TOTAL anges, utvärderas uttrycket över en hel tabell.

Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner, utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

Begränsningar:

- Rekursiva anrop returnerar NULL.
- Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller

tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen.

Exempel: diagramuttryck

Exempel	Resultat
`Above( Sum(Sales) )`	Returnerar värdet på `Sales` för raden ovanför den aktuella raden.

Exempel – grunderna i Above

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Year`
  - `Sales`

Laddningsskript

```
Example:
LOAD * INLINE [
Year, Sales
2019, 1000
2020, 1500
2021, 1200
2022, 1800
2023, 1700
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Year`

Skapa följande mått:

- `=Sum(Sales)`, för att få fram den årliga försäljningen.
- `=Above(Sum(Sales))`, för att få fram försäljningsvärdet för föregående år.
- `=Sum(Sales) - Above(Sum(Sales))`, för att beräkna variansen mellan värdet på `Sales` för innevarande år och värdet för föregående år.
- `=RangeSum(Above(Sales, 1, 2))`, för att visa en rullande tvåårig sammanställning av de senaste två årens försäljning.
- `=(Sales - Above(Sales)) / Above(Sales)`, för att ange förändringen i `Sales` som en kvot/procentandel. Om du vill visa värdet i procent väljer du **Nummer > Enkel formatering > 12,34%** under **Nummerformatering**.

Resultattabell

Year	Sum (Sales)	Above (Sum (Sales))	Sum(Sales) - Above(Sum (Sales))	RangeSum (Above(Sales, 1, 2)	(Sales - Above (Sales)) / Above (Sales)
Totals	**7200**	**-**	**-**	**0**	**-**
2019	1000	-	-	0	-
2020	1500	1000	500	1000	50.00%
2021	1200	1500	-300	2500	-20.00%
2022	1800	1200	600	2700	50.00%
2023	1700	1800	-100	3000	-5.56%

När du tittar på resultaten kan du jämföra `Sales`-värdena under den aktuella perioden med föregående period.

- Måttet `Sum(Sales)- Above(Sum(Sales))` ger förändringen i `Sales`-värdet vid en jämförelse av aktuell period med den tidigare.
- Måttet `RangeSum(Above(Sales, 1, 2))` ger summan av de två föregående `Sales` -värdena.
- Måtte `(Sales - Above(Sales)) / Above(Sales)` ger förändringen i `Sales` jämfört med den tidigare perioden i procent.

Funktionen kan även användas i andra diagram, inte bara tabeller - exempelvis i stapeldiagram.

Gör följande:

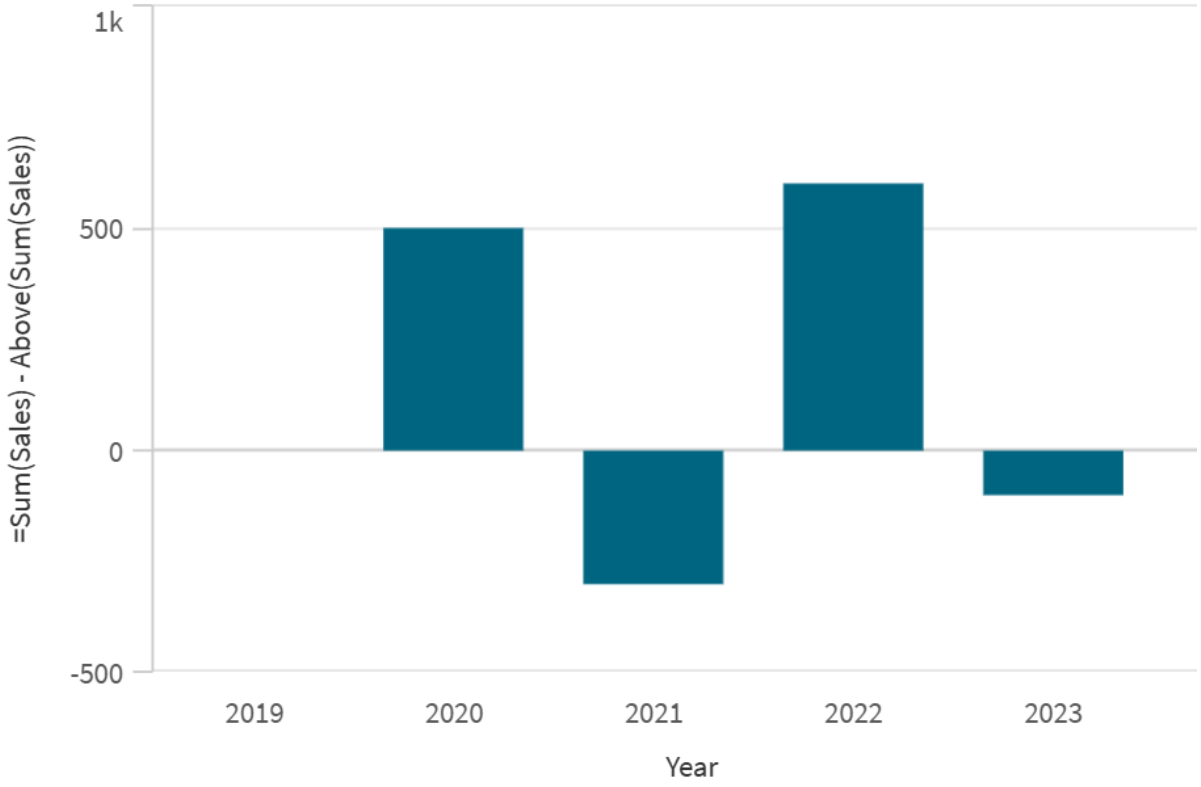
Skapa ett stapeldiagram och lägg sedan till det här fältet som en dimension:

- `Year`

Skapa följande mått:

- `=Sum(Sales) - Above(Sum(Sales))`, för att ange variationen från år till år.

Stapeldiagram som visar resultaten för funktionen Above



Om du använder funktionen `Above` i ett stapeldiagram får du en visuell jämförelse av variansen mellan försäljningen under den aktuella perioden och försäljningen under föregående period.

Exempel – Above-scenario

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `SalesYear`
  - `SalesAmount`

Laddningsskript

```
Example:
LOAD * INLINE [
SalesYear, SalesAmount
```

```
2019, 500
2020, 800
2021, 400
2022, 700
2023, 600
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `SalesYear`

Skapa följande mått:

- `=Sum(SalesAmount)`, för att sammanställa försäljningen.
- `=RangeSum(Above(SalesAmount, 0, RowNo()))`, för att beräkna en löpande summa av `SalesAmount`-värdena.

Resultattabell

SalesYear	Sum(SalesAmount)	RangeSum(Above(SalesAmount, 0, RowNo()))
Totals	**3000**	**0**
2019	500	500
2020	800	1300
2021	400	1700
2022	700	2400
2023	600	3000

Resultaten visar hur du kan använda en kombination av funktionerna `RangeSum` och `Above` för att skapa en kumulativ löpande total.

Funktionen kan även användas i andra diagram, inte bara tabeller - exempelvis i stapeldiagram.

Gör följande:

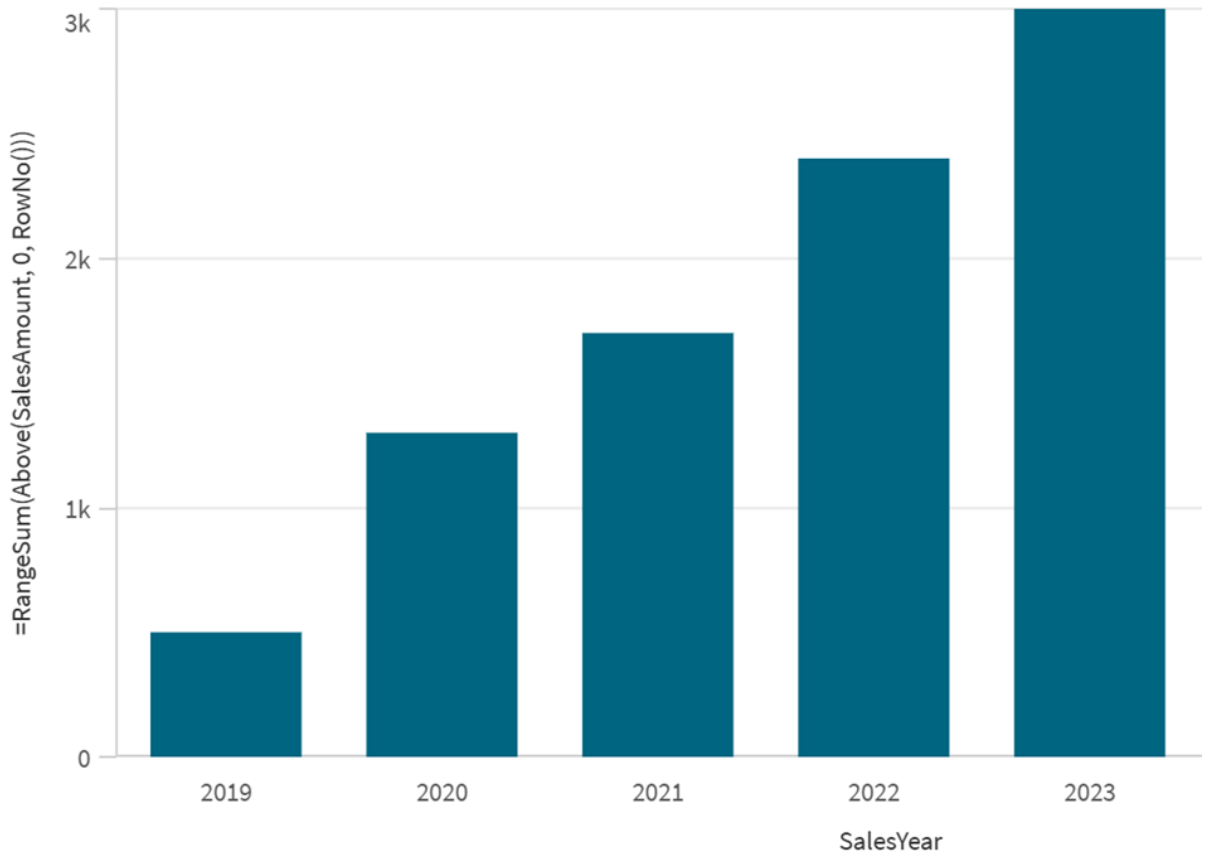
Skapa ett stapeldiagram och lägg sedan till det här fältet som en dimension:

- `SalesYear`

Skapa följande mått:

- `=RangeSum(Above(SalesAmount, 0, RowNo()))`, för att ange det ackumulerade försäljningsvärdet.

Stapeldiagram som visar det ackumulerade försäljningsvärdet per år



Exempel – diagramuttryck som använder Above

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Två datatabeller som heter `Monthnames` och `Sales2013`.

Laddningsskript

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
```

```
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];
Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.
- `=Above(Sum(Sales))`, för att visa summan av försäljningen under föregående rad.
- `=Sum(Sales)+Above(Sum(Sales))`, för att summera summan av försäljningsvärdet med summan av försäljningen för kundraden ovanför den aktuella raden.
- `=Sum(Sales)+Above(Sum(Sales), 3)`, för att summera `Sum(Sales)`-värdet för aktuell `Customer` till värdet för `Customer` som är tre rader högre upp.
- `=IF(Sum(Sales)>Above(Sum(Sales)), 'Higher')`, att lägga till etiketten "`Higher`" till kunder vars `Sum(Sales)`-resultat är större än `Sum(Sales)` för kunden i raden ovan.

Resultattabell

Customer	Sum (Sales)	Above (Sum (Sales))	Sum (Sales)+Above (Sum(Sales))	Sum (Sales)+Above (Sum(Sales), 3)	IF(Sum (Sales)>Above (Sum(Sales)), 'Higher')
Totals	**2566**	**-**	**-**	**-**	**-**
Astrida	587	-	-	-	-
Betacab	539	587	1126	-	-
Canutility	683	539	1222	-	Higher
Divadip	757	683	1440	1344	Higher

Måttet `Above(Sum(Sales))` returnerar NULL för kunden `Astrida` eftersom det inte finns någon rad ovanför. Resultatet för kunden `Betacab` visar värdet på `Sum(Sales)` för `Astrida`. Resultatet för `Canutility` visar värdet för `Sum(Sales)` för `Betacab`, och så vidare.

För måttet `Sum(Sales)+Above(Sum(Sales))` ger raden för `Betacab` resultatet av att lägga till värdena för `Sum(Sales)` för raderna `Betacab` + `Astrida` (539+587). Resultatet för raden `Canutility` visar resultatet då `Sum(Sales)`-värden har lagts till för `Canutility` + `Betacab` (683+539).

Måttet `Sum(Sales)+Above(Sum(Sales), 3)` har argumentet `offset` satt till `3`. Detta tar värdet i den rad som är 3 rader ovanför den aktuella raden. Det lägger till `Sum(Sales)`-värdet för aktuell `Customer` på värdet för `Customer` tre rader högre upp. Till exempel läggs `Sum(Sales)` för `Divadip` till `Sum(Sales)` för `Astrida`, den kund som finns tre rader ovanför `Divadip`. Värdena som returneras för de första tre `Customer`-raderna är null eftersom det saknas data.

Måttet `IF(Sum(Sales)>Above(Sum(Sales)), 'Higher')`, returnerar värdet `Higher` för kunderna `Canutility` och `Divadip` eftersom deras försäljningsresultat båda är högre än summan av försäljningen för kunden i raden ovan, `Betacab` respektive `Canutility`.

Funktionen kan även användas i andra diagram, inte bara tabeller - exempelvis i stapeldiagram.

För andra diagramtyper, ska du omvandla diagrammet till den raka tabellmotsvarigheten så att du enkelt kan tolka vilken rad funktionen är relaterad till.

Exempel – Above-scenario baserat på sorteringsordning

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Tre datatabeller som heter `Monthnames`, `Sales2013` och `Product`.

Laddningsskript

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
```

```
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');

Product:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Customer`
- `Product`
- `Month`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.
- `=Above(Sum(Sales))`, för att visa summan av försäljningen under föregående rad.

Resultattabell

Customer	Product	Month	Sum(Sales)	Above(Sum(Sales))
Totals	**-**	**-**	**2566**	**-**
Astrida	AA	Jan	46	-
Astrida	AA	Feb	60	46

Customer	Product	Month	Sum(Sales)	Above(Sum(Sales))
Totals	-	-	**2566**	-
Astrida	AA	Mar	70	60
Astrida	AA	Apr	13	70
Astrida	AA	May	78	13
Astrida	AA	Jun	20	78
Astrida	AA	Jul	45	20
Astrida	AA	Aug	65	45
Astrida	AA	Sep	78	65
Astrida	AA	Oct	12	78
Astrida	AA	Nov	78	12
Astrida	AA	Dec	22	78
Astrida	BB	Jan	46	-
Astrida	BB	Feb	60	46

Resultatet för uttryck som innehåller funktionerna `Above`, `Below`, `Top` och `Bottom` i diagram med flera dimensioner beror på i vilken ordning kolumndimensionerna sorteras av Qlik Sense. Qlik Sense evaluerar funktionerna baserat på de kolumnsegment som är resultatet från den dimension som kommer sist i sorteringsordningen. Sorteringsordningen för kolumner styrs från egenskapspanelen under **Sortering**. Den motsvarar inte nödvändigtvis den ordning i vilken kolumnerna visas i en tabell.

I följande resultattabell är den sista dimensionen i sorteringsordningen `Month`, så `Above`-funktionen utvärderar utifrån månader. Det finns en serie resultat för varje `Product`-värde för varje månad (`Jan` till `Dec`) - ett kolumnsegment. Detta följs av en serie för nästa kolumnsegment för varje `Month` för nästa `Product`. Det finns ett kolumnsegment för varje `Customer`-värde för varje `Product`.

Exempel – Above-scenario med en annan sorteringsordning

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Tre datatabeller som heter `Monthnames`, `Sales2013` och `Product`.

Laddningsskript

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
```

```
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');

Product:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Customer`
- `Product`
- `Month`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.
- `=Above(Sum(Sales))`, för att visa summan av försäljningen under föregående rad.

Ändra tabellens sorteringsordning i egenskapspanelen. Välj **Sortering** och dra sedan dimensionen **Produkt** till den tredje positionen i sorteringsordningen.

Resultattabell

Customer	Product	Month	Sum(Sales)	Above(Sum(Sales))
Totals	-	-	**2566**	-
Astrida	AA	Jan	46	-
Astrida	BB	Jan	46	46
Astrida	AA	Feb	60	-
Astrida	BB	Feb	60	60
Astrida	AA	Mar	70	-
Astrida	BB	Mar	70	70
Astrida	AA	Apr	13	-
Astrida	BB	Apr	13	13
Astrida	AA	May	78	-
Astrida	BB	May	78	78
Astrida	AA	Jun	20	-
Astrida	BB	Jun	20	20
Astrida	AA	Jul	45	-
Astrida	BB	Jul	45	45

I resultattabellen är den senast sorterade dimensionen `Product`. Funktionen `Above` evalueras för varje `Product`, och eftersom det bara finns två produkter, `AA` och `BB`, innehåller varje serie bara ett resultat som inte är null. I rad `BB` för månaden `Jan` är värdet för `Above(Sum(Sales))` `46`. För rad `AA` är värdet null. Värdet i varje `AA`-rad, oavsett månad, är alltid null eftersom det inte finns något värde för `Product` över `AA`. Den andra serien värderas efter `AA` och `BB` för månaden `Feb`, för `Customer`-värdet `Astrida`. När alla månader har utvärderats för `Astrida` upprepas sekvensen för den andra kunden `Betacab` och så vidare.

Resultaten i tabellen visar endast en partiell lista över data. För att se hela resultatet kan du prova exemplet i en app.

Exempel – Användning av funktionen Above med RangeAvg

Diagramuttryck

Översikt

Funktionen `Above` kan användas som indata för intervallfunktionerna. I detta exempel används funktionen i en `RangeAvg`.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Tre datatabeller som heter `Monthnames`, `Sales2013` och `Product`.

Laddningsskript

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');

Product:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
```

```
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=RangeAvg (Above(Sum(Sales),1,3))`

Resultattabell*

Customer	RangeAvg (Above(Sum(Sales),1,3))
Totals	**-**
Astrida	-
Betacab	587
Canutility	563
Divadip	603

I argumenten för funktionen Above, är `offset` inställt på `1` och `count` är inställt på `3`. Funktionen hittar resultatet för uttrycket `Sum(Sales)` på de tre raderna som är precis ovanför aktuell rad i kolumnsegmentet (där det finns en rad). De här tre värdena används som indata för funktionen `RangeAvg` som räknar ut medelvärdet för ett angivet talintervall.

Se även:

p *Below - diagramfunktion (page 1473)*
p *Bottom - diagramfunktion (page 1483)*
p *Top - diagramfunktion (page 1521)*
p *RangeAvg (page 1579)*

Below - diagramfunktion

Below() utvärderar ett uttryck på en rad under den aktuella raden inom ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden direkt nedanför. För diagram som inte är tabeller utvärderar **Below()** för raden nedanför den aktuella kolumnen i diagrammets raka tabellmotsvarighet.

Syntax:

```
Below([TOTAL] expr [ , offset [,count ]])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Om en **offset** n större än 1 anges, flyttas utvärderingen av uttrycket n antal rader nedanför den aktuella raden. Om startpunkt anges till 0 utvärderas uttrycket på den aktuella raden. Anges ett negativt värde för startpunkten gör det att **Below**-funktionen fungerar likadant som **Above**-funktionen med motsvarande positivt värde för startpunkten.
count	Om en tredje parameter, **count**, anges till större än 1, returnerar funktionen ett intervall av **count**-värden: ett för varje **count**-tabellrad. Raderna räknas nedåt från den ursprungliga cellen. I denna form kan funktionen användas som argument i någon av de speciella intervallfunktionerna. *Intervallfunktioner (page 1576)*
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

På kolumnsegmentets sista rad returneras värdet NULL, eftersom det inte finns någon rad nedanför denna.

Ett kolumnsegment definieras som en konsekutiv underuppsättning med celler som har samma värden för dimensionerna i den aktuella sorteringsordningen. Postöverskridande diagramfunktioner beräknas i kolumnsegmentet exklusive dimensionen längst till höger i motsvarande raka tabelldiagram. Om det enbart finns en dimension i diagrammet, eller om kvalificeraren TOTAL anges, utvärderas uttrycket över en hel tabell.

Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner, utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

Begränsningar:

- Rekursiva anrop returnerar NULL.
- Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen.

Exempel: diagramuttryck

Exempel	Resultat
`Below( Sum(Sales) )`	Returnerar värdet på `Sales` för raden under den aktuella raden.

Exempel – grunderna i Below

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Year`
  - `Sales`

Laddningsskript

```
Example:
LOAD * INLINE [
Year, Sales
2019, 1000
2020, 1500
2021, 1200
2022, 1800
2023, 1700
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Year`

Skapa följande mått:

- `=Sum(Sales)`, för att få fram den årliga försäljningen.
- `=Below(Sum(Sales))`, för att få fram försäljningsvärdet för nästa år.

- `=Sum(Sales) - Below(Sum(Sales))`, för att beräkna variansen mellan värdet på `Sales` för innevarande år och värdet för nästa år.
- `=RangeSum(Below(Sales, 1, 2))`, för att visa en rullande tvåårig sammanställning av de kommande två årens försäljning.
- `=(Sales - Below(Sales)) / Below(Sales)`, för att ange förändringen i `Sales` som en kvot/procentandel. Om du vill visa värdet i procent väljer du **Nummer > Enkel formatering > 12,34%** under **Nummerformatering**.

Resultattabell

Year	Sum (Sales)	Below (Sum (Sales))	Sum(Sales) - Below(Sum (Sales))	RangeSum (Below(Sales, 1, 2)	(Sales - Below (Sales)) / Below (Sales)
Totals	**7200**	**-**	**-**	**0**	**-**
2019	1000	1500	-500	2700	-33.33%
2020	1500	1200	300	3000	25.00%
2021	1200	1800	-600	3500	-33.00%
2022	1800	1700	100	1700	5.88%
2023	1700	-	-	0	-

När du tittar på resultaten kan du jämföra `Sales`-värdena under den aktuella perioden med nästa period.

- Måttet `Sum(Sales) - Below(Sum(Sales))` ger förändringen i `Sales`-värdet vid en jämförelse av aktuell period med nästa.
- Måttet `RangeSum(Below(Sales, 1, 2))` ger summan av de två kommande `Sales` -värdena.
- Måtte `(Sales - Below(Sales)) / Below(Sales)` ger förändringen i `Sales` jämfört med nästa perioden i procent.

Funktionen kan även användas i andra diagram, inte bara tabeller - exempelvis i stapeldiagram.

Gör följande:

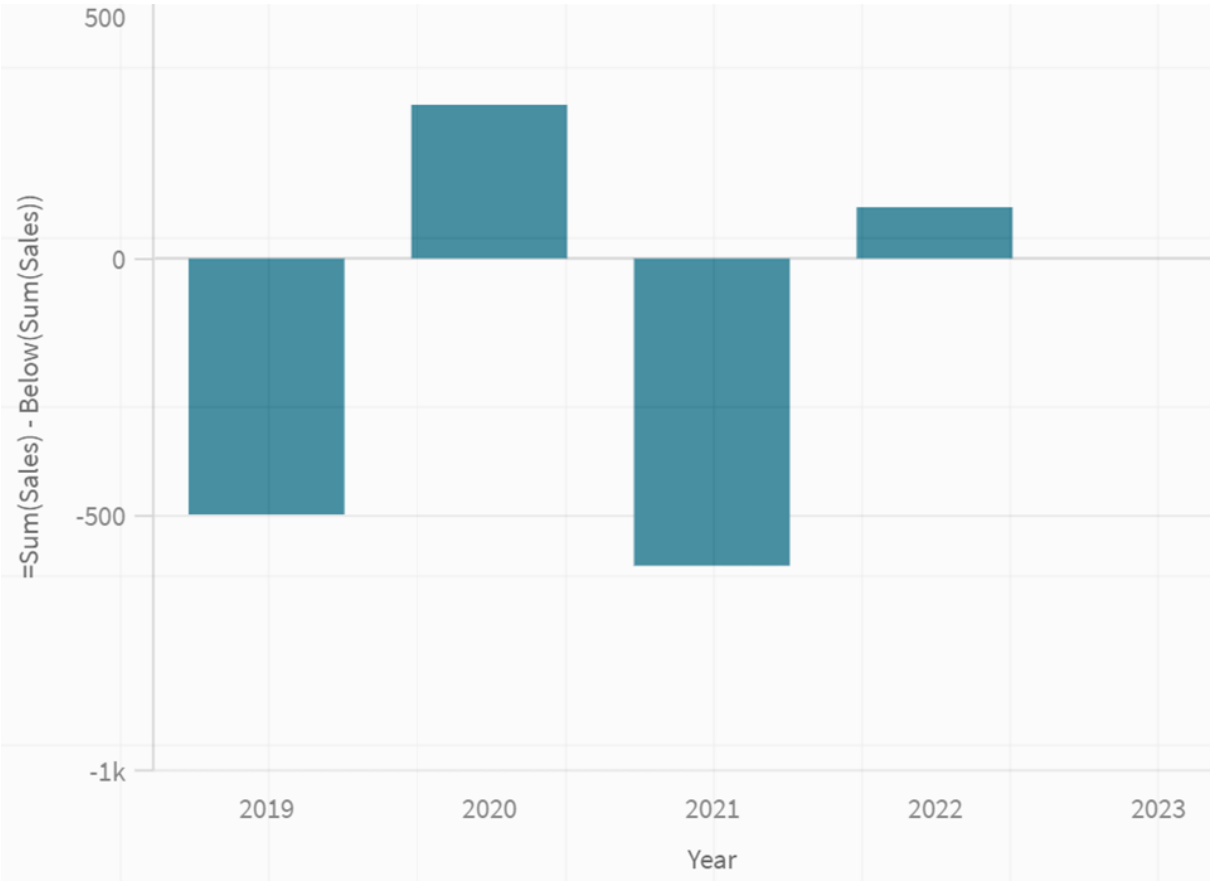
Skapa ett stapeldiagram och lägg sedan till det här fältet som en dimension:

- `Year`

Skapa följande mått:

- `=Sum(Sales) - Below(Sum(Sales))`, för att ange variationen från år till år.

Stapeldiagram som visar resultaten för funktionen Below



Om du använder funktionen `Below` i ett stapeldiagram får du en visuell jämförelse av variansen mellan försäljningen under den aktuella perioden och försäljningen under nästa period.

Exempel – Below-scenario

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `SalesYear`
  - `SalesAmount`

Laddningsskript

```
Example:
LOAD * INLINE [
```

```
SalesYear, SalesAmount
2019, 500
2020, 800
2021, 400
2022, 700
2023, 600
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `SalesYear`

Skapa följande mått:

- `=Sum(SalesAmount)`, för att sammanställa försäljningen.
- `=RangeSum(Below(SalesAmount, 0, NoOfRows() - RowNo()))`, för att beräkna en löpande summa av `SalesAmount`-värdena.

Resultattabell

SalesYear	Sum(SalesAmount)	RangeSum(Below(SalesAmount, 0, NoOfRows() - RowNo()))
Totals	**3000**	**0**
2019	500	2400
2020	800	1900
2021	400	1100
2022	700	700
2023	600	0

Resultaten visar hur du kan använda kombinationen av funktionerna `RangeSum` och `Below` för att skapa en kumulativ löpande total.

Funktionen kan även användas i andra diagram, inte bara tabeller - exempelvis i stapeldiagram.

Gör följande:

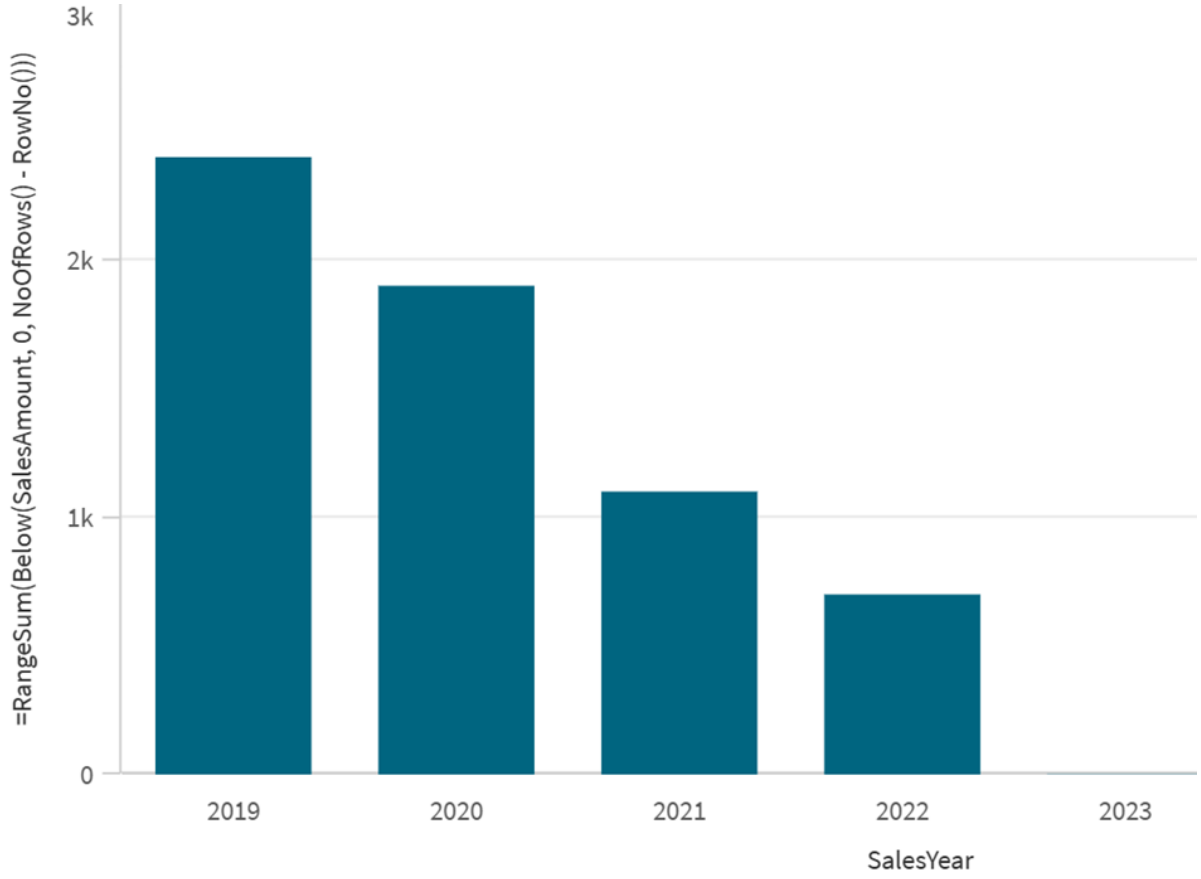
Skapa ett stapeldiagram och lägg sedan till det här fältet som en dimension:

- `SalesYear`

Skapa följande mått:

- `=RangeSum(Below(SalesAmount, 0, NoOfRows() - RowNo()))`, för att ange det ackumulerade löpande försäljningsvärdet.

Stapeldiagram som visar det ackumulerade löpande försäljningsvärdet per år



Genom att använda funktionen `Below` i kombination med `RangeSum` i ett stapeldiagram får man en visuell visning av de ackumulerade löpande försäljningsvärdena.

Exempel – diagramuttryck som använder Below

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Två datatabeller som heter `Monthnames` och `Sales2013`.

Laddningsskript

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
```

```
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.
- `=Below(Sum(Sales))`, för att visa summan av försäljningen under föregående rad.
- `=Sum(Sales)+Below(Sum(Sales))`, för att summera summan av försäljningsvärdet med summan av försäljningen för kundraden nedanför den aktuella raden.
- `=Sum(Sales)+Below(Sum(Sales), 3)`, för att summera `Sum(Sales)`-värdet för aktuell `Customer` till värdet för `Customer` som är tre rader längre ned.
- `=If(Sum(Sales)>Below(Sum(Sales)), 'Higher')`, att lägga till etiketten "`Higher`" till kunder vars `Sum(Sales)`-resultat är större än `Sum(Sales)` för kunden i raden nedanför.

Resultattabell

Customer	Sum (Sales)	Below (Sum (Sales))	Sum (Sales)+Below (Sum(Sales))	Sum (Sales)+Below (Sum(Sales), 3)	If(Sum (Sales)>Below(Sum (Sales)), 'Higher')
Totals	**2566**	**-**	**-**	**-**	**-**
Astrida	587	539	1126	1334	Higher
Betacab	539	683	1222	-	-
Canutility	683	757	1440	-	-
Divadip	757	-	-	-	-

Resultatet för uttryck som innehåller funktionerna `Above`, `Below`, `Top` och `Bottom` i diagram med flera dimensioner beror på i vilken ordning kolumndimensionerna sorteras av Qlik Sense. Qlik Sense evaluerar funktionerna baserat på de kolumnsegment som är resultatet från den dimension som kommer sist i sorteringsordningen. Sorteringsordningen för kolumner styrs från egenskapspanelen under **Sortering**. Den motsvarar inte nödvändigtvis den ordning i vilken kolumnerna visas i en tabell. Ett exempel på hur du använder sorteringsordning finns på *Exempel – Above-scenario baserat på sorteringsordning (page 1467)*.

Måttet `Below(Sum(Sales))` returnerar NULL för kunden `Divadip` eftersom det inte finns någon rad nedanför. Resultatet för raden `Canutility` visar värdet på `Sum(Sales)` för `Divadip`. Resultatet för `Betacab` visar värdet för `Sum(Sales)` för `Canutility`, och så vidare.

För måttet `Sum(Sales)+Below(Sum(Sales))` ger raden för `Astrida` resultatet av att lägga till värdena för `Sum(Sales)` för raderna `Betacab` + `Astrida` (539+587). Resultatet för raden `Betacab` visar resultatet då `Sum(Sales)`-värden har lagts till för `Canutility` + `Betacab` (539+683).

Måttet `Sum(Sales)+Below(Sum(Sales), 3)` har argumentet `offset`, angivet som `3` och har den effekten att värdet hämtas i raden tre rader nedanför den aktuella raden. Det lägger till `Sum(Sales)`-värdet för aktuell `Customer` på värdet från `Customer` tre rader längre ner. Värdena för de lägsta tre `Customer`-raderna är null.

Måttet `IF(Sum(Sales)>Below(Sum(Sales)), 'Higher')`, returnerar värdet `Higher` för kunden `Astrida` eftersom deras försäljningsresultat är högre än summan av försäljningen för kunden i raden nedanför `Betacab`.

Funktionen kan även användas i andra diagram, inte bara tabeller – exempelvis i stapeldiagram.

För andra diagramtyper, ska du omvandla diagrammet till den raka tabellmotsvarigheten så att du enkelt kan tolka vilken rad funktionen är relaterad till.

Exempel – Användning av funktionen Below med RangeAvg

Diagramuttryck

Översikt

Funktionen `Below` kan användas som indata för intervallfunktionerna. I detta exempel används funktionen i en `RangeAvg`.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Två datatabeller som heter `Monthnames` och `Sales2013`.

Laddningsskript

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=RangeAvg (Below(Sum(Sales),1,3))`

Resultattabell

Customer	RangeAvg (Below(Sum(Sales),1,3))
Totals	**-**
Astrida	660
Betacab	720
Canutility	757
Divadip	-

I argumenten för funktionen `Below`, är `offset` inställt på `1` och `count` är inställt på `3`. Funktionen hittar resultatet för uttrycket `Sum(Sales)` på de tre raderna som är precis nedanför aktuell rad i kolumnsegmentet (där det finns en rad). De här tre värdena används som indata för funktionen `RangeAvg` som räknar ut medelvärdet för ett angivet talintervall.

Se även:

p *Above - diagramfunktion (page 1459)*
p *Bottom - diagramfunktion (page 1483)*
p *Top - diagramfunktion (page 1521)*
p *RangeAvg (page 1579)*

Bottom - diagramfunktion

Bottom() utvärderar ett uttryck på den sista (nedersta) raden i ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden nederst. För diagram som inte är tabeller görs utvärderingen på den sista raden i den aktuella kolumnen i diagrammets raka tabellmotsvarighet.

Syntax:

```
Bottom([TOTAL] expr [ , offset [,count ]])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Om en **offset**n större än 1 anges flyttas utvärderingen av uttrycket upp n antal rader ovanför den nedersta raden. Anges ett negativt värde för startpunkten gör det att **Bottom**-funktionen fungerar likadant som **Top**-funktionen med motsvarande positivt värde för startpunkten.
count	Om en tredje parameter, **count**, anges till större än 1, returnerar funktionen inte bara ett värde, utan ett intervall av **count**-värden: ett för var och en av de sista **count**-raderna i det aktuella kolumnsegmentet. I denna form kan funktionen användas som argument i någon av de speciella intervallfunktionerna. *Intervallfunktioner (page 1576)*
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

Ett kolumnsegment definieras som en konsekutiv underuppsättning med celler som har samma värden för dimensionerna i den aktuella sorteringsordningen. Postöverskridande diagramfunktioner beräknas i kolumnsegmentet exklusive dimensionen längst till höger i motsvarande raka tabelldiagram. Om det enbart finns en dimension i diagrammet, eller om kvalificeraren TOTAL anges, utvärderas uttrycket över en hel tabell.

Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner, utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

Begränsningar:

- Rekursiva anrop returnerar NULL.
- Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen.

Exempel: diagramuttryck

Exempel	Resultat
`Bottom( Sum(Sales) )`	Returnerar värdet på `Sales` för den sista (nedre) raden i ett kolumnsegment i en tabell.

Exempel – grunderna i Bottom

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Year`
  - `Sales`

Laddningsskript

```
Example:
LOAD * INLINE [
Year, Sales
2019, 1000
2020, 1500
```

```
2021, 1200
2022, 1800
2023, 1700
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Year`

Skapa följande mått:

- `=Sum(Sales)`, för att få fram den årliga försäljningen.
- `=Bottom(Sum(Sales))`, för att få fram försäljningsvärdet för förra året.
- `=Sum(Sales) - Bottom(Sum(Sales))`, för att beräkna variansen mellan värdet på `Sales` för innevarande år och värdet för förra året.
- `=Sum(Sales)+Bottom(Sum(Sales), 3)`, för att beräkna försäljningsvärdet för innevarande år och värdet för det tredje senaste året.
- `=(Sales - Bottom(Sales)) / Bottom(Sales)`, för att ange förändringen i `Sales` som en kvot/procentandel jämfört med förra året. Om du vill visa värdet i procent väljer du **Nummer > Enkel formatering > 12,34%** under **Nummerformatering**.

Sortera tabellen efter dimensionen `Year` i stigande ordning.

Resultattabell

Year	Sum (Sales)	Bottom (Sum (Sales))	Sum(Sales) - Bottom(Sum (Sales))	Sum (Sales)+Bottom (Sum(Sales), 3)	(Sales - Bottom (Sales)) / Bottom (Sales)
Totals	**7200**	**1700**	**5500**	**8400**	**-**
2019	1000	1700	-700	2200	-41.18%
2020	1500	1700	-200	2700	-11.76%
2021	1200	1700	-500	2400	-29.41%
2022	1800	1700	100	3000	5.88%
2023	1700	1700	0	2900	0.00%

När du tittar på resultaten kan du jämföra `Sales`-värdena under den aktuella perioden med den sista eller senaste perioden.

- Måttet `Sum(Sales) - Bottom(Sum(Sales))` ger förändringen i `Sales`-värdet vid en jämförelse av aktuell period med den senaste.
- Måttet `Sum(Sales)+Bottom(Sum(Sales), 3)` visar summan av `Sales` för den aktuella raden och den tredje senaste perioden.

- Måtte `(Sales - Bottom(Sales)) / Bottom(Sales)` ger förändringen i `Sales` jämfört med den senaste perioden i procent.

Funktionen kan även användas i andra diagram, inte bara tabeller - exempelvis i stapeldiagram.

Gör följande:

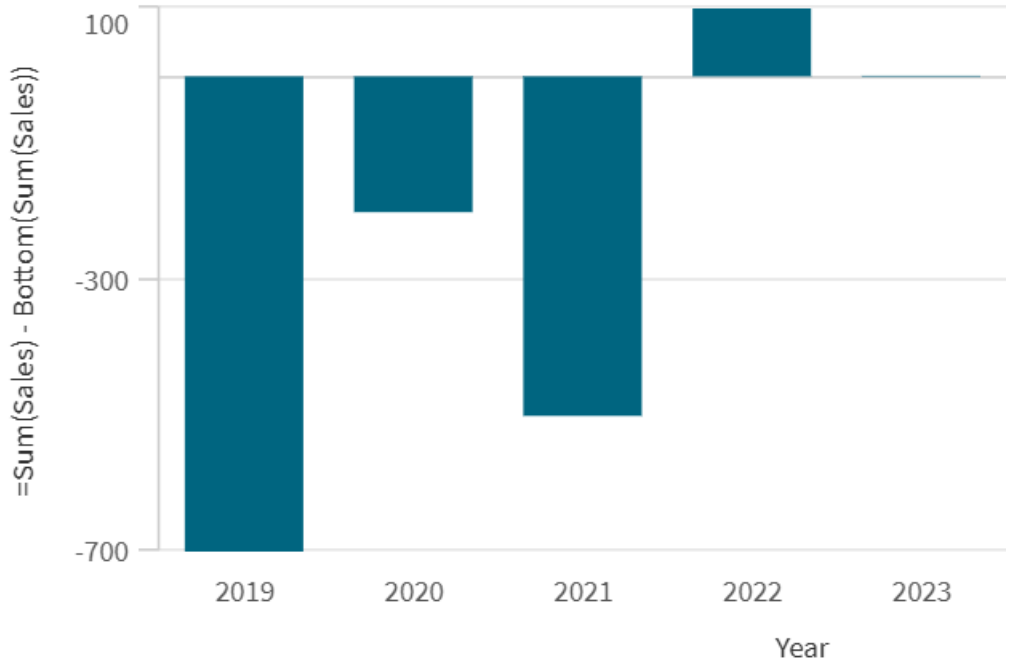
Skapa ett stapeldiagram och lägg sedan till det här fältet som en dimension:

- `Year`

Skapa följande mått:

- `=Sum(Sales) - Bottom(Sum(Sales))`för att returnera variansen i försäljning mellan den aktuella perioden och den senaste perioden.

Stapeldiagram som visar resultaten för funktionen Bottom



Om du använder funktionen `Bottom` i ett stapeldiagram får du en visuell jämförelse av variansen mellan försäljningen under den aktuella perioden och försäljningen under den senaste perioden.

Exempel – Bottom-scenario

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `OrderID`
  - `Product`
  - `SalesValue`
  - `OrderDate`

Laddningsskript

```
Example:
LOAD * INLINE [
OrderID, Product, SalesValue, OrderDate
1, "Product A", 500, '2023-01-10'
2, "Product B", 300, '2023-01-15'
3, "Product C", 450, '2023-02-01'
4, "Product D", 700, '2023-02-20'
5, "Product E", 350, '2023-03-05'
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Product`

Skapa följande mått:

- `=RangeAvg (Bottom(Sum(SalesValue),1,3))`, för att returnera genomsnittet av de 3 senaste raderna.
- `=Sum(SalesValue) - RangeAvg (Bottom(Sum(SalesValue),1,3))`, för att beräkna variansen mellan den aktuella raden och genomsnittet av de 3 senaste raderna.
- `=Sum(SalesValue)`, för att beräkna totalt `SalesValue` för den aktuella raden.

Resultattabell

Product	RangeAvg (Bottom(Sum (SalesValue),1,3))	Sum(SalesValue) - RangeAvg (Bottom (Sum(SalesValue),1,3))	Sum (SalesValue)
Totals	**500**	**1800**	**2300**
2019	500	0	500
2020	500	-200	300
2021	500	-50	450
2022	500	200	700
2023	500	-150	350

Resultaten visar hur du kan använda kombinationen av funktionerna `RangeSum` och `Bottom` för att analysera och jämföra försäljningsvärden.

Exempel – diagramuttryck som använder Bottom

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Två datatabeller som heter `Monthnames` och `Sales2013`.

Laddningsskript

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
```

```
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.
- `=Bottom(Sum(Sales))`, för att visa summan av försäljningen för `Customer` på nedersta raden.
- `=Sum(Sales)+Bottom(Sum(Sales))`, för att lägga till `Sum(Sales)`-värdet av aktuell `Customer` med `Customer` i den nedersta raden.
- `=Sum(Sales)+Bottom(Sum(Sales), 3)`, för att lägga till `Sum(Sales)`-värdet av aktuell `Customer` med `Customer` i tredje raden nedifrån.

Resultattabell

Customer	Sum (Sales)	Bottom(Sum (Sales))	Sum(Sales)+Bottom (Sum(Sales))	Sum(Sales)+Bottom(Sum (Sales), 3)
Totals	**2566**	**757**	**3323**	**3105**
Astrida	587	757	1344	1126
Betacab	539	757	1296	1078
Canutilty	683	757	1440	1222
Divadip	757	757	1514	1296

Måttet `Bottom(Sum(Sales))` returnerar `757` för alla rader eftersom det är värdet för den nedersta raden: `Divadip`.

För måttet `Sum(Sales)+Bottom(Sum(Sales))`, ger raden `Astrida` `1344`, vilket är resultatet av att lägga till `Sum (Sales)`-värdet för `Astrida` och `Divadip`, som är `Customer` i nedersta raden.

För måttet `Sum(Sales)+Bottom(Sum(Sales), 3)` är argumentet `offset` inställt på `3`. Uttrycket lägger till värdet `Sum(Sales)` för den aktuella raden och den tredje raden nedifrån, `Betacab`. Till exempel returneras värdet `1126` för kunden `Astrida`. Detta är resultatet av att lägga till värdet `Sum(Sales)` för `Astrida` och värdet `SumSales` för `Betacab` (`587+539`).

Se även:

p *Top - diagramfunktion (page 1521)*

Column - diagramfunktion

Column() returnerar det värde som återfinns i kolumnen som motsvarar **ColumnNo** i en rak tabell om dimensionerna ignoreras. Exempelvis returnerar **Column(2)** värdet för den andra måttkolumnen.

Syntax:

```
Column(ColumnNo)
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
ColumnNo	Kolumnnumret för en kolumn i tabellen som innehåller ett mått. *Funktionen Column() ignorerar dimensionskolumner.*

Begränsningar:

- Rekursiva anrop returnerar NULL.
- Om **ColumnNo** refererar till en kolumn för vilken det inte finns något mått, returneras ett NULL-värde.
- Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen.

Exempel: diagramuttryck

Exempel	Resultat
`Column(1)/Column(2)`	Returnerar värdet för det första måttet dividerat med värdet för det andra måttet i diagrammet.

Exempel – grunderna i Column

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `Customer`
    - `Product`
    - `UnitSales`
    - `UnitPrice`

Laddningsskript

```
Example:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Customer`
- `Product`
- `UnitSales`
- `UnitPrice`

Skapa följande mått:

- `=Sum(UnitPrice*UnitSales)`, för att få fram den årliga försäljningen.
- `=Sum(TOTAL UnitPrice*UnitSales)`, för att få fram det totala försäljningsvärdet.
- `=100*Column(1)/Column(2)`, för att beräkna procenten på den totala försäljningen.

Resultattabell

Customer	Product	UnitPrice	UnitSales	Sum (UnitPrice*UnitSales)	Sum(TOTAL UnitPrice*UnitSales)	100*Column (1)/Column (2)
Totals				**505**	**505**	**100.00**
Astrida	AA	15	10	150	505	29.70
Astrida	AA	16	4	64	505	12.67
Astrida	BB	9	9	81	505	16.04
Betacab	BB	10	5	50	505	9.90
Betacab	CC	20	2	40	505	7.92
Betacab	DD	25		0	505	0.00
Canutility	AA	15	8	120	505	23.76
Canutility	CC	19		0	505	0.00

Resultaten visar att du kan analysera data genom att hänvisa till en kolumnposition och återanvända den logiken i ett måttuttryck. Till exempel tar måttet `100*Column(1)/Column(2)` värdet från det första måttet, `Sum(UnitPrice*UnitSales)`, dividerar det med värdet från det andra måttet, `Sum(TOTAL UnitPrice*UnitSales)`, och multiplicerar sedan resultatet med `100`.

Exempel – Column-scenario

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Month`
  - `Sales`
  - `Expenses`
  - `Profit`

Laddningsskript

```
Example:
LOAD * INLINE [
Month, Sales, Expenses, Profit
Jan, 10000, 4000, 6000
```

```
Feb, 12000, 4500, 7500
Mar, 15000, 5000, 10000
Apr, 11000, 4200, 6800
May, 13000, 4800, 8200
Jun, 14000, 4700, 9300
Jul, 16000, 5100, 10900
Aug, 12500, 4300, 8200
Sep, 15500, 4900, 10600
Oct, 13500, 4600, 8900
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Product`

Skapa följande mått:

- `=Sum(Sales)`, för att sammanställa försäljningen.
- `=Sum(Expenses)`, för att sammanställa försäljningen.
- `=Column(1) - Column(2)`, för att beräkna skillnaden mellan kolumnen för försäljning och kolumnen för kostnader.

Resultattabell

Month	Sum(Sales)	Sum(Expenses)	Column(1) - Column(2)
Totals	**132500**	**46100**	**86400**
Jan	10000	4000	6000
Feb	12000	4500	7500
Mar	15000	5000	10000
Apr	11000	4200	6800
May	13000	4800	8200
Jun	14000	4700	9300
Jul	16000	5100	10900
Aug	12500	4300	8200
Sep	15500	4900	8200
Oct	13500	4600	8900

Tabellresultaten visar hur man använder funktionen `Column` för att utföra dataanalys. Måttet `Column(1) - Column(2)` anger det försäljningsbelopp som återstår efter att kostnaderna har dragits av (den andra måttkolumnen).

Exempel – Column-scenario analyserar försäljningsdata

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ProductName`
  - `Q1_Sales`
  - `Q2_Sales`
  - `Q3_Sales`
  - `Q4_Sales`

Laddningsskript

```
Example:
LOAD * INLINE [
ProductName, Q1_Sales, Q2_Sales, Q3_Sales, Q4_Sales
Product A, 1200, 1500, 1100, 1300
Product B, 1400, 1600, 1200, 1450
Product C, 1350, 1700, 1150, 1550
Product D, 1250, 1550, 1300, 1500
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `ProductName`

Skapa följande mått:

- `=Sum(Q1_Sales)`, för att sammanställa försäljningen från första kvartalet.
- `=Sum(Q2_Sales)`, för att sammanställa försäljningen från andra kvartalet.
- `=Sum(Q3_Sales)`, för att sammanställa försäljningen från tredje kvartalet.
- `=Sum(Q4_Sales)`, för att sammanställa försäljningen från fjärde kvartalet.
- `=Column(1) - Column(2)`, för att beräkna förhållandet mellan `Q1_Sales` och `Q2_Sales`.
- `=(Column(2) - Column(1)) / Column(1)` för att beräkna tillväxttakten mellan två perioder.

Resultattabell

ProductName	Sum (Q1_ Sales)	Sum (Q2_ Sales)	Sum (Q3_ Sales)	Sum (Q4_ Sales)	Column(1) - Column(2)	(Column(2) - Column(1)) / Column(1)
Totals	**5200**	**6350**	**4750**	**5800**	**0.81889763779528**	**0.22115384615385**
Product A	1200	1500	1100	1300	0.8	0.25
Product B	1400	1600	1200	1450	0.875	0.14285714285714
Product C	1350	1700	1150	1550	0.79411764705882	0.25925925925926
Product D	1250	1550	1300	1500	0.80645161290323	0.24

Tabellresultaten visar att du kan använda funktionen `Column` för att analysera data.

Dimensionality - diagramfunktion

Dimensionality() returnerar antalet dimensioner för den aktuella raden. Vad gäller pivottabeller returnerar funktionen det totala antalet dimensionskolumner som har icke-aggregerat innehåll, d.v.s. inte innehåller delsummor eller dolda aggregeringar.

Syntax:

```
Dimensionality ( )
```

Returnerad datatyp: heltal

Begränsningar:

Funktionen är endast tillgänglig i diagram. För alla diagramtyper utom pivottabeller returnerar den antalet dimensioner i alla rader utom summan, som blir 0.

Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen.

Exempel: diagramuttryck som använder Dimensionality

Diagramuttryck

Översikt

Använd funktionen `Dimensionality` i en pivottabell som ett diagramuttryck där du vill använda olika cellformatering beroende på antalet dimensioner i en rad som har icke-aggregerade data. Det här exemplet tillämpar en bakgrundsfärg till tabellceller som motsvarar ett visst tillstånd.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `Country`
    - `Product`
    - `Sales`
    - `Budget`

Laddningsskript

```
Example:
Load * inline [
Country, Product, Sales, Budget
Sweden, AA, 100000, 50000
Germany, AA, 125000, 175000
Canada, AA, 105000, 98000
Norway, AA, 74850, 68500
Ireland, AA, 49000, 48000
Sweden, BB, 98000, 99000
Germany, BB, 115000, 175000
Norway,BB,71850,68500
Ireland,BB,31000,48000
] (delimiter is ',');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till dessa fält som raddimensioner:

- `Country`
- `Product`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av `Sales`-värdena.
- `=Sum(Budget)`, för att beräkna summan av `Budget`-värdena.
- `=Dimensionality()`, för att ange antalet dimensioner för aktuell rad.

I panelen **Egenskaper** för mätvärdet `Sum(Sales)` anger du följande uttryck som **Bakgrundsfärguttryck**:

```
=If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156),
  If(Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)
  )
 )
```

Utdata från bakgrundsfärgsuttrycket ger en ljusröd eller mörkröd cellbakgrund till tabellcellerna när de tillämpliga villkoren är uppfyllda.

Pivottabell som visar resultat för radens dimensioner: Country, Product

Country Product			
	Sum(Sales)	Sum(Budget)	Dimensionality()
(+) Canada	105000	98000	1
(-) Germany	240000	350000	1
AA	125000	175000	2
BB	115000	175000	2
(+) Ireland	80000	96000	1
(+) Norway	146700	137000	1
(+) Sweden	198000	149000	1

När den första radens dimension (`Country`) visas, returnerar funktionen `Dimensionality 1`. När du expanderar en raddimension, t.ex. `Germany`, för att visa den andra raddimensionen (`Product`), returnerar funktionen `2`.

Bakgrundsfärguttrycken för måtte `Sum(Sales)` utför beräkningar baserade på dimensionalitet och försäljning för att lyfta fram olika resultat. När dimensionen är `1` visas alla `Country` med `Sum(Sales)` som är mindre än `Sum(Budget)` i ljusrött, i det här exemplet `Germany` och `Ireland`. När dimensionen är `2` visas alla `Product` med `Sum(Sales)` som är mindre än `Sum(Budget)` i mörkrött, i det här exemplet `AA` och `BB` för `Germany`.

Exempel – Dimensionality-scenario

Diagramuttryck

Översikt

I det här exemplet används funktionen `Dimensionality` för att tillämpa detaljer på en rapport och markera tabellceller för vissa dimensioner.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Region`
  - `SalesProduct`
  - `Year`
  - `SalesValue`

- Quantity
- Product

Laddningsskript

```
Example:
LOAD * INLINE [
Region, SalesProduct, Year, SalesValue, Quantity, Profit
North, Laptop, 2023, 20000, 15, 4000
North, Mobile, 2023, 15000, 25, 3000
North, Laptop, 2022, 18000, 20, 3500
North, Mobile, 2022, 14000, 22, 2800
South, Laptop, 2023, 22000, 18, 4500
South, Mobile, 2023, 16000, 27, 3200
South, Laptop, 2022, 19000, 19, 3800
South, Mobile, 2022, 15000, 23, 3100
East, Laptop, 2023, 21000, 16, 4200
East, Mobile, 2023, 15500, 24, 3100
East, Laptop, 2022, 18500, 18, 3900
East, Mobile, 2022, 14500, 21, 2950
West, Laptop, 2023, 23000, 17, 4700
West, Mobile, 2023, 16500, 26, 3300
West, Laptop, 2022, 19500, 20, 4000
West, Mobile, 2022, 15500, 24, 3100
North, Tablet, 2023, 12000, 10, 2500
South, Tablet, 2023, 13000, 12, 2700
East, Tablet, 2023, 12500, 11, 2600
West, Tablet, 2023, 13500, 13, 2800
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till dessa fält som raddimensioner:

- Year
- SalesProduct
- Region

Skapa följande mått:

- `=If(Dimensionality()=1, 'Total', 'Detail')`, för att ange om det är en `Total`- eller `Detail`-rad.
- `=If(Dimensionality()=3, 'Region-Level', If(Dimensionality()=2, 'Product-Level', 'Summary-Level'))`, för att ange vilken dimension som visas.
- `=Sum(SalesValue)`, för att beräkna summan av försäljningen.

I panelen **Egenskaper** för mätvärdet `Sum(SalesValue)` anger du följande uttryck som **Bakgrundsfärguttryck**:

```
=If(Dimensionality()=1, LightGray())
```

Utdata från bakgrundsfärgsuttrycket ger en ljusgrå cellbakgrund till tabellcellerna när dimensionen är `1`.

Pivottabell som visar resultat för radens dimensioner: Country, Product

År			
Försäljningsprodukt			
Region			
	Om (Dimensionality ()=1, 'Totalt', 'Detalj')	Om(Dimensionality()=3, 'Regionnivå', Om (Dimensionality()=2, 'Produktnivå', 'Sammanfattningsnivå'))	Summa (Försäljningsvärde)
(+) 2022	Totalt	Sammanfattningsnivå	134000
(-) 2023	Totalt	Sammanfattningsnivå	200000
(-)Bärbar dator	Detaljerad	Produktnivå	86000
Öst	Detaljerad	Regionsnivå	21000
Nord	Detaljerad	Regionsnivå	20000
Syd	Detaljerad	Regionsnivå	22000
Väst	Detaljerad	Regionsnivå	23000
(-) Mobil	Detaljerad	Produktnivå	63000
(-) Surfplatta	Detaljerad	Produktnivå	51000

Du kan se hur funktionen `Dimensionality` kan användas för att förbättra rapportens detaljnivå och lyfta fram alla dimensioner på den översta raden.

Exists

Exists() avgör om ett specifikt fältvärde redan har laddats in i fältet i dataladdningsskriptet. Funktionen returnerar TRUE eller FALSE, så att det kan användas i **where**-satsen för en **LOAD**-sats eller en **IF**-sats.

> *Du kan även använda **Not Exists()** för att avgöra om ett fältvärde inte har laddats, men du bör vara försiktig om du använder **Not Exists()** i en where-sats. **Exists()**-funktionen testar både tidigare laddade tabeller och tidigare laddade värden i den aktuella tabellen. Så endast den första förekomsten laddas. När den kommer till den andra förekomsten har värdet redan laddats. Se exemplen för mer information.*

Syntax:

```
Exists(field_name [, expr])
```

Returnerad datatyp: Booleansk

Argument:

Argument

Argument	Beskrivning
field_name	Namnet på fältet där du vill söka efter ett värde. Du kan använda explicita fältnamn utan citattecken. Fältet måste redan vara laddat av skriptet. Det vill säga, du kan inte hänvisa till ett fält som laddas i en sats längre ned i skriptet.
expr	Värdet som du vill kontrollera om det finns. Du kan använda ett explicit värde eller ett uttryck som hänvisar till ett eller flera fält i den aktuella load-satsen. *Du kan inte hänvisa till fält som inte ingår i den aktuella load-satsen.* Det här argumentet är valfritt Om du utesluter det kommer funktionen att kontrollera om värdet **field_name** i den aktuella posten redan existerar.

Exempel och resultat:

Exempel 1

```
Exists (Employee)
```

Returnerar -1 (True) om fältets värde **Employee** i den aktuella posten redan finns i en tidigare läst post som innehåller detta fält.

Satserna `Exists (Employee, Employee)` och `Exists (Employee)` är ekvivalenta.

Exempel 2

```
Exists(Employee, 'Bill')
```

Returnerar -1 (True) om fältvärdet **'Bill'** hittas i det aktuella innehållet i fältet **Employee**.

Exempel 3

```
Employees:
LOAD * inline [
Employee|ID|Salary
Bill|001|20000
John|002|30000
Steve|003|35000
] (delimiter is '|');

Citizens:
Load * inline [
```

```
Employee|Address
Bill|New York
Mary|London
Steve|Chicago
Lucy|Madrid
Lucy|Paris
John|Miami
] (delimiter is '|') where Exists (Employee);

Drop Tables Employees;
```

Detta resulterar i en tabell som du kan använda i en tabellvisualisering med dimensionerna Employee och Address.

where-satsen, `where Exists (Employee)`, betyder att enbart namnen från tabellen Citizens som också finns i Employees laddas till den nya tabellen. Satsen Drop avlägsnar tabellen Employees för att undvika sammanblandning.

Resultat

Employee	Address
Bill	New York
John	Miami
Steve	Chicago

Exempel 4

```
Employees:
Load * inline [
Employee|ID|Salary
Bill|001|20000
John|002|30000
Steve|003|35000
] (delimiter is '|');

Citizens:
Load * inline [
Employee|Address
Bill|New York
Mary|London
Steve|Chicago
Lucy|Madrid
Lucy|Paris
John|Miami
] (delimiter is '|') where not Exists (Employee);

Drop Tables Employees;
```

where-satsen innehåller not: `where not Exists (Employee).`

Detta innebär att enbart namnen från tabellen Citizens som inte finns i Employees laddas in i den nya tabellen.

Observera att det finns två värden för Lucy i tabellen Citizens, men att endast ett inkluderas i resultattabellen. När du laddar den första raden med värdet Lucy inkluderas det i fältet Employee. När den andra raden kontrolleras finns alltså redan värdet.

Resultat

Namn för den anställde	Adress
Mary	London
Lucy	Madrid

Exempel 5

Det här exemplet visar hur du laddar alla värden.

```
Employees:
Load Employee As Name;
LOAD * inline [
Employee|ID|Salary
Bill|001|20000
John|002|30000
Steve|003|35000
] (delimiter is '|');

Citizens:
Load * inline [
Employee|Address
Bill|New York
Mary|London
Steve|Chicago
Lucy|Madrid
Lucy|Paris
John|Miami
] (delimiter is '|') where not Exists (Name, Employee);

Drop Tables Employees;
```

För att hämta alla värden för Lucy ändrades två saker:

- En tidigare laddning i tabellen Employees infogades där Employee döptes om till Name.
  `Load Employee As Name;`
- Where-villkoret i Citizens ändrades till:
  `not Exists (Name, Employee).`

Det skapar fält för Name och Employee. När den andra raden med Lucy kontrolleras finns den fortfarande inte i Name.

Resultat

Namn för den anställde	Adress
Mary	London
Lucy	Madrid
Lucy	Paris

FieldIndex

FieldIndex() returnerar den placering som fältvärdet **value** har i fältet **field_name** (i laddningsordning).

Syntax:

```
FieldIndex(field_name , value)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
field_name	Namnet på det fält för vilket index krävs. Till exempel, kolumnen i en tabell. Måste anges som ett strängvärde. Detta innebär att fältnamnet måste omges av enkla citattecken.
value	Värdet för fältet **field_name**.

Begränsningar:

- Om **value** inte påträffas bland fältvärdena i fältet **field_name** returneras 0.
- Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen. Den här begränsningen gäller inte funktionen ekvivalent skript.

Exempel och resultat:

I följande exempel används fältet: **First name** från tabellen **Names**.

Exempel och resultat

Exempel	Resultat
Lägg till exempeldata i appen och kör den.	Tabellen **Names** är laddad, som i exempeldata,

Exempel	Resultat
Diagramfunktion: I en tabell som innehåller dimensionen First name, lägger du till följande som mått.	
`FieldIndex ('First name','John')`	1, eftersom John visas först i laddningsordningen för fältet **First name**. Observera att i en filterruta skulle **John** visas som nummer 2 från toppen eftersom den sorteras i alfabetisk ordning och inte i laddningsordning.
`FieldIndex ('First name','Peter')`	4, eftersom **FieldIndex()** returnerar endast ett värde, nämligen den första förekomsten i laddningsordningen.
Skriptfunktion: Då tabellen **Names** är inläst, som i exempeldata:	

`John1:`
 	
`Load FieldIndex('First name','John') as MyJohnPos`
 	
`Resident Names;`
	`MyJohnPos=1`, eftersom John visas först i laddningsordningen för fältet **First name**. Observera att i en filterruta skulle **John** visas som nummer 2 från toppen eftersom den sorteras i alfabetisk ordning och inte i laddningsordning.

`Peter1:`
 	
`Load FieldIndex('First name','Peter') as MyPeterPos`
 	
`Resident Names;`
	`MyPeterPos=4`, eftersom **FieldIndex()** returnerar endast ett värde, nämligen den första förekomsten i laddningsordningen.

Data som används i exemplet:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');

John1:
Load FieldIndex('First name','John') as MyJohnPos
Resident Names;

Peter1:
Load FieldIndex('First name','Peter') as MyPeterPos
Resident Names;
```

FieldValue

FieldValue() returnerar det värde som återfinns i position **elem_no** i fältet **field_name** (i inläsningsordning).

```
FieldValue(field_name , elem_no)
```

Returnerad datatyp: Dual

Argument

Argument	Beskrivning
field_name	Namnet på det fält för vilket värdet krävs. Till exempel, kolumnen i en tabell. Måste anges som ett strängvärde. Detta innebär att fältnamnet måste omges av enkla citattecken.
elem_no	Lägesnumret (elementnumret) på fältet efter laddningsordning, som värdet returneras för. Detta kan motsvara en rad i en tabell, men det beror på i vilken ordning elementen (raderna) läses in.

- Om **elem_no** är större än antalet fältvärden returneras NULL.
- Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen. Den här begränsningen gäller inte funktionen ekvivalent skript.

Exempel

Laddningsskript

Ladda följande data som en inline load i Skriptredigeraren för att skapa exemplet nedan.

```
Names:

LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC |No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');

John1:

Load FieldValue('First name',1) as MyPos1
Resident Names;

Peter1:
```

```
Load FieldValue('First name',5) as MyPos2
Resident Names;
```

Skapa en visualisering

Skapa en tabell visualisering i ett Qlik Sense-ark. Lägg till fälten **First name**, **MyPos1** och **MyPos2** i tabellen.

Resultat

First name	MyPos1	MyPos2
Jane	John	Jane
John	John	Jane
Mark	John	Jane
Peter	John	Jane
Sue	John	Jane

Förklaring

FieldValue('First name','1') leder till John som värde för **MyPos1** för alla förnamn eftersom John visas först i laddningsordningen för fältet **First name**. Observera att John skulle visas som nummer 2 från toppen i en filterruta, efter Jane, eftersom den sorteras alfabetiskt och inte efter inläsningsordning.

FieldValue('First name','5') leder till Jane som värde för **MyPos2** för alla förnamn eftersom Jane visas som det femte i laddningsordningen för fältet **First name**.

FieldValueCount

FieldValueCount() är en **heltal**sfunktion som returnerar antalet distinkta värden i ett fält.

En partiell inläsning kan ta bort värden från data, vilka inte kommer att återspeglas i antalet returnerade. Det returnerade numret kommer att motsvara alla distinkta värden som lästes in, antingen i den första inläsningen eller någon efterföljande partiell inläsning.

> *Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen. Den här begränsningen gäller inte funktionen ekvivalent skript.*

Syntax:

```
FieldValueCount(field_name)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
field_name	Namnet på det fält för vilket värdet krävs. Till exempel, kolumnen i en tabell. Måste anges som ett strängvärde. Detta innebär att fältnamnet måste omges av enkla citattecken.

Exempel och resultat:

I följande exempel används fältet **First name** från tabellen **Names**.

Exempel och resultat

Exempel	Resultat
Lägg till exempeldata i appen och kör den.	Tabellen **Names** är laddad, som i exempeldata,
Diagramfunktion: I en tabell som innehåller dimensionen First name, lägger du till följande som mått.	
`FieldValueCount('First name')`	5 eftersom **Peter** visas två gånger.
`FieldValueCount('Initials')`	6 eftersom **Initials** endast innehåller distinkta värden.
Skriptfunktion: Då tabellen **Names** är inläst, som i exempeldata:	

`FieldCount1:`
 	
`Load FieldValueCount('First name') as MyFieldCount1`
 	
`Resident Names;`
	`MyFieldCount1`=5, eftersom ”Peter” visas två gånger

`FieldCount2:`
 	
`Load FieldValueCount('Initials') as MyInitialsCount1`
 	
`Resident Names;`
	`MyFieldCount1`=6, eftersom ”Initials” endast innehåller distinkta värden.

Data som används i exempel:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

```
FieldCount1:
Load FieldValueCount('First name') as MyFieldCount1
Resident Names;

FieldCount2:
Load FieldValueCount('Initials') as MyInitialsCount1
Resident Names;
```

LookUp

LookUp() letar in i en tabell som redan är laddad och returnerar värdet av **field_name** vilket motsvarande den första förekomsten av värdet **match_field_value** i fältet **match_field_name**. Tabellen kan vara den aktuella tabell eller en annan tabell som har laddats.

Syntax:

```
LookUp(field_name, match_field_name, match_field_value [, table_name])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
field_name	Namnet på det fält för vilket returvärdet krävs. Inmatat värde måste anges som en sträng (t ex en litteral inom citationstecken).
match_field_name	Namnet på fältet som **match_field_value** ska sökas i. Inmatat värde måste anges som en sträng (t ex en litteral inom citationstecken).
match_field_value	Värdet som ska sökas upp i **match_field_name**-fältet.
table_name	Namnet på tabellen där värdet ska sökas. Inmatat värde måste anges som en sträng (t.ex. en litteral inom citationstecken). Om **table_name** utelämnas, antas aktuell tabell.

Argument utan citattecken syftar på den aktuella tabellen. Om du vill hänvisa till andra tabeller omger du ett argument med enkla citattecken.

Begränsningar:

Sökordningen är densamma som laddningsordningen, såvida inte tabellen är resultatet av komplexa operationer som join-operationer. I så fall är ordningen inte väldefinierad. Både **field_name** och **match_field_name** måste vara fält i samma tabell, som angivet med **table_name**.

Om ingen motsvarighet påträffas, returneras NULL.

Exempel

Laddningsskript

Ladda följande data som en inline load i Skriptredigeraren för att skapa exemplet nedan.

```
ProductList:
Load * Inline [
ProductID|Product|Category|Price
1|AA|1|1
2|BB|1|3
3|CC|2|8
4|DD|3|2
] (delimiter is '|');

OrderData:
Load *, Lookup('Category', 'ProductID', ProductID, 'ProductList') as CategoryID
Inline [
InvoiceID|CustomerID|ProductID|Units
1|Astrida|1|8
1|Astrida|2|6
2|Betacab|3|10
3|Divadip|3|5
4|Divadip|4|10
] (delimiter is '|');

Drop Table ProductList;
```

Skapa en visualisering

Skapa en tabellvisualisering i ett Qlik Sense-ark. Lägg till fälten **ProductID**, **InvoiceID**, **CustomerID**, **Units** och **CategoryID** i tabellen.

Resultat

Resultattabell

ProductID	InvoiceID	CustomerID	Units	CategoryID
1	1	Astrida	8	1
2	1	Astrida	6	1
3	2	Betacab	10	2
3	3	Divadip	5	2
4	4	Divadip	10	3

Förklaring

Exempeldata använder **Lookup()**-funktionen i följande form:

```
Lookup('Category', 'ProductID', ProductID, 'ProductList')
```

ProductList-tabellen laddas föst.

Lookup()-funktionen används för att bygga **OrderData**-tabellen. Den anger det tredje argumentet som **ProductID**. Detta är det fält för vilket en sökning efter värdet ska göras i det andra argumentet **'ProductID'** i **ProductList** enligt vad som anges av de omgivande enkla citattecknen.

Funktionen returnerar värdet för "**Category**" (i tabellen **ProductList**), laddad som **CategoryID**.

Satsen **drop** raderar tabellen **ProductList** från datamodellen eftersom den inte behövs, vilket ger tabellen **OrderData** som resultat.

> *Lookup()-funktionen är flexibel och kan komma åt alla tabeller som laddats tidigare. Det går dock långsamt jämfört med Applymap()-funktionen.*

Se även:

p *ApplyMap (page 1560)*

NoOfRows

NoOfRows() returnerar antalet rader i det aktuella kolumnsegmentet i en tabell. För bitmappsdiagram returnerar **NoOfRows()** antalet rader i diagrammets raka tabellmotsvarighet.

Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner, utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

> *Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen.*

Syntax:

```
NoOfRows([TOTAL])
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

Exempel: diagramuttryck som använder NoOfRows

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
        Temp:
LOAD * inline [
Region|SubRegion|RowNo()|NoOfRows()
Africa|Eastern
Africa|Western
Americas|Central
Americas|Northern
Asia|Eastern
Europe|Eastern
Europe|Northern
Europe|Western
Oceania|Australia
] (delimiter is '|');
```

Diagramuttryck

Skapa en tabellvisualisering i ett Qlik Sense-ark med **Region** och **SubRegion** som dimensioner. Lägg till `RowNo( ), NoOfRows()` och `NoOfRows(Total)` som mått.

Resultat

Region	SubRegion	RowNo()	NoOfRows()	NoOfRows (Total)
Africa	Eastern	1	2	9
Africa	Western	2	2	9
Americas	Central	1	2	9
Americas	Northern	2	2	9
Asia	Eastern	1	1	9
Europe	Eastern	1	3	9
Europe	Northern	2	3	9
Eurrope	Western	3	3	9
Oceania	Australia	1	1	9

Förklaring

I det här exemplet är sorteringsordningen efter den första dimensionen Region. Det leder till att varje kolumnsegment består av en grupp med regioner som har samma värde, till exempel Africa.

Kolumnen **RowNo()** visar radnummer för varje kolumnsegment. Till exempel finns det två rader för regionen Africa. Radnumreringen börjar sedan på 1 igen för nästa kolumnsegment, som är Americas.

Kolumnen **NoOfRows()** räknar antalet rader i varje kolumnsegment. Till exempel har Europa tre rader i kolumnsegmentet.

Kolumnen **NoOfRows(Total)** bortser från dimensionerna på grund av argumentet `TOTAL` för `NoOfRows()` och räknar raderna i tabellen.

Om tabellen sorteras på den andra dimensionen, SubRegion, baseras kolumnsegment på den dimensionen så att radnumreringen ändras för varje SubRegion.

Se även:

p *RowNo - diagramfunktion (page 701)*

Peek

Peek() returnerar värdet för ett fält i en tabell för en rad som redan har laddats. Radnumret kan anges, liksom tabellen. Om inget radnummer anges används posten som laddades senast.

Funktionen `Peek()` används vanligen för att hitta de relevanta gränserna i en tidigare laddad tabell, det vill säga det första eller sista värdet i ett specifikt fält. I de flesta fall lagras det här värdet i en variabel för att användas senare, till exempel som ett villkor i en do-while-loop.

```
Peek(
field_name
[, row_no[, table_name ] ])
```

Returnerad datatyp: Dual

Argument

Argument	Beskrivning
field_name	Namnet på det fält för vilket returvärdet krävs.Inmatat värde måste anges som en sträng (t ex en litteral inom citationstecken).
row_no	Den rad i tabellen som anger det fält som krävs. Kan vara ett uttryck, men måste lösas till ett heltal. 0 betecknar den första posten, 1 den andra, och så vidare. Negativa tal markerar ordningen från slutet av tabellen. -1 anger den senaste lästa posten. Om ingen **row_no** angivits, antas -1.

Argument	Beskrivning
table_name	En tabelletikett utan avslutande kolon. Om inget **table_name** har angetts antas den aktuella tabellen. Om det används utanför **LOAD**-satsen, eller refererar till en annan tabell, måste tabellnamnet **table_name** anges.

Funktionen kan bara returnera värden från poster som redan laddats. Det betyder att ett anrop som använder -1 som `row_no` kommer att returnera NULL för den första posten i en tabell.

Exempel och resultat:

Exempel 1

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
EmployeeDates:
Load * Inline [
EmployeeCode|StartDate|EndDate
101|02/11/2010|23/06/2012
102|01/11/2011|30/11/2013
103|02/01/2012|
104|02/01/2012|31/03/2012
105|01/04/2012|31/01/2013
106|02/11/2013|
] (delimiter is '|');

First_last_Employee:
Load
EmployeeCode,
Peek('EmployeeCode',0,'EmployeeDates') As FirstCode,
Peek('EmployeeCode',-1,'EmployeeDates') As LastCode
Resident EmployeeDates;
```

Resultattabell

Employee code	StartDate	EndDate	FirstCode	LastCode
101	02/11/2010	23/06/2012	101	106
102	01/11/2011	30/11/2013	101	106
103	02/01/2012	101	106	
104	02/01/2012	31/03/2012	101	106
105	01/04/2012	31/01/2013	101	106
106	02/11/2013	101	106	

FirstCode = 101, eftersom `Peek('EmployeeCode',0, 'EmployeeDates')` returnerar det första värdet för EmployeeCode i tabellen EmployeeDates.

LastCode = 106, eftersom `Peek('EmployeeCode',-1, 'EmployeeDates')` returnerar det sista värdet för EmployeeCode i tabellen EmployeeDates.

Att ersätta värdet på argumentet **row_no** returnerar värdena för andra rader i tabellen, enligt följande:

`Peek('EmployeeCode',2, 'EmployeeDates')` returnerar det tredje värdet, 103, i tabellen som FirstCode.

Observera dock att om inte tabellen specificeras som det tredje argumentet **table_name** i dessa exempel, refererar funktionen till den aktuella tabellen (i detta fall, internt).

Exempel 2

Om du vill komma åt data längre ned i en tabell kan du göra det i två steg: först laddar du hela tabellen i en temporär tabell och sedan sorterar du om den med användning av **Peek()**.

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
T1:
LOAD * inline [
ID|Value
1|3
1|4
1|6
3|7
3|8
2|1
2|11
5|2
5|78
5|13
] (delimiter is '|');

T2:

LOAD *,
IF(ID=Peek('ID'), Peek('List')&','&Value,Value) AS List
RESIDENT T1
ORDER BY ID ASC;
DROP TABLE T1;
```

Create a table in a sheet in your application with **ID**, **List**, and **Value** as the dimensions.

Resultattabell

ID	Lista	Värde
1	3,4	4
1	3,4,6	6
1	3	3
2	1,11	11
2	1	1
3	7,8	8

ID	Lista	Värde
3	7	7
5	2,78	78
5	2,78,13	13
5	2	2

Uttrycket **IF()** byggs från den temporära tabellen T1.
`Peek('ID')` hänvisar till fältet ID i föregående rad i den aktuella tabellen T2.
`Peek('List')` hänvisar till fältet List i föregående rad i tabellen T2, som byggs då uttrycket utvärderas.

Uttrycket utvärderas på följande sätt:
Om det aktuella värdet för ID är samma som det tidigare värdet för ID, då ska värdet för Peek('List') sammanlänkas med det aktuella värdet för Value. Annars skriver du det aktuella värdet av Value endast.

Om Peek('List') redan innehåller ett sammanlänkat resultat, kommer det nya resultatet Peek('List') sammanlänkas till det.

*Observera satsen **Order by**. Detta anger hur tabellen ordnas (genom ID i stigande ordning). Utan detta använder funktionen Peek() godtycklig ordning som den interna tabellen har, vilket kan leda till oförutsägbara resultat.*

Exempel 3

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
Amounts:
Load
Date#(Month,'YYYY-MM') as Month,
Amount,
Peek(Amount) as AmountMonthBefore
Inline
[Month,Amount
2022-01,2
2022-02,3
2022-03,7
2022-04,9
2022-05,4
2022-06,1];
```

Resultattabell

Amount	AmountMonthBefore	Month
1	4	2022-06
2	-	2022-01

Amount	AmountMonthBefore	Month
3	2	2022-02
4	9	2022-05
7	3	2022-03
9	7	2022-04

Fältet AmountMonthBefore kommer att innehålla mängden från föregående månad.

Eftersom parametrarna row_no och table_name är utelämnade, används standardvärdena. I det här exemplet är följande tre funktionsanrop likvärdiga:

- Peek(Amount)
- Peek(Amount,-1)
- Peek(Amount,-1,'Amounts')

När -1 används som row_no kommer värdet från föregående rad att användas. Genom att ersätta det här värdet går det att hämta värden för de andra raderna i tabellen:

Peek(Amount,2) returnerar det tredje värdet i tabellen: 7.

Exempel 4

Data måste vara korrekt sorterade för att få rätt resultat men det är tyvärr inte alltid fallet. Funktionen Peek () kan inte användas för att hänvisa till data som inte har laddats än. Du kan undvika sådana problem genom att använda temporära tabeller och köra flera pass för dina data.

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
tmp1Amounts:
Load * Inline
[Month,Product,Amount
2022-01,B,3
2022-01,A,8
2022-02,B,4
2022-02,A,6
2022-03,B,1
2022-03,A,6
2022-04,A,5
2022-04,B,5
2022-05,B,6
2022-05,A,7
2022-06,A,4
2022-06,B,8];

tmp2Amounts:
Load *,
If(Product=Peek(Product),Peek(Amount)) as AmountMonthBefore
Resident tmp1Amounts
Order By Product, Month Asc;
```

```
Drop Table tmp1Amounts;

Amounts:
Load *,
If(Product=Peek(Product),Peek(Amount)) as AmountMonthAfter
Resident tmp2Amounts
Order By Product, Month Desc;
Drop Table tmp2Amounts;
```

Förklaring

Den inledande tabellen är sorterad på månad. Det betyder att peek()-funktionen i många fall skulle returnera mängden för fel produkt. Det innebär att tabellen måste sorteras om. Detta utförs genom att köra ett andra pass för dina data och skapa den nya tabellen tmp2Amounts. Observera satsen Order By. Först ordnar den posterna efter produkt och därefter efter månad i stigande ordning.

If()-funktionen behövs eftersom AmountMonthBefore endast ska beräknas om den föregående raden innehåller data för samma produkt men bara för den föregående månaden. Genom att jämföra produkten på den aktuella raden med produkten på föregående rad kan det här villkoret valideras.

När den andra tabellen har skapats skippas den första tabellen tmp1Amounts med en Drop Table-sats.

Slutligen körs ett tredje pass för data, men nu med månaderna sorterade i omvänd ordning. Det betyder att även AmountMonthAfter kan beräknas.

Order By-satser anger hur tabellen ska ordnas. Utan detta använder funktionen Peek() en godtycklig ordning som den interna tabellen har, vilket kan leda till oförutsägbara resultat.

Resultat

Resultattabell

Month	Product	Amount	AmountMonthBefore	AmountMonthAfter
2022-01	A	8	-	6
2022-02	B	3	-	4
2022-03	A	6	8	6
2022-04	B	4	3	1
2022-05	A	6	6	5
2022-06	B	1	4	5
2022-01	A	5	6	7
2022-02	B	5	1	6
2022-03	A	7	5	4
2022-04	B	6	5	8

Month	Product	Amount	AmountMonthBefore	AmountMonthAfter
2022-05	A	4	7	-
2022-06	B	8	6	-

Exempel 5

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
T1:
Load * inline [
Quarter, Value
2003q1, 10000
2003q1, 25000
2003q1, 30000
2003q2, 1250
2003q2, 55000
2003q2, 76200
2003q3, 9240
2003q3, 33150
2003q3, 89450
2003q4, 1000
2003q4, 3000
2003q4, 5000
2004q1, 1000
2004q1, 1250
2004q1, 3000
2004q2, 5000
2004q2, 9240
2004q2, 10000
2004q3, 25000
2004q3, 30000
2004q3, 33150
2004q4, 55000
2004q4, 76200
2004q4, 89450 ];

T2:
Load *, rangesum(SumVal,peek('AccSumVal')) as AccSumVal;
Load Quarter, sum(Value) as SumVal resident T1 group by Quarter;
```

Resultat

Resultattabell

Quarter	SumVal	AccSumVal
2003q1	65000	65000
2003q2	132450	197450

Quarter	SumVal	AccSumVal
2003q3	131840	329290
2003q4	9000	338290
2004q1	5250	343540
2004q2	24240	367780
2004q3	88150	455930
2004q4	220650	676580

Förklaring

Load-satsen **Load *, rangesum(SumVal,peek('AccSumVal')) as AccSumVal** har ett rekursivt anrop där de tidigare värdena läggs till i det aktuella värdet. Den här åtgärden används till att beräkna en ackumulering av värden i skriptet.

Previous

Previous() returnerar värdet av uttrycket **expr** genom att använda data från en tidigare indatapost som inte uteslutits till följd av en **where**-sats. För den första posten i en intern tabell kommer funktionen att returnera NULL.

Syntax:

```
Previous(expr)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas. Uttrycket kan innehålla kapslade funktioner **Previous()** för att komma åt poster längre bak. Data hämtas direkt från indatakällan och man kan således referera även till fält som inte lästs in i Qlik Sense, dvs. även om de inte har lagrats i dess associativa databas.

Begränsningar:

För den första posten i en intern tabell kommer funktionen att returnera NULL.

Exempel:

Skriv följande i ditt laddningsskript

```
Sales2013:
```

```
Load *, (Sales - Previous(Sales) )as Increase Inline [

Month|Sales

1|12

2|13

3|15

4|17

5|21

6|21

7|22

8|23

9|32

10|35

11|40

12|41

] (delimiter is '|');
```

Genom att använda funktionen **Previous()** i satsen **Load** kan vi jämföra det aktuella värdet av Sales med föregående värde och använda det i ett tredje fält, Increase.

Resultattabell

Månad	Försäljning	Ökning
1	12	-
2	13	1
3	15	2
4	17	2
5	21	4
6	21	0
7	22	1
8	23	1
9	32	9

Månad	Försäljning	Ökning
10	35	3
11	40	5
12	41	1

Top - diagramfunktion

Top() utvärderar ett uttryck på den första (översta) raden i ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden högst upp. För diagram som inte är tabeller görs **Top()**-utvärderingen på den första raden i den aktuella kolumnen i diagrammets raka tabellmotsvarighet.

Syntax:

```
Top([TOTAL] expr [ , offset [,count ]])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Att specificera en **offset** på n, större än 1, flyttar ner utvärderingen av uttrycket n rader nedanför den översta raden. Anges ett negativt värde för startpunkten gör det att **Top**-funktionen fungerar likadant som **Bottom**-funktionen med motsvarande positivt värde för startpunkten.
count	Om en tredje parameter, **count**, anges till större än 1, returnerar funktionen ett intervall av **count**-värden, ett för var och en av de sista **count**-raderna i det aktuella kolumnsegmentet. I denna form kan funktionen användas som argument i någon av de speciella intervallfunktionerna. *Intervallfunktioner (page 1576)*
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

Ett kolumnsegment definieras som en konsekutiv underuppsättning med celler som har samma värden för dimensionerna i den aktuella sorteringsordningen. Postöverskridande diagramfunktioner beräknas i kolumnsegmentet exklusive dimensionen längst till höger i motsvarande raka tabelldiagram. Om det enbart finns en dimension i diagrammet, eller om kvalificeraren TOTAL anges, utvärderas uttrycket över en hel tabell.

Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner, utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

Begränsningar:

- Rekursiva anrop returnerar NULL.
- Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen.

Exempel: diagramuttryck

Exempel	Resultat
`Top( Sum(Sales) )`	Returnerar värdet på `Sales` för den första (översta) raden i ett kolumnsegment i en tabell.

Exempel – grunderna i Top

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Year`
  - `Sales`

Laddningsskript

```
Example:
LOAD * INLINE [
Year, Sales
2019, 1000
2020, 1500
```

```
2021, 1200
2022, 1800
2023, 1700
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Year`

Skapa följande mått:

- `=Sum(Sales)`, för att få fram den årliga försäljningen.
- `=Top(Sum(Sales))`, för att få fram försäljningsvärdet för det första året, det vill säga den första (översta) raden i en kolumn i en tabell.
- `=Sum(Sales) - Top(Sum(Sales))`, för att beräkna variansen mellan värdet på `Sales` för innevarande rad och värdet för den första (översta) raden i en kolumn.
- `=Sum(Sales)-Top(Sum(Sales), 3)`, för att beräkna variansen mellan försäljningsvärdet för innevarande rad och värdet för den tredje raden uppifrån.
- `=(Sales - Top(Sales)) / Top(Sales)`, för att ange förändringen i `Sales` som en kvot/procentandel jämfört med första raden. Om du vill visa värdet i procent väljer du **Nummer > Enkel formatering > 12,34%** under **Nummerformatering**.

Sortera tabellen efter dimensionen `Year` i stigande ordning.

Resultattabell

Year	Sum (Sales)	Top(Sum (Sales))	Sum(Sales) - Top (Sum(Sales))	Sum(Sales)-Top (Sum(Sales), 3)	(Sales - Top (Sales)) / Top (Sales)
Totals	**7200**	**1000**	**6200**	**6000**	**-**
2019	1000	1000	0	-200	0.00%
2020	1500	1000	500	300	50.00%
2021	1200	1000	200	0	20.00%
2022	1800	1000	800	600	80.00%
2023	1700	1000	700	500	70.00%

När du tittar på resultaten kan du jämföra `Sales`-värdena under den aktuella perioden med den översta (eller första) perioden.

- Måttet `Sum(Sales) - Top(Sum(Sales))` ger förändringen i `Sales`-värdet vid en jämförelse av aktuell period med den första.
- Måttet `Sum(Sales)-Top(Sum(Sales), 3)` visar summan av `Sales` för den aktuella raden minus den tredje raden uppifrån.

- Måttet `=(Sales - Top(Sales)) / Top(Sales)` ger förändringen i `Sales` jämfört med första perioden i procent.

Funktionen kan även användas i andra diagram, inte bara tabeller - exempelvis i stapeldiagram.

Gör följande:

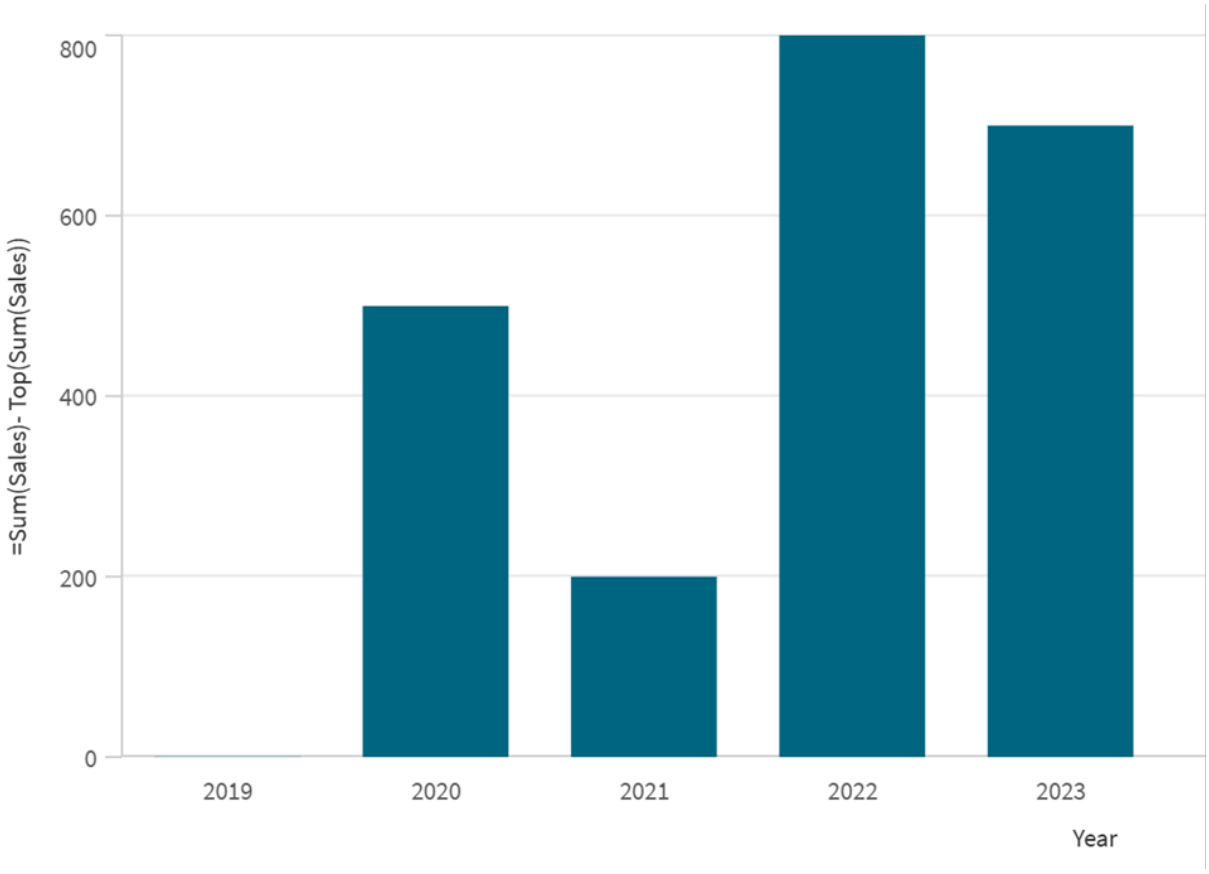
Skapa ett stapeldiagram och lägg sedan till det här fältet som en dimension:

- `Year`

Skapa följande mått:

- `=Sum(Sales)- Top(Sum(Sales))`för att returnera variansen i försäljning mellan den aktuella perioden och den översta perioden.

Stapeldiagram som visar resultaten för funktionen Top



Om du använder funktionen `Top` i ett stapeldiagram får du en visuell jämförelse av variansen mellan försäljningen under den aktuella perioden och försäljningen under den översta (eller första) perioden.

Exempel – Top-scenario

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `OrderID`
  - `Product`
  - `SalesValue`
  - `OrderDate`

Laddningsskript

```
Example:
LOAD * INLINE [
OrderID, Product, SalesValue, OrderDate
1, "Product A", 500, '2023-01-10'
2, "Product B", 300, '2023-01-15'
3, "Product C", 450, '2023-02-01'
4, "Product D", 700, '2023-02-20'
5, "Product E", 350, '2023-03-05'
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Product`

Skapa följande mått:

- `=RangeAvg(Top(Sum(SalesValue),1,3))` för att returnera medelvärdet av de tre översta raderna.
- `=Sum(SalesValue) - RangeAvg (Top(Sum(SalesValue),1,3))`, för att beräkna variansen mellan den aktuella raden och genomsnittet av de tre översta raderna.
- `=Sum(SalesValue)`, för att beräkna totalt `SalesValue` för den aktuella raden.

Resultattabell

Product	RangeAvg(Top(Sum (SalesValue),1,3))	Sum(SalesValue) - RangeAvg (Top (Sum(SalesValue),1,3))	Sum (SalesValue)
Totals	**416.67**	**1883.33**	**2300**
2019	416.67	83.33	500

Product	RangeAvg(Top(Sum (SalesValue),1,3))	Sum(SalesValue) - RangeAvg (Top (Sum(SalesValue),1,3))	Sum (SalesValue)
Totals	**416.67**	**1883.33**	**2300**
2020	416.67	--116.67	300
2021	416.67	33.33	450
2022	416.67	283.33	700
2023	416.67	-66.67	350

Resultaten visar hur du kan använda kombinationen av funktionerna `RangeSum` och `Top` för att analysera och jämföra försäljningsvärden.

Exempel – diagramuttryck som använder Top

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Två datatabeller som heter `Monthnames` och `Sales2013`.

Laddningsskript

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
```

```
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Customer`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.
- `=Top(Sum(Sales))`, för att visa summan av försäljningen för `Customer` på översta raden.
- `=Sum(Sales)+Top(Sum(Sales))`, för att lägga till `Sum(Sales)`-värdet av aktuell `Customer` med `Customer` i den översta raden.
- `=Sum(Sales)+Top(Sum(Sales), 3)`, för att lägga till `Sum(Sales)`-värdet av aktuell `Customer` med `Customer` i tredje raden uppifrån.

Resultattabell

Customer	Sum (Sales)	Top(Sum (Sales))	Sum(Sales)+Top(Sum (Sales))	Sum(Sales)+Top(Sum (Sales), 3)
Totals	**2566**	**587**	**3153**	**3249**
Astrida	587	587	1174	1270
Betacab	539	587	1126	1222
Canutilty	683	587	1270	1366
Divadip	757	587	1344	1440

Måttet `Top(Sum(Sales))` returnerar `587` för alla rader eftersom det är värdet för den översta raden: `Astrida`.

För måttet `Sum(Sales)+Top(Sum(Sales))`, ger raden `Betacab` `1126`, vilket är resultatet av att lägga till `Sum (Sales)`-värdet för `Betacab` och `Astrida`, som är `Customer` i översta raden.

För måttet `Sum(Sales)+Top(Sum(Sales), 3)` är argumentet `offset` inställt på `3`. Uttrycket lägger till värdet `Sum(Sales)` för den aktuella raden och den tredje raden uppifrån, `Canutility`. Till exempel returneras värdet `1440` för kunden `Divadip`. Detta är resultatet av att lägga till värdet `Sum(Sales)` för `Divadip` och värdet `SumSales` för `Canutility` (`757+683`).

Se även:

p *Bottom - diagramfunktion (page 1483)*
p *Above - diagramfunktion (page 1459)*
p *Sum - diagramfunktion (page 401)*
p *RangeAvg (page 1579)*
p *Intervallfunktioner (page 1576)*

SecondaryDimensionality - diagramfunktion

SecondaryDimensionality() returnerar antalet dimensionspivottabellrader som har icke-aggregerat innehåll, d.v.s. inte innehåller delsummor eller dolda aggregeringar. Denna funktion motsvarar funktionen **dimensionality()** för horisontella pivottabelldimensioner.

Syntax:

```
SecondaryDimensionality( )
```

Returnerad datatyp: heltal

Begränsningar:

- Såvida funktionen **SecondaryDimensionality** inte används i pivottabeller returnerar den alltid 0.
- Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen.

Exempel – grunderna i SecondaryDimensionality

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Date`
  - `Product`
  - `Category`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Date, Product, Category, Sales
2024-01-01, Apple, Fruit, 100
2024-01-01, Banana, Fruit, 80
2024-02-01, Apple, Fruit, 120
2024-02-01, Banana, Fruit, 90
2024-03-01, Apple, Fruit, 110
```

```
2024-03-01, Banana, Fruit, 85
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en raddimension:

- `Date`

Lägg till dessa fält som kolumndimensioner:

- `Category`
- `Product`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna den totala försäljningen.
- `=SecondaryDimensionality()`, för att returnera dimensionstalet för att demonstrera användningen av denna funktion.

Pivot table showing results for the dimension: Category

	(+) Fruit	
Date	**Sum(Sales)**	**SecondaryDimensionality()**
2024-01-01	180	1
2024-02-01	120	1
2024-03-01	110	1

Pivottabellens resultat visar endast dimensionen `Category` som en kolumn. Därför returnerar funktionen `SecondaryDimensionality` värdet 1.

Om du expanderar dimensionen `Category`, `Fruit`, kommer tabellresultaten nu att innehålla värdena för dimensionen `Product`, `Apple` och `Banana`.

Pivot table showing results for two dimensions: Category, Product

Date	(-) Fruit			
	Apple		**Banana**	
	Sum(Sales)	**SecondaryDimensionality()**	**Sum (Sales)**	**SecondaryDimensionality()**
2024-01-01	100	2	80	2
2024-02-01	120	2	90	2
2024-03-01	110	2	85	2

Funktionen `SecondaryDimensionality` returnerar nu ett värde på `2`, eftersom det finns värden för två dimensioner i pivottabellen: `Category` och `Product`.

Exempel – Använda SecondaryDimensionality för att tillämpa villkorlig formatering i ett diagramuttryck

Diagramuttryck

Översikt

I det här exemplet används funktionen i ett diagramuttryck i en pivottabell för att tillämpa villkorlig cellformatering beroende på vilken dimensionsnivå som visas.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `Year`
    - `Quarter`
    - `Product Category`
    - `Sales`

Laddningsskript

```
Example:
Load * inline [
Year, Quarter, Product Category, Sales
2023, Q1, Electronics, 5000
2023, Q1, Furniture, 3000
2023, Q2, Electronics, 4000
2023, Q2, Furniture, 3500
2023, Q3, Electronics, 6000
2023, Q3, Furniture, 4500
2023, Q4, Electronics, 7000
2023, Q4, Furniture, 5000
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en raddimension:

- `Product Category`

Lägg till dessa fält som kolumndimensioner:

- `Year`
- `Quarter`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna den totala försäljningen.

I panelen **Egenskaper** för mätvärdet `Sum(Sales)` anger du följande uttryck som **Bakgrundsfärguttryck**:

```
IF(SecondaryDimensionality() = 1,
RGB(144, 238, 144),  // Green for Year-level
RGB(173, 216, 230),  // Blue for Quarter-level
)
```

Uttrycket ger tabellcellerna en grön bakgrund när funktionen `SecondaryDimensionality` ger resultatet 1 och en blå bakgrund när funktionen ger ett annat resultat.

Pivottabell som visar resultat för en dimension: **År** med grön cellbakgrund

Produktkategori	**(+) 2023**
Elektronik	22000
Möbler	16000

Resultatet visar att när endast dimensionen `Year` visas, och värdet `SecondaryDimensionality` därför är 1, visas `Sum(Sales)`-cellerna med grön bakgrund. När du expanderar tabellen för att visa `Sum(Sales)` per `Quarter` visas bakgrunden nu i blått eftersom det finns två dimensioner som visas, `Year` och `Quarter`, och värdet `SecondaryDimensionality` är nu `2`.

Pivot table showing results for two dimensions: **Year** and **Quarter** with blue cell background

Product Category	**(-) 2023**			
	Q1	**Q2**	**Q3**	**Q4**
Electronics	5000	4000	6000	7000
Furniture	3000	3500	4500	5000

After - diagramfunktion

After() returnerar värdet för ett uttryck utvärderat med en pivottabells dimensionsvärden som de visas i kolumnen efter den aktuella kolumnen inom ett radsegment i pivottabellen.

Syntax:

```
after([TOTAL] expr [, offset [, count ]])
```

Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen.

Funktionen returnerar NULL i alla diagramtyper utom pivottabeller.

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Om en **offset** n större än 1 anges, flyttas utvärderingen av uttrycket n kolumner till höger om den aktuella kolumnen. En förskjutning på 0 utvärderar uttrycket i den aktuella kolumnen. Anges ett negativt värde för startpunkten gör det att **After**-funktionen fungerar likadant som **Before**-funktionen med motsvarande positivt värde för startpunkten.
count	Om en tredje parameter **count** anges till större än 1 returnerar funktionen ett intervall av värden, ett för var och en av tabellkolumnerna upp till värdet för **count** räknat åt höger från den ursprungliga cellen.
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

I radsegmentets sista kolumn kommer värdet NULL att returneras, eftersom det inte finns någon cell efter denna.

Om pivottabellen har flera horisontella dimensioner, inbegriper det aktuella radsegmentet endast kolumner som har samma värden som den aktuella kolumnen i samtliga dimensionsrader utom den rad som visar den sista dimensionen i fältens inbördes sorteringsordning. Fältens inbördes sorteringsordning för horisontella dimensioner i pivottabeller definieras helt enkelt som dimensionernas ordning uppifrån och ned.

Exempel: diagramuttryck

Exempel	Resultat
`After( Sum(Sales ))`	Returnerar summan av `Sales` från nästa kolumn omedelbart till höger i förhållande till aktuell kolumn.
`After( Sum(Sales ), 2)`	Returnerar summan av `Sales` från den kolumn som ligger två kolumner till höger om aktuell kolumn.
`After( Total Sum ( Sales ))`	Returnerar totalsumman av `Sales` från nästa kolumn omedelbart till höger i förhållande till aktuell kolumn.
`RangeAvg (After (Sum(x),1,3))`	Returnerar medelvärdet för de tre resultaten av `sum(x)`-funktionen utvärderat på de tre kolumnerna omedelbart till höger om aktuell kolumn.

Exempel – grunderna i After

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Year`
  - `Quarter`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Year, Quarter, Sales
2023, Q1, 5000
2023, Q2, 4000
2023, Q3, 6000
2023, Q4, 7000
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en raddimension:

- `Year`

Lägg till detta fält som en kolumndimension:

- `Quarter`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.
- `=After(Sum(Sales))`, för att visa summan av försäljningen under följande kvartal.

Resultattabell

År	Q1		Q2		Q3		Q4	
	Sum (Sales)	After (Sum (Sales))	Sum (Sales)	After (Sum (Sales))	Sum (Sales)	After (Sum (Sales))	Sum (Sales)	After (Sum (Sales))
2023	5000	4000	4000	6000	6000	7000	7000	-

När du tittar på resultaten kan du se hur funktionen `After` returnerar värdet för nästa kolumn som ligger omedelbart till höger om kolumnen `Sum(Sales)` för varje `Quarter`. Till exempel, i `Q1` är `Sum(Sales)`-värdet för den perioden `5000`. Utdata för funktionen `After` för `Q1` är `4000` eftersom detta är värdet för kolumnen omedelbart till höger om `Sum(Sales)` för `Q1`, nämligen `Sum(Sales)` för `Q2`.

Exempel – After-scenario

Diagramuttryck

Översikt

En datauppsättning innehåller flera års försäljningssiffror för sålda enheter hos ett elektronikföretag. Företaget vill presentera den totala försäljningen för varje år och jämföra enhetsförsäljningen från år till år.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Year`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Year, Sales
2021, 1350
2022, 1840
2023, 2250
2024, 1750
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en kolumndimension:

- `Year`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.
- `=Sum(Sales) - After(Sum(Sales))`, för att beräkna skillnaden i försäljning från år till år.

Resultattabell

År	2021		2022		2023		2024	
	Sum (Sales)	Sum (Sales)-After (Sum (Sales))	Sum (Sales)	Sum (Sales)-After (Sum (Sales))	Sum (Sales)	Sum (Sales)-After (Sum (Sales))	Sum (Sales)	Sum (Sales)-After (Sum (Sales))
2023	1350	-490	1840	-410	2250	500	1750	-

Resultatet visar summan av försäljningen för varje år och skillnaden mellan försäljningssiffrorna från ett år till ett annat. År `2022` var till exempel `Sum(Sales)` `1840`. Utdata från funktionen `After` visar att detta värde var `410` enheter mindre än summan av försäljningen för `2023`.

Exempel – After advanced-scenario

Diagramuttryck

Översikt

En datauppsättning innehåller försäljningssiffror för flera år. Företaget vill presentera den totala försäljningen för innevarande år och en rullande total för de kommande tre årens försäljning.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `SalesYear`
    - `Sales`

Laddningsskript

```
Example:
Load * inline [
SalesYear, Sales
2021, 1300
2022, 1800
2023, 2000
2024, 2150
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en kolumndimension:

- `SalesYear`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.
- `=RangeSum(After(Sum(Sales),1,3))`, för att beräkna summan av försäljningen under de kommande tre åren.

Resultattabell

2021		2022		2023		2024	
Sum (Sales)	RangeSum (After(Sum (Sales),1, 3))	Sum (Sales)	RangeSum (After(Sum (Sales),1, 3))	Sum (Sales)	RangeSum (After(Sum (Sales),1, 3))	Sum (Sales)	RangeSum (After(Sum (Sales),1, 3))
1300	5950	1800	4150	2000	2150	2150	0

Resultatet visar hur man använder funktionen `After` för att få fram summan av försäljningsvärdena för de kommande tre åren. I `2021` är till exempel den årliga `Sum(Sales)` `1300` och summan av försäljningen för de kommande tre åren (`2022, 2023, 2024`) är `5950`. Ett värde på `0` returneras för `2024` eftersom uppgifterna för kommande år inte är tillgängliga.

Before - diagramfunktion

Before()returnerar värdet av ett uttryck utvärderat men en pivottabells dimensionsvärden som de visas i kolumnen framför den aktuella kolumnen inom ett radsegment i pivottabellen.

Syntax:

```
Before([TOTAL] expr [, offset [, count]])
```

Funktionen returnerar NULL i alla diagramtyper utom pivottabeller.

Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen.

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Om en **offset** n större än 1 anges, flyttas utvärderingen av uttrycket n kolumner till vänster om den aktuella kolumnen. En förskjutning på 0 utvärderar uttrycket i den aktuella kolumnen. Anges ett negativt värde för startpunkten gör det att **Before**-funktionen fungerar likadant som **After**-funktionen med motsvarande positivt värde för startpunkten.
count	Om en tredje parameter **count** anges till större än 1 returnerar funktionen ett intervall av värden, ett för var och en av tabellkolumnerna upp till värdet för **count** räknat åt vänster från den ursprungliga cellen.
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

I radsegmentets första kolumn kommer värdet NULL att returneras eftersom det inte finns någon kolumn före denna.

Om pivottabellen har flera horisontella dimensioner, inbegriper det aktuella radsegmentet endast kolumner som har samma värden som den aktuella kolumnen i samtliga dimensionsrader utom den rad som visar den sista dimensionen i fältens inbördes sorteringsordning. Fältens inbördes sorteringsordning för horisontella dimensioner i pivottabeller definieras helt enkelt som dimensionernas ordning uppifrån och ned.

Exempel: diagramuttryck

Exempel	Resultat
`Before( Sum (Sales ))`	Returnerar summan av `Sales` från föregående kolumn omedelbart till vänster i förhållande till aktuell kolumn.
`Before( Sum (Sales ), 2)`	Returnerar summan av `Sales` från den kolumn som ligger två kolumner till vänster om aktuell kolumn.
`Before( Total Sum( Sales ))`	Returnerar den totala summan av `Sales` från föregående kolumn omedelbart till vänster i förhållande till aktuell kolumn.
`RangeAvg ( Before(Sum (x),1,3))`	Returnerar medelvärdet för de tre resultaten av `sum(x)`-funktionen utvärderad på de tre kolumnerna omedelbart till vänster om den aktuella kolumnen.

Exempel – grunderna i Before

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Year`
  - `Quarter`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Year, Quarter, Sales
2023, Q1, 5000
2023, Q2, 4000
2023, Q3, 6000
2023, Q4, 7000
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en raddimension:

- `Year`

Lägg till detta fält som en kolumndimension:

- `Quarter`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.
- `=Before(Sum(Sales))`, för att visa summan av försäljningen under föregående kvartal.

Resultattabell

År	Q1		Q2		Q3		Q4	
	Sum (Sales)	Before (Sum (Sales))	Sum (Sales)	Before (Sum (Sales))	Sum (Sales)	Before (Sum (Sales))	Sum (Sales)	Before (Sum (Sales))
2023	5000	-	4000	5000	6000	4000	7000	6000

När du tittar på resultaten kan du se hur funktionen `Before` returnerar värdet för föregående kolumn som ligger omedelbart till vänster om kolumnen `Sum(Sales)` för varje `Quarter`. Under Q4 är t.ex. `Sum(Sales)`-värdet för den perioden `7000`. Utdata för funktionen `Before` för `Q4` är `6000`, vilket är värdet för kolumnen direkt till vänster om `Sum(Sales)` för `Q4`, nämligen `Sum(Sales)` för `Q3`.

Exempel – Använda Before-funktionen för att beräkna försäljningstillväxten på årsbasis

Diagramuttryck

Översikt

En datauppsättning innehåller försäljningssiffror för flera år. Företaget vill presentera den totala försäljningen för varje år och försäljningstillväxten baserat på värdet från föregående år.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Year`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Year, Sales
2021, 1300
2022, 1800
2023, 2000
2024, 2100
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en kolumndimension:

- `Year`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.
- `=(Sum(Sales) - Before(Sum(Sales))) / Before(Sum(Sales))`, för att beräkna den årliga försäljningstillväxten. Om du vill visa värdet i procent väljer du **Nummer > Enkel formatering > 12,34%** under **Nummerformatering**.

Resultattabell

2021		2022		2023		2024	
Sum (Sales)	(Sum(Sales) - Before(Sum (Sales)))/Before(Sum (Sales))	Sum (Sales)	(Sum(Sales) - Before(Sum (Sales)))/Before(Sum (Sales))	Sum (Sales)	(Sum(Sales) - Before(Sum (Sales)))/Before(Sum (Sales))	Sum (Sales)	(Sum(Sales) - Before(Sum (Sales)))/Before(Sum (Sales))
1300	-	1800	38.46%	2000	11.11%	2100	5.00%

Utdata från funktionen `Before` visar tillväxten från år till år utifrån föregående års försäljningssiffror. Till exempel var försäljningssumman för `2022` `38.46%` högre än `Sum(Sales)` för `2021`.

Du kan också använda funktionen `After` för att göra samma beräkning av försäljningstillväxten i det här scenariot. När du anger ett negativt värde för startpunkten i funktionsparametrarna gör det att `After`-funktionen fungerar likadant som `Before`-funktionen med motsvarande positivt värde för startpunkten. Till exempel ger det följande uttrycket med funktionen `After` (med en negativ startpunkt på `-1`) samma resultat för årlig försäljningstillväxt som funktionen `Before`: `=(Sum(Sales)-After(Sum(Sales),-1))/(After(Sum(Sales),-1))`.

First - diagramfunktion

First() returnerar värdet för ett uttryck utvärderat med en pivottabells dimensionsvärden som de visas i den första kolumnen i det aktuella radsegmentet i pivottabellen. Funktionen returnerar NULL i alla diagramtyper utom pivottabeller.

> *Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen.*

Syntax:

```
First([TOTAL] expr [, offset [, count]])
```

Argument

Argument	Beskrivning
expression	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Om en **offset** n större än 1 anges, flyttas utvärderingen av uttrycket n kolumner till höger om första kolumnen. En förskjutning på 0 utvärderar uttrycket i den första kolumnen i aktuellt radsegment. Om startpunkt anges till 0 utvärderas uttrycket på den aktuella raden. Anges ett negativt värde för startpunkten gör det att **First**-funktionen fungerar likadant som **Last**-funktionen med motsvarande positivt värde för startpunkten.
count	Om en tredje parameter **count** anges till större än 1 returnerar funktionen ett intervall av värden, ett för var och en av tabellkolumnerna upp till värdet för **count** räknat åt höger från den ursprungliga cellen.
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

Om pivottabellen har flera horisontella dimensioner, inbegriper det aktuella radsegmentet endast kolumner som har samma värden som den aktuella kolumnen i samtliga dimensionsrader utom den rad som visar den sista dimensionen i fältens inbördes sorteringsordning. Fältens inbördes sorteringsordning för horisontella dimensioner i pivottabeller definieras helt enkelt som dimensionernas ordning uppifrån och ned.

Exempel: diagramuttryck

Exempel	Resultat
`First( Sum (Sales ))`	Returnerar summan av `Sales` från den första kolumnen i det aktuella radsegmentet.

Exempel	Resultat
`First( Sum (Sales ), 2)`	Returnerar summan av `Sales` som ligger två kolumner till höger om den första kolumn i aktuellt radsegment.
`First( Total Sum( Sales ))`	Returnerar totalsumman av `Sales` från den första kolumnen i det aktuella radsegmentet.
`RangeAvg (First (Sum(x),1,5))`	Returnerar ett medelvärde av resultaten från funktionen `sum(x)` utvärderat på de fem kolumnerna längst till vänster i det aktuella radsegmentet.

Exempel – grunderna i First

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Year`
  - `Quarter`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Year, Quarter, Sales
2023, Q1, 5000
2023, Q2, 4000
2023, Q3, 6000
2023, Q4, 7000
2024, Q1, 4500
2024, Q2, 3250
2024, Q3, 6500
2024, Q4, 7250
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en raddimension:

- `Year`

Lägg till detta fält som en kolumndimension:

- `Quarter`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.
- `=First(Sum(Sales))`, för att beräkna försäljningssumman från den första kolumnen i det aktuella radsegmentet.

Resultattabell

År	Q1		Q2		Q3		Q4	
	Sum (Sales)	Första (Summa (Försäljning))	Sum (Sales)	Första (Summa (Försäljning))	Sum (Sales)	Första (Summa (Försäljning))	Sum (Sales)	Första (Summa (Försäljning))
2023	5000	5000	4000	5000	6000	5000	7000	5000
2024	4500	4500	3250	4500	6500	4500	7250	4500

När du tittar på resultatet kan du se hur funktionen `First` returnerar summan av försäljningsvärdet för den första kolumnen i varje radsegment för varje kvartal. Till exempel, i `Q4` är `Sum(Sales)`-värdet för den perioden `7000`. Värdet `First(Sum(Sales)` för `Q4` är `5000`, vilket är värdet för `Sum(Sales)` i `Q1`, det första kolumnvärdet på den aktuella raden, `2023`.

Exempel – First scenario

Diagramuttryck

Översikt

En datauppsättning innehåller försäljningssiffror per år för olika produkter under en treårsperiod. Företaget vill presentera den totala försäljningen för innevarande år och försäljningstillväxten sedan det första året i perioden.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Date`
  - `Product`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Date, Product, Sales
```

```
2021, Apple, 100
2021, Banana, 150
2021, Carrot, 120
2022, Carrot, 80
2022, Banana, 200
2022, Apple, 110
2023, Carrot, 160
2023, Banana, 300
2023, Apple, 150
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en raddimension:

- `Product`

Lägg till detta fält som en kolumndimension:

- `Date`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna den årliga försäljningen.
- `=(Sum(Sales) - First(Sales)) / First(Sales)`för att beräkna försäljningstillväxten sedan det första året i datauppsättningen. Om du vill visa värdet i procent väljer du **Nummer > Enkel formatering > 12,34%** under **Nummerformatering**.

Resultattabell

Produkt	2021		2022		2023	
	Sum (Sales)	(Summa (Försäljning) - Första (Försäljning)) / Första (Försäljning)	Sum (Sales)	(Sum (Försäljning) - First (Försäljning)) / First (Försäljning)	Sum (Sales)	(Sum (Försäljning) - First (Försäljning)) / First (Försäljning)
Apple	100	0,00 %	110	10,00 %	150	50.00%
Banan	150	0.00%	200	33.33%	300	100.00%
Morot	120	0,00%	80	-33,33%	160	33,33%

Resultatet visar de årliga försäljningssiffrorna för varje år och tillväxttakten per år i procent jämfört med försäljningsvärdena från det första året. Till exempel ökade försäljningen av `Apple` från ett initialt värde på `100` under `2021` till ett värde på `150` under `2023`, vilket är en ökning med `50%`. Försäljningen av `Carrot` minskade från `120` under `2021` till `80` under `2022`, en minskning med `33.33%`.

Exempel – Första scenariot med en intervallsumma för att presentera budgetsiffror

Diagramuttryck

Översikt

En datauppsättning innehåller budgetsiffror per kvartal för en tvåårsperiod per produktkategori. Företaget vill presentera budgeten för innevarande kvartal och den löpande summan av budgeten för varje år.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `BudgetYear`
  - `BudgetQuarter`
  - `QuarterNo`
  - `BudgetAmount`
  - `Category`

Laddningsskript

```
Example:
Load * inline [
BudgetYear, BudgetQuarter, QuarterNo, BudgetAmount, Category
2023, Q1, 1, 2100, Online
2023, Q2, 2, 2500, Online
2023, Q3, 3, 3100, Online
2023, Q4, 4, 4100, Online
2023, Q1, 1, 5100, Store
2023, Q2, 2, 5500, Store
2023, Q3, 3, 6100, Store
2023, Q4, 4, 7100, Store
2024, Q1, 1, 2300, Online
2024, Q2, 2, 2800, Online
2024, Q3, 3, 3300, Online
2024, Q4, 4, 4400, Online
2024, Q1, 1, 5300, Store
2024, Q2, 2, 5800, Store
2024, Q3, 3, 6300, Store
2024, Q4, 4, 7400, Store
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till dessa fält som raddimensioner:

- `BudgetYear`
- `Category`

Lägg till detta fält som en kolumndimension:

- `BudgetQuarter`

Skapa följande mått:

- `=Sum(BudgetAmount)`, för att beräkna budgetförsäljningssiffran.
- `=RangeSum(First(Sum(BudgetAmount), 1, QuarterNo))`, för att beräkna den löpande summan av budgetförsäljningen för varje kvartal genom att summera intervallet från det första kvartalet till `QuarterNo` som utvärderas för varje kvartal.

Resultattabell

Budgetår Kategori	Q1		Q2		Q3		Q4	
	Summa (Budgetbelopp)	Intervallsumma (Första (Summa (Budgetbelopp), 1, Kvartalsnummer))	Sum (Budget Amount)	RangeSum(First (Sum (Budget Amount), 1, QuarterNo))	Sum (Budget Amount)	RangeSum(First (Sum (Budget Amount), 1, QuarterNo))	Sum (Budget Amount)	Intervall Summa (Första (Summa (Budgetbelopp), 1, Kvartalsnr))
(-) 2023	7200	7200	8000	15200	9200	24400	11200	35600
Online	2100	2100	2500	4600	3100	7700	4100	11800
Lagra	5100	5100	5500	10600	6100	16700	7100	23800
(-) 2024	7600	7600	8600	16200	9600	25800	11800	37600
Online	2300	2300	2800	5100	3300	8400	4400	12800
Lagra	5300	5300	5800	11100	6300	17400	7400	24800

Resultatet visar budgetsiffrorna för varje `Year` och `Category`, och den löpande totalsumman för `Quarter`. Till exempel, i `Q2 2023`, är `QuarterNo 2`. Därför utvärderas funktionsuttrycket som `RangeSum(First(Sum(BudgetAmount), 1, 2))`, som summerar `BudgetAmount`-kolumnerna för `Q1` (`7200`) och `Q2` (`8000`) och returnerar värdet `15200`. Resultatet visar summan av kolumnerna `Sum(BudgetAmount)` för `Q1` och `Q2`. När det gäller `Category Online` under `2024`, är `Sum(BudgetAmount)` för `Q1 2300` och för `Q2` är det `2800`. Därför returnerar `RangeSum`-uttrycket `5100`.

Last - diagramfunktion

Last() returnerar värdet för ett uttryck utvärderat med en pivottabells dimensionsvärden som de visas i den sista kolumnen i det aktuella radsegmentet i pivottabellen. Funktionen returnerar NULL i alla diagramtyper utom pivottabeller.

> *Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen.*

Syntax:

```
Last([TOTAL] expr [, offset [, count]])
```

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Om en **offset** n större än 1 anges, flyttas utvärderingen av uttrycket n kolumner till vänster om den sista kolumnen. En förskjutning på 0 utvärderar uttrycket i den sista kolumnen i aktuellt radsegment. Anges ett negativt värde för startpunkten gör det att **Last**-funktionen fungerar likadant som **First**-funktionen med motsvarande positivt värde för startpunkten.
count	Om en tredje parameter **count** anges till större än 1 returnerar funktionen ett intervall av värden, ett för var och en av tabellkolumnerna upp till värdet för **count** räknat åt vänster från den ursprungliga cellen.
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

Om pivottabellen har flera horisontella dimensioner, inbegriper det aktuella radsegmentet endast kolumner som har samma värden som den aktuella kolumnen i samtliga dimensionsrader utom den rad som visar den sista dimensionen i fältens inbördes sorteringsordning. Fältens inbördes sorteringsordning för horisontella dimensioner i pivottabeller definieras helt enkelt som dimensionernas ordning uppifrån och ned.

Exempel: diagramuttryck

Exempel	Resultat
`Last( Sum(Sales ))`	Returnerar summan av `Sales` från den sista kolumnen i det aktuella radsegmentet.
`Last( Sum(Sales ), 2)`	Returnerar summan av `Sales` från den kolumn som ligger två kolumner till vänster om den sista kolumnen i aktuellt radsegment.

Exempel	Resultat
`Last( Total Sum ( Sales ))`	Returnerar totalsumman av `Sales` från den sista kolumnen i det aktuella radsegmentet.
`RangeAvg (Last (Sum(x),1,5))`	Returnerar ett medelvärde av resultaten av funktionen `sum(x)` utvärderad på de fem kolumnerna längst till höger i det aktuella radsegmentet.

Exempel – grunderna i Last

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Year`
  - `Quarter`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Year, Quarter, Sales
2023, Q1, 5000
2023, Q2, 4000
2023, Q3, 6000
2023, Q4, 7000
2024, Q1, 4500
2024, Q2, 3250
2024, Q3, 6500
2024, Q4, 7250
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en raddimension:

- `Year`

Lägg till detta fält som en kolumndimension:

- `Quarter`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.
- `=Last(Sum(Sales))`, för att beräkna försäljningssumman från den sista kolumnen i det aktuella radsegmentet.

Resultattabell

År	Q1		Q2		Q3		Q4	
	Sum (Sales)	Last (Sum (Sales))	Sum (Sales)	Last (Sum (Sales))	Sum (Sales)	Last (Sum (Sales))	Sum (Sales)	Last (Sum (Sales))
2023	5000	7000	4000	7000	6000	7000	7000	7000
2024	4500	7250	3250	7250	6500	7250	7250	7250

När du tittar på resultatet kan du se hur funktionen `Last` returnerar summan av försäljningsvärdet för den sista kolumnen i varje radsegment för varje kvartal. Till exempel, i `Q1` är `Sum(Sales)`-värdet för den perioden `5000`. Värdet `Last(Sum(Sales)` för `Q1` är `7000`, vilket är värdet för `Sum(Sales)` i `Q4`, det sista kolumnvärdet på den aktuella raden, `2023`.

Exempel – Last scenario

Diagramuttryck

Översikt

En datauppsättning innehåller försäljningssiffror för olika produkter under en tremånadersperiod. Företaget vill presentera den totala försäljningen för innevarande period och försäljningstillväxten jämfört med periodens sista (eller senaste) försäljningssiffror.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Date`
  - `Product`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Date, Product, Sales
2024-01-01, Apple, 100
2024-01-01, Banana, 150
2024-01-01, Carrot, 120
2024-02-01, Carrot, 80
2024-02-01, Banana, 200
```

```
2024-02-01, Apple, 110
2024-03-01, Carrot, 160
2024-03-01, Banana, 300
2024-03-01, Apple, 150
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en raddimension:

- `Product`

Lägg till detta fält som en kolumndimension:

- `Date`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna den årliga försäljningen.
- `=(Last(Sum(Sales)) - Sum(Sales)) / Sum(Sales)`, för att beräkna försäljningstillväxten från den aktuella perioden till den senaste perioden. Om du vill visa värdet i procent väljer du **Nummer > Enkel formatering > 12,34%** under **Nummerformatering**.

Resultattabell

Produkt	2024-01-01		2024-02-01		2024-03-01	
	Sum (Sales)	(Last(Sum (Sales)) - Sum (Sales)) / Sum (Sales)	Sum (Sales)	(Last(Sum (Sales)) - Sum (Sales)) / Sum (Sales)	Sum (Sales)	(Last(Sum (Sales)) - Sum(Sales)) / Sum(Sales)
Äpple	100	50,00%	110	36,36%	150	0,00%
Banan	150	100,00%	200	50,00%	300	0,00%
Morot	120	33,33%	80	100,00%	160	0,00%

I det här exemplet kan du se hur försäljningen har utvecklats från början av försäljningsperioden till den senaste perioden. Under den första perioden `2024-01-01` var t.ex. försäljningen för `Apple` `100` jämfört med de senaste försäljningssiffrorna på `150` under perioden `2024-03-01`. Detta indikerar en ökning med `50%`.

Exempel – Använda Last för att beräkna budgetavvikelse

Diagramuttryck

Översikt

En datauppsättning innehåller budgetsiffror per kvartal över en tvåårsperiod. Företaget vill presentera det totala budgetbeloppet för varje kvartal och avvikelsen mellan budgetbeloppet för den aktuella perioden jämfört med den genomsnittliga kvartalsbudgeten för året.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `Year`
    - `Quarter`
    - `Budget`

Laddningsskript

```
Example:
Load * inline [
Year, Quarter, Budget
2023, Q1, 1200
2023, Q2, 1100
2023, Q3, 1300
2023, Q4, 1400
2024, Q1, 1550
2024, Q2, 1230
2024, Q3, 1400
2024, Q4, 1600
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en raddimension:

- `Year`

Lägg till detta fält som en kolumndimension:

- `Quarter`

Skapa följande mått:

- `=Sum(Budget)`, för att beräkna budgeten för kvartalet.
- `=Sum(Budget) - RangeAvg(Last(Budget, 1, 4))`, för att beräkna avvikelsen mellan `Budget`-beloppet för den aktuella perioden och den genomsnittliga kvartalsbudgeten för året.

Resultattabell

År	Q1		Q2		Q3		Q4	
	Sum (Budget)	Sum (Budget) - RangeAvg(Last (Budget, 1, 4))	Sum (Budget)	Sum (Budget) - RangeAvg(Last (Budget, 1, 4))	Sum (Budget)	Sum (Budget) - RangeAvg(Last (Budget, 1, 4))	Sum (Budget)	Sum (Budget) - RangeAvg(Last (Budget, 1, 4))
2023	1200	-50	1100	-150	1300	50	1400	150
2024	1550	105	1230	-215	1400	-45	1600	155

Med hjälp av funktionen `Last` kan du se hur budgetbeloppet per kvartal varierar jämfört med den genomsnittliga kvartalsbudgeten. Under `2023` är t.ex. den totala budgeten för året `5000` (`1200 + 1100 + 1300 + 1400`), och därför är den genomsnittliga kvartalsbudgeten `1250`. Eftersom budgetbeloppet för `Q1` är `1200` är det `50` lägre jämfört med den genomsnittliga kvartalsbudgeten på `1250`, medan budgetavvikelsen för `Q4` är `150` över den genomsnittliga kvartalsbudgeten.

ColumnNo - diagramfunktion

ColumnNo() returnerar numret på den aktuella kolumnen inom det aktuella radsegmentet i en pivottabell. Första kolumnen är nummer 1.

Syntax:

```
ColumnNo([total])
```

Argument

Argument	Beskrivning
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

Om pivottabellen har flera horisontella dimensioner, inbegriper det aktuella radsegmentet endast kolumner som har samma värden som den aktuella kolumnen i samtliga dimensionsrader utom den rad som visar den sista dimensionen i fältens inbördes sorteringsordning. Fältens inbördes sorteringsordning för horisontella dimensioner i pivottabeller definieras helt enkelt som dimensionernas ordning uppifrån och ned.

Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen.

Exempel: diagramuttryck

Exempel	Resultat
`IF( ColumnNo()=1, 0, Sum(Sales) / Before(Sum (Sales)))`	Om `ColumnNo` är lika med 1 kommer diagrammet att visa noll. I annat fall returneras resultatet av `Sum(Sales) / Before(Sum(Sales)`. I det här exemplet kommer diagrammet att visa värden från kolumn 2 och framåt, medan kolumn 1 visar noll.

Exempel – grunderna i ColumnNo

Diagramuttryck

Översikt

En datauppsättning innehåller kvartalsvisa försäljningsdata. I detta exempel multipliceras försäljningsvärdet för varje period med periodens kolumnnummer.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Period`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Period, Sales
Q1, 1000
Q2, 2000
Q3, 3000
Q4, 4000
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en kolumndimension:

- `Period`

Skapa följande mått:

- `=Sum(Sales) * ColumnNo()`, för att beräkna summan av försäljningsvärdena multiplicerat med kolumnnumret, vilket är kvartalet i det här exemplet.

Resultattabell

Q1	Q2	Q3	Q4
1000	4000	9000	16000

Summan av försäljningsvärdet för varje kvartal multipliceras med utdata från funktionen `ColumnNo`. Till exempel är värdet på försäljningen i `Q4` 4000, och utdata från funktionen `ColumnNo` returnerar `4` eftersom `Q4` är den fjärde kolumnen. Därför multiplicerar måttuttrycket `4000` med `4` och returnerar `16000`.

Exempel – ColumnNo-scenario

Diagramuttryck

Översikt

En datauppsättning innehåller kvartalsvisa försäljningssiffror. Ett företag vill öka försäljningsdata för Q1 och Q3 (de udda kolumnerna) med 5 % och öka försäljningen för de återstående kolumnerna (Q2 och Q4) med 10 %.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Period`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Period, Sales
Q1, 1000
Q2, 2000
Q3, 3000
Q4, 4000
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en kolumndimension:

- `Period`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.
- `=If(Odd(ColumnNo()), Amount * 1.05, Amount * 1.10)`, att öka försäljningsbeloppen för de udda kolumnerna med 5 % och för de övriga kolumnerna med 10 %.

Resultattabell

Q1		Q2		Q3		Q4	
Sum (Sales)	Om(Udda (KolumnNr ()), Belopp * 1.05, Belopp * 1.10)	Sum (Sales)	Om(Udda (KolumnNr ()), Belopp * 1.05, Belopp * 1.10)	Sum (Sales)	Om(Udda (KolumnNr ()), Belopp * 1.05, Belopp * 1.10)	Sum (Sales)	Om(Udda (KolumnNr ()), Belopp * 1.05, Belopp * 1.10)
1000	1050	2000	2200	3000	3150	4000	4400

Resultatet visar att försäljningsvärdena för de udda kolumnerna, 1 och 3 (`Q1` och `Q3`), har ökat med 5 %, medan de återstående jämna kolumnerna, 2 och 4 (`Q2` och `Q4`), har ökat med 10 %.

Exempel – ColumnNo advanced scenario

Diagramuttryck

Översikt

I det här exemplet används samma datauppsättning som i det föregående scenariot för att presentera den kumulativa summan för värdena i varje period.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Period`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Period, Sales
Q1, 1000
Q2, 2000
Q3, 3000
Q4, 4000
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en kolumndimension:

- `Period`

Skapa följande mått:

- `=RangeSum(Before(Sum(Sales), 0, ColumnNo()))`, för att beräkna den kumulativa summan.

Resultattabell

Q1	Q2	Q3	Q4
1000	3000	6000	10000

I det här exemplet visas hur du använder funktionen `ColumnNo` för att generera en löpande totalsumma. Följande förklaring ger mer information om måttuttrycket: `RangeSum(Before(Sum(Sales), 0, ColumnNo()))`.

`Sum(Sales)`: beräknar försäljningssumman för den aktuella cellen i pivottabellen.

`Before(Sum(Sales), 0, ColumnNo())`: Funktionen `Before` hämtar värden från de föregående kolumnerna i pivottabellen.

- Argumentet `Sum(Sales)` säkerställer att funktionen hämtar summerad `Sales` för de föregående cellerna.
- Offset-värdet på `0` anger kolumnens offset (återstående i det aktuella radsegmentet i den aktuella kolumnen).
- `ColumnNo()` fastställer hur många kolumner tillbaka som ska ingå när data hämtas. Den representerar dynamiskt positionen för den aktuella kolumnen (för kolumn 3 är t.ex. `ColumnNo()` lika med `3`, så funktionen tittar tillbaka över tre kolumner).

`RangeSum()`: Funktionen `RangeSum` beräknar summan av det angivna intervallet av värden. Den hanterar null-värden genom att behandla dem som noll. I detta uttryck genererar `RangeSum` effektivt en kumulativ total genom att summera alla föregående `Sales`-värden som hämtats av funktionen `Before()`, samt försäljningen i den aktuella cellen.

NoOfColumns - diagramfunktion

NoOfColumns() returnerar antalet kolumner i det aktuella radsegmentet i en pivottabell.

> *Sortering på y-värden i diagram, eller sortering efter uttryckskolumner i tabeller, är inte tillåtet när denna diagramfunktion används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade. När du använder den här diagramfunktion i en visualisering eller tabell kommer sorteringen av visualiseringen att återgå till den sorterade inmatningen av den här funktionen.*

Syntax:

```
NoOfColumns([total])
```

Argument

Argument	Beskrivning
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

I en pivottabellen med flera horisontella dimensioner, inbegriper det aktuella radsegmentet endast kolumner som har samma värden som den aktuella kolumnen i samtliga dimensionsrader utom den rad som visar den sista dimensionen i fältens inbördes sorteringsordning. Fältens inbördes sorteringsordning för horisontella dimensioner i pivottabeller definieras helt enkelt som dimensionernas ordning uppifrån och ned.

Exempel: diagramuttryck

Exempel	Resultat
`IF(ColumnNo() > NoOfColumns() - 2, Sum(Sales), Null())`	Om `ColumnNo` är större än `NoOfColumns` minus `2`, kommer diagrammet att returnera värdet `Sum(Sales)`. Om `ColumnNo` i stället är mindre än det, kommer det att returnera `NULL`. I det här exemplet, från och med den tredje kolumnen, kommer det att returnera värden medan kolumn 1 och 2 inte kommer att returnera några värden.

Exempel – grunderna i NoOfColumn

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Period`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Period, Sales
Q1, 1000
Q2, 2000
Q3, 3000
Q4, 4000
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en kolumndimension:

- `Period`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna summan av försäljningen.
- `=NoOfColumns()`, för att beräkna antalet kolumner.

Results table

Q1		Q2		Q3		Q4	
Sum (Sales)	NoOfColumns()	Sum (Sales)	NoOfColumns()	Sum (Sales)	NoOfColumns()	Sum (Sales)	NoOfColumns()
1000	4	2000	4	3000	4	4000	4

Resultatet visar summan av försäljningen för perioderna `Q1` till `Q4`. Funktionen `NoOfColumns` returnerar ett värde på `4` eftersom det finns `4` kolumner för det aktuella radsegmentet.

Exempel – NoOfColumns-scenario

Diagramuttryck

Översikt

En datauppsättning innehåller kvartalsvisa försäljningssiffror. Du vill visa försäljningsvärden endast för de två senaste kvartalen (Q3 och Q4) i pivottabellen.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Product`
  - `Quarter`
  - `Sales`

Laddningsskript

```
Example:
Load * inline [
Product, Quarter, Sales
Apple, Q1, 100
Apple, Q2, 120
Apple, Q3, 140
Apple, Q4, 160
Banana, Q1, 150
Banana, Q2, 180
Banana, Q3, 200
Banana, Q4, 220
Cherry, Q1, 80
```

```
Cherry, Q2, 90
Cherry, Q3, 110
Cherry, Q4, 150
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny pivottabell och lägg till det här fältet som en raddimension:

- `Product`

Lägg till detta fält som en kolumndimension:

- `Quarter`

Skapa följande mått:

- `=IF(ColumnNo() > NoOfColumns() - 2, Sum(Sales), Null())`, för att visa värdena endast för de två sista kolumnerna i pivottabellen (`Q3` och `Q4`).

Results table

Product	Q1	Q2	Q3	Q4
Apple	-	-	140	160
Banana	-	-	200	220
Cherry	-	-	110	150

Resultatet visar försäljningsvärden endast för kolumn 3 och 4, de två sista kvartalen på året.

8.17 Logiska funktioner

Den här delen beskriver funktioner som hanterar logiska operationer. Alla funktioner kan användas både i dataladdningsskriptet och diagramuttryck.

IsNum

Returnerar -1 (True) om uttrycket kan tolkas som ett tal, annars 0 (False).

```
IsNum( expr )
```

IsText

Returnerar -1 (True) om uttrycket består av text, annars 0 (False).

```
IsText( expr )
```

*Både **IsNum** och **IsText** returnerar 0 om uttrycket är NULL.*

Exempel:

I det följande exemplet laddas en inline-tabell med blandade textvärden och numeriska värden, och två fält läggs till för att kontrollera om värdet är ett numeriskt värde eller ett textvärde.

```
Load *, IsNum(Value), IsText(Value)
Inline [
Value
23
Green
Blue
12
33Red];
```

Den resulterande tabellen ser ut så här:

Resulting table

Value	IsNum(Value)	IsText(Value)
23	-1	0
Green	0	-1
Blue	0	-1
12	-1	0
33Red	0	-1

8.18 Mappningsfunktioner

Den här delen beskriver funktioner som hanterar mappningstabeller. En mappningstabell kan användas för att ersätta fältvärden eller fältnamn under skriptexekveringen.

Mappningsfunktioner kan endast användas i dataladdningsskriptet.

Mappningsfunktioner – en översikt

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

ApplyMap

Skriptfunktionen **ApplyMap** används för att mappa utdata för ett uttryck mot en tidigare inläst mappningstabell.

```
ApplyMap ('mapname', expr [ , defaultexpr ] )
```

MapSubstring

Skriptfunktionen **MapSubstring** används för att mappa delar av ett valfritt uttryck mot en tidigare inläst mappningstabell. Mappningen är skiftlägeskänslig och icke-iterativ och delsträngar mappas från vänster till höger.

```
MapSubstring ('mapname', expr)
```

ApplyMap

Skriptfunktionen **ApplyMap** används för att mappa utdata för ett uttryck mot en tidigare inläst mappningstabell.

Syntax:

```
ApplyMap('map_name', expression [ , default_mapping ] )
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
map_name	Namnet på en mappningstabell som har skapats tidigare med hjälp av satserna **mapping load** eller **mapping select**. Dess namn måste stå inom enkla, raka citationstecken. *Om du använder den här funktionen i en makroexpanderad variabel och hänvisar till en kartläggningstabell som inte finns misslyckas funktionen och fältet skapas inte.*
expression	Uttrycket vars resultat ska mappas.
default_ mapping	Om detta anges ska det här värdet användas som ett standardvärde om mappningstabellen inte innehåller något värde som matchar expression. Om inget anges returneras värdet av expression som det är.

Utdatafältet för ApplyMap ska inte ha samma namn som ett av dess indatafält. Det kan leda till oväntade resultat. Exempel som inte ska användas: `ApplyMap('Map', A) as A.`

Exempel:

I det här exemplet laddar vi en lista med säljare med en landskod som står för det land där de är bosatta. Vi använder en tabell som mappar en landskod till ett land för att ersätta landskoden med landets namn. Enbart tre länder är definierade i mappningstabellen, övriga landskoder mappas till 'Rest of the world'.

```
// Load mapping table of country codes:
map1:
mapping LOAD *
```

```
Inline [
CCode, Country
Sw, Sweden
Dk, Denmark
No, Norway
] ;

// Load list of salesmen, mapping country code to country
// If the country code is not in the mapping table, put Rest of the world
Salespersons:
LOAD *,
ApplyMap('map1', CCode,'Rest of the world') As Country
Inline [
CCode, Salesperson
Sw, John
Sw, Mary
Sw, Per
Dk, Preben
Dk, Olle
No, Ole
Sf, Risttu
] ;

// We don't need the CCode anymore
Drop Field 'CCode';
```

Den resulterade tabellen (Salespersons) ser ut så här:

Resulting table

Salesperson	Country
John	Sweden
Mary	Sweden
Per	Sweden
Preben	Denmark
Olle	Denmark
Ole	Norway
Risttu	Rest of the world

MapSubstring

Skriptfunktionen **MapSubstring** används för att mappa delar av ett valfritt uttryck mot en tidigare inläst mappningstabell. Mappningen är skiftlägeskänslig och icke-iterativ och delsträngar mappas från vänster till höger.

Syntax:

```
MapSubstring('map_name', expression)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
map_name	Namnet på en mappningstabell som tidigare lästs via en **mapping load**- eller **mapping select**-sats. Namnet måste stå inom enkla, raka citationstecken. *Om du använder den här funktionen i en makroexpanderad variabel och hänvisar till en kartläggningstabell som inte finns misslyckas funktionen och fältet skapas inte.*
expression	Uttrycket vars resultat ska mappas av delsträngar.

Exempel:

I det här exemplet laddar vi en lista med produktmodeller. Varje modell har en uppsättning attribut som beskrivs av en sammansatt kod. Genom att använda mappningstabellen med MapSubstring kan vi utöka attributkoderna till en beskrivning.

```
map2:
mapping LOAD *
Inline [
AttCode, Attribute
R, Red
Y, Yellow
B, Blue
C, Cotton
P, Polyester
S, Small
M, Medium
L, Large
] ;

Productmodels:
LOAD *,
MapSubString('map2', AttCode) as Description
Inline [
Model, AttCode
Twixie, R C S
Boomer, B P L
Raven, Y P M
Seedling, R C L
SeedlingPlus, R C L with hood
Younger, B C with patch
MultiStripe, R Y B C S/M/L
] ;
```

```
// We don't need the AttCode anymore
Drop Field 'AttCode';
```

Den resulterande tabellen ser ut så här:

Resulting table

Model	Description
Twixie	Red Cotton Small
Boomer	Blue Polyester Large
Raven	Yellow Polyester Medium
Seedling	Red Cotton Large
SeedlingPlus	Red Cotton Large with hood
Younger	Blue Cotton with patch
MultiStripe	Red Yellow Blue Cotton Small/Medium/Large

8.19 Matematiska funktioner

Den här delen beskriver funktioner för matematiska konstanter och booleska värden. De här funktionerna har inga parametrar, men parenteserna är ändå obligatoriska.

Alla funktioner kan användas både i dataladdningsskriptet och diagramuttryck.

e

Funktionen returnerar basen för naturliga logaritmer, **e** (2,71828 ...).

```
e( )
```

false

Funktionen returnerar ett dualt värde med textvärde 'False' och numeriskt värde 0. Det returnerade värdet kan användas som logiskt falskt i uttryck.

```
false( )
```

pi

Funktionen returnerar värdet av π (3,14159 ...).

```
pi( )
```

rand

Funktionen returnerar ett slumpmässigt tal mellan 0 och 1. Det kan användas för att skapa exempeldata.

```
rand( )
```

Exempel:

Det här exempelskriptet skapar en tabell med 1 000 poster med slumpvist valda versaler, det vill säga tecken i intervallet 65 till 91 (65+26).

```
Load
     Chr( Floor(rand() * 26) + 65) as UCaseChar,
     RecNo() as ID
     Autogenerate 1000;
```

true

Funktionen returnerar ett dualt värde med textvärde 'True' och numeriskt värde -1. Det returnerade värdet kan användas som logiskt sant i uttryck.

```
true( )
```

8.20 NULL-funktioner

Den här delen beskriver funktioner för att returnera eller hitta NULL-värden.

Alla funktioner kan användas både i dataladdningsskriptet och diagramuttryck.

NULL-funktioner – en översikt

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

EmptyIsNull

EmptyIsNull-funktionen konverterar tomma strängar till NULL. Den returnerar alltså NULL om parametern är en tom sträng. Annars returnerar den parametern.

```
EmptyIsNull (expr )
```

IsNull

Funktionen **IsNull** testar om värdet på ett uttryck är NULL och om det är så, returnerar -1 (True), annars 0 (False).

```
IsNull (expr )
```

Null

Null-funktionen returnerar ett null-värde.

```
NULL( )
```

EmptyIsNull

EmptyIsNull-funktionen konverterar tomma strängar till NULL. Den returnerar alltså NULL om parametern är en tom sträng. Annars returnerar den parametern.

Syntax:

```
EmptyIsNull(exp )
```

Returnerad datatyp: NULL om parametern är en tom sträng, annars returnerar den parametern

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Exempel: diagramuttryck

Exempel	Resultat
`EmptyIsNull(AdditionalComments)`	Det här uttrycket returnerar NULL för eventuella tomma strängvärden för fältet `AdditionalComments` istället för tomma strängar. Strängar som inte är tomma och tal returneras.
`EmptyIsNull(PurgeChar(PhoneNumber, ' -()'))`	Det här uttrycket tar bort alla streck, blanksteg och parenteser från fältet `PhoneNumber`. Om det inte finns några tecken kvar returnerar funktionen `EmptyIsNull` den tomma strängen som NULL. Ett tomt phone number är ju samma sak som inget phone number.

Exempel – grunderna i EmptyIsNull

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `CustomerID`
  - `OrderID`
  - `Product`
  - `Status`
  - `PhoneNumber`

Laddningsskript

```
Example:
LOAD * inline [
CustomerID, OrderID, Product, Status, PhoneNumber
1, 1001, Widget,, (0123) 456 789
2, 1002, Gizmo, Open,-
3, 1003, Gadget, Closed,()
4, 1004, Widget,,678-9888
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `CustomerID`
- `OrderID`
- `Product`
- `Status`

Skapa följande beräknade dimensioner:

- `=EmptyIsNull(Status)`, för att som NULL returnera alla tomma strängvärden i fältet `Status`.
- `=EmptyIsNull(PurgeChar(PhoneNumber, ' -()'))`, för att ta bort alla streck, blanksteg och parenteser från fältet `PhoneNumber`. Om det inte finns några tecken kvar returnerar funktionen `EmptyIsNull` den tomma strängen som NULL. Ett tomt telefonnummer är ju samma sak som inget telefonnummer.

Följande tabell innehåller tomma celler för att visa hur funktionen `EmptyIsNull` används.

Results table

CustomerID	OrderID	Product	Status	EmptyIsNull (Status)	EmptyIsNull(PurgeChar (PhoneNumber, ' -()'))
1	1001	Widget		-	0123456789
2	1002	Gizmo	Open	Open	-
3	1003	Gadget	Closed	Closed	-
4	1004	Widget		-	6789888

I kolumnen `Status` ser du att poster med tomma värden visas som tomma celler med en transparent bakgrundsfärg som standard. Utdata från åtgärden `EmptyIsNull(Status)` returnerar NULL för dessa tomma värden, vilket indikeras av streckstecknet (-) med grå cellbakgrund.

Den andra åtgärden använder funktionen `EmptyIsNull` med *PurgeChar (page 1810)* för att ta bort eventuella streck (-), blanksteg och parenteser () från fältet `PhoneNumber` och sedan returnera eventuella återstående tomma strängar som NULL. Cellerna med nullvärden visas nu med ett streck (-) och grå bakgrundsformatering.

Exempel – Ersätta nullvärden med text och lägga till villkorlig formatering

Diagramuttryck

Översikt

En försäljningschef vill analysera försäljningsdata och enkelt identifiera de poster som innehåller saknade värden.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ID`
  - `Name`
  - `Age`
  - `Email`
  - `Sales`
  - `Region`
  - `OrderStatus`

Laddningsskript

```
Example:
LOAD * inline [
ID,Name, Age, Email, Sales, Region, OrderStatus
1, John Smith, 30, john@email.com, 1000, North, Active
2, Jane Doe, '', jane@email.com, 1500, South, Active
3, Bob Johnson, 45, '', 800, East, Inactive
4, Alice Brown, 28, alice@email.com, '', West, Active
5, Charlie Lee, '', charlie@email.com, 1200, '', Active
6, Eva Green, 35, eva@email.com, 950, North, ''
7, David White, 50, '', '', South, Inactive
8, Fiona Black, 42, fiona@email.com, 1100, East, Active
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `ID`

Skapa följande beräknade dimensioner:

- `=If(IsNull(EmptyIsNull(Age)), 'NULL', EmptyIsNull(Age))` att som NULL returnera alla tomma strängvärden i fältet `Age` och sedan identifiera och fylla i alla nollvärden med värdet `NULL` i stället för ett streck (-).
- `=If(IsNull(EmptyIsNull(Email)), 'NULL', EmptyIsNull(Email))` att som NULL returnera alla tomma strängvärden i fältet `Email` och sedan identifiera och fylla i alla nollvärden med värdet `NULL` i stället för ett streck (-).
- `=If(IsNull(EmptyIsNull(Sales)), 'NULL', EmptyIsNull(Sales))` att som NULL returnera alla tomma strängvärden i fältet `Sales` och sedan identifiera och fylla i alla nollvärden med värdet `NULL` i stället för ett streck (-).
- `=If(IsNull(EmptyIsNull(Region)), 'NULL', EmptyIsNull(Region))` att som NULL returnera alla tomma strängvärden i fältet `Region` och sedan identifiera och fylla i alla nollvärden med värdet `NULL` i stället för ett streck (-).
- `=If(IsNull(EmptyIsNull(OrderStatus)), 'NULL', EmptyIsNull(OrderStatus))` att som NULL returnera alla tomma strängvärden i fältet `OrderStatus` och sedan identifiera och fylla i alla nollvärden med värdet `NULL` i stället för ett streck (-).

Ställ in villkorlig cellformatering för att ändra bakgrundsfärgen till röd för `NULL`-värden och grön för alla andra värden. I panelen **Egenskaper** för varje dimension och mätvärde anger du följande uttryck som **Bakgrundsfärguttryck**:

- `ID` (dimension): `=RGB(200,255,200)`
- `Age` (mätvärde): `=If(IsNull(EmptyIsNull(Age)), RGB(255,200,200), RGB(200,255,200))`
- `Email` (mätvärde): `=If(IsNull(EmptyIsNull(Email)), RGB(255,200,200), RGB(200,255,200))`
- `Sales` (mätvärde): `=If(IsNull(EmptyIsNull(Sales)), RGB(255,200,200), RGB(200,255,200))`
- `Region` (mätvärde): `=If(IsNull(EmptyIsNull(Region)), RGB(255,200,200), RGB(200,255,200))`
- `OrderStatus` (mätvärde): `=If(IsNull(EmptyIsNull(OrderStatus)), RGB(255,200,200), RGB(200,255,200))`

Results table

ID	If(IsNull(EmptyIsNull(Age)), 'NULL', EmptyIsNull(Age))	If(IsNull(EmptyIsNull(Email)), 'NULL', EmptyIsNull(Email))	If(IsNull(EmptyIsNull(Sales)), 'NULL', EmptyIsNull(Sales))	If(IsNull(EmptyIsNull(Region)), 'NULL', EmptyIsNull(Region))	If(IsNull(EmptyIsNull(OrderStatus)), 'NULL', EmptyIsNull(OrderStatus))
1	30	john@email.com	1000	North	Active
2	NULL	jane@email.com	1500	South	Active
3	45	NULL	800	East	Inactive
4	28	alice@email.com	NULL	West	Active
5	NULL	charlie@email.com	1200	NULL	Active
6	35	eva@email.com	950	North	NULL

ID	If(IsNull (EmptyIsNull (Age)), 'NULL', EmptyIsNull (Age))	If(IsNull (EmptyIsNull (Email)), 'NULL', EmptyIsNull(Email))	If(IsNull (EmptyIsNull (Sales)), 'NULL', EmptyIsNull (Sales))	If(IsNull (EmptyIsNull (Region)), 'NULL', EmptyIsNull (Region))	If(IsNull (EmptyIsNull (OrderStatus)), 'NULL', EmptyIsNull (OrderStatus))
7	50	NULL	NULL	South	Inactive
8	42	fiona@email.com	1100	East	Active

Resultatet visar att du enkelt kan identifiera poster med saknade värden genom att använda funktionen `EmptyIsNull` och lägga till villkorlig formatering.

IsNull

Funktionen **IsNull** testar om värdet på ett uttryck är NULL och om det är så, returnerar -1 (True), annars 0 (False).

Syntax:

```
IsNull(expr )
```

> *En sträng med längden noll inte räknas som NULL och gör att **IsNull** returnerar False.*

Returnerad datatyp: Booleansk

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Exempel: diagramuttryck

Exempel	Resultat
`IsNull(Productname)`	Returnerar `-1` (True) om `ProductName` är ett null-värde, annars `0` (False).

Exempel – grunderna i IsNull

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ID`
  - `Value`

Laddningsskript

```
Example:
LOAD * Inline
[ID, Value
0,
1,NULL
2,-
3,378];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ID`
- `Value`

Skapa följande beräknade dimensioner:

- `=If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value )`, för att returnera `NULL` om fältet `Value` inte har något värde, ett streck (-) eller värdet `NULL`, annars returneras värdet.
- `=If(IsNull(If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value )), 'T', 'F')`, för att returnera värdet `T` (True) eller `F` (False) beroende på om fältet `Value` innehåller ett null-värde eller inte.

Results table

ID	Value	If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value)	If(IsNull(If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value)), 'T', 'F')
0		-	T
1	NULL	-	T
2	-	-	T
3	378	378	F

Kolumnen `Value` visar de data som har matats in i datauppsättningen. Den första åtgärden (kolumn 3) returnerar `NULL` för de första 3 raderna vars värde uppfyllde villkoren i `IF`-satsen: `len(trim(Value))= 0 or Value='NULL' or Value='-'`. Den sista raden visar det värde som angetts, `378`.

Den andra åtgärden (kolumn 4) tar värdet av uttrycket i kolumn 3 och returnerar `T` om `NULL` eller `F` om inte `NULL`. Resultatet visar att de första 3 raderna är nollvärden.

Laddningsskript

I detta exempel laddas en inline-tabell med fyra rader, där de tre första raderna innehåller antingen ingenting, ett streck - eller texten NULL i kolumnen `Value`. Vi omvandlar dessa värden till sanna NULL-värdesåtergivningar med den mellersta som föregår **LOAD** med hjälp av **Null**-funktionen.

Första föregående **LOAD** lägger till ett fält som kontrollerar om värdet är NULL med hjälp av **IsNull**-funktionen.

```
NullsDetectedAndConverted:
LOAD *,
If(IsNull(ValueNullConv), 'T', 'F') as IsItNull;

LOAD *,
If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value ) as ValueNullConv;

LOAD * Inline
[ID, Value
0,
1,NULL
2,-
3, 378];
```

Nedan visas den resulterande tabellen. I kolumnen `ValueNullConv` representeras `NULL`-värdena av `-`.

Results table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	378	378	F

Exempel – IsNull-scenario

Diagramuttryck

Översikt

En datauppsättning med försäljningsdata analyseras för att se vilka fält som har nollvärden.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- En `NULLINTERPRET`-variabel som ställs in på `NULL`.
- Följande fält i datatabellen:
  - `SalesID`
  - `Name`

- Age
- Email
- Region
- Status

Laddningsskript

```
Set NULLINTERPRET = NULL;
Example:
LOAD * Inline [
SalesID, Name, Age, Email, Region, Status
1, John Smith, 30, john@email.com, North, Active
2, Jane Doe, NULL, jane@email.com, South, Active
3, Bob Johnson, 45, NULL, East, Inactive
4, Alice Brown, 28, alice@email.com, West, Active
5, Charlie Lee, NULL, charlie@email.com, NULL, Active
6, Eva Green, 35, eva@email.com, North, NULL
7, David White, 50, NULL, South, Inactive
8, Fiona Black, 42, fiona@email.com, East, Active
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- SalesID
- Name

Skapa följande beräknade dimensioner:

- =If(IsNull(Age), 'Age unavailable', Age), för att fylla i fältet Age med värdet Age unavailable om det är NULL.
- =If(IsNull(Email), 'Email unavailable', Email), för att fylla i fältet Email med värdet Email unavailable om det är NULL.
- =If(IsNull(Region), 'Region unavailable', Region), för att fylla i fältet Region med värdet Region unavailable om det är NULL.

Results table

SalesID	Name	If(IsNull(Age), 'Age unavailable', Age	If(IsNull(Email), 'Email unavailable', Email)	If(IsNull(Region), 'Region unavailable', Region)
1	John Smith	30	john@email.com	North
2	Jane Doe	Age unavailable	jane@email.com	South
3	Bob Johnson	45	Email unavailable	East

SalesID	Name	If(IsNull(Age), 'Age unavailable', Age	If(IsNull(Email), 'Email unavailable', Email)	If(IsNull(Region), 'Region unavailable', Region)
4	Alice Brown	28	alice@email.com	West
5	Charlie Lee	Age unavailable	charlie@email.com	Region unavailable
6	Eva Green	35	eva@email.com	North
7	David White	50	Email unavailable	South
8	Fiona Black	42	fiona@email.com	East

Resultatet av funktionen `IsNull`, som söker efter och returnerar true om nollvärden hittas, visar att flera fält i datauppsättningen innehåller nollvärden. Till exempel identifierades ett nollvärde i kolumnen `Age` för `Charlie Lee`, och utdata `Age unavailable` returnerades.

NULL

Null-funktionen returnerar ett null-värde.

Syntax:

```
Null( )
```

Returnerad datatyp: NULL

Exempel: diagramuttryck

Exempel	Resultat
`If(Product = 'A', Null(),Sum(Sales))`	Detta uttryck returnerar `NULL` för `Product A`, annars returnerar det summan av försäljningen för produkten.

Exempel – grunderna i Null

Diagramuttryck

Översikt

En enkel datauppsättning används för att demonstrera de olika användningsområdena för funktionen `Null`.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ID`
  - `Value`

Laddningsskript

```
Example:
LOAD * Inline
[ID, Value
0,
1,NULL
2,-
3,378];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ID`
- `Value`

Skapa följande beräknade dimensioner:

- `=If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value )`, för att returnera `NULL` om fältet `Value` har något av följande: inget värde, ett streck (-) eller värdet `NULL`, annars returneras värdet.
- `=If(IsNull(If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value )), 'T', 'F')`, för att returnera värdet `T` (True) eller `F` (False) beroende på om fältet `Value` innehåller ett null-värde eller inte.

Results table

ID	Value	If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value)	If(IsNull(If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value)), 'T', 'F')
0		-	T
1	NULL	-	T
2	-	-	T
3	378	378	F

Kolumnen `Value` visar de data som har matats in i datauppsättningen. Den första åtgärden (kolumn 3) använder `Null`-funktionen för att returnera `NULL` för de värden som uppfyllde villkoren i `IF`-satsen: `len(trim(Value))= 0 or Value='NULL' or Value='-'`. Till exempel uppfyllde de första 3 raderna kraven och returnerade `NULL`. Den sista raden visar det värde som angetts, `378`.

Den andra åtgärden (kolumn 4) tar värdet i kolumn 3 och returnerar `T` om `NULL` eller `F` om inte `NULL`. Resultatet visar att de första 3 raderna är nollvärden.

Laddningsskript

I detta exempel laddas en inline-tabell med fyra rader, där de tre första raderna innehåller antingen ingenting, ett streck - eller texten NULL i kolumnen `Value`. Vi omvandlar dessa värden till äkta NULL-värderepresentationer.

Den mellersta som föregår **LOAD** utför konverteringen med funktionen `Null`.

Den första föregående **LOAD** lägger till ett fält som kontrollerar om värdet är NULL, endast som illustration i detta exempel.

```
NullsDetectedAndConverted:
LOAD *,
If(IsNull(ValueNullConv), 'T', 'F') as IsItNull;

LOAD *,
If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value ) as ValueNullConv;

LOAD * Inline
[ID, Value
0,
1,NULL
2,-
3, 386];
```

Nedan visas den resulterande tabellen. I kolumnen `ValueNullConv` representeras `NULL`-värdena av `-`.

Results table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	386	386	F

Exempel – utesluta värden med hjälp av Null

Diagramuttryck

Översikt

Försäljningsdata analyseras för att fastställa den totala försäljningen och den försäljning som inte är relaterad till `Product A`.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `Product`
    - `Sales`

Laddningsskript

```
Example:
LOAD * Inline [
Product, Sales
A, 100
B, 150
C, 200
D, 250
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Product`

Skapa följande mått:

- `=Sum(Sales)`, för att beräkna den totala försäljningen.
- `=If(Product = 'A', Null(),Sum(Sales))`, för att beräkna den totala försäljningen för `Product` inte är `A`.

Results table

Product	Sum(Sales)	If(Product = 'A', Null(),Sum(Sales))
A	100	-
B	150	150
C	200	200
D	250	250

Detta exempel visar ett enkelt sätt att beräkna den totala försäljningen och sedan utesluta försäljningen för en specifik produkt med hjälp av funktionen `Null` för att ställa in värdet till `NULL`. I den andra åtgärden returnerade t.ex. alla värden för `Product` `A` `NULL` och uteslöts.

8.21 Intervallfunktioner

Intervallfunktionerna är funktioner som tar en mängd värden och producerar ett enda värde som resultat. Alla intervallfunktioner kan användas i både datainläsningsskriptet och i diagramuttryck.

I en visualisering kan till exempel en intervallfunktion beräkna ett enda värde utifrån en postöverskridande uppsättning. I dataladdningsskriptet kan en intervallfunktion beräkna ett enda värde utifrån en uppsättning värden i en intern tabell.

Intervallfunktionerna ersätter följande allmänna numeriska funktioner: ***numsum****,* ***numavg****,* ***numcount****,* ***nummin*** *och* ***nummax****, som nu bör ses som obsoleta.*

Grundläggande intervallfunktioner

RangeMax

RangeMax() returnerar de högsta numeriska värden som hittas inom uttrycket eller fältet.

```
RangeMax (first_expr[, Expression])
```

RangeMaxString

RangeMaxString() returnerar det sista värdet i textsorteringsordningen som den finner i uttrycket eller fältet.

```
RangeMaxString (first_expr[, Expression])
```

RangeMin

RangeMin() returnerar de lägsta numeriska värden som hittas inom uttrycket eller fältet.

```
RangeMin (first_expr[, Expression])
```

RangeMinString

RangeMinString() returnerar det första värdet i textsorteringsordningen som den finner i uttrycket eller fältet.

```
RangeMinString (first_expr[, Expression])
```

RangeMode

RangeMode() finner det vanligast förekommande värdet (lägesvärdet) i uttrycket eller fältet.

```
RangeMode (first_expr[, Expression])
```

RangeOnly

RangeOnly() är en dual-funktion som returnerar ett värde om uttrycket utvärderas till ett unikt värde. Om så inte är fallet returneras **NULL**.

```
RangeOnly (first_expr[, Expression])
```

RangeSum

RangeSum() returnerar summan av ett värdeintervall. Alla icke-numeriska värden behandlas som 0.

```
RangeSum (first_expr[, Expression])
```

Räkneintervallfunktioner

RangeCount

RangeCount() returnerar antalet värden, både textvärden och talvärden, i uttrycket eller fältet.

```
RangeCount (first_expr[, Expression])
```

RangeMissingCount

RangeMissingCount() returnerar antalet icke-metriska värden (inklusive NULL) i uttrycket eller fältet.

```
RangeMissingCount (first_expr[, Expression])
```

RangeNullCount

RangeNullCount() finner antalet NULL-värden i uttrycket eller fältet.

```
RangeNullCount (first_expr[, Expression])
```

RangeNumericCount

RangeNumericCount() finner antalet numeriska värden i ett uttryck eller fält.

```
RangeNumericCount (first_expr[, Expression])
```

RangeTextCount

RangeTextCount() returnerar antalet textvärden i ett uttryck eller fält.

```
RangeTextCount (first_expr[, Expression])
```

Statistiska intervallfunktioner

RangeAvg

RangeAvg() returnerar medelvärdet av ett intervall. Indata till funktionen kan vara antingen ett intervall med värden eller ett uttryck.

```
RangeAvg (first_expr[, Expression])
```

RangeCorrel

RangeCorrel() returnerar korrelationskoefficienten för två datauppsättningar. Korrelationskoefficienten är ett mått på förhållandet mellan datauppsättningarna.

```
RangeCorrel (x_values , y_values[, Expression])
```

RangeFractile

RangeFractile() returnerar det värde som motsvarar den n:te **fraktilen** (kvantilen) av ett talintervall.

```
RangeFractile (fractile, first_expr[ ,Expression])
```

RangeKurtosis

RangeKurtosis() returnerar det värde som motsvarar kurtosis hos ett talintervall.

```
RangeKurtosis (first_expr[, Expression])
```

RangeSkew

RangeSkew() returnerar det värde som motsvarar skevheten hos ett talintervall.

```
RangeSkew (first_expr[, Expression])
```

RangeStdev

RangeStdev() finner standardavvikelsen hos ett talintervall.

```
RangeStdev (expr1[, Expression])
```

Finansiella intervallfunktioner

RangeIRR

RangeIRR() returnerar internräntan för en serie kassaflöden, representerad av indatavärdena.

```
RangeIRR (value[, value][, Expression])
```

RangeNPV

RangeNPV() returnerar det aktuella nett värdet på en investering baserad på en rabattsats och en serie framtida periodiska betalningar (negativa värden) och inkomster (positiva värden). Resultatet anges i det fördefinierade talformatet **money**.

```
RangeNPV (discount_rate, value[, value][, Expression])
```

RangeXIRR

RangeXIRR() returnerar internräntan (per år) för ett schema av kassaflöden som inte nödvändigtvis är periodiska. För att beräkna intern avkastningsgrad för en serie periodiska kassaflöden används **RangeIRR**-funktionen.

```
RangeXIRR (values, dates[, Expression])
```

RangeXNPV

RangeXNPV() returnerar det aktuella nettovärdet för ett kassaflödesschema (inte nödvändigtvis periodiskt) motsvarat av parvisa tal i de uttryck som ges av **pmt** och **date**. Alla betalningar diskonteras utifrån ett 365-dagarsår.

```
RangeXNPV (discount_rate, values, dates[, Expression])
```

Se även:

p *Postöverskridande funktioner (page 1456)*

RangeAvg

RangeAvg() returnerar medelvärdet av ett intervall. Indata till funktionen kan vara antingen ett intervall med värden eller ett uttryck.

Syntax:

```
RangeAvg(first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Om inget numeriskt värde påträffas returneras NULL.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`RangeAvg (1,2,4)`	Returnerar 2.33333333
`RangeAvg (1,'xyz')`	Returnerar 1
`RangeAvg (null( ), 'abc')`	Returnerar NULL

Exempel:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
RangeTab3:

LOAD recno() as RangeID, RangeAvg(Field1,Field2,Field3) as MyRangeAvg INLINE [

Field1, Field2, Field3

10,5,6

2,3,7

8,2,8

18,11,9

5,5,9

9,4,2
```

```
];
```

Den resulterande tabellen visar de returnerade värdena för MyRangeAvg för varje post i tabellen.

Resultattabell

Områdes-ID	MyRangeAvg
1	7
2	4
3	6
4	12.666
5	6.333
6	5

Exempel med uttryck:

```
RangeAvg (Above(MyField),0,3))
```

Returnerar ett glidande genomsnitt för resultatet av intervallet av tre värden för **MyField** beräknat på den aktuella raden och de två raderna ovanför den aktuella raden. Om du anger det tredje argumentet som 3 returnerar **Above()**-funktionen tre värden, där det finns tillräckligt med rader ovanför, vilka används som indata för **RangeAvg()**-funktionen.

Data som används i exempel:

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeAvg (Above (MyField,0,3))	Comments
10	10	Eftersom det här är den översta raden består intervallet endast av ett värde.
2	6	Det finns endast en rad ovanför den här raden, så intervallet är: 10,2.
8	6.6666666667	Motsvarar RangeAvg(10,2,8)
18	9.333333333	-
5	10. 333333333	-
9	10.6666666667	-

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
```

```
18
5
9
] ;
```

Se även:

p *Avg - diagramfunktion (page 469)*
p *Count - diagramfunktion (page 409)*

RangeCorrel

RangeCorrel() returnerar korrelationskoefficienten för två datauppsättningar. Korrelationskoefficienten är ett mått på förhållandet mellan datauppsättningarna.

Syntax:

```
RangeCorrel(x_value , y_value[, Expression])
```

Returnerad datatyp: numeriska

Dataserien ska anges som (x,y)-par. Ett exempel: för att utvärdera två serier med data array 1 och array 2, där array 1 = 2,6,9 och array 2 = 3,8,4 skulle du skriva `RangeCorrel (2,3,6,8,9,4)`, vilket returnerar 0,269.

Argument:

Argument

Argument	Beskrivning
x-value, y-value	Varje värde motsvarar ett enstaka värde eller ett intervall av värden som de returneras av en postöverskridande funktion med en tredje valbar parameter. Varje värde eller värdeintervall måste svara mot ett **x-value** eller intervall av **y-values**.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

För att funktionen ska kunna beräknas krävs minst två uppsättningar koordinater.

Textvärden, NULL-värden och saknade värden returnerar NULL.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeCorrel (2,3,6,8,9,4,8,5)`	Returnerar 0,2492. Den här funktionen kan läsas in i skriptet eller läggas till i en visualisering i uttrycksredigeraren.

Exempel:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
RangeList:

Load * Inline [

ID1|x1|y1|x2|y2|x3|y3|x4|y4|x5|y5|x6|y6

01|46|60|70|13|78|20|45|65|78|12|78|22

02|65|56|22|79|12|56|45|24|32|78|55|15

03|77|68|34|91|24|68|57|36|44|90|67|27

04|57|36|44|90|67|27|57|68|47|90|80|94

] (delimiter is '|');



XY:

LOAD recno() as RangeID, * Inline [

X|Y

2|3

6|8

9|4

8|5

](delimiter is '|');
```

I en tabell med ID1 som en dimension och mått RangeCorrel(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6)) hittar funktionen **RangeCorrel()** värdet för **Correl** i intervallet med sex x,y-par för vart och ett av ID1-värdena.

Resultattabell

ID1	MyRangeCorrel
01	-0.9517
02	-0.5209
03	-0.5209
04	-0.1599

Exempel:

```
XY:

LOAD recno() as RangeID, * Inline [

X|Y

2|3

6|8

9|4

8|5

](delimiter is '|');
```

I en tabell med RangeID som en dimension och måttet: RangeCorrel(Below(X,0,4,BelowY,0,4)), använder funktionen **RangeCorrel()** resultaten av funktionerna **Below()**, vilka eftersom det tredje argumentet (count) är inställt på 4, ger ett intervall på fyra x-y-värden från den laddade tabellen XY.

Resultattabell

Områdes-ID	MyRangeCorrel2
01	0.2492
02	-0.9959
03	-1.0000
04	-

Värdet för RangeID 01 är detsamma som att manuellt ange RangeCorrel(2,3,6,8,9,4,8,5). För de övriga värdena för RangeID är den serie som produceras av Below()-funktionen: (6,8,9,4,8,5), (9,4,8,5) och (8,5), där den sista producerar ett null-resultat.

Se även:

p *Correl - diagramfunktion (page 472)*

RangeCount

RangeCount() returnerar antalet värden, både textvärden och talvärden, i uttrycket eller fältet.

Syntax:

```
RangeCount(first_expr[, Expression])
```

Returnerad datatyp: heltal

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska räknas.
Expression	Valfria uttryck eller fält som innehåller det intervall med data som ska räknas.

Begränsningar:

NULL-värden räknas ej.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeCount (1,2,4)`	Returnerar 3
`RangeCount (2,'xyz')`	Returnerar 2
`RangeCount (null( ))`	Returnerar 0
`RangeCount (2,'xyz', null())`	Returnerar 2

Exempel:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
RangeTab3:

LOAD recno() as RangeID, RangeCount(Field1,Field2,Field3) as MyRangeCount INLINE [

Field1, Field2, Field3

10,5,6

2,3,7

8,2,8

18,11,9

5,5,9
```

```
9,4,2

];
```

Den resulterande tabellen visar de returnerade värdena för MyRangeCount för varje post i tabellen.

Resultattabell

Intervall-ID	Mitt intervallantal
1	3
2	3
3	3
4	3
5	3
6	3

Exempel med uttryck:

```
RangeCount (Above(MyField,1,3))
```

Returnerar antalet värden som finns i de tre resultaten för **MyField**. Om du anger det första argumentet för **Above()**-funktionen som 1 och det andra argumentet som 3, returnerar den värdena från de första tre fälten ovanför den aktuella raden, om det finns tillräckligt med rader. De används som indata för **RangeCount()**-funktionen.

Data som används i exempel:

Exempeldata

MyField	RangeCount(Above(MyField,1,3))
10	0
2	1
8	2
18	3
5	3
9	3

Data som används i exempel:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Se även:

p *Count - diagramfunktion (page 409)*

RangeFractile

RangeFractile() returnerar det värde som motsvarar den n:te **fraktilen** (kvantilen) av ett talintervall.

> *RangeFractile() använder linjär interpolering mellan de närmaste rangordningsnumren när fraktilen beräknas.*

Syntax:

```
RangeFractile(fractile, first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
fractile	Ett tal mellan 0 och 1 som motsvarar den fraktil (kvantil uttryckt som bråkdel) som ska beräknas.
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeFractile (0.24,1,2,4,6)`	Returnerar 1.72
`RangeFractile(0.5,1,2,3,4,6)`	Returnerar 3
`RangeFractile (0.5,1,2,5,6)`	Returnerar 3.5

Exempel:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
RangeTab:
```

```
LOAD recno() as RangeID, RangeFractile(0.5,Field1,Field2,Field3) as MyRangeFrac INLINE [

Field1, Field2, Field3

10,5,6

2,3,7

8,2,8

18,11,9

5,5,9

9,4,2

];
```

Den resulterande tabellen visar de returnerade värdena för MyRangeFrac för varje post i tabellen.

Resultattabell

Områdes-ID	MyRangeFrac
1	6
2	3
3	8
4	11
5	5
6	4

Exempel med uttryck:

```
RangeFractile (0.5, Above(Sum(MyField),0,3))
```

I det här exemplet innehåller den postöverskridande funktionen **Above()** valbara parametrar för offset och count. Detta ger ett intervall av resultat som kan användas som indata för vilken som helst av intervallfunktionerna. I det här fallet returnerar `Above(Sum(MyField),0,3)` värdena för `MyField` för den aktuella raden och de två raderna ovanför. De här värdena tillhandahåller indata för funktionen **RangeFractile()**. Så, för den nedersta raden i tabellen nedan, detta motsvarar `RangeFractile(0.5, 3,4,6)`, det vill säga, beräkna 0,5-fraktilen för serien 3, 4 och 6. De första två raderna i tabellen nedan, antalet värden i intervallet minskas i enlighet därmed, där det inte finns några rader ovanför den aktuella raden. Liknande resultat fås för andra postöverskridande funktioner.

Exempeldata

MyField	RangeFractile(0.5, Above(Sum(MyField),0,3))
1	1
2	1.5
3	2
4	3
5	4
6	5

Data som används i exempel:

```
RangeTab:

LOAD * INLINE [

MyField

1

2

3

4

5

6

] ;
```

Se även:

p *Above - diagramfunktion (page 1459)*
p *Fractile - diagramfunktion (page 476)*

RangeIRR

RangeIRR() returnerar internräntan för en serie kassaflöden, representerad av indatavärdena.

Internränta är den räntesats som ges vid en investering i form av betalning (negativa värden) och inkomst (positiva värden) som infaller regelbundet.

Funktionen använder en förenklad version av Newton-metoden för att beräkna intern avkastningsgrad (IRR).

Syntax:

```
RangeIRR(value[, value][, Expression])
```

Returnerad datatyp: numeriska

Argument

Argument	Beskrivning
value	Ett enstaka värde eller ett intervall av värden som de returneras av en postöverskridande funktion med en tredje valfri parameter. För att funktionen ska kunna beräknas krävs minst ett negativt och ett positivt värde.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Textvärden, NULL-värden samt saknade värden ignoreras.

Exempeltabell

Exempel	Resultat
`RangeIRR(-70000,12000,15000,18000,21000,26000)`	Returnerar 0.0866

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.
`RangeTab3:`
`LOAD *,`
`recno() as RangeID,`
`RangeIRR(Field1,Field2,Field3) as RangeIRR;`
`LOAD * INLINE [`
`Field1|Field2|Field3`
`-10000|5000|6000`
`-2000|NULL|7000`
`-8000|'abc'|8000`
`-1800|11000|9000`
`-5000|5000|9000`
`-9000|4000|2000`
`] (delimiter is '|');`
	Den resulterande tabellen visar de returnerade värdena för RangeIRR för varje post i tabellen.
	RangeID RangeIRR
	1 0.0639
	2 0.8708
	3 -
	4 5.8419
	5 0.9318
	6 -0.2566

Se även:

p *Postöverskridande funktioner (page 1456)*

RangeKurtosis

RangeKurtosis() returnerar det värde som motsvarar kurtosis hos ett talintervall.

Syntax:

```
RangeKurtosis(first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Om inget numeriskt värde påträffas returneras NULL.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeKurtosis (1,2,4,7)`	Returnerar -0.28571428571429

Se även:

p *Kurtosis - diagramfunktion (page 484)*

RangeMax

RangeMax() returnerar de högsta numeriska värden som hittas inom uttrycket eller fältet.

Syntax:

```
RangeMax(first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Om inget numeriskt värde påträffas returneras NULL.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeMax (1,2,4)`	Returnerar 4
`RangeMax (1,'xyz')`	Returnerar 1
`RangeMax (null( ), 'abc')`	Returnerar NULL

Exempel:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
RangeTab3:

LOAD recno() as RangeID, RangeMax(Field1,Field2,Field3) as MyRangeMax INLINE [

Field1, Field2, Field3

10,5,6

2,3,7

8,2,8

18,11,9

5,5,9

9,4,2

];
```

Den resulterande tabellen visar de returnerade värdena för MyRangeMax för varje post i tabellen.

Resultattabell

Områdes-ID	MyRangeMax
1	10
2	7
3	8
4	18
5	9
6	9

Exempel med uttryck:

```
RangeMax (Above(MyField,0,3))
```

Returnerar maxvärdet inom intervallet av tre värden för **MyField**, uträknade på den aktuella raden och de två raderna ovanför den aktuella raden. Om du anger det tredje argumentet som 3 returnerar **Above()**-funktionen tre värden, där det finns tillräckligt med rader ovanför, vilka används som indata för **RangeMax ()**-funktionen.

Data som används i exempel:

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeMax (Above(Sum(MyField),1,3))
10	10
2	10
8	10
18	18
5	18
9	18

Data som används i exempel:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

RangeMaxString

RangeMaxString() returnerar det sista värdet i textsorteringsordningen som den finner i uttrycket eller fältet.

Syntax:

```
RangeMaxString(first_expr[, Expression])
```

Returnerad datatyp: sträng

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeMaxString (1,2,4)`	Returnerar 4
`RangeMaxString ('xyz','abc')`	Returnerar 'xyz'
`RangeMaxString (5,'abc')`	Returnerar 'abc'
`RangeMaxString (null( ))`	Returnerar NULL

Exempel med uttryck:

```
RangeMaxString (Above(MaxString(MyField),0,3))
```

Returnerar det sista (i textsorteringsordning) av de tre resultaten av **MaxString(MyField)**-funktionen som utvärderats på den aktuella raden och de två raderna ovanför den aktuella raden.

Data som används i exempel:

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeMaxString(Above(MaxString(MyField),0,3))
10	10
abc	abc
8	abc
def	def
xyz	xyz
9	xyz

Data som används i exempel:

```
RangeTab:

LOAD * INLINE [

MyField

10

'abc'

8

'def'

'xyz'

9

] ;
```

Se även:

p *MaxString - diagramfunktion (page 604)*

RangeMin

RangeMin() returnerar de lägsta numeriska värden som hittas inom uttrycket eller fältet.

Syntax:

```
RangeMin(first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Om inget numeriskt värde påträffas returneras NULL.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeMin (1,2,4)`	Returnerar 1
`RangeMin (1,'xyz')`	Returnerar 1
`RangeMin (null( ), 'abc')`	Returnerar NULL

Exempel:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
RangeTab3:

LOAD recno() as RangeID, RangeMin(Field1,Field2,Field3) as MyRangeMin INLINE [

Field1, Field2, Field3

10,5,6

2,3,7

8,2,8

18,11,9

5,5,9

9,4,2

];
```

Den resulterande tabellen visar de returnerade värdena för MyRangeMin för varje post i tabellen.

Resultattabell

Områdes-ID	MyRangeMin
1	5
2	2
3	2
4	9
5	5
6	2

Exempel med uttryck:

```
RangeMin (Above(MyField,0,3)
```

Returnerar minimivärdet inom intervallet av tre värden för **MyField**, uträknade på den aktuella raden och de två raderna ovanför den aktuella raden. Om du anger det tredje argumentet som 3 returnerar **Above()**-funktionen tre värden, där det finns tillräckligt med rader ovanför, vilka används som indata för **RangeMin ()**-funktionen.

Data som används i exempel:

Exempeldata

MyField	RangeMin(Above(MyField,0,3))
10	10
2	2
8	2
18	2
5	5
9	5

Data som används i exempel:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Se även:

p *Min - diagramfunktion (page 385)*

RangeMinString

RangeMinString() returnerar det första värdet i textsorteringsordningen som den finner i uttrycket eller fältet.

Syntax:

```
RangeMinString(first_expr[, Expression])
```

Returnerad datatyp: sträng

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeMinString (1,2,4)`	Returnerar 1
`RangeMinString ('xyz','abc')`	Returnerar 'abc'
`RangeMinString (5,'abc')`	Returnerar 5
`RangeMinString (null( ))`	Returnerar NULL

Exempel med uttryck:

```
RangeMinString (Above(MinString(MyField),0,3))
```

Returnerar det första (i textsorteringsordning) av de tre resultaten av **MinString(MyField)**-funktionen som utvärderats på den aktuella raden och de två raderna ovanför den aktuella raden.

Data som används i exempel:

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeMinString(Above(MinString(MyField),0,3))
10	10
abc	10
8	8
def	8
xyz	8
9	9

Data som används i exempel:

```
RangeTab:

LOAD * INLINE [

MyField

10

'abc'

8

'def'

'xyz'

9

] ;
```

Se även:

p *MinString - diagramfunktion (page 607)*

RangeMissingCount

RangeMissingCount() returnerar antalet icke-metriska värden (inklusive NULL) i uttrycket eller fältet.

Syntax:

```
RangeMissingCount(first_expr[, Expression])
```

Returnerad datatyp: heltal

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska räknas.
Expression	Valfria uttryck eller fält som innehåller det intervall med data som ska räknas.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeMissingCount (1,2,4)`	Returnerar 0
`RangeMissingCount (5,'abc')`	Returnerar 1
`RangeMissingCount (null( ))`	Returnerar 1

Exempel med uttryck:

```
RangeMissingCount (Above(MinString(MyField),0,3))
```

Returnerar antalet icke-numeriska värden i de tre resultaten av **MinString(MyField)**-funktionen som utvärderats på den aktuella raden och de två raderna ovanför den aktuella raden.

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeMissingCount (Above(MinString (MyField),0,3))	Explanation
10	2	Returnerar 2 eftersom det inte finns några rader ovanför den här raden, så två av de tre värdena saknas.
abc	2	Returnerar 2 eftersom det bara finns 1 rad ovanför den aktuella raden och den aktuella raden är icke-numerisk ("abc").

MyField	RangeMissingCount (Above(MinString (MyField),0,3))	Explanation
8	1	Returnerar 1 eftersom 1 av de 3 raderna inkluderar ett icke-numeriskt värde ("abc").
def	2	Returnerar 2 eftersom 2 av de 3 raderna inkluderar icke-numeriska värden ("def" och "abc").
xyz	2	Returnerar 2 eftersom 2 av de 3 raderna inkluderar icke-numeriska värden (" xyz" och "def").
9	2	Returnerar 2 eftersom 2 av de 3 raderna inkluderar icke-numeriska värden (" xyz" och "def").

Data som används i exempel:

```
RangeTab:

LOAD * INLINE [

MyField

10

'abc'

8

'def'

'xyz'

9

] ;
```

Se även:

p *MissingCount - diagramfunktion (page 414)*

RangeMode

RangeMode() finner det vanligast förekommande värdet (lägesvärdet) i uttrycket eller fältet.

Syntax:

```
RangeMode(first_expr {, Expression})
```

Returnerad datatyp: numeriska

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Om mer än ett värde är lika vanligt förekommande, returneras NULL.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeMode (1,2,9,2,4)`	Returnerar 2
`RangeMode ('a',4,'a',4)`	Returnerar NULL
`RangeMode (null( ))`	Returnerar NULL

Exempel:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
RangeTab3:

LOAD recno() as RangeID, RangeMode(Field1,Field2,Field3) as MyRangeMode INLINE [

Field1, Field2, Field3

10,5,6

2,3,7

8,2,8

18,11,9

5,5,9

9,4,2
```

```
];
```

Den resulterande tabellen visar de returnerade värdena för **MyRangeMode** för varje post i tabellen.

Resultattabell

Områdes-ID	MyRangMode
1	-
2	-
3	8
4	-
5	5
6	-

Exempel med uttryck:

```
RangeMode (Above(MyField,0,3))
```

Returnerar det vanligast förekommande värdet i de tre resultaten av **MyField** som utvärderats på den aktuella raden och de två raderna ovanför den aktuella raden. Om du anger det tredje argumentet som 3 returnerar **Above()**-funktionen tre värden, där det finns tillräckligt med rader ovanför, vilka används som indata för **RangeMode()**-funktionen.

Data som används i exemplet:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeMode(Above(MyField,0,3))
10	Returnerar 10 eftersom det inte fiinns några rader ovanför, så att det enskilda värdet är det vanligast förekommande.
2	-
8	-

MyField	RangeMode(Above(MyField,0,3))
18	-
5	-
9	-

Se även:

p *Mode - diagramfunktion (page 391)*

RangeNPV

RangeNPV() returnerar det aktuella nett värdet på en investering baserad på en rabattsats och en serie framtida periodiska betalningar (negativa värden) och inkomster (positiva värden). Resultatet anges i det fördefinierade talformatet **money**.

För kassaflöden som inte nödvändigtvis är periodiska, se *RangeXNPV (page 1617)*.

Syntax:

```
RangeNPV(discount_rate, value[,value][, Expression])
```

Returnerad datatyp: numeriska

Argument

Argument	Beskrivning
discount_rate	Räntesatsen per period.
value	En betalning eller inkomst som inträffar i slutet av varje period. Varje värde kan vara ett enstaka värde eller ett intervall av värden som de returneras av en postöverskridande funktion med en tredje valfri parameter.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Textvärden, NULL-värden samt saknade värden ignoreras.

Exempel	Resultat
`RangeNPV(0.1,-10000,3000,4200,6800)`	Avkastning 1188,44

Exempel	Resultat

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.
`RangeTab3:`
`LOAD *,`
`recno() as RangeID,`
`RangeNPV(Field1,Field2,Field3) as RangeNPV;`
`LOAD * INLINE [`
`Field1|Field2|Field3`
`10|5|-6000`
`2|NULL|7000`
`8|'abc'|8000`
`18|11|9000`
`5|5|9000`
`9|4|2000`
`] (delimiter is '|');`
	Den resulterande tabellen visar de returnerade värdena för RangeNPV för varje post i tabellen.
	RangeID RangeNPV
	1 $-49.13
	2 $777.78
	3 $98.77
	4 $25.51
	5 $250.83
	6 $20.40

Se även:

p *Postöverskridande funktioner (page 1456)*

RangeNullCount

RangeNullCount() finner antalet NULL-värden i uttrycket eller fältet.

Syntax:

```
RangeNullCount(first_expr [, Expression])
```

Returnerad datatyp: heltal

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeNullCount (1,2,4)`	Returnerar 0
`RangeNullCount (5,'abc')`	Returnerar 0
`RangeNullCount (null( ), null( ))`	Returnerar 2

Exempel med uttryck:

```
RangeNullCount (Above(Sum(MyField),0,3))
```

Returnerar antalet NULL-värden i de tre resultaten av **Sum(MyField)**-funktionen som utvärderats på den aktuella raden och de två raderna ovanför den aktuella raden.

*Att kopiera **MyField** i exemplet nedan resulterar inte i NULL-värde.*

Exempeldata

MyField	RangeNullCount(Above(Sum(MyField),0,3))
10	Returnerar 2 eftersom det inte finns några rader ovanför den här raden, så 2 av de 3 värdena saknas (=NULL).
”abc”	Returnerar 1 eftersom det bara finns en rad ovanför den aktuella raden, så ett av de tre värdena saknas (=NULL).
8	Returnerar 0 eftersom ingen av de tre raderna är ett NULL-värde.

Data som används i exempel:

```
RangeTab:

LOAD * INLINE [

MyField

10

'abc'

8
```

```
] ;
```

Se även:

p *NullCount - diagramfunktion (page 419)*

RangeNumericCount

RangeNumericCount() finner antalet numeriska värden i ett uttryck eller fält.

Syntax:

```
RangeNumericCount(first_expr[, Expression])
```

Returnerad datatyp: heltal

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeNumericCount (1,2,4)`	Returnerar 3
`RangeNumericCount (5,'abc')`	Returnerar 1
`RangeNumericCount (null( ))`	Returnerar 0

Exempel med uttryck:

```
RangeNumericCount (Above(MaxString(MyField),0,3))
```

Returnerar antalet numeriska värden i de tre resultaten av **MaxString(MyField)**-funktionen som utvärderats på den aktuella raden och de två raderna ovanför den aktuella raden.

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeNumericCount(Above(MaxString(MyField),0,3))
10	1
abc	1
8	2
def	1
xyz	1
9	1

Data som används i exempel:

```
RangeTab:

LOAD * INLINE [

MyField

10

'abc'

8

def

xyz

9

] ;
```

Se även:

p *NumericCount - diagramfunktion (page 424)*

RangeOnly

RangeOnly() är en dual-funktion som returnerar ett värde om uttrycket utvärderas till ett unikt värde. Om så inte är fallet returneras **NULL**.

Syntax:

```
RangeOnly(first_expr[, Expression])
```

Returnerad datatyp: dual

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Exempel och resultat:

Exempel	Resultat
`RangeOnly (1,2,4)`	Returnerar NULL
`RangeOnly (5,'abc')`	Returnerar NULL
`RangeOnly (null( ), 'abc')`	Returnerar 'abc'
`RangeOnly(10,10,10)`	Returnerar 10

Se även:

p *Only - diagramfunktion (page 395)*

RangeSkew

RangeSkew() returnerar det värde som motsvarar skevheten hos ett talintervall.

Syntax:

```
RangeSkew(first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Om inget numeriskt värde påträffas returneras NULL.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`rangeskew (1,2,4)`	Returnerar 0.93521952958283
`rangeskew (above (SalesValue,0,3))`	Returnerar en glidande skevhet av intervallet av tre värden som returnerats från funktionen above() beräknad på den aktuella raden och de två raderna ovanför den aktuella raden.

Data som används i exemplet:

Exempeldata

CustID	RangeSkew(Above(SalesValue,0,3))
1-20	-, -, 0,5676, 0,8455, 1,0127, -0,8741, 1,7243, -1,7186, 1,5518, 1,4332, 0, 1,1066, 1,3458, 1,5636, 1,5439, 0,6952, -0,3766

```
SalesTable:
LOAD recno() as CustID, * inline [
SalesValue
101
163
126
139
167
86
83
22
32
70
108
124
176
113
95
32
42
92
61
21
] ;
```

Se även:

p *Skew - diagramfunktion (page 516)*

RangeStdev

RangeStdev() finner standardavvikelsen hos ett talintervall.

Syntax:

```
RangeStdev(first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Om inget numeriskt värde påträffas returneras NULL.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeStdev (1,2,4)`	Returnerar 1.5275252316519
`RangeStdev (null( ))`	Returnerar NULL
`RangeStdev (above (SalesValue),0,3))`	Returnerar en glidande standard av intervallet av tre värden som returnerats från funktionen above() beräknad på den aktuella raden och de två raderna ovanför den aktuella raden.

Data som används i exemplet:

Exempeldata

CustID	RangeStdev(SalesValue, 0,3))
1-20	-,43,841, 34,192, 18,771, 20,953, 41,138, 47,655, 36,116, 32,716, 25,325, 38 000, 27,737, 35,553, 33,650, 42,532, 33,858, 32,146, 25,239, 35,595

```
SalesTable:
LOAD recno() as CustID, * inline [
SalesValue
101
```

```
163
126
139
167
86
83
22
32
70
108
124
176
113
95
32
42
92
61
21
] ;
```

Se även:

p *Stdev - diagramfunktion (page 519)*

RangeSum

RangeSum() returnerar summan av ett värdeintervall. Alla icke-numeriska värden behandlas som 0.

Syntax:

```
RangeSum(first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

RangeSum-funktionen behandlar alla icke-numeriska värden som 0.

Exempel och resultat:

Exempel

Exempel	Resultat
`RangeSum (1,2,4)`	Returnerar 7
`RangeSum (5,'abc')`	Returnerar 5
`RangeSum (null( ))`	Returnerar 0

Exempel:

Lägg till exempelskriptet i din applikation och kör det. När du vill se resultatet lägger du till fälten som listas i resultatkolumnen i ett ark i din applikation.

```
RangeTab3:

LOAD recno() as RangeID, Rangesum(Field1,Field2,Field3) as MyRangeSum INLINE [

Field1, Field2, Field3

10,5,6

2,3,7

8,2,8

18,11,9

5,5,9

9,4,2

];
```

Den resulterande tabellen visar de returnerade värdena för MyRangeSum för varje post i tabellen.

Resultattabell

Områdes-ID	MyRangeSum
1	21
2	12
3	18
4	38
5	19
6	15

Exempel med uttryck:

```
RangeSum (Above(MyField,0,3))
```

Returnerar summan av de tre värdena för **MyField)**: från den aktuella raden och de två raderna ovanför den aktuella raden. Om du anger det tredje argumentet som 3 returnerar **Above()**-funktionen tre värden, där det finns tillräckligt med rader ovanför, vilka används som indata för **RangeSum()**-funktionen.

Data som används i exempel:

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeSum(Above(MyField,0,3))
10	10
2	12
8	20
18	28
5	31
9	32

Data som används i exempel:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Se även:

p *Sum - diagramfunktion (page 401)*
p *Above - diagramfunktion (page 1459)*

RangeTextCount

RangeTextCount() returnerar antalet textvärden i ett uttryck eller fält.

Syntax:

```
RangeTextCount(first_expr[, Expression])
```

Returnerad datatyp: heltal

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeTextCount (1,2,4)`	Returnerar 0
`RangeTextCount (5,'abc')`	Returnerar 1
`RangeTextCount (null( ))`	Returnerar 0

Exempel med uttryck:

```
RangeTextCount (Above(MaxString(MyField),0,3))
```

Returnerar antal textvärden i de tre resultaten av **MaxString(MyField)**-funktionen som utvärderats på den aktuella raden och de två raderna ovanför den aktuella raden.

Data som används i exempel:

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	MaxString(MyField)	RangeTextCount(Above(Sum(MyField),0,3))
10	10	0
abc	abc	1
8	8	1
def	def	2
xyz	xyz	2
9	9	2

Data som används i exempel:

```
RangeTab:

LOAD * INLINE [

MyField

10

'abc'

8

null()

'xyz'

9

] ;
```

Se även:

p *TextCount - diagramfunktion (page 430)*

RangeXIRR

RangeXIRR() returnerar internräntan (per år) för ett schema av kassaflöden som inte nödvändigtvis är periodiska. För att beräkna intern avkastningsgrad för en serie periodiska kassaflöden används **RangeIRR**-funktionen.

Qliks XIRR-funktionalitet (**XIRR()**- och **RangeXIRR()**-funktionerna) använder följande ekvation för att lösa värdet `Rate` och fastställa korrekt XIRR-värde:

```
XNPV(Rate, pmt, date) = 0
```

Ekvationen löses med en förenklad version av Newton-metoden.

Syntax:

```
RangeXIRR(value, date{, value, date})
```

Returnerad datatyp: numeriska

Argument

Argument	Beskrivning
value	Ett kassaflöde eller en serie kassaflöden som motsvarar ett schema över betalningsdatum. Värdeserien måste innehålla minst ett negativt och ett positivt värde.
date	Ett betalningsdatum eller ett schema över betalningsdatum som motsvarar kassaflödesbetalningar.

När du arbetar med den här funktionen gäller följande begränsningar:

- Textvärden, NULL-värden samt saknade värden ignoreras.
- Alla betalningar diskonteras utifrån ett 365-dagarsår.
- Denna funktion kräver minst en giltig negativ och minst en giltig positiv betalning (med motsvarande giltiga datum). Om dessa betalningar inte anges returneras ett NULL-värde.

Följande ämnen kan hjälpa dig att arbeta med den här funktionen:

- *RangeXNPV (page 1617)*: använd den här funktionen för att beräkna det aktuella nettovärdet för ett schema av kassaflöden som inte nödvändigtvis är periodiskt.
- *XIRR (page 446)*: **XIRR()**-funktionen returnerar den aggregerade interna avkastningsgraden (per år) för ett schema av kassaflöden (som inte nödvändigtvis är periodiskt).

Den underliggande algoritmen som används av den här funktionen varierar mellan olika versioner av För klienthanterat Qlik Sense. Mer information om de senaste uppdateringarna av algoritmen finns i supportartikeln Korrigeringar och uppdateringar av XIRR-funktioner*.*

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`RangeXIRR(-2500,'2008-01-01',2750,'2008-09-01')`	Returnerar 0,1532

Se även:

p *RangeIRR (page 1589)*
p *RangeXNPV (page 1617)*
p *XIRR (page 446)*
≤ Korrigeringar och uppdateringar av XIRR-funktioner

RangeXNPV

RangeXNPV() returnerar det aktuella nettovärdet för ett kassaflödesschema (inte nödvändigtvis periodiskt) motsvarat av parvisa tal i de uttryck som ges av **pmt** och **date**. Alla betalningar diskonteras utifrån ett 365-dagarsår.

Syntax:

```
RangeXNPV(discount_rate, value, date{, value, date})
```

Returnerad datatyp: numeriska

Argument

Argument	Beskrivning
discount_rate	**discount_rate** är den årliga ränta som betalningarna ska diskonteras med.

Argument	Beskrivning
value	Ett kassaflöde ellen en serie kassaflöden som motsvarar ett schema över betalningsdatum. Varje värde kan vara ett enstaka värde eller ett intervall av värden som de returneras av en postöverskridande funktion med en tredje valfri parameter. Värdeserien måste innehålla minst ett negativt och ett positivt värde.
date	Ett betalningsdatum eller ett schema över betalningsdatum som motsvarar kassaflödesbetalningar.

När du arbetar med den här funktionen gäller följande begränsningar:

- Textvärden, NULL-värden samt saknade värden ignoreras.
- Alla betalningar diskonteras utifrån ett 365-dagarsår.

Exempel – skript

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Finansiella uppgifter som finns i en tabell som heter `RangeTab3`.
- Användning av funktionen **RangeXNPV()** för att beräkna aktuellt nettovärde.

Laddningsskript

```
RangeTab3:
LOAD *,
recno() as RangeID,
RangeXNPV(DiscountRate,Value1,Date1,Value2,Date2) as RangeXNPV;
LOAD * INLINE [
DiscountRate|Value1|Date1|Value2|Date2
0.1|-100|2021-01-01|100|2022-01-01|
0.1|-100|2021-01-01|110|2022-01-01|
0.1|-100|2021-01-01|125|2022-01-01|
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `RangeID`
- `RangeXNPV`

Resultattabell

Områdes-ID	OmrådesXNPV
1	-$9.09
2	-$0.00
3	$13.64

Exempel – diagramuttryck

Laddningsskript och diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Finansiella uppgifter som finns i en tabell som heter `RangeTab3`.
- Användning av funktionen **RangeXNPV()** för att beräkna aktuellt nettovärde.

Laddningsskript

```
RangeTab3:
LOAD *,
recno() as RangeID,
RangeXNPV(DiscountRate,Value1,Date1,Value2,Date2) as RangeXNPV;
LOAD * INLINE [
DiscountRate|Value1|Date1|Value2|Date2
0.1|-100|2021-01-01|100|2022-01-01|
0.1|-100|2021-01-01|110|2022-01-01|
0.1|-100|2021-01-01|125|2022-01-01|
] (delimiter is '|');
```

Resultat

Gör följande:

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till följande beräkning som ett mått.

```
=RangeXNPV(0.1, -2500,'2008-01-01',2750,'2008-09-01')
```

Resultattabell

=XIRR(betalningar, datum)
$80.25

Se även:

p *XNPV (page 452)*

8.22 Relaterade funktioner

Det här är en grupp av funktioner som beräknar egenskaper för enskilda dimensionvärden i ett diagram, med hjälp av redan aggregerade tal.

Funktioner är relationsberoende i den mening att funktionens utdata inte bara beror på värdet för själva datapunkten, utan även på värdets relation till andra datapunkter. Till exempel så kan rangordning inte beräknas utan en jämförelse med andra dimenionsvärden.

Dessa funktioner kan endast användas i diagramuttryck. De kan inte användas i laddningsskriptet.

En dimension behövs i diagrammet eftersom detta definierar de andra datapunkter som behövs för jämförelsen. Därmed är en relationsfunktion inte meningsfull i ett diagram utan dimensioner (till exempel ett KPI-objekt).

Rangordningsfunktioner

> *Alternativet Visa inte nollvärden inaktiveras automatiskt när dessa funktioner används. NULL-värden ignoreras.*

Rank

Rank() utvärderar raderna i diagrammet i uttrycket, och visar för varje rad den relativa placeringen av dimensionsvärdet utvärderat i uttrycket. Funktionen utvärderar uttrycket, jämför resultatet med resultaten på de andra raderna som innehåller det aktuella kolumnsegmentet och returnerar rangordningsnumret för den aktuella raden inom segmentet.

```
Rank - diagramfunktion([TOTAL [<fld {, fld}>]] expr[, mode[, fmt]])
```

HRank

HRank() utvärderar uttrycket och jämför resultatet med resultaten i de andra kolumnerna som innehåller det aktuella radsegmentet i en pivottabell. Funktionen returnerar sedan rangordningsnumret för den aktuella kolumnen inom segmentet.

```
HRank - diagramfunktion([TOTAL] expr[, mode[, fmt]])
```

Klustringsfunktioner

KMeans2D

Egenskapsgruppen **Platslicens** innehåller egenskaper som är kopplade till licensen för systemet Qlik Sense. Alla fält är obligatoriska och får inte vara tomma.

Egenskaper för platslicens

Egenskapsnamn	Beskrivning
Innehavarens namn	Användarnamnet för Qlik Sense-produktägaren.
Innehavarens företag	Namnet på organisationen som Qlik Sense-produktägaren ingår i.

Egenskapsnamn	Beskrivning
Serienummer	Serienumret som är tilldelat till Qlik Sense-programvaran.
Kontrollnummer	Kontrollnumret som är tilldelat till Qlik Sense-programvaran.
Åtkomst till **LEF**	License Enabler File (LEF) som är tilldelat till Qlik Sense-programvaran.

KMeans2D() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring, och för varje diagramrad visas kluster-ID för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1 respektive coordinate_2. Dessa är båda aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters. Data normaliseras med normparametern (valfritt).

```
KMeans2D - diagramfunktion(num_clusters, coordinate_1, coordinate_2 [, norm])
```

KMeansND

KMeansND() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring och för varje diagramrad visas kluster-ID för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1 och coordinate_2 osv. upp till n kolumner. Dessa är alla aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters.

```
KMeansND - diagramfunktion(num_clusters, num_iter, coordinate_1, coordinate_2
[,coordinate_3 [, ...]])
```

KMeansCentroid2D

KMeansCentroid2D() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring, och för varje diagramrad visas önskad koordinat för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1 respektive coordinate_2. Dessa är båda aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters. Data normaliseras med normparametern (valfritt).

```
KMeansCentroid2D - diagramfunktion(num_clusters, coordinate_no, coordinate_1,
coordinate_2 [, norm])
```

KMeansCentroidND

KMeansCentroidND() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring och för varje diagramrad visas önskad koordinat för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1, coordinate_2 osv. upp till n kolumner. Dessa är alla aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters.

```
KMeansCentroidND - diagramfunktion(num_clusters, num_iter, coordinate_no,
coordinate_1, coordinate_2 [,coordinate_3 [, ...]])
```

Funktioner för uppdelning av tidsserier

STL_Trend

STL_Trend är en funktion för uppdelning av tidsserier. Tillsammans med **STL_Seasonal** och **STL_Residual** används denna funktion för att dela upp en tidsserie i säsongs-, trend- och residualkomponenter. För STL-algoritmen används uppdelning i tidsserier för att identifiera både återkommande

säsongsvariationer och en allmän trend för ett givet indatamätetal och andra parametrar. **STL_Trend**-funktionen identifierar en allmän trend oberoende av säsongsmönster och cykler från tidsseriedata.

```
STL_Trend - diagramfunktion(target_measure, period_int [,seasonal_smoother
[,trend_smoother]])
```

STL_Seasonal

STL_Seasonal är en funktion för uppdelning av tidsserier. Tillsammans med **STL_Trend** och **STL_Residual** används denna funktion för att dela upp en tidsserie i säsongs-, trend- och residualkomponenter. För STL-algoritmen används uppdelning i tidsserier för att identifiera både återkommande säsongsvariationer och en allmän trend för ett givet indatamätetal och andra parametrar. **STL_Seasonal**-funktionen kan identifiera ett säsongsmönster i en tidsserie och separera denna från den allmänna trenden som visas av data.

```
STL_Seasonal - diagramfunktion(target_measure, period_int [,seasonal_smoother
[,trend_smoother]])
```

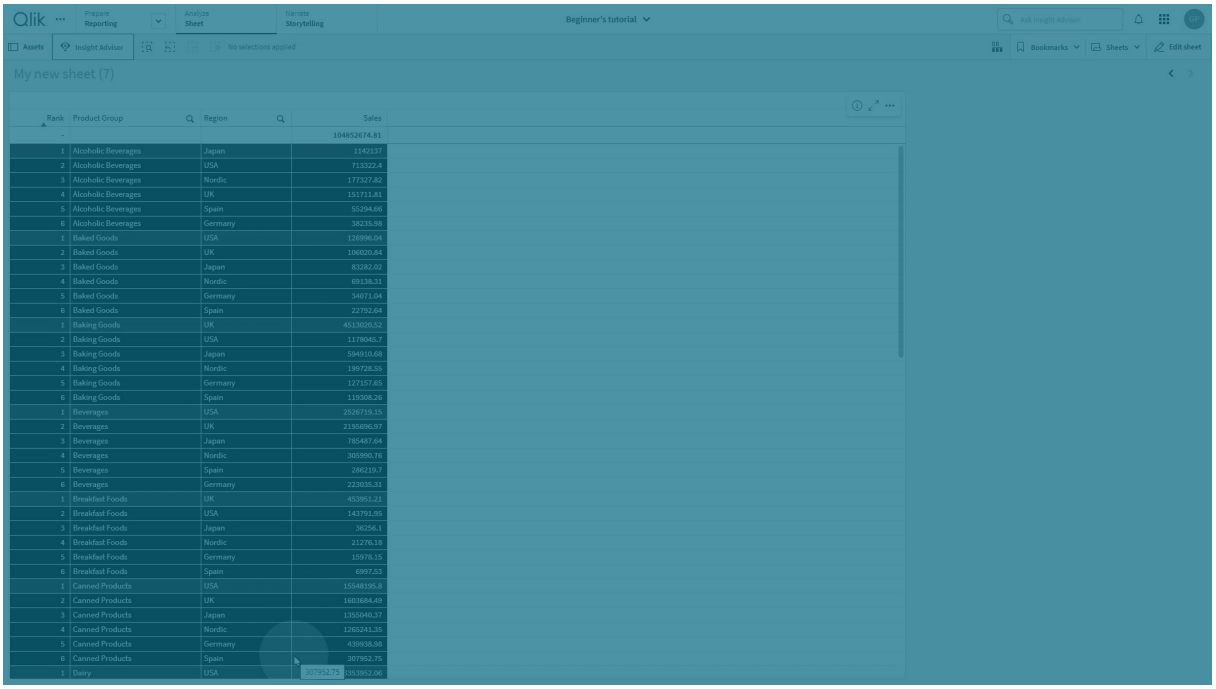
STL_Residual

STL_Residual är en funktion för uppdelning av tidsserier. Tillsammans med **STL_Seasonal** och **STL_Trend** används denna funktion för att dela upp en tidsserie i säsongs-, trend- och residualkomponenter. För STL-algoritmen används uppdelning i tidsserier för att identifiera både återkommande säsongsvariationer och en allmän trend för ett givet indatamätetal och andra parametrar. När denna åtgärd utförs kommer en del av indatamätetalen varken passa in för säsongs- eller trendkomponenten och kommer att definieras som residualkomponenten. **STL_Residual**-diagramfunktionen fångar upp denna del av beräkningen.

```
STL_Residual - diagramfunktion(target_measure, period_int [,seasonal_smoother
[,trend_smoother]])
```

Rank - diagramfunktion

Rank() utvärderar raderna i diagrammet i uttrycket, och visar för varje rad den relativa placeringen av dimensionsvärdet utvärderat i uttrycket. Funktionen utvärderar uttrycket, jämför resultatet med resultaten på de andra raderna som innehåller det aktuella kolumnsegmentet och returnerar rangordningsnumret för den aktuella raden inom segmentet.



Kolumnsegment

För andra diagram än tabeller definieras det aktuella kolumnsegmentet som det visas i diagrammets raka tabellmotsvarighet.

Syntax:

```
Rank([TOTAL] expr[, mode[, fmt]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
mode	Anger resultatets talrepresentation.
fmt	Anger resultatets textrepresentation.
TOTAL	Om diagrammet är endimensionellt eller om uttrycket föregås av kvalificeraren **TOTAL** utvärderas funktionen längs med hela kolumnen. Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

Rangordningen returneras som ett dualt värde, som i det fall där varje rad har en unik rangordning är ett heltal mellan 1 och antalet rader i det aktuella kolumnsegmentet.

Om flera rader har samma rangordningsnummer, kan text- och talrepresentationerna för gruppen kontrolleras med parametrarna **mode** och **fmt**.

mode

Det andra argumentet, **mode**, kan ta följande värden:

Exempel med **mode**

Värde	Beskrivning
0 (standard)	Om alla rangordningsnummer inom gruppen är lägre än det mellersta värdet i den totala rangordningen, får alla rader i gruppen det lägsta rangordningsnumret inom gruppen. Om alla rangordningsnummer inom gruppen är högre än det mellersta värdet i den totala rangordningen, får alla rader i gruppen det högsta rangordningsnumret inom gruppen. Om rangordningsnumren inom gruppen spänner över det mellersta värdet i den totala rangordningen, får alla rader i gruppen ett värde som motsvarar medelvärdet av det högsta och det lägsta rangordningsnumret i hela kolumnsegmentet.
1	Lägsta rangordningstalet ges åt alla rader.
2	Medelvärdet ges som rangordningstal åt alla rader.
3	Högsta rangordningstalet ges åt alla rader.
4	Lägsta rangordningstalet på första raden, därefter ökning med ett för varje rad

fmt

Det tredje argumentet, **fmt**, kan ta följande värden:

Exempel med **fmt**

Värde	Beskrivning
0 (standard)	Lägsta värdet - högsta värdet på alla rader (exempelvis 3 - 4).
1	Lägsta värdet på alla rader.
2	Lägsta värdet på första raden, tom sträng på övriga rader.

Ordningen på rader inom **mode** 4 och **fmt**2 bestäms av sorteringsordningen som angivits för dimensionerna i diagrammet.

Exempel och resultat:

Skapa två visualiseringar från dimensionerna Product och Salesoch ytterligare en från Product och UnitSales. Lägg till mått på det sätt som visas i nedanstående tabell.

Rangordningsexempel

Exempel	Resultat
Exempel 1. Skapa en tabell med dimensionerna `Customer` och `Sales` och måttet `Rank(Sales)`	Resultatet beror på dimensionernas sorteringsordning. Om tabellen sorteras på Customer listas i tabellen alla värden för Sales för Astrida, sedan Betacab och så vidare. Resultaten för Rank(Sales) visar 10 för Sales-värde 12, 9 för Sales-värde 13, och så vidare, med rangordningsvärdet 1 returnerat för Sales-värde 78. Nästa kolumnsegment börjar med Betacab, där det första värdet för Sales i segmentet är 12. Rangordningsvärdet för Rank(Sales) för detta ges som 11. Om tabellen sorteras på Sales består kolumnsegmenten av värdena för Sales och motsvarande Customer. Eftersom det finns två Sales-värden 12 (för Astrida och Betacab), är Rank (Sales)-värdet för det kolumnsegmentet 1-2 för varje Customer-värde. Detta beror på att det finns två värden för Customer för Sales-värdet 12. Om det hade funnits 4 värden skulle resultatet blivit 1-4 för alla rader. Detta visar hur resultatet ser ut för standardvärdet (0) för argumentet fmt.
Exempel 2. Ersätt kunddimensionen med produktdimensionen och lägg till måttet `Rank(Sales,1,2)`	Detta returnerar 1 på den första raden för varje kolumnsegment och lämnar alla andra rader tomma eftersom argumenten **mode** och **fmt** är inställda på 1 respektive 2.

Resultat för exempel 1, med tabellen sorterad efter Customer:

Resultattabell

Customer	Sales	Rank(Sales)
Astrida	12	10
Astrida	13	9
Astrida	20	8
Astrida	22	7
Astrida	45	6
Astrida	46	5
Astrida	60	4
Astrida	65	3
Astrida	70	2

Customer	Sales	Rank(Sales)
Astrida	78	1
Betcab	12	11

Resultat för exempel 1, med tabellen sorterad efter Sales:

Resultattabell

Customer	Sales	Rank(Sales)
Astrida	12	1-2
Betacab	12	1-2
Astrida	13	1
Betacab	15	1
Astrida	20	1
Astrida	22	1-2
Betacab	22	1-2
Betacab	24	1-2
Canutility	24	1-2

Data som används i exempel:

```
ProductData:

Load * inline [

Customer|Product|UnitSales|UnitPrice

Astrida|AA|4|16

Astrida|AA|10|15

Astrida|BB|9|9

Betacab|BB|5|10

Betacab|CC|2|20

Betacab|DD|0|25

Canutility|AA|8|15

Canutility|CC|0|19

] (delimiter is '|');
```

```
Sales2013:
crosstable (Month, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Se även:

p *Sum - diagramfunktion (page 401)*

HRank - diagramfunktion

HRank() utvärderar uttrycket och jämför resultatet med resultaten i de andra kolumnerna som innehåller det aktuella radsegmentet i en pivottabell. Funktionen returnerar sedan rangordningsnumret för den aktuella kolumnen inom segmentet.

Syntax:

```
HRank([ TOTAL ] expr [ , mode [, fmt ] ])
```

Returnerad datatyp: dual

> *Den här funktionen fungerar endast i pivottabeller. I alla andra diagramtyper returnerar den NULL.*

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
mode	Anger resultatets talrepresentation.
fmt	Anger resultatets textrepresentation.
TOTAL	Om diagrammet är endimensionellt eller om uttrycket föregås av kvalificeraren **TOTAL** utvärderas funktionen längs med hela kolumnen. Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

Om pivottabellen är endimensionell eller om uttryck föregås av bestämningen **total**, är det aktuella radsegmentet alltid liktydigt med hela raden. Om pivottabellen har flera horisontella dimensioner, inbegriper det aktuella radsegmentet endast kolumner som har samma värden som den aktuella kolumnen i samtliga dimensionsrader utom den rad som visar den sista dimensionen i fältens inbördes sorteringsordning.

Rangordningsnumret returneras som ett dualt värde med en text- och en talrepresentation, vilket i det fall där varje kolumn har ett unikt rangordningsnummer är ett heltal mellan 1 och antalet kolumner i det aktuella radsegmentet.

Om flera kolumner har samma rangordningsnummer, kan text- och talrepresentationerna för gruppen kontrolleras med argumenten **mode** och **format**.

Det andra argumentet, **mode**, anger funktionsresultatets talrepresentation:

Exempel med **mode**

Värde	Beskrivning
0 (standard)	Om alla rangordningsnummer inom gruppen är lägre än det mellersta värdet i den totala rangordningen, får alla kolumner i gruppen det lägsta rangordningsnumret inom gruppen. Om alla rangordningsnummer inom gruppen är högre än det mellersta värdet i den totala rangordningen, får alla kolumner i gruppen det högsta rangordningsnumret inom gruppen. Om rangordningsnumren inom gruppen spänner över det mellersta värdet i den totala rangordningen, får alla rader i gruppen ett värde som motsvarar medelvärdet av det högsta och det lägsta rangordningsnumret i hela kolumnsegmentet.
1	Lägsta rangordningsnumret på alla kolumner i gruppen
2	Medelvärdet ges som rangordningsnummer på alla kolumner i gruppen
3	Högsta rangordningsnumret på alla kolumner i gruppen
4	lägsta rangordningsnumret på första kolumnen, därefter ökning med ett för varje kolumn.

Det tredje argumentet, **format**, anger funktionsresultatets textrepresentation:

Exempel med **format**

Värde	Beskrivning
0 (standard)	Lägsta värdet**&' - '&**högsta värdet på alla kolumner i gruppen (exempelvis 3 - 4).
1	Lägsta värdet på alla kolumner i gruppen
2	Lägsta värdet på första kolumnen, tom sträng på följande kolumner i gruppen

Ordningen på kolumnerna inom **mode** 4 och **format**2 bestäms av sorteringsordningen som angivits för dimensionerna i diagrammet.

Exempel:

```
HRank( sum( Sales ))

HRank( sum( Sales ), 2 )

HRank( sum( Sales ), 0, 1 )
```

Optimering med k-medelvärde: ett exempel från verkligheten

I det här exemplet beskrivs ett verkligt fall där funktionerna k-medelvärdesklustring och centroid används för en datauppsättning. Funktionen KMeans delar upp datapunkter i kluster med liknande egenskaper. Klustren blir mer kompakta och differentierade efter hand som KMeans-algoritmen används upprepade gånger, där antalet kan konfigureras.

KMeans har många användningsområden, och några exempel på hur kluster kan användas är kundsegmentering, upptäcka bedrägerier, förutsäga förlust av konton, målgruppsstyra kundincitament, identifiera cyberkriminella och optimera leveransrutter. Algoritmen för k-medelvärdesklustring används alltmer när företag vill dra slutsatser om mönster och optimera produkterna som erbjuds.

Qlik Sense Funktionerna KMeans och Centroid

I Qlik Sense finns två KMeans-funktioner som grupperar datapunkter i kluster baserat på likheter. Läs mer i *KMeans2D - diagramfunktion (page 1638)* och *KMeansND - diagramfunktion (page 1653)*. Funktionen **KMeans2D** accepterar två dimensioner och den fungerar bra om du vill illustrera resultatet med ett **spridningsdiagram**. Med funktionen **KMeansND** kan du använda fler än två dimensioner. Eftersom det är lätt att visa tvådimensionella resultat i standarddiagram, använder vi KMeans i ett **spridningsdiagram** med två dimensioner i följande demonstration. K-medelvärdesklustring kan visualiseras med färgläggning efter uttryck eller efter dimension så som beskrivs i exemplet.

Qlik Senses centroidfunktioner avgör den aritmetiska medelpositionen för alla datapunkter i klustret och identifierar en mittpunkt, eller centroid, för detta kluster. För varje rad i diagrammet (eller post), visar centroidfunktionen den koordinat i klustret som datapunkten har tilldelats. Läs mer i *KMeansCentroid2D - diagramfunktion (page 1668)* och *KMeansCentroidND - diagramfunktion (page 1669)*.

Översikt över användning och exempel

Följande exempel visar de olika stegen i ett simulerat verkligt scenario. Ett textilföretag i delstaten New York i USA måste minska sina utgifter genom att minimera leveranskostnaderna. Ett sätt att göra detta är att flytta lagren närmare företagets distributörer. Företaget använder 118 distributörer i delstaten New York. Demonstrationen simulerar hur en driftchef delar in distributörer i fem grupper baserat på geografiska områden med hjälp av funktionen KMeans. Sedan identifieras fem optimala lagerplatser som ligger centralt i dessa kluster med hjälp av funktionen Centroid. Målet är att hitta kartkoordinater som kan användas för att hitta fem centrala lagerplatser.

Datauppsättningen

Datauppsättningen baseras på slumpmässigt genererade namn och adresser i delstaten New York, med verkliga koordinater för latitud och longitud. Datauppsättningen innehåller följande tio kolumner: id, first_name, last_name, telephone, address, city, state, zip, latitude, longitude. Nedan finns datauppsättningen som en fil som du kan hämta till en lokal dator och sedan ladda upp till Qlik Sense eller använda inline i Skriptredigeraren. Appen som skapas får namnet *Distributors KMeans and Centroid* och det första arket i appen heter *Distribution cluster analysis*.

Använd den här länken om du vill hämta filen med exempeldata: DistributorData.csv

Distributor-datauppsättning: inline-laddning för Dataladdningsredigeraren i Qlik Sense (page 1636)

Rubrik: DistributorData

Totalt antal poster: 118

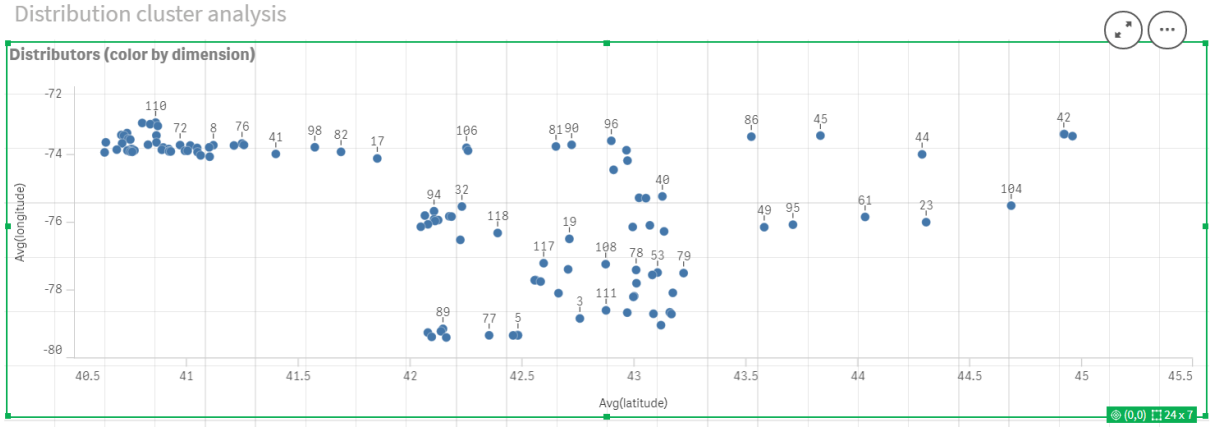
Använda funktionen KMeans2D

I det här exemplet visas hur du konfigurerar ett **spridnings**diagram. Datauppsättningen *DistributorData* används, funktionen **KMeans2D** tillämpas och diagrammet färgläggs per dimension.

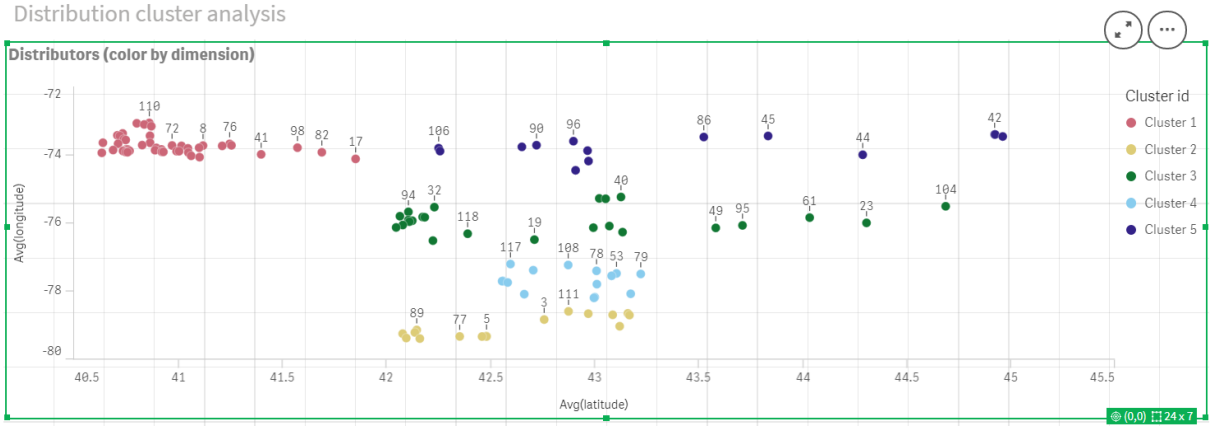
Observera att KMeans-funktionerna i Qlik Sense stöder automatisk klustring med en metod som kallas djupskillnad (depth difference, DeD). När användaren anger 0 som antal kluster bestäms det optimala antalet kluster för den datauppsättningen. I det här exemplet skapas dock en variabel för **num_clusters**-argumentet (syntaxen beskrivs i *KMeans2D - diagramfunktion (page 1638)*). Önskat antal kluster (k=5) specificeras därför av en variabel.

1. Ett **spridningsdiagram** dras till arket och får namnet *Distributors (by dimension)*.
2. En **variabel** skapas för att ange antalet kluster. **Variabeln** ges namnet *vDistClusters*. För variabeln **Definition** anges *5*.
3. Konfiguration av **Data** för diagrammet:
    a. Under **Dimensioner** väljs fältet *id* för **Bubbla**. *Cluster id* anges för **Etikett**.
    b. Under **Mått** är *Avg([latitude])* uttrycket för **X-axel**.
    c. Under **Mått** är *Avg([longitude])* uttrycket för **Y-axel**.
4. Konfiguration av **Utseende**:
    a. Under **Färger och teckenförklaring** väljs **Anpassad** för **Färger**.
    b. **Per dimension** väljs för färgläggning av diagrammet.
    c. Följande uttryck skrivs: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    d. Kryssrutan vid **Låsta färger** markeras.

Spridningsdiagram innan funktionen KMeans med färgläggning per dimension används



Spridningsdiagram efter att funktionen KMeans med färgläggning per dimension används



Lägga till en tabell: *Distributörer*

Det kan vara praktiskt att ha en tabell där du snabbt kan titta på relevanta data. I **spridningsdiagrammet** visas *ID:n* i en tabell där motsvarande distributörsnamn har lagts till som referens.

1. En **Enkel tabell** med namnet *Distributors* dras till arket där följande **Kolumner** (Dimensioner) har lagts till: *id*, *first_name* och *last_name*.

Tabell: Namn på distributörer

Distributors

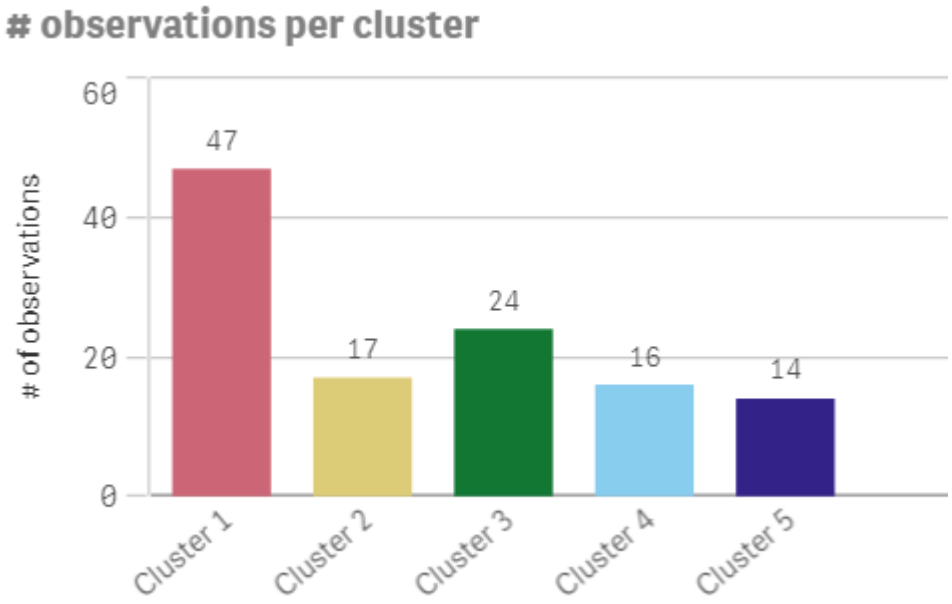
id	first_name	last_name
1	Kaiya	Snow
2	Dean	Roy
3	Eden	Paul
4	Bryanna	Higgins
5	Elisabeth	Lee
6	Skylar	Robinson
7	Cody	Bailey
8	Dario	Sims
9	Deacon	Hood

Lägger till ett **stapeldiagram**: *# observations per cluster*

För exemplet med placering av lager är det praktiskt att veta hur många distributörer som ska använda varje lager. Därför skapas ett **stapeldiagram** som mäter hur många distributörer som ingår i varje kluster.

1. Ett **stapeldiagram** dras till arket. Diagrammet ges namnet: *# observations per cluster*.
2. Konfiguration av **Data** för **stapeldiagrammet**:
    a. En **Dimension** med etiketten *Clusters* läggs till (etiketten kan läggas till efter att uttrycket har tillämpats). Följande uttryck skrivs: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    b. Ett **Mått** med etiketten *# of observations* läggs till. Följande uttryck skrivs: *=count(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id))*
3. Konfiguration av **Utseende**:
    a. Under **Färger och teckenförklaring** väljs **Anpassad** för **Färger**.
    b. **Per dimension** väljs för färgläggning av diagrammet.
    c. Följande uttryck skrivs: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    d. Kryssrutan vid **Låsta färger** markeras.
    e. **Visa teckenförklaring** inaktiveras.
    f. Under **Presentation** anges **Auto** för **Värdeetiketter**.
    g. Under **x-axel**: **kluster**, **endast etiketter** väljs.

Stapeldiagram: antal observationer per kluster



Använda funktionen **Centroid2D**

En andra tabell läggs till för funktionen **Centroid2D** som kommer att identifiera koordinaterna för potentiella lagerplatser. I tabellen visas den centrala platsen (mittpunktsvärdena) för de fem identifierade distributörsgrupperna.

1. En **Tabell** dras till arket och ges namnet *Cluster centroids*. Följande kolumner läggs till:
    a. En **Dimension** med namnet *Clusters* läggs till. Följande uttryck skrivs: *=pick(aggr (KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1,'Warehouse 1','Warehouse 2','Warehouse 3','Warehouse 4','Warehouse 5'*)
    b. Ett **Mått** med namnet *latitude (D1)* läggs till. Följande uttryck skrivs: *=only(aggr (KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id))*
    Notera att parametern **coordinate_no** motsvarar första dimensionen, dimension(0). I det här fallet ritas dimensionen *latitude* på x-axeln. Om vi arbetade med funktionen **CentroidND** och det fanns upp till sex dimensioner, kunde dessa parametrar bara vara något av sex värden: 0, 1, 2, 3, 4 eller 5.
    c. Ett **Mått** med namnet *longitude (D2)* läggs till. Följande uttryck skrivs: *=only(aggr (KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id))*
    Parametern **coordinate_no** i det här uttrycket motsvarar den andra dimensionen, dimension (1). Dimensionen *longitude* ritas på y-axeln.

Tabell: beräkningar för klustrens mittpunkter

Cluster centroids

Clusters	latitude (D1)	longitude (D2)
Totals	-	-
Warehouse 1	40.945422240426	-73.719966482979
Warehouse 2	42.590538729412	-79.067889217647
Warehouse 3	42.805089516667	-75.901621883333
Warehouse 4	42.8581692625	-77.6800485875
Warehouse 5	43.436770771429	-73.734622635714

Placering av mittpunkter på karta

Nästa steg är att placera mittpunkterna på kartan. Om så önskas kan apputvecklaren placera visualiseringen på olika ark.

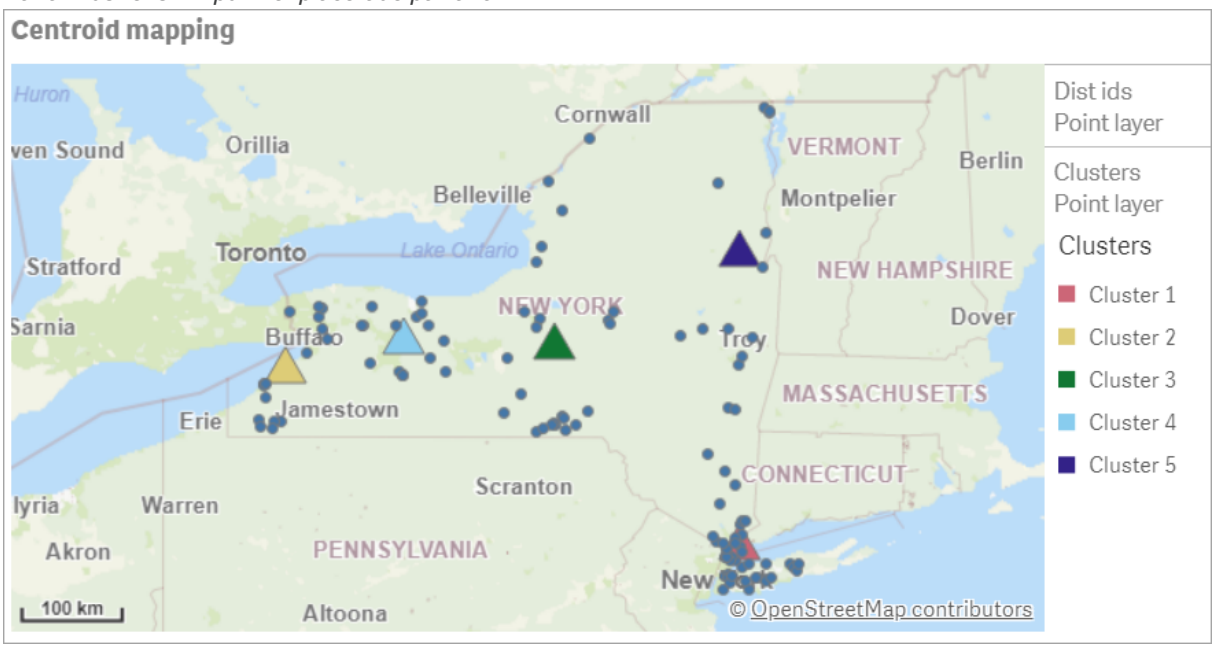
1. En **karta** med namnet *Centroid mapping* dras till arket.
2. I delavsnittet om**skikt** är **lägg till skikt** valt, sedan väljs **punktskikt**.
    a. I **Fält** väljs *id*, och *Dist ids* läggs till vid **Etikett**.
    b. I delavsnittet **Plats** väljs kryssrutan **Latitud- och longitudfält**.
    c. Vid **Latitud** väljs fältet *latitude*.
    d. Vid **Longitud** väljs fältet *longitude*.
    e. I delavsnittet **Storlek & form** väljs **Bubbla** vid **Form**, och **Storlek** minskas till önskad storlek med skjutreglaget.
    f. I delavsnittet **Färger** väljs **En färg**. Blått väljs för **Färg** och grått för **Konturfärg** (dessa val kan ändras enligt önskemål).
3. I delavsnittet **Skikt** läggs ett andra **Punktskikt** till genom att välja först **Lägg till skikt** och sedan **Punktskikt**.
    a. Följande uttryck skrivs: *=aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)*
    b. Vid **Etikett** läggs *Clusters* till.
    c. I delavsnittet **Plats** väljs kryssrutan **Latitud- och longitudfält**.
    d. Vid **Latitud**, som i det här fallet ritas längs x-axeln, läggs följande uttryck till: *=aggr (KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id)*
    e. Vid **Longitud**, som i det här fallet ritas längs y-axeln, läggs följande uttryck till: *=aggr (KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id)*
    f. I delavsnittet **Storlek & form** väljs **Triangel** vid **Form**, och **Storlek** minskas till önskad storlek med skjutreglaget.
    g. Under **Färger och teckenförklaring** väljs **Anpassad** vid **Färger**.

h. **Per dimension** väljs för färgläggning av diagrammet. Följande uttryck skrivs: *=pick(aggr (KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1,'Cluster 1','Cluster 2','Cluster 3','Cluster 4','Cluster 5')*

i. Dimensionen ges etiketten *Clusters*.

4. I **Kartinställningar** väljs **Anpassad** vid **Projektion**. **Metersystemet** väljs vid **Mätenheter**.

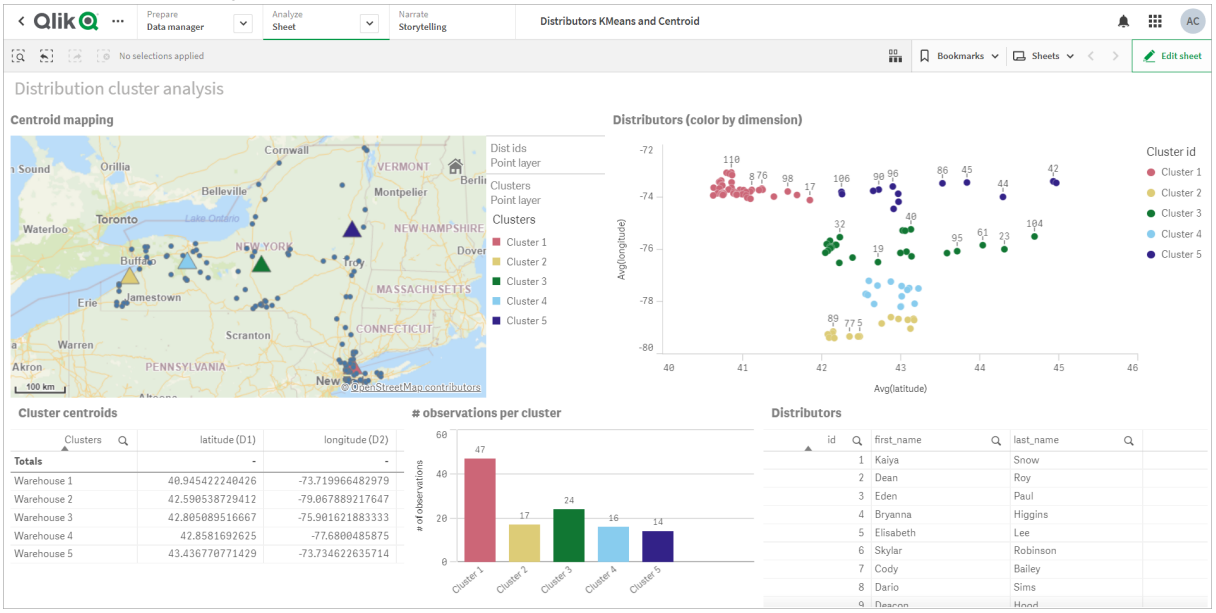
Karta: klustrens mittpunkter placerade på kartan



Slutsats

I det här exemplet från verkligheten har funktionen KMeans använts för att dela in distributörer i grupper eller kluster baserat på likhet; i det här fallet hur nära de ligger varandra geografiskt. Funktionen Centroid tillämpades på klustren för att identifiera fem uppsättningar kartkoordinater. Dessa koordinater ger en första central plats där lager kan byggas eller sökas. Centroid-funktionen tillämpas på **kartdiagrammet** så att appanvändare kan se var mittpunkterna är belägna i relation till datapunkterna i klustret. Resulterande koordinater representerar potentiella platser för lager, som skulle kunna minska leveranskostnader för distributörer i delstaten New York.

App: exempel på analys med KMeans och Centroid



Distributor-datauppsättning: inline-laddning för Dataladdningsredigeraren i Qlik Sense

```
DistributorData:
Load * Inline [
id,first_name,last_name,telephone,address,city,state,zip,latitude,longitude
1,Kaiya,Snow,(716) 201-1212,6231 Tonawanda Creek Rd #APT 308,Lockport,NY,14094,43.08926,-
78.69313
2,Dean,Roy,(716) 201-1588,6884 E High St,Lockport,NY,14094,43.16245,-78.65036
3,Eden,Paul,(716) 202-4596,4647 Southwestern Blvd #APT 350,Hamburg,NY,14075,42.76003,-78.83194
4,Bryanna,Higgins,(716) 203-7041,418 Park Ave,Dunkirk,NY,14048,42.48279,-79.33088
5,Elisabeth,Lee,(716) 203-7043,36 E Courtney St,Dunkirk,NY,14048,42.48299,-79.31928
6,Skylar,Robinson,(716) 203-7166,26 Greco Ln,Dunkirk,NY,14048,42.4612095,-79.3317925
7,Cody,Bailey,(716) 203-7201,114 Lincoln Ave,Dunkirk,NY,14048,42.4801269,-79.322232
8,Dario,Sims,(408) 927-1606,N Castle Dr,Armonk,NY,10504,41.11979,-73.714864
9,Deacon,Hood,(410) 244-6221,4856 44th St,Woodside,NY,11377,40.748372,-73.905445
10,Zackery,Levy,(410) 363-8874,61 Executive Blvd,Farmingdale,NY,11735,40.7197457,-73.430239
11,Rey,Hawkins,(412) 344-8687,4585 Shimerville Rd,Clarence,NY,14031,42.972075,-78.6592452
12,Phillip,Howard,(413) 269-4049,464 Main St #101,Port Washington,NY,11050,40.8273756,-
73.7009971
13,Shirley,Tyler,(434) 985-8943,114 Glann Rd,Apalachin,NY,13732,42.0482515,-76.1229725
14,Aniyah,Jarvis,(440) 244-1808,87 N Middletown Rd,Pearl River,NY,10965,41.0629,-74.0159
15,Alayna,Woodard,(478) 335-3704,70 W Red Oak Ln,West Harrison,NY,10604,41.0162722,-73.7234926
16,Jermaine,Lambert,(508) 561-9836,24 Kellogg Rd,New Hartford,NY,13413,43.0555739,-75.2793197
17,Harper,Gibbs,(239) 466-0238,Po Box 33,Cottekill,NY,12419,41.853392,-74.106082
18,Osvaldo,Graham,(252) 246-0816,6878 Sand Hill Rd,East Syracuse,NY,13057,43.073215,-76.081448
19,Roberto,Wade,(270) 469-1211,3936 Holley Rd,Moravia,NY,13118,42.713044,-76.481227
20,Kate,Mcguire,(270) 788-3080,6451 State 64 Rte #3,Naples,NY,14512,42.707366,-77.380489
21,Dale,Andersen,(281) 480-5690,205 W Service Rd,Champlain,NY,12919,44.9645392,-73.4470831
22,Lorelai,Burch,(302) 644-2133,1 Brewster St,Glen Cove,NY,11542,40.865177,-73.633019
23,Amiyah,Flowers,(303) 223-0055,46600 Us Interstate 81 Rte,Alexandria
Bay,NY,13607,44.309626,-75.988365
```

```
24,Mckinley,Clements,(303) 918-3230,200 Summit Lake Dr,Valhalla,NY,10595,41.101145,-73.778298
25,Marc,Gibson,(607) 203-1233,25 Robinson St,Binghamton,NY,13901,42.107416,-75.901614
26,Kali,Norman,(607) 203-1400,1 Ely Park Blvd #APT 15,Binghamton,NY,13905,42.125866,-75.925026
27,Laci,Cain,(607) 203-1437,16 Zimmer Road,Kirkwood,NY,13795,42.066516,-75.792627
28,Mohammad,Perez,(607) 203-1652,71 Endicott Ave #APT 12,Johnson City,NY,13790,42.111894,-
75.952187
29,Izabelle,Pham,(607) 204-0392,434 State 369 Rte,Port Crane,NY,13833,42.185838,-75.823074
30,Kiley,Mays,(607) 204-0870,244 Ballyhack Rd #14,Port Crane,NY,13833,42.175612,-75.814917
31,Peter,Trevino,(607) 205-1374,125 Melbourne St.,Vestal,NY,13850,42.080254,-76.051124
32,Ani,Francis,(607) 208-4067,48 Caswell St,Afton,NY,13730,42.232065,-75.525674
33,Jared,Sheppard,(716) 386-3002,4709 430th Rte,Bemus Point,NY,14712,42.162175,-79.39176
34,Dulce,Atkinson,(914) 576-2266,501 Pelham Rd,New Rochelle,NY,10805,40.895449,-73.782602
35,Jayla,Beasley,(716) 526-1054,5010 474th Rte,Ashville,NY,14710,42.096859,-79.375561
36,Dane,Donovan,(718) 545-3732,5014 31st Ave,Woodside,NY,11377,40.756967,-73.909506
37,Brendon,Clay,(585) 322-7780,133 Cummings Ave,Gainesville,NY,14066,42.664309,-78.085651
38,Asia,Nunez,(718) 426-1472,2407 Gilmore ,East Elmhurst,NY,11369,40.766662,-73.869185
39,Dawson,Odonnell,(718) 342-2179,5019 H Ave,Brooklyn,NY,11234,40.633245,-73.927591
40,Kyle,Collins,(315) 733-7078,502 Rockhaven Rd,Utica,NY,13502,43.129184,-75.226726
41,Eliza,Hardin,(315) 331-8072,502 Sladen Place,West Point,NY,10996,41.3993,-73.973003
42,Kasen,Klein,(518) 298-4581,2407 Lake Shore Rd,Chazy,NY,12921,44.925561,-73.387373
43,Reuben,Bradford,(518) 298-4581,33 Lake Flats Dr,Champlain,NY,12919,44.928092,-73.387884
44,Henry,Grimes,(518) 523-3990,2407 Main St,Lake Placid,NY,12946,44.291487,-73.98474
45,Kyan,Livingston,(518) 585-7364,241 Alexandria Ave,Ticonderoga,NY,12883,43.836553,-73.43155
46,Kaitlyn,Short,(516) 678-3189,241 Chance Dr,Oceanside,NY,11572,40.638534,-73.63079
47,Damaris,Jacobs,(914) 664-5331,241 Claremont Ave,Mount Vernon,NY,10552,40.919852,-73.827848
48,Alivia,Schroeder,(315) 469-4473,241 Lafayette Rd,Syracuse,NY,13205,42.996446,-76.12957
49,Bridget,Strong,(315) 298-4355,241 Maltby Rd,Pulaski,NY,13142,43.584966,-76.136317
50,Francis,Lee,(585) 201-7021,166 Ross St,Batavia,NY,14020,43.0031502,-78.17487
51,Makaila,Phelps,(585) 201-7422,58 S Main St,Batavia,NY,14020,42.99941,-78.1939285
52,Jazlynn,Stephens,(585) 203-1087,1 Sinclair Dr,Pittsford,NY,14534,43.084157,-77.545452
53,Ryann,Randolph,(585) 203-1519,331 Eaglehead Rd,East Rochester,NY,14445,43.10785,-77.475552
54,Rosa,Baker,(585) 204-4011,42 Ossian St,Dansville,NY,14437,42.560761,-77.70088
55,Marcel,Barry,(585) 204-4013,42 Jefferson St,Dansville,NY,14437,42.557735,-77.702983
56,Dennis,Schmitt,(585) 204-4061,750 Dansville Mount Morris Rd,Dansville,NY,14437,42.584458,-
77.741648
57,Cassandra,Kim,(585) 204-4138,3 Perine Ave APT1,Dansville,NY,14437,42.562865,-77.69661
58,Kolton,Jacobson,(585) 206-5047,4925 Upper Holly Rd,Holley,NY,14470,43.175957,-78.074465
59,Nathanael,Donovan,(718) 393-3501,9604 57th Ave,Corona,NY,11373,40.736077,-73.864858
60,Robert,Frazier,(718) 271-3067,300 56th Ave,Corona,NY,11373,40.735304,-73.873997
61,Jessie,Mora,(315) 405-8991,9607 Forsyth Loop,Watertown,NY,13603,44.036466,-75.833437
62,Martha,Rollins,(347) 242-2642,22 Main St,Corona,NY,11373,40.757727,-73.829331
63,Emely,Townsend,(718) 699-0751,60 Sanford Ave,Corona,NY,11373,40.755466,-73.831029
64,Kylie,Cooley,(347) 561-7149,9608 95th Ave,Ozone Park,NY,11416,40.687564,-73.845715
65,Wendy,Cameron,(585) 571-4185,9608 Union St,Scottsville,NY,14546,43.013327,-77.7907839
66,Kayley,Peterson,(718) 654-5027,961 E 230th St,Bronx,NY,10466,40.889275,-73.850555
67,Camden,Ochoa,(718) 760-8699,59 Vark St,Yonkers,NY,10701,40.929322,-73.89957
68,Priscilla,Castillo,(910) 326-7233,9359 Elm St,Chadwicks,NY,13319,43.024902,-75.26886
69,Dana,Schultz,(913) 322-4580,99 Washington Ave,Hastings on Hudson,NY,10706,40.99265,-
73.879748
70,Blaze,Medina,(914) 207-0015,60 Elliott Ave,Yonkers,NY,10705,40.921498,-73.896682
71,Finnegan,Tucker,(914) 207-0015,90 Hillside Drive,Yonkers,NY,10705,40.922514,-73.892911
72,Pranav,Palmer,(914) 214-8376,5 Bruce Ave,Harrison,NY,10528,40.970916,-73.711493
73,Kolten,Wong,(914) 218-8268,70 Barker St,Mount Kisco,NY,10549,41.211993,-73.723202
74,Jasiah,Vazquez,(914) 231-5199,30 Broadway,Dobbs Ferry,NY,10522,41.004629,-73.879825
75,Lamar,Pierce,(914) 232-0380,68 Ridge Rd,Katonah,NY,10536,41.256662,-73.707964
76,Carla,Coffey,(914) 232-0469,197 Beaver Dam Rd,Katonah,NY,10536,41.247934,-73.664363
```

```
77,Brooklynn,Harmon,(716) 595-3227,8084 Glasgow Rd,Cassadega,NY,14718,42.353861,-79.329558
78,Raquel,Hodges,(585) 398-8125,809 County Road ,Victor,NY,14564,43.011745,-77.398806
79,Jerimiah,Gardner,(585) 787-9127,809 Houston Rd,Webster,NY,14580,43.224204,-77.491353
80,Clarence,Hammond,(720) 746-1619,809 Pierpont Ave,Piermont,NY,10968,41.0491181,-73.918622
81,Rhys,Gill,(518) 427-7887,81 Columbia St,Albany,NY,12210,42.652824,-73.752096
82,Edith,Parrish,(845) 452-7621,81 Glenwood Ave,Poughkeepsie,NY,12603,41.691058,-73.910829
83,Kobe,Mcintosh,(845) 371-1101,81 Heitman Dr,Spring Valley,NY,10977,41.103227,-74.054396
84,Ayden,Waters,(516) 796-2722,81 Kingfisher Rd,Levittown,NY,11756,40.738939,-73.52826
85,Francis,Rogers,(631) 427-7728,81 Knollwood Ave,Huntington,NY,11743,40.864905,-73.426107
86,Jaden,Landry,(716) 496-4038,12839 39th Rte,Chaffee,NY,14030,43.527396,-73.462786
87,Giancarlo,Campos,(518) 885-5717,1284 Saratoga Rd,Ballston Spa,NY,12020,42.968594,-73.862847
88,Eduardo,Contreras,(716) 285-8987,1285 Saunders Sett Rd,Niagara Falls,NY,14305,43.122963,-
79.029274
89,Gabriela,Davidson,(716) 267-3195,1286 Mee Rd,Falconer,NY,14733,42.147339,-79.137976
90,Evangeline,Case,(518) 272-9435,1287 2nd Ave,Watervliet,NY,12189,42.723132,-73.703818
91,Tyrone,Ellison,(518) 843-4691,1287 Midline Rd,Amsterdam,NY,12010,42.9730876,-74.1700608
92,Bryce,Bass,(518) 943-9549,1288 Leeds Athens Rd,Athens,NY,12015,42.259381,-73.876897
93,Londyn,Butler,(518) 922-7095,129 Argersinger Rd,Fultonville,NY,12072,42.910969,-74.441917
94,Graham,Becker,(607) 655-1318,129 Baker Rd,Windsor,NY,13865,42.107271,-75.66408
95,Rolando,Fitzgerald,(315) 465-4166,17164 County 90 Rte,Mannsville,NY,13661,43.713443,-
76.06232
96,Grant,Hoover,(518) 692-8363,1718 County 113 Rte,Schaghticote,NY,12154,42.900648,-73.585036
97,Mark,Goodwin,(631) 584-6761,172 Cambon Ave,Saint James,NY,11780,40.871152,-73.146032
98,Deacon,Cantu,(845) 221-7940,172 Carpenter Rd,Hopewell Junction,NY,12533,41.57388,-73.77609
99,Tristian,Walsh,(516) 997-4750,172 E Cabot Ln,Westbury,NY,11590,40.7480397,-73.54819
100,Abram,Alexander,(631) 588-3817,172 Lorenzo Cir,Ronkonkoma,NY,11779,40.837123,-73.09367
101,Lesly,Bush,(516) 489-3791,172 Nassau Blvd,Garden City,NY,11530,40.71147,-73.660753
102,Pamela,Espinoza,(716) 201-1520,172 Niagara St ,Lockport,NY,14094,43.169871,-78.70093
103,Bryanna,Newton,(914) 328-4332,172 Warren Ave,White Plains,NY,10603,41.047207,-73.79572
104,Marcelo,Schmitt,(315) 393-4432,319 Mansion Ave,Ogdensburg,NY,13669,44.690246,-75.49992
105,Layton,Valenzuela,(631) 676-2113,319 Singingwood Dr,Holbrook,NY,11741,40.801391,-73.058993
106,Roderick,Rocha,(518) 671-6037,319 Warren St,Hudson,NY,12534,42.252527,-73.790629
107,Camryn,Terrell,(315) 635-1680,3192 Olive Dr,Baldinsville,NY,13027,43.136843,-76.260303
108,Summer,Callahan,(585) 394-4195,3192 Smith Road,Canandaigua,NY,14424,42.875457,-77.228039
109,Pierre,Novak,(716) 665-2524,3194 Falconer Kimball Stand Rd,Falconer,NY,14733,42.138439,-
79.211091
110,Kennedi,Fry,(315) 543-2301,32 College Rd,Selden,NY,11784,40.861624,-73.04757
111,Wyatt,Pruitt,(716) 681-4042,277 Ransom Rd,Lancaster ,NY,14086,42.87702,-78.591302
112,Lilly,Jensen,(631) 841-0859,2772 Schliegel Blvd,Amityville,NY,11701,40.708021,-73.413015
113,Tristin,Hardin,(631) 920-0927,278 Fulton Street,West Babylon,NY,11704,40.733578,-73.357321
114,Tanya,Stafford,(716) 484-0771,278 Sampson St,Jamestown,NY,14701,42.0797,-79.247805
115,Paris,Cordova,(607) 589-4857,278 Washburn Rd,Spencer,NY,14883,42.225046,-76.510257
116,Alfonso,Morse,(718) 359-5582,200 Colden St,Flushing,NY,11355,40.750403,-73.822752
117,Maurice,Hooper,(315) 595-6694,4435 Italy Hill Rd,Branchport,NY,14418,42.597957,-77.199267
118,Iris,Wolf,(607) 539-7288,444 Harford Rd,Brooktondale,NY,14817,42.392164,-76.30756
];
```

KMeans2D - diagramfunktion

KMeans2D() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring, och för varje diagramrad visas kluster-ID för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1 respektive coordinate_2. Dessa är båda aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters. Data normaliseras med normparametern (valfritt).

KMeans2D returnerar ett värde per datapunkt. Det returnerade värdet är en dual och är det heltalsvärde som motsvarar klustret varje datapunkt har tilldelats till.

Syntax:

```
KMeans2D(num_clusters, coordinate_1, coordinate_2 [, norm])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
num_clusters	Heltal som anger antalet kluster.
coordinate_1	Aggregeringen som beräknar den första koordinaten, vanligtvis x-axeln på spridningsdiagrammet som kan skapas från diagrammet. Den ytterligare parametern, coordinate_2, beräknar den andra koordinaten.
norm	Den valfria normaliseringsmetoden tillämpas på datauppsättningen före k-medelvärdesklustring. Möjliga värden: 0 eller ”none” för ingen normalisering 1 eller ”zscore” för z-poängsnormalisering 2 eller ”minmax” för min-max-normalisering Om ingen parameter anges eller om den angivna parametern är felaktig används ingen normalisering. Z-poäng normaliserar data baserat på funktionens median och standardavvikelse. Z-poäng säkerställer inte att varje funktion har samma skala men det fungerar bättre än min-max för behandling av outliers. Min-max-normalisering säkerställer att funktionerna har samma skala genom att ta de minsta och största värdena för varje och räkna om varje datapunkt.

Exempel: diagramuttryck

I det här exemplet skapar vi ett spridningsdiagram med datauppsättningen *Iris*, och använder sedan KMeans för att färglägga data efter uttryck.

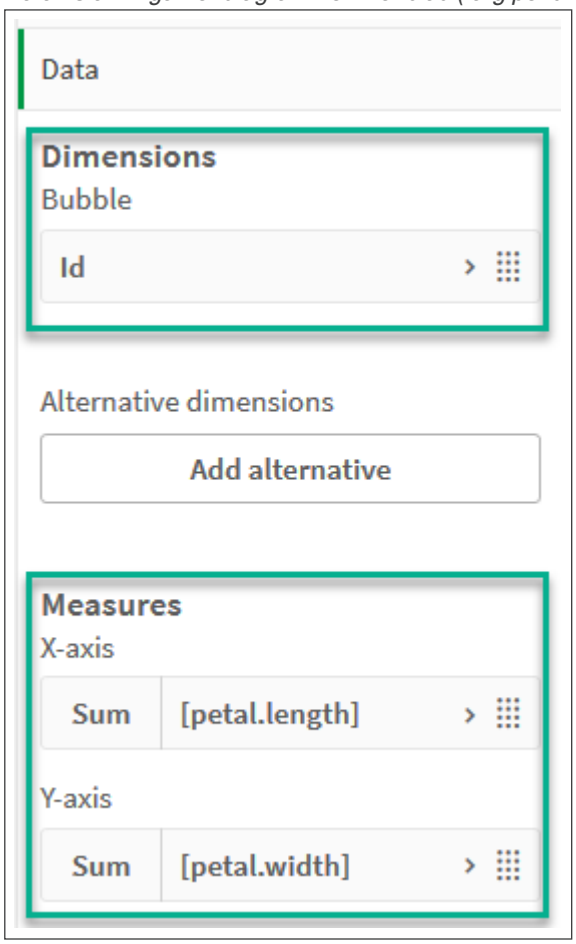
Vi skapar även en variabel för argumentet *num_clusters*, och sedan använder vi en variabelinmatningsruta för att ändra antalet kluster.

Datauppsättningen *Iris* finns offentligt tillgänglig i en mängd format. Vi tillhandahåller dessa data som en inline-tabell som laddas med Skriptredigeraren i Qlik Sense. Observera att vi lagt till en *ID*-kolumn i datatabellen i det här exemplet.

När vi laddat data i Qlik Sense, gör vi följande:

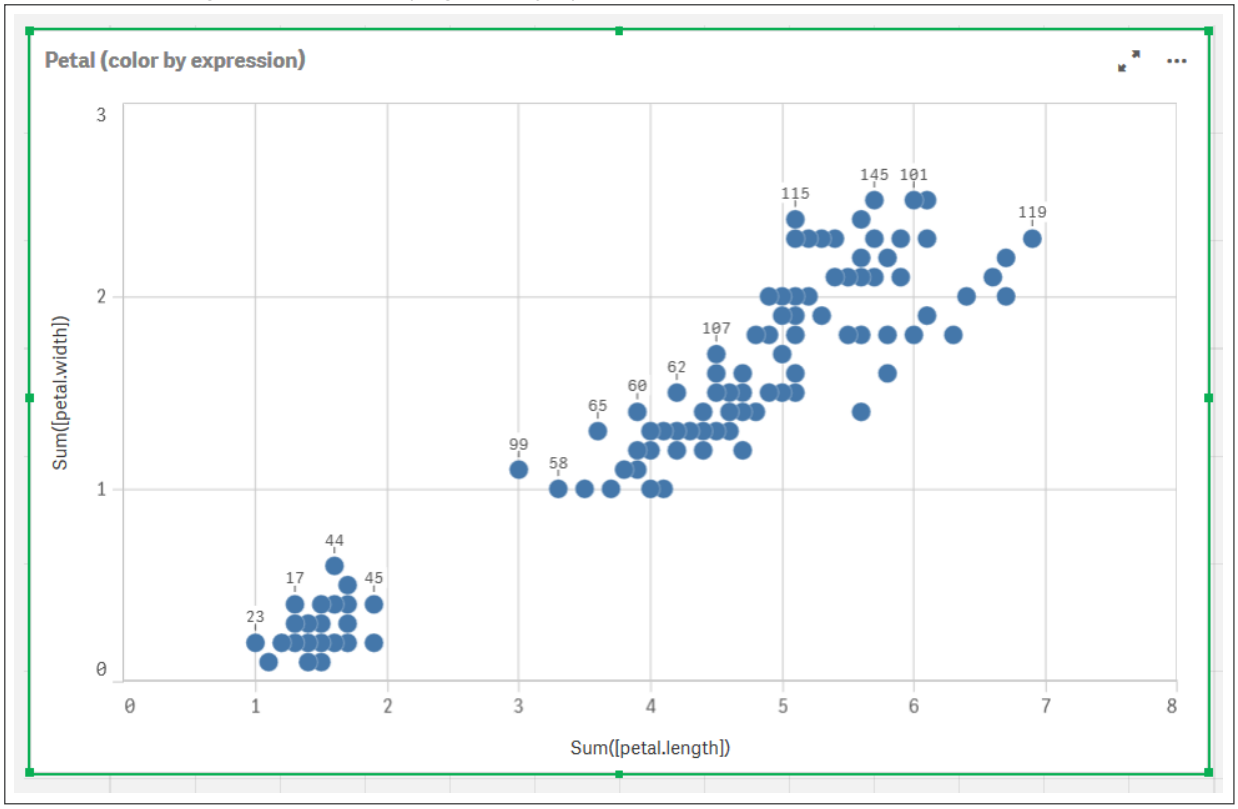
1. Dra ett **spridningsdiagram** till ett nytt ark. Döp diagrammet till *Kronblad (färg per uttryck)*.
2. Skapa en variabel för att ange antalet kluster. För variabeln **Namn** anger du *KmeansPetalClusters*. För variabeln **Definition** anger du *=2*.
3. Konfigurera **Data** för diagrammet:
    i. Under **Dimensioner** väljer du *id* för fältet **Bubbla**. Ange ett kluster-ID för Etikett.
    ii. Under **Mått** väljer du *Sum([petal.length])* för uttrycket för **X-axel**.
    iii. Under **Mått** väljer du *Sum([petal.width])* för uttrycket för **Y-axel**.

    Datainställningar för diagrammet Kronblad (färg per uttryck)



Datapunkterna ritas ut på diagrammet.

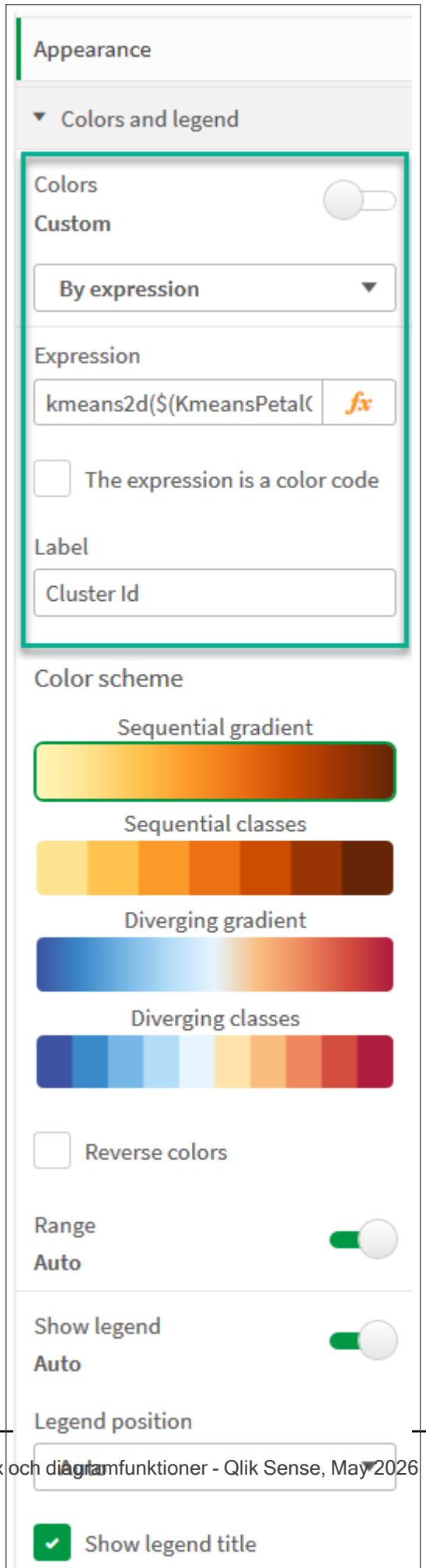
Datapunkter på diagrammet Kronblad (färg per uttryck)



4. Konfigurera **Utseende** för diagrammet:
    i. Under **Färger och teckenförklaring** väljer du **Anpassad** för **Färger**.
    ii. Välj att färglägga diagrammet**Per uttryck**.
    iii. Ange följande för **Uttryck**: *kmeans2d($(KmeansPetalClusters), Sum([petal.length]), Sum ([petal.width]))*
    Observera att *KmeansPetalClusters* är variabeln vi angav som *2*.
    Eller ange följande: *kmeans2d(2, Sum([petal.length]), Sum([petal.width]))*
    iv. Avmarkera kryssrutan **Uttrycket är en färgkod**.

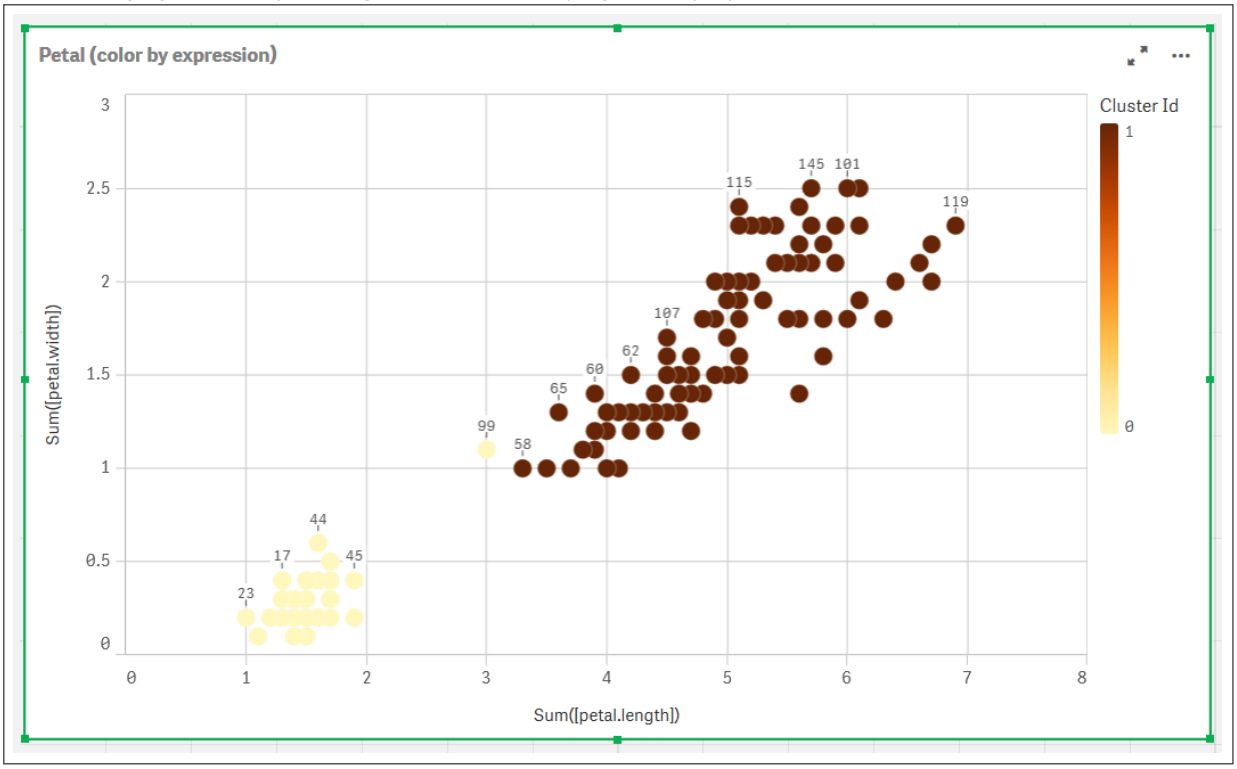
v. Ange följande för **Etikett**: *Kluster-ID*

Utseendeinställningar för diagrammet Petal (color by expression)


Appearance
Colors and legend
Colors
Custom
By expression
Expression
kmeans2d($(KmeansPetal(
The expression is a color code
Label
Cluster Id
Color scheme
Sequential gradient
Sequential classes
Diverging gradient
Diverging classes
Reverse colors
Range
Auto
Show legend
Auto
Legend position
Auto
Show legend title

De två klustren i diagrammet färgläggs per KMeans-uttrycket.

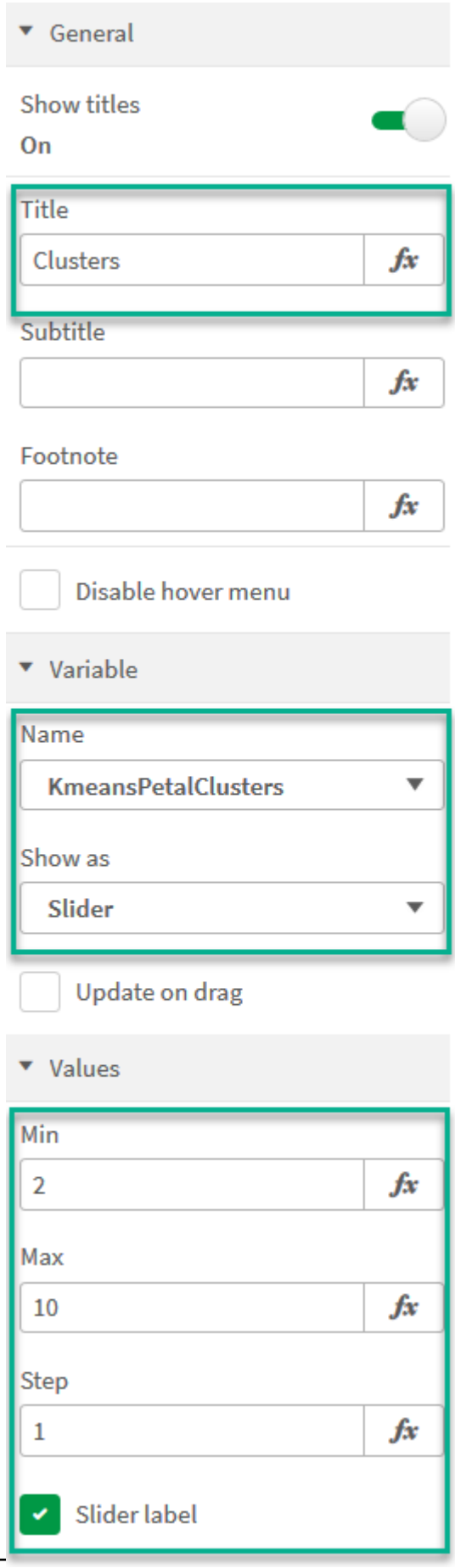
Klustren färglagda per uttryck i diagrammet Kronblad (färg per uttryck)



5. Lägg till en **Variabelinmatning**-ruta för antalet kluster.
    i. I **resurs**panelen, öppna **Diagram**.
    ii. Under **Instrumentpanelsobjekt**, dra en **Variabelinmatning**-ruta till arket.
    Om vi inte hade tillgång till dashboard-buntan kunde vi fortfarande ändra antalet kluster med hjälp av den variabel som vi skapade, eller direkt som ett heltal i uttrycket.
    iii. Under **Utseende**, klicka på **Allmänt**.
    iv. Ange följande för **Titel**: *Kluster*
    v. Klicka på **Variabel**.
    vi. Välj följande variabel för **Namn**: *KmeansPetalClusters*.
    vii. Välj **Reglage** för **Visa som**.

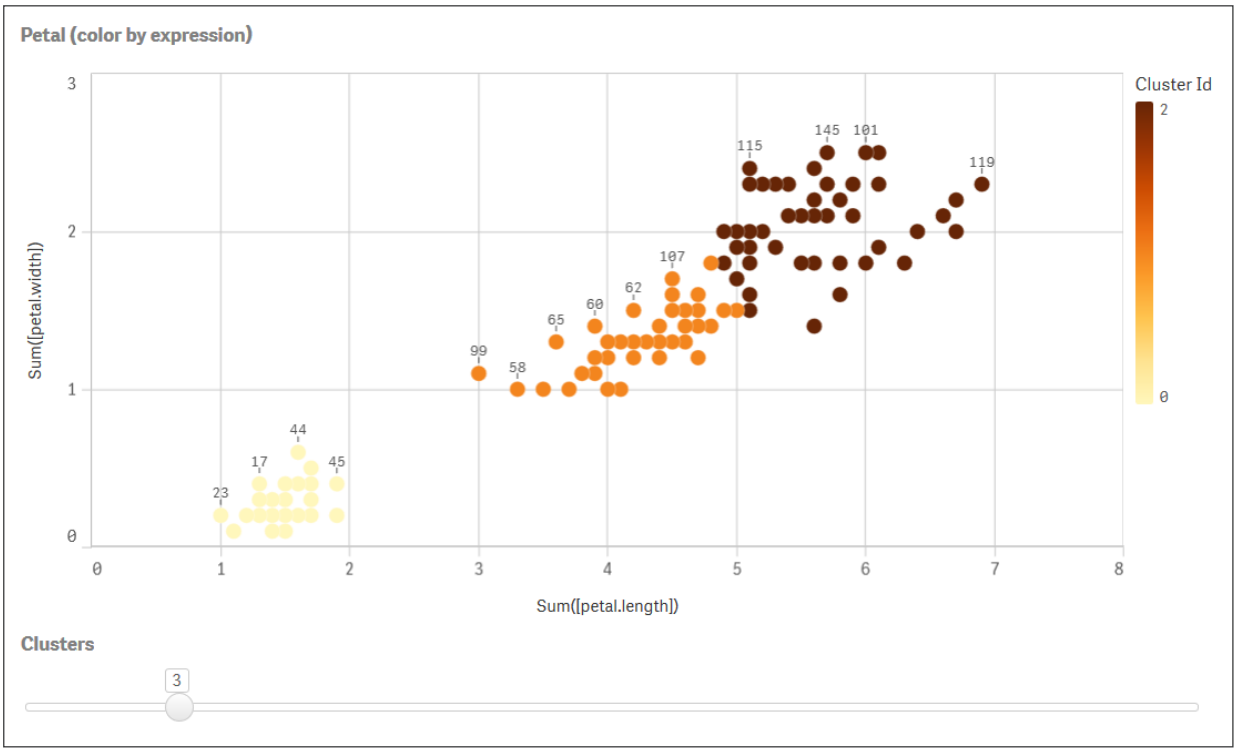
viii. Välj **Värden**, och konfigurera inställningarna efter behov,

Utseende för Kluster variabel inmatningsruta


General
Show titles
On
Title
Clusters
Subtitle
Footnote
Disable hover menu
Variable
Name
KmeansPetalClusters
Show as
Slider
Update on drag
Values
Min
2
Max
10
Step
1
Slider label

När vi är klara med redigeringen kan vi ändra antalet kluster med hjälp av skjutreglaget i inmatningsrutan för variabeln *Kluster*.

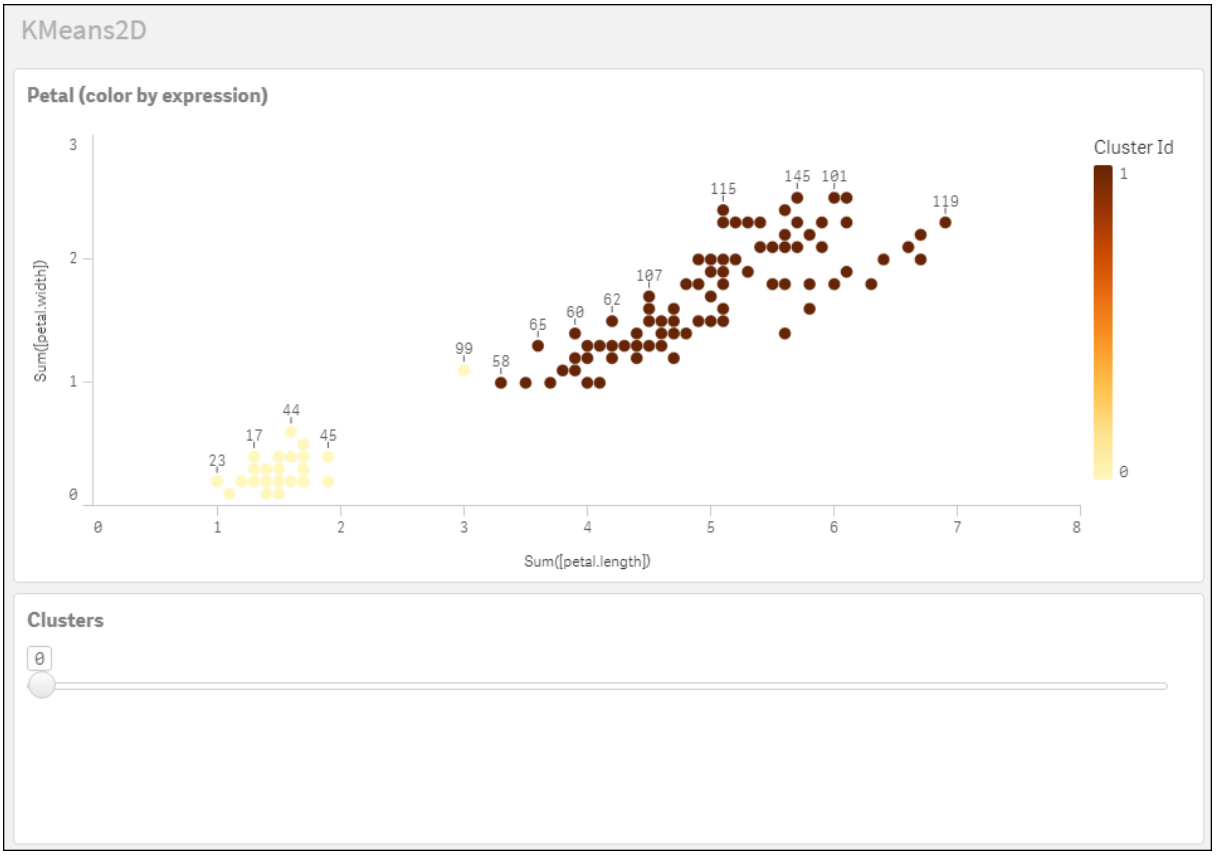
Klustren färglagda per uttryck i diagrammet Kronblad (färg per uttryck)



Automatisk klustring

KMeans-funktioner stöder automatisk klustring med en metod som kallas djupskillnad (depth difference, DeD). När användaren anger 0 som antal kluster bestäms ett optimalt antal kluster för den datauppsättningen. Observera att medan ett heltal som anges för antalet kluster (*k*) inte returneras explicit, beräknas det inom KMeans-algoritmen. Om till exempel 0 specificeras i funktionen för värdet av *KmeansPetalClusters* eller anges via en variabelindataruta, beräknas klustertilldelningar automatiskt för datauppsättningen baserat på ett optimalt antal kluster.

Metoden KMeans-djupskillnad avgör det optimala antalet kluster när (k) är inställt på 0



Iris-datauppsättning: inline-laddning för Dataladdningsredigeraren i Qlik Sense

```
IrisData:
Load * Inline [
sepal.length, sepal.width, petal.length, petal.width, variety, id
5.1, 3.5, 1.4, 0.2, Setosa, 1
4.9, 3, 1.4, 0.2, Setosa, 2
4.7, 3.2, 1.3, 0.2, Setosa, 3
4.6, 3.1, 1.5, 0.2, Setosa, 4
5, 3.6, 1.4, 0.2, Setosa, 5
5.4, 3.9, 1.7, 0.4, Setosa, 6
4.6, 3.4, 1.4, 0.3, Setosa, 7
5, 3.4, 1.5, 0.2, Setosa, 8
4.4, 2.9, 1.4, 0.2, Setosa, 9
4.9, 3.1, 1.5, 0.1, Setosa, 10
5.4, 3.7, 1.5, 0.2, Setosa, 11
4.8, 3.4, 1.6, 0.2, Setosa, 12
4.8, 3, 1.4, 0.1, Setosa, 13
4.3, 3, 1.1, 0.1, Setosa, 14
5.8, 4, 1.2, 0.2, Setosa, 15
5.7, 4.4, 1.5, 0.4, Setosa, 16
5.4, 3.9, 1.3, 0.4, Setosa, 17
5.1, 3.5, 1.4, 0.3, Setosa, 18
5.7, 3.8, 1.7, 0.3, Setosa, 19
5.1, 3.8, 1.5, 0.3, Setosa, 20
5.4, 3.4, 1.7, 0.2, Setosa, 21
```

```
5.1, 3.7, 1.5, 0.4, Setosa, 22
4.6, 3.6, 1, 0.2, Setosa, 23
5.1, 3.3, 1.7, 0.5, Setosa, 24
4.8, 3.4, 1.9, 0.2, Setosa, 25
5, 3, 1.6, 0.2, Setosa, 26
5, 3.4, 1.6, 0.4, Setosa, 27
5.2, 3.5, 1.5, 0.2, Setosa, 28
5.2, 3.4, 1.4, 0.2, Setosa, 29
4.7, 3.2, 1.6, 0.2, Setosa, 30
4.8, 3.1, 1.6, 0.2, Setosa, 31
5.4, 3.4, 1.5, 0.4, Setosa, 32
5.2, 4.1, 1.5, 0.1, Setosa, 33
5.5, 4.2, 1.4, 0.2, Setosa, 34
4.9, 3.1, 1.5, 0.1, Setosa, 35
5, 3.2, 1.2, 0.2, Setosa, 36
5.5, 3.5, 1.3, 0.2, Setosa, 37
4.9, 3.1, 1.5, 0.1, Setosa, 38
4.4, 3, 1.3, 0.2, Setosa, 39
5.1, 3.4, 1.5, 0.2, Setosa, 40
5, 3.5, 1.3, 0.3, Setosa, 41
4.5, 2.3, 1.3, 0.3, Setosa, 42
4.4, 3.2, 1.3, 0.2, Setosa, 43
5, 3.5, 1.6, 0.6, Setosa, 44
5.1, 3.8, 1.9, 0.4, Setosa, 45
4.8, 3, 1.4, 0.3, Setosa, 46
5.1, 3.8, 1.6, 0.2, Setosa, 47
4.6, 3.2, 1.4, 0.2, Setosa, 48
5.3, 3.7, 1.5, 0.2, Setosa, 49
5, 3.3, 1.4, 0.2, Setosa, 50
7, 3.2, 4.7, 1.4, Versicolor, 51
6.4, 3.2, 4.5, 1.5, Versicolor, 52
6.9, 3.1, 4.9, 1.5, Versicolor, 53
5.5, 2.3, 4, 1.3, Versicolor, 54
6.5, 2.8, 4.6, 1.5, Versicolor, 55
5.7, 2.8, 4.5, 1.3, Versicolor, 56
6.3, 3.3, 4.7, 1.6, Versicolor, 57
4.9, 2.4, 3.3, 1, Versicolor, 58
6.6, 2.9, 4.6, 1.3, Versicolor, 59
5.2, 2.7, 3.9, 1.4, Versicolor, 60
5, 2, 3.5, 1, Versicolor, 61
5.9, 3, 4.2, 1.5, Versicolor, 62
6, 2.2, 4, 1, Versicolor, 63
6.1, 2.9, 4.7, 1.4, Versicolor, 64
5.6, 2.9, 3.6, 1.3, Versicolor, 65
6.7, 3.1, 4.4, 1.4, Versicolor, 66
5.6, 3, 4.5, 1.5, Versicolor, 67
5.8, 2.7, 4.1, 1, Versicolor, 68
6.2, 2.2, 4.5, 1.5, Versicolor, 69
5.6, 2.5, 3.9, 1.1, Versicolor, 70
5.9, 3.2, 4.8, 1.8, Versicolor, 71
6.1, 2.8, 4, 1.3, Versicolor, 72
6.3, 2.5, 4.9, 1.5, Versicolor, 73
6.1, 2.8, 4.7, 1.2, Versicolor, 74
6.4, 2.9, 4.3, 1.3, Versicolor, 75
6.6, 3, 4.4, 1.4, Versicolor, 76
```

```
6.8, 2.8, 4.8, 1.4, Versicolor, 77
6.7, 3, 5, 1.7, Versicolor, 78
6, 2.9, 4.5, 1.5, Versicolor, 79
5.7, 2.6, 3.5, 1, Versicolor, 80
5.5, 2.4, 3.8, 1.1, Versicolor, 81
5.5, 2.4, 3.7, 1, Versicolor, 82
5.8, 2.7, 3.9, 1.2, Versicolor, 83
6, 2.7, 5.1, 1.6, Versicolor, 84
5.4, 3, 4.5, 1.5, Versicolor, 85
6, 3.4, 4.5, 1.6, Versicolor, 86
6.7, 3.1, 4.7, 1.5, Versicolor, 87
6.3, 2.3, 4.4, 1.3, Versicolor, 88
5.6, 3, 4.1, 1.3, Versicolor, 89
5.5, 2.5, 4, 1.3, Versicolor, 90
5.5, 2.6, 4.4, 1.2, Versicolor, 91
6.1, 3, 4.6, 1.4, Versicolor, 92
5.8, 2.6, 4, 1.2, Versicolor, 93
5, 2.3, 3.3, 1, Versicolor, 94
5.6, 2.7, 4.2, 1.3, Versicolor, 95
5.7, 3, 4.2, 1.2, Versicolor, 96
5.7, 2.9, 4.2, 1.3, Versicolor, 97
6.2, 2.9, 4.3, 1.3, Versicolor, 98
5.1, 2.5, 3, 1.1, Versicolor, 99
5.7, 2.8, 4.1, 1.3, Versicolor, 100
6.3, 3.3, 6, 2.5, Virginica, 101
5.8, 2.7, 5.1, 1.9, Virginica, 102
7.1, 3, 5.9, 2.1, Virginica, 103
6.3, 2.9, 5.6, 1.8, Virginica, 104
6.5, 3, 5.8, 2.2, Virginica, 105
7.6, 3, 6.6, 2.1, Virginica, 106
4.9, 2.5, 4.5, 1.7, Virginica, 107
7.3, 2.9, 6.3, 1.8, Virginica, 108
6.7, 2.5, 5.8, 1.8, Virginica, 109
7.2, 3.6, 6.1, 2.5, Virginica, 110
6.5, 3.2, 5.1, 2, Virginica, 111
6.4, 2.7, 5.3, 1.9, Virginica, 112
6.8, 3, 5.5, 2.1, Virginica, 113
5.7, 2.5, 5, 2, Virginica, 114
5.8, 2.8, 5.1, 2.4, Virginica, 115
6.4, 3.2, 5.3, 2.3, Virginica, 116
6.5, 3, 5.5, 1.8, Virginica, 117
7.7, 3.8, 6.7, 2.2, Virginica, 118
7.7, 2.6, 6.9, 2.3, Virginica, 119
6, 2.2, 5, 1.5, Virginica, 120
6.9, 3.2, 5.7, 2.3, Virginica, 121
5.6, 2.8, 4.9, 2, Virginica, 122
7.7, 2.8, 6.7, 2, Virginica, 123
6.3, 2.7, 4.9, 1.8, Virginica, 124
6.7, 3.3, 5.7, 2.1, Virginica, 125
7.2, 3.2, 6, 1.8, Virginica, 126
6.2, 2.8, 4.8, 1.8, Virginica, 127
6.1, 3, 4.9, 1.8, Virginica, 128
6.4, 2.8, 5.6, 2.1, Virginica, 129
7.2, 3, 5.8, 1.6, Virginica, 130
7.4, 2.8, 6.1, 1.9, Virginica, 131
```

```
7.9, 3.8, 6.4, 2, Virginica, 132
6.4, 2.8, 5.6, 2.2, Virginica, 133
6.3, 2.8, 5.1, 1.5, Virginica, 134
6.1, 2.6, 5.6, 1.4, Virginica, 135
7.7, 3, 6.1, 2.3, Virginica, 136
6.3, 3.4, 5.6, 2.4, Virginica, 137
6.4, 3.1, 5.5, 1.8, Virginica, 138
6, 3, 4.8, 1.8, Virginica, 139
6.9, 3.1, 5.4, 2.1, Virginica, 140
6.7, 3.1, 5.6, 2.4, Virginica, 141
6.9, 3.1, 5.1, 2.3, Virginica, 142
5.8, 2.7, 5.1, 1.9, Virginica, 143
6.8, 3.2, 5.9, 2.3, Virginica, 144
6.7, 3.3, 5.7, 2.5, Virginica, 145
6.7, 3, 5.2, 2.3, Virginica, 146
6.3, 2.5, 5, 1.9, Virginica, 147
6.5, 3, 5.2, 2, Virginica, 148
6.2, 3.4, 5.4, 2.3, Virginica, 149
5.9, 3, 5.1, 1.8, Virginica, 150
];
```

KMeansND - diagramfunktion

KMeansND() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring och för varje diagramrad visas kluster-ID för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1 och coordinate_2 osv. upp till n kolumner. Dessa är alla aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters.

KMeansND returnerar ett värde per datapunkt. Det returnerade värdet är en dual och är det heltalsvärde som motsvarar klustret varje datapunkt har tilldelats till.

Syntax:

```
KMeansND(num_clusters, num_iter, coordinate_1, coordinate_2 [,coordinate_3 [,...]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
num_clusters	Heltal som anger antalet kluster.
num_iter	Antalet upprepningar av klustring med ominitierade klustercenter.
coordinate_1	Aggregeringen som beräknar den första koordinaten, vanligtvis x-axeln (på ett spridningsdiagram som kan skapas från diagrammet). De ytterligare parametrarna beräknar den andra, tredje och fjärde koordinaten osv.

Exempel: diagramuttryck

I det här exemplet skapar vi ett spridningsdiagram med datauppsättningen *Iris*, och använder sedan KMeans för att färglägga data efter uttryck.

Vi skapar även en variabel för argumentet *num_clusters*, och sedan använder vi en variabelinmatningsruta för att ändra antalet kluster.

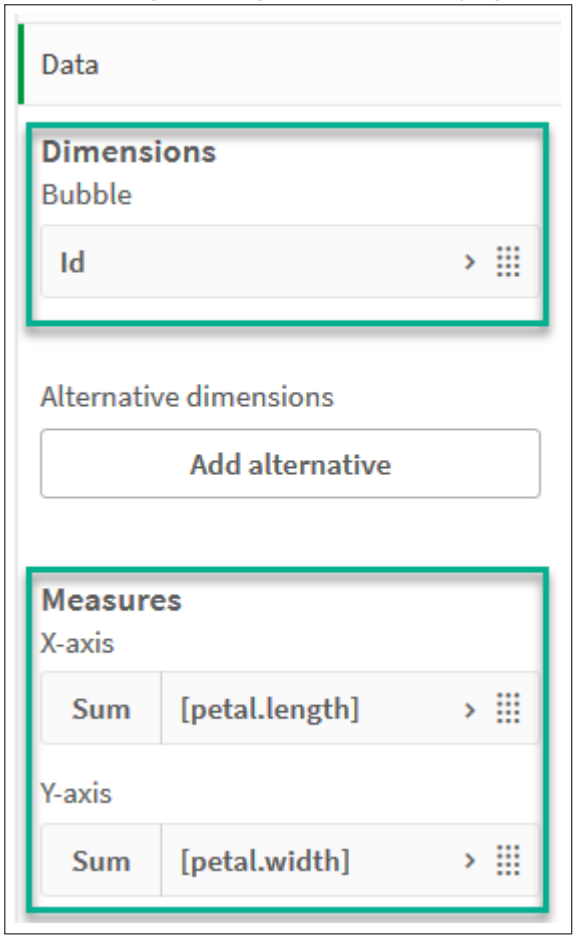
Vi skapar även en variabel för argumentet *num_iter*, och sedan använder vi en variabelinmatningsruta för att ändra antalet upprepningar.

Datauppsättningen *Iris* finns offentligt tillgänglig i en mängd format. Vi tillhandahåller dessa data som en inline-tabell som laddas med Skriptredigeraren i Qlik Sense. Observera att vi lagt till en *ID*-kolumn i datatabellen i det här exemplet.

När vi laddat data i Qlik Sense, gör vi följande:

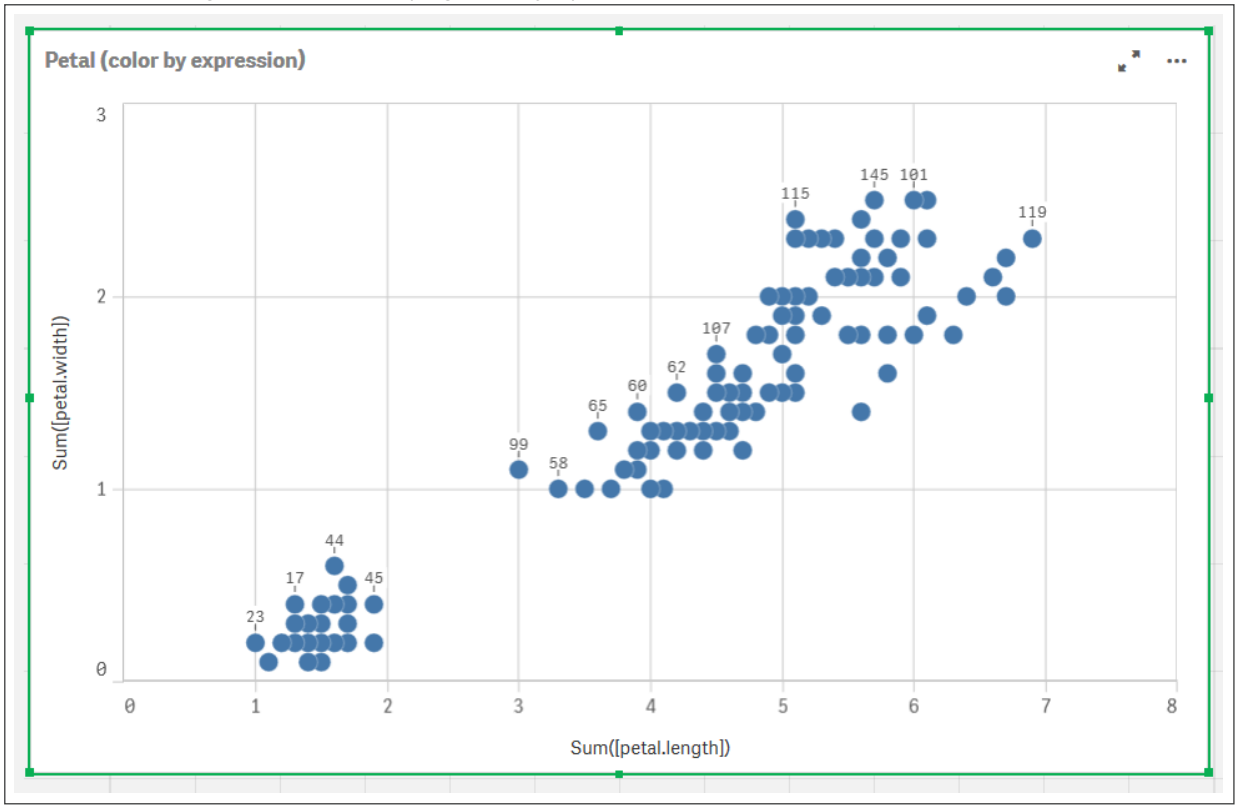
1. Dra ett **spridningsdiagram** till ett nytt ark. Döp diagrammet till *Kronblad (färg per uttryck)*.
2. Skapa en variabel för att ange antalet kluster. För variabeln **Namn** anger du *KmeansPetalClusters*. För variabeln **Definition** anger du *=2*.
3. Skapa en variabel för att ange antalet upprepningar. För variabeln **Namn** anger du *KmeansNumberIterations*. För variabeln **Definition** anger du *=1*.
4. Konfigurera **Data** för diagrammet:
    i. Under **Dimensioner** väljer du *id* för fältet **Bubbla**. Ange ett kluster-ID för Etikett.
    ii. Under **Mått** väljer du *Sum([petal.length])* för uttrycket för **X-axel**.
    iii. Under **Mått** väljer du *Sum([petal.width])* för uttrycket för **Y-axel**.

Datainställningar för diagrammet Kronblad (färg per uttryck)



Datapunkterna ritas ut på diagrammet.

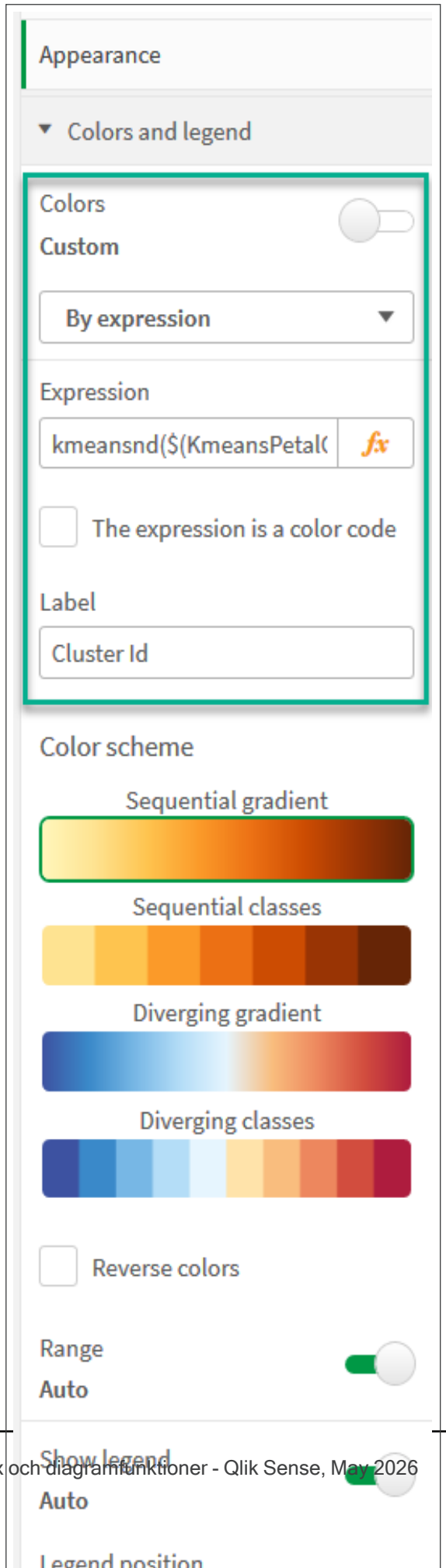
Datapunkter på diagrammet Kronblad (färg per uttryck)



5. Konfigurera **Utseende** för diagrammet:
    i. Under **Färger och teckenförklaring** väljer du **Anpassad** för **Färger**.
    ii. Välj att färglägga diagrammet**Per uttryck**.
    iii. Ange följande för **Uttryck**: *kmeansnd($(KmeansPetalClusters),$(KmeansNumberIterations), Sum([petal.length]), Sum([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
    Observera att *KmeansPetalClusters* är variabeln vi angav som *2*. *KmeansNumberIterations* är variabeln vi angav som *1*.
    Eller ange följande: *kmeansnd(2, 2, Sum([petal.length]), Sum([petal.width]),Sum ([sepal.length]), Sum([sepal.width]))*
    iv. Avmarkera kryssrutan **Uttrycket är en färgkod**.

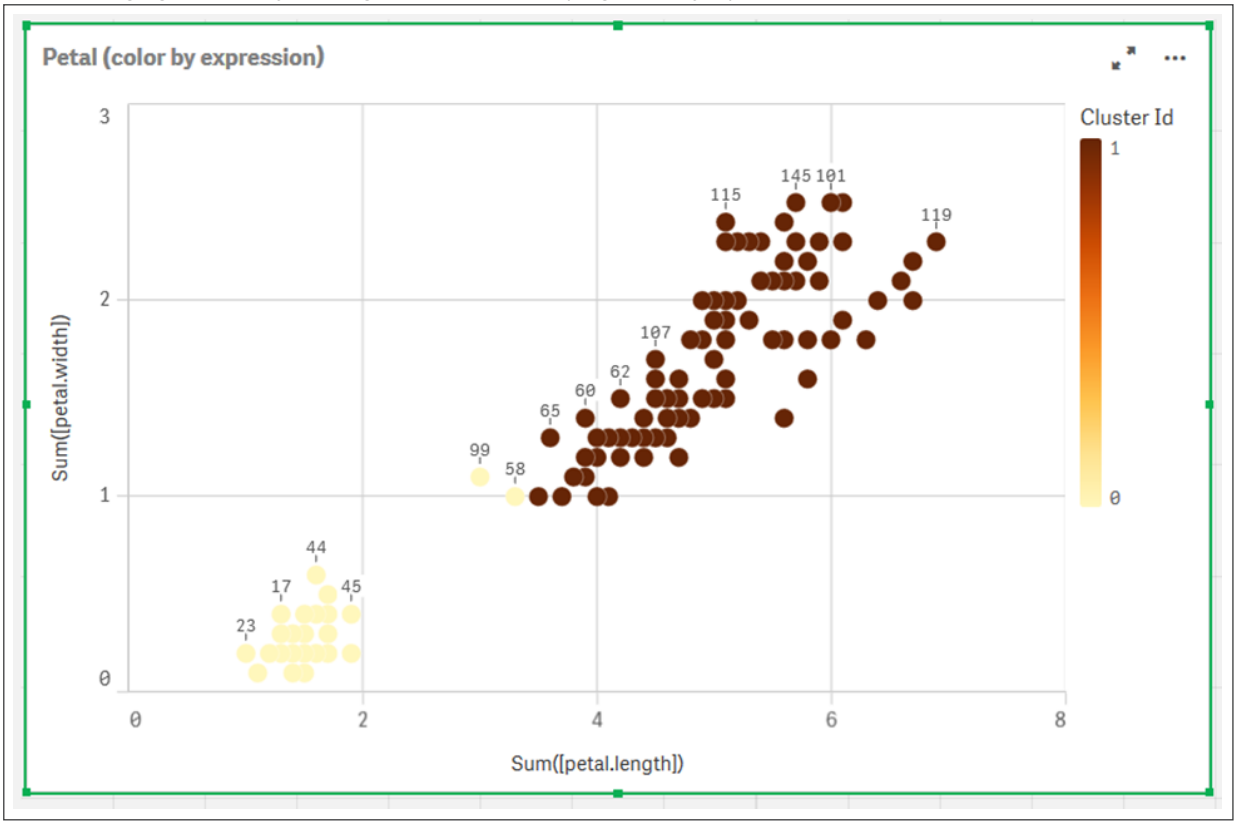
v. Ange följande för **Etikett**: *Kluster-ID*

Utseendeinställningar för diagrammet Petal (color by expression)


Appearance
Colors and legend
Colors
Custom
By expression
Expression
kmeansnd($(KmeansPetal(
The expression is a color code
Label
Cluster Id
Color scheme
Sequential gradient
Sequential classes
Diverging gradient
Diverging classes
Reverse colors
Range
Auto
Show legend
Auto
Legend position

De två klustren i diagrammet färgläggs per KMeans-uttrycket.

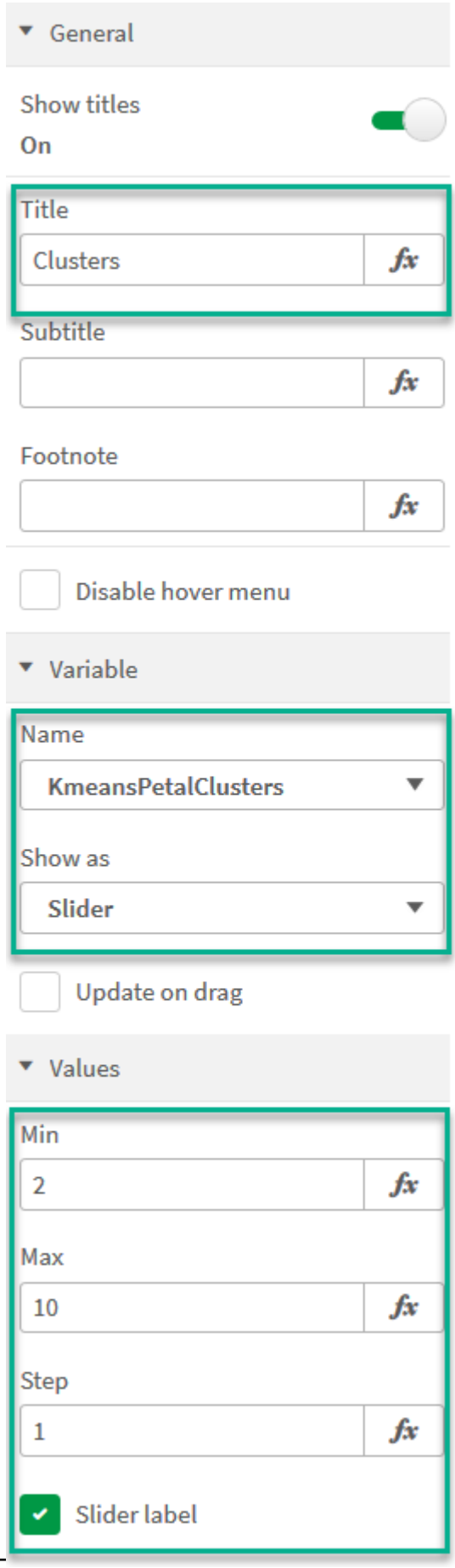
Klustren färglagda per uttryck i diagrammet Kronblad (färg per uttryck)



6. Lägg till en **Variabelinmatning**-ruta för antalet kluster.
   i. I **resurs**panelen, öppna **Diagram**.
   ii. Under **Instrumentpanelsobjekt**, dra en **Variabelinmatning**-ruta till arket.
   Om vi inte hade tillgång till dashboard-buntan kunde vi fortfarande ändra antalet kluster med hjälp av den variabel som vi skapade, eller direkt som ett heltal i uttrycket.
   iii. Under **Utseende**, klicka på **Allmänt**.
   iv. Ange följande för **Titel**: *Kluster*
   v. Klicka på **Variabel**.
   vi. Välj följande variabel för **Namn**: *KmeansPetalClusters*.
   vii. Välj **Reglage** för **Visa som**.

viii. Välj **Värden**, och konfigurera inställningarna efter behov,

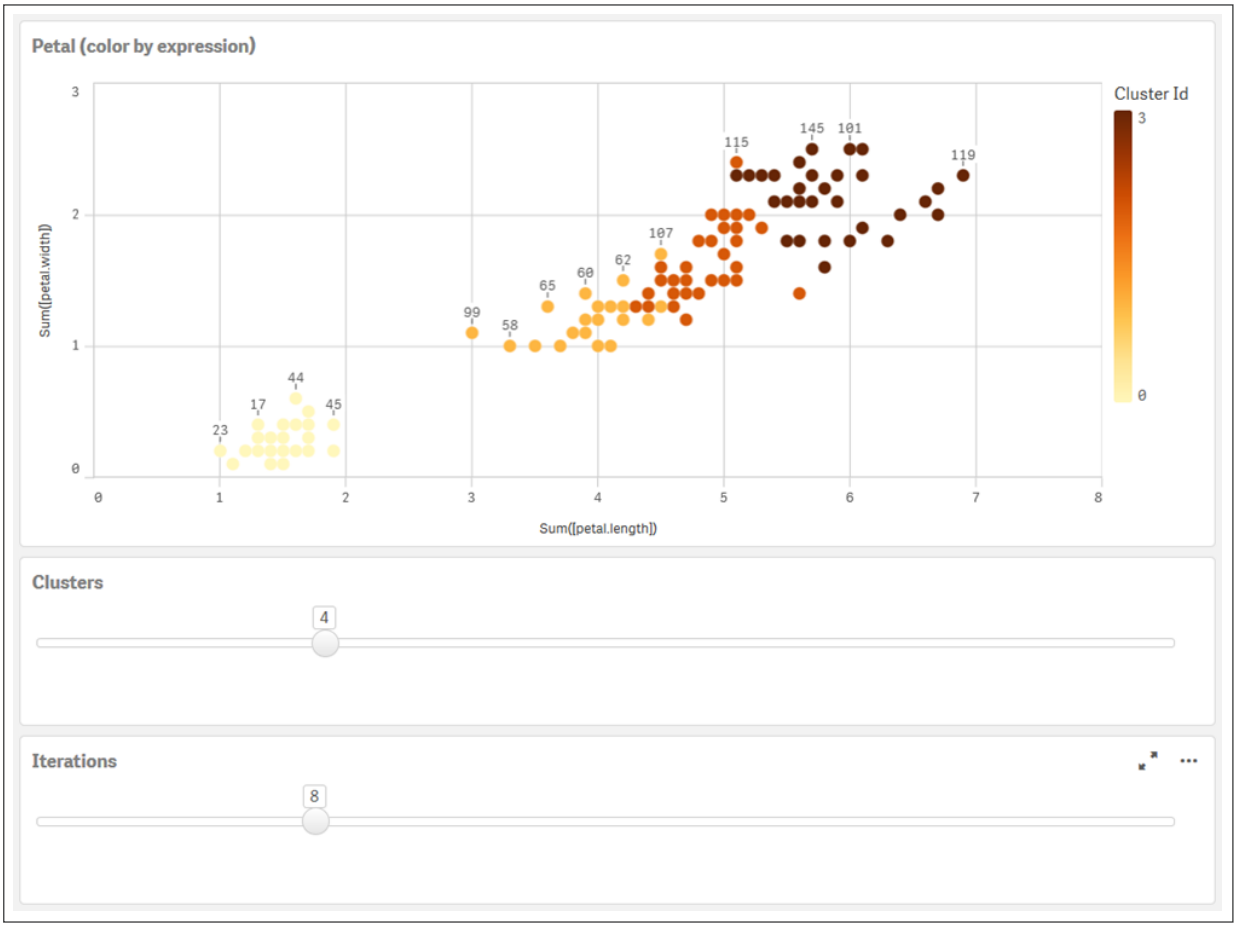
Utseende för Kluster variabel inmatningsruta


General
Show titles
On
Title
Clusters
Subtitle
Footnote
Disable hover menu
Variable
Name
KmeansPetalClusters
Show as
Slider
Update on drag
Values
Min
2
Max
10
Step
1
Slider label

7. Lägg till en ruta för **Variabelinmatning** för antalet upprepningar.
    i. Dra en ruta för **Variabelinmatning** till arket.
    ii. Under **Utseende** väljer du **Allmänt**.
    iii. Ange följande för **Rubrik**: *Upprepningar*
    iv. Under **Utseende** väljer du **Variabel**.
    v. Välj följande variabel under **Namn**: *KmeansNumberIterations*.
    vi. Konfigurera ytterligare inställningar efter behov,

    Vi kan nu ändra antalet kluster och upprepningar med hjälp av skjutreglagen i inmatningsrutorna för variabel.

    Klustren färglagda per uttryck i diagrammet Kronblad (färg per uttryck)

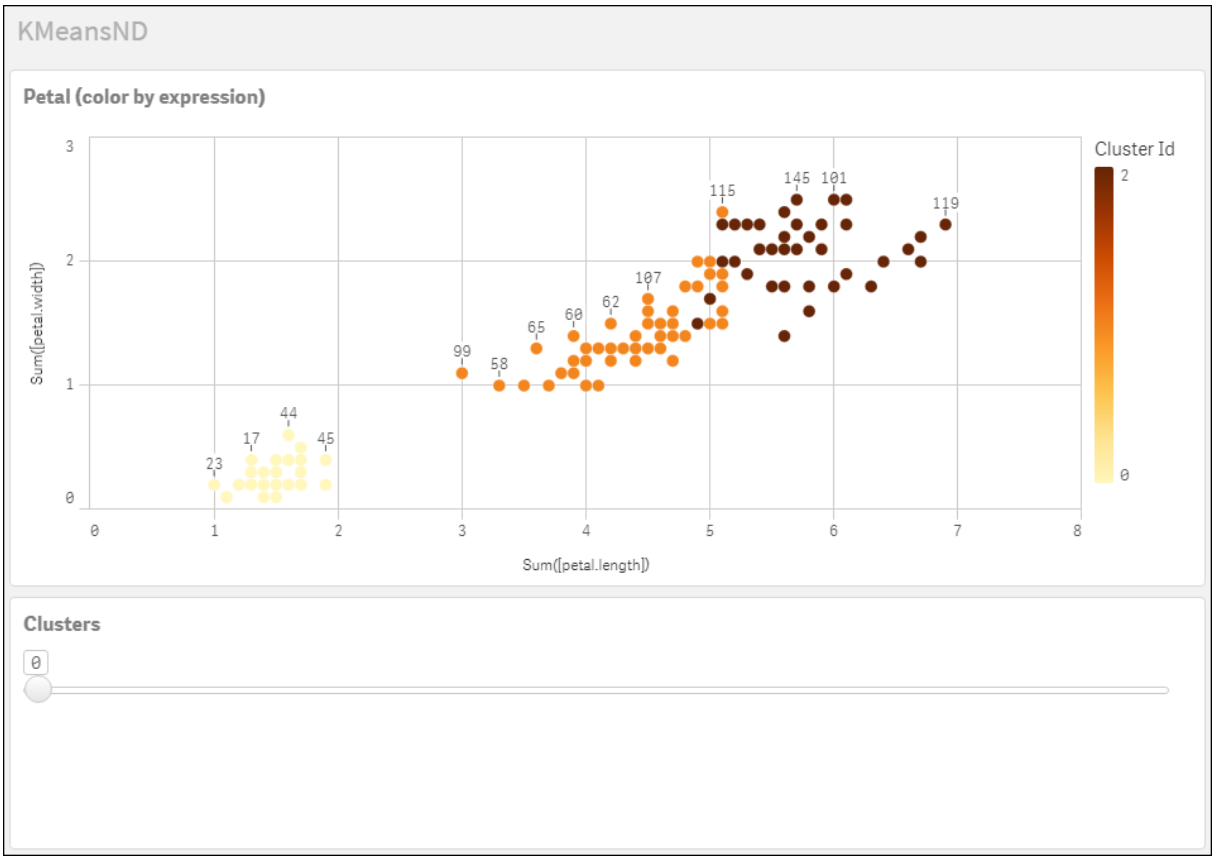


Automatisk klustring

KMeans-funktioner stöder automatisk klustring med en metod som kallas djupskillnad (depth difference, DeD). När användaren anger 0 som antal kluster bestäms ett optimalt antal kluster för den datauppsättningen. Observera att medan ett heltal som anges för antalet kluster (*k*) inte returneras explicit, beräknas det inom KMeans-algoritmen. Om till exempel 0 specificeras i funktionen för värdet av *KmeansPetalClusters* eller anges via en variabelindataruta, beräknas klustertilldelningar automatiskt för

datauppsättningen baserat på ett optimalt antal kluster. Med Iris-datauppsättningen; om 0 väljs för antalet kluster kommer algoritmen att bestämma (automatisk klustring) ett optimalt antal kluster (3) för denna datauppsättning.

Metoden KMeans-djupskillnad avgör det optimala antalet kluster när (k) är inställt på 0.



Iris-datauppsättning: inline-laddning för Dataladdningsredigeraren i Qlik Sense

```
IrisData:
Load * Inline [
sepal.length, sepal.width, petal.length, petal.width, variety, id
5.1, 3.5, 1.4, 0.2, Setosa, 1
4.9, 3, 1.4, 0.2, Setosa, 2
4.7, 3.2, 1.3, 0.2, Setosa, 3
4.6, 3.1, 1.5, 0.2, Setosa, 4
5, 3.6, 1.4, 0.2, Setosa, 5
5.4, 3.9, 1.7, 0.4, Setosa, 6
4.6, 3.4, 1.4, 0.3, Setosa, 7
5, 3.4, 1.5, 0.2, Setosa, 8
4.4, 2.9, 1.4, 0.2, Setosa, 9
4.9, 3.1, 1.5, 0.1, Setosa, 10
5.4, 3.7, 1.5, 0.2, Setosa, 11
4.8, 3.4, 1.6, 0.2, Setosa, 12
4.8, 3, 1.4, 0.1, Setosa, 13
4.3, 3, 1.1, 0.1, Setosa, 14
5.8, 4, 1.2, 0.2, Setosa, 15
5.7, 4.4, 1.5, 0.4, Setosa, 16
```

```
5.4, 3.9, 1.3, 0.4, Setosa, 17
5.1, 3.5, 1.4, 0.3, Setosa, 18
5.7, 3.8, 1.7, 0.3, Setosa, 19
5.1, 3.8, 1.5, 0.3, Setosa, 20
5.4, 3.4, 1.7, 0.2, Setosa, 21
5.1, 3.7, 1.5, 0.4, Setosa, 22
4.6, 3.6, 1, 0.2, Setosa, 23
5.1, 3.3, 1.7, 0.5, Setosa, 24
4.8, 3.4, 1.9, 0.2, Setosa, 25
5, 3, 1.6, 0.2, Setosa, 26
5, 3.4, 1.6, 0.4, Setosa, 27
5.2, 3.5, 1.5, 0.2, Setosa, 28
5.2, 3.4, 1.4, 0.2, Setosa, 29
4.7, 3.2, 1.6, 0.2, Setosa, 30
4.8, 3.1, 1.6, 0.2, Setosa, 31
5.4, 3.4, 1.5, 0.4, Setosa, 32
5.2, 4.1, 1.5, 0.1, Setosa, 33
5.5, 4.2, 1.4, 0.2, Setosa, 34
4.9, 3.1, 1.5, 0.1, Setosa, 35
5, 3.2, 1.2, 0.2, Setosa, 36
5.5, 3.5, 1.3, 0.2, Setosa, 37
4.9, 3.1, 1.5, 0.1, Setosa, 38
4.4, 3, 1.3, 0.2, Setosa, 39
5.1, 3.4, 1.5, 0.2, Setosa, 40
5, 3.5, 1.3, 0.3, Setosa, 41
4.5, 2.3, 1.3, 0.3, Setosa, 42
4.4, 3.2, 1.3, 0.2, Setosa, 43
5, 3.5, 1.6, 0.6, Setosa, 44
5.1, 3.8, 1.9, 0.4, Setosa, 45
4.8, 3, 1.4, 0.3, Setosa, 46
5.1, 3.8, 1.6, 0.2, Setosa, 47
4.6, 3.2, 1.4, 0.2, Setosa, 48
5.3, 3.7, 1.5, 0.2, Setosa, 49
5, 3.3, 1.4, 0.2, Setosa, 50
7, 3.2, 4.7, 1.4, Versicolor, 51
6.4, 3.2, 4.5, 1.5, Versicolor, 52
6.9, 3.1, 4.9, 1.5, Versicolor, 53
5.5, 2.3, 4, 1.3, Versicolor, 54
6.5, 2.8, 4.6, 1.5, Versicolor, 55
5.7, 2.8, 4.5, 1.3, Versicolor, 56
6.3, 3.3, 4.7, 1.6, Versicolor, 57
4.9, 2.4, 3.3, 1, Versicolor, 58
6.6, 2.9, 4.6, 1.3, Versicolor, 59
5.2, 2.7, 3.9, 1.4, Versicolor, 60
5, 2, 3.5, 1, Versicolor, 61
5.9, 3, 4.2, 1.5, Versicolor, 62
6, 2.2, 4, 1, Versicolor, 63
6.1, 2.9, 4.7, 1.4, Versicolor, 64
5.6, 2.9, 3.6, 1.3, Versicolor, 65
6.7, 3.1, 4.4, 1.4, Versicolor, 66
5.6, 3, 4.5, 1.5, Versicolor, 67
5.8, 2.7, 4.1, 1, Versicolor, 68
6.2, 2.2, 4.5, 1.5, Versicolor, 69
5.6, 2.5, 3.9, 1.1, Versicolor, 70
5.9, 3.2, 4.8, 1.8, Versicolor, 71
```

```
6.1, 2.8, 4, 1.3, Versicolor, 72
6.3, 2.5, 4.9, 1.5, Versicolor, 73
6.1, 2.8, 4.7, 1.2, Versicolor, 74
6.4, 2.9, 4.3, 1.3, Versicolor, 75
6.6, 3, 4.4, 1.4, Versicolor, 76
6.8, 2.8, 4.8, 1.4, Versicolor, 77
6.7, 3, 5, 1.7, Versicolor, 78
6, 2.9, 4.5, 1.5, Versicolor, 79
5.7, 2.6, 3.5, 1, Versicolor, 80
5.5, 2.4, 3.8, 1.1, Versicolor, 81
5.5, 2.4, 3.7, 1, Versicolor, 82
5.8, 2.7, 3.9, 1.2, Versicolor, 83
6, 2.7, 5.1, 1.6, Versicolor, 84
5.4, 3, 4.5, 1.5, Versicolor, 85
6, 3.4, 4.5, 1.6, Versicolor, 86
6.7, 3.1, 4.7, 1.5, Versicolor, 87
6.3, 2.3, 4.4, 1.3, Versicolor, 88
5.6, 3, 4.1, 1.3, Versicolor, 89
5.5, 2.5, 4, 1.3, Versicolor, 90
5.5, 2.6, 4.4, 1.2, Versicolor, 91
6.1, 3, 4.6, 1.4, Versicolor, 92
5.8, 2.6, 4, 1.2, Versicolor, 93
5, 2.3, 3.3, 1, Versicolor, 94
5.6, 2.7, 4.2, 1.3, Versicolor, 95
5.7, 3, 4.2, 1.2, Versicolor, 96
5.7, 2.9, 4.2, 1.3, Versicolor, 97
6.2, 2.9, 4.3, 1.3, Versicolor, 98
5.1, 2.5, 3, 1.1, Versicolor, 99
5.7, 2.8, 4.1, 1.3, Versicolor, 100
6.3, 3.3, 6, 2.5, Virginica, 101
5.8, 2.7, 5.1, 1.9, Virginica, 102
7.1, 3, 5.9, 2.1, Virginica, 103
6.3, 2.9, 5.6, 1.8, Virginica, 104
6.5, 3, 5.8, 2.2, Virginica, 105
7.6, 3, 6.6, 2.1, Virginica, 106
4.9, 2.5, 4.5, 1.7, Virginica, 107
7.3, 2.9, 6.3, 1.8, Virginica, 108
6.7, 2.5, 5.8, 1.8, Virginica, 109
7.2, 3.6, 6.1, 2.5, Virginica, 110
6.5, 3.2, 5.1, 2, Virginica, 111
6.4, 2.7, 5.3, 1.9, Virginica, 112
6.8, 3, 5.5, 2.1, Virginica, 113
5.7, 2.5, 5, 2, Virginica, 114
5.8, 2.8, 5.1, 2.4, Virginica, 115
6.4, 3.2, 5.3, 2.3, Virginica, 116
6.5, 3, 5.5, 1.8, Virginica, 117
7.7, 3.8, 6.7, 2.2, Virginica, 118
7.7, 2.6, 6.9, 2.3, Virginica, 119
6, 2.2, 5, 1.5, Virginica, 120
6.9, 3.2, 5.7, 2.3, Virginica, 121
5.6, 2.8, 4.9, 2, Virginica, 122
7.7, 2.8, 6.7, 2, Virginica, 123
6.3, 2.7, 4.9, 1.8, Virginica, 124
6.7, 3.3, 5.7, 2.1, Virginica, 125
7.2, 3.2, 6, 1.8, Virginica, 126
```

```
6.2, 2.8, 4.8, 1.8, Virginica, 127
6.1, 3, 4.9, 1.8, Virginica, 128
6.4, 2.8, 5.6, 2.1, Virginica, 129
7.2, 3, 5.8, 1.6, Virginica, 130
7.4, 2.8, 6.1, 1.9, Virginica, 131
7.9, 3.8, 6.4, 2, Virginica, 132
6.4, 2.8, 5.6, 2.2, Virginica, 133
6.3, 2.8, 5.1, 1.5, Virginica, 134
6.1, 2.6, 5.6, 1.4, Virginica, 135
7.7, 3, 6.1, 2.3, Virginica, 136
6.3, 3.4, 5.6, 2.4, Virginica, 137
6.4, 3.1, 5.5, 1.8, Virginica, 138
6, 3, 4.8, 1.8, Virginica, 139
6.9, 3.1, 5.4, 2.1, Virginica, 140
6.7, 3.1, 5.6, 2.4, Virginica, 141
6.9, 3.1, 5.1, 2.3, Virginica, 142
5.8, 2.7, 5.1, 1.9, Virginica, 143
6.8, 3.2, 5.9, 2.3, Virginica, 144
6.7, 3.3, 5.7, 2.5, Virginica, 145
6.7, 3, 5.2, 2.3, Virginica, 146
6.3, 2.5, 5, 1.9, Virginica, 147
6.5, 3, 5.2, 2, Virginica, 148
6.2, 3.4, 5.4, 2.3, Virginica, 149
5.9, 3, 5.1, 1.8, Virginica, 150
];
```

KMeansCentroid2D - diagramfunktion

KMeansCentroid2D() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring, och för varje diagramrad visas önskad koordinat för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1 respektive coordinate_2. Dessa är båda aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters. Data normaliseras med normparametern (valfritt).

KMeansCentroid2D returnerar ett värde per datapunkt. Det returnerade värdet är en dual och är en av koordinaterna för positionen som motsvarar klustercentret datapunkten har tilldelats till.

Syntax:

```
KMeansCentroid2D(num_clusters, coordinate_no, coordinate_1, coordinate_2 [, norm])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
num_clusters	Heltal som anger antalet kluster.

Argument	Beskrivning
coordinate_no	Önskat koordinatantal för centroiderna (motsvarande, till exempel, x-, y- eller z-axeln).
coordinate_1	Aggregeringen som beräknar den första koordinaten, vanligtvis x-axeln på spridningsdiagrammet som kan skapas från diagrammet. Den ytterligare parametern, coordinate_2, beräknar den andra koordinaten.
norm	Den valfria normaliseringsmetoden tillämpas på datauppsättningen före k-medelvärdesklustring. Möjliga värden: 0 eller ”none” för ingen normalisering 1 eller ”zscore” för z-poängsnormalisering 2 eller ”minmax” för min-max-normalisering Om ingen parameter anges eller om den angivna parametern är felaktig används ingen normalisering. Z-poäng normaliserar data baserat på funktionens median och standardavvikelse. Z-poäng säkerställer inte att varje funktion har samma skala men det fungerar bättre än min-max för behandling av outliers. Min-max-normalisering säkerställer att funktionerna har samma skala genom att ta de minsta och största värdena för varje och räkna om varje datapunkt.

Automatisk klustring

KMeans-funktioner stöder automatisk klustring med en metod som kallas djupskillnad (depth difference, DeD). När användaren anger 0 som antal kluster bestäms ett optimalt antal kluster för den datauppsättningen. Observera att medan ett heltal som anges för antalet kluster (*k*) inte returneras explicit, beräknas det inom KMeans-algoritmen. Om till exempel 0 specificeras i funktionen för värdet av *KmeansPetalClusters* eller anges via en variabelindataruta, beräknas klustertilldelningar automatiskt för datauppsättningen baserat på ett optimalt antal kluster.

KMeansCentroidND - diagramfunktion

KMeansCentroidND() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring och för varje diagramrad visas önskad koordinat för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1, coordinate_2 osv. upp till n kolumner. Dessa är alla aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters.

KMeansCentroidND returnerar ett värde per rad. Det returnerade värdet är en dual och är en av koordinaterna för positionen som motsvarar klustercentret datapunkten har tilldelats till.

Syntax:

```
KMeansCentroidND((num_clusters, num_iter, coordinate_no, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
num_clusters	Heltal som anger antalet kluster.
num_iter	Antalet upprepningar av klustring med ominitierade klustercenter.
coordinate_no	Önskat koordinatantal för centroiderna (motsvarande, till exempel, x-, y- eller z-axeln).
coordinate_1	Aggregeringen som beräknar den första koordinaten, vanligtvis x-axeln (på ett spridningsdiagram som kan skapas från diagrammet). De ytterligare parametrarna beräknar den andra, tredje och fjärde koordinaten osv.

Automatisk klustring

KMeans-funktioner stöder automatisk klustring med en metod som kallas djupskillnad (depth difference, DeD). När användaren anger 0 som antal kluster bestäms ett optimalt antal kluster för den datauppsättningen. Observera att medan ett heltal som anges för antalet kluster (*k*) inte returneras explicit, beräknas det inom KMeans-algoritmen. Om till exempel 0 specificeras i funktionen för värdet av *KmeansPetalClusters* eller anges via en variabelindataruta, beräknas klustertilldelningar automatiskt för datauppsättningen baserat på ett optimalt antal kluster.

STL_Trend - diagramfunktion

STL_Trend är en funktion för uppdelning av tidsserier. Tillsammans med **STL_Seasonal** och **STL_Residual** används denna funktion för att dela upp en tidsserie i säsongs-, trend- och residualkomponenter. För STL-algoritmen används uppdelning i tidsserier för att identifiera både återkommande säsongsvariationer och en allmän trend för ett givet indatamätetal och andra parametrar. **STL_Trend**-funktionen identifierar en allmän trend oberoende av säsongsmönster och cykler från tidsseriedata.

De tre STL-funktionerna är relaterade till indatamätetal genom en enkel summering:

STL_Trend + **STL_Seasonal** + **STL_Residual** = indatamätetal

STL (uppdelning per säsong och trend baserat på förlust) använder datautjämningstekniker och genom dess indataparametrar kan användare justera perioderna för beräkningarna som utförs. Dessa perioder fastställer hur tidsdimensionen för indatamåtten segmenteras i analysen.

För **STL_Trend** krävs minst ett indatamätvärde (`target_measure`) och ett heltalsvärde för dess `period_int`, och den returnerar ett flyttalsvärde. Indatamätetalet kommer att vara en aggregering som varierar längs tidsdimensionen. Du kan även inkludera värden för `seasonal_smoother` och `trend_smoother` för att justera den utjämnande algoritmen.

Du kan arbeta med den här funktionen genom att ange den direkt i uttrycksredigeraren för ett diagram

Syntax:

```
STL_Trend(target_measure, period_int [,seasonal_smoother [,trend_smoother]])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
target_measure	Måttet som ska delas upp i säsongskomponenter och trendkomponenter. Detta ska vara ett mått såsom Sum(Sales) eller Sum(Passengers), som varierar längs tidsdimensionen. Detta kan inte vara ett konstant värde.
period_int	Datauppsättningens tidsperiod. Den här parametern är ett heltalsvärde som representerar antal diskreta steg som ingår i en period eller en säsongscykel för signalen. Till exempel, om tidsserien är indelad i ett delavsnitt för varje kvartal av året så måste du ställa in **period_int** på 4 för att definiera tidsperioden till År.
seasonal_smoother	Säsongsutjämnarens längd. Detta ska vara ett heltal. Säsongsutjämnare använder data för en specifik fas i säsongsvariationen över ett antal perioder. Ett diskret steg av tidsdimensionen används från varje period. Säsongsutjämnaren anger antal perioder som används för utjämning. Till exempel om tidsdimensionen segmenteras per månad och perioden är År (12) så beräknas säsongskomponenten på så sätt att varje månad av året beräknas utifrån data från samma månad, både i det året och i näraliggande år. **seasonal_smoother**-värdet är antalet år som används för utjämning.
trend_smoother	Trendutjämnarens längd. Detta ska vara ett heltal. Trendutjämnaren använder samma tidsskala som **period_int**-parametern och dess värde är antalet partiklar som används för utjämning. Till exempel, om en tidsserie segmenteras per månad så är trendutjämnaren antalet månader som används för trendutjämning.

STL_Trend-diagramfunktionen används ofta i kombination med följande funktioner:

Relaterade funktioner

Funktion	Interaktion
STL_Seasonal - diagramfunktion (page 1672)	Detta är den funktion som används för att beräkna säsongskomponenten för en tidsserie.

Funktion	Interaktion
STL_Residual - diagramfunktion (page 1674)	När du delar upp ett indatamätetal i säsongs- och trendkomponenter så kommer en del av måttets variation inte att passa in inom någon av de två huvudkomponenterna. Funktionen **STL_Residual** beräknar den här delen av upplösningen.

En introduktionskurs med ett fullständigt exempel på hur man använder den här funktionen finns i *Introduktionskurs - upplösning av tidsserie i Qlik Sense (page 1676)*.

STL_Seasonal - diagramfunktion

STL_Seasonal är en funktion för uppdelning av tidsserier. Tillsammans med **STL_Trend** och **STL_Residual** används denna funktion för att dela upp en tidsserie i säsongs-, trend- och residualkomponenter. För STL-algoritmen används uppdelning i tidsserier för att identifiera både återkommande säsongsvariationer och en allmän trend för ett givet indatamätetal och andra parametrar. **STL_Seasonal**-funktionen kan identifiera ett säsongsmönster i en tidsserie och separera denna från den allmänna trenden som visas av data.

De tre STL-funktionerna är relaterade till indatamätetal genom en enkel summering:

STL_Trend + **STL_Seasonal** + **STL_Residual** = indatamätetal

STL (uppdelning per säsong och trend baserat på förlust) använder datautjämningstekniker och genom dess indataparametrar kan användare justera perioderna för beräkningarna som utförs. Dessa perioder fastställer hur tidsdimensionen för indatamåtten segmenteras i analysen.

För **STL_Seasonal** krävs minst ett indatamätvärde (`target_measure`) och ett heltalsvärde för dess `period_int`, och den returnerar ett flyttalsvärde. Indatamätetalet kommer att vara en aggregering som varierar längs tidsdimensionen. Du kan även inkludera värden för `seasonal_smoother` och `trend_smoother` för att justera den utjämnande algoritmen.

Du kan arbeta med den här funktionen genom att ange den direkt i uttrycksredigeraren för ett diagram

Syntax:

```
STL_Seasonal(target_measure, period_int [,seasonal_smoother [,trend_
smoother]])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
target_measure	Måttet som ska delas upp i säsongskomponenter och trendkomponenter. Detta ska vara ett mått såsom Sum(Sales) eller Sum(Passengers), som varierar längs tidsdimensionen. Detta kan inte vara ett konstant värde.
period_int	Datauppsättningens tidsperiod. Den här parametern är ett heltalsvärde som representerar antal diskreta steg som ingår i en period eller en säsongscykel för signalen. Till exempel, om tidsserien är indelad i ett delavsnitt för varje kvartal av året så måste du ställa in **period_int** på 4 för att definiera tidsperioden till År.
seasonal_smoother	Säsongsutjämnarens längd. Detta ska vara ett heltal. Säsongsutjämnare använder data för en specifik fas i säsongsvariationen över ett antal perioder. Ett diskret steg av tidsdimensionen används från varje period. Säsongsutjämnaren anger antal perioder som används för utjämning. Till exempel om tidsdimensionen segmenteras per månad och perioden är År (12) så beräknas säsongskomponenten på så sätt att varje månad av året beräknas utifrån data från samma månad, både i det året och i näraliggande år. **seasonal_smoother**-värdet är antalet år som används för utjämning.
trend_smoother	Trendutjämnarens längd. Detta ska vara ett heltal. Trendutjämnaren använder samma tidsskala som **period_int**-parametern och dess värde är antalet partiklar som används för utjämning. Till exempel, om en tidsserie segmenteras per månad så är trendutjämnaren antalet månader som används för trendutjämning.

STL_Seasonal-diagramfunktionen används ofta i kombination med följande funktioner:

Relaterade funktioner

Funktion	Interaktion
STL_Trend - diagramfunktion (page 1670)	Detta är den funktion som används för att beräkna trendkomponenten för en tidsserie.

Funktion	Interaktion
STL_Residual - diagramfunktion (page 1674)	När du delar upp ett indatamätetal i säsongs- och trendkomponenter så kommer en del av måttets variation inte att passa in inom någon av de två huvudkomponenterna. Funktionen **STL_Residual** beräknar den här delen av upplösningen.

En introduktionskurs med ett fullständigt exempel på hur man använder den här funktionen finns i *Introduktionskurs - upplösning av tidsserie i Qlik Sense (page 1676)*.

STL_Residual - diagramfunktion

STL_Residual är en funktion för uppdelning av tidsserier. Tillsammans med **STL_Seasonal** och **STL_Trend** används denna funktion för att dela upp en tidsserie i säsongs-, trend- och residualkomponenter. För STL-algoritmen används uppdelning i tidsserier för att identifiera både återkommande säsongsvariationer och en allmän trend för ett givet indatamätetal och andra parametrar. När denna åtgärd utförs kommer en del av indatamätetalen varken passa in för säsongs- eller trendkomponenten och kommer att definieras som residualkomponenten. **STL_Residual**-diagramfunktionen fångar upp denna del av beräkningen.

De tre STL-funktionerna är relaterade till indatamätetal genom en enkel summering:

STL_Trend + **STL_Seasonal** + **STL_Residual** = indatamätetal

STL (uppdelning per säsong och trend baserat på förlust) använder datautjämningstekniker och genom dess indataparametrar kan användare justera perioderna för beräkningarna som utförs. Dessa perioder fastställer hur tidsdimensionen för indatamåtten segmenteras i analysen.

Eftersom en uppdelning av tidsserier främst letar efter säsongsrelaterade och allmänna variationer i data så anses residualkomponenten vara den minst viktiga av de tre. En skev eller periodisk residualkomponent kan dock bidra till att identifiera problem i beräkningen, som t.ex. fel tidsperiodinställningar.

För **STL_Residual** krävs minst ett indatamätetal (`target_measure`) och ett heltalsvärde för dess `period_int` och den returnerar ett flyttalsvärde. Indatamätetalet kommer att vara en aggregering som varierar längs tidsdimensionen. Du kan även inkludera värden för `seasonal_smoother` och `trend_smoother` för att justera den utjämnande algoritmen.

Du kan arbeta med den här funktionen genom att ange den direkt i uttrycksredigeraren för ett diagram

Syntax:

```
STL_Residual(target_measure, period_int [,seasonal_smoother [,trend_
smoother]])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
target_ measure	Måttet som ska delas upp i säsongskomponenter och trendkomponenter. Detta ska vara ett mått såsom Sum(Sales) eller Sum(Passengers), som varierar längs tidsdimensionen. Detta kan inte vara ett konstant värde.
period_int	Datauppsättningens tidsperiod. Den här parametern är ett heltalsvärde som representerar antal diskreta steg som ingår i en period eller en säsongscykel för signalen. Till exempel, om tidsserien är indelad i ett delavsnitt för varje kvartal av året så måste du ställa in **period_int** på 4 för att definiera tidsperioden till År.
seasonal_ smoother	Säsongsutjämnarens längd. Detta ska vara ett heltal. Säsongsutjämnare använder data för en specifik fas i säsongsvariationen över ett antal perioder. Ett diskret steg av tidsdimensionen används från varje period. Säsongsutjämnaren anger antal perioder som används för utjämning. Till exempel om tidsdimensionen segmenteras per månad och perioden är År (12) så beräknas säsongskomponenten på så sätt att varje månad av året beräknas utifrån data från samma månad, både i det året och i näraliggande år. **seasonal_smoother**-värdet är antalet år som används för utjämning.
trend_ smoother	Trendutjämnarens längd. Detta ska vara ett heltal. Trendutjämnaren använder samma tidsskala som **period_int**-parametern och dess värde är antalet partiklar som används för utjämning. Till exempel, om en tidsserie segmenteras per månad så är trendutjämnaren antalet månader som används för trendutjämning.

STL_Residualdiagramfunktionen används ofta i kombination med följande funktioner:

Relaterade funktioner

Funktion	Interaktion
STL_Seasonal - diagramfunktion (page 1672)	Detta är den funktion som används för att beräkna säsongskomponenten för en tidsserie.
STL_Trend - diagramfunktion (page 1670)	Detta är den funktion som används för att beräkna trendkomponenten för en tidsserie.

En introduktionskurs med ett fullständigt exempel på hur man använder den här funktionen finns i *Introduktionskurs – upplösning av tidsserie i Qlik Sense (page 1676)*.

Introduktionskurs – upplösning av tidsserie i Qlik Sense

Den här introduktionskursen demonstrerar hur man använder tre diagramfunktioner för att lösa upp en tidsserie med STL-algoritmen.

Den här introduktionskursen använder tidsseriedata för det antal passagerare som använder ett flygbolag varje månad för att demonstrera hur STL-algoritmen fungerar. Diagramfunktionerna **STL_Trend**, **STL_Seasonal** och **STL_Residual** kommer att användas för att skapa visualiseringarna. Mer information om upplösning av tidsserier i Qlik Sense finns i *Funktioner för uppdelning av tidsserier (page 1621)*.

Skapa en app

Börja genom att skapa en ny app och importera datauppsättningen till den.

Ladda ned den här datauppsättningen:

Introduktionskurs – upplösning av tidsserie

Den här filen innehåller data om ett flygbolags passagerarantal per månad.

Gör följande:

1. Från hubben klickar du på **Skapa ny app**.
2. Öppna appen och släpp *Tutorial - Time series decomposition.csv* i den.

Förbereda och ladda data

För att Qlik Sense ska kunna tolka fältet YearMonth korrekt kan du behöva använda Datahanteraren för att identifiera fältet som ett datafält i stället för ett fält med strängvärden. Det här steget hanteras normalt automatiskt, men i det här fallet presenteras data i det lite ovanliga formatet *ÅÅÅÅ-MM*.

1. I Datahanteraren markerar du tabellen och klickar på .
2. Med fältet *YearMonth* markerat klickar du på och anger **Fälttyp** till **Datum**.
3. Under **Indataformat** anger du *YYYY-MM*.
4. Under **Visningsformat** anger du *YYYY-MM*. Klicka på **OK**.
   Fältet ska nu visa kalenderikonen.
5. Klicka på **Ladda data**.

Nu är du klar att börja använda STL-funktionerna för att representera dina data visuellt.

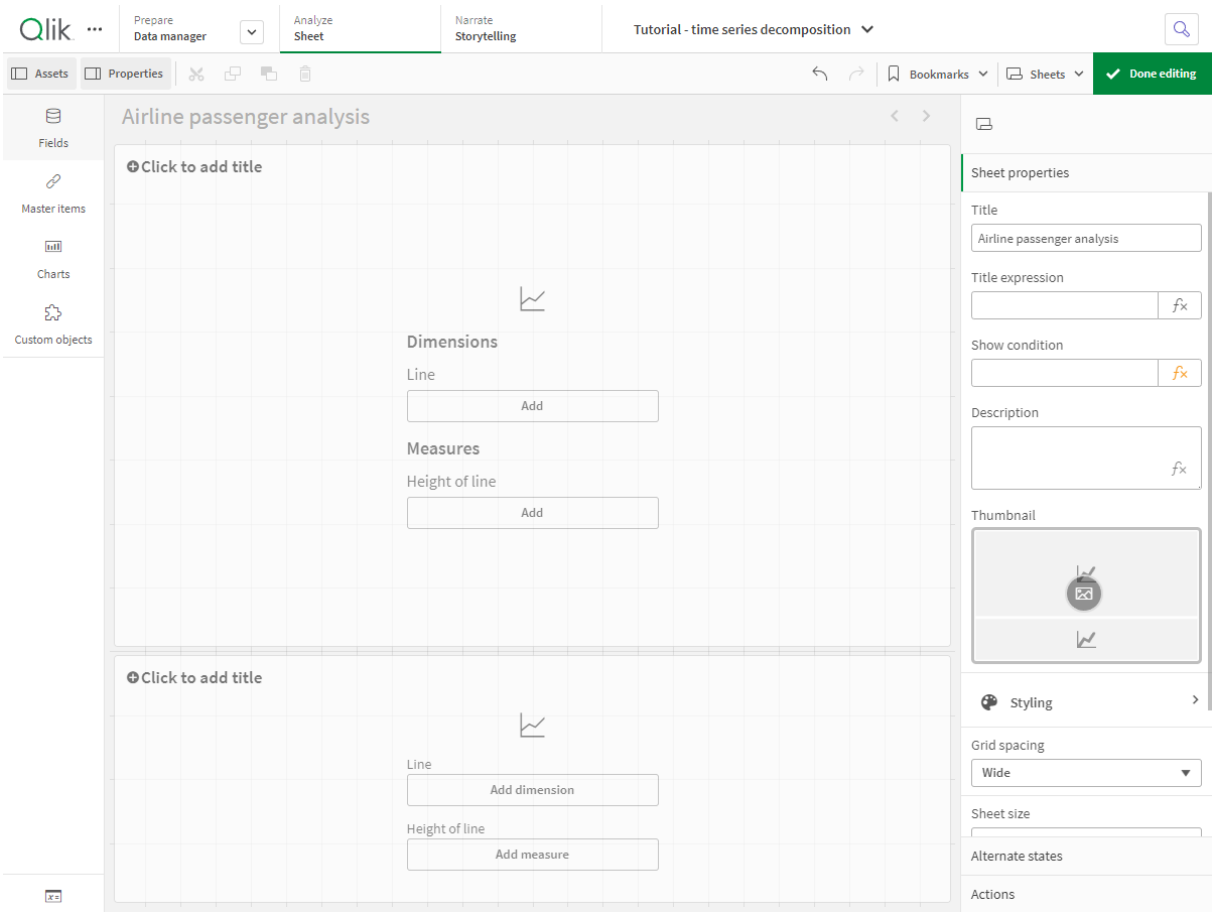
Skapa visualiseringarna

Sedan skapar du två linjediagram för att demonstrera hur diagramfunktionerna **STL_Trend**, **STL_Seasonal** och **STL_Residual** fungerar.

Öppna ett nytt ark och ge det en titel.

Lägg till två linjediagram till arket. Ändra storlek och placering på diagrammen så att de stämmer med följande bild.

Qlik Sense-rutnätskontur för ett tomt appark



Första linjediagrammet: trend- och säsongskomponenter

Gör följande:

1. Lägg till titeln *Säsong och trend* till det första diagrammet.
2. Lägg till *YearMonth* som dimension och ge den etiketten *Datum*.
3. Lägg till följande mått och ge det etiketten *Passagerare per månad*:
   =Sum(Passengers)
4. Under **Data** expanderar du mätvärdet *Passagerare per månad* och klickar på **Lägg till trendlinje**.
5. Ange **Typ** till **Linjär**.
   Du kommer att jämföra den här trendlinjen med trendkomponentens utjämnade utdata.
6. Lägg till följande mätvärde för att rita upp trendkomponenten och ge det etiketten *Trend*:
   =STL_Trend(SUM(Passengers), 12)
7. Lägg sedan till följande mätvärde för att rita upp säsongskomponenten och ge det etiketten *Säsong*:
   =STL_Seasonal(SUM(Passengers), 12)
8. Under **Utseende** > **Presentation** anger du **Bläddringslist** till **Ingen**.
9. Behåll standardfärgerna eller ändra dem efter eget önskemål.

Andra linjediagrammet: överskottskomponent

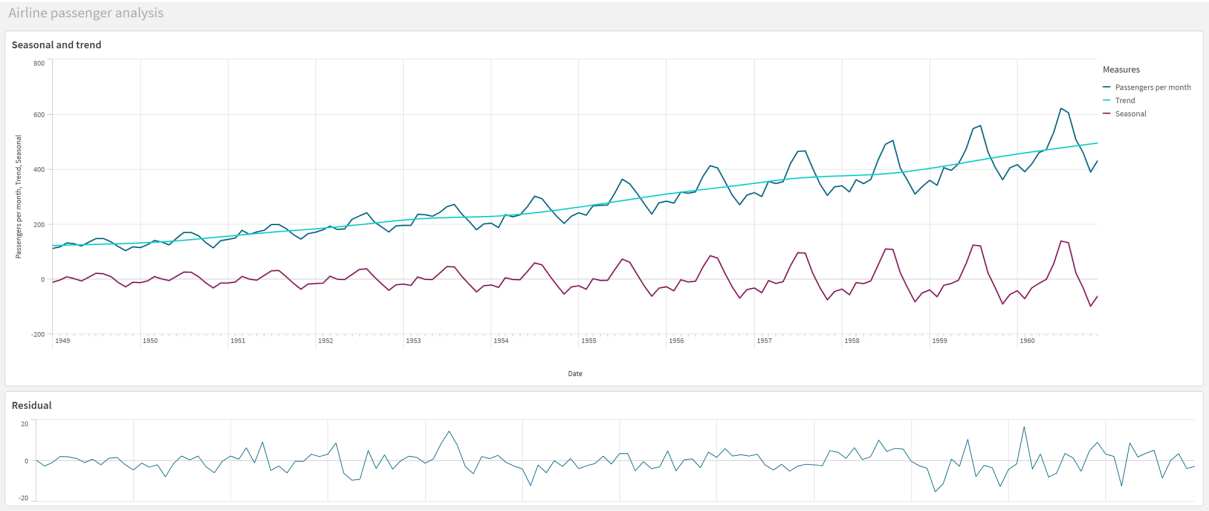
Sedan konfigurerar du det andra linjediagrammet. Den här visualiseringen kommer att visa tidsseriens överskottskomponent.

Gör följande:

1. Dra ett linjediagram till arket. Lägg till titeln *Överskott*.
2. Lägg till *Datum* som dimension.
3. Lägg till följande mätvärde och ge det etiketten *Överskott*:
   =STL_Residual(SUM(Passengers), 12)
4. Under **Utseende** > **Presentation** anger du **Bläddringslist** till **Ingen**.

Ditt ark ska nu likna det här nedanför.

Qlik Sense-ark för analys av flygpassagerare



Tolka och förklara data

Med STL-diagramfunktionerna kan vi få ett antal insikter från våra tidsseriedata.

Trendkomponent

Den statistiska informationen i trendkomponenten har rensats från säsongsdata. Detta gör det lättare att se allmänna fluktuationer över tid som inte upprepas. Jämfört med den räta, linjära trendlinjen för *Passagerare per månad* fångar STL-trendkomponenten upp trender som förändras. Den visar några tydliga avvikelser, samtidigt som den presenterar informationen på ett läsbart sätt. De utjämnande funktionerna i STL-algoritmen hjälpte till att framhäva detta.

De lägre nivåer på antalet flygpassagerare som syns i STL-trenddiagrammet kan förklaras som en del av den ekonomiska påverkan från lågkonjunkturer som inträffade under 1950-talet.

Säsongskomponent

Den trendrensade säsongskomponenten isolerade återkommande fluktuationer genom tidsserien, och tog bort allmän trendinformation från den delen av analysen. Vi började med en datauppsättning som bestod av aggregerade data över år och månader. Med dessa data är det implicit att vi segmenterar dem i enmånadspartiklar. Genom att definiera periodvärdet 12 anger vi att diagrammet ska modellera säsongsmönster under ettårscykler (tolv månader).

I dessa data finns det ett upprepat säsongsmönster av ett ökat antal flygpassagerare under sommarmånaderna, följt av minskningar under vintermånaderna. Detta stämmer med att sommaren normalt är en populär tid att ta semester och resa. Vi ser också att under tidsserien ökar dessa säsongscykler dramatiskt i amplitud.

Överskottskomponent

Diagrammet för överskottskomponenten visar all information som inte fångades upp i trend- och säsongsupplösningen. Överskottskomponenten innefattar statistiskt brus, men den kan också tyda på en felaktig inställning av argumenten för STL-funktionerna för trend och säsong. I allmänhet gäller att om det finns periodiska svängningar i signalens överskottskomponent, eller om den information som visas

uppenbarligen inte är slumpmässig, är det oftast ett tecken på att det finns information i tidsserien som för närvarande inte fångas upp i säsongs- eller trendkomponenterna. I så fall måste du gå tillbaka till definitionerna av varje funktionsargument och kanske ändra periodiciteten.

Jämnare värden

Eftersom vi inte angav några värden för trend- och säsongsutjämningen använder funktionen standardvärden för dessa parametrar. I Qlik Sense ger standardvärdena för utjämning i STL-algoritmen effektiva resultat. Därför kan i de flesta fall dessa argument utelämnas i uttrycken.

Om argumenten för säsongs- eller trendutjämning anges till 0 i någon av de tre STL-funktionerna använder algoritmen standardvärden i stället för värdena 0.

Trendutjämningsvärdet använder den dimension som anges i diagrammet. Eftersom fältet *YearMonth* presenterar data per månad kommer trendutjämningsvärdet att vara antalet månader. Säsongsutjämningen använder den periodicitet som definieras. I det här fallet definierade vi att en period varar i tolv månader (ett år), så säsongsutjämningsvärdet är antalet år. Detta kanske låter förvirrande, men det betyder bara att om vi vill hitta ett säsongsberoende måste vi titta över ett antal säsonger. Detta antal är säsongsutjämningsvärdet.

Annan användbar information

Eftersom säsongscyklerna ökar i amplitud med tiden, kan en mer avancerad analys använda logaritmfunktioner för att skapa en multiplikativ upplösning. I praktiken kan man skapa ett enkelt mått på den relativa amplituden i Qlik Sense genom att dividera säsongskomponenten med trendkomponenten. När vi gör detta ser vi att med tiden får sommartopparna i varje cykel en större relativ amplitud. Amplituden på dalarna under vintern blir dock inte större med tiden.

8.23 Statistiska fördelningsfunktioner

Funktioner för statistisk fördelning returnerar sannolikheten för förekomsten av olika möjliga resultat för en given inmatad variabel. Du kan använda dessa funktioner för att beräkna det potentiella värdet för dina datapunkter.

De tre grupperna av statistiska fördelningsfunktioner som beskrivs nedan har implementerats i Qlik Sense med Cephes-funktionsbiblioteket. För referenser och mer information om de algoritmer som används, exakthet och så vidare, se: ≤ Cephes library. Cephes-funktionsbiblioteket används med tillstånd.

- Funktioner för sannolikhet beräknar sannolikheten vid fördelningspunkten för det angivna värdet.
    - Funktioner för frekvens används för diskreta fördelningar.
    - Funktioner för täthet används för kontinuerliga funktioner.
- Funktioner för Dist beräknar den ackumulerade sannolikheten för fördelningen vid den punkt i fördelningen som anges av det angivna värdet.
- Funktioner för Inv beräknar det inverterade värdet, baserat på den ackumulerade sannolikheten för fördelningen.

Alla funktioner kan användas både i dataladdningsskriptet och diagramuttryck.

Översikt av statistiska fördelningsfunktioner

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Betatäthet

`BetaDensity()` returnerar sannolikheten för betafördelningen.

```
BetaDensity (value, alpha, beta)
```

Betafördelning

`BetaDist()` returnerar den ackumulerade sannolikheten för betafördelningen.

```
BetaDist (value, alpha, beta)
```

Betainvers

`BetaINV()` returnerar inversionen av den ackumulerade sannolikheten för betafördelningen.

```
BetaInv (prob, alpha, beta)
```

Binomialfördelning

`BinomDist()` returnerar den ackumulerade sannolikheten för binomialfördelningen.

```
BinomDist (value, trials, trial_probability)
```

Binomialfrekvens

`BinomFrequency()` returnerar binomialfördelningen.

```
BinomFrequency (value, trials, trial_probability)
```

BinomInv

`BinomInv()` returnerar inversionen av den ackumulerade sannolikheten för binomialfördelningen.

```
BinomInv (prob, trials, trial_probability)
```

ChiDensity

`ChiDensity()` returnerar den ensidiga sannolikheten för chi^2-fördelningen. Funktionen Chi^2-densitet är associerad med ett chi^2-test.

```
ChiDensity (value, degrees_freedom)
```

ChiDist

`ChiDist()` returnerar den ensidiga sannolikheten för chi^2-fördelningen. Funktionen Chi^2-densitet är associerad med ett chi^2-test.

```
ChiDist (value, degrees_freedom)
```

ChiInv

`ChiInv()` returnerar motsatsen till den ensidiga sannolikheten hos chi^2-fördelningen.

```
ChiInv (prob, degrees_freedom)
```

FDensity

`FDensity()` returnerar sannolikheten för F-fördelningen.

```
FDensity (value, degrees_freedom1, degrees_freedom2)
```

FDist

`FDist()` returnerar den ackumulerade sannolikheten för F-fördelningen.

```
FDist (value, degrees_freedom1, degrees_freedom2)
```

FInv

`FInv()`returnerar inversionen av den ackumulerade sannolikheten för F-fördelningen.

```
FInv (prob, degrees_freedom1, degrees_freedom2)
```

GammaDensity

`GammaDensity()` returnerar sannolikheten för gammafördelningen.

```
GammaDensity (value, k, θ)
```

GammaDist

`GammaDist()` returnerar den ackumulerade sannolikheten för gammafördelningen.

```
GammaDist (value, k, θ)
```

GammaInv

`GammaInv()` returnerar inversionen av den ackumulerade sannolikheten för gammafördelningen.

```
GammaInv (prob, k, θ)
```

NormDist

`NormDist()` returnerar den kumulativa normalfördelningen hos det angivna medelvärdet och standardavvikelsen. Om mean = 0 och standard_dev = 1, returnerar funktionen standardnormalfördelningen.

```
NormDist (value, mean, standard_dev)
```

NormInv

`NormInv()` returnerar inversen av den normala kumulativa fördelningen för det angivna medelvärdet och standardavvikelsen.

```
NormInv (prob, mean, standard_dev)
```

PoissonDist

`PoissonDist()` returnerar den ackumulerade sannolikheten för Poisson-fördelningen.

```
PoissonDist (value, mean)
```

Poissonfrekvens

`PoissonFrequency()` returnerar Poisson-fördelningen.

```
PoissonFrequency (value, mean)
```

PoissonInv

`PoissonInv()` returnerar inversionen av den ackumulerade sannolikheten för Poisson-fördelningen.

```
PoissonInv (prob, mean)
```

TDensitet

`TDensity()` returnerar värdet för studentens `t`-täthetsfunktion där ett numeriskt värde är ett beräknat värde av `t`, vars sannolikhet ska beräknas.

```
TDensity (value, degrees_freedom, tails)
```

TDist

`TDist()` returnerar sannolikheten hos studentens `t`-fördelning där ett numeriskt värde är ett beräknat värde av `t` vars sannolikhet ska kalkyleras.

```
TDist (value, degrees_freedom, tails)
```

TInv

`TInv()` returnerar `t`-värdet hos studentens `t`-fördelning som en funktion av sannolikhet och frihetsgrader.

```
TInv (prob, degrees_freedom)
```

Se även:

p *Statistiska aggregeringsfunktioner (page 462)*

BetaDensity

`BetaDensity()` returnerar sannolikheten för betafördelningen.

Syntax:

```
BetaDensity(value, alpha, beta)
```

Returnerad datatyp: tal

Argument

Argument	Beskrivning
`value`	Det värde där fördelningen ska utvärderas. Värdet måste vara mellan 0 och 1.
`alpha`	Ett positivt tal definierar den första formens parameter. Det är exponenten för den slumpmässiga variabeln
`beta`	Ett positivt tal definierar den andra formens parameter. Det anger talet för nämnarens frihetsgrad.

BetaDist

`BetaDist()` returnerar den ackumulerade sannolikheten för betafördelningen.

Syntax:

```
BetaDist(value, alpha, beta)
```

Returnerad datatyp: tal

Argument

Argument	Beskrivning
value	Det värde där fördelningen ska utvärderas. Värdet måste vara mellan 0 och 1.
alpha	Ett positivt tal definierar den första formens parameter. Det är exponenten för den slumpmässiga variabeln
beta	Ett positivt tal definierar den andra formens parameter. Det är exponenten som styr formen av fördelning.

Denna funktion hänger samman med funktionen `BetaInv` på följande sätt:
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BetaInv

`BetaINV()` returnerar inversionen av den ackumulerade sannolikheten för betafördelningen.

Syntax:

```
BetaInv(prob, alpha, beta)
```

Returnerad datatyp: tal

Argument

Argument	Beskrivning
prob	En sannolikhet som associeras med Beta-sannolikhetsfördelningen. Måste vara ett tal mellan 0 och 1.
alpha	Ett positivt tal definierar den första formens parameter. Det är exponenten för den slumpmässiga variabeln
beta	Ett positivt tal definierar den andra formens parameter. Det är exponenten som styr formen av fördelning.

Denna funktion hänger samman med funktionen `BetaDist` på följande sätt:
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BinomDist

`BinomDist()` returnerar den ackumulerade sannolikheten för binomialfördelningen.

Syntax:

```
BinomDist(value, trials, trial_probability)
```

Returnerad datatyp: tal

Argument

Argument	Beskrivning
`value`	Det värde där fördelningen ska utvärderas. Värdet måste vara ett heltal som är större än noll och inte större än antalet testperioder.
`trials`	Ett positivt heltal som anger antalet testperioder.
`trial_probability`	Sannolikheten för framgång i respektive testperiod. Den är alltid ett tal mellan 0 och 1.

Denna funktion hänger samman med funktionen `BinomInv` på följande sätt:
`If prob = BinomDIST(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value`

BinomFrequency

`BinomFrequency()` returnerar binomialfördelningen.

Syntax:

```
BinomFrequency(value, trials, trial_probability)
```

Returnerad datatyp: tal

Argument

Argument	Beskrivning
`value`	Det värde där fördelningen ska utvärderas. Värdet måste vara ett heltal som är större än noll och inte större än antalet försök.
`trials`	Ett positivt heltal som anger antalet testperioder.
`trial_probability`	Sannolikheten för framgång i respektive testperiod. Den är alltid ett tal mellan 0 och 1.

BinomInv

`BinomInv()` returnerar inversionen av den ackumulerade sannolikheten för binomialfördelningen.

Syntax:

```
BinomInv(prob, trials, trial_probability)
```

Returnerad datatyp: tal

Argument

Argument	Beskrivning
prob	En sannolikhet som associeras med Binomial-sannolikhetsfördelningen. Måste vara ett tal mellan 0 och 1.
trials	Ett positivt heltal som anger antalet testperioder.
trial_probability	Sannolikheten för framgång i respektive testperiod. Den är alltid ett tal mellan 0 och 1.

Denna funktion hänger samman med funktionen `BinomDist` på följande sätt:
`If prob = BinomDist(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value`

ChiDensity

`ChiDensity()` returnerar den ensidiga sannolikheten för chi^2-fördelningen. Funktionen Chi^2-densitet är associerad med ett chi^2-test.

Syntax:

```
ChiDensity(value, degrees_freedom)
```

Returnerad datatyp: tal

Argument

Argument	Beskrivning
value	Det värde där fördelningen ska utvärderas. Värdet får inte vara negativt.
degrees_freedom	Ett positivt heltal som anger täljarens frihetsgrad.

ChiDist

`ChiDist()` returnerar den ensidiga sannolikheten för chi^2-fördelningen. Funktionen Chi^2-densitet är associerad med ett chi^2-test.

Syntax:

```
CHIDIST(value, degrees_freedom)
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
value	Det värde där fördelningen ska utvärderas. Värdet får inte vara negativt.
degrees_freedom	Ett positivt heltal som anger frihetsgrad.

Denna funktion hänger samman med funktionen **ChiInv** på följande sätt:
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Exempel och resultat:

Exempel	Resultat
CHIDIST(8, 15)	Returnerar 0.9238

ChiInv

`ChiInv()` returnerar motsatsen till den ensidiga sannolikheten hos chi^2-fördelningen.

Syntax:

```
CHIINV(prob, degrees_freedom)
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
prob	En sannolikhet som har samband med chi^2-fördelningen. Måste vara ett tal mellan 0 och 1.
degrees_ freedom	Ett heltal som anger frihetsgrad.

Denna funktion hänger samman med funktionen **ChiDist** på följande sätt:
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Exempel och resultat:

Exempel	Resultat
CHIINV(0.9237827, 15)	Returnerar 8.0000

FDensity

`FDensity()` returnerar sannolikheten för `F`-fördelningen.

Syntax:

```
FDensity(value, degrees_freedom1, degrees_freedom2)
```

Returnerad datatyp: tal

Argument

Argument	Beskrivning
`value`	Det värde där fördelningen ska utvärderas. Värdet får inte vara negativt.
`degrees_freedom1`	Ett positivt heltal som anger täljarens frihetsgrad.
`degrees_freedom2`	Ett positivt heltal som anger nämnarens frihetsgrad.

FDist

`FDist()` returnerar den ackumulerade sannolikheten för `F`-fördelningen.

Syntax:

```
FDist(value, degrees_freedom1, degrees_freedom2)
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
value	Det värde där fördelningen ska utvärderas. Värdet får inte vara negativt.
degrees_freedom1	Ett positivt heltal som anger täljarens frihetsgrad.
degrees_freedom2	Ett positivt heltal som anger nämnarens frihetsgrad.

Denna funktion hänger samman med funktionen **FInv** på följande sätt:
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Exempel och resultat:

Exempel	Resultat
FDIST(15, 8, 6)	Returnerar 0,0019

FInv

`FInv()`returnerar inversionen av den ackumulerade sannolikheten för `F`-fördelningen.

Syntax:

```
FInv(prob, degrees_freedom1, degrees_freedom2)
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
prob	En sannolikhet som har samband med F-fördelningen och måste vara ett tal mellan 0 och 1.
degrees_ freedom	Ett heltal som anger frihetsgrad.

Denna funktion hänger samman med funktionen **FDist** på följande sätt:
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Exempel och resultat:

Exempel	Resultat
FINV(0.0019369, 8, 6)	Returnerar 15.0000

GammaDensity

`GammaDensity()` returnerar sannolikheten för gammafördelningen.

Syntax:

```
GammaDensity(value, k, θ)
```

Returnerad datatyp: tal

Argument

Argument	Beskrivning
value	Det värde där fördelningen ska utvärderas. Värdet får inte vara negativt.
k	Ett positivt tal definierar den andra formens parameter.
θ	Ett positivt tal definierar skalans parameter.

GammaDist

`GammaDist()` returnerar den ackumulerade sannolikheten för gammafördelningen.

Syntax:

```
GammaDist(value, k, θ)
```

Returnerad datatyp: tal

Argument

Argument	Beskrivning
value	Det värde där fördelningen ska utvärderas. Värdet får inte vara negativt.
k	Ett positivt tal definierar den andra formens parameter.
θ	Ett positivt tal definierar skalans parameter.

Denna funktion hänger samman med funktionen `GammaINV` på följande sätt:
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

GammaInv

`GammaInv()` returnerar inversionen av den ackumulerade sannolikheten för gammafördelningen.

Syntax:

```
GammaInv(prob, k, θ)
```

Returnerad datatyp: tal

Argument

Argument	Beskrivning
prob	En sannolikhet som associeras med Gamma-sannolikhetsfördelningen. Måste vara ett tal mellan 0 och 1.
k	Ett positivt tal definierar den andra formens parameter.
θ	Ett positivt tal definierar skalans parameter.

Denna funktion hänger samman med funktionen `GammaDist` på följande sätt:
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

NormDist

`NormDist()` returnerar den kumulativa normalfördelningen hos det angivna medelvärdet och standardavvikelsen. Om mean = 0 och standard_dev = 1, returnerar funktionen standardnormalfördelningen.

Syntax:

```
NORMDIST(value, [mean], [standard_dev], [cumulative])
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
value	Det värde där fördelningen ska utvärderas.
mean	Valfritt värde som anger fördelningens aritmetiska medelvärde. Om du inte anger detta argument är standardvärdet 0.
standard_dev	Valfritt positivt värde som anger fördelningens standardavvikelse. Om du inte anger detta argument är standardvärdet 1.
cumulative	Du kan valfritt välja att använda en standardnormalfördelning eller en kumulativ fördelning. 0 = standard normalfördelning 1 = kumulativ fördelning (standard)

Denna funktion hänger samman med funktionen **NormInv** på följande sätt:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Exempel och resultat:

Exempel	Resultat
NORMDIST(0.5, 0, 1)	Returnerar 0,6915

NormInv

`NormInv()` returnerar inversen av den normala kumulativa fördelningen för det angivna medelvärdet och standardavvikelsen.

Syntax:

```
NORMINV(prob, mean, standard_dev)
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
prob	En sannolikhet som har samband med normalfördelningen. Måste vara ett tal mellan 0 och 1.
mean	Ett värde som anger fördelningens aritmetiska medelvärde.
standard_dev	Ett positivt värde som anger fördelningens standardavvikelse.

Denna funktion hänger samman med funktionen **NormDist** på följande sätt:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Exempel och resultat:

Exempel	Resultat
NORMINV(0.6914625, 0, 1)	Returnerar 0.5000

PoissonDist

`PoissonDist()` returnerar den ackumulerade sannolikheten för Poisson-fördelningen.

Syntax:

```
PoissonDist(value, mean)
```

Returnerad datatyp: tal

Argument

Argument	Beskrivning
`value`	Det värde där fördelningen ska utvärderas. Värdet får inte vara negativt.
`mean`	Ett positivt tal definierar det genomsnittliga resultatet.

Denna funktion hänger samman med funktionen `PoissonInv` på följande sätt:
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

PoissonFrequency

`PoissonFrequency()` returnerar Poisson-fördelningen.

Syntax:

```
PoissonFrequency(value, mean)
```

Returnerad datatyp: tal

Argument

Argument	Beskrivning
`value`	Det värde där fördelningen ska utvärderas. Värdet får inte vara negativt.
`mean`	Ett positivt tal definierar det genomsnittliga resultatet.

PoissonInv

`PoissonInv()` returnerar inversionen av den ackumulerade sannolikheten för Poisson-fördelningen.

Syntax:

```
PoissonInv(prob, mean)
```

Returnerad datatyp: tal

Argument

Argument	Beskrivning
`prob`	En sannolikhet som associeras med Poisson-sannolikhetsfördelningen. Måste vara ett tal mellan 0 och 1.
`mean`	Ett positivt tal definierar det genomsnittliga resultatet.

Denna funktion hänger samman med funktionen `PoissonDIST` på följande sätt:
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

TDensity

`TDensity()` returnerar värdet för studentens `t`-täthetsfunktion där ett numeriskt värde är ett beräknat värde av `t`, vars sannolikhet ska beräknas.

Syntax:

```
TDensity(value, degrees_freedom)
```

Returnerad datatyp: tal

Argument

Argument	Beskrivning
`value`	Det värde där fördelningen ska utvärderas. Värdet får inte vara negativt.
`degrees_freedom`	Ett positivt heltal som anger frihetsgrad.

TDist

`TDist()` returnerar sannolikheten hos studentens `t`-fördelning där ett numeriskt värde är ett beräknat värde av `t` vars sannolikhet ska kalkyleras.

Syntax:

```
TDist(value, degrees_freedom, tails)
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
value	Det värde där fördelningen ska utvärderas. Värdet får inte vara negativt.
degrees_freedom	Ett positivt heltal som anger frihetsgrad.
tails	Måste vara antingen 1 (ensidig fördelning) eller 2 (dubbelsidig fördelning).

Denna funktion hänger samman med funktionen **TInv** på följande sätt:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value`

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Exempel och resultat:

Exempel	Resultat
TDIST(1, 30, 2)	Returnerar 0.3253

TInv

`TInv()` returnerar `t`-värdet hos studentens `t`-fördelning som en funktion av sannolikhet och frihetsgrader.

Syntax:

```
TINV(prob, degrees_freedom)
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
prob	En dubbelsidig sannolikhet som har samband med t-fördelningen. Måste vara ett tal mellan 0 och 1.
degrees_ freedom	Ett heltal som anger frihetsgrad.

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Denna funktion hänger samman med funktionen **TDist** på följande sätt:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value.`

Exempel och resultat:

Exempel	Resultat
TINV(0.3253086, 30)	Returnerar 1.0000

8.24 Strängfunktioner

Den här delen beskriver funktioner för att hantera och modifiera strängar.

Alla funktioner kan användas både i dataladdningsskriptet och i diagramuttryck, utom **Evaluate** som endast kan användas i dataladdningsskriptet.

Strängfunktioner - översikt

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Capitalize

Funktionen **Capitalize()** konverterar det första tecknet i varje ord i en textsträng till versaler och konverterar alla andra tecken till gemener.

```
Capitalize (text)
```

Chr

Chr() returnerar det Unicode-tecken som motsvarar indatavärdets heltal, även kallat kodpunkt.

```
Chr (int)
```

CountRegEx

CountRegEx() returnerar antalet förekomster av det angivna reguljära uttrycksmönstret i texten i indatasträngen. Om det inte blir någon träff returneras 0.

```
CountRegEx (text, regex)
```

Evaluate

Evaluate() undersöker om indatatexten kan utvärderas som ett giltigt Qlik Sense-uttryck och returnerar i så fall värdet för uttrycket som en sträng. Om indatasträngen inte är ett giltigt uttryck returneras NULL.

```
Evaluate (expression_text)
```

ExtractRegEx

ExtractRegEx() extraherar text från ett indatastränguttryck med hjälp av det angivna reguljära uttrycksmönstret. Funktionen returnerar ett nullvärde om inga matchningar hittas.

```
ExtractRegEx (text, regex [, field_no])
```

ExtractRegExGroup

ExtractRegExGroup() extraherar text från ett indatastränguttryck med hjälp av det sammansatta reguljära uttrycksmönster som anges. När du använder funktionen anger du den grupp som ska användas i den sammansatta regexen. Funktionen returnerar ett nullvärde om inga matchningar hittas.

```
ExtractRegExGroup (text, regex, group [, field_no])
```

FindOneOf

FindOneOf() söker en sträng för att hitta positionen för eventuella tecken som ingår i en uppsättning angivna tecken. Positionen för den N:e förekomsten av ett tecken från sökuppsättningen returneras, där N är den valfria tredje parametern för funktionen. Om ingen tredje parameter anges returneras den första förekomsten. Om ingen motsvarighet påträffas, returneras **0**.

```
FindOneOf (text, char_set[, count])
```

Hash128

Hash128() returnerar ett 128-bitars hashvärde av de kombinerade indatauttrycksvärdena. Resulterar i en 22-teckensträng.

```
Hash128 (expr{, expression})
```

Hash160

Hash160() returnerar ett 160-bitars hashvärde av de kombinerade indatauttrycksvärdena. Resulterar i en 27-teckensträng.

```
Hash160 (expr{, expression})
```

Hash256

Hash256() returnerar ett 256-bitars hashvärde av de kombinerade indatauttrycksvärdena. Resulterar i en 43-teckensträng.

```
Hash256 (expr{, expression})
```

Index

Index() söker en sträng för att hitta startpositionen för den n:e förekomsten av en angiven delsträng. Ett valbart tredje argument ger värdet för n, vilket annars är 1. Ett negativt värde söker från slutet av strängen. Strängens positioner är numrerade vänster till höger från **1** och uppåt.

```
Index (text, substring[, count])
```

IndexRegEx

IndexRegEx() söker igenom indatasträngen och returnerar startpositionen för den n:te förekomsten av det angivna reguljära uttrycksmönstret. Ett valbart tredje argument **count** ger värdet för n, vilket annars är 1. Strängens positioner är numrerade vänster till höger från 1 och uppåt. Om ingen matchning hittas returnerar funktionen 0.

```
IndexRegEx (text, regex [, count])
```

IndexRegExGroup

ExtractRegExGroup() extraherar text från ett indatastränguttryck med hjälp av det sammansatta reguljära uttrycksmönster som anges. När du använder funktionen anger du den grupp som ska användas i den sammansatta regexen. Funktionen returnerar ett nullvärde om inga matchningar hittas.

```
IndexRegExGroup (text, regex, group [, count])
```

IsJson

IsJson() testar om en angiven sträng innehåller giltiga JSON-data (JavaScript Object Notation). Du kan även validera en specifik JSON-datatyp.

```
IsJson (json [, type])
```

IsRegEx

IsRegEx() returnerar huruvida den angivna texten är ett giltigt reguljärt uttryck.

```
IsRegEx (expr [, debug)
```

JsonGet

JsonGet() returnerar sökvägen till en JSON-datasträng (JavaScript Object Notation). JSON-data måste vara giltiga, men kan innehålla extra mellanslag eller nya rader.

```
JsonGet (json, path)
```

JsonSet

JsonSet() modifierar en sträng som innehåller JSON-data (JavaScript Object Notation). Den kan ställa in eller infoga ett JSON-värde med den nya platsen som anges av sökvägen. JSON-data måste vara giltiga, men kan innehålla extra mellanslag eller nya rader.

```
JsonSet (json, path, value)
```

KeepChar

KeepChar() returnerar en sträng som består av de tecken från den första strängen som matchar tecknen i den andra strängen. Denna funktion är skiftlägeskänslig.

```
KeepChar (text, keep_chars)
```

Left

Left() returnerar en sträng som består av de första (längst till vänster) tecknen i indatasträngen, där antalet tecken fastställs av det andra argumentet.

```
Left (text, count)
```

Len

Len() returnerar längden för indatasträngen.

```
Len (text)
```

LevenshteinDist

LevenshteinDist() returnerar Levenshtein-avståndet mellan två strängar. Det definieras som det minsta antalet enkelteckensredigeringar (infogningar, borttagningar eller ersättningar) som krävs för att förvandla en sträng till en annan. Funktionen är användbar vid ungefärliga strängjämförelser.

```
LevenshteinDist (text1, text2)
```

Lower

Lower() konverterar alla tecken i indatasträngen till gemener.

```
Lower (text)
```

LTrim

LTrim() returnerar strängen rensad på inledande blankstegstecken.

```
LTrim (text)
```

MatchRegEx

MatchRegEx() jämför en indatasträng (**text**) med ett eller flera reguljära uttryck (**regex**). Den returnerar den numeriska platsen för de reguljära uttrycksmönster som matchar. Returnerar ett matchningssvar endast om utvärderingen av det reguljära uttrycket matchar hela textinmatningen. Om inga matchningar hittas returnerar funktionen 0.

```
MatchRegEx (text, regex1 [ , regex2,...regexN])
```

Mid

Mid() returnerar den del av indatasträngen som börjar vid positionen för det tecken som definieras av det andra argumentet, ”start”, och returnerar det antal tecken som definieras av det tredje argumentet ”count”. Om ”count” utelämnas returneras resten av indatasträngen. Det första tecknet i indatasträngen numreras som 1.

```
Mid (text, start[, count])
```

Ord

Ord() returnerar det numeriska värdet (ASCII eller Unicode) för det första tecknet i en sträng. Denna funktion är användbar för att utvärdera eller jämföra strängar baserat på deras underliggande teckenkoder, t.ex. vid sortering eller filtrering av strängar med icke-standardiserade tecken.

```
Ord (text)
```

PurgeChar

PurgeChar() returnerar en sträng som består av de tecken som finns i indatasträngen (”text”), utom alla eventuella tecken som visas i det andra argumentet (”remove_chars”).

```
PurgeChar (text, remove_chars)
```

Repeat

Repeat() bildar en sträng som består av indatasträngen som upprepats det antal gånger som definieras av det andra argumentet.

```
Repeat (text[, repeat_count])
```

Replace

Replace() returnerar en sträng efter att ha ersatt alla förekomster av en given delsträng inom indatasträngen med en annan delsträng. Funktionen är inte rekursiv och fungerar från vänster till höger.

```
Replace (text, from_str, to_str)
```

ReplaceRegEx

ReplaceRegEx() returnerar en sträng efter att ha ersatt en eller flera matchningar mellan en indatasträng och ett angivet reguljärt uttrycksmönster. Den text som ersätter den matchande texten anges i argumentet **to_str**. Funktionen går från vänster till höger, men om du anger ett negativt värde för **occurrence** läses funktionen från höger till vänster.

```
ReplaceRegEx (text, regex, to_str [, occurrence])
```

ReplaceRegExGroup

ReplaceRegExGroup() returnerar en sträng efter att ha ersatt en eller flera matchningar mellan en indatasträng och det sammansatta reguljära uttrycksmönster som anges. Den text som ersätter den matchande texten anges i argumentet **to_str**. Funktionen går från vänster till höger, men om du anger ett negativt värde för **occurrence** läses funktionen från höger till vänster.

```
ReplaceRegExGroup (text, regex, to_str, group [, occurrence])
```

Right

Right() returnerar en sträng som består av de sista (längst till höger) tecknen av indatasträngen, där antalet tecken fastställs av det andra argumentet.

```
Right (text, count)
```

RTrim

RTrim() returnerar strängen rensad på avslutande blankategstecken.

```
RTrim (text)
```

SubField

Subfield() används för att extrahera delsträngkomponenter från ett överordnat strängfält där de ursprungliga postfälten består av två eller fler delar, som separeras av en avgränsare.

```
SubField (text, delimiter[, field_no ])
```

SubFieldRegEx

SubFieldRegEx() extraherar text från ett indatastränguttryck med hjälp av det angivna reguljära uttrycksmönstret som en avgränsare. Funktionen returnerar ett nullvärde om inga matchningar hittas.

```
SubFieldRegEx (text, regex_delimiter [, field_no])
```

SubStringCount

SubstringCount() returnerar antalet förekomster av angiven delsträng i indatasträngtexten. Om det inte blir någon träff returneras 0.

```
SubStringCount (text, substring)
```

TextBetween

TextBetween() returnerar texten i indatasträngen som finns mellan de tecken som definierats som avgränsare.

```
TextBetween (text, delimiter1, delimiter2[, n])
```

Trim

Trim() returnerar strängen rensad på inledande och avslutande blankstegstecken.

```
Trim (text)
```

Upper

Upper() konverterar alla tecken i indatasträngen till versaler för alla texttecken i uttrycket. Tal och symboler ignoreras.

```
Upper (text)
```

Arbeta med reguljära uttryck i laddningsskript och diagramuttryck

Förbättra dina analyser och din dataförberedelse genom att använda reguljära uttryck (regex) i data-laddningsskript och i diagramuttryck. Reguljära uttryck erbjuder avancerade, flexibla och kraftfulla alternativ för att hitta och ersätta textbaserade data. För att arbeta med reguljära

uttryck i Qlik Sense använder du de tillgängliga funktionerna för skript och diagram.

Så här arbetar du med reguljära uttryck

Reguljära uttryck (regex) erbjuder omfattande alternativ för avancerad mönstermatchning. Skriv ett reguljärt uttryck för att ange ett visst textmönster som du vill identifiera. Du kanske t.ex. vill hitta förekomster av e-postadresser eller URL:er i större mängder naturligt språk.

Det finns många olika format som reguljära uttryck kan skrivas i, vart och ett med sina egna syntaxregler. De tillgängliga funktionerna för reguljära uttryck använder Perls syntax för reguljära uttryck.

Använd reguljära uttryck i Qlik Sense, använd tillgängliga regexfunktioner för skript och diagram. Användningen av dessa funktioner förutsätter att du har en grundläggande förståelse för hur reguljära uttryck fungerar. Se *Tillgängliga funktioner och exempel (page 1702)* för mer information och exempel.

Användningsfall

Du kan använda reguljära uttryck för:

- Extrahering och separering av information från text som innehåller flera olika behållare. Till exempel e-postadresser, telefonnummer, URL:er med mera.
- Standardisering av formatering av data.
- Söka efter och/eller ersätta text

Med de regex-funktioner som finns tillgängliga kan du göra följande i laddningsskript:

- Transformera data och läsa in dem för användning i analyser.
- Validera data för att säkerställa att de följer formaterings- och andra efterlevnadsstandarder.
- Identifiera, maskera eller ändra personliga identitetsuppgifter (PII) till en form som är lämplig för konsumenter av ditt analysinnehåll.
- Selektivt läsa in data i applikationer beroende på om de matchar specifika textmönster eller inte.

Användningsfallen för regexfunktioner i diagramuttryck liknar dem för laddningsskript. Med hjälp av funktionerna i diagramuttryck kan du också:

- Transformera data i farten, villkorligt eller statiskt, beroende på användarnas behov och interaktioner.
- Använd funktionerna för att definiera etiketter, färginställningar och andra diagram-specifika egenskaper.

Prestandaöverväganden

De funktioner för reguljära uttryck som finns i laddningsskript och diagramuttryck används bäst för avancerade strängtransformeringar där flexibilitet är ett krav. Att använda regexfunktioner är mer resurskrävande än alternativen utan regex. För enklare datatransformeringar bör du överväga att använda andra tillgängliga skript- och diagramfunktioner, inklusive strängfunktioner. För en lista över tillgängliga strängfunktioner, se *Strängfunktioner (page 1695)*.

När du arbetar med stora datavolymer kan regexoperationer för att söka efter matchningar av stora eller negativa **n**-värden orsaka minskad prestanda. I tillgängliga regex-funktioner motsvarar **n**-värdet funktionsargument som **occurrence**, **count** och **field_no**.

Tillgängliga funktioner och exempel

- *CountRegEx (page 1707)*
- *ExtractRegEx (page 1717)*
- *ExtractRegExGroup (page 1722)*
- *IndexRegEx (page 1754)*
- *IndexRegExGroup (page 1757)*
- *IsRegEx (page 1765)*
- *MatchRegEx (page 1800)*
- *ReplaceRegEx (page 1820)*
- *ReplaceRegExGroup (page 1826)*
- *SubFieldRegEx (page 1844)*

Capitalize

Funktionen **Capitalize()** konverterar det första tecknet i varje ord i en textsträng till versaler och konverterar alla andra tecken till gemener.

Syntax:

```
Capitalize(text)
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Sträng som ska utvärderas.

Exempel: diagramuttryck

Exempel	Resultat
`Capitalize( 'star trek' )`	Returnerar `Star Trek`
`Capitalize( 'AA bb cC Dd' )`	Returnerar `Aa Bb Cc Dd`

Exempel – grunderna i versalisering

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `TextToCapitalize`.

Laddningsskript

```
Example:
Load * inline [
TextToCapitalize
50 Cent
a-ha
BIGBANG
girl in red
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `TextToCapitalize`

Skapa följande beräknade dimension:

- `=Capitalize(TextToCapitalize)`, för att beräkna vilka tecken som ska kapitaliseras.

Resultattabell

Text att versalisera	Versalisera(Text att versalisera)
50 Cent	50 Cent
a-ha	A-Ha
BIGBANG	Bigbang
girl in red	Girl In Red

Det sker ingen ändring av den första dataraden eftersom den första teckenuppsättningen börjar med ett nummer och den andra redan har stor bokstav.

I den andra raden skrivs det första och tredje tecknet med stor bokstav.

I den tredje raden är endast den första bokstaven versal och alla efterföljande bokstäver är gemener.

I den fjärde raden är den första bokstaven i varje ord versal.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Load
String,
Capitalize(String)
Inline
[String
rHode iSland
```

```
washingTon d.C.
new york];
```

Resultattabell

Sträng	Storbokstav(Textsträng)
rHode iSland	Rhode Island
washingTon d.C.	Washington D.C.
new york	New York

Exempel – versaliseringsscenario

Diagramuttryck

Översikt

En datauppsättning med kundnamn matas in i olika format, t.ex. alla gemener, alla versaler eller blandade skiftlägen. Funktionen `Capitalize` används för att rensa data så att kundens för- och efternamn skrivs med stor bokstav när de visas i instrumentpanelen.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `CustomerName`.

Laddningsskript

```
Example:
Load * inline [
CustomerName
john smith
JANE DOE
miCHAel Brown
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `CustomerName`

Skapa följande beräknade dimension:

- `=Capitalize(CustomerName)` att skriva kundnamnen med inledande versal.

Resultattabell

Kundnamn	Versalisera(Kundnamn)
JANE DOE	Jane Doe

Kundnamn	Versalisera(Kundnamn)
john smith	John Smith
miCHAel Brown	Michael Brown

Jämför utdata från funktionen `Capitalize` med de ursprungliga `CustomerName`-strängvärdena som laddades i skriptet. Du kan se hur funktionen ändrade till korrekt skiftläge för namnen. Genom att visa kundnamn med standardiserad versalisering förbättras instrumentpanelens utseende, konsekvens och läsbarhet.

Chr

Chr() returnerar det Unicode-tecken som motsvarar indatavärdets heltal, även kallat kodpunkt.

Syntax:

```
Chr(int)
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
int	Det heltal som ska utvärderas.

Exempel: diagramuttryck

Exempel	Resultat
Chr(65)	Returnerar strängen `A`
Chr(163)	Returnerar strängen `£`
Chr(35)	Returnerar strängen `#`

Exempel – grunderna i Chr

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `InputInt`. Fältet innehåller det heltalsvärde som ska utvärderas.

Laddningsskript

```
Example:
Load * inline [
```

```
InputInt
65
163
35
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `InputInt`

Skapa följande mått:

- `=Chr(InputInt)` för att beräkna värdena i `InputInt` och returnera ett Unicode-tecken.

Resultattabell

InputInt	Chr(InputInt)
35	#
65	A
163	£

Den första raden returnerar strängen `#` eftersom det är det Unicode-tecken som motsvarar värdet 35.

Den andra raden returnerar strängen `A` eftersom det är det Unicode-tecken som motsvarar värdet 65.

Den tredje raden returnerar strängen `£` eftersom det är det Unicode-tecken som motsvarar värdet 163.

Exempel – Chr-scenario

Diagramuttryck

Översikt

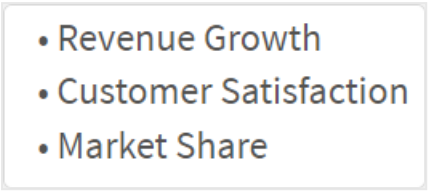
I det här exemplet visas hur du använder `Chr`-funktionen för att skapa en punktlista över textsträngarna i ett textobjekt.

Resultat

Öppna ett ark. Skapa ett nytt textobjekt och lägg sedan till följande kod i ett mått:

- `= Chr(149) & ' Revenue Growth' & Chr(10) & Chr(149) & ' Customer Satisfaction' & Chr(10) & Chr(149) & ' Market Share'`

Utdata för mätuttrycket i textobjekt



Utgången härleds på följande sätt:

- `Chr(149)` returnerar en punkt (•).
- `Chr(10)` infogar en radbrytning (`newline`) mellan artiklarna.
- Ampersand (`&`) är en operator för strängkonkatenering som konkatenerar de enskilda textdelarna till en enda textsträng.

CountRegEx

CountRegEx() returnerar antalet förekomster av det angivna reguljära uttrycksmönstret i texten i indatasträngen. Om det inte blir någon träff returneras 0.

Den här funktionen utför regexoperationer som är skiftlägeskänsliga. Alternativt kan du använda varianten **CountRegExI()** för att utföra skiftlägesokänsliga regexoperationer.

Syntax:

```
CountRegEx (text, regex)
```

Returnerad datatyp: numeriska

Argument

Argument	Beskrivning
text	Indatasträngens text inom vilken du vill söka efter ett reguljärt uttryck.
regex	Det reguljära uttrycksmönstret som ska sökas efter.

Exempel på funktioner

Exempel	Resultat
`CountRegEx('abc123','[a-z]')`	Returnerar `3`.
`CountRegEx('abc123','[a-z][0-9]')`	Returnerar `1`.
`CountRegEx('abc123','[0-9]')`	Returnerar `3`.
`CountRegEx('ABC','[a-z]{3}')`	Returnerar `0`.
`CountRegExI('ABC','[a-z]{3}')`	Returnerar `1`. Eftersom **CountRegExI()**-varianten används är textsökningar inte skiftlägeskänsliga.

Användning

Exempel på användningsområden för **CountRegEx()** är:

- Räknar antalet gånger ett visst textmönster förekommer i en sträng. Du kan t.ex. söka efter förekomster av e-postadresser, telefonnummer och annan information i dokument och e-postmeddelanden.
- Validering av huruvida data uppfyller ett specifikt syntaxkrav. Du kan till exempel identifiera förekomsten av fel vid datainmatning.

Exempel 1 – laddningsskript för att räkna giltiga årsintervall

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell med namnet `Vehicles` som innehåller en lista över fordon, tillsammans med skick och årsmodell för varje fordon.
- För varje post i `Vehicles` görs en beräkning för att avgöra om årsintervallet för fordonet uppfyller den förväntade syntaxen. Giltiga årsintervall är bland annat 1990-, 2000-, 2010- och 2020-talet. Denna beräkning utförs med hjälp av funktionen **CountRegEx()**.

Laddningsskript

```
Vehicles:
Load CountRegEx(Model_YearRange, '(^19[0-9][0-9]s$)|(^20[0-9][0-9]s$)') as Valid_Year_Data,
* Inline [
Vehicle_ID, Condition, Model_YearRange
1, Good,2010Os
2,Fair,2020s
3, Poor,2020s
4, Good,210s
5, Excellent,2000s
6, Excellent,2020s
7, Excellent,202s
8, Good,2010s
9, Poor,2010ss
10, Poor,1990s
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Vehicle_ID`
- `Valid_Year_Data`

Resultattabell

Vehicle_ID	Valid_Year_Data
1	0
2	1
3	1
4	0
5	1
6	1
7	0
8	1
9	0
10	1

Du kan också lägga till en KPI på arket och lägga till detta mått:

```
Sum(Valid_Year_Data)
```

Detta diagram visar det totala antalet poster med giltiga *Model_YearRange*-värden.

> *Funktionen **MatchRegEx()** är också användbar för den här typen av valideringsanvändningsfall.*

Exempel 2 – laddningsskript för att räkna antalet e-postadresser i textsträngar

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datatabell med namnet `Employees` som innehåller uppgifter om ett företags anställda. Alla e-postadresser för varje anställd behålls i en enda sträng.
- Skapandet av ett fält, `NoOfEmails`, för att beräkna hur många e-postadresser som finns i indatasträngen för varje anställd.

Laddningsskript

```
Employees:
```

```
Load CountRegExI(Emails, '@[a-zA-Z0-9.-]*\.?[a-zA-Z0-9]*\.com') as NoOfEmails,
* Inline `
ID     Emails
1      john_doe@test.com john_michael_doe@example.com j.doe123@example123.com
2      jane_doe@test.com
3      Michael_Andrews@test.com Michael_John_Andrews@example.com
4      charlie_kent@support.test.com charlie_peter_kent@support.example.com
cp.kent.1985@test123.com c_p_kent@test123.com
5      frances_johnson@test.com frances_johnson@example.com
6      rebecca_smith@hr.test.com
` (delimiter is '\t');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- ID
- Emails
- NoOfEmails

Resultattabell

ID	Emails	NoOfEmails
1	john_doe@test.com john_michael_doe@example.com j.doe123@example123.com	3
2	jane_doe@test.com	1
3	Michael_Andrews@test.com Michael_John_Andrews@example.com	2
4	charlie_kent@support.test.com charlie_peter_kent@support.example.com cp.kent.1985@test123.com c_p_kent@test123.com	4
5	frances_johnson@test.com frances_johnson@example.com	2
6	rebecca_smith@hr.test.com	1

Exempel 3 – diagramuttryck för att räkna förekomster av sträng

Diagramuttryck och resultat

Översikt

Öppna Skriptredigeraren och lägg till följande laddningsskript till ett nytt delavsnitt.

Laddningsskriptet innehåller texten från ett antal företagskorrespondenser som skickades ut till anställda i en organisation. Datan har redan rensats från all personliga identitetsuppgifter (PII), förutom företagsnamnet och den e-postadress som skickade varje korrespondens. Texten i varje meddelande har sammanfattats till en enda sammanhängande sträng. Med andra ord har all formatering och alla tecken för nya rader tagits bort.

Våra krav:

- Identifiera vilka meddelanden som innehåller en sträng som innehåller e-postadressen *TestCompanyNameCorrespondence@test.com*, tillsammans med en specifik introduktion till denna adress som indikerar att den var avsändaren.
- Vi vill ha flexibilitet när det gäller vilka introduktioner som ska identifieras, men fortfarande också ha vissa bestämda begränsningar. Introduktionerna `Sender` och `Sent by` anses båda vara giltiga. För att uppnå detta mål kan regex-mönster användas.
- E-postadressers känslighet för versaler och gemener är inte en faktor. Därför ska sökningen vara skiftlägesokänslig.

Laddningsskript

```
Correspondences:
Load * Inline [
ID|MessageBody
1|Sent by-- TestCompanyNameCorrespondence@test.com Good morning TestCompanyName team! I hope you are all well. I am just reaching out about the fabulous benefits package we launched just year. Our goal is to take a poll of what you all think. Can you please rate the new options from 1-10 in a response to this email, by this Tuesday? Thank you!
2|Sender: TestCompanyNameIT@test.com Did you know you can access our IT portal anytime, from any of your company devices? It's true! Reach out to your team lead to learn more.
3|Sender: TESTCOMPANYNAMECORRESPONDENCE@test.COM URGENT: This is a message to inform employees of an ongoing issue with our local branch. The branch will be closed until further notice. Thank you
4|Sender: TestCompanyNameVolunteering@test.com Dear team! We are looking for volunteers to help with this year's charity event! We need 40 volunteers to help with the event. If you are interested, please send an email to your direct supervisor. Thanks, TestCompanyName team members, for helping make the world a better place!
5|Sender is TestCompanyNameCorrespondence@test.com Hi folks, it's time to announce the annual TestCompanyName staff party! The party will be taking place at the office this year. Please make sure to RSVP, and we hope to see you there! Sincerely, the TestCompanyName management team
] (delimiter is |);
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ID`
- `MessageBody`

Lägg till följande mått:

```
=CountRegExI(MessageBody,'(Sender.*TestCompanyNameCorrespondence@test.com)|(Sent by.*TestCompanyNameCorrespondence@test.com)')
```

I måttegenskaperna ställer du in **Totalfunktionen** till **Sum**. Detta konfigurerar totalraden längst upp i tabellen så att det totala antalet poster som bidrar till räkningen räknas.

Måttet räknar antalet matchningar för det angivna regex-mönstret i meddelandets innehåll. I regexmönstret anger `.*` valfritt antal av alla tecken utom tecken för ny rad. Mönstret tar också hänsyn till variationen i hur e-postadressen introduceras: både `Sender` och `Sent by` är giltiga matchningar. **CountRegExI()**-varianten av funktionen säkerställer sökningar utan skiftlägeskänslighet.

Resultattabell

ID	MessageBody	=CountRegExI(MessageBody,'(Sender.*TestCompanyNameCorrespondence@test.com)|(Sent by.*TestCompanyNameCorrespondence@test.com)')
Totals	-	**3**
1	Sent by-- TestCompanyNameCorrespondence@test.com Good morning TestCompanyName team! I hope you are all well. I am just reaching out about the fabulous benefits package we launched just year. Our goal is to take a poll of what you all think. Can you please rate the new options from 1-10 in a response to this email, by this Tuesday? Thank you!	1
2	Sender: TestCompanyNameIT@test.com Did you know you can access our IT portal anytime, from any of your company devices? It's true! Reach out to your team lead to learn more.	0
3	Sender: TESTCOMPANYNAMECORRESPONDENCE@TEST.COM URGENT: This is a message to inform employees of an ongoing issue with our local branch. The branch will be closed until further notice. Thank you	1
4	Sender: TestCompanyNameVolunteering@test.com Dear team! We are looking for volunteers to help with this year's charity event! We need 40 volunteers to help with the event. If you are interested, please send an email to your direct supervisor. Thanks, TestCompanyName team members, for helping make the world a better place!	0

ID	MessageBody	=CountRegExl(MessageBody,'(Sender.*TestCompanyNameCorrespondence@test.com)|(Sent by.*TestCompanyNameCorrespondence@test.com)')
5	Sender is TestCompanyNameCorrespondence@test.com Hi folks, it's time to announce the annual TestCompanyName staff party! The party will be taking place at the office this year. Please make sure to RSVP, and we hope to see you there! Sincerely, the TestCompanyName management team	1

Av de fem meddelandena är det tre som matchar det angivna regexmönstret. Varje meddelande innehåller en matchning.

*Funktionen **MatchRegEx()** är också användbar för den här typen av valideringsanvändningsfall.*

Evaluate

Evaluate() undersöker om indatatexten kan utvärderas som ett giltigt Qlik Sense-uttryck och returnerar i så fall värdet för uttrycket som en sträng. Om indatasträngen inte är ett giltigt uttryck returneras NULL.

Syntax:

```
Evaluate(expression_text)
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
text	Sträng som ska utvärderas.

Denna strängfunktion kan inte användas i diagramuttryck.

Funktionsexempel och resultat

Funktionsexempel	Resultat
`Evaluate( 5 * 8 )`	Returnerar 40

Exempel – grunderna i utvärdering

Laddningsskript

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `ExpressionText`.
- En föregående laddning som använder funktionen `Evaluate` för att utvärdera fältet `ExpressionText` och returnera två nya fält: `Evaluated` och `Evaluatedx4`.

Laddningsskript

```
Example:
Load
ExpressionText
, Evaluate(ExpressionText) as Evaluated
, Evaluate(ExpressionText)*4 as Evaluatedx4
;
Load * Inline
[ExpressionText
4
5+3
100/5
123*2
329-9
0123456789012345678
1234567890123456789
0123456.5512
0123456.551247999457895236485934646 9
Today()
Bob
];
```

Resultat

Följande fält laddas i datatabellen:

- `ExpressionText`
- `Evaluated`
- `Evaluatedx4`

Results table

ExpressionText	Evaluated	Evaluatedx4
4	4	16
5+3	8	32
100/5	20	80
123*2	246	984
329-9	320	1280
0123456789012345678	0123456789012345678	4.9382715604938e+17
1234567890123456789	1234567890123456789	4.9382715604938e+18
0123456.5512	0123456.5512	493826.2048
0123456.55124799945789523648593 46469	0123456.55124799945789523648593 46469	493826.204992
Today()	11/5/2024	182404
Bob	-	-

Utdata från `Evaluate`-funktionen returnerar värden för alla `ExpressionText`-strängar utom den sista raden. Strängen `Bob` kan inte utvärderas, så funktionen returnerar inget värde – NULL. Alla rader har utvärderats, till exempel lägger den andra raden till `5` och `3` för att få `8`. Vissa av `ExpressionText`-strängarna returnerar strängen som den är eftersom strängen redan utvärderas till ett tal. Till exempel utvärderas strängen `0123456789012345678` till samma tal.

Även om Qlik Sense har en 14-siffrig precisionsgräns för numeriska värden, utvärderade `Evaluate`-funktionen 18-siffriga strängar som ett tal och tillämpade multiplikation i skriptet. Detta är användbart för att hantera mycket stora tal i skriptet.

Exempel – Utvärderingsscenario

Laddningsskript

Översikt

En dataupsättning med försäljningsdata innehåller produkter, priser och rabatter. Detta exempel tar fram det rabatterade priset för produkterna.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Product`
  - `Price`
  - `Discount %`
- En föregående inläsning som tar värdena `Price` och `Discount %` och beräknar ett ytterligare fält som heter `DiscountedPrice`. Det här nya fältet använder `Evaluate`-funktionen för att beräkna priset minus rabatten.

Laddningsskript

```
Example:
LOAD
Product,
Price,
[Discount (%)],
Price * (1 - Evaluate([Discount (%)] / 100)) AS DiscountedPrice
INLINE [
Product, Price, Discount (%)
Laptop, 1000, 15
Smartphone, 800, 10
Tablet, 600, 20
Headphones, 200, 5
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Product`
- `Price`
- `Discount %`
- `DiscountedPrice`

Results table

Product	Price	Discount %	DiscountedPrice
Headphones	200	5	190
Tablet	600	20	480
Smartphone	800	10	720
Laptop	1000	15	850

Utdata från `Evaluate`-funktionen visar hur du kan använda funktionen för att beräkna värden.

ExtractRegEx

ExtractRegEx() extraherar text från ett indatastränguttryck med hjälp av det angivna reguljära uttrycksmönstret. Funktionen returnerar ett nullvärde om inga matchningar hittas.

Den här funktionen utför regexoperationer som är skiftlägeskänsliga. Alternativt kan du använda varianten **ExtractRegExI()** för att utföra skiftlägesokänsliga regexoperationer.

Syntax:

```
ExtractRegEx (text, regex [, field_no])
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Stränguttryck som innehåller den text som ska extraheras i returvärdet.
regex	Det reguljära uttryck som ska användas för att extrahera text.
field_no	Numret på den matchning som ska extraheras. Detta är användbart när flera träffar för det reguljära uttrycket kan finnas i texten. Ange t.ex. värdet `4` för att få fram den fjärde matchningen. Det här är ett valfritt argument. Ange ett negativt värde för att vända ordningen på matchningarna. Följande gäller oavsett om funktionen används i ett laddningsskript eller ett diagramuttryck: • Om **field_no** är positivt kommer funktionen att returnera ett värde som identifierar matchningar från vänster till höger. • Om **field_no** är negativt kommer funktionen att returnera ett värde som identifierar matchningar från höger till vänster. När du använder funktionen i ett laddningsskript: • Om du använder funktionen **ExtractRegEx()** i en **LOAD**-sats och **field_no** utelämnas, genererar funktionen lika många poster som det finns matchningar. • Om flera fält laddas med hjälp av **ExtractRegEx()** och inget av dem specificerar ett **field_no**-argument skapas de kartesiska produkterna för alla kombinationer. När funktionen används i ett diagram-uttryck: • Om **field_no** utelämnas är standardvärdet `1`.

Exempel på funktioner

Exempel	Resultat
`ExtractRegEx('a,b c;1 2,3','[a-z]|[0-9]', 1)`	Returnerar `a`, som är den första matchningen som hittas.

Exempel	Resultat
`ExtractRegEx('a,b c;1 2,3','[a-z]|[0-9]', 2)`	Returnerar `b`, som är den andra matchningen som hittas.
`ExtractRegEx('a,b c;1 2,3','[a-z]|[0-9]', 4)`	Returnerar `1`, som är den fjärde matchningen som hittas.
`ExtractRegEx ('abc','a|b',-1)`	Returnerar `b`. När ett negativt värde anges räknas matchningarna i omvänd ordning.
`ExtractRegEx('A,B C;1 2,3','[a-c]|[4-9]')`	Returnerar ett nollvärde, eftersom **ExtractRegEx()** är skiftlägeskänsligt och den skiftlägesokänsliga varianten inte användes.
`ExtractRegExI('A,B C;1 2,3','[a-c]|[4-9]')`	Returnerar `A`, eftersom den skiftlägeskänsliga varianten **ExtractRegExI()** användes.

Användning

Du kan använda **Extract RegEx()** för att extrahera information som du vill isolera från data som också kan innehålla annan information (t.ex. fritext eller JSON-strängar). Exempel:

- Extrahera e-postadresser, telefonnummer, kontonummer och annan information från text.
- Extrahera numeriska värden från text (t.ex. valuta).
- Standardisera formateringen av text eller numeriska data.

Exempel 1 – laddningsskript för att extrahera e-postadresser från JSON

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell `OrderDetails` som innehåller ett fält `OrderJson`.
- Fältet `OrderJson` innehåller en JSON-sträng med orderdata. Formatering, t.ex. indrag och radbrytningar, har tagits bort från JSON. Orderdata innehåller e-postadresser som vi vill extrahera till separata värden.
- En ny tabell `EmailAddresses`, som laddas efter `OrderDetails`. I `EmailAddresses` **LOAD**-satsen läser vi in `OrderJson`-fältet från `OrderDetails`-tabellen och skapar ett nytt beräknat fält, `Email`, som innehåller de extraherade e-postadresserna. `OrderDetails` tas sedan bort.

Våra krav:

- E-postadresser i en specifik domän: `example.com`.
- E-postadresser som innehåller ett obegränsat antal alfanumeriska tecken samt en definierad uppsättning specialtecken.

Laddningsskript

```
OrderDetails:

Load * Inline `
OrderJson
{"Orders":[{"order_id":"12345","customer":{"name":"John
Doe","email":"john.doe@example.com"},"items":
{"product":"Laptop","quantity":2,"price":1200},"total_price":2400},{"order_
id":"12346","customer":{"name":"Jane Doe","email":"jane.doe@example.com"},"items":
{"product":"Tablet","quantity":4,"price":400},"total_price":1600},{"order_
id":"12347","customer":{"name":"Amalia Craig","email":"amalia.craig@example.com"},"items":
{"product":"Desktop Computer","quantity":1,"price":1900},"total_price":1900},{"order_
id":"12348","customer":{"name":"Ken Roberts","email":"ken.roberts@example.com"},"items":
{"product":"Phone Charger Cable","quantity":110,"price":1900},"total_price":1210}]}
` (delimiter is '\t');

EmailAddresses:

Load OrderJson, ExtractRegEx(OrderJson, '[a-zA-Z0-9!#$%^&*-_+=~{|}\/.'']+@example.com') as
Email Resident OrderDetails;

Drop table OrderDetails;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- Email

Resultattabell

Email
amalia.craig@example.com
jane.doe@example.com
john.doe@example.com
ken.roberts@example.com

Genom att inte ange något argument för **field_no** extraheras alla matchningar automatiskt till separata värden.

Exempel 2 – laddningsskript för att extrahera numeriska värden från valutabelopp

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell `Sales` som innehåller försäljningsdata.
- I tabellen `Sales` finns ett fält med namnet `Amount`, som innehåller transaktionsbelopp i USD. Dessa belopp uttrycks som strängar, som inkluderar valutasymboler och koder.
- Definitionen av ett anpassat fält inom `Sales`, `OrderAmountNumeric_USD`, som extraherar varje belopp till ett rent numeriskt värde. Dessa data måste vara i numerisk form så att vi kan använda dem i aggregeringar av diagram.

Laddningsskript

```
SET DateFormat='MM-DD-YYYY';

Sales:

Load ExtractRegEx(Amount, '[0-9]+.[0-9]{2}') as OrderAmountNumeric_USD, * Inline `

OrderID, OrderDate, Product, Amount
1, '12-27-2024', Software, $100.31 USD
2, '12-28-2024', Electronics, $513.93 USD
3, '12-29-2024', Grocery, $34.66 USD
4, '12-30-2024', Furniture, $1124.58 USD
5, '12-31-2024', Software, $95.95 USD
6, '01-01-2025', Software, $601.00 USD
7, '01-02-2025', Grocery, $1534.20 USD
8, '01-03-2025', Produce, $51.49 USD
9, '01-04-2025', Home Improvement, $176.12 USD
10, '01-05-2025', Produce, $432.77 USD
11, '01-06-2025', Software, $270.01 USD
12, '01-07-2025', Electronics, $83.85 USD
13, '01-08-2025', Furniture, $65.65 USD
14, '01-09-2025', Electronics, $1526.39 USD
15, '01-10-2025', Software, $432.44 USD
16, '01-11-2025', Furniture, $888.15 USD
17, '01-12-2025', Furniture, $300.00 USD
18, '01-13-2025', Grocery, $109.53 USD
19, '01-14-2025', Furniture, $487.97 USD
20, '01-15-2025', Software, $65.09 USD
`;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `OrderID`
- `Amount`
- `OrderAmountNumeric_USD`

Resultattabell

OrderID	Amount	OrderAmountNumeric_USD
1	$100.31 USD	100.31
2	$513.93 USD	513.93
3	$34.66 USD	34.66
4	$1124.58 USD	1124.58
5	$95.95 USD	95.95
6	$601.00 USD	601.00
7	$1534.20 USD	1534.20
8	$51.49 USD	51.49
9	$176.12 USD	176.12
10	$432.77 USD	432.77
11	$270.01 USD	270.01
12	$83.85 USD	83.85
13	$65.65 USD	65.65
14	$1526.39 USD	1526.39
15	$432.44 USD	432.44
16	$888.15 USD	888.15
17	$300.00 USD	300.00
18	$109.53 USD	109.53
19	$487.97 USD	487.97
20	$65.09 USD	65.09

Exempel 3 – diagramuttryck för att extrahera e-postadresser från JSON

Diagramuttryck och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell `OrderDetails` som innehåller ett fält `OrderJson`.
- Fältet `OrderJson` innehåller en JSON-sträng med orderdata. Formatering, t.ex. indrag och radbrytningar, har tagits bort från JSON. Orderdata innehåller e-postadresser som vi vill extrahera till separata värden.

Våra krav:

- E-postadresser i en specifik domän: `example.com`.
- E-postadresser som innehåller ett obegränsat antal alfanumeriska tecken samt en definierad uppsättning specialtecken.

Laddningsskript

```
OrderDetails:

Load * Inline `
OrderJson
{"Orders":[{"order_id":"12345","customer":{"name":"John
Doe","email":"john.doe@example.com"},"items":
{"product":"Laptop","quantity":2,"price":1200},"total_price":2400},{"order_
id":"12346","customer":{"name":"Jane Doe","email":"jane.doe@example.com"},"items":
{"product":"Tablet","quantity":4,"price":400},"total_price":1600},{"order_
id":"12347","customer":{"name":"Amalia Craig","email":"amalia.craig@example.com"},"items":
{"product":"Desktop Computer","quantity":1,"price":1900},"total_price":1900},{"order_
id":"12348","customer":{"name":"Ken Roberts","email":"ken.roberts@example.com"},"items":
{"product":"Phone Charger Cable","quantity":110,"price":1900},"total_price":1210}]}
` (delimiter is '\t');
```

Resultat

Ladda data och öppna ett ark. Skapa en tabell och lägg till vart och ett av följande uttryck som beräknade dimensioner:

- `=ExtractRegEx(OrderJson,'[a-zA-Z0-9!#$%^&*-_+=~{|}\/.'']+@example.com',1)`
- `=ExtractRegEx(OrderJson,'[a-zA-Z0-9!#$%^&*-_+=~{|}\/.'']+@example.com',2)`
- `=ExtractRegEx(OrderJson,'[a-zA-Z0-9!#$%^&*-_+=~{|}\/.'']+@example.com',3)`
- `=ExtractRegEx(OrderJson,'[a-zA-Z0-9!#$%^&*-_+=~{|}\/.'']+@example.com',4)`

Resultattabell

=ExtractRegEx (OrderJson,'[a-zA-Z0-9!#$%^&*-_+=~ {|}\V.'']+@example.com',1)	=ExtractRegEx (OrderJson,'[a-zA-Z0-9!#$%^&*-_+=~ {|}\V.'']+@example.com',2)	=ExtractRegEx (OrderJson,'[a-zA-Z0-9!#$%^&*-_+=~ {|}\V.'']+@example.com',3)	=ExtractRegEx (OrderJson,'[a-zA-Z0-9!#$%^&*-_+=~ {|}\V.'']+@example.com',4)
john.doe@example.com	jane.doe@example.com	amalia.craig@example.com	ken.roberts@example.com

ExtractRegExGroup

ExtractRegExGroup() extraherar text från ett indatastränguttryck med hjälp av det sammansatta reguljära uttrycksmönster som anges. När du använder funktionen anger du den grupp som ska användas i den sammansatta regexen. Funktionen returnerar ett nullvärde om inga matchningar hittas.

Om du använder funktionen **ExtractRegExGroup()** i en **LOAD**-sats och **field_no** utelämnas, kommer funktionen att returnera flera poster. Om flera fält läses in med hjälp av **ExtractRegExGroup()** skapas de kartesiska produkterna för alla kombinationer.

Den här funktionen utför regexoperationer som är skiftlägeskänsliga. Alternativt kan du använda varianten **ExtractRegExGroupI()** för att utföra skiftlägesokänsliga regexoperationer.

Syntax:

```
ExtractRegExGroup (text, regex, group [, field_no])
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Stränguttryck som innehåller den text som ska extraheras i returvärdet.
regex	Det reguljära uttryck som ska användas för att extrahera text.
group	Gruppens nummer, om det rör sig om ett sammansatt reguljärt uttryck. Om det reguljära uttrycket bara innehåller en grupp använder du istället funktionen **ExtractRegEx()**. Alternativt kan du använda **ExtractRegExGroup()** med ett **group**-värde på `0`. Du kan ange ett negativt värde för **group** för att söka efter matchningar från höger till vänster.
field_no	Numret på den matchning som ska extraheras. Detta är användbart när flera träffar för det reguljära uttrycket kan finnas i texten. Ange t.ex. värdet `4` för att få fram den fjärde matchningen. Det här är ett valfritt argument. Standard är 1 om det inte anges.

Exempel på funktioner

Exempel	Resultat
`ExtractRegExGroup('abc123 def456','([a-z]+)([0-9]+)',1)`	Returnerar `abc` (första gruppen i första matchningen).
`ExtractRegExGroup('abc123 def456','([a-z]+)([0-9]+)',1,2)`	Returnerar `def` (första gruppen i andra matchningen).
`ExtractRegExGroup('abc123 def456','([a-z]+)([0-9]+)',2)`	Returnerar `123` (andra gruppen i första matchningen).
`ExtractRegExGroup('abc123 def456','([a-z]+)([0-9]+)',2,2)`	Returnerar `456` (andra gruppen i andra matchningen).

Användning

Du kan använda **Extract RegEx()** för att extrahera information som du vill isolera från data som också kan innehålla annan information (t.ex. fritext eller JSON-strängar). Exempel:

- Extrahera e-postadresser, telefonnummer, kontonummer och annan information från text.
- Extrahera numeriska värden från text (t.ex. valuta).
- Standardisera formateringen av text eller numeriska data.

Exempel 1 – laddningsskript för att analysera transaktionskoder

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell med namnet `Transactions`, som innehåller information om transaktioner. Vissa detaljer fångas upp med transaktionskoder som använder följande syntax: *Transaktionsår-Transaktionskälla* (dvs. köp online eller i butik)-*Associerad distributionscentral*
- Skapandet av flera nya fält för att extrahera varje detalj, var och en återanvänder samma sammansatta reguljära uttryck.

Våra krav:

- Årtalet kan vara en valfri kombination av fyra siffror.
- Värdena *ONLINE* och *INSTORE* är de enda godtagbara värdena för transaktionskällan.
- Distributionscentralen måste ha exakt fem nummer.

Laddningsskript

```
Transactions:

Load
       recno() as RecordID,
       ExtractRegExGroup(TransactionCode,'([0-9]{4})-(ONLINE|INSTORE)-([0-9]{5})',0) as
TransactionCode_Unparsed,
       ExtractRegExGroup(TransactionCode,'([0-9]{4})-(ONLINE|INSTORE)-([0-9]{5})',1) as
TransactionYear,
       ExtractRegExGroup(TransactionCode,'([0-9]{4})-(ONLINE|INSTORE)-([0-9]{5})',2) as
TransactionSource,
       ExtractRegExGroup(TransactionCode,'([0-9]{4})-(ONLINE|INSTORE)-([0-9]{5})',3) as
TransactionDC,
* Inline `
TransactionCode, Category
2025-ONLINE-60019, Product A
2024-INSTORE-60020, Product B
2025-ONLINE-60018, Product C
2024-ONLINE-60020, Product A
```

```
2025-INSTORE-60019, Product B
2025-ONLINE-60017, Product D
`;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `RecordID`
- `TransactionCode`
- `TransactionCode_Unparsed`
- `TransactionYear`
- `TransactionSource`
- `TransactionDC`

Resultattabell

RecordID	TransactionCode	TransactionCode_Unparsed	TransactionYear	TransactionSource	TransactionDC
1	2025-ONLINE-60019	2025-ONLINE-60019	2025	ONLINE	60019
2	2024-INSTORE-60020	2024-INSTORE-60020	2024	INSTORE	60020
3	2025-ONLINE-60018	2025-ONLINE-60018	2025	ONLINE	60018
4	2024-ONLINE-60020	2024-ONLINE-60020	2024	ONLINE	60020
5	2025-INSTORE-60019	2025-INSTORE-60019	2025	INSTORE	60019
6	2025-ONLINE-60017	2025-ONLINE-60017	2025	ONLINE	60017

Dessa resultat visar hur argumentet **group** gör att du kan återanvända en enda regex för flera operationer. Fältet `TransactionCode_Unparsed`, som använder ett **group**-värde på 0, ger inget ytterligare värde i det här fallet, men visas här för att demonstrera funktionen.

Exempel 2 – laddningsskript för att extrahera kontaktinformation för företag

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell som heter `BusinessContactInfo`, som innehåller information om företag som hittades på respektive företags webbplats.
- `ContactInfo` är ett fält som innehåller fritext, inklusive företagets e-postadresser och telefonnummer.
- Skapandet av flera nya fält för att extrahera varje detalj, var och en återanvänder samma sammansatta reguljära uttryck.

Våra krav:

- E-postadresser måste uppfylla en specifik uppsättning krav och syntax.
- Telefonnumret måste vara ett NANP-nummer med 10 siffror. Vi vill tillåta att riktnumret sätts inom parentes och tillåta blanksteg och bindestreck på vissa ställen.

Laddningsskript

```
BusinessContactInfo:

Load
       ExtractRegExGroupI(ContactInfo, '([a-zA-Z0-9!#$%^&*-_+=~{|}\/.'']+@[a-zA-Z0-9!#$%^&*-_+=~
{|}\/.'']{1,50}\.[a-zA-Z0-9!#$%^&*\-_+=~{|}\/.'']{1,50})|(\({0,1}[0-9]{3}\){0,1}[ -]*[0-9]{3}[
-]*[0-9]{4})',1,1) as CompanyEmail,
       ExtractRegExGroupI(ContactInfo, '([a-zA-Z0-9!#$%^&*-_+=~{|}\/.'']+@[a-zA-Z0-9!#$%^&*-_+=~
{|}\/.'']{1,50}\.[a-zA-Z0-9!#$%^&*\-_+=~{|}\/.'']{1,50})|(\({0,1}[0-9]{3}\){0,1}[ -]*[0-9]{3}[
-]*[0-9]{4})',2,2) as CompanyPhoneNum,
* Inline `
ID     CompanyName     ContactInfo
1      Company A       Email is: Company1@example.com, Phone number is: (123) 456-7890
2      Company B       Email is: company2@test.com, Phone # is: 0123456790
3      Company C       Email is: company3@placeholder.com, Phone no. is: 234-567-8901

` (delimiter is '\t');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `CompanyName`
- `CompanyEmail`
- `CompanyPhoneNum`

Resultattabell

CompanyName	CompanyEmail	CompanyPhoneNum
Company A	Company1@example.com	(123) 456-7890
Company B	company2@test.com	0123456790
Company C	company3@placeholder.com	234-567-8901

Samma sammansatta reguljära uttryck återanvänds för att hämta olika information. Argumentet **group** anger vilken av de två grupperna i det reguljära uttrycket som ska sökas, och argumentet **field_no** anger vilken övergripande matchning (över hela strängen) som vi vill hitta.

ExtractRegEGroupxI()-varianten av funktionen säkerställer sökningar utan skiftlägeskänslighet.

Exempel 3 – laddningsskript för att extrahera komponenter från ISBN-koder

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Skapandet av en variabel, `ISBN_RegEx`, för att lagra det reguljära uttryck som vi vill använda för alla extraktioner.
- En uppsättning ISBN-koder för böcker som en bokhandel vill beställa från flera leverantörer.

Vi måste extrahera följande från varje ISBN-kod:

- EAN
- Grupp
- Registrant
- Publication
- Checksum

Laddningsskript

```
SET ISBN_RegEx = 'ISBN[ ]*([0-9]{3})-([0-9]{1})-([0-9]{4})-([0-9]{4})-([0-9]{1})';

ISBN:
LOAD Supplier,
ExtractRegExGroup(Books,'$(ISBN_RegEx)',1) AS EAN,
ExtractRegExGroup(Books,'$(ISBN_RegEx)',2) AS Group,
ExtractRegExGroup(Books,'$(ISBN_RegEx)',3) AS Registrant,
ExtractRegExGroup(Books,'$(ISBN_RegEx)',4) AS Publication,
ExtractRegExGroup(Books,'$(ISBN_RegEx)',5) AS Checksum;
// Split the ISBN with the Group function in a preceding load to avoid generating a cartesian
product
LOAD *,
ExtractRegEx(SupplierBooks, '$(ISBN_RegEx)') AS Books
INLINE [
Supplier, SupplierBooks
Supplier 1, ISBN 123-3-1234-1234-0 ISBN 012-2-0123-0123-4 ISBN 000-1-0123-0123-2 ISBN 234-5-
2345-2345-1 ISBN 555-2-5555-5555-3 ISBN 222-4-2222-2222-2
Supplier 2, ISBN 000-0-3333-3333-3 ISBN 333-3-3333-3333-3 ISBN 555-1-5151-5151-3 ISBN 232-1-
2323-2323-1 ISBN 008-0-7777-7777-3 ISBN 888-0-9999-0000-0
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Supplier`
- `EAN`
- `Group`
- `Registrant`
- `Publication`
- `Checksum`

Resultattabell

Supplier	EAN	Group	Registrant	Publication	Checksum
Supplier 1	000	1	0123	0123	2
Supplier 1	012	2	0123	0123	4
Supplier 1	123	3	1234	1234	0
Supplier 1	222	4	2222	2222	2
Supplier 1	234	5	2345	2345	1
Supplier 1	555	2	5555	5555	3
Supplier 2	000	0	3333	3333	3
Supplier 2	008	0	7777	7777	3
Supplier 2	232	1	2323	2323	1
Supplier 2	333	3	3333	3333	3
Supplier 2	555	1	5151	5151	3
Supplier 2	888	0	9999	0000	0

Exempel 4 – diagramuttryck för att extrahera kontaktinformation för företag (med jämförelse med **ExtractRegEx()**)

Diagramuttryck och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell som heter `BusinessContactInfo`, som innehåller information om företag som hittades på respektive företags webbplats.

- `ContactInfo` är ett fält som innehåller fritext, inklusive företagets e-postadresser och telefonnummer. Vi vill extrahera varje e-postadress och telefonnummer med diagramuttryck.

Våra krav:

- E-postadresser måste uppfylla en specifik uppsättning krav och syntax.
- Telefonnumret måste vara ett NANP-nummer med 10 siffror. Vi vill tillåta att riktnumret sätts inom parentes och tillåta blanksteg och bindestreck på vissa ställen.

Laddningsskript

```
BusinessContactInfo:

Load * Inline `
ID      CompanyName     ContactInfo
1       Company A       Email is: Company1@example.com, Phone number is: (123) 456-7890
2       Company B       Email is: company2@test.com, Phone # is: 0123456790
3       Company C       Email is: company3@placeholder.com, Phone no. is: 234-567-8901

` (delimiter is '\t');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `CompanyName`

Lägg till följande beräknade dimensioner i tabellen:

- `=ExtractRegExGroupI(ContactInfo, '([a-zA-Z0-9!#$%^&*-_+=~{|}\/.'']+@[a-zA-Z0-9!#$%^&*-_+=~{|}\/.'']{1,50}\.[a-zA-Z0-9!#$%^&*\-_+=~{|}\/.'']{1,50})|(\({0,1}[0-9]{3}\){0,1}[ -]*[0-9]{3}[ -]*[0-9]{4})',1,1)`
- `=ExtractRegExGroupI(ContactInfo, '([a-zA-Z0-9!#$%^&*-_+=~{|}\/.'']+@[a-zA-Z0-9!#$%^&*-_+=~{|}\/.'']{1,50}\.[a-zA-Z0-9!#$%^&*\-_+=~{|}\/.'']{1,50})|(\({0,1}[0-9]{3}\){0,1}[ -]*[0-9]{3}[ -]*[0-9]{4})',2,2)`

Resultattabell

CompanyName	=ExtractRegExGroupI(ContactInfo, '([a-zA-Z0-9!#$%^&*-_+=~{\|}\/."]+@[a-zA-Z0-9!#$%^&*-_+=~{|}\/."]{1,50}\.[a-zA-Z0-9!#$%^&*\-_+=~{|}\/."]{1,50})|(\({0,1}[0-9]{3}\){0,1}[-]*[0-9]{3}[-]*[0-9]{4})',1,1)	=ExtractRegExGroupI(ContactInfo, '([a-zA-Z0-9!#$%^&*-_+=~{\|}\/."]+@[a-zA-Z0-9!#$%^&*-_+=~{|}\/."]{1,50}\.[a-zA-Z0-9!#$%^&*\-_+=~{|}\/."]{1,50})|(\({0,1}[0-9]{3}\){0,1}[-]*[0-9]{3}[-]*[0-9]{4})',2,2)
Company A	Company1@example.com	(123) 456-7890
Company B	company2@test.com	0123456790
Company C	company3@placeholder.com	234-567-8901

Samma sammansatta reguljära uttryck återanvänds för att hämta olika information. Argumentet **group** anger vilken av de två grupperna i det reguljära uttrycket som ska sökas, och argumentet **field_no** anger vilken övergripande matchning (över hela strängen) som vi vill hitta.

ExtractRegEGroupI()-varianten av funktionen säkerställer sökningar utan skiftlägeskänslighet.

Exempel 5 – URL-parsning (med jämförelse på **ExtractRegEx()**)

Diagramuttryck och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell med namnet `Correspondence`, som innehåller e-postmeddelandetexten i ett fält med namnet `EmailBody`.
- Innehåll i e-postmeddelanden som innehåller URL:er.

Vi vill extrahera den andra URL:en, om den finns, i följande komponenter:

- Full URL
- Protocol
- Domän
- Sökväg

Vi vill använda ett enda reguljärt uttryck för att styra all extrahering.

Laddningsskript

```
Correspondence:

Load * Inline `
ID     EmailBody
1      Thanks again for this morning's meeting! You can find the meeting minutes posted here:
https://example.com/resourceexample. If you still have any questions, always feel free to ask
me or one of the other team members. Here are a few learning resources that might help you:
http://www.example.ca/training1.pptx http://www.example.ca/training2.pptx
http://www.example.ca/training3.pptx Thanks again!
2      Hi, you'll want to visit our company website for that, it's available at
https://www.example.se.
3      Hello all, I just wanted to let you know that our online stores are now up and running! I
couldn't be more excited. We are already seeing quite a bit of traffic and volume sold, which
is very promising! For Product A, go to https://www.examplestore1.com/products. For Product B,
you'll want go to https://www.examplestore2.com/products. Product C, go check out
https://www.examplestore3.com/products. Cheers!

` (delimiter is '\t');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `ID`

Lägg till dess beräknade dimensioner:

- *URL 2*:
  `=ExtractRegEx(EmailBody,'([a-zA-Z0-9]+):\/\/(([a-zA-Z0-9]+\.)*([a-zA-Z0-9]+)\.([a-zA-Z0-9]+))((\/[a-zA-Z0-9]+)*(\/([a-zA-Z0-9]+)(\.[a-zA-Z0-9]+)?))?',2)`
- *URL 2 Protocol*:
  `=ExtractRegExGroup(EmailBody,'([a-zA-Z0-9]+):\/\/(([a-zA-Z0-9]+\.)*([a-zA-Z0-9]+)\.([a-zA-Z0-9]+))((\/[a-zA-Z0-9]+)*(\/([a-zA-Z0-9]+)(\.[a-zA-Z0-9]+)?))?',1,2)`
- *URL 2 Domain*:
  `=ExtractRegExGroup(EmailBody,'([a-zA-Z0-9]+):\/\/(([a-zA-Z0-9]+\.)*([a-zA-Z0-9]+)\.([a-zA-Z0-9]+))((\/[a-zA-Z0-9]+)*(\/([a-zA-Z0-9]+)(\.[a-zA-Z0-9]+)?))?',2,2)`
- *URL 2 Path*:
  `=ExtractRegExGroup(EmailBody,'([a-zA-Z0-9]+):\/\/(([a-zA-Z0-9]+\.)*([a-zA-Z0-9]+)\.([a-zA-Z0-9]+))((\/[a-zA-Z0-9]+)*(\/([a-zA-Z0-9]+)(\.[a-zA-Z0-9]+)?))?',6,2)`

Resultattabell

ID	URL 2	URL 2 Protocol	URL 2 Domain	URL 2 Path
1	http://www.example.ca/training1.pptx	http	www.example.ca	/training1.pptx
2	-	-	-	-
3	https://www.examplestore2.com/products	https	www.examplestore2.com	/products

ExtractRegEx() returnerar hela URL:en, medan **ExtractRegExGroup()** returnerar enskilda delar av URL:en, vilket motsvarar det **group**-värde vi använde. Posten med värdet 2 på `ID` innehåller bara en URL, så inga data returneras för den.

Samma reguljära uttryck används i alla diagramuttryck för att returnera olika typer av information. En uppdelning av de grupper som definieras i det reguljära uttrycket följer.

Grupper i reguljärt uttryck

Group number	Regex	URL part
1	([a-zA-Z0-9]+)	Protokoll
2	(([a-zA-Z0-9]+\.)*([a-zA-Z0-9]+)\.([a-zA-Z0-9]+))	Domain (grupp 3, 4 och 5)
3	([a-zA-Z0-9]+\.)*	Root domain
4	([a-zA-Z0-9]+)	Top-level domain

Group number	Regex	URL part
5	([a-zA-Z0-9]+)	Path (grupp 7, 8, 9 och 10) (valfritt)
6	((\/[a-zA-Z0-9]+)*(\/([a-zA-Z0-9]+)(\.[a-zA-Z0-9]+)?))?	Sub-directory
7	(\/[a-zA-Z0-9]+)*	File (grupp 9 och 10)
8	(\/([a-zA-Z0-9]+)(\.[a-zA-Z0-9]+)?)	Fil
9	([a-zA-Z0-9]+)	Filnamn
10	(\.[a-zA-Z0-9]+)?	Komplement till fil (valfritt)

FindOneOf

FindOneOf() söker en sträng för att hitta positionen för eventuella tecken som ingår i en uppsättning angivna tecken. Positionen för den N:e förekomsten av ett tecken från sökuppsättningen returneras, där N är den valfria tredje parametern för funktionen. Om ingen tredje parameter anges returneras den första förekomsten. Om ingen motsvarighet påträffas, returneras **0**.

Syntax:

```
FindOneOf(text, char_set[, count])
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
char_set	En teckenuppsättning att söka efter i **text**.
count	Definierar vilka förekomster av något av tecknen som sökningen ska inkludera. Till exempel, söker värdet 2 efter den andra förekomsten.

Exempel: diagramuttryck

Exempel	Resultat
`FindOneOf( 'my example text string', 'et%s' )`	Returnerar `4` eftersom `e` är det fjärde tecknet i exempelsträngen.
`FindOneOf( 'my example text string', 'et%s', 3 )`	Returnerar `12` eftersom sökningen avser något av tecknen `e`, `t`, `%` eller `s`, och `t` är den tredje förekomsten i position `12` i exempelsträngen.
`FindOneOf( 'my example text string', '¤%&' )`	Returnerar `0` eftersom inget av tecknen `¤`, `%` eller `&` förekommer i exempelsträngen.

Exempel – grunderna i FindOneOf

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen som heter `Phrase`, som innehåller de ursprungliga textsträngarna som ska bearbetas.

Laddningsskript

```
Example:
Load *
Inline
[Phrase
Many tiny beads
For a very long time
Has the potential for growth
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Phrase`

Skapa följande beräknade dimensioner:

- `=FindOneOf(Phrase, 'e', 1)`
- `=FindOneOf(Phrase, 'y', 1)`
- `=FindOneOf(Phrase, 'ey', 1)`
- `=FindOneOf(Phrase, 'e', 2)`
- `=FindOneOf(Phrase, 'y', 2)`
- `=FindOneOf(Phrase, 'ey', 2)`

Resultattabell

Phrase	FindOneOf (Phrase, 'e', 1)	FindOneOf (Phrase, 'y', 1)	FindOneOf (Phrase, 'ey', 1)	FindOneOf (Phrase, 'e', 2)	FindOneOf (Phrase, 'y', 2)	FindOneOf (Phrase, 'ey', 2)
For a very long time	8	10	8	20	0	10

Phrase	FindOneOf (Phrase, 'e', 1)	FindOneOf (Phrase, 'y', 1)	FindOneOf (Phrase, 'ey', 1)	FindOneOf (Phrase, 'e', 2)	FindOneOf (Phrase, 'y', 2)	FindOneOf (Phrase, 'ey', 2)
Has the potential for growth	7	0	7	12	0	12
Many tiny beads	12	4	4	0	9	9

För varje beräknad dimension returnerar utdata positionen för förekomsten av tecknen från sökuppsättningen för varje fras. Den första förekomsten av bokstaven `e` finns t.ex. på position `8`, `7` och `12` för respektive fras. Den andra förekomsten av bokstaven `e` i varje fras finns på position `20`, `12`, och `0` (hittades inte). Den första förekomsten av bokstäverna "`e` eller `y`"finns t.ex. på position `8`, `7` och `4` för respektive fras. Lägg märke till att i frasen `For a very long time` finns den första förekomsten av "`e` eller `y`" på position `8`, ett `e`, och den andra förekomsten på position `10`, ett `y`.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Example:
Load *, FindOneOf(InputText, SearchFor, Occurrence) AS FindOneOf_Matches
Inline
InputText, SearchFor, Occurrence
my example text string, et%s,1
my example text string, et%s,3
my example text string, ¤%&,1
];
```

Resultattabell

Indatatext	SökEfter	Förekomst	FindOneOf_Matchningar
mitt textsträngsexempel	och%s	1	4
mitt textsträngsexempel	och%s	3	12
mitt textsträngsexempel	¤%&	1	0

Exempel – FindOneOf-scenariot

Diagramuttryck

Översikt

I det här exemplet används funktionen `FindOneOf` för att analysera kundkommentarer i ett dataset med kundorder och identifiera order som kan kräva åtgärder. Varje order har ett `CustomerComment`-fält där kunderna kan lämna anteckningar eller kommentarer om sina order. Genom att analysera kommentarerna kan du identifiera specifika nyckelord eller tecken, t.ex. "!", "@", "#", som kan tyda på brådska eller särskilda önskemål.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `TicketID` ärendets id-nummer
  - `CustomerComment` den ursprungliga textsträngen

Laddningsskript

```
Example:
Load * inline [
TicketID, CustomerComment
1, I need this order ASAP!
2, Please confirm my order @12345.
3, Can you update my order?
4, I have a question about #discount.
5, Thank you!
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `TicketID`
- `CustomerComment`

Skapa följande beräknade dimensioner:

- `=FindOneOf(CustomerComment, '!,@,#')`
- `=If(FindOneOf(CustomerComment, '!,@,#')>0,'High Priority','Normal Priority')`

Resultattabell

TicketID	Kundkommentar	FindOneOf (CustomerComment, '!,@,#')	If(FindOneOf(CustomerComment, '!,@,#')>0,'Hög prioritet','Normal prioritet')
1	Jag behöver denna order så snart som möljigt!	23	Hög prioritet
2	Bekräfta min order @12345.	25	Hög prioritet
3	Kan ni uppdatera min order?	0	Normal prioritet
4	Jag har en fråga om #discount.	25	Hög prioritet
5	Tack!	10	Hög prioritet

Jämför utdata från funktionen `FindOneOf` med de ursprungliga `CustomerComment`-strängvärdena som laddades i skriptet. Med hjälp av värdena `!,@,#`, kunde funktionen returnera den position där dessa tecken förekommer i kommentarssträngen. Den slutliga åtgärden använder en villkorlig `If`-sats för att identifiera poster som innehåller något av sökteckнen och markera dem som `High Priority`. Om inget av sökteckнen `!,@,#` hittas, t.ex. rad 3, klassificeras ärendet som `Normal Priority`.

Hash128

Hash128() returnerar ett 128-bitars hashvärde av de kombinerade indatauttrycksvärdena. Resulterar i en 22-teckensträng. Hashvärden är användbara för att maskera personliga identitetsuppgifter (PII), t.ex. kundnamn, personnummer eller kontonummer.

Syntax:

```
Hash128(expr{, expression})
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
expr	Sträng som ska utvärderas.

Exempel: diagramuttryck

Exempel	Resultat
`Hash128( 'abc', 'xyz', '123' )`	Returnerar `MA&5]6+3=:>:>G%S<U*S2+`
`Hash128 ( Region, Year, Month )` Note: `Region`, `Year`, and `Month` are table fields.	Returnerar `G7*=6GKPJ(Z+)^KM?<$'A+`

Exempel – grunderna i Hash128

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält som kommer att användas för att skapa ett 128-bitars hashvärde:
  - `Region`
  - `Year`
  - `Month`

Laddningsskript

```
Example:
Load *
inline [
Region, Year, Month
abc, xyz, 123
EU, 2022, 01
UK, 2022, 02
US, 2022, 02
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Region`
- `Year`
- `Month`

Skapa följande beräknade dimension:

- `=Hash128(Region, Year, Month)` för att beräkna 128-bitars hashvärde.

Resultattabell

Region	År	Månad	Hash128(Region, År, Månad)
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2+
EU	2022	01	B40^K&[T@!;VB'XR]<5=/$
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!
US	2022	02	C6@#]4#_G-(]J7EQY#KRW0

För varje rad sammanfogas hashsträngarna för `Region`, `Year` och `Month` och returnerar en sträng med 22 tecken.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Hash_128:
Load *,
Hash128(Region, Year, Month) as Hash128_Region_Year_Month;
Load * inline [
Region, Year, Month
abc, xyz, 123
EU, 2022, 01
UK, 2022, 02
US, 2022, 02 ];
```

Resultattabell

Region	År	Månad	Hash128_Region_År_Månad
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2+
EU	2022	01	B40^K&[T@!;VB'XR]<5=/$
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!
US	2022	02	C6@#]4#_G-(]J7EQY#KRW0

Exempel – Hash128-scenario med kopior av poster

Diagramuttryck

Översikt

I det här exemplet skapas ett unikt hashvärde för varje produktpost. Hashvärdet fungerar som en unik identifierare, vilket gör att företaget kan upptäcka och hantera kopior på ett effektivt sätt.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `TransactionID`
  - `ProductID`
  - `ProductName`
  - `SupplierID`

Laddningsskript

```
Example:
Load *
inline [
TransactionID, ProductID, ProductName, SupplierID
1, 101, Widget A, S001, 100
2, 102, Widget B, S002, 200
3, 101, Widget A, S001, 50
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Transaction ID`
- `ProductID`
- `ProductName`
- `SupplierID`

Skapa följande beräknade dimension:

- `=Hash128(ProductID, ProductName, SupplierID)`, för att beräkna en nyckel med 22 tecken.

Resultattabell

Transaktions-ID	Produkt-ID	Produktnamn	Leverantörs-ID	Hash128(ProduktID, Produktnamn, LeverantörsID)
1	101	Widget A	S001	CY`&^ (N,E/#`TJT4&]/7N%
2	102	Widget B	S002	O7(1;V1%MG;Z+A [/H)G5&0
3	101	Widget A	S001	CY`&^ (N,E/#`TJT4&]/7N%

Jämför utdata från `Hash128`-funktionen. Lägg märke till att den första och tredje posten `TransactionID` har kopierade hashvärden eftersom kombinationen av strängarna `ProductID`, `ProductName` och `SupplierID` är densamma för dessa rader.

Exempel – Hash128-scenario för att anonymisera data

Diagramuttryck

Översikt

Ett företag inom finansiella tjänster hanterar känsliga kunduppgifter, till exempel kreditkortsnummer, personnummer och kundnamn. För att skydda integriteten och säkerställa efterlevnad av bestämmelser som allmänna dataskyddsförordningen (GDPR) eller lagen om rätt till sjukförsäkring och ersättning (HIPAA) måste de anonymisera känsliga data när de skapar rapporter för analys. Men de måste också vara konsekventa, så att de kan spåra samma kund i olika datauppsättningar utan att avslöja kundens identitet.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `CustomerID`
  - `CustomerName`
  - `SSN`

Laddningsskript

```
Example:
Load *
inline [
```

```
CustomerID, CustomerName, SSN
101, John Doe, 123-45-6789|
102, Jane Smith, 987-65-4321
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `CustomerID`
- `CustomerName`
- `SSN`

Skapa följande beräknade dimensioner:

- `=Hash128(CustomerName)`
- `=Hash128(SSN)`

Resultattabell

Kund-id	Kundnamn	Personnummer	Hash128(Kundnamn)	Hash128 (Personnummer)
101	John Doe	123-45-6789|	GMD (*B9*!PQ&MTY2@$4A\+	N& [PSH:I$M?OF<5/9LV&N#
102	Jane Smith	987-65-4321	EP@9]GSI)5G\I*,B?H,C?)	HCFXMR'2=\3WF6?_ \6LN;$

Utdata genererar unika nycklar som har följande fördelar:

- Datasekretess: känsliga originaldata är inte direkt synliga, men hashvärdet förblir konsekvent mellan olika poster. Detta möjliggör analys, till exempel aggregering av kunders aktiviteter, utan att avslöja kundens faktiska identitet.
- Efterlevnad av regler: genom att hasha känsliga fält säkerställer företaget att man följer reglerna för datasekretess.
- Säkerhet: hashfunktionen producerar en fastställd storlek på icke-reversibla utdata , vilket ger ett extra skikt av skydd för känsliga data.

Hash160

Hash160() returnerar ett 160-bitars hashvärde av de kombinerade indatauttrycksvärdena. Resulterar i en 27-teckensträng. Hashvärden är användbara för att maskera personliga identitetsuppgifter (PII), t.ex. kundnamn, personnummer eller kontonummer.

Syntax:

```
Hash160(expr{, expression})
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
expr	Sträng som ska utvärderas.

Exempel: diagramuttryck

Exempel	Resultat
`Hash160( 'abc', 'xyz', '123' )`	Returnerar `MA&5]6+3=:>;>G%S<U*S2I:`=X*`
`Hash160( Region, Year, Month )` Note: `Region`, `Year`, and `Month` are table fields.	Returnerar `G7*=6GKPJ (Z+)^KM?<$'AI.)?U$`

Exempel – grunderna i Hash160

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält som kommer att användas för att skapa ett 160-bitars hashvärde:
  - `Region`
  - `Year`
  - `Month`

Laddningsskript

```
Example:
Load *
inline [
Region, Year, Month
abc, xyz, 123
EU, 2022, 01
UK, 2022, 02
US, 2022, 02
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Region`
- `Year`
- `Month`

Skapa följande beräknade dimension:

- `=Hash160(Region, Year, Month)` för att beräkna 160-bitars hashvärde.

Resultattabell

Region	År	Månad	Hash160(Region, Year, Month)
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2I:`=X*
EU	2022	01	B40^K&[T@!;VB'XR]<5=//_F853
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!T"FWZ
US	2022	02	C6@#]4#_G-(]J7EQY#KRW`@KF+W

För varje rad sammanfogas hashsträngarna för `Region`, `Year` och `Month` och returnerar en sträng med 27 tecken.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Hash_160:
Load *,
Hash160(Region, Year, Month) as Hash160_Region_Year_Month;
Load * inline [
Region, Year, Month
abc, xyz, 123
EU, 2022, 01
UK, 2022, 02
US, 2022, 02 ];
```

Resultattabell

Region	År	Månad	Hash160_Region_Year_Month
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2I:`=X*
EU	2022	01	B40^K&[T@!;VB'XR]<5=//_F853
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!T"FWZ
US	2022	02	C6@#]4#_G-(]J7EQY#KRW`@KF+W

Exempel – Hash160-scenario med kopior av poster

Diagramuttryck

Översikt

I det här exemplet skapas ett unikt hashvärde för varje produktpost. Hashvärdet fungerar som en unik identifierare, vilket gör att företaget kan upptäcka och hantera kopior på ett effektivt sätt.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `TransactionID`
  - `ProductID`
  - `ProductName`
  - `SupplierID`

Laddningsskript

```
Example:
Load *
inline [
TransactionID, ProductID, ProductName, SupplierID
1, 101, Widget A, S001
2, 102, Widget B, S002
3, 101, Widget A, S001
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `TransactionID`
- `ProductID`
- `ProductName`
- `SupplierID`

Skapa följande beräknade dimension:

- `=Hash160(ProductID, ProductName, SupplierID)` för att beräkna en nyckel med 27 tecken.

Resultattabell

TransactionID	Produkt-ID	ProductName	SupplierID	Hash160(ProductID, ProductName, SupplierID)
1	101	Widget A	S001	CY`&^ (N,E/#`TJT4&]/7N1Q+*T3
2	102	Widget B	S002	O7(1;V1%MG;Z+A [/H)G5&`6RA0(
3	101	Widget A	S001	CY`&^ (N,E/#`TJT4&]/7N1Q+*T3

Jämför utdata från `Hash160`-funktionen. Lägg märke till att den första och tredje posten `TransactionID` har kopierade hashvärden eftersom kombinationen av strängarna `ProductID`, `ProductName` och `SupplierID` är densamma för dessa rader.

Exempel – Hash160-scenario för att anonymisera data

Diagramuttryck

Översikt

Ett företag inom finansiella tjänster hanterar känsliga kunduppgifter, till exempel kreditkortsnummer, personnummer och kundnamn. För att skydda integriteten och säkerställa efterlevnad av bestämmelser som allmänna dataskyddsförordningen (GDPR) eller lagen om rätt till sjukförsäkring och ersättning (HIPAA) måste de anonymisera känsliga data när de skapar rapporter för analys. Men de måste också vara konsekventa, så att de kan spåra samma kund i olika datauppsättningar utan att avslöja kundens identitet.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `CustomerID`
  - `CustomerName`
  - `SSN`

Laddningsskript

```
Example:
Load *
inline [
CustomerID, CustomerName, SSN
101, John Doe, 123-45-6789|
102, Jane Smith, 987-65-4321
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `CustomerID`
- `CustomerName`
- `SSN`

Skapa följande beräknade dimensioner:

- `=Hash160(CustomerName)`
- `=Hash160(SSN)`

Resultattabell

Kund-id	Kundnamn	SSN	Hash160(CustomerName)	Hash160(SSN)
101	John Doe	123-45-6789|	GMD (*B9*!PQ&MTY2@$4A\KCL?[0	N& [PSH:I$M?OF<5/9LV&N*;`CZ (
102	Jane Smith	987-65-4321	EP@9]GSI)5G\I*,B?H,C?A$%VG:	HCFXMR'2=\3WF6?_ \6LN;.*6:K?

Utdata genererar unika nycklar som har följande fördelar:

- Datasekretess: känsliga originaldata är inte direkt synliga, men hashvärdet förblir konsekvent mellan olika poster. Detta möjliggör analys, till exempel aggregering av kunders aktiviteter, utan att avslöja kundens faktiska identitet.
- Efterlevnad av regler: genom att hasha känsliga fält säkerställer företaget att man följer reglerna för datasekretess.
- Säkerhet: hashfunktionen producerar en fastställd storlek på icke-reversibla utdata , vilket ger ett extra skikt av skydd för känsliga data.

Hash256

Hash256() returnerar ett 256-bitars hashvärde av de kombinerade indatauttrycksvärdena. Resulterar i en 43-teckensträng. Hashvärden är användbara för att maskera personliga identitetsuppgifter (PII), t.ex. kundnamn, personnummer eller kontonummer.

Syntax:

```
Hash256(expr{, expression})
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
expr	Sträng som ska utvärderas.

Exempel: diagramuttryck

Exempel	Resultat
`Hash256( 'abc', 'xyz', '123' )`	Returnerar `MA&5]6+3=:>;>G%S<U*S2I:`=X*A.IO*8N\%Y7Q;YEJ`
`Hash256( Region, Year, Month )` Note: `Region`, `Year`, and `Month` are table fields.	Returnerar `G7*=6GKPJ(Z+)^KM?<$'AI.)?U$#X2RB[:0ZP=+Z`F:`

Exempel – grunderna i Hash256

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält som kommer att användas för att skapa ett 256-bitars hashvärde:
  - `Region`
  - `Year`
  - `Month`

Laddningsskript

```
Example:
Load *
inline [
Region, Year, Month
abc, xyz, 123
EU, 2022, 01
UK, 2022, 02
US, 2022, 02
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Region`
- `Year`
- `Month`

Skapa följande beräknade dimension:

- `=Hash256(Region, Year, Month)` för att beräkna 256-bitars hashvärde.

Resultattabell

Region	År	Månad	Hash256(Region, År, Månad)
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2I:`=X*A.IO*8N\%Y7Q;YEJ
EU	2022	01	B40^K&[T@!;VB'XR]<5=//_F853?BE6'G&,YH*T'MF)
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!T"FWZT=4\#V`M%6_\0C>4
US	2022	02	C6@#]4#_G-(]J7EQY#KRW`@KF+W-0]`[Z8R+#'")=+0

För varje rad sammanfogas hashsträngarna för `Region`, `Year` och `Month` och returnerar en sträng med 43 tecken.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Hash_256:
Load *,
Hash256(Region, Year, Month) as Hash256_Region_Year_Month;
Load * inline [
Region, Year, Month
abc, xyz, 123
EU, 2022, 01
UK, 2022, 02
US, 2022, 02 ];
```

Resultattabell

Region	År	Månad	Hash256_Region_År_Månad
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2I:`=X*A.IO*8N\%Y7Q;YEJ
EU	2022	01	B40^K&[T@!;VB'XR]<5=//_F853?BE6'G&,YH*T'MF)
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!T"FWZT=4\#V`M%6_\0C>4
US	2022	02	C6@#]4#_G-(]J7EQY#KRW`@KF+W-0]`[Z8R+#'")=+0

Exempel – Hash256-scenario med kopior av poster

Diagramuttryck

Översikt

I det här exemplet skapas ett unikt hashvärde för varje produktpost. Hashvärdet fungerar som en unik identifierare, vilket gör att företaget kan upptäcka och hantera kopior på ett effektivt sätt.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:

- `TransactionID`
- `ProductID`
- `ProductName`
- `SupplierID`

Laddningsskript

```
Example:
Load *
inline [
TransactionID, ProductID, ProductName, SupplierID
1, 101, Widget A, S001, 100
2, 102, Widget B, S002, 200
3, 101, Widget A, S001, 50
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Transaction ID`
- `ProductID`
- `ProductName`
- `SupplierID`

Skapa följande beräknade dimension:

- `=Hash256(ProductID, ProductName, SupplierID)` för att beräkna en nyckel med 43 tecken.

Resultattabell

Transaktions-ID	Produkt-ID	Produktnamn	Leverantörs-ID	Hash256(ProduktID, Produktnamn, LeverantörsID)
1	101	Widget A	S001	CY`&^ (N,E/#`TJT4&]/7N1Q+*T3K=35VDVQ>>; %+A0]
2	102	Widget B	S002	O7(1;V1%MG;Z+A[/H)G5&`6RA0 (`%6T?YZQ.+>*&\JF
3	101	Widget A	S001	CY`&^ (N,E/#`TJT4&]/7N1Q+*T3K=35VDVQ>>; %+A0]

Jämför utdata från `Hash256`-funktionen. Lägg märke till att den första och tredje posten `TransactionID` har kopierade hashvärden eftersom kombinationen av strängarna `ProductID`, `ProductName` och `SupplierID` är densamma för dessa rader.

Exempel – Hash256-scenario för att anonymisera data

Diagramuttryck

Översikt

Ett företag inom finansiella tjänster hanterar känsliga kunduppgifter, till exempel kreditkortsnummer, personnummer och kundnamn. För att skydda integriteten och säkerställa efterlevnad av bestämmelser som allmänna dataskyddsförordningen (GDPR) eller lagen om rätt till sjukförsäkring och ersättning (HIPAA) måste de anonymisera känsliga data när de skapar rapporter för analys. Men de måste också vara konsekventa, så att de kan spåra samma kund i olika datauppsättningar utan att avslöja kundens identitet.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `CustomerID`
    - `CustomerName`
    - `SSN`

Laddningsskript

```
Example:
Load *
inline [
CustomerID, CustomerName, SSN
101, John Doe, 123-45-6789|
102, Jane Smith, 987-65-4321
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `CustomerID`
- `CustomerName`
- `SSN`

Skapa följande beräknade dimensioner:

- `=Hash256(CustomerName)`
- `=Hash256(SSN)`

Resultattabell

Kund-id	Kundnamn	Personnummer	Hash256(Kundnamn)	Hash256 (Personnummer)
101	John Doe	123-45-6789|	GMD(*B9*!PQ&MTY2@$4A\KCL?[0@T%D@8C#6&UR8/95	N&[PSH:I$M?OF<5/9LV&N*;`CZ((X'"9Y2FKJC6W04[
102	Jane Smith	987-65-4321	EP@9]GSI)5G\I*,B?H,C?A$%VG:SL#6I>Q1:L<0O7UW	HCFXMR'2=\3WF6?_\6LN;.*6:K?Y:%MZ#*PP]WI`YS<

Utdata genererar unika nycklar som har följande fördelar:

- Datasekretess: känsliga originaldata är inte direkt synliga, men hashvärdet förblir konsekvent mellan olika poster. Detta möjliggör analys, till exempel aggregering av kunders aktiviteter, utan att avslöja kundens faktiska identitet.
- Efterlevnad av regler: genom att hasha känsliga fält säkerställer företaget att man följer reglerna för datasekretess.
- Säkerhet: hashfunktionen producerar en fastställd storlek på icke-reversibla utdata , vilket ger ett extra skikt av skydd för känsliga data.

Index

Index() söker en sträng för att hitta startpositionen för den n:e förekomsten av en angiven delsträng. Ett valbart tredje argument ger värdet för n, vilket annars är 1. Ett negativt värde söker från slutet av strängen. Strängens positioner är numrerade vänster till höger från **1** och uppåt.

Syntax:

```
Index(text, substring[, count])
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
substring	En teckensträng att söka efter i **text**. *Om delsträngen inte finns i texten returnerar Index 0.*
count	Definierar vilken förekomst av **substring** som sökningen ska göras för. Till exempel, söker värdet 2 efter den andra förekomsten. Ett negativt värde börjar från slutet av strängen.

Exempel: diagramuttryck

Exempel	Resultat
`Index( 'abcdefg', 'cd' )`	Returnerar `3`
`Index( 'abcdabcd', 'b', 2 )`	Returnerar `6` (den andra förekomsten av `b`)
`Index( 'abcdabcd', 'b',-2 )`	Returnerar `2` (den andra förekomsten av `b` räknat från slutet)
`Left( Date, Index( Date,'-') -1 )` where `Date = 1997-07-14`	Returnerar `1997`
`Mid( Date, Index( Date, '-', 2 ) -2, 2 )` where `Date = 1997-07-14`	Returnerar `07`
`Index( 'abc', 'x' )`	Returnerar `0` (`x` existerar inte i strängen `abc`)
`Index( 'abc', 'a', 2 )`	Returnerar `0` (det finns ingen andra förekomst av `a`)

Exempel – grunderna i Index

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `InputString`.

Laddningsskript

```
Example:
Load * inline [
InputString
abcdefg
abcdabcd ];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `InputString`

Skapa följande beräknade dimensioner:

- `=Index(InputString, 'cd')`
- `=Index(InputString, 'e')`

- `=Index(InputString, 'b', -1)`

Resultattabell

Indatasträng	Index(Indatasträng, 'cd')	Index(Indatasträng, 'e')	Index(Indatasträng, 'b', -1)
abcdabcd	3	0	6
abcdefg	3	5	2

Den första raden med strängen `abcdabcd` returnerar:

- `3` - den första förekomsten av `cd`
- `0` - för att `e` inte finns i strängen
- `6` - den första förekomsten av `b` räknat från slutet av strängen

Den andra raden med strängen `abcdefg` returnerar:

- `3` - den första förekomsten av `cd`
- `5` - den första förekomsten av `e`
- `2` - den första förekomsten av `b` räknat från slutet av strängen

En negativ `count`-parameter skannar strängen i omvänd riktning. Den position som returneras anges dock alltid från vänster till höger, även om den skannas baklänges.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Example:
Load
*,
index(String, 'cd') as Index_CD,              // returns 3  in Index_CD
index(String, 'b') as Index_B,               // returns 2 in Index_B
index(String, 'b', -1) as Index_B2;          // returns 2 or 6 in Index_B2
Load * inline [
String
abcdefg
abcdabcd ];
```

Resultattabell

Sträng	Index_CD	Index_B	Index_B2
abcdefg	3	2	2
abcdabcd	3	2	6

Exempel – Index-scenario

Diagramuttryck

Översikt

Ett kundsupportcenter har en datauppsättning som innehåller ämnesrader från e-postmeddelanden från kunder. Supportcentret vill kategorisera dessa e-postmeddelanden baserat på de nyckelord som förekommer i ämnesraden, till exempel `Refund`, `Complaint` och `Technical Issue`. Ämnesraderna lagras i ett fält i datasetet som heter `EmailSubject`.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `EmailSubject`.

Laddningsskript

```
Example:
Load * inline [
EmailSubject
Request for a Refund - Order 12345
Technical Issue with Product XYZ
Complaint about delivery service
Follow-up on Refund Request
Technical Issue not resolved
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `EmailSubject`

Skapa följande beräknade dimension:

- `=If(Index(EmailSubject, 'Refund') > 0, 'Refund', 'Other')`, för att beräkna vilken ämnestext i e-postmeddelanden som avser återbetalningar.

Resultat

E-postämne	**Om(Index(EmailSubject, 'Återbetalning') > 0, 'Återbetalning', 'Övrigt')**
Klagomål på leveransservice	Övrigt
Uppföljning av begäran om återbetalning	Återbetalning

E-postämne	Om(Index(EmailSubject, 'Återbetalning') > 0, 'Återbetalning', 'Övrigt')
Begäran om återbetalning – Order 12345	Återbetalning
Tekniskt problem ej löst	Övrigt
Tekniskt problem med produkt XYZ	Övrigt

Utdata från funktionen `Index` har korrekt identifierat alla ämnesrader som innehåller strängen `Refund`. Det här är ett exempel på hur du kan analysera och kategorisera data med hjälp av kapslade funktioner och ge användarna mervärde.

IndexRegEx

IndexRegEx() söker igenom indatasträngen och returnerar startpositionen för den n:te förekomsten av det angivna reguljära uttrycksmönstret. Ett valbart tredje argument **count** ger värdet för n, vilket annars är 1. Strängens positioner är numrerade vänster till höger från 1 och uppåt. Om ingen matchning hittas returnerar funktionen 0.

Den här funktionen utför regexoperationer som är skiftlägeskänsliga. Alternativt kan du använda varianten **IndexRegExI()** för att utföra skiftlägesokänsliga regexoperationer.

Syntax:

```
IndexRegEx (text, regex [, count])
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
text	Indatasträngens text inom vilken du vill söka efter ett reguljärt uttryck.
regex	Det reguljära uttryck som ska användas för att söka i indatasträngen.
count	Numret på matchningen. Detta är användbart när flera träffar för det reguljära uttrycket kan finnas i texten. Ange t.ex. värdet `4` för att få fram positionen för den fjärde matchningen. Det här är ett valfritt argument. Standard är 1 om det inte anges. Du kan ange ett negativt värde för att söka efter matchningar från höger till vänster.

Exempel på funktioner

Exempel	Resultat
`IndexRegEx('abc123','[a-z][0-9]+')`	Returnerar `3` (startpositionen för den första matchningen).

Exempel	Resultat
`IndexRegEx('abc123','[a-z][0-9]+',2)`	Returnerar `0` (regex har inte några andra matchning).
`IndexRegEx('ABC123','[a-z][0-9]+')`	Returnerar `0`, eftersom **IndexRegEx()** är skiftlägeskänsligt.
`IndexRegExI('ABC123','[a-z][0-9]+')`	Returnerar `3`. Den skiftlägesokänsliga varianten av funktionen, **IndexRegExI()**, används.

Användning

Användningsfall för denna funktion inkluderar:

- Identifiera var specifika textmönster förekommer inom större textmassor. Du kanske till exempel vill veta var ett mönster för e-postadresser används i en serie långa e-postmeddelanden.
- **IndexRegEx()** är särskilt användbart för avancerad databehandling och används ofta i det första steget i en längre och mer komplex datatransformering. Det är vanligtvis lättare att lösa problem med andra regexfunktioner som **ExtractRegEx()**, **MatchRegEx()** och **CountRegEx()**, men det kan finnas tillfällen då **IndexRegEx()** erbjuder lösningar som dessa funktioner inte kan tillhandahålla.

Exempel 1 – laddningsskript för att identifiera böcker med ISBN

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Tabellen `ISBN` innehåller e-postmeddelandeinnehåll i ett fält med namnet `EmailBody`.

Vi vill hitta positionerna för alla 13-siffriga ISBN-koder från det naturliga språket i e-postmeddelandet.

Laddningsskript

```
SET ISBN_RegEx = 'ISBN[ ]*([0-9]{3})-([0-9]{1})-([0-9]{4})-([0-9]{4})-([0-9]{1})';

ISBN:
LOAD IndexRegEx(EmailText,'$(ISBN_RegEx)',1) AS ISBNCode1,
IndexRegEx(EmailText,'$(ISBN_RegEx)',2) AS ISBNCode2,
IndexRegEx(EmailText,'$(ISBN_RegEx)',3) AS ISBNCode3,
* INLINE `
EmailText
Hi there! I have some books that I'm interested in ordering from your distribution center.
Just wanted to make sure they were in stock before I put the order in on the system - do you
think you could tell me whether you have these in stock, and how many you might be able to
ship us? Thanks! Item 1: ISBN 123-3-1234-1234-0. This is one I've had a really hard time
getting a hold of lately. Do you think you would have 5 in stock? If not, can I put a hold on
the next 5 you get? Item 2: ISBN 012-2-0123-0123-4 ... This one is not high-priority, but if
you have one, I'll take it! Item 3: ISBN 000-1-0123-0123-2. Customers have been requesting
```

```
this one for a long time, and we haven't had it in stock for years due to it being out of
print. Any chance you might have 7 of them? Thanks!
` (delimiter is '/t/';
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ISBNCode1`
- `ISBNCode2`
- `ISBNCode3`

Resultattabell

ISBNCode1	ISBNCode2	ISBNCode3
301	487	588

Exempel 2 – diagramuttryck för att identifiera kontakter med sekundära telefonnummer

Diagramuttryck och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell som heter `BusinessContactInfo`, som innehåller information om företag som hittades på respektive företags webbplats.
- `ContactInfo` är ett fält som innehåller företagets telefonnummer. Vi vill ta reda på vilka företag som har flera telefonnummer tillgängliga.

Våra krav:

- Telefonnumret måste vara ett NANP-nummer med 10 siffror.
- Vi vill att riktnumret ska kunna anges inom parentes.
- Vi vill tillåta ett enskilt blanksteg eller bindestreck mellan varje segment av ett telefonnummer.

Laddningsskript

```
BusinessContactInfo:

Load * Inline `
ID      CompanyName      ContactInfo
1       Company A        (123) 456-7890 (023) 123-4567
2       Company B        0123456790 1357913579 0246802468
3       Company C        234-567-8901

` (delimiter is '\t');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `CompanyName`

Lägg till denna beräknade dimension i tabellen:

```
=if(IndexRegEx(ContactInfo, '\({0,1}[0-9]{3}\){0,1}[ -]{0,1}[0-9]{3}[ -]{0,1}[0-9]{4}', 2)=0,
'No secondary phone number(s)', 'Has secondary phone number(s)')
```

Resultattabell

CompanyName	=if(IndexRegEx(ContactInfo, '\({0,1}[0-9]{3}\){0,1}[-]{0,1}[0-9]{3}[-]{0,1}[0-9]{4}', 2)=0, 'No secondary phone number(s)', 'Has secondary phone number(s)')
Company A	Has secondary phone number(s)
Company B	Has secondary phone number(s)
Company C	No secondary phone number(s)

`Company A` har ett sekundärt telefonnummer. `Company B` har två sekundära telefonnummer. `Company C` har tre sekundära telefonnummer.

*Funktionen **MatchRegEx()** är också användbar för den här typen av valideringsanvändningsfall.*

IndexRegExGroup

IndexRegExGroup() söker igenom indatasträngen och returnerar startpositionen för den n:te förekomsten av det sammansatta reguljära uttrycksmönster som anges. Ett valbart fjärde argument **count** ger värdet för n, vilket annars är 1. Strängens positioner är numrerade vänster till höger från 1 och uppåt. Om ingen matchning hittas returnerar funktionen 0.

Den här funktionen utför regexoperationer som är skiftlägeskänsliga. Alternativt kan du använda varianten **IndexRegExGroupI()** för att utföra skiftlägesokänsliga regexoperationer.

Syntax:

```
IndexRegExGroup (text, regex, group [, count])
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
text	Indatasträngens text inom vilken du vill söka efter ett reguljärt uttryck.

Argument	Beskrivning
regex	Det reguljära uttryck som ska användas för att söka i indatasträngen.
group	Gruppens nummer, om det rör sig om ett sammansatt reguljärt uttryck. Ett **group**-värde på `0` returnerar indexet för hela regex. Men om det reguljära uttrycket bara behöver returnera indexet för hela matchningen, använder du funktionen **IndexRegEx()** i stället. Du kan ange ett negativt värde för **group** för att söka efter matchningar från höger till vänster.
count	Numret på matchningen. Detta är användbart när flera träffar för det reguljära uttrycket kan finnas i texten. Ange t.ex. värdet `4` för att få fram positionen för den fjärde matchningen. Det här är ett valfritt argument. Standard är 1 om det inte anges. Du kan ange ett negativt värde för att söka efter matchningar från höger till vänster.

Exempel på funktioner

Exempel	Resultat
`IndexRegExGroup ('abc123','([a-z])([0-9]+)',0)`	Returnerar `3` (startposition för komplett regex).
`IndexRegExGroup ('abc123','([a-z])([0-9]+)',1)`	Returnerar `3` (startpositionen för den första gruppen).
`IndexRegExGroup ('abc123','([a-z])([0-9]+)',2)`	Returnerar `4`, som är startpositionen för den andra gruppen. I det inmatade regexmönstret motsvarar strängen `([0-9]+)` den andra gruppen.
`IndexRegExGroup ('ABC123','([a-z])([0-9]+)',1)`	Returnerar `0`, eftersom **IndexRegExGroup()** är skiftlägeskänsligt.
`IndexRegExGroupI ('ABC123','([a-z])([0-9]+)',1)`	Returnerar `3`. Den skiftlägesokänsliga varianten av funktionen, **IndexRegExI()**, används.

Användning

Användningsfall för denna funktion inkluderar:

- Identifiera var specifika textmönster förekommer inom större textmassor. Du kanske till exempel vill veta var domänen för en e-postadress används i en serie långa e-postmeddelanden.
- **IndexRegExGroup()** är särskilt användbart för avancerad databehandling och används ofta i det första steget i en längre och mer komplex datatransformering. Det är vanligtvis lättare att lösa problem med andra regexfunktioner som **ExtractRegExGroup()**, **MatchRegEx()** och **CountRegEx()**, men det kan finnas tillfällen då **IndexRegExGroup()** erbjuder lösningar som dessa funktioner inte kan tillhandahålla.

Exempel 1 – laddningsskript för att hitta positioner för ISBN-komponenter

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Skapandet av en variabel, `ISBN_RegEx`, för att lagra det reguljära uttryck som vi vill använda.
- En uppsättning ISBN-koder för böcker som en bokhandel vill beställa från flera leverantörer.

För varje ISBN-kod måste vi extrahera startpositionen för var och en av följande:

- EAN
- Grupp
- Registrant
- Publication
- Checksum

Laddningsskript

```
SET ISBN_RegEx = 'ISBN[ ]*([0-9]{3})-([0-9]{1})-([0-9]{4})-([0-9]{4})-([0-9]{1})';

ISBN:
LOAD Supplier,
IndexRegExGroup(Books,'$(ISBN_RegEx)',1) AS EAN,
IndexRegExGroup(Books,'$(ISBN_RegEx)',2) AS Group,
IndexRegExGroup(Books,'$(ISBN_RegEx)',3) AS Registrant,
IndexRegExGroup(Books,'$(ISBN_RegEx)',4) AS Publication,
IndexRegExGroup(Books,'$(ISBN_RegEx)',5) AS Checksum;
// Split the ISBN with the Group function in a preceding load to avoid generating a cartesian
product
LOAD *,
ExtractRegEx(SupplierBooks, '$(ISBN_RegEx)') AS Books
INLINE [
Supplier, SupplierBooks
Supplier 1, ISBN 123-3-1234-1234-0 ISBN 012-2-0123-0123-4 ISBN 000-1-0123-0123-2 ISBN 234-5-
2345-2345-1 ISBN 555-2-5555-5555-3 ISBN 222-4-2222-2222-2
Supplier 2, ISBN 000-0-3333-3333-3 ISBN 333-3-3333-3333-3 ISBN 555-1-5151-5151-3 ISBN 232-1-
2323-2323-1 ISBN 008-0-7777-7777-3 ISBN 888-0-9999-0000-0
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Supplier`
- `EAN`

- `Group`
- `Registrant`
- `Publication`
- `Checksum`

Resultattabell

Supplier	EAN	Group	Registrant	Publication	Checksum
Supplier 1	6	10	12	17	22
Supplier 1	6	10	12	17	22
Supplier 1	6	10	12	17	22
Supplier 1	6	10	12	17	22
Supplier 1	6	10	12	17	22
Supplier 1	6	10	12	17	22
Supplier 2	6	10	12	17	22
Supplier 2	6	10	12	17	22
Supplier 2	6	10	12	17	22
Supplier 2	6	10	12	17	22
Supplier 2	6	10	12	17	22
Supplier 2	6	10	12	17	22

I det här fallet innehåller varje bok komponenter i samma positioner.

Exempel 2 – diagramuttryck för att hitta URL-positioner (med jämförelse på IndexRegEx())

Diagramuttryck och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell med namnet `Correspondence`, som innehåller e-postmeddelandetexten i ett fält med namnet `EmailBody`.
- Innehåll i e-postmeddelanden som innehåller URL:er.

Vi vill hitta positionen för den andra URL:en, om sådan finns, samt positionerna för följande komponenter:

- Domän
- Sökväg

Vi vill använda ett enda reguljärt uttryck för att köra alla operationer.

Laddningsskript

```
Correspondence:

Load * Inline `
ID      EmailBody
1       Thanks again for this morning's meeting! You can find the meeting minutes posted here:
https://example.com/resourceexample. If you still have any questions, always feel free to ask
me or one of the other team members. Here are a few learning resources that might help you:
http://www.example.ca/training1.pptx http://www.example.ca/training2.pptx
http://www.example.ca/training3.pptx Thanks again!
2       Hi, you'll want to visit our company website for that, it's available at
https://www.example.se.
3       Hello all, I just wanted to let you know that our online stores are now up and running! I
couldn't be more excited. We are already seeing quite a bit of traffic and volume sold, which
is very promising! For Product A, go to https://www.examplestore1.com/products. For Product B,
you'll want go to https://www.examplestore2.com/products. Product C, go check out
https://www.examplestore3.com/products. Cheers!

` (delimiter is '\t');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `ID`

Lägg till dess beräknade dimensioner:

- *URL 2-position*:
  `=IndexRegExGroup(EmailBody,'([a-zA-Z0-9]+):\/\/(([a-zA-Z0-9]+\.)*([a-zA-Z0-9]+)\.([a-zA-Z0-9]+))((\/[a-zA-Z0-9]+)*(\/([a-zA-Z0-9]+)(\.[a-zA-Z0-9]+)?))?',2)`
- *URL 2 Domänposition*:
  `=IndexRegExGroup(EmailBody,'([a-zA-Z0-9]+):\/\/(([a-zA-Z0-9]+\.)*([a-zA-Z0-9]+)\.([a-zA-Z0-9]+))((\/[a-zA-Z0-9]+)*(\/([a-zA-Z0-9]+)(\.[a-zA-Z0-9]+)?))?',2,2)`
- *URL 2 Sökvägsposition*:
  `=IndexRegExGroup(EmailBody,'([a-zA-Z0-9]+):\/\/(([a-zA-Z0-9]+\.)*([a-zA-Z0-9]+)\.([a-zA-Z0-9]+))((\/[a-zA-Z0-9]+)*(\/([a-zA-Z0-9]+)(\.[a-zA-Z0-9]+)?))?',6,2)`

Resultattabell

ID	URL 2-position	URL 2 Domänposition	URL 2 Sökvägsposition
1	274	281	295
2	0	0	0
3	298	306	327

IndexRegEx() returnerar startpositionen för hela URL:en, medan **IndexRegExGroup()** returnerar startpositionerna för enskilda delar av URL:en, vilket motsvarar det **group**-värde vi använde. Posten med värdet 2 på `ID` innehåller bara en URL, så värden med 0 returneras.

Samma reguljära uttryck används i alla diagramuttryck för att returnera olika typer av information. En uppdelning av de grupper som definieras i det reguljära uttrycket följer.

Grupper i reguljärt uttryck

Group number	Regex	URL part
1	([a-zA-Z0-9]+)	Protokoll
2	(([a-zA-Z0-9]+\.)*([a-zA-Z0-9]+)\.([a-zA-Z0-9]+))	Domain (grupp 3, 4 och 5)
3	([a-zA-Z0-9]+\.)*	Root domain
4	([a-zA-Z0-9]+)	Top-level domain
5	([a-zA-Z0-9]+)	Path (grupp 7, 8, 9 och 10) (valfritt)
6	((\/[a-zA-Z0-9]+)*(\/([a-zA-Z0-9]+)(\.[a-zA-Z0-9]+)?))?	Sub-directory
7	(\/[a-zA-Z0-9]+)*	File (grupp 9 och 10)
8	(\/([a-zA-Z0-9]+)(\.[a-zA-Z0-9]+)?)	Fil
9	([a-zA-Z0-9]+)	Filnamn
10	(\.[a-zA-Z0-9]+)?	Komplement till fil (valfritt)

IsJson

IsJson() testar om en angiven sträng innehåller giltiga JSON-data (JavaScript Object Notation). Du kan även validera en specifik JSON-datatyp.

Syntax:

```
value IsJson(json [, type])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
json	Sträng att testa. Det kan innehålla extra mellanslag eller nyrader.

Argument	Beskrivning
type	Valfritt argument som anger vilken JSON-datatyp som ska testas. • 'value' (standard) • 'objekt' • 'object' • 'sträng' • 'number' • "Boolean" • 'null'

Exempel: giltiga och ogiltiga diagramuttryck

Exempel	Resultat
`IsJson( 'null' )`	Returnerar `-1 (true)`
`IsJson( '"abc"', 'value' )`	Returnerar `-1 (true)`
`IsJson( '"abc"', 'string' )`	Returnerar `-1 (true)`
`IsJson( 123, 'number' )`	Returnerar `-1 (true)`
`IsJson( 'text' )`	Returnerar `0 (false)`, `'text'` är inte ett giltigt JSON-värde
`IsJson( '"text"', 'number' )`	Returnerar `0 (false)`, `'"text"'` är inte ett giltigt JSON-tal
`IsJson( '"text"', 'text' )`	Returnerar `0 (false)`, `'text'` är inte en giltig JSON-typ

Exempel – grunderna i IsJson

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `API_Response`.
- Ett härlett fält som heter `ID`. I detta fält används funktionen `Rec()` för att räkna upp de ingående posterna.

Laddningsskript

```
Example:
Load
Recno() AS ID, API_Response
inline [
API_Response
```

```
'{"id": 1, "name": "Alice"}'
'{"id": 2, "name": "Bob"}'
'{invalid json string}'
'{"id": 4, "name": "Charlie"}'
'{"id": 5, name: "David"}'
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ID`
- `API_Response`

Skapa följande beräknade dimension:

- `=IsJson(API_Response)`, för att beräkna om värdena i `API_Response` är giltiga JSON.

Resultattabell

ID	API_Response	IsJson(API_Response)
1	{"id": 1, "name": "Alice"}	-1 (true)
2	{"id": 2, "name": "Bob"}	-1 (true)
3	{invalid json string}	0 (false)
4	{"id": 4, "name": "Charlie"}	-1 (true)
5	{"id": 5, name: "David"}	0 (false)

Utmatningen returnerar `-1`, eller true, för de värden som har giltig JSON-syntax.

Utmatningen returnerar `0`, eller false, för följande ID-poster (`3` och `5`) med ogiltig JSON:

- `{invalid json string}` - Textsträngen representerar inte en giltig JSON-svarsstruktur.
- `{"id": 5, name: "David"}` - Det andra nyckel-värdeparet saknar citattecken runt nyckeln `"name"`.

Exempel – IsJson-scenario

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `OrderDetails`.

Laddningsskript

```
Example:
Load *
INLINE [OrderDetails
'{ "order_id": "12345", "customer": { "name": "John Doe", "email": "john.doe@example.com"},
"items": {"product": "Laptop", "quantity": 2, "price": 1200 }, "total_price": 2400 }'
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `OrderDetails`

Skapa följande mått:

- `IsJson( OrderDetails)`, för att beräkna om värdena i `OrderDetails` är giltiga JSON.
- `IsJson( JsonGet ( OrderDetails, '/items/price' ), 'number' )` med hjälp av funktionen `JsonGet` hämtas JSON-texten för nyckeln `price` och det kontrolleras att värdet för `price` är ett tal.

Resultattabell

OrderDetails	IsJson (OrderDetails)	IsJson(JsonGet (OrderDetails, '/items/price'), 'number')
{ "order_id": "12345", "customer": { "name": "John Doe", "email": "john.doe@example.com"}, "items": {"product": "Laptop", "quantity": 2, "price": 1200 }, "total_price": 2400 }	-1 (true)	-1 (true)

Den första åtgärden returnerar `-1 (true)` eftersom `OrderDetails` innehåller giltig JSON-syntax.

Den andra åtgärden returnerar `-1 (true)` eftersom värdet på `price`-nyckeln är ett giltigt tal, `1200`.

IsRegEx

IsRegEx() returnerar huruvida den angivna texten är ett giltigt reguljärt uttryck.

Den här funktionen utför regexoperationer som är skiftlägeskänsliga. Alternativt kan du använda varianten **IsRegExI()** för att utföra skiftlägesokänsliga regexoperationer.

```
IsRegEx (expr [, debug])
```

Returnerad datatyp: Dual

Argument

Argument	Beskrivning
expr	Stränguttryck som innehåller det reguljära uttryck som ska utvärderas.
debug	Valfritt argument. Om något värde anges för detta argument returnerar uttrycket en texthint som beskriver hur ett ogiltigt reguljärt uttryck kan åtgärdas.

Exempel på funktioner

Exempel	Resultat
`IsRegEx('[a-z]')`	Returnerar `-1` (sant).
`IsRegEx(']0-1[', 1)`	Detta exempel returnerar ett felmeddelande med en ledtråd för att korrigera det ogiltiga reguljära uttrycket `]0-1[`.
`IsRegEx('[a-z')`	Returnerar `0` (falskt). Inget felmeddelande visas eftersom argumentet **debug** inte har angetts.

Användning

Du kan använda **IsRegEx()** för att testa om de reguljära uttryck du skriver är giltiga i Qlik Sense. Du kan till exempel läsa in en tabell i din applikation med indatatext och en utvärdering av textens giltighet som regex.

Exempel 1 – laddar giltighet för regex i datamodell

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Ett fält, `reg_exp`, som innehåller indatasträngar som kanske eller kanske inte är giltiga reguljära uttryck.
- Två beräknade fält för att kontrollera om respektive värde i `reg_exp` är ett giltigt reguljärt uttryck eller inte:
  - `validity_no_hint`: Visar `-1` (sant) om regex är giltig och 0 (falskt) om den är ogiltig.
  - `validity_with_hint`: Visar -1 (sant) om regex är giltig. Om regexet är ogiltigt visas ett autogenererat felmeddelande som beskriver problem med inmatningen, med möjliga tips för att åtgärda dem.

Laddningsskript

```
[Expression Validity]:
       Load
               *,
               IsRegEx(reg_exp) as validity_no_hint,
               IsRegEx(reg_exp,1) as validity_with_hint
               ;
Load
*
Inline
`
reg_exp
[a-z]+
([a-z)(0-9)+
abcdefg|[0-7
\$|¢(0-9)
`;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- reg_exp
- validity_no_hint
- validity_with_hint

Resultattabell

reg_exp	validity_no_hint	validity_with_hint
([a-z)(0-9)+	0	-1
[a-z]+	-1	Tips-text (autogenererad)
\$|¢(0-9)	-1	Tips-text (autogenererad)
abcdefg|[0-7	0	-1

Exempel 2 – laddar endast giltig regex i datamodell

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Ett fält, reg_exp, som innehåller indatasträngar som kanske eller kanske inte är giltiga reguljära uttryck. Detta fält finns i en tabell som heter Regular Expressions.

- En **where**-sats som bara laddar poster i datamodellen när indata är ett giltigt reguljärt uttryck.

Laddningsskript

```
[Regular Expressions]:

Load
*
Inline
`
reg_exp
[a-z]+
([a-z)(0-9)+
abcdefg|[0-7
\$|¢(0-9)
`
WHERE
IsRegEx(reg_exp)=True()
;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `reg_exp`

Resultattabell

reg_exp
[a-z]+
\$|¢(0-9)

`[a-z]+` och `\$|¢(0-9)` är giltiga reguljära uttryck, så de har laddats in i datamodellen. `([a-z)(0-9)+` och `abcdefg|[0-7` är inte giltiga reguljära uttryck, så de har inte laddats in i datamodellen.

Exempel 3 – diagramuttryck

Diagramuttryck och resultat

Översikt

I det här exemplet läser vi in en uppsättning indatasträngar i datamodellen och lägger till dem i en tabell, tillsammans med två beräknade dimensioner som utvärderar regexens giltighet. De två beräknade dimensionerna förmedlar samma information på olika sätt.

Öppna Skriptredigeraren och lägg till följande laddningsskript till ett nytt delavsnitt.

Laddningsskriptet innehåller ett fält, `reg_exp`, som innehåller indatasträngar som kanske eller kanske inte är giltiga reguljära uttryck. Detta fält finns i en tabell som heter `Regular Expressions`.

Laddningsskript

```
[Regular Expressions]:

Load
*
Inline
`
reg_exp
[a-z]+
([a-z)(0-9)+
abcdefg|[0-7
\$|¢(0-9)
`;
```

Resultat

Gör följande:

1. Ladda data och öppna ett ark. Skapa en ny tabell. Lägg till `reg_exp` som en dimension .
2. Lägg nu till följande beräknade dimensioner:
   `=IsRegEx(reg_exp)`
   Detta är ett grundläggande diagramuttryck som visar `-1` (sant) om indata är ett giltigt reguljärt uttryck och `0` (falskt) om det inte är det.
3. Lägg till ytterligare en beräknad dimension:
   `=if(IsRegEx(reg_exp), 'Valid regex', 'Invalid regex')`
   Detta är ett villkorligt uttryck som ger mer information om giltigheten av indata.

Resultattabell

reg_exp	=IsRegEx(reg_exp)	=if(IsRegEx(reg_exp), 'Valid regex', 'Invalid regex')
([a-z)(0-9)+	0	Invalid regex
[a-z]+	-1	Valid regex
\$|¢(0-9)	-1	Valid regex
abcdefg|[0-7	0	Invalid regex

JsonArray

JsonArray aggregerar och konkatenerar JSON-data.

Syntax:

```
JsonArray([DISTINCT][TOTAL] value [, sort_weight])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras.
value	Indatafält. Uttrycket som innehåller data som ska sorteras. Det första (lägsta) värdet i **sort_weight** hittas, varifrån motsvarande värde i uttrycket **value** fastställs. Om du sätter ett minustecken framför **sort_weight** returnerar funktionen det sista (högsta) sorterade värdet i stället.
sort_weight	Indatafält. Uttrycket som innehåller data som ska sorteras. Det första (lägsta) värdet i **sort_weight** hittas, varifrån motsvarande värde i uttrycket **value** fastställs. Om du sätter ett minustecken framför **sort_weight** returnerar funktionen det sista (högsta) sorterade värdet i stället.

Exempel:

Följande laddningsskript laddar och aggregerar data med **JsonArray**.

```
Data:
LOAD Id,
       JsonArray(ValueOrNull) AS Json,
       JsonArray(DISTINCT ValueOrNull) AS JsonDistinct,
       JsonArray(ValueOrNull, Order) AS JsonSorted
GROUP BY Id;

LOAD *, If(Value='-',Null(),Value) AS ValueOrNull;

LOAD * INLINE '
       Id | Value          | Order
       1  | 123            | 1
       2  | "json text"    | 2
       2  | normal text    | 1
       3  | [1,2,3]        | 3
       3  | {"name":"abc"} | 2
       3  | [1,2,3]        | 1
       4  | 1              | 1
       4  | 2 and text     | 2
       4  |                | 3
       4  | -              | 4
       ' (delimiter is '|');
```

Detta resulterar i följande datatabell:

Resultat

Id	Json	JsonDistinct	JsonSorted
1	[123]	[123]	[123]
2	["json text","normal text"]	["json text","normal text"]	["normal text","json text"]
3	[[1,2,3],[1,2,3],{"name":"abc"}]	[[1,2,3],{"name":"abc"}]	[[1,2,3],{"name":"abc"},[1,2,3]]
4	[1,"2 and text",""]	[1,"2 and text",""]	[1,"2 and text",""]

JsonGet

JsonGet() returnerar sökvägen till en JSON-datasträng (JavaScript Object Notation). JSON-data måste vara giltiga, men kan innehålla extra mellanslag eller nya rader.

Syntax:

```
value JsonGet(json, path)
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
json	Sträng som innehåller JSON-data.
path	Sökvägen måste specificeras enligt ≤ RFC 6901. Det kommer att tillåta sökning av egenskaper inuti JSON-data utan att använda komplexa delsträngar eller indexfunktioner.

Exempel: giltiga och ogiltiga diagramuttryck

Exempel	Resultat
`JsonGet( '{"a":{"xyz":"XYZ"},"b":[123,"abc","ABC"]}', '' )`	Returnerar `{"a":{"xyz":"XYZ"},"b":[123,"abc","ABC"]}`
`JsonGet( '{"a":{"xyz":"XYZ"},"b":[123,"abc","ABC"]}', '/a' )`	Returnerar `{"xyz":"XYZ"}`
`JsonGet( '{"a":{"xyz":"XYZ"},"b":[123,"abc","ABC"]}', '/a/xyz' )`	Returnerar `"XYZ"`
`JsonGet( '{"a":{"xyz":"XYZ"},"b":[123,"abc","ABC"]}', '/b' )`	Returnerar `[123,"abc","ABC"]`
`JsonGet( '{"a":{"xyz":"XYZ"},"b":[123,"abc","ABC"]}', '/b/0' )`	Returnerar `123`
`JsonGet( '{"a":{"xyz":"XYZ"},"b":[123,"abc","ABC"]}', '/b/1' )`	Returnerar `abc`
`JsonGet( '{"a":{"xyz":"XYZ"},"b":[123,"abc","ABC"]}', '/b/2' )`	Returnerar `ABC`

Exempel	Resultat
`JsonGet( '{"a":"b"}','/b' )`	Returnerar `null`. Sökvägen pekar inte till en giltig del av JSON-datan.
`JsonGet( '{"a"}','/a' )`	Returnerar `null`. JSON-data är inte giltig JSON (medlemmen `"a"` har inget värde).

Exempel – grunderna i JsonGet

Diagramuttryck

Översikt

Du får JSON-data med orderinformation som en textsträng på en enda rad. Du vill hämta följande information:

- varans pris och validering av att värdet är ett tal
- kundens e-postadress
- totalpris (pris * antal)

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `OrderDetails`.

Laddningsskript

```
Example:
Load  *
INLINE [OrderDetails
'{ "order_id": "12345", "customer": { "name": "John Doe", "email": "john.doe@example.com"},
"items": {"product": "Laptop", "quantity": 2, "price": 1200 }, "total_price": 2400 }'
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `OrderDetails`

Skapa följande mått:

- `IsJson( JsonGet ( OrderDetails, '/items/price' ), 'number' )` tillsammans med `IsJson`-funktionen hämtar JSON-texten för `price` och bekräftar om värdet för `price` är ett tal.
- `JsonGet( OrderDetails, '/customer/email' )`, för att hämta kundens e-postadress med hjälp av rätt sökväg.

- `JsonGet( OrderDetails, '/total_price' ), )`, för att hämta det totala priset med hjälp av rätt sökväg.

Resultattabell

OrderDetails	IsJson(JsonGet (OrderDetails, '/items/price'), 'number')	JsonGet(OrderDetails, '/customer/email')	JsonGet(OrderDetails, '/total_price'),)
{ "order_id": "12345", "customer": { "name": "John Doe", "email": "john.doe@example.com"}, "items": {"product": "Laptop", "quantity": 2, "price": 1200 }, "total_price": 2400 }	-1 (true)	john.doe@example.com	2400

Den första åtgärden returnerar `-1 true` eftersom nyckeln `price` har värdet `1200`, som är ett tal.

Den andra åtgärden returnerar kundens e-postadress, `john.doe@example.com`, från `OrderDetails` med hjälp av rätt sökväg till `email`-nyckeln i JSON-strängen.

Den tredje åtgärden returnerar `2400`, som är det totala priset för ordern från fältet `OrderDetails`, med hjälp av rätt sökväg till nyckeln `total_price` i JSON-strängen.

JsonObject

JsonObject används för att bygga JSON-objekt.

Du kan lägga till valfritt antal namn-värde-par med JsonObject.

Syntax:

```
JsonObject(name, value [name, value, ...])
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
json	Sträng som innehåller JSON-data.
namn	Namnet på fältet i JSON-format.
värde	Strängvärdet i JSON-format.

Exempel:

Följande laddningsskript laddar och ställer in data som ett JSON-objekt.

```
Data:
Load *,
       JsonObject('id', Id,
                'name', Name,
                'address', Address,
                'phone', Phone,
                'fax', FaxOrNull) AS Json;

LOAD *, If(Fax='-',Null(),Fax) AS FaxOrNull;

LOAD * INLINE [
       Id, Name,       Address,     Phone,     Fax
       1,  John Doe,   Oak Way,   1 234 567, 1 234 568
       2,  Jane Doe, Maple Way, 123456,    -
       3,  Mr Xavier,       Spruce Way,   1-800-MRX
];
```

Detta resulterar i följande JSON-data:

```
{"id":1,"name":"John Doe","address":"Oak Way","phone":"1 234 567","fax":"1 234 568"}
{"id":2,"name":"Jane Doe","address":"Maple Way","phone":123456}
{"id":3,"name":"Mr Xavier","address":"Spruce Way","phone":"1-800-MRX","fax":""}
```

JsonSet

JsonSet() modifierar en sträng som innehåller JSON-data (JavaScript Object Notation). Den kan ställa in eller infoga ett JSON-värde med den nya platsen som anges av sökvägen. JSON-data måste vara giltiga, men kan innehålla extra mellanslag eller nya rader.

Du kan lägga till valfritt antal namn-värde-par med JsonSet.

Syntax:

```
value JsonSet(json, path, value)
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
json	Sträng som innehåller JSON-data.
path	Sökvägen måste specificeras enligt ≤ RFC 6901. Det tillåter uppbyggnad av egenskaper inuti JSON-data utan att använda komplexa delsträngar eller indexfunktioner och konkatenering.
value	Det nya strängvärdet i JSON-format.

Exempel: giltiga och ogiltiga diagramuttryck

Exempel	Resultat
`JsonSet( '{}','/a','"b"' )`	Returnerar `{"a":"b"}`
`JsonSet( '[]','/0','"x"' )`	Returnerar `["x"]`
`JsonSet( '"abc"','','123' )`	Returnerar `123`
`JsonSet( '"abc"','/x','123' )`	Returnerar `null`. Sökvägen pekar inte till en giltig del av JSON-datan.
`JsonSet( '{"a": {"b":"c"}}','a/b','"x"' )`	Returnerar `null`. Sökvägen är ogiltig.
`JsonSet( '{"a":"b"}','/a','abc' )`	Returnerar `null`. Värdet är inte en giltig JSON. En sträng måste vara inom citattecken.

Exempel – grunderna i JsonSet

Diagramuttryck

Översikt

JSON-data fås som en textsträng på en enda rad. Du vill uppdatera priset på artikeln och det totala priset för ordern i enlighet med detta.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `OrderDetails`.

Laddningsskript

```
Example:
Load  *
INLINE [OrderDetails
'{ "order_id": "12345", "customer": { "name": "John Doe", "email": "john.doe@example.com"},
"items": {"product": "Laptop", "quantity": 2, "price": 1200 }, "total_price": 2400 }'
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `OrderDetails`

Skapa följande mått:

- `JsonSet(JsonSet(OrderDetails,'/items/price',1100),'/total_price',2200)`, för att ställa in priset till `1100` och det totala priset till `2200` med hjälp av rätt sökväg till respektive nyckel i JSON-strängen.

Resultattabell

OrderDetails	JsonSet(JsonSet (OrderDetails,'/items/price',1100),'/total_ price',2200)
{ "order_id": "12345", "customer": { "name": "John Doe", "email": "john.doe@example.com"}, "items": {"product": "Laptop", "quantity": 2, "price": 1200 }, "total_price": 2400 }	{"order_id":"12345","customer":{"name":"John Doe","email":"john.doe@example.com"},"items": {"product":"Laptop","quantity":2,"price":1100},"total_ price":2200}

I slutet av åtgärden uppdateras nycklarna `price` och `total price` till `1100` respektive `2200`.

JsonSetEx

JsonSetEx() modifierar strängar som innehåller JSON-data (JavaScript Object Notation). Den kan ställa in eller infoga JSON-värden med den nya platsen som anges av sökvägen. JSON-data måste vara giltiga, men kan innehålla extra mellanslag eller nya rader.

Du kan lägga till valfritt antal namn-värde-par med JsonSetEx.

Syntax:

```
JsonSetEx(json, path, value, ...)
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
json	Sträng som innehåller JSON-data.
path	Sökvägen måste specificeras enligt ≤ RFC 6901. Det tillåter uppbyggnad av egenskaper inuti JSON-data utan att använda komplexa delsträngar eller indexfunktioner och konkatenering.
value	Det nya strängvärdet i JSON-format.

Exempel:

Följande laddningsskript laddar och formaterar data till JSON-format.

```
Data:
Load *,
       JsonSetEx('{"fax":123}',
               '/name', Name,
               '/address', Address,
               '/phone', Phone,
               '/fax', FaxOrNull) AS Json;

LOAD *, If(Fax='-',Null(),Fax) AS FaxOrNull;

LOAD * INLINE [
       Id, Name,       Address,     Phone,     Fax
       1,  John Doe,   Oak Way,   1 234 567, 1 234 568
       2,  Jane Doe, Maple Way, 123456,    -
       3,  Mr Xavier,       Spruce Way,   1-800-MRX
];
```

Detta resulterar i följande JSON-data:

```
{"name":"John Doe","address":"Oak Way","phone":"1 243 567","fax":"1 234 568"}
{"name":"Jane Doe","address":"Maple Way","phone":123456}
{"fax":"","name":"Mr Xavier","address":"Spruce Way","phone":"1-800-MRX"}
```

KeepChar

KeepChar() returnerar en sträng som består av de tecken från den första strängen som matchar tecknen i den andra strängen. Denna funktion är skiftlägeskänslig.

Syntax:

```
KeepChar(text, keep_chars)
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
keep_chars	En sträng som innehåller tecknen i **text** som ska behållas. Detta argument är skiftlägeskänsligt.

Exempel: diagramuttryck

Exempel	Resultat
`KeepChar ( 'a1b2c3','123' )`	Returnerar `123`
`KeepChar ( 'a1b2c3','1234' )`	Returnerar `123`
`KeepChar ( 'a1b22c3','1234' )`	Returnerar `1223`
`KeepChar( 'a1b2c3','312' )`	Returnerar `123`

Exempel – grunderna i KeepChar

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `InputText` som innehåller den ursprungliga textsträngen som ska bearbetas.
  - `,CharsToKeep` som innehåller en lista över tecken som ska behållas från den ursprungliga textsträngen efter bearbetning.

Laddningsskript

```
Example:
Load * inline [
InputText, CharsToKeep
'a1b2c3', '123'
'a^b^c', 'abc'
'A^b^c', 'abc'
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `InputText`
- `CharsToKeep`

Skapa följande beräknade dimension:

- `=KeepChar(InputText, CharsToKeep)`, för att beräkna de tecken i `InputText` som matchar tecknen i `CharsToKeep`.

Resultattabell

InputText	CharsToKeep	KeepChar(InputText, CharsToKeep)
a1b2c3	123	123
a^b^c	abc	abc
A^b^c	abc	bc

Kolumnen `KeepChar(InputText, CharsToKeep)` returnerar endast de tecken i `InputText` som exakt matchar tecknen i `CharsToKeep`. Till exempel returnerar den tredje raden `bc` eftersom det är de enda tecknen i `InputText` som matchar tecknen i `CharsToKeep`. Det versala tecknet `A` returneras inte eftersom posterna i `CharsToKeep` endast är gemena.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Example:
Load
*,
keepchar(String1, String2) as KeepChar;
Load * inline [
String1, String2
'a1b2c3', '123'
];
```

Resultattabell

sträng(1)	Sträng2	KeepChar
a1b2c3	123	123

Exempel – KeepChar-scenario för att analysera ostrukturerad text

Diagramuttryck

Översikt

I det här exemplet används funktionen för att analysera ostrukturerad text (bokstäver, siffror och symboltecken) och leverera strukturerad utdata som endast innehåller numeriska tecken.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `InputText` som innehåller den ursprungliga textsträngen som ska bearbetas.
  - `,CharsToKeep` som innehåller en lista över tecken som ska behållas från den ursprungliga textsträngen efter bearbetning.

Laddningsskript

```
Example:
Load * inline [
InputText, InputCategory, CharsToKeep
'The numeric code for input processing is 123-456|789.', 'Code', '0123456789'
'Delivery of batch number: 333*456789.', 'Batch', '0123456789'
'Find the goods received in bay 16a', 'Bay', '0123456789'
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `InputText`
- `CharsToKeep`

Skapa följande beräknade dimension:

- `=KeepChar(InputText, CharsToKeep)`, för att beräkna de tecken i `InputText` som matchar tecknen i `CharsToKeep`.

Resultattabell

InputText	CharsToKeep	KeepChar(InputText, CharsToKeep)
Leverans av batchnummer: 333*456789	0123456789	333456789
Hitta de mottagna varorna i vik 16a	0123456789	16
Den numeriska koden för bearbetning av inmatning är 123-456|789	0123456789	123456789

Jämför utdata från funktionen `KeepChar` med de ursprungliga `InputText`-strängvärdena som laddades i skriptet. Funktionen `KeepChar` har tagit bort alla icke-numeriska tecken från inmatningstexten.

Se även:

p *PurgeChar (page 1810)*

Left

Left() returnerar en sträng som består av de första (längst till vänster) tecknen i indatasträngen, där antalet tecken fastställs av det andra argumentet.

Syntax:

```
Left(text, count)
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
count	Definierar antal tecken som ska inkluderas från vänster sida av strängen **text**.

Exempel: diagramuttryck

Exempel	Resultat
`Left( 'abcdef', 3 )`	Returnerar `abc`

Exempel – grunderna Left

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `InputText`
  - `NoOfChars`

Laddningsskript

```
Example:
Load * inline [
InputText, NoOfChars
'abcdef', 3
'2021-07-14', 4
'2021-07-14', 2
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `InputText`
- `NoOfChars`

Skapa följande beräknade dimension:

- `=Left(InputText,NoOfChars)`

Resultattabell

InputText	AntalTecken	Vänster(IndataText,AntalTecken)
abcdef	3	abc
2021-07-14	2	20
2021-07-14	4	2021

Den första raden returnerar `abc`, eftersom detta är de 3 första tecknen från vänster i strängen `InputText`.

Den andra raden returnerar `20`, eftersom detta är de 2 första tecknen från vänster i strängen `InputText`.

Den tredje raden returnerar `2021`, eftersom detta är de 4 första tecknen från vänster i strängen `InputText`.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Example:
Load
*,
left(Text,NoOfChars) as Left;
Load * inline [
Text, NoOfChars
'abcdef', 3
'2021-07-14', 4
'2021-07-14', 2
];
```

Resultattabell

Text	AntalTecken	Vänster
abcdef	3	abc
2021-07-14	4	2021
2021-07-14	2	20

Exempel – Left-scenario

Diagramuttryck

Översikt

I det här exemplet analyseras orderkoderna i en datauppsättning, som innehåller en regionkod, år och ordernummer, för att skapa ett nytt fält för dataanalys.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `OrderID`
  - `CustomerName`
  - `SalesAmount`

Laddningsskript

```
Example:
Load * inline [
OrderID, CustomerName, SalesAmount
US2024001, John Doe, 1500
EU2024002, Jane Smith, 2300
AS2024003, Mike Davis, 1800
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `OrderID`
- `CustomerName`
- `SalesAmount`

Skapa följande beräknade dimension:

- `=Left(OrderID, 2)` för att beräkna regionen.

Resultattabell

Order-ID	Kundnamn	Försäljningsbelopp	Vänster (OrderID, 2)
AS2024003	Mike Davis	1800	AS
EU2024002	Jane Smith	2300	EU
US2024001	John Doe	1500	USA

Utdata från `Left`-funktionen analyserar OrderID för att extrahera regionen och skapa ett nytt fält som kan användas för ytterligare dataanalys.

Exempel – vänster avancerat scenario

Diagramuttryck

Översikt

I det här exemplet analyseras produktkoderna i en datauppsättning, som innehåller en avdelningskategori och en unik identifierare, för att skapa ett nytt fält för dataanalys.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `ProductCode`.

Laddningsskript

```
Example:
Load * inline [
ProductCode
ELC12345
HOM67890
FAS09876
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `ProductCode`

Skapa följande beräknade dimension:

- `=Pick(Match(left(ProductCode,3),'ELC','HOM','FAS'),'Electronics','Home Appliances','Fashion')`, för att beräkna avdelningen.

Resultattabell

Produktkod	Välj(Matcha(vänster (Produktkod,3),'ELC','HOM','FAS'),'Elektronik','Hushållsapparater','Mode')
ELC12345	Elektronik
FAS09876	Mode
HOM67890	Hushållsapparater

I utmatningen används funktionerna `Pick`, `Match` och `Left` för att extrahera avdelningen från produktkoden och skapa ett nytt fält för dataanalys.
p Se även *Index (page 1750)*, som ger möjlighet till en mer komplex stränganalys.

Len

Len() returnerar längden för indatasträngen.

Syntax:

```
Len(text)
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
text	Sträng som ska utvärderas.

Exempel: diagramuttryck

Exempel	Resultat
`Len('Peter')`	Returnerar `5`

Exempel – grunderna i Len

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `CustomerComment`. Fältet innehåller den ursprungliga textsträngen som ska utvärderas.

Laddningsskript

```
Example:
Load  * inline [
CustomerComment
Please deliver after 5 PM.
Thank you for the quick service!
Can you add a gift wrap?
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `CustomerComment`

Skapa följande mått:

- `=Len(CustomerComment)`

Resultattabell

Kundkommentar	Len(CustomerComment)
Kan ni lägga till en presentförpackning?	24
Leverera efter kl. 17.00.	26
Tack för den snabba servicen!	32

Utdata från åtgärden `Len(CustomerComment)` returnerar längden på inmatningssträngen `CustomerComment`.

Exempel – Len-scenario

Diagramuttryck

Översikt

Det här exemplet analyserar kundfeedback genom att titta på längden på varje kommentar som en indikator på dess djup eller detaljnivå.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `TicketID`
  - `Description`

Laddningsskript

```
Example:
Load * inline [
TicketID, Description
1001, "I received the wrong product."
1002, "The package arrived damaged, and I would like a replacement."
1003, "I've been trying to reset my password for two days, and I haven't received an email."
1004, "My order status shows 'Delivered' but I have not received my package yet."
1005, "Is product XYZ available in size Large?"
1006, "I need help with an exchange for a faulty product, and I've attached photos."
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `TicketID`
- `Description`

Skapa följande beräknade dimension:

- `=If(Len(Description) < 30, 'Short',If(Len(Description) <= 50, 'Medium', 'Long'))` för att beräkna och tilldela en kategori (Short, Medium, Long) till ärendet baserat på beskrivningens längd.

Resultattabell

TicketID	Beskrivning	If(Len(Description) < 30, 'Kort',If(Len (Description) <= 50, 'Medel', 'Lång'))
1001	Jag fick fel produkt.	Kort

TicketID	Beskrivning	If(Len(Description) < 30, 'Kort',If(Len (Description) <= 50, 'Medel', 'Lång'))
1002	Paketet anlände skadat och jag vill ha ett utbyte.	Lång
1003	Jag har försökt återställa mitt lösenord i två dagar, men jag har inte fått något e-postmeddelande.	Lång
1004	Finns produkt XYZ tillgänglig i storlek Large?	Medel
1005	Min orderstatus visar "Delivered" men jag har inte fått mitt paket än.	Long
1006	Jag behöver hjälp med ett byte av en felaktig produkt och jag har bifogat foton.	Long

Utdata från den beräknade dimensionen visar hur du kan använda funktionen `Len` för att kategorisera data genom att tolka längden på textsträngar.

Exempel – Len-scenario med strängmanipulation

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `InputText`.

Laddningsskript

```
Example:
Load  * inline [
InputText
this is a sample text string
capitalize first letter only
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `InputText`

Skapa följande beräknade dimensioner:

- `=Upper(Left(InputText,1))` , för att konvertera den första bokstaven i textsträngen till versaler.
- `=Mid(InputText,Len(upper(Left(InputText,1)))+1)`, för att ta bort den första bokstaven från textsträngen.
- `=Upper(left(InputText,1)) & Mid(InputText,len(upper(left(InputText,1)))+1)`, för att kombinera resultatet från den första beräknade dimensionen med resultatet från den andra beräknade dimensionen.

Resultattabell

InputText	VERSALER (VÄNSTER (Indatatext,1))	EXTEXT (Indatatext,LÄNGD (VERSALER (VÄNSTER (Indatatext,1)))+1)	VERSALER(VÄNSTER(Indatatext,1)) & EXTEXT(Indatatext,LÄNGD (VERSALER(VÄNSTER (Indatatext,1)))+1)
detta är ett exempel på en textsträng	T	hans är en exempeltextsträng	detta är ett exempel på en textsträng
versalisera endast första bokstaven	C	ersalisera endast första bokstaven	Versalisera endast första bokstaven

I den första beräknade dimensionen kombineras funktionerna `Upper` och `Left` för att returnera den första bokstaven i `InputText` som versal. I den andra beräknade dimensionen använder `Mid`-funktionen `Len`-funktionen för att returnera en textsträng som tar bort det första tecknet från `InputText`. Den tredje beräknade dimensionen kombinerar den första och den andra beräknade dimensionen och returnerar `InputText`-strängen med ett versalt första tecken.

Laddningsskript

I detta exempel används samma funktioner (`Upper`, `Mid` och `Len`) som i scenariot med diagramuttryck. Laddningsskriptet skapar ett nytt fält, `NewInputText`, som returnerar `InputText` med ett versalt första tecken.

```
Example:
Load InputText, First&Second as NewInputText;
Load *, mid(InputText,len(First)+1) as Second;
Load *, upper(left(InputText,1)) as First;
Load  * inline [
InputText
this is a sample text string
capitalize first letter only ];
```

Resultattabell

InputText	Nyindatatext
this is a sample text string	This is a sample text string
capitalize first letter only	Capitalize first letter only

LevenshteinDist

LevenshteinDist() returnerar Levenshtein-avståndet mellan två strängar. Det definieras som det minsta antalet enkelteckensredigeringar (infogningar, borttagningar eller ersättningar) som krävs för att förvandla en sträng till en annan. Funktionen är användbar vid ungefärliga strängjämförelser.

Syntax:

```
LevenshteinDist(text1, text2)
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
text1	Första strängen.
text2	Den andra strängen som kommer att jämföras med den första strängen för att beräkna det minsta antalet redigeringar av enstaka tecken.

Exempel: diagramuttryck

Exempel	Resultat
`LevenshteinDist( 'Kitten','Sitting' )`	Returnerar `3`

Exempel – Grunderna i LevenshteinDist

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `InputText`.

Laddningsskript

```
Example:
Load * inline [
InputText
Sliver
SSiver
SSiveer
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `InputText`

Skapa följande mått:

- `=LevenshteinDist('Silver', InputText)`, för att beräkna det minsta antalet redigeringar av enstaka tecken som krävs för att förändra strängvärdena för `InputText` till ordet `Silver`.

Resultattabell

InputText	LevenshteinDist('Silver', InputText)
Sliver	2
SSiveer	3
SSiver	2

Utdata från funktionen `LevenshteinDist` ger det antal ändringar som krävs för att förändra `InputText` till den förväntade texten, `Silver`. Första raden kräver till exempel två ändringar för att ändra ordet `Sliver` till `Silver`. Den andra raden kräver tre ändringar: 1) Ta bort det extra tecknet `S`. 2) Ta bort det extra tecknet `e`. 3) Infoga ett nytt tecken `l`.

Exempel – LevenshteinDist-scenario

Diagramuttryck

Översikt

I detta exempel konsolideras produktnamn från olika system. Produktnamnen har inte alltid samma stavning på grund av skrivfel, förkortningar, utrymme eller andra variationer. Med funktionen `LevenshteinDist` kan du mäta likheten mellan två produktnamn och identifiera vilka som sannolikt hänvisar till samma produkt, även om namnen inte är identiska.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ProductA`
  - `ProductB`

Laddningsskript

```
Example:
Load * inline [
ProductA, ProductB
Coca Cola 330ml, CocaCola 330 ml
```

```
Pepsi 500 ml, Pepsi 500ml
Sprite Zero 600 ml, SpriteZero600ml
Red Bull 250ml, Redbull 250ml
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ProductA`
- `ProductB`

Skapa följande mått:

- `=LevenshteinDist(ProductA, ProductB)`, för att beräkna det minsta antalet redigeringar av enstaka tecken som krävs för att förändra strängvärdet för `ProductB` till att matcha `ProductA`.

Resultattabell

ProductA	ProduktB	LevenshteinDist(ProduktA, ProduktB)
Coca Cola 330 ml	CocaCola 330 ml	2
Pepsi 500 ml	Pepsi 500 ml	1
Red Bull 250 ml	Redbull 250 ml	2
Sprite Zero 600 ml	Sprite Zero 600 ml	3

Levenshtein-avstånd är en typ av fuzzy matchning som ofta används som en del av stavningskontroller, optisk teckenigenkänning och korrigeringssystem inom områden som kunddatahantering, lagersystem och behandling av dokument, där små variationer i text ofta förekommer.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

Laddningsskript

```
Example:
Load *, recno() as ID;
Load 'Silver' as String_1,* inline [
String_2
Sliver
SSiver
SSiveer ];

Example:
Load *, recno()+3 as ID;
Load 'Gold' as String_1,* inline [
String_2
Bold
Bool
Bond ];
```

```
Example:
Load *, recno()+6 as ID;
Load 'Ove' as String_1,* inline [
String_2
Ove
Uve
Üve ];

Example:
Load *, recno()+9 as ID;
Load 'ABC' as String_1,* inline [
String_2
DEFG
abc
ビビビ ];

set nullinterpret = '<NULL>';
Example:
Load *, recno()+12 as ID;
Load 'X' as String_1,* inline [
String_2
''
<NULL>
1 ];

R1:
Load
ID,
String_1,
String_2,
LevenshteinDist(String_1, String_2) as LevenshteinDistance
resident Example;

Drop table Example;
```

Resultattabell

ID	Sträng_1	Sträng_2	LevenshteinDistance
1	Silver	Sliver	2
2	Silver	SSiver	2
3	Silver	SSiveer	3
4	Guld	Fetstil	1
5	Guld	Bool	3
6	Guld	Bond	2
7	Ove	Ove	0
8	Ove	Uve	1
9	Ove	Üve	1

ID	Sträng_1	Sträng_2	LevenshteinDistance
10	ABC	DEFG	4
11	ABC	abc	3
12	ABC	surr	3
13	X		1
14	X	-	1
15	X	1	1

Lower

Lower() konverterar alla tecken i indatasträngen till gemener.

Syntax:

```
Lower(text)
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Sträng som ska utvärderas.

Exempel: diagramuttryck

Exempel	Resultat
`Lower( 'abcD' )`	Returnerar `abcd`

Exempel – grunderna i Lower

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `InputText`.

Laddningsskript

```
Example:
Load * inline [
InputText
rHode iSland
```

```
washingTon d.C.
new york
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `InputText`

Skapa följande beräknade dimension:

- `=Lower(InputText)` för att konvertera alla tecken i fältet `InputText` till gemener.

Resultattabell

InputText	Lower(InputText)
New York	New York
rHode iSland	rhode island
washingTon d.C.	washington d.c.

Utdata från funktionen `Lower` konverterar alla värden till gemener.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Load
String,
Lower(String)
Inline
[String
rHode iSland
washingTon d.C.
new york];
```

Resultattabell

Sträng	Lägre(Textsträng)
rHode iSland	rhode island
washingTon d.C.	washington d.c.
new york	new york

Exempel – Lower-scenario

Diagramuttryck

Översikt

En datauppsättning med kunddata innehåller e-postadresser med tecken med både versaler och gemena för domänen. Företaget vill standardisera dessa adresser så att endast gemener används.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `CustomerID`
  - `EmailAddress`
  - `Inquiry`

Laddningsskript

```
Example:
Load * inline [
CustomerID,    EmailAddress, Inquiry
1001,john.doe@Gmail.com, Inquiry about product return
1002,mary.jane@yahoo.com, Inquiry about order status
1003,a.smith@GMAIL.COM, Inquiry about shipping details
1004,lisa.black@outlook.com, General product inquiry
1005,james.wong@Yahoo.COM, Question about pricing
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `CustomerID`
- `EmailAddress`

Skapa följande beräknade dimension:

- `=Lower(EmailAddress)` för att separera domändelen av e-postmeddelandet som en ny kolumn och konvertera tecknen till gemener.

Resultattabell

Kund-id	EmailAddress	Lower(EmailAddress)
1001	john.doe@Gmail.com	john.doe@gmail.com
1002	MARY.JANE@YAHOO.COM	mary.jane@yahoo.com
1003	a.smith@GMAIL.COM	a.smith@gmail.com
1004	lisa.black@Outlook.Com	lisa.black@outlook.com
1005	james.wong@Yahoo.COM	james.won@yahoo.com

Utdata från funktionen `Lower` returnerar alla e-postadresser med gemener.

LTrim

LTrim() returnerar strängen rensad på inledande blankstegstecken.

Syntax:

```
LTrim(text)
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Sträng som ska utvärderas.

Exempel: diagramuttryck

Exempel	Resultat
`LTrim( ' abc' )`	Returnerar `abc`
`LTrim( 'abc ' )`	Returnerar `abc`

Exempel – grunderna i LTrim

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `InputText`.

Laddningsskript

```
Set verbatim=1;
Example:
Load * inline [
InputText
'   abc    '
' def  '];
```

Satsen "`Set verbatim=1`" finns med i exemplet för att säkerställa att blankstegen inte rensas automatiskt före demonstration av `LTrim` -funktionen. Mer information finns i Verbatim (page 239).

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `InputText`

Skapa följande beräknade dimension:

- `=LTrim(InputText)` för att ta bort inledande blanksteg från `InputText`.

Resultattabell

InputText	LTrim(Indatatext)
” abc ”	”abc ”
” def ”	”def ”

Utdata från funktionen `LTrim` tog bort alla inledande blanksteg till vänster om originaltexten men behöll alla efterföljande blanksteg .

Exempel – LTrim-scenario

Diagramuttryck

Översikt

Ett CRM-system (Customer Relationship Management) innehåller poster med inkonsekvent datainmatning som innehåller extra inledande blanksteg. För rapporteringsändamål måste uppgifterna rensas för att ta bort dessa blanksteg och för att säkerställa korrekt sortering och gruppering av kundnamn.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `CustomerName`.

Laddningsskript

```
Set verbatim=1;
Example:
Load * inline [
CustomerName
'  John Doe'
'Jane Smith'
'  Michael Johnson'
'Emily Davis'
];
```

Satsen "`Set verbatim=1`" finns med i exemplet för att säkerställa att blankstegen inte rensas automatiskt före demonstration av `LTrim` -funktionen. Mer information finns i Verbatim (page 239).

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `CustomerName`

Skapa följande beräknade dimension:

- `=LTrim(CustomerName)` för att ta bort inledande blanksteg från `CustomerName`.

Resultattabell

Kundnamn	LTrim(Kundnamn)
' Michael Johnson'	'Michael Johnson'
' John Doe'	'John Doe'
'Emily Davis'	'Emily Davis'
'Jane Smith'	'Jane Smith'

Utmatningen visar att funktionen `LTrim` tog bort alla inledande blanksteg från de ursprungliga strängvärdena i `CustomerName`.

Exempel – LTrim avancerat scenario

Diagramuttryck

Översikt

I detta exempel tas alla inledande blanksteg bort från den ursprungliga textsträngen. Diagramuttrycket innehåller åtgärder som använder funktionen `Len` för att räkna tecknen i strängen före och efter användning av funktionen `LTrim`.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `String`.

Laddningsskript

```
Set verbatim=1;
Example:
Load * inline [
String
'   abc    '
' def  '];
```

Satsen "Set verbatim=1" finns med i exemplet för att säkerställa att blankstegen inte rensas automatiskt före demonstration av `LTrim`-funktionen. Mer information finns i Verbatim (page 239).

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `String`

Skapa följande beräknade dimension:

- `=LTrim(String)` för att ta bort extra inledande blanksteg.

Skapa följande mått:

- `=Len(String)`, för att räkna längden på den ursprungliga strängen.
- `=Len(LTrim(String))`, för att räkna längden på strängen efter att de inledande blankstegen har tagits bort.

Resultattabell

Sträng	LTrim(Sträng)	Len(Sträng)	Längd(TrimVänster (Sträng))
' abc '	'abc '	10	7
' def '	'def '	6	5

När du jämför resultatet från funktionen `LTrim` med de ursprungliga strängvärdena i skriptet kan du se hur alla inledande blanksteg har tagits bort.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Set verbatim=1;
Example:

Load *,
len(LtrimString) as LtrimStringLength;
Load *,
ltrim(String) as LtrimString;
Load *,
len(String) as StringLength;
Load * Inline [
String
'   abc    '
' def  '];
```

Satsen ”Set verbatim=1” finns med i exemplet för att säkerställa att utrymmena inte rensas automatiskt före demonstration av ltrim-funktionen. Mer information finns i Verbatim (page 239).

Resultattabell

Sträng	SträngLängd	TrimVänsterSträng	TrimVänsterSträngLängd
abc	10	abc	7
def	6	def	5

Se även:

p *RTrim (page 1835)*

MatchRegEx

MatchRegEx() jämför en indatasträng (**text**) med ett eller flera reguljära uttryck (**regex**). Den returnerar den numeriska platsen för de reguljära uttrycksmönster som matchar. Returnerar ett matchningssvar endast om utvärderingen av det reguljära uttrycket matchar hela textinmatningen. Om inga matchningar hittas returnerar funktionen 0.

Den här funktionen utför regexoperationer som är skiftlägeskänsliga. Alternativt kan du använda varianten **MatchRegExI()** för att utföra skiftlägesokänsliga regexoperationer.

```
MatchRegEx (text, regex1 [ , regex2,...regexN])
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
text	Indatasträngens text till vilken du vill matcha ett eller flera reguljärt uttryck. För att en matchning ska ske måste denna indatasträng vara en exakt matchning med **regex** mönstret.
regex	Det reguljära uttryck som ska försöka matcha indatasträngen. Du kan ange mer än ett reguljärt uttryck i ett och samma uttryck, separerade som efterföljande argument.

Exempel på funktioner

Exempel	Resultat
`MatchRegEx ('testSample','test')`	Returnerar `0` . Funktionen returnerar endast ett matchningssvar när det finns en exakt matchning mellan **text** och **regex** mönstret. Även om `testSample` innehåller `test`, innehåller den annan text (`Sample`), vilket förhindrar en exakt matchning.
`MatchRegEx('abc123','[a-z]','[0-9]','[a-z]+[0-9]+','.+')`	Returnerar `3` .Det tredje regexmönstret är det första som matchar datan.
`MatchRegEx('ABC','[abc]+','[123]')`	Returnerar `0`, eftersom **MatchRegEx()** är skiftlägeskänslig.
`MatchRegExI('ABC','[abc]+','[123]')`	Returnerar `1` .Den skiftlägesokänsliga varianten av funktionen, **MatchRegExI()**, används.

Användning

Användningsfall för denna funktion inkluderar:

- Validering av text för att uppfylla formaterings- och efterlevnadsstandarder och för att identifiera fel i data.
- Identifiera skillnader i hur text är formaterad. Du kan till exempel identifiera hur många telefonnummer som använder format A, format B och så vidare.

Exempel 1 – laddningsskript för att analysera telefonnummerformat

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell `BusinessContactInfo` som innehåller telefonnummer till kontaktpersoner för olika företag. Telefonnumren är 10-siffriga NANP-telefonnummer.
- Ett beräknat fält, `PhoneNumberFormat`, för att bedöma om respektive telefonnummer matchar någon specifik uppsättning av telefonnummerformat.

Det finns tre olika format som vi vill kolla efter: (###) ###-####, ###-###-#### och ##########.

Laddningsskript

```
BusinessContactInfo:

Load MatchRegEx(ContactInfo,'\({0,1}[0-9]{3}\) [0-9]{3}-[0-9]{4}', '[0-9]{3}-[0-9]{3}-[0-9]
{4}', '[0-9]{10}') as PhoneNumberFormat,
* Inline `
ID      CompanyName     ContactInfo
1       Company A       (123) 456-7890
2       Company B       0123456790
3       Company C       234-567-8901
4       Company D       (024) 680-2456
5       Company E       (135) 791-3579
6       Company F       8901234567
7       Company G       235-235-2352
8       Company H       (555) (555) (5555)
9       Company I       2342342343
10      Company J       444-444-4444
` (delimiter is '\t');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `CompanyName`
- `ContactInfo`
- `PhoneNumberFormat`

Resultattabell

CompanyName	ContactInfo	PhoneNumberFormat
Company A	(123) 456-7890	1
Company B	0123456790	3
Company C	234-567-8901	2
Company D	(024) 680-2456	1
Company E	(135) 791-3579	1
Company F	8901234567	3
Company G	235-235-2352	2
Company H	(555) (555) (5555)	0
Company I	2342342343	3
Company J	444-444-4444	2

Härifrån kan du till exempel skapa KPI:er för att beräkna det totala antalet för varje telefonnummerformat, samt antalet telefonnummer som inte har ett erkänt format:

Exempel på aggregeringsuttryck för att räkna förekomster av respektive format

Format	Antal instanser	Diagramuttryck
(###) ###-####	3	=Count({<PhoneNumberFormat={'1'}>} PhoneNumberFormat)
###-###-####	3	=Count({<PhoneNumberFormat={'2'}>} PhoneNumberFormat)
##########	3	=Count({<PhoneNumberFormat={'3'}>} PhoneNumberFormat)
Unrecognized format	1	=Count({<PhoneNumberFormat={'0'}>} PhoneNumberFormat)

Exempel 2 – diagramuttryck för att identifiera ISBN-format

Diagramuttryck och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell `Books` med ISBN-koder för olika böcker.

Från dessa data vill vi identifiera om varje ISBN-kod är en giltig 10-siffrig eller 13-siffrig kod, baserat på syntax som anges i ett regex-mönster.

Laddningsskript

```
Books:

Load * Inline [
Book
ISBN 123-3-1234-1234-0
ISBN 012-2-0123-0123-4
ISBN 000-1-0123-0123-2
ISBN 0-111-23456-7
ISBN 555-2-5555-5555-3
ISBN 222-4-2222-2222-2
ISBN 1-901-23456-8
ISBN 333-3-3333-3333-3
ISBN 555-1-5151-5151-3
ISBN 232-1-2323-2323-1
ISBN 2-444-44444-4
ISBN 888-0-9999-0000-0
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Book`

Lägg till denna beräknade dimension och ge den etiketten *ISBNFormat*:

```
=if(MatchRegEx(Book, 'ISBN[ ]*([0-9]{3})-([0-9]{1})-([0-9]{4})-([0-9]{4})-([0-9]{1})', 'ISBN[
]*([0-9]{1})-([0-9]{3})-([0-9]{5})-([0-9]{1})')=1, '13-digit',
       if(MatchRegEx(Book, 'ISBN[ ]*([0-9]{3})-([0-9]{1})-([0-9]{4})-([0-9]{4})-([0-9]{1})', 'ISBN[
]*([0-9]{1})-([0-9]{3})-([0-9]{5})-([0-9]{1})')=2, '10-digit',
               'Invalid ISBN'))
```

Resultattabell

Book	ISBN Format
ISBN 000-1-0123-0123-2	13-digit
ISBN 0-111-23456-7	10-digit
ISBN 1-901-23456-8	10-digit
ISBN 2-444-44444-4	10-digit
ISBN 012-2-0123-0123-4	13-digit
ISBN 123-3-1234-1234-0	13-digit

Book	ISBN Format
ISBN 222-4-2222-2222-2	13-digit
ISBN 232-1-2323-2323-1	13-digit
ISBN 333-3-3333-3333-3	13-digit
ISBN 555-1-5151-5151-3	13-digit
ISBN 555-2-5555-5555-3	13-digit
ISBN 888-0-9999-0000-0	13-digit

Mid

Mid() returnerar den del av indatasträngen som börjar vid positionen för det tecken som definieras av det andra argumentet, ”start”, och returnerar det antal tecken som definieras av det tredje argumentet ”count”. Om ”count” utelämnas returneras resten av indatasträngen. Det första tecknet i indatasträngen numreras som 1.

Syntax:

```
Mid(text, start[, count])
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
start	Heltal som definierar positionen för det första tecknet i **text** som ska inkluderas.
count	Definierar stränglängden för utdatasträngen. Om den utelämnas inkluderas alla tecken från positionen som definieras av **start**.

Exempel: diagramuttryck

Exempel	Resultat
`Mid( 'abcdef',3 )`	Returnerar `cdef`
`Mid( 'abcdef',3, 2 )`	Returnerar `cd`

Exempel – grunderna i Mid

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `InputText` den ursprungliga textsträngen
  - `StartPosition` startpositionen för det första tecknet som ska inkluderas från den ursprungliga textsträngen efter bearbetning
  - `CharacterCount` antalet tecken som ska returneras (valfri parameter)

Laddningsskript

```
Example:
Load * inline [
InputText, StartPosition, CharacterCount
abcdef, 3, 2
abcdef, 2, 3
210714, 3, 2
210714, 2, 3
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `InputText`
- `StartPosition`
- `CharacterCount`

Skapa följande beräknade dimensioner:

- `=mid(InputText,StartPosition)` för att beräkna de tecken som ska extraheras från `InputText` med utgångspunkt från värdet i fältet `StartPosition`. Funktionen returnerar alla tecken från värdet `StartPosition` till slutet av `InputString` eftersom ingen parameter för `Count` har angetts.
- `=mid(InputText,StartPosition,CharacterCount)` för att beräkna de tecken som ska extraheras från `InputText` med utgångspunkt från värdet i fältet `StartPosition`. Värdet i `CharacterCount` bestämmer antalet tecken som returneras, med startvärde från värdet i fältet `StartPosition`.

Resultattabell

Indatatext	Startposition	AntalTecken	Mid (Indatatext,Startposition)	Mid (Indatatext,Startposition,AntalTecken
abcdef	2	3	bcdef	bcd
abcdef	3	2	cdef	cd
20210714	2	3	10714	107
20210714	3	2	0714	07

Den första raden returnerar värdet `bcdef` för det första uttrycket. Uttrycket börjar vid position 2 och returnerar alla tecken eftersom argumentet `Count` inte är inställt. Det andra uttrycket börjar också vid position 2 men returnerar bara tecknen `bcd` eftersom argumentet `Count` är 2.

Den andra raden returnerar värdet `cdef` för det första uttrycket. Uttrycket börjar vid position 3 och returnerar alla tecken eftersom argumentet `Count` inte är inställt. Det andra uttrycket börjar också vid position 3 men returnerar bara tecknen `cd` eftersom argumentet `Count` är 2.

Den tredje raden returnerar värdet `10714` för det första uttrycket. Uttrycket börjar vid position 2 och returnerar alla tecken eftersom argumentet `Count` inte är inställt. Det andra uttrycket börjar också vid position 2 men returnerar bara tecknen `107` eftersom argumentet `Count` är 3.

Den fjärde raden returnerar värdet `0714` för det första uttrycket. Uttrycket börjar vid position 3 och returnerar alla tecken eftersom argumentet `Count` inte är inställt. Det andra uttrycket börjar också vid position 2 men returnerar bara tecknen `07` eftersom argumentet `Count` är 2.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Example:
Load *,
mid(Text,StartPosition) as Mid1,
mid(Text,StartPosition,Count) as Mid2;
Load * inline [
Text, StartPosition, Count
'abcdef', 3, 2
'abcdef', 2, 3
'210714', 3, 2
'210714', 2, 3
];
```

Resultattabell

Text	Startposition	Mitt1	Antal	Mid2
abcdef	3	cdef	2	cd
abcdef	2	bcdef	3	bcd
210714	3	0714	2	07
210714	2	10714	3	107

Exempel – Mid-scenario

Diagramuttryck

Översikt

En datakälla innehåller en kolumn som heter `OrderID` och som använder följande strukturerade format: `ORDYYYY-COUNTRYCODE-ORDERNUMBER`. För att uppfylla ett rapporteringskrav måste du extrahera och presentera landskoden som en separat kolumn.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Orders`.
- Ett fält i datatabellen heter `OrderID`.

Laddningsskript

```
Orders:
Load * inline [
OrderID
ORD2024-US-123456
ORD2024-UK-654321
ORD2024-CA-789123
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `OrderID`

Skapa följande beräknade dimension:

- `=Mid(OrderID, 9, 2)` för att extrahera den tvåbokstaviga landskoden från `OrderID`.

Resultattabell

Order-ID	Mid(OrderID, 9, 2)
ORD2024-CA-789123	CA
ORD2024-UK-654321	Storbritannien
ORD2024-US-123456	USA

Utdata från funktionen `Mid` har framgångsrikt extraherat landskoderna med två bokstäver från strängvärdena på `OrderID`.

Se även:

p *Index (page 1750)*

Ord

Ord() returnerar det numeriska värdet (ASCII eller Unicode) för det första tecknet i en sträng. Denna funktion är användbar för att utvärdera eller jämföra strängar baserat på deras underliggande teckenkoder, t.ex. vid sortering eller filtrering av strängar med icke-standardiserade tecken.

Syntax:

```
Ord(text)
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.

Exempel: diagramuttryck

Exempel	Resultat
`Ord( 'A' )`	Returnerar heltalet `65`.
`Ord( 'Ab' )`	Returnerar heltalet `65`.

Exempel – grunderna i Ord

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Chinese`
  - `Western`

Laddningsskript

```
Example:
Load * inline [
Chinese, Western
古琴, Guqin
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Chinese`
- `Western`

Skapa följande beräknade dimensioner:

- `=ord(Chinese)` för att returnera det numeriska värdet av värdet i fältet `Chinese`.
- `=ord(Western)`, för att returnera det numeriska värdet av värdet i fältet `Western`.

Resultattabell

Kinesiska	Västerländska	ord(Kinesiska)	ord(Western)
古琴	Guqin	21476	71

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Example:
Load *,
ord(Chinese) as OrdUnicode,
ord(Western) as OrdASCII;
Load * inline [
Chinese, Western
古琴, Guqin ];
```

Resultattabell

Kinesiska	Västerländska	OrdUnicode	OrdASCII
古琴	Guqin	21476	71

Exempel – Ord-scenario

Diagramuttryck

Översikt

En datauppsättning med produktkoder innehåller vissa koder som betraktas som ogiltiga eftersom de börjar med ett specialtecken i stället för ett alfanumeriskt tecken. Med hjälp av funktionen `Ord` för att identifiera de ogiltiga produktkoderna skapas en ny åtgärd för att märka posterna med antingen giltig eller ogiltig.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Products`.
- Ett fält i datatabellen heter `ProductCode`.

Laddningsskript

```
Products:
Load * inline [
ProductCode
AB123
#EX45
12XY9
DEF456
*KLM78
GH789
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `ProductCode`

Skapa följande beräknade dimensioner:

- `=Ord(ProductCode)`, för att returnera det numeriska värdet för det första tecknet.
- `=If(Ord(ProductCode) < 48 OR (Ord(ProductCode) > 57 AND Ord(ProductCode) < 65) OR Ord (ProductCode) > 122, 'Invalid Product Code', 'Valid Product Code')` för att kontrollera om kodens ordinalvärde är alfanumeriskt. Om den inte är det flaggas den posten som `Not Valid`.

Resultattabell

Produktkod	Ord(ProductCode)	If(Ord(ProductCode) < 48 OR (Ord(ProductCode) > 57 AND Ord(ProductCode) < 65) OR Ord (ProductCode) > 122, 'Invalid Product Code', 'Valid Product Code')
*KLM78	42	Invalid Product Code
#EX45	35	Invalid Product Code
12XY9	49	Valid Product Code
AB123	65	Valid Product Code
DEF456	68	Valid Product Code
GH789	71	Valid Product Code

Genom att jämföra resultatet från funktionen `Ord` med ett intervall av giltiga alfanumeriska värden kan du avgöra vilka produktkoder som inte är giltiga.

PurgeChar

PurgeChar() returnerar en sträng som består av de tecken som finns i indatasträngen (”text”), utom alla eventuella tecken som visas i det andra argumentet (”remove_chars”).

Syntax:

```
PurgeChar(text, remove_chars)
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
remove_chars	En sträng som innehåller tecknen i **text** som ska tas bort.

Exempel: diagramuttryck

Exempel	Resultat
`PurgeChar( 'a1b2c3','123' )`	Returnerar `abc`
`PurgeChar( 'a1b2c3','1234' )`	Returnerar `abc`
`PurgeChar( 'a1b22c3','1234' )`	Returnerar `abc`
`PurgeChar( 'a1b2c3','312' )`	Returnerar `abc`
`PurgeChar ( 'a1b2c3','Abc' )`	Returnerar `a123`

Exempel – grunderna i PurgeChar

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `InputText`
  - `CharsToPurge`

Laddningsskript

```
Example:
Load * inline [
InputText, CharsToPurge
'a1b2c3', '123'
'a^b^c', '^'
'A^b^c', 'A^'
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `InputText`
- `CharsToPurge`

Skapa följande beräknade dimension:

- `=PurgeChar(InputText,CharsToPurge)` för att beräkna de tecken i `InputText` som matchar tecknen i `CharsToPurge` och endast returnera de tecken som inte matchar.

Results table

InputText	CharsToPurge	PurgeChar(InputText,CharsToPurge)
A^b^c	A^	bc
a^b^c	^	abc
a1b2c3	123	abc

Utdata från funktionen `PurgeChar` returnerar endast de tecken från `InputText` som inte matchar tecknen i `CharsToPurge`. Den första raden returnerar t.ex. `bc` och tar bort övriga tecken.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Example:
Load
*,
purgechar(String1, String2) as PurgeChar;
Load * inline [
String1, String2
'a1b2c3', '123'
];
```

Resultattabell

sträng(1)	Sträng2	PurgeChar
a1b2c3	123	abc

Exempel – PurgeChar-scenario

Diagramuttryck

Översikt

En datauppsättning innehåller kundnamn och telefonnummer. I det här exemplet analyseras telefonnummersträngarna så att endast numeriska tecken behålls och alla andra tecken, t.ex. parenteser, bindestreck, punkter, plustecken och blanksteg, tas bort.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `CustomerName`
  - `PhoneNumber`

Laddningsskript

```
Example:
Load * inline [
CustomerName, PhoneNumber
John Doe, (123) 456-7890
Jane Smith, 123-456-7890
Bob Johnson, 123.456.7890
Alice Brown, +1 (123) 456 7890
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `CustomerName`
- `PhoneNumber`

Skapa följande beräknade dimension:

- `=PurgeChar(PhoneNumber, '()- .+ ')` för att extrahera de icke-numeriska tecknen från fältet `PhoneNumber` och endast returnera numeriska tecken.

Results table

CustomerName	PhoneNumber	PurgeChar(PhoneNumber,'()-.+ ')
Alice Brown	+1 (123) 456 7890	11234567890
Bob Johnson	123.456.7890	1234567890
Jane Smith	123-456-7890	1234567890
John Doe	(123) 456-7890	1234567890

Utdata från `PurgeChar`-funktionen har tagit bort alla icke-numeriska tecken från `PhoneNumber`-strängen och returnerar endast siffror.

Se även:

p *KeepChar (page 1777)*

Repeat

Repeat() bildar en sträng som består av indatasträngen som upprepats det antal gånger som definieras av det andra argumentet.

Syntax:

```
Repeat(text[, repeat_count])
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
repeat_count	Definierar antalet gånger tecknen i strängen **text** ska upprepas i utdatasträngen.

Exempel: diagramuttryck

Exempel	Resultat
`Repeat( ' * ', 4 )`	Returnerar `****`

Exempel – grunderna i Repeat

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `InputText`.

Laddningsskript

```
Example:
Load * inline [
InputText
hello world!
hOw aRe you?
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `InputText`

Skapa följande beräknade dimension:

- `=Repeat(InputText,2)` för att beräkna antalet gånger som värdet i `InputText` ska upprepas. I detta exempel kommer varje sträng att upprepas två gånger.

Resultattabell

Indatatext	Upprepa(Indatatext,2)
hej världen!	`hello world!hello world!`
hUr mÅr du?	`hOw aRe you?hOw aRe you?`

Utdata från funktionen `Repeat` innehåller till exempel två förekomster av strängen `InputText`, rad 1 returnerar `hello world!hello world!`.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Example:
Load *,
repeat(String,2) as Repeat;
Load * inline [
String
hello world!
hOw aRe you? ];
```

Resultattabell

Sträng	Upprepa
hello world!	hello world!hello world!
hOw aRe you?	hOw aRe you?hOw aRe you?

Exempel – Repeat-scenario

Diagramuttryck

Översikt

I det här exemplet används data om kundnöjdhet för att konvertera en numerisk nöjdhetsgrad till en visuell representation med hjälp av smileys ☺.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `Customer`
  - `SatisfactionScore`

Laddningsskript

```
Example:
Load * inline [
Customer, SatisfactionScore
John,4
```

```
Mary,2
Peter,5
Susan,3
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Customer`
- `SatisfactionScore`

Skapa följande beräknade dimension:

- `=Repeat('☺',(SatisfactionScore))`, för att beräkna och visa poängen visuellt med emojis.

Resultattabell

Kund	SatisfactionScore	Repeat ('☺',SatisfactionScore)
John	4	☺☺☺☺
Mary	2	☺☺
Peter	5	☺☺☺☺☺
Susan	3	☺☺☺

Utdata från funktionen `Repeat` visar en visuell representation av kundnöjdheten med hjälp av smileys som representerar den numeriska nöjdhetsgraden.

Exempel – Repeat avancerat scenario

Diagramuttryck

Översikt

En datauppsättning med kundkonton innehåller kontonummer med varierande teckenlängder. I det här exemplet visas hur du använder funktionen `Repeat` för att standardisera längden på kontonummer.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `AccountNo`.

Laddningsskript

```
Example:
Load * inline [
AccountNo
```

```
123
45125783
41523
489654
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `AccountNo`

Skapa följande beräknade dimension:

- `=Repeat(0,10 - Len(AccountNo)) & AccountNo` för att fylla på kontonumret med nollor till en total längd på 10 tecken.

Resultattabell

AccountNo	Repeat(0, 10 - Len(AccountNo)) & AccountNo
123	0000000123
41523	0000041523
489654	0000489654
45125783	0045125783

Utdata från funktionen `Repeat` visar en standardiserad längd på 10 tecken för alla kontonummer.

Replace

Replace() returnerar en sträng efter att ha ersatt alla förekomster av en given delsträng inom indatasträngen med en annan delsträng. Funktionen är inte rekursiv och fungerar från vänster till höger.

Syntax:

```
Replace(text, from_str, to_str)
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
from_str	En sträng som kan förekomma en eller flera gånger inom indatasträngen **text**.
to_str	Strängen som ersätter alla förekomster av **from_str** inom strängen **text**.

Exempel: diagramuttryck

Exempel	Resultat
`Replace( 'abccde','cc','xyz' )`	Returnerar `abxyzde`

Exempel – grunderna i Replace

Diagramuttryck

Översikt

En datauppsättning med kundnamn innehåller vissa felstavningar. I det här exemplet används funktionen `Replace` för att bearbeta och ersätta alla felaktiga förekomster av "Jhon" med "John".

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `CustomerID`
  - `CustomerName`

Laddningsskript

```
Example:
Load * inline [
CustomerID, CustomerName
1, Jhon Smith
2, Jhon Doe
3, John Williams
4, Jhonathan Harris
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `CustomerID`
- `CustomerName`

Skapa följande beräknade dimension:

- `=Replace(CustomerName,'Jhon','John')`, för att ersätta alla förekomster av `Jhon` med `John`.

Resultattabell

Kund-id	Kundnamn	Ersätt(Kundnamn,'Jhon','John')
1	Jhon Smith	John Smith

Kund-id	Kundnamn	Ersätt(Kundnamn,'Jhon','John')
2	Jhon Doe	John Doe
3	John Williams	John Williams
4	Jhonathan Harris	Johnathan Harris

Utdata från funktionen `Repeat` hittar och korrigerar alla förekomster av `Jhon` med `John`.

Exempel – Replace-scenario

Diagramuttryck

Översikt

Ett dataset med produkter använder inkonsekvent märkning för kategorikoderna. Exempelvis har elektriska produkter flera olika kodvarianter, t.ex. `ELEC`, `ELC` och `elc`. För att standardisera koderna används funktionen `Replace` för att korrigera alla inkonsekventa kategorikoder för både el- och möbelprodukter.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ProductID`
  - `CategoryCode`

Laddningsskript

```
Example:
Load * inline [
ProductID, CategoryCode
1, ELEC
2, ELC
3, FURN
4, FRN
5, ELEC
6, ELC
7, elc
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ProductID`
- `CategoryCode`

Skapa följande beräknade dimension:

- `=Replace(Replace(Upper(CategoryCode), 'ELC', 'ELEC'),'FRN', 'FURN')`, för att ersätta alla förekomster av `ELC` med `ELEC` och alla förekomster av `FRN` med `FURN`.

Resultattabell

Produkt-ID	Kategorikod	Ersätt(Ersätt(Versal (Kategorikod), 'ELC', 'ELEC'),'FRN', 'FURN')
1	El	El
2	EL	EL
3	MÖB	MÖB
4	MÖB	Möbler
5	El	El
6	El	El
7	El	El

Utdata från funktionen `Replace` har ersatt alla förekomster av `ELC` med `ELEC`, och `FRN` med `FURN`. Observera att även koden `elc` har ersatts. Även om funktionen `Replace` är skiftlägeskänslig, använder dimensionsformeln `Upper`-funktionen för att skiftlägesanpassa alla `CategoryCodes` innan strängen ersätts. Detta gör att strängersättningen är skiftlägesoberoende.

Se även:

ReplaceRegEx

ReplaceRegEx() returnerar en sträng efter att ha ersatt en eller flera matchningar mellan en indatasträng och ett angivet reguljärt uttrycksmönster. Den text som ersätter den matchande texten anges i argumentet **to_str**. Funktionen går från vänster till höger, men om du anger ett negativt värde för **occurrence** läses funktionen från höger till vänster.

Den här funktionen utför regexoperationer som är skiftlägeskänsliga. Alternativt kan du använda varianten **ReplaceRegExI()** för att utföra skiftlägesokänsliga regexoperationer.

```
ReplaceRegEx (text, regex, to_str [, occurrence])
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Den text i indatasträngen som du vill ersätta, helt eller delvis, med den nya texten från **to_str**.

Argument	Beskrivning
regex	Det reguljära uttryck som definierar när text ska ersättas. Matchningar mellan detta argument och **text**-argumentet ersätts.
to_str	Den nya texten som du vill använda för att ersätta befintligt innehåll från **text**.
occurrence	Numret på matchningen (mellan inmatad text och reguljärt uttryck) som ska ersättas med ny text. Det här är ett valfritt argument. Standardvärdet är 0 om det utelämnas. När värdet 0 används, eller om argumentet utelämnas, ersätts alla matchningar mellan **text** och **regex** med **to_str**. Du kan ange ett negativt värde för **occurrence** för att hitta matchningar från höger till vänster.

Exempel på funktioner

Exempel	Resultat
`ReplaceRegEx ('abc123','[0-9]','x')`	Returnerar `abcxxx`. I det här exemplet ersätts all text från indatasträngen som matchar regexmönstret med den nya texten.
`ReplaceRegEx ('abc123','[0-9]','x',1)`	Returnerar `abcx23`. Endast den första förekomsten av en matchning mellan indatasträngen och regexmönstret ersätts.
`ReplaceRegEx ('abc123','[0-9]','x',4)`	Returnerar `abc123` . Det finns bara tre möjliga matchningar mellan indatasträngen och regexmönstret, så ingen ändring utförs.
`ReplaceRegEx ('ABC123','[a-c]','5')`	Returnerar `ABC123`. Ingen ändring görs eftersom **ReplaceRegEx()** är skiftlägeskänsligt och den skiftlägesokänsliga varianten inte används.
`ReplaceRegExI ('ABC123','[a-c]','5')`	Returnerar `555123`. Den skiftlägesokänsliga varianten av funktionen, **ReplaceRegExI()**, används.

Användning

Du kan använda **ReplaceRegEx()** för att ändra text så att den uppfyller formaterings- och efterlevnadsstandarder, särskilt när det finns komplexa mönster i dina data. Du kan t.ex. formatera om telefonnummersträngar så att endast numeriska värden inkluderas.

Vid behov kan du också använda den här funktionen för att maskera känslig information, t.ex. personliga identitetsuppgifter (PII), så att den inte visas för användare som analyserar din applikation.

Exempel 1 – laddningsskript för att uppdatera anställdas e-postadresser

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell `Employees` som innehåller e-postadresserna till de anställda i en organisation.
- Skapande av ett nytt fält, `Email_Updated`, för att utföra en sök- och ersättningsoperation på värden i fältet `Email_Original`.

Efter att företaget har bytt namn har alla e-postadresser uppdaterats för att återspegla det nya namnet.

Våra krav:

- Konvertera alla domänreferenser från `test.com` till `newcompanyname.com`.
- Vi måste använda regexoperationer utan skiftlägeskänslighet.

Laddningsskript

```
Employees:

Load
ReplaceRegExI(Email_Original, '@[a-zA-Z0-9.-]*\.?test\.com', '@newcompanyname.com') as Email_
Updated,
* Inline [
ID, Email_Original
1, john_doe@test.com
2, jane_doe@test.com
3, Michael_Andrews@test.com
4, charlie_kent@test.com
5, frances_johnson@test.com
6, rebecca_smith@test.com
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ID`
- `Email_Original`
- `Email_Updated`

Resultattabell

ID	Email_Original	Email_Updated
1	john_doe@test.com	john_doe@newcompanyname.com
2	jane_doe@test.com	jane_doe@newcompanyname.com
3	Michael_Andrews@test.com	Michael_ Andrews@newcompanyname.com
4	charlie_kent@support.test.com	charlie_ kent@newcompanyname.com
5	frances_johnson@test.com	frances_ johnson@newcompanyname.com
6	rebecca_smith@hr.test.com	rebecca_ smith@newcompanyname.com

Exempel 2 – laddningsskript för att formatera ISBN-koder

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- Variabler`ISBN_13` och `ISBN_10`, som lagrar de reguljära uttryck som ska användas för 13-siffriga respektive 10-siffriga ISBN-strängmönster.
- En tabell `Employees` som innehåller numeriska råvärden för ISBN-bokkoder. Tabellen innehåller både 13-siffriga och 10-siffriga ISBN-nummer.
- Flera operationer för att extrahera varje del av ett ISBN och sedan returnera en sträng som visar hela koden i det förväntade standardformatet. Vi använder en kombination av **ReplaceRegEx()**, **MatchRegEx()**, **ExtractRegExGroup()** och **Pick()**.

Våra krav:

- Vi vill formatera siffrorna till mer läsbar text med hjälp av den standardiserade formateringen:
  - ISBN *###-#-####-####-#*
  - ISBN *#-###-#####-#*

Laddningsskript

```
Set ISBN_13 = '^([0-9]{3})([0-9]{1})([0-9]{4})([0-9]{4})([0-9]{1})$';
Set ISBN_10 = '^([0-9]{1})([0-9]{3})([0-9]{5})([0-9]{1})$';

Formatted_Codes:

Load
```

```
Pick(MatchRegEx(ISBN_Unformatted, '$(ISBN_13)', '$(ISBN_10)'), ReplaceRegEx(ISBN_Unformatted,
'$(ISBN_13)', 'ISBN ' & ISBN_Component1 & '-' & ISBN_Component2 & '-' & ISBN_Component3 & '-'
& ISBN_Component4 & '-' & ISBN_Component5), ReplaceRegEx(ISBN_Unformatted, '$(ISBN_10)', 'ISBN
' & ISBN_Component1 & '-' & ISBN_Component2 & '-' & ISBN_Component3 & '-' & ISBN_Component4))
as ISBN_Formatted;

Load *,
Pick(MatchRegEx(ISBN_Unformatted, '$(ISBN_10)', '$(ISBN_13)'), ExtractRegExGroup(ISBN_
Unformatted, '$(ISBN_10)', 1, 1), ExtractRegExGroup(ISBN_Unformatted, '$(ISBN_13)', 1, 1)) as
ISBN_Component1,
Pick(MatchRegEx(ISBN_Unformatted, '$(ISBN_10)', '$(ISBN_13)'), ExtractRegExGroup(ISBN_
Unformatted, '$(ISBN_10)', 2, 1), ExtractRegExGroup(ISBN_Unformatted, '$(ISBN_13)', 2, 1)) as
ISBN_Component2,
Pick(MatchRegEx(ISBN_Unformatted, '$(ISBN_10)', '$(ISBN_13)'), ExtractRegExGroup(ISBN_
Unformatted, '$(ISBN_10)', 3, 1), ExtractRegExGroup(ISBN_Unformatted, '$(ISBN_13)', 3, 1)) as
ISBN_Component3,
Pick(MatchRegEx(ISBN_Unformatted, '$(ISBN_10)', '$(ISBN_13)'), ExtractRegExGroup(ISBN_
Unformatted, '$(ISBN_10)', 4, 1), ExtractRegExGroup(ISBN_Unformatted, '$(ISBN_13)', 4, 1)) as
ISBN_Component4,
Pick(MatchRegEx(ISBN_Unformatted, '$(ISBN_10)', '$(ISBN_13)'), ExtractRegExGroup(ISBN_
Unformatted, '$(ISBN_10)', 5, 1), ExtractRegExGroup(ISBN_Unformatted, '$(ISBN_13)', 5, 1)) as
ISBN_Component5
inline `
ISBN_Unformatted
1233123412340
0122012301234
0001012301232
0111234567
5552555555553
2224222222222
1901234568
3333333333333
5551515151513
2321232323231
2444444444
8880999900000
`;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `ISBN_Formatted`

Resultattabell

ISBN_Formatted
ISBN 000-1-0123-0123-2
ISBN 0-111-23456-7
ISBN 1-901-23456-8

ISBN_Formatted
ISBN 2-444-44444-4
ISBN 012-2-0123-0123-4
ISBN 123-3-1234-1234-0
ISBN 222-4-2222-2222-2
ISBN 232-1-2323-2323-1
ISBN 333-3-3333-3333-3
ISBN 555-1-5151-5151-3
ISBN 555-2-5555-5555-3
ISBN 888-0-9999-0000-0

Exempel 3 – diagramuttryck för att uppdatera sekundära e-postadresser

Diagramuttryck och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell `Employees` som innehåller e-postadresserna till de anställda i en organisation. Dessa lagras i en enda sträng för varje anställd, i vilken flera e-postadresser lagras.

Våra krav:

- Sekundära e-postadresser måste uppdateras så att de använder domänen `newcompanyname.com`. Vi vill inte uppdatera några primära e-postadresser.
- Vi måste använda regexoperationer utan skiftlägeskänslighet.

Laddningsskript

```
Employees:

Load * Inline [
ID, Emails
1, Primary: john_doe@test.com Secondary: john_michael_doe@example.com
2, Primary: jane_doe@test.com Secondary: jane_tabitha_doe_doe@example.com
3, Primary: Michael_Andrews@test.com Secondary: Michael_John_Andrews@example.com
4, Primary: charlie_kent@test.com Secondary: charlie_peter_kent@example.com
5, Primary: frances_johnson@test.com Secondary: frances_johnson@example.com
6, Primary: rebecca_smith@test.com Secondary: rebecca_danielle_smith@example.com
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ID`
- `Emails`

Lägg till följande beräknade dimension:

```
=ReplaceRegExI(Emails,'@[a-zA-Z0-9.-]*\.?[a-zA-Z0-9]*\.com','@newcompanyname.com',2)
```

Resultattabell

ID	Emails	=ReplaceRegExI(Emails,'@[a-zA-Z0-9.-]*\.?example\.com','@newcompanyname.com',2)
1	Primary: john_doe@test.com Secondary: john_michael_doe@example.com	Primary: john_doe@test.com Secondary: john_michael_doe@newcompanyname.com
2	Primary: jane_doe@test.com Secondary: jane_tabitha_doe_doe@example.com	Primary: jane_doe@test.com Secondary: jane_tabitha_doe_doe@newcompanyname.com
3	Primary: Michael_Andrews@test.com Secondary: Michael_John_Andrews@example.com	Primary: Michael_Andrews@test.com Secondary: Michael_John_Andrews@newcompanyname.com
4	Primary: charlie_kent@test.com Secondary: charlie_peter_kent@example.com	Primary: charlie_kent@test.com Secondary: charlie_peter_kent@newcompanyname.com.com
5	Primary: frances_johnson@test.com Secondary: frances_johnson@example.com	Primary: frances_johnson@test.com Secondary: frances_johnson@newcompanyname.com
6	Primary: rebecca_smith@test.com Secondary: rebecca_danielle_smith@example.com	Primary: rebecca_smith@test.com Secondary: rebecca_danielle_smith@newcompanyname.com

Genom att ange värdet 2 för argumentet **occurrence** kan vi uppdatera enbart den andra förekomsten som matchar regexmönstret.

ReplaceRegExGroup

ReplaceRegExGroup() returnerar en sträng efter att ha ersatt en eller flera matchningar mellan en indatasträng och det sammansatta reguljära uttrycksmönster som anges. Den text som ersätter den matchande texten anges i argumentet **to_str**. Funktionen går från vänster till höger, men om du anger ett negativt värde för **occurrence** läses funktionen från höger till vänster.

Den här funktionen utför regexoperationer som är skiftlägeskänsliga. Alternativt kan du använda varianten **ReplaceRegExGroupI()** för att utföra skiftlägesokänsliga regexoperationer.

```
ReplaceRegExGroup (text, regex, to_str, group [, occurrence])
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Den text i indatasträngen som du vill ersätta, helt eller delvis, med den nya texten från **to_str**.
regex	Det reguljära uttryck som definierar när text ska ersättas. Matchningar mellan detta argument och **text**-argumentet ersätts.
to_str	Den nya texten som du vill använda för att ersätta befintligt innehåll från **text**.
group	Gruppens nummer, om det rör sig om ett sammansatt reguljärt uttryck. Om det reguljära uttrycket bara innehåller en grupp använder du istället funktionen **ReplaceRegEx()**. Alternativt kan du använda **ReplaceRegExGroup()** med ett **group**-värde på `0`. Du kan ange ett negativt **group**-värde för att söka efter grupper från höger till vänster.
occurrence	Numret på matchningen (mellan inmatad text och reguljärt uttryck) som ska ersättas med ny text. Det här är ett valfritt argument. Standardvärdet är 0 om det utelämnas. När värdet 0 används, eller om argumentet utelämnas, ersätts alla matchningar mellan **text** och **regex** med **to_str**. Du kan ange ett negativt värde för **occurrence** för att hitta matchningar från höger till vänster.

Exempel på funktioner

Exempel	Resultat
`ReplaceRegExGroup ('abc123','([a-z])([0-9]+)','x',0)`	Returnerar `abx` (ersätt fullständig matchning).
`ReplaceRegExGroup ('abc123','([a-z])([0-9]+)','x',1)`	Returnerar `abx123` (ersätt endast den första gruppen i matchningen).
`ReplaceRegExGroup ('abc123','([a-z])([0-9]+)','x',2)`	Returnerar `abcx` (ersätt endast den andra gruppen i matchningen).

Exempel	Resultat
`ReplaceRegExGroup ('abc123abc123','([a-z])([0-9]+)','x',1,2)`	Returnerar `abc123abx123`. Det finns två matchningar mellan den första gruppen i inmatningstexten och regexmönstret. Det är dock bara den andra förekomsten som ersätts.
`ReplaceRegExGroup ('ABC123ABC123','([a-z])([0-9]+)','x',1)`	Returnerar `ABC123ABC123`. Med andra ord ersätts ingen text. Det här är eftersom funktionen är skiftlägeskänslig och den skiftlägesokänsliga varianten av funktionen inte används.
`ReplaceRegExGroupI ('ABC123ABC123','([a-z])([0-9]+)','x',1)`	Returnerar `ABx123ABx123`. I det här exemplet ersätts texten eftersom den skiftlägesokänsliga varianten av funktionen, **ReplaceRegExGroupI()**, används.

Användning

Du kan använda **ReplaceRegExGroup()** för att ändra text så att den uppfyller formaterings- och efterlevnadsstandarder, särskilt när du behöver identifiera komplexa mönster i dina data. Med det extra argumentet **group** för den här funktionen kan du uppdatera endast vissa delar av ett större textmönster. Du kan t.ex. uppdatera specifika delar av en URL.

Vid behov kan du också använda den här funktionen för att maskera känslig information, t.ex. personliga identitetsuppgifter (PII), så att den inte visas för användare som analyserar din applikation.

Exempel 1 – laddningsskript för att uppdatera domänen i URL:er

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell `URLs` som inledningsvis innehåller en lista med indatasträngar där vi vill identifiera och uppdatera specifika URL:er. Vi vill uppdatera domänen i dessa URL:er.
- Skapandet av ett fält, `ResourceURL_Updated`, som innehåller modifierad inmatningstext med de bearbetade URL:erna.
- En variabel`URL_RegEx` för att lagra det reguljära uttryck som används för att identifiera URL:er som ska bearbetas.

Våra krav:

- Varje indatasträng innehåller två URL:er, där resursens URL anges i den första URL:en. Vi vill inte uppdatera några andra URL:er än den första som anges.
- URL:en som ska uppdateras måste vara i formatet <https eller https>://<domain>.com/<resource path>. Resurssökvägen är valfri men används inte i det här exemplet.

Laddningsskript

```
Set URL_RegEx = '(https?):\/\/([a-zA-Z0-9]+)\.([a-zA-Z0-9]*)\.??([a-zA-Z0-9]*))\/?((([a-zA-
Z0-9]*)*)((\/?([a-zA-Z0-9]*)*))*)\/{0,1}([\.,;]+(?=\s))*?';

URLs:

Load
ReplaceRegExGroup(URLsList, '$(URL_RegEx)', 'replacement-server', 2,1) as ResourceURL_Updated;

Load * Inline `
URLsList
The resource is located at https://testserver.com/files. For more help, see:
https://support.company.com.
The resource is located at https://testserver.com/files/worksheet. For more help, see:
https://support.company.com.
The resource is located at https://testserver.com/files/book. For more help, see:
https://support.company.com.
The resource is located at https://testserver.com/files/form. For more help, see:
https://support.company.com.
The resource is located at https://testserver.com/datamodel. For more help, see:
https://support.company.com.
The resource is located at https://testserver.com/resourcenew. For more help, see:
https://support.company.com.
The resource is located at https://testserver.com/page. For more help, see:
https://support.company.com.
The resource is located at https://testserver.com/page/overview. For more help, see:
https://support.company.com.
` (delimiter is '\t');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `ResourceURL_Updated`

Resultattabell

ResourceURL_Updated
The resource is located at https://replacement-server/datamodel. For more help, see: https://support.company.com.
The resource is located at https://replacement-server/files. For more help, see: https://support.company.com.
The resource is located at https://replacement-server/files/book. For more help, see: https://support.company.com.
The resource is located at https://replacement-server/files/form. For more help, see: https://support.company.com.

ResourceURL_Updated
The resource is located at https://replacement-server/files/worksheet. For more help, see: https://support.company.com.
The resource is located at https://replacement-server/page. For more help, see: https://support.company.com.
The resource is located at https://replacement-server/page/overview. For more help, see: https://support.company.com.
The resource is located at https://replacement-server/resourcenew. For more help, see: https://support.company.com.

Exempel 2 - diagramuttryck för att uppdatera protokoll i URL:er

Diagramuttryck och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell `URLsList` som innehåller en lista med indatasträngar där vi vill identifiera och uppdatera specifika URL:er. Vi vill uppdatera protokollet från http till https i länkarna till supportwebbplatsen.

Våra krav:

- Varje indatasträng innehåller två URL:er, där länken till supportwebbplatsen visas som den andra URL:n. Vi vill inte uppdatera några andra URL:er än den andra som anges.
- URL:en som ska uppdateras måste vara i formatet <https eller https>://<domain>.com/<resource path>. Resurssökvägen är valfri men används inte i det här exemplet.

Laddningsskript

```
URLs:

Load * Inline `
URLsList
The resource is located at https://testserver.com/files. For more help, see:
http://support.company.com.
The resource is located at https://testserver.com/files/worksheet. For more help, see:
http://support.company.com.
The resource is located at https://testserver.com/files/book. For more help, see:
http://support.company.com.
The resource is located at https://testserver.com/files/form. For more help, see:
http://support.company.com.
The resource is located at https://testserver.com/datamodel. For more help, see:
http://support.company.com.
The resource is located at https://testserver.com/resourcenew. For more help, see:
http://support.company.com.
```

```
The resource is located at https://testserver.com/page. For more help, see:
http://support.company.com.
The resource is located at https://testserver.com/page/overview. For more help, see:
http://support.company.com.
` (delimiter is '\t');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- URLsList

Lägg till denna beräknade dimension:

```
=ReplaceRegExGroup(URLsList, '(https?):\/\/([[a-zA-Z0-9]+)\.([a-zA-Z0-9]*)\.??([a-zA-Z0-
9]*))\/?(([a-zA-Z0-9]*)*)((\/?([a-zA-Z0-9]*)*))*)\/{0,1}([\.,;]+(?=\s))*?', 'https', 1,2)
```

Resultattabell

URLsList	=ReplaceRegExGroup(URLsList, '(https?):\/\/([[a-zA-Z0-9]+)\.([a-zA-Z0-9]*)\.??([a-zA-Z0-9]*))\/?(([a-zA-Z0-9]*)*)((\/?([a-zA-Z0-9]*)*))*)\/{0,1}([\.,;]+(?=\s))*?', 'https', 1,2)
The resource is located at https://testserver.com/files. For more help, see: http://support.company.com.	The resource is located at https://testserver.com/datamodel. For more help, see: https://support.company.com.
The resource is located at https://testserver.com/files/worksheet. For more help, see: http://support.company.com.	The resource is located at https://testserver.com/files. For more help, see: https://support.company.com.
The resource is located at https://testserver.com/files/book. For more help, see: http://support.company.com.	The resource is located at https://testserver.com/files/book. For more help, see: https://support.company.com.
The resource is located at https://testserver.com/files/form. For more help, see: http://support.company.com.	The resource is located at https://testserver.com/files/form. For more help, see: https://support.company.com.
The resource is located at https://testserver.com/datamodel. For more help, see: http://support.company.com.	The resource is located at https://testserver.com/files/worksheet. For more help, see: https://support.company.com.
The resource is located at https://testserver.com/resourcenew. For more help, see: http://support.company.com.	The resource is located at https://testserver.com/page. For more help, see: https://support.company.com.

URLsList	=ReplaceRegExGroup(URLsList, '(https?):\/\/(([a-zA-Z0-9]+)\.([a-zA-Z0-9]*)\.??([a-zA-Z0-9]*))\/?((([a-zA-Z0-9]*)*)((\/?([a-zA-Z0-9]*)*))*)\/{0,1}([\.,;]+(?=\s))*?', 'https', 1,2)
The resource is located at https://testserver.com/page. For more help, see: http://support.company.com.	The resource is located at https://testserver.com/page/overview. For more help, see: https://support.company.com.
The resource is located at https://testserver.com/page/overview. For more help, see: http://support.company.com.	The resource is located at https://testserver.com/resourcenew. For more help, see: https://support.company.com.

Right

Right() returnerar en sträng som består av de sista (längst till höger) tecknen av indatasträngen, där antalet tecken fastställs av det andra argumentet.

Syntax:

```
Right(text, count)
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
count	Definierar det antal tecken som ska inkluderas från delen längst till höger i strängen **text**.

Exempel: diagramuttryck

Exempel	Resultat
`Right( 'abcdef', 3 )`	Returnerar `def`

Exempel – grunderna i Right

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:

- InputText
- NoOfChars

Laddningsskript

```
Example:
Load * inline [
InputText, NoOfChars
'abcdef', 3
'2021-07-14', 5
'2021-07-14', 2
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- InputText
- NoOfChars

Skapa följande beräknade dimension:

- =Right(InputText,NoOfChars), för att beräkna de tecken i InputText som måste extraheras från den högra sidan.

Resultattabell

InputText	AntalTecken	Höger(Indatatext,AntalTecken)
2021-07-14	5	07-14
abcdef	3	def
2021-07-14	2	14

Den första raden returnerar 07-14, eftersom detta är de 5 första tecknen från vänster i strängen InputText.

Den andra raden returnerar def, eftersom detta är de 3 första tecknen från höger i strängen InputText.

Den tredje raden returnerar 14, eftersom detta är de 2 första tecknen från höger i strängen InputText.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Example:
Load
*,
right(Text,NoOfChars) as Right;
Load * inline [
Text, NoOfChars
'abcdef', 3
'2021-07-14', 5
'2021-07-14', 2
];
```

Resultattabell

Text	AntalTecken	Höger
abcdef	3	def
2021-07-14	5	7-14
2021-07-14	2	14

Exempel – Right-scenario

Diagramuttryck

Översikt

I det här exemplet analyseras orderkoderna i en datauppsättning, som innehåller en regionkod och ordernummer, för att skapa ett nytt fält för dataanalys.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `OrderID`
  - `CustomerName`
  - `SalesAmount`

Laddningsskript

```
Example:
Load * inline [
OrderID, CustomerName, SalesAmount
US1000001, John Doe, 1500
EU3000002, Jane Smith, 2300
AS2000003, Mike Davis, 1800
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `OrderID`
- `CustomerName`
- `SalesAmount`

Skapa följande beräknade dimension:

- `=Right(OrderID, 7)` för att dela ut ordernumret.

Resultattabell

Order-ID	Kundnamn	Försäljningsbelopp	Höger (Order-ID, 7)
AS2000003	Mike Davis	1800	2000003
EU3000002	Jane Smith	2300	3000002
US1000001	John Doe	1500	1000001

Jämför utdata från `Right` -funktionen med `OrderID`-indata. Den beräknade dimensionen får ett numeriskt värde som nu kan användas för uppgifter som att sortera utdata i numerisk ordning i stället för alfabetisk.

RTrim

RTrim() returnerar strängen rensad på avslutande blankategstecken.

Syntax:

```
RTrim(text)
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Sträng som ska utvärderas.

Exempel: diagramuttryck

Exempel	Resultat
`RTrim( ' abc' )`	Returnerar `abc`
`RTrim( 'abc ' )`	Returnerar `abc`

Exempel – grunderna i RTrim

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `InputText`.

Laddningsskript

```
Set verbatim=1;
Example:
```

```
Load * inline [
InputText
'   abc    '
' def  '];
```

Satsen `Set verbatim=1` finns med i exemplet för att säkerställa att blankstegen inte rensas automatiskt före demonstration av `RTrim` -funktionen. Mer information finns i Verbatim (page 239).

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `InputText`

Skapa följande beräknade dimension:

- `=RTrim(InputText)` för att ta bort efterföljande blanksteg från `InputText`.

Resultattabell

InputText	RTrim(InputText)
” abc ”	” abc”
” def ”	” def”

Utdata från funktionen `RTrim` tog bort alla efterföljande blanksteg till höger om originaltexten men behöll alla inledande blanksteg.

Exempel – RTrim-scenario

Diagramuttryck

Översikt

Ett CRM-system (Customer Relationship Management) innehåller poster med inkonsekvent datainmatning som innehåller extra efterföljande blanksteg. För rapporteringsändamål måste uppgifterna rensas för att ta bort dessa blanksteg och för att säkerställa korrekt sortering och gruppering av kundnamn.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `CustomerName`.

Laddningsskript

```
Set verbatim=1;
Example:
Load * inline [
```

```
CustomerName
'John Doe  '
'Jane Smith'
'Michael Johnson  '
'Emily Davis'
];
```

> *Satsen "Set verbatim=1" finns med i exemplet för att säkerställa att utrymmena inte rensas automatiskt före demonstration av trim-funktionen. Mer information finns i Verbatim (page 239).*

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `CustomerName`

Skapa följande beräknade dimension:

- `=RTrim(CustomerName)` för att ta bort efterföljande blanksteg från `CustomerName`.

Resultattabell

Kundnamn	RTrim(CustomerName)
'Emily Davis'	'Emily Davis'
'Jane Smith'	'Jane Smith'
'John Doe '	'John Doe'
'Michael Johnson '	'Michael Johnson'

Utmatningen visar att funktionen `RTrim` tog bort alla efterföljande blanksteg från de ursprungliga strängvärdena i `CustomerName`.

Exempel – RTrim avancerat scenario

Diagramuttryck

Översikt

I detta exempel tas alla efterföljande blanksteg bort från den ursprungliga textsträngen. Diagramuttrycket innehåller åtgärder som använder funktionen `Len` för att räkna tecknen i strängen före och efter användning av funktionen `RTrim`.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `String`.

Laddningsskript

```
Set verbatim=1;
Example:
Load * inline [
String
'   abc    '
' def  '];
```

> *Satsen "Set verbatim=1" finns med i exemplet för att säkerställa att blankstegen inte rensas automatiskt före demonstration av `RTrim`-funktionen. Mer information finns i Verbatim (page 239).*

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `String`

Skapa följande beräknade dimension:

- `=RTrim(String)` för att ta bort extra efterföljande blanksteg.

Skapa följande mått:

- `=Len(String)`, för att räkna längden på den ursprungliga strängen.
- `=Len(RTrim(String))`, för att räkna längden på strängen efter att de efterföljande blankstegen har tagits bort.

Resultattabell

Sträng	HögerTrim(Sträng)	Len(Sträng)	Längd(HögerTrim (Sträng))
' abc '	' abc'	10	6
'def '	'def'	6	4

När du jämför resultatet från funktionen `RTrim` med de ursprungliga strängvärdena i skriptet kan du se hur alla efterföljande blanksteg har tagits bort.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Set verbatim=1;
Example:
Load *, len(RtrimString) as RtrimStringLength;
Load *, rtrim(String) as RtrimString;
Load *, len(String) as StringLength;
Load * Inline [
String
```

```
'   abc    '
' def  '];
```

Satsen "`Set verbatim=1`" finns med i exemplet för att säkerställa att blankstegen inte rensas automatiskt före demonstration av `rtrim`-funktionen. Mer information finns i Verbatim (page 239).

Resultattabell

Sträng	SträngLängd	HögerTrimSträng	HögerTrimSträngLängd
' abc '	10	' abc'	6
' def '	6	' def'	4

Se även:

p *LTrim (page 1795)*

SubField

Subfield() används för att extrahera delsträngkomponenter från ett överordnat strängfält där de ursprungliga postfälten består av två eller fler delar, som separeras av en avgränsare.

Subfield()-funktionen kan användas till exempel för att extrahera förnamn och efternamn från en lista av poster som innehåller fullständiga namn, komponentdelarna i ett sökvägsnamn eller för att extrahera data från kommaavgränsade tabeller.

Om du använder funktionen **Subfield()** i en **LOAD**-sats med den valbara parametern field_no utelämnad kommer en fullständig post genereras för varje delsträng. Om flera fält laddas med hjälp av **Subfield()** skapas de kartesiska produkterna för alla kombinationer.

Syntax:

```
SubField(text, delimiter[, field_no ])
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Den ursprungliga strängen. Detta kan vara en hårdkodad text, en variabel, en dollarteckensexpansion eller ett annat uttryck.
delimiter	Ett tecken med indata-**text** som delar in strängen i dess komponentdelar.

Argument	Beskrivning
field_no	Det valbara tredje argumentet är ett heltal som vilka av delsträngarna i den överordnade strängen **text** som ska returneras. Använd värdet 1 om du vill returnera den första delsträngen, 2 om du vill returnera den delsträngen och så vidare. • Om **field_no** är ett positivt värde extraheras understrängar från vänster till höger. • Om **field_no** är ett negativt värde extraheras understrängar från höger till vänster.

SubField() kan användas istället för att använda komplexa kombinationer av funktioner, som t.ex. Len(), Right(), Left(), Mid() och andra strängfunktioner.

Exempel: diagramuttryck

Exempel	Resultat
`SubField( 'abc;cde;efg', ';' ,2 )`	Returnerar `cde`
`SubField( Null, ';' ,1 )`	Returnerar en tom sträng
`SubField( ';', ';' ,1 )`	Returnerar en tom sträng
`SubField( '\Users\ext_jrb\Documents\Qlik\Sense\Apps;','\',-3 )`	Returnerar `Qlik`

Exempel – grunderna i SubField

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `Name`.

Laddningsskript

```
Example:
Load * inline [
Name
Dave Owen
Joe Tem
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `Name`

Skapa följande beräknade dimensioner:

- `=SubField(Name, ' ',1)` för att extrahera den första delsträngen som visas före mellanslagsavgränsaren `' '`.
- `=SubField(Name, ' ',-1)`, för att extrahera den första delsträngen som visas före mellanslagsavgränsaren `' '`, med början från höger om strängen.

Results table

Name	SubField(Name,' ',1)	SubField(Name,' ',-1)
Dave Owen	Dave	Owen
Joe Tem	Joe	Tem

`SubField()`-funktionen extraherar den första delsträngen i `Name` genom att ställa in `field_no` på 1. Eftersom värdet i `field_no` är positivt, används ordningen vänster till höger vid extrahering av delsträngen. Ett andra funktionsanrop extraherar den andra delsträngen genom att ställa in argumentet `field_no` på -1, vilket extraherar delsträngen i ordningen höger till vänster.

Laddningsskript

Laddningsskript

Öppna Skriptredigeraren och lägg till följande laddningsskript till ett nytt delavsnitt.

```
FullName:
LOAD * inline [
Name
'Dave Owen'
'Joe Tem'
];

SepNames:
Load Name,
SubField(Name, ' ',1) as FirstName,
SubField(Name, ' ',-1) as Surname
Resident FullName;
Drop Table FullName;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Name`
- `FirstName`
- `SurName`

Results table

Name	FirstName	SurName
Dave Owen	Dave	Owen
Joe Tem	Joe	Tem

Förklaring

`SubField`-funktionen extraherar den första delsträngen, `Name`, genom att ställa in `field_no` på 1. Eftersom värdet i `field_no` är positivt, används ordningen vänster till höger vid extrahering av delsträngen. Ett andra funktionsanrop extraherar den andra delsträngen genom att ställa in argumentet `field_no` på -1, vilket extraherar delsträngen i ordningen höger till vänster.

Exempel – SubField-scenario

Diagramuttryck

Översikt

En dataupsättning med produkter innehåller en produkttabell. Varje produkt i tabellen har ett taggfält som identifierar produktområdet. Produkter kan ha flera värden för taggfältet. Widget A har till exempel följande taggar: elektronik, prylar, hem. Taggvärdena separeras av pipetecknet (|) som avgränsare. Det här exemplet visar hur du extraherar specifika taggvärden med hjälp av `SubField`-funktionen.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `ProductID`
  - `ProductName`
  - `Tags`
  - `Sales`

Laddningsskript

```
Example:
Load  *
INLINE [
ProductID, ProductName, Tags, Sales
1, Widget A, Electronics|Gadgets|Home, 150
2, Widget B, Electronics|Accessories, 200
3, Widget C, Furniture|Home, 100
4, Widget D, Gadgets|Accessories, 250
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ProductID`
- `ProductName`
- `Tags`

Skapa följande beräknade dimensioner:

- `=SubField(Tags, '|',1)` för att ta ut den första posten i `Tags`-fältet.
- `=SubField(Tags, '|',2)`, för att ta ut den andra posten i `Tags`-fältet.
- `=SubField(Tags, '|',3)` för att ta ut den tredje posten i `Tags`-fältet.

Results table

ProductID	ProductName	Tags	SubField (Tags, '|', 1)	SubField (Tags, '|', 2)	SubField (Tags, '|', 3)
1	Widget A	Electronics|Gadgets|Home	Electronics	Gadgets	Home
2	Widget B	Electronics|Accessories	Electronics	Accessories	-
3	Widget C	Furniture|Home	Furniture	Home	-
4	Widget D	Gadgets|Accessories	Gadgets	Accessories	-

Utdata från `SubField`-funktionerna har framgångsrikt returnerat taggarna från de relevanta positionerna i den ursprungliga `Tags`-strängen.

Laddningsskript

Laddningsskript

Öppna Skriptredigeraren och lägg till följande laddningsskript till ett nytt delavsnitt.

```
LOAD DISTINCT
Instrument,
SubField(Player,',') as Player,
SubField(Project,',') as Project;

Load * inline [
Instrument|Player|Project
Guitar|Neil,Mike|Music,Video
Guitar|Neil|Music,OST
Synth|Neil,Jen|Music,Video,OST
Synth|Jo|Music
Guitar|Neil,Mike|Music,OST
] (delimiter is '|');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `Instrument`
- `Player`
- `Project`

Results table

Instrument	Player	Project
Guitar	Mike	Music
Guitar	Mike	Video
Guitar	Mike	OST
Guitar	Neil	Music
Guitar	Neil	Video
Guitar	Neil	OST
Synth	Jen	Music
Synth	Jen	Video
Synth	Jen	OST
Synth	Jo	Music
Synth	Neil	Music
Synth	Neil	Video
Synth	Neil	OST

Förklaring

Det här exemplet visar hur användning av flera förekomster av `Subfield()`-funktionen, där parametern `field_no` har utelämnats, inom samma `LOAD`-sats skapar kartesiska produkter av alla kombinationerna. Alternativet `DISTINCT` används för att undvika att skapa dubbletter av poster.

SubFieldRegEx

SubFieldRegEx() extraherar text från ett indatastränguttryck med hjälp av det angivna reguljära uttrycksmönstret som en avgränsare. Funktionen returnerar ett nullvärde om inga matchningar hittas.

Den här funktionen utför regexoperationer som är skiftlägeskänsliga. Alternativt kan du använda varianten **SubFieldRegExI()** för att utföra skiftlägesokänsliga regexoperationer.

Syntax:

```
SubFieldRegEx (text, regex_delimiter [, field_no])
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Indatasträngens text inom vilken du vill söka efter ett reguljärt uttrycksmönster.
regex_ delimiter	Stränguttryck som innehåller det reguljära uttryck som ska användas som avgränsare. Du kan ange mer än en avgränsare i samma reguljära uttrycksmönster genom att separera dem med ett |-tecken (lodstreck).
field_no	Heltalsvärde som anger numret på den underavdelning som du vill extrahera. Ange t.ex. värdet 4 för att få fram den fjärde underavdelningen. Det här är ett valfritt argument. Följande gäller oavsett om funktionen används i ett laddningsskript eller ett diagramuttryck: • Om **field_no** är positivt kommer funktionen att returnera ett värde som identifierar matchningar från vänster till höger. • Om **field_no** är negativt kommer funktionen att returnera ett värde som identifierar matchningar från höger till vänster. När du använder funktionen i ett laddningsskript: • Om du använder funktionen **SubFieldRegEx()** i en **LOAD**-sats och **field_no** utelämnas, genererar funktionen lika många poster som det finns matchningar. • Om flera fält laddas med hjälp av **SubFieldRegEx()** och inget av dem specificerar ett **field_no**-argument skapas de kartesiska produkterna för alla kombinationer. När funktionen används i ett diagram-uttryck: • Om **field_no** utelämnas är standardvärdet 1.

Exempel på funktioner

Exempel	Resultat
`SubFieldRegEx('a,b c;1 2,3',' |,|;', 1)`	Returnerar a. I det här exemplet anges tre avgränsningstecken (blanksteg, komma och semikolon).
`SubFieldRegEx ('a;b;c;1;2;3',';', 2)`	Returnerar b. I det här fallet anges endast ett avgränsningstecken (semikolon).
`SubFieldRegEx('a,b c;1 2,3',' |,|;', 4)`	Returnerar 1.

Användning

Du kan använda **SubFieldRegEx()** för att extrahera flera enskilda värden från en avgränsad lista. Om du t.ex. har en kommaseparerad lista som behållare i ett större block med naturligt språk kan du använda den här funktionen för att isolera varje enskilt värde från listan till en ny post i datamodellen.

Exempel 1 – laddningsskript för att separera taggar med transaktioner

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell `Transaction_By_Tag`, som inledningsvis innehåller transaktionsdata. Dessa data bearbetas för att extrahera enskilda taggvärden till nya poster.
- Skapandet av ett fält, `TransactionTag`, för att lagra de extraherade taggvärdena.

I fältet som innehåller transaktionstaggarna finns flera taggar i textsträngar, åtskilda av avgränsningstecken. Avgränsaren kan vara något av följande tecken: , | ;

Laddningsskript

```
Transaction_By_Tag:

Load SubFieldRegEx(Tags, ',|;|\|') as TransactionTag,
* Inline [
ID     Product Tags    Amount
1      Product A       Special order,Eligible for return,Warranty included     100.93
2      Product B       Ineligible for return|No warranty available     51.11
3      Product C       No warranty available   12.83
4      Product B       Special order;Liquidation sale;Warranty included        209.48
] (delimiter is '\t');

drop fields Tags;
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ID`
- `Product`
- `TransactionTag`

Resultattabell

ID	Product	TransactionTag
1	Product A	Eligible for return
1	Product A	Warranty included
1	Product A	Special order
2	Product B	No warranty available
2	Product B	Ineligible for return
3	Product C	No warranty available
4	Product B	Liquidation sale
4	Product B	Special order
4	Product B	Warranty included

Detta exempel visar att **SubFieldRegEx()**, när det används i ett laddningsskript, returnerar en separat rad för varje matchning som hittas i indatasträngen.

Exempel 2 – diagramuttryck för att separera taggar efter transaktion

Laddningsskript och resultat

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En tabell `Transaction_By_Tag`, som innehåller transaktionsdata.
- Ett fält, `Tags`, som innehåller textsträngar för att lagra alla taggar som är associerade med en transaktion.

Våra krav:

- I fältet som innehåller transaktionstaggarna finns flera taggar i textsträngar, åtskilda av avgränsningstecken. Avgränsaren kan vara något av följande tecken: , | ;
- Vi vill använda diagramuttryck för att separera varje tagg i sitt eget fält, med upp till tre taggar möjliga att lägga till i en transaktion.

Laddningsskript

```
Transaction_By_Tag:

Load * Inline [
ID     Product Tags    Amount
1      Product A       Special order, Eligible for return, Warranty included   100.93
2      Product B       Ineligible for return | No warranty available   51.11
3      Product C       No warranty available   12.83
```

```
4      Product B       Special order;Liquidation sale;Warranty included        209.48
] (delimiter is '\t');
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ID`
- `Tags`

Lägg till dess beräknade dimensioner:

- `=SubFieldRegEx(Tags, ',|;|\|', 1)`
- `=SubFieldRegEx(Tags, ',|;|\|', 2)`
- `=SubFieldRegEx(Tags, ',|;|\|', 3)`

Resultattabell

ID	Tags	=SubFieldRegEx (Tags, ',|;||', 1)	=SubFieldRegEx (Tags, ',|;||', 2)	=SubFieldRegEx (Tags, ',|;||', 3)
1	Special order, Eligible for return, Warranty included	Special order	Eligible for return	Warranty included
2	Ineligible for return | No warranty available	Ineligible for return	No warranty available	-
3	No warranty available	No warranty available	-	-
4	Special order;Liquidation sale;Warranty included	Special order	Liquidation sales	Warranty included

SubStringCount

SubstringCount() returnerar antalet förekomster av angiven delsträng i indatasträngtexten. Om det inte blir någon träff returneras 0.

Syntax:

```
SubStringCount(text, sub_string)
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
sub_string	En sträng som kan förekomma en eller flera gånger inom indatasträngen **text**.

Exempel: diagramuttryck

Exempel	Resultat
`SubStringCount( 'abcdefgcdxyz', 'cd' )`	Returnerar 2
`SubStringCount( 'abcdefgcdxyz', 'dc' )`	Returnerar 0

Exempel – grunderna SubStringCount

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `FruitSentence`.

Laddningsskript

```
Example:
Load * inline [
FruitSentence
"I love eating an apple every day.Apple juice is great too!"
"Apple pie is my favorite dessert. I also enjoy apple jam."
"There is an apple and a banana on the table."
"I don't like apples as much as oranges."
"An apple a day keeps the doctor away, but two apples are even better!"
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `FruitSentence`

Skapa följande beräknade dimension:

- `=SubStringCount(FruitSentence, 'apple')` för att beräkna hur många gånger ordet `apple` förekommer i `FruitSentence`.

Results table

FruitSentence	SubStringCount(FruitSentence, 'apple')
An apple a day keeps the doctor away, but two apples are even better!	2

FruitSentence	SubStringCount(FruitSentence, 'apple')
Apple pie is my favorite dessert. I also enjoy apple jam.	1
I don't like apples as much as oranges.	1
I love eating an apple every day. Apple juice is great too!	1
There is an apple and a banana on the table.	1

Tabellen är sorterad på dimensionen `FruitSentence` i stigande ordning. Funktionen `SubStringCount` returnerar antalet gånger som ordet `apple` förekommer i varje mening. I den första meningen returneras t.ex. `2`, eftersom ordet förekommer två gånger i meningen. I den andra och fjärde meningen ignorerar funktionen ordet `Apple` eftersom det inte matchar skiftläget för söktermen `apple`.

Om du vill göra en skiftlägesokänslig sökning kan du först konvertera indatasträngen till versaler eller gemener och sedan använda funktionen `SubStringCount` på resultatet. Om du till exempel vill söka efter alla förekomster av ordet apple, oavsett skiftläge, använder du följande måttuttryck: `=SubStringCount(Upper(FruitSentence),'APPLE')`. Uttrycket använder funktionen `SubStringCount` med funktionen `Upper` för att konvertera hela strängen till versaler, och söker sedan efter ordet `APPLE`.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript. Detta exempel innehåller en föregående laddning som använder funktionen `SubStringCount` med funktionen `Upper` för att konvertera hela strängen till versaler, och sedan söker efter ordet `APPLE`.

```
Example:
Load *,
SubStringCount(Upper(FruitSentence),'APPLE') as SubStringCount_APPLE;
Load * inline [
FruitSentence
"I love eating an apple every day. Apple juice is great too!"
"Apple pie is my favorite dessert. I also enjoy apple jam."
"There is an apple and a banana on the table."
"I don't like apples as much as oranges."
"An apple a day keeps the doctor away, but two apples are even better!"
];
```

Resultattabell

Fruktmening	AntalDelsträngar_ÄPPLE
I love eating an apple every day. Apple juice is great too!	2
Apple pie is my favorite dessert. I also enjoy apple jam.	2

Fruktmening	AntalDelsträngar_ÄPPLE
There is an apple and a banana on the table.	1
I don't like apples as much as oranges.	1
An apple a day keeps the doctor away, but two apples are even better!	2

Exempel – SubStringCount-scenario

Diagramuttryck

Översikt

En datamängd med fruktprodukter innehåller ett fält med produktbeskrivningar. Detta exempel beräknar antalet gånger som följande fruktnamn förekommer i varje beskrivning: äpple, banan eller apelsin.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `ProductDescription`.

Laddningsskript

```
Example:
Load * inline [
ProductDescription
"Fresh apple and banana smoothie."
"Organic apple, banana, and orange juice. Great for breakfast."
"A refreshing orange soda made with real orange juice."
"Banana chips with apple and cinnamon flavor."
"Delicious apple pie with a hint of cinnamon and vanilla."
"Tropical banana and orange mix for smoothies."
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `ProductDescription`

Skapa följande beräknade dimensioner:

- `=SubStringCount(ProductDescription, 'apple')` för att beräkna antalet förekomster av ordet `apple` i strängen `ProductDescription`.

- `=SubStringCount(ProductDescription, 'banana')` för att beräkna antalet förekomster av ordet `banana` i strängen `ProductDescription`.
- `=SubStringCount(ProductDescription, 'orange')` för att beräkna antalet förekomster av ordet `orange` i strängen `ProductDescription`.

Results table

ProductDescription	SubStringCount (ProductDescription, 'apple')	SubStringCount (ProductDescription, 'banana')	SubStringCount (ProductDescription, 'orange')
A refreshing orange soda made with real orange juice.	0	0	2
Banana chips with apple and cinnamon flavor.	1	0	0
Delicious apple pie with a hint of cinnamon and vanilla.	1	0	0
Fresh apple and banana smoothie.	1	1	0
Organic apple, banana, and orange juice. Great for breakfast.	1	1	1
Tropical banana and orange mix for smoothies.	0	1	1

Utdata från funktionen `SubStringCount` returnerar framgångsrikt antalet gånger som varje delsträng (`apple, banana, orange`) förekommer i produktbeskrivningen. I den första beskrivningen till exempel, returnerar funktionen 2 eftersom det finns två förekomster av ordet `orange`. Argumentet `sub_string` är skiftlägeskänsligt, så i den andra beskrivningen räknar funktionen inte ordet `Banana` eftersom det inte matchar delsträngen `banana` i mätuttrycket.

Om du vill göra en skiftlägesokänslig sökning kan du först konvertera indatasträngen till versaler eller gemener och sedan använda funktionen `SubStringCount` på resultatet. Om du till exempel vill söka efter alla förekomster av ordet apple, oavsett skiftläge, använder du följande måttuttryck: `=SubStringCount(Upper(ProductDescription),'APPLE')`. Uttrycket använder funktionen `SubStringCount` med funktionen `Upper` för att konvertera hela strängen till versaler, och söker sedan efter ordet `APPLE`.

Exempel – SubStringCount-scenario parsning av nästlade data

Diagramuttryck

Översikt

En datauppsättning med it-poster innehåller en lista med id-nummer och kategorier för ändringar. I datatabellen härleds fältet `Categories` från en hierarki på flera nivåer som listar den översta överordnade kategorin följt av alla underkategorier, till exempel Software > Program > Patch. Det här scenariot visar hur man plockar ut det sista värdet, eller bladvärdet, från den här ojämna kategorihierarkin med hjälp av en kombination av funktionerna `SubStringCount` och `SubField`.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
    - `ChangeID` innehåller det ursprungliga ändrings-ID som ska behandlas.
    - `Categories` innehåller en lista över ändringskategorier. Kategorierna skiljs åt med semikolon och listas i hierarkisk ordning från vänster till höger, från den översta överordnade kategorin till den mest detaljerade underkategorin.

Laddningsskript

```
Example:
Load * inline [
ChangeID,Categories
CHG00100,Software;Program;Patch
CHG00101,Hardware;Server
CHG00102,Hardware;Network;Router
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ChangeID`
- `Categories`

Skapa följande beräknade dimensioner:

- `=SubStringCount(Categories,';')+1`för att beräkna antalet kategorinivåer i förändringshierarkin.
- `=SubField(Categories, ';' ,SubStringCount(Categories,';')+1)` för att extrahera kategorin längst till höger i förändringshierarkin.

Results table

ChangeID	Categories	SubStringCount (Categories,';')+1	SubField(Categories, ';' ,SubStringCount (Categories,';')+1)
CHG00100	Software;Program;Patch	3	Patch
CHG00101	Hardware;Server	2	Server
CHG00102	Hardware;Network;Router	3	Router

I det här exemplet nästlas `SubStringCount`-funktionen inuti `SubField`-funktionen. Den första beräknade dimensionen, `SubStringCount()`, beräknar antalet nästlade värden i fältet `Categories`. Detta bestäms genom att räkna antalet semikolon `';'` och lägga till 1 till resultatet. Denna utdata infogas sedan som den tredje parametern i funktionen `SubField` för att extrahera kategorin längst till höger i `Categories`-fältet.

Granska till exempel resultaten för `ChangeID`-posten `CHG00100`. I den första beräknade dimensionen är `SubStringCount` 2. Uttrycket lägger sedan till 1 till det resultatet för att fastställa att fältet `Categories` har tre nästlade kategorivärden, `Software`, `Program` och `Patch`. Funktionen `SubField` i den andra beräknade dimensionen, använder sedan detta resultat för att extrahera den tredje kategorin, `Patch`.

TextBetween

TextBetween() returnerar texten i indatasträngen som finns mellan de tecken som definierats som avgränsare.

Syntax:

```
TextBetween(text, delimiter1, delimiter2[, n])
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
delimiter1	Anger det första avgränsartecknet (eller strängen) som ska sökas efter i **text**.
delimiter2	Anger det andra avgränsartecknet (eller strängen) som ska sökas efter i **text**.
n	Definierar vilken förekomst av avgränsarparet som ska sökas emellan. Till exempel returnerar värdet 2 tecknen mellan den andra förekomsten av **delimiter1** och den andra förekomsten av **delimiter2**.

Exempel: diagramuttryck

Exempel	Resultat
`TextBetween( '<abc>', '<', '>' )`	Returnerar `abc`
`TextBetween( '<abc><de>', '<', '>',2 )`	Returnerar `de`

Exempel	Resultat
`TextBetween( 'abc', '<', '>' )`	Returnerar `NULL` Om någon av avgränsarna inte hittas i strängen returneras `NULL`.
`TextBetween( '<a<b', '<', '>' )`	Returnerar `NULL` Om någon av avgränsarna inte hittas i strängen returneras `NULL`.
`TextBetween( '<>', '<', '>' )`	Returnerar en sträng med längden noll.
`TextBetween( '<abc>', '<', '>', 2 )`	Returnerar `NULL`, eftersom n är större än antalet förekomster av avgränsarna.

Exempel – grunderna i TextBetween

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `InputText`.

Laddningsskript

```
Example:
Load * inline [
InputText,
<abc><de>
<def><ghi><jk>
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `InputText`

Skapa följande beräknade dimensioner:

- `=TextBetween(InputText,'<','>')` för att beräkna den första förekomsten av de tecken som visas mellan avgränsarna `'<'` och `'>'` i fältet `InputText`.
- `=TextBetween(InputText,'<','>',2)` för att beräkna den andra förekomsten av de tecken som visas mellan avgränsarna `'<'` och `'>'` i fältet `InputText`.

Results table

InputText	TextBetween(InputText,'<','>')	TextBetween(InputText,'<','>',2)
<abc><de>	abc	de
<def><ghi><jkl>	def	ghi

I den första beräknade dimensionen returnerar utdatan från `TextBetween`-funktionen den första förekomsten av de tecken som visas mellan avgränsarna `'<'` och `'>'`. Till exempel, rad 1 returnerar `abc`. Den andra beräknade dimensionen har ett `n`-argument på `2`, så utdatan returnerar den andra förekomsten av de tecken som visas mellan avgränsarna `'<'` och `'>'`. Till exempel, rad 1 returnerar `de`.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Load *,
textbetween(Text,'<','>') as TextBetween,
textbetween(Text,'<','>',2) as SecondTextBetween;
Load * inline [
Text
<abc><de>
<def><ghi><jkl>];
```

Resultattabell

Text	TextBetween	SecondTextBetween
<abc><de>	abc	de
<def><ghi><jkl>	def	ghi

Exempel – TextBetween-scenario

Diagramuttryck

Översikt

En datauppsättning innehåller en datatabell med e-postadresser. I det här exemplet används funktionen `TextBetween` för att extrahera andra nivåns domän från adressen.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Följande fält i datatabellen:
  - `UserID`
  - `EmailAddress`

Laddningsskript

```
Example:
Load * inline [
```

```
UserID, EmailAddress
1, alice@example.com
2, bob@sample.org
3, charlie@domain.net
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `UserID`
- `EmailAddress`

Skapa följande beräknade dimension:

- `=TextBetween(EmailAddress,'@', '.')` för att returnera texten mellan tecknen `'@'` och `'.'` inom fältet `EmailAddress`.

Results table

UserID	EmailAddress	TextBetween(EmailAddress,'@', '.')
1	alice@example.com	example
2	bob@sample.org	sample
3	charlie@domain.net	domain

Utdata från `TextBetween`-funktionen returnerar andranivå-domänen för varje användar-ID genom att söka efter innehåll mellan de angivna avgränsarna: `'@'` och `'.'`.

Trim

Trim() returnerar strängen rensad på inledande och avslutande blankstegstecken.

Syntax:

```
Trim(text)
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Sträng som ska utvärderas.

Exempel: diagramuttryck

Exempel	Resultat
`Trim( ' abc' )`	Returnerar `abc`
`Trim( 'abc ' )`	Returnerar `abc`
`Trim( ' abc ' )`	Returnerar `abc`

Exempel – grunderna i Trim

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `InputText`.

Laddningsskript

```
Set verbatim=1;
Example:
Load * inline [
InputText
'   abc    '
' def  '];
```

> *Satsen ”Set verbatim=1” finns med i exemplet för att säkerställa att utrymmena inte rensas automatiskt före demonstration av trim-funktionen. Mer information finns i Verbatim (page 239).*

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `InputText`

Skapa följande beräknade dimension:

- `=Trim(InputText)` för att ta bort extra inledande eller efterföljande blanksteg från `InputText`.

Resultattabell

Indatatext	Trimma(Indatatext)
” abc ”	”abc”
” def ”	”def”

Utdata från `Trim`-funktionen tar bort alla inledande och efterföljande blanksteg.

Exempel – Trim-scenario

Diagramuttryck

Översikt

Ett CRM-system (Customer Relationship Management) innehåller poster med inkonsekvent datainmatning som innehåller extra inledande blanksteg. För rapporteringsändamål måste uppgifterna rensas för att ta bort dessa blanksteg och för att säkerställa korrekt sortering och gruppering av kundnamn.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `CustomerName`.

Laddningsskript

```
Set verbatim=1;
Example:
Load * inline [
CustomerName
'  John Doe '
'Jane Smith '
'   Michael Johnson'
'Emily Davis'
];
```

> *Satsen "Set verbatim=1" finns med i exemplet för att säkerställa att utrymmena inte rensas automatiskt före demonstration av trim-funktionen. Mer information finns i Verbatim (page 239).*

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `CustomerName`

Skapa följande beräknade dimension:

- `=Trim(CustomerName)` för att ta bort extra inledande eller efterföljande blanksteg från `CustomerName`.

Resultattabell

Kundnamn	Trim(Kundnamn)
' Michael Johnson'	'Michael Johnson'
' John Doe '	'John Doe'

Kundnamn	Trim(Kundnamn)
'Emily Davis'	'Emily Davis'
'Jane Smith '	'Jane Smith'

Utdata visar att `Trim`-funktionen tog bort alla inledande och efterföljande blanksteg från de ursprungliga strängvärdena i `CustomerName`.

Exempel – Trim avancerat scenario

Diagramuttryck

Översikt

I detta exempel tas alla inledande och efterföljande blanksteg bort från den ursprungliga textsträngen. Diagramuttrycket innehåller åtgärder som använder funktionen `Len` för att räkna tecknen i strängen före och efter användning av funktionen `Trim`.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

- En datauppsättning som läses in i en datatabell som heter `Example`.
- Ett fält i datatabellen heter `String`.

Laddningsskript

```
Set verbatim=1;
Example:
Load * inline [
String
'   abc    '
' def  '];
```

Satsen ”Set verbatim=1” finns med i exemplet för att säkerställa att utrymmena inte rensas automatiskt före demonstration av trim-funktionen. Mer information finns i Verbatim (page 239).

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `String`

Skapa följande beräknade dimension:

- `=Trim(String)` för att ta bort extra inledande eller efterföljande blanksteg

Skapa följande mått:

- `=Len(String)` för att beräkna längden på den ursprungliga strängen
- `=Len(Trim(String))` för att räkna längden på strängen efter att blankstegen har tagits bort.

Resultattabell

Sträng	Trim(Sträng)	Len(Sträng)	Len(Trim(Sträng))
' abc '	'abc'	10	3
' def '	'def'	6	3

När du jämför resultatet från `Trim`-funktionen med de ursprungliga strängvärdena i skriptet kan du se hur alla inledande och efterföljande blanksteg har tagits bort.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Set verbatim=1;
Example:
Load *, len(TrimString) as TrimStringLength;
Load *, trim(String) as TrimString;
Load *, len(String) as StringLength;
Load * inline [
String
'   abc    '
' def  '](delimiter is '\t');
```

Satsen ”Set verbatim=1” finns med i exemplet för att säkerställa att utrymmena inte rensas automatiskt före demonstration av trim-funktionen. Mer information finns i Verbatim (page 239).

Resultattabell

Sträng	SträngLängd	Trimma sträng	Trimma stränglängd
abc	10	abc	3
def	6	”def”	3

Upper

Upper() konverterar alla tecken i indatasträngen till versaler för alla texttecken i uttrycket. Tal och symboler ignoreras.

Syntax:

```
Upper(text)
```

Returnerad datatyp: sträng

Argument

Argument	Beskrivning
text	Sträng som ska utvärderas.

Exempel: diagramuttryck

Exempel	Resultat
`Upper( ' abcD' )`	Returnerar `ABCD`

Exempel – grunderna i Upper

Diagramuttryck

Översikt

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

En datauppsättning som läses in i en datatabell som heter `Example`.

- Ett fält i datatabellen heter `InputText`.

Laddningsskript

```
Example:
Load * inline [
InputText
rHode iSland
washingTon d.C.
new york
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till det här fältet som en dimension:

- `InputText`

Skapa följande beräknade dimension:

- `=Upper(InputText)` för att konvertera alla tecken i fältet `InputText` till versaler.

Resultattabell

InputText	Versal(Indatatext)
New York	New York
Rhode Island	Rhode Island
washingTon d.C.	WASHINGTON D.C.

Utdata från `Upper`-funktionen konverterar alla värden till versaler.

Laddningsskript

Följande kod visar hur du använder funktionen i ett laddningsskript.

```
Load
String,Upper(String)
Inline
[String
rHode iSland
washingTon d.C.
new york];
```

Resultattabell

Sträng	Övre(String)
rHode iSland	RHODE ISLAND
washingTon d.C.	WASHINGTON D.C.
new york	NEW YORK

Exempel 6 – övre scenario

Diagramuttryck

Översikt

En datauppsättning innehåller ISO-landskoder och landsnamn i olika format, t.ex. alla gemener, alla versaler eller blandade skiftlägen. Målet är att rensa datan så att varje landskod och namn formateras så att versaler används.

Öppna dataladdningsredigeraren och lägg till laddningsskriptet nedan i ett nytt delavsnitt.

Laddningsskriptet innehåller:

En datauppsättning som läses in i en datatabell som heter `Example`.

- Följande fält i datatabellen:
  - `ISO CountryCode`
  - `CountryName`

Laddningsskript

```
Country:
Load  * inline [
ISO_CountryCode, CountryName
gbr, United kingdom
Ind, india
CAN, Canada
Are, United arab emirates
];
```

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ISO_CountryCode`
- CountryName

Skapa följande beräknade dimensioner:

- `=Upper(ISO_CountryCode)` för att skapa en kolumn med landskoder som enbart använder versaler.
- `=Upper(CountryName)` för att skapa en kolumn med landsnamn som enbart använder versaler.

Resultattabell

ISO_ CountryCode	CountryName	Upper(ISO_Landskod)	Upper(LandetsNamn)
Är	Förenade Arabemiraten	ARE	Förenade Arabemiraten
KAN	Kanada	KAN	Kanada
GBR	Storbritannien	GBR	STORBRITANNIEN
Indien	indien	IND	INDIEN

`Upper`-funktionen har konverterat alla landskoder och landsnamn till att använda versaler.

Exempel – avancerat Upper-scenario

Diagramuttryck

Översikt

I det här exemplet används samma datauppsättning som i det föregående scenariot, men landsnamn och landskod kombineras och strängarna formateras med hjälp av funktionerna `Capitalize` och `Upper`.

Resultat

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- `ISO_CountryCode`
- `CountryName`

Skapa följande beräknade dimension:

- `=Capitalize(CountryName) & ' (' & Upper(ISO_CountryCode) & ')'` för att skapa en ny dimension som konkatenerar `CountryName` och `ISO_CountryCode` och formaterar landsnamnen så att de använder inledande versal och landskoderna så att de använder versaler.

Resultattabell

ISO_CountryCode	CountryName	Capitalize(CountryName) & '(' & Upper(ISO_CountryCode) & ')'
Är	Förenade Arabemiraten	Förenade Arabemiraten (ARE)

ISO_CountryCode	CountryName	Capitalize(CountryName) & '(' & Upper(ISO_CountryCode) & ')'
CAN	Kanada	Kanada (CAN)
gbr	Storbritannien	Storbritannien (GBR)
Ind	Indien	Indien (IND)

8.25 Systemfunktioner

Systemfunktioner tillhandahåller funktioner för att komma åt egenskaperna för system, enhet och Qlik Sense applikation.

Översikt av systemfunktioner

En del av funktionerna beskrivs mer ingående efter översikten. För de här funktionerna kan du klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Upphovsperson()

Den här funktionen returnerar en textsträng som anger egenskapen upphovsman för den aktuella applikationen. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

> *Egenskapen upphovsman kan inte anges i den aktuella versionen av Qlik Sense. Om du migrerar ett QlikView-dokument kvarstår egenskapen upphovsman.*

BeräknaMått()

När de används i ett dimensionsuttryck åsidosätter **CalcDim()** motorns automatiska tolkning av hur uttrycket hanteras internt. Specifikt förhindrar funktionen att ett nytt internt fält skapas i datamodellen. Istället behandlas urval som görs i applikationen som urval i tillämpliga fält från den ursprungliga datamodellen.

```
CalcDim - diagramfunktion(expr)
```

Klientplattform()

Den här funktionen returnerar användaragentsträngen i klientwebbläsaren. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

```
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/35.0.1916.114 Safari/537.36
```

Datornamn

Denna funktion returnerar en sträng som anger namnet på aktuell dator enligt operativsystemet. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

> *Om datorns namn består av fler än 15 tecken ingår bara de 15 första i strängen.*

```
ComputerName( )
```

Dokumentnamn

Den här funktionen returnerar en textsträng som anger namnet på den aktuella Qlik Sense applikationen, utan sökväg men med filtillägg. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

```
DocumentName( )
```

Dokumentsökväg

Denna funktion returnerar en sträng som innehåller den fullständiga sökvägen till den aktuella Qlik Sense applikationen. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

```
DocumentPath( )
```

Funktionen stöds inte i standardläget. .

Dokumenttitel

Den här funktionen returnerar en sträng som innehåller titeln på den aktuella Qlik Sense applikationen. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

```
DocumentTitle( )
```

Motorversion

Denna funktion returnerar komplett information om Qlik Sense-motorversionen i form av en sträng.

```
EngineVersion ()
```

HämtaKollationsspråk

Den här skriptfunktionen returnerar kulturnamnet för den sorteringsspråkvariant som används. Om variabeln CollationLocale inte har ställts in returneras användarens faktiska datorspråkvariant.

```
GetCollationLocale( )
```

HämtaObjektfält

GetObjectField() returnerar -fältuttrycket för dimensionen. **Index** är ett valfritt heltal som anger den dimension som ska returneras.

```
GetObjectField - diagramfunktion([index])
```

GetSysAttr

Funktionen **GetSysAttr()** används för att returnera information om klientorganisationen och utrymmet där en applikation är placerad. Med funktionen **GetSysAttr()** kan du utvärdera klientorganisationens ID och namn, utrymmets ID och namn, och andra detaljer om miljön där applikationen finns. Funktionen förenklar identifiering av utrymme och klientorganisation för en Qlik Sense applikation som kan ha flyttats mellan olika miljöer. Funktionen gör det enkelt att identifiera den specifika miljö som du arbetar i. Det är speciellt praktiskt under stadierna utveckling, test, acceptans och produktion (DTAP) i en applikations livscykel.

```
GetSysAttr (name)
```

Om du använder funktionen i För klienthanterat Qlik Sense kommer den bara att returnera tomma datavärden.

HämtaAnvändarAttribut

GetUserAttr()-funktionen returnerar information för den användare som använder en applikation. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

```
HämtaAnvändarAttribut (name)
```

Om du använder funktionen i För klienthanterat Qlik Sense kommer den bara att returnera tomma datavärden.

Gruppdimensionsindex

Den här funktionen returnerar indexet för det aktiva fältet för den angivna cykliska dimensionen. Det första fältet har ett indexvärde på 1. Ange namnen på den cykliska dimensionen i enkla citattecken. Den kan bara användas i ett diagramuttryck.

```
GroupDimensionIndex(dim_name Expression)
```

Gruppdimensionsetikett

Den här funktionen returnerar fältetiketten för det aktuella steget i den angivna cykliska dimensionen. Ange namnen på den cykliska dimensionen i enkla citattecken. Den kan bara användas i ett diagramuttryck.

```
GroupDimensionLabel(dim_name Expression)
```

ÄrPartiellOmladdning

Den här funktionen returnerar -1 (True) om den aktuella återinläsningen är partiell, annars 0 (False).

```
IsPartialReload ()
```

InObject

Diagramfunktionen **InObject()** utvärderar huruvida det aktuella objektet innehålls inuti ett annat objekt med det ID som anges i funktionsargumentet. Objektet kan vara ett ark eller en visualisering.

```
InObject - diagramfunktion(id_str)
```

Objekt-ID

Diagramfunktionen **ObjectId()** returnerar ID:t för det objekt som uttrycket utvärderas för. Funktionen har ett valfritt argument som anger vilken typ av objekt funktionen gäller. Objektet kan vara ett ark eller en visualisering. Denna funktion är bara tillgänglig i diagramuttryck.

```
ObjectId - diagramfunktion([object_type_str])
```

OSUser

Den här funktionen returnerar en sträng som innehåller namnet på den användare som är kopplad för närvarande. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

```
OSUser( )
```

I Qlik Sense Desktop och Qlik Sense Client-Managed mobil returnerar den här funktionen alltid "Personal\Me".

Produktversion

Denna funktion returnerar komplett information om Qlik Sense-version och byggnummer i form av en sträng.

Funktionen är utfasad och ersatt med **EngineVersion()**.

```
ProductVersion ()
```

Omladdningstid

Den här funktionen returnerar en tidsmarkör för när dataladdningen senast avslutades. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

```
ReloadTime( )
```

Delstatsnamn

StateName() returnerar namnet på det alternativa tillståndet för visualiseringen där den används. StateName kan till exempel användas för att skapa visualiseringar med text och färger som är dynamiska för att återspegla när tillståndet för en visualisering ändras. Denna funktion kan användas i diagramuttryck men kan inte användas för att fastställa det tillstånd som uttrycket syftar på.

```
StateName - diagramfunktion()
```

CalcDim - diagramfunktion

När de används i ett dimensionsuttryck åsidosätter **CalcDim()** motorns automatiska tolkning av hur uttrycket hanteras internt. Specifikt förhindrar funktionen att ett nytt internt fält skapas i datamodellen. Istället behandlas urval som görs i applikationen som urval i tillämpliga fält från den ursprungliga datamodellen.

Denna funktion kan inte användas iladdningsskriptet. Dessutom är en bästa praxis för systemprestanda att ladda de fält som används i den beräknade dimensionen som logiska fält via laddningsskriptet när det är möjligt. **CalcDim()** erbjuder ett sätt att styra motorns beteende i scenarier där det inte är möjligt att ladda det beräknade fältet från skriptet.

I Qlik Sense realiseras dimensionuttrycket som standard som ett fält-i-farten – det vill säga som ett faktiskt fält, synligt i det aktuella urvalsfältet, byggt på begäran. Urval tillämpas och härleds till detta fält, liknande alla logiska fält från skriptet.

Ett uttryck som är omslutet av **CalcDim()** kommer att bete sig på samma sätt som alla beräknade dimensionuttryck fungerar i QlikView, vilket är en ad hoc-mappning av de beräknade värdena till de underliggande källfältvärdena. Alla fält-i-farten-beräkningar kan överföras till beräknade dimensioner med hjälp av funktionen **CalcDim()** eftersom den beräknade dimensionen måste uppfylla färre krav baserat på

datamodellen än motsvarande fält-i-farten-beräkning. Denna omvandling kan vara önskvärd för att garantera ett liknande beteende för alla beräknade dimensioner där datamodellen i vissa fall kan förhindra användningen av en fält-i-farten-beräkning, men inte i andra.
≤ Beräknade fält

```
CalcDim(expr)
```

Returnerad datatyp: Dual

Argument

Argument	Beskrivning
expr	Ett uttryck som används som en dimension, eller som ett segment inom ett uttryck i en dimension.

Användning

CalcDim() är användbart när du behöver mer kontroll över hur Qlik-motorn hanterar fälten du anropar i ett dimension-uttryck.

Som standard bestämmer Qlik Sense automatiskt om interna fält ska skapas och refereras i datamodellen när du använder ett anpassat dimension-uttryck. Qlik Sense skapar och använder dessa interna fält närhelst det är möjligt. När detta inte är möjligt hanteras uttrycket med hänvisning till de befintliga fält i datamodellen som det är beroende av.

Beroende på ditt användningsområde kanske denna automatiska behandling inte alltid är önskvärd. Om du till exempel använder en variabel i ett dimension-uttryck kan den automatiska skapandet av nya fält bli problematiskt. I det här fallet använder du **CalcDim()** för att tvinga uttrycket att hanteras utan denna metod.

Här är några vanliga användningsfall:

- Du har migrerat QlikView-innehåll till Qlik Sense och du vill bevara QlikView-bearbetningsmetoden för beräknade dimensioner.
- Du vill förenkla urvalsbeteendet i din applikation. Du föredrar till exempel att visa fältval som val i fält från den ursprungliga datamodellen, snarare än val i nya fält.
- Ditt användningsfall kräver ett ad hoc-uttryck för konsekvens och prestanda. Till exempel, om ditt dimensionuttryck innehåller användarkontrollerbara dynamiska visualiseringsparametrar, såsom kontrollerbar formatering, avskärningsvärden eller perioder.

Exempel på funktioner

Exempel	Resultat
`=CalcDim(if(Year < 2023, 'Older', Year))`	Om du gör ett val i en visualisering som använder den här dimensionen kommer det att behandlas och visas som ett val i det associerade *Year*-fältvärdet från datamodellen. Omvänt, låt oss säga att du inte bäddar in uttrycket i **CalcDim()**. Då skulle det se ut så här: `=if(Year < 2023, 'Older', Year)` Att göra ett val i ovanstående uttryck skulle bearbetas och visas som ett val i det interna fältet *=if(Year < 2023, 'Older', Year)*, vilket inte finns i datamodellen.

Exempel 1 – Enkel jämförelse

Diagramuttryck och resultat

Översikt

I det här exemplet kommer du att:

- Lägg till en inline-tabell och läs in data i en applikation.
- Skapa en tabell för att visualisera data och utföra beräkningar.
- Gör val i tabellen för att observera skillnaden i beteende.

Öppna Skriptredigeraren och lägg till följande laddningsskript till ett nytt delavsnitt.

Laddningsskriptet laddar en inline-tabell. Den här tabellen innehåller två fält: ett fält för personens förnamn och ett fält för personens efternamn.

Laddningsskript

```
Names:

Load * inline [
FirstName,LastName
John,Doe
Amanda,Honda
Amalia,Craig
Cart,Lynch
];
```

Tabellskapande

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- *FirstName*
- *LastName*

Lägg sedan till vart och ett av följande uttryck som dimensioner i tabellen:

- *=FirstName & ' ' & LastName*
- *=CalcDim(FirstName & ' ' & LastName)*

Tabellen ska se ut på följande sätt.

Qlik Sense-tabell med exempel på resultat

FirstName	LastName	=FirstName & ' ' & LastName	=CalcDim(FirstName & ' ' & LastName)
John	Doe	John Doe	John Doe
Amanda	Honda	Amanda Honda	Amanda Honda
Amalia	Craig	Amalia Craig	Amalia Craig
Cart	Lynch	Cart Lynch	Cart Lynch

Jämförelse: Göra val

I kolumnen *=FirstName & ' ' & LastName* i tabellen väljer du värdet *JohnDoe*. Observera att i verktygsfältet för urval väljs värdet som ett värde från ett fält som heter *=FirstName' 'LastName*.

Rensa urvalet.

I kolumnen *=CalcDim(FirstName ' ' LastName)* i tabellen väljer du värdet *JohnDoe*. I urvalsverktygsfältet väljs värdet som värden från två olika fält: *FirstName* och *LastName*.

Dessa två uttryck presenterar samma information: en konkatenering av fälten *FirstName* och *LastName*. Skillnaden är att utan **CalcDim()** skapas ett syntetiskt fält för att lagra beräkningen. Med **CalcDim()** används de ursprungliga fälten från datamodellen som grund för den beräknade dimensionen.

Exempel 2 – Praktiskt användningsfall för att ändra standardbeteendet

Diagramuttryck och resultat

Översikt

I det här exemplet skapar vi dynamiska uttryck för dimensionen som refererar till en variabel, i stället för referenser till statiska värden.

I det här exemplet kommer du att:

- Lägg till en inline-tabell och läs in data i applikationen.
- Skapa en variabel för att dynamiskt lagra ett värde.
- Skapa en tabell för att visualisera data och utföra beräkningar.
- Gör val i tabellen för att observera skillnaden i beteende.

Öppna Skriptredigeraren och lägg till följande laddningsskript till ett nytt delavsnitt.

Laddningsskriptet laddar en inline-tabell. Denna tabell innehåller ett datumfält och ett beräknat fält, *Year*, som härleds från rådatumen.

Laddningsskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:

Load *,

RowNo() as ID,
Year(date) as Year
;

Load * Inline
[
date,amount
'1/19/2022',37.23
'2/28/2022',88.27
'3/16/2022',53.80
'4/1/2022',82.06
'5/16/2022',87.21
'6/26/2022',45.89
'7/9/2022',36.23
'8/2/2022',76.11
'9/26/2022',84.21
'10/29/2022',67.67
'1/1/2023',100.49
'2/24/2023',99.47
'3/6/2023',50.02
'4/10/2023',2.05
'5/20/2023',75.64
'6/25/2023',98.11
'7/19/2023',30.00
'9/27/2023',76.11
'10/26/2023',43.21
'11/12/2023',135.41
'1/15/2024',67.24
];
```

Skapande av variabel

Ladda data och öppna ett ark. Skapa en variabel med namnet *vCutoffYear*, och ge den följande **definition**: *2023*

Tabellskapande

Ladda data och öppna ett ark. Skapa en ny tabell och lägg till dessa fält som dimensioner:

- *date*
- *Year*

- *amount*

Lägg sedan till vart och ett av följande uttryck som dimensioner i tabellen:

- *=if(Year $(vCutoffYear), 'Older', Year)*
- *=CalcDim(if(Year $(vCutoffYear), 'Older', Year))*

Tabellen ska se ut på följande sätt.

Qlik Sense-tabell med exempel på resultat

date	Year	amount	if(Year $(vCutoffYear), 'Older', Year)	=CalcDim(if(Year $(vCutoffYear), 'Older', Year))
1/19/2022	2022	37.23	Older	Older
2/28/2022	2022	88.27	Older	Older
3/16/2022	2022	53.80	Older	Older
4/1/2022	2022	82.06	Older	Older
5/16/2022	2022	87.21	Older	Older
6/26/2022	2022	45.89	Older	Older
7/9/2022	2022	36.23	Older	Older
8/2/2022	2022	76.11	Older	Older
9/26/2022	2022	84.21	Older	Older
10/29/2022	2022	67.67	Older	Older
1/1/2023	2023	100.49	2023	2023
2/24/2023	2023	99.47	2023	2023
3/6/2023	2023	50.02	2023	2023
4/10/2023	2023	2.05	2023	2023
5/20/2023	2023	75.64	2023	2023
6/25/2023	2023	98.11	2023	2023
7/19/2023	2023	30.00	2023	2023
9/27/2023	2023	76.11	2023	2023
10/26/2023	2023	43.21	2023	2023
11/12/2023	2023	135.41	2023	2023
1/15/2024	2024	67.24	2024	2024

Jämförelser, del 1: Beteende utan CalcDim()

I tabellen ovan är det ingen skillnad mellan de två kolumnerna med beräknad dimension.

I kolumnen *if(Year $(vCutoffYear), 'Older', Year)* i tabellen väljer du värdet *2023*. Observera att i verktygsfältet för urval väljs värdet som ett värde från ett fält som heter *=if(Year2023,'Older',Year)*.

Ändra sedan **Definition** av *vCutoffYear* variabeln till *2024*. Observera att även om urvalsfältet *=if(Year2023,'Older',Year)* är detsamma och behåller det valda värdet *2023*, har datatabellen ändrats. Värdet *2023* har ändrats till värdet *Older*. Om du väljer *Older* i kolumnen *if(Year $(vCutoffYear), 'Older', Year)*, görs valet *Older* i ett nytt fält: *=if(Year2024,'Older',Year)*.

I huvudsak skapas ett nytt fält internt för varje värde av *vCutoffYear* variabeln. När man arbetar med samtidiga och föränderliga val är detta beteende kanske inte önskvärt och kan göra datatolkningen mer komplicerad.

Jämförelser, del 2: Beteende med CalcDim()

Nu när du har sett uttryckets beteende när det inte är omslutet av **CalcDim()**, radera alla urval i applikationen. Återställ variabeln *vCutoffYear* till värdet *2023*.

I kolumnen *=CalcDim(if(Year $(vCutoffYear), 'Older', Year))* i tabellen väljer du värdet *2023*. Observera att i urvalsstapeln väljs värdet som ett värde från det underliggande fältet, *Year*.

Ändra sedan **Definition** av *vCutoffYear* variabeln till *2024*. Urvalsfältet *Year* är detsamma, det behåller det valda värdet *2023* och data i tabellen har förändrats, som syntes utan **CalcDim()**. Men om du nu väljer *Older* i *=CalcDim(if(Year $(vCutoffYear), 'Older', Year))* kolumnen, väljs värdet *2024* i *Year*.

Alla val du gör i en beräknad dimension omsluten av **CalcDim()** kommer att återspeglas som val i de underliggande fälten som refereras från den ursprungliga datamodellen. Detta kan förenkla datatolkningen och förhindrar att röriga, komplexa uttryck visas i urvalsstapeln när användare konsumerar analysinnehållet.

EngineVersion

Denna funktion returnerar komplett information om Qlik Sense-motorversionen i form av en sträng.

Syntax:

```
EngineVersion()
```

GetSysAttr

Funktionen **GetSysAttr()** används för att returnera information om klientorganisationen och utrymmet där en applikation är placerad. Med funktionen **GetSysAttr()** kan du utvärdera klientorganisationens ID och namn, utrymmets ID och namn, och andra detaljer om miljön där applikationen finns. Funktionen förenklar identifiering av utrymme och klientorganisation för en Qlik Sense applikation som kan ha flyttats mellan olika miljöer. Funktionen gör det enkelt att identifiera den specifika miljö som du arbetar i. Det är speciellt praktiskt under stadierna utveckling, test, acceptans och produktion (DTAP) i en applikations livscykel.

Denna funktion kan användas i applikationens laddningsskript eller diagramuttryck, antingen direkt som ett uttryck eller i ett uttryck som lagrats som en variabel. Om du förväntar dig att din applikation flyttas mellan utrymmen eller klientorganisationer under dess livscykel, använd funktionen i ett diagramuttryck för att

dynamiskt hämta utrymmes- och klientorganisationsinformation utan att ladda om applikationen. När du gör det, lagra funktionsuttrycket i en variabel utanför laddningsskriptet, eller använd funktionen direkt i diagramuttrycket.

Använd **GetSysAttr()** för att fastställa följande information om din applikation:

- ID för den klientorganisation där applikationen finns.
- Namn på den klientorganisation där applikationen har sin plats.
- URL för den klientorganisation där applikationen har sin plats.
- Det alias-URL som har definierats för den klientorganisation som innehåller applikationen.
- Den region i klientorganisationen som innehåller applikationen.
- ID för det utrymme där applikationen har sin plats.
- Namn på det utrymme där applikationen har sin plats.
- Typ av utrymme (delat eller hanterat) där applikationen har sin plats.

Om du använder den här funktionen i För klienthanterat Qlik Sense kommer den att returnera tomma datavärden. Alltså kan du använda funktionen för att ta fram laddningsskript i För klienthanterat Qlik Sense utan att fel inträffar, i syfte att senare kunna ladda upp applikationerna till Qlik Cloud.

Den fullständiga dokumentationen för Qlik Cloud-funktionen finns i GetSysAttr - skript och diagramfunktion.

HämtaAnvändarAttribut

Denna funktion returnerar användarattribut, inklusive e-postadress, ämne och grupper, för en utvald applikation. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

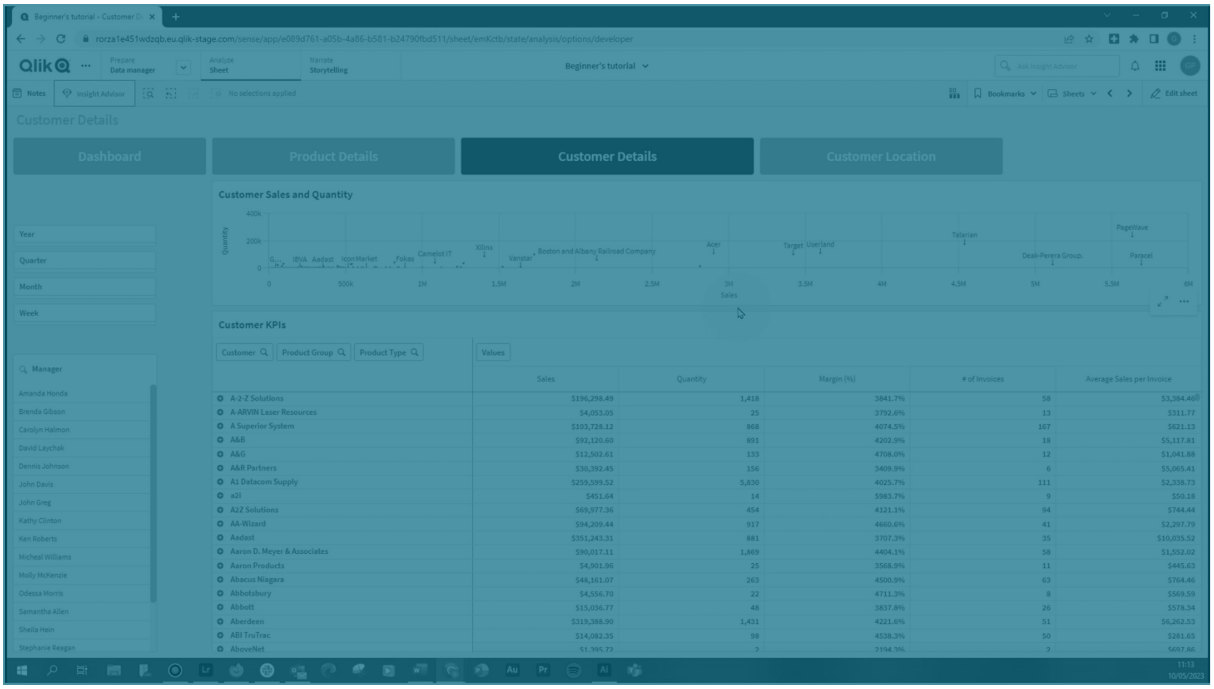
Om du använder den här funktionen i För klienthanterat Qlik Sense kommer den att returnera tomma datavärden. Alltså kan du använda funktionen för att ta fram laddningsskript i För klienthanterat Qlik Sense utan att fel inträffar, i syfte att senare kunna ladda upp applikationerna till Qlik Cloud.

Den fullständiga dokumentationen för Qlik Cloud-funktionen finns i GetSysAttr – skript och diagramfunktion.

InObject - diagramfunktion

Diagramfunktionen **InObject()** utvärderar huruvida det aktuella objektet innehålls inuti ett annat objekt med det ID som anges i funktionsargumentet. Objektet kan vara ett ark eller en visualisering.

Den här funktionen kan användas för att visa hierarkin av objekt i ett ark, från arkobjektet på den högsta nivån till visualiseringar som är kapslade inuti andra visualiseringar. Denna funktion kan användas tillsammans med funktionerna **if** och **ObjectId** för att skapa anpassad navigering i dina applikationer.




```
InObject(id_str)
```

Returnerad datatyp: Boolesk

I Qlik Sense, representeras det booleska sanna värdet av -1, och det falska värdet representeras av 0.

Argument

Argument	Beskrivning
id_str	Ett strängvärde som representerar ID för det objekt som utvärderas.

Du kan använda **Bädda in**-menyn för att hämta objekt-ID:n.

Gör följande:

1. I analysläge, högerklicka på en visualisering eller ett ark, och klicka sedan på **Dela** > **Bädda in**.
2. Kopiera objekt- eller ark-ID:t.

Den här funktionen kan ge oväntade resultat om den anropas i ett objekt (till exempel en knapp) inuti en behållare som är ett originalobjekt. Den här begränsningen gäller även originalobjektet filterruta, som är en behållare för ett antal listrutor. Detta beror på det sätt som originalobjekt använder objekthierarkin.

InObject() används ofta i kombination med följande funktioner:

Relaterade funktioner

Funktion	Interaktion
if (page 655)	Funktionerna **if** och **ObjectId** kan användas tillsammans för att skapa villkorsuttryck. Visualiseringar kan till exempel få villkorad färgläggning genom uttryck som använder dessa funktioner.
ObjectId - diagramfunktion (page 1880)	På liknande sätt som **if** används även **ObjectId** med **InObject** för att skapa villkorsuttryck.

Exempel 1 – grundläggande funktion

Diagramuttryck och resultat

Följande grundläggande exempel visar hur man avgör om ett objekt finns inuti ett annat objekt. I detta fall kontrollerar vi om ett **Text och bild**-objekt finns inuti ett arkobjekt genom att använda ID:t för arket som argument.

Gör följande:

1. Öppna ett nytt ark och öppna **Diagram** i panelen **Tillgångar**.
2. Under **Äldre**, dra ett **Text & bild**diagram till arket.
3. I egenskapsfönstret klickar du på **Lägg till mått**.
4. Klicka på f_x för att öppna uttrycksredigeraren.
5. Klistra in följande uttryck i dialogrutan:
   `=InObject()`
6. Ändra uttrycket så att det innehåller ID för ditt ark som en sträng inuti parentesen.
   Om arket till exempel har ID 1234-5678, använder du följande:
   `=InObject('1234-5678')`
7. Klicka på **Tillämpa**.

Värdet `-1` visas i diagrammet, vilket anger att uttrycket har utvärderats till att vara sant.

Exempel 2 – objekt med villkorsstyrda färger

Diagramuttryck och resultat

Översikt

Följande exempel visar hur du skapar anpassade navigeringsknappar som visar olika färger för att ange vilket ark som är öppet.

Börja med att skapa en ny applikation och öppna Dataladdningsredigeraren. Klistra in följande laddningsskript i en ny flik. Observera att själva data är en platshållare och inte kommer att användas i exempelinnehållet.

Laddningsskript

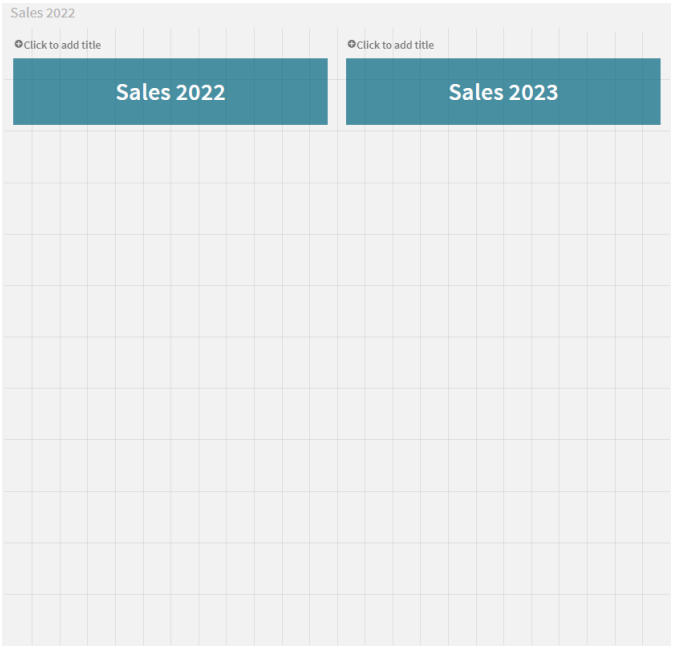
```
Transactions:

Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'4/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'7/26/2022',45.89
8198,'8/9/2022',36.23
8199,'9/22/2022',25.66
8200,'11/23/2022',82.77
8201,'12/27/2022',69.98
8202,'1/1/2023',76.11
8203,'2/8/2022',25.12
8204,'3/19/2022',46.23
8205,'6/26/2022',84.21
8206,'9/14/2022',96.24
8207,'11/29/2022',67.67
];
```

Skapa visualiseringarna

Läs in data och skapa två nya ark. Ge den namnen *Försäljning 2022* och *Försäljning 2023*.

Bygg sedan två knappobjekt som kommer att användas för att navigera mellan de två arken.

Gör följande:

1. Lägg till två **Knapp**-objekt till arket.
2. Under **Utseende** > **Allmänt** anger du **Etikett** för knapparna till *Försäljning 2022* och *Försäljning 2023* respektive.
3. Ordna knapparna så att de matchar följande bild.
   Arrangemang för arket Försäljning 2022 med två navigeringsknappar
   

4. Välj knappen *Försäljning 2022* och expandera **Åtgärder och navigering** i egenskapsfönstret.
5. Klicka på **Lägg till åtgärd**. Under **Navigering** väljer du **Gå till ett ark**.
6. Under **Ark** väljer du *Försäljning 2022*.
7. Upprepa dessa inställningar för knappar så att knappen **Försäljning 2023** länkas till arket *Försäljning 2023*.
8. Konvertera knapparna till huvudobjekt genom att högerklicka på dem och välja **Lägg till bland originalen**.

Du kan nu kopiera knapparna en och en och klistra in dem i arket *Försäljning 2023* med samma storlek och arrangemang på arket.

Skapa villkorsstyrda färger

Konfigurera sedan knapparna så att de är blå om de är länkade till det öppna arket, och ljusgrå om de är länkade till det ark som inte är öppet.

Gör följande:

1. Öppna arket *Försäljning 2022* och hämta arkets ID från URL-adressen. Håll arket *Försäljning 2022* öppet.

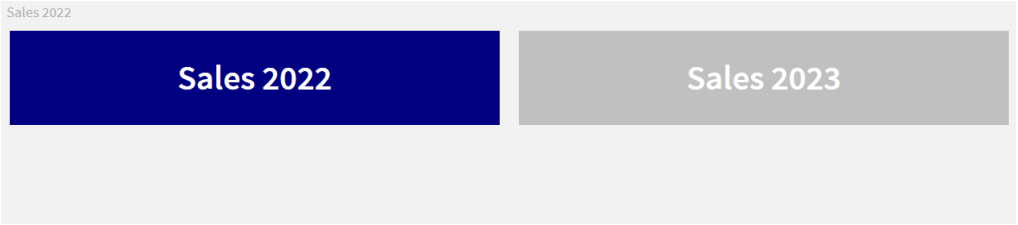
2. Klicka på originalobjekt-knappen **Försäljning 2022** och välj **Redigera** i egenskapsfönstret.
3. Under **Utseende** > **Bakgrund** väljer du att färglägga knappen **Per uttryck**.
4. I **Uttryck** klistrar du in följande text:
   =if(InObject(''), Blue(), LightGray())
5. Inuti parentesen i ovanstående uttryck klistrar du in ark-ID för arket *Försäljning 2022*.

Knappen har nu konfigurerats så att den blir blå om arket *Försäljning 2022* är öppet, och ljusgrått om det inte är öppet.

Upprepa ovanstående instruktioner för arket *Försäljning 2023*, och länka knapporiginalobjektet **Försäljning 2023** till ark-ID för *Försäljning 2023*.

Båda arken ska nu ha två knappar, där knappen för det öppna arket är blå.

Arket Försäljning 2022 med blå färg för att visa att Försäljning 2022 visas



IsPartialReload

Den här funktionen returnerar -1 (True) om den aktuella återinläsningen är partiell, annars 0 (False).

Syntax:

```
IsPartialReload()
```

ObjectId - diagramfunktion

Diagramfunktionen **ObjectId()** returnerar ID:t för det objekt som uttrycket utvärderas för. Funktionen har ett valfritt argument som anger vilken typ av objekt funktionen gäller. Objektet kan vara ett ark eller en visualisering. Denna funktion är bara tillgänglig i diagramuttryck.

Syntax:

```
ObjectId([object_type_str])
```

Returnerad datatyp: sträng

Funktionens enda argument, **object_type_str**, är valfritt och refererar till ett strängvärde som representerar objektets typ.

Argument

Argument	Beskrivning
object_type_str	Ett strängvärde som representerar den typ av objekt som utvärderas.

Om inget argument anges i funktionsuttrycket returnerar **ObjectId()** ID för det objekt i vilket uttrycket används. Om du vill returnera ID för det arkobjekt i vilket visualiseringen visas använder du *ObjectId ('sheet')*.

För visualiseringsobjekt som är kapslade inuti andra visualiseringsobjekt, anger du önskad objekttyp i funktionsargumentet för att få andra resultat. För exempelvis ett **Text och bild**-diagram inuti en behållare använder du *'text-image'* för att returnera **Text och bild**-objektet och *'container'* för att returnera ID för behållaren.

Du kan använda menyn **Bädda in** för att hämta objekt-ID:n.

Gör följande:

1. I analysläge, högerklicka på en visualisering eller ett ark, och klicka sedan på **Dela** > **Bädda in**.
2. Kopiera objekt- eller ark-ID:t.

Begränsningar:

Den här funktionen kan ge oväntade resultat om den anropas i ett objekt (till exempel en knapp) inuti en behållare som är ett originalobjekt. Den här begränsningen gäller även originalobjektet filterruta, som är en behållare för ett antal listrutor. Detta beror på det sätt som originalobjekt använder objekthierarkin.

Diagramuttrycket *ObjectId('sheet')* returnerar en tom sträng i dessa fall, medan *ObjectId('masterobject')* visar ID för ägande originalobjekt.

ObjectId() används ofta i kombination med följande funktioner:

Relaterade funktioner

Funktion	Interaktion
if (page 655)	Funktionerna **if** och **ObjectId** kan användas tillsammans för att skapa villkorsuttryck. Visualiseringar kan till exempel få villkorad färgläggning genom uttryck som använder dessa funktioner.

Funktion	Interaktion
InObject - diagramfunktion (page 1875)	På liknande sätt som **if** används även **InObject** med **ObjectID** för att skapa villkorsuttryck.

Exempel 1 – returnera diagramobjekt-ID

Diagramuttryck och resultat

Följande grundläggande exempel visar hur du returnerar ID för en visualisering.

Gör följande:

1. Öppna ett nytt ark och öppna **Diagram** i panelen **Tillgångar**.
2. Under **Äldre**, dra ett **Text & bild**-diagram till arket.
3. I egenskapsfönstret klickar du på **Lägg till mått**.
4. Klicka på $f\times$ för att öppna uttrycksredigeraren.
5. Klistra in följande uttryck i dialogrutan:
   `=ObjectId()`
6. Klicka på **Tillämpa**.

ID för **Text och bild**-objektet visas i visualiseringen.

Du kan få samma resultat med följande uttryck:

```
=ObjectId('text-image')
```

Exempel 2 – returnera ark-ID

Diagramuttryck och resultat

Följande grundläggande exempel visar hur du returnerar ID för det ark i vilket en visualisering visas.

Gör följande:

1. Öppna ett nytt ark och öppna **Diagram** i panelen **Tillgångar**.
2. Under **Äldre**, dra ett **Text & bild**-diagram till arket.
3. I egenskapsfönstret klickar du på **Lägg till mått**.
4. Klicka på $f\times$ för att öppna uttrycksredigeraren.
5. Klistra in följande uttryck i dialogrutan:
   `=ObjectId('sheet')`
6. Klicka på **Tillämpa**.

Arkets ID visas i visualiseringen.

Exempel 3 – kapslat uttryck

Diagramuttryck och resultat

Följande exempel visar hur funktionen **ObjectId()** kan kapslas inuti andra uttryck.

Gör följande:

1. Öppna ett nytt ark och öppna **Diagram** i panelen **Tillgångar**.
2. Under **Äldre**, dra ett **Text & bild**-diagram till arket.
3. I egenskapsfönstret klickar du på **Lägg till mått**.
4. Klicka på f_x för att öppna uttrycksredigeraren.
5. Klistra in följande uttryck i dialogrutan:
   =if(InObject(ObjectId('text-image')), 'I Text och bild', 'Inte i Text och bild')
6. Klicka på **Tillämpa**.

Texten *I Text och bild* visas i diagrammet, vilket anger att det objekt som refereras till i uttrycket är ett **Text och bild**-diagram.

Ett mer detaljerat exempel med villkorsstyrd färgläggning finns i exemplet om *InObject - diagramfunktion (page 1875)*

ProductVersion

Denna funktion returnerar komplett information om Qlik Sense-version och byggnummer i form av en sträng. Funktionen är utfasad och ersatt med **EngineVersion()**.

Syntax:

```
ProductVersion()
```

StateName - diagramfunktion

StateName() returnerar namnet på det alternativa tillståndet för visualiseringen där den används. StateName kan till exempel användas för att skapa visualiseringar med text och färger som är dynamiska för att återspegla när tillståndet för en visualisering ändras. Denna funktion kan användas i diagramuttryck men kan inte användas för att fastställa det tillstånd som uttrycket syftar på.

Syntax:

```
StateName ()
```

Example 1:

Dynamisk text

```
='Region - ' & if(StateName() = '$', 'Default', StateName())
```

Example 2:

Dynamiska färger

```
if(StateName() = 'Group 1', rgb(152, 171, 206),
      if(StateName() = 'Group 2', rgb(187, 200, 179),
            rgb(210, 210, 210)
      )
)
```

8.26 Tabellfunktioner

Tabellfunktionerna returnerar information om den datatabell som för närvarande läses. Om inget tabellnamn har angetts och funktionen används inom en **LOAD**-sats antas den aktuella tabellen.

Alla funktioner kan användas i dataladdningsskriptet, men endast **NoOfRows** kan användas i ett diagramuttryck.

Tabellfunktioner - översikt

En del av funktionerna beskrivs mer ingående efter översikten. För de här funktionerna kan du klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

FieldName

Skriptfunktionen **FieldName** returnerar namnet på fältet med det angivna numret inom en tidigare inläst tabell. Om funktionen används inom en **LOAD**-sats får den inte referera till den tabell som för närvarande används.

```
FieldName (field_number ,table_name)
```

FieldNumber

Skriptfunktionen **FieldNumber** returnerar numret på ett angivet fält i en tidigare inläst tabell. Om funktionen används inom en **LOAD**-sats får den inte referera till den tabell som för närvarande används.

```
FieldNumber (field_name ,table_name)
```

NoOfFields

Skriptfunktionen **NoOfFields** returnerar antalet fält i en tidigare inläst tabell. Om funktionen används inom en **LOAD**-sats får den inte referera till den tabell som för närvarande används.

```
NoOfFields (table_name)
```

NoOfRows

Funktionen **NoOfRows** returnerar antalet rader (poster) i en tidigare inläst tabell. Om funktionen används inom en **LOAD**-sats får den inte referera till den tabell som för närvarande används.

```
NoOfRows (table_name)
```

NoOfTables

Den här skriptfunktionen returnerar antalet tidigare inlästa tabeller.

```
NoOfTables()
```

TableName

Den här skriptfunktionen returnerar namnet på tabellen med det angivna numret.

```
TableName(table_number)
```

TableNumber

Den här skriptfunktionen returnerar numret på den angivna tabellen. Den första tabellen har nummer 0.

Om table_name inte finns returneras NULL.

```
TableNumber(table_name)
```

Exempel:

I det här exemplet vill vi skapa en tabell med information om tabeller och fält som har laddats.

Först ska vi ladda några exempeldata. Detta skapar de två tabeller som kommer att användas för att illustrera tabellfunktionerna som beskrivs i detta avsnitt.

```
Characters:
Load Chr(RecNo()+Ord('A')-1) as Alpha, RecNo() as Num autogenerate 26;

ASCII:
Load
 if(RecNo()>=65 and RecNo()<=90,RecNo()-64) as Num,
 Chr(RecNo()) as AsciiAlpha,
 RecNo() as AsciiNum
autogenerate 255
 Where (RecNo()>=32 and RecNo()<=126) or RecNo()>=160 ;
```

Sedan itererar vi igenom tabellerna som har laddats med hjälp av funktionen **NoOfTables** och sedan genom fälten i varje tabell, med hjälp av funktionen **NoOfFields** och laddar information med hjälp av tabellfunktionerna.

```
//Iterate through the loaded tables
For t = 0 to NoOfTables() - 1

//Iterate through the fields of table
 For f = 1 to NoOfFields(TableName($(t)))
  Tables:
  Load
   TableName($(t)) as Table,
   TableNumber(TableName($(t))) as TableNo,
   NoOfRows(TableName($(t))) as TableRows,
   FieldName($(f),TableName($(t))) as Field,
   FieldNumber(FieldName($(f),TableName($(t))),TableName($(t))) as FieldNo
   Autogenerate 1;
 Next f
Next t;
```

Tabellen Tables som skapas ser ut så här:

Load table

Table	TableNo	TableRows	Field	FieldNo
Characters	0	26	Alpha	1
Characters	0	26	Num	2
ASCII	1	191	Num	1
ASCII	1	191	AsciiAlpha	2
ASCII	1	191	AsciiNum	3

FieldName

Skriptfunktionen **FieldName** returnerar namnet på fältet med det angivna numret inom en tidigare inläst tabell. Om funktionen används inom en **LOAD**-sats får den inte referera till den tabell som för närvarande används.

Syntax:

```
FieldName(field_number ,table_name)
```

Argument:

Argument

Argument	Beskrivning
field_number	Fältnumret till det fält som du vill referera till.
table_name	Den tabell som innehåller det fält som du vill referera till.

Exempel:

```
LET a = FieldName(4,'tab1');
```

FieldNumber

Skriptfunktionen **FieldNumber** returnerar numret på ett angivet fält i en tidigare inläst tabell. Om funktionen används inom en **LOAD**-sats får den inte referera till den tabell som för närvarande används.

Syntax:

```
FieldNumber(field_name ,table_name)
```

Argument:

Argument

Argument	Beskrivning
field_name	Fältets namn.
table_name	Namnet på den tabell som innehåller fältet.

Om fältet field_name inte finns i table_name, eller table_name inte finns, returnerar funktionen 0.

Exempel:

```
LET a = FieldNumber('Customer','tab1');
```

NoOfFields

Skriptfunktionen **NoOfFields** returnerar antalet fält i en tidigare inläst tabell. Om funktionen används inom en **LOAD**-sats får den inte referera till den tabell som för närvarande används.

Syntax:

```
NoOfFields(table_name)
```

Argument:

Argument

Argument	Beskrivning
table_name	Tabellens namn.

Exempel:

```
LET a = NoOfFields('tab1');
```

NoOfRows

Funktionen **NoOfRows** returnerar antalet rader (poster) i en tidigare inläst tabell. Om funktionen används inom en **LOAD**-sats får den inte referera till den tabell som för närvarande används.

Syntax:

```
NoOfRows(table_name)
```

Argument:

Argument

Argument	Beskrivning
table_name	Tabellens namn.

Exempel:

```
LET a = NoOfRows('tab1');
```

8.27 Trigonometriska och hyperboliska funktioner

Den här delen beskriver funktioner för att utföra trigonometriska och hyperboliska operationer. I alla funktionerna är argumenten uttryck som resulterar i vinklar mätta i radianer, där **x** ska tolkas som ett verkligt tal.

Alla vinklar mäts i radianer.

Alla funktioner kan användas både i dataladdningsskriptet och diagramuttryck.

cos

Cosinus av **x**. Resultatet blir ett tal mellan -1 och 1.

```
cos( x )
```

acos

Invers cosinus av **x**. Funktionen är bara definierad om -1≤**x**≤1. Resultatet blir ett tal mellan 0 och π.

```
acos( x )
```

sin

Sinus av **x**. Resultatet blir ett tal mellan -1 och 1.

```
sin( x )
```

asin

Invers sinus av **x**. Funktionen är bara definierad om -1≤**x**≤1. Resultatet blir ett tal mellan - π/2 och π/2.

```
asin( x )
```

tan

Tangens av **x**. Resultatet blir ett reellt tal.

```
tan( x )
```

atan

Invers tangens av **x**. Resultatet blir ett tal mellan - π/2 och π/2.

```
atan( x )
```

atan2

Tvådimensionell generalisering av invers tangens-funktionen. Returnerar vinkeln mellan origo och den punkt som motsvaras av koordinaterna **x** och **y**. Resultatet blir ett tal mellan - π och + π.

```
atan2( y,x )
```

cosh

Cosinus hyperbolicus av **x**. Resultatet blir ett positivt reellt tal.

```
cosh( x )
```

sinh

Sinus hyperbolicus av **x**. Resultatet blir ett reellt tal.

```
sinh( x )
```

tanh

Tangens hyperbolicus av **x**. Resultatet blir ett reellt tal.

```
tanh( x )
```

acosh

Invers cosinus hyperbolicus av **x**. Resultatet blir ett positivt reellt tal.

```
acosh( x )
```

asinh

Invers sinus hyperbolicus av **x**. Resultatet blir ett reellt tal.

```
asinh( x )
```

atanh

Invers tangens hyperbolicus av **x**. Resultatet blir ett reellt tal.

```
atanh( x )
```

Exempel:

Följande skript laddar exempeldata och laddar sedan en tabell som innehåller de beräknade trigonometriska och hyperboliska operationerna för värdena.

```
SampleData:
LOAD * Inline
[Value
-1
0
1];

Results:
Load *,
cos(Value),
acos(Value),
sin(Value),
asin(Value),
tan(Value),
atan(Value),
atan2(Value, Value),
cosh(Value),
sinh(Value),
tanh(Value)
```

```
RESIDENT SampleData;

Drop Table SampleData;
```

8.28 Fönsterfunktioner

Fönsterfunktioner utför beräkningar från flera rader och skapar ett värde för varje rad separat. Fönsterfunktioner kan bara beräknas när hela tabellen har lästs in.

Du kan använda fönsterfunktionerna för att utföra åtgärder som:

- Jämföra ett individuellt talvärde i en rad med medelvärdet, maxvärdet eller minimivärdet i kolumnen.
- Beräkna placeringen för ett enskilt värde, antingen i kolumnen eller i hela tabellen.

Fönsterfunktioner ändrar inte antalet poster i tabellen men kan utföra liknande uppgifter, t.ex. aggregeringsfunktioner, relationella funktioner eller intervallfunktioner.

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Window
Window-funktionen utför beräkningar från flera rader och skapar ett värde för varje rad separat.

```
Window (main_expr, [partition1, partition2, ...], [sort_type, [sort_expr],
[filter_expr], [start_expr, end_expr]])
```

WRank
WRank-funktionen utför rangordningsberäkningar inuti **Window**.

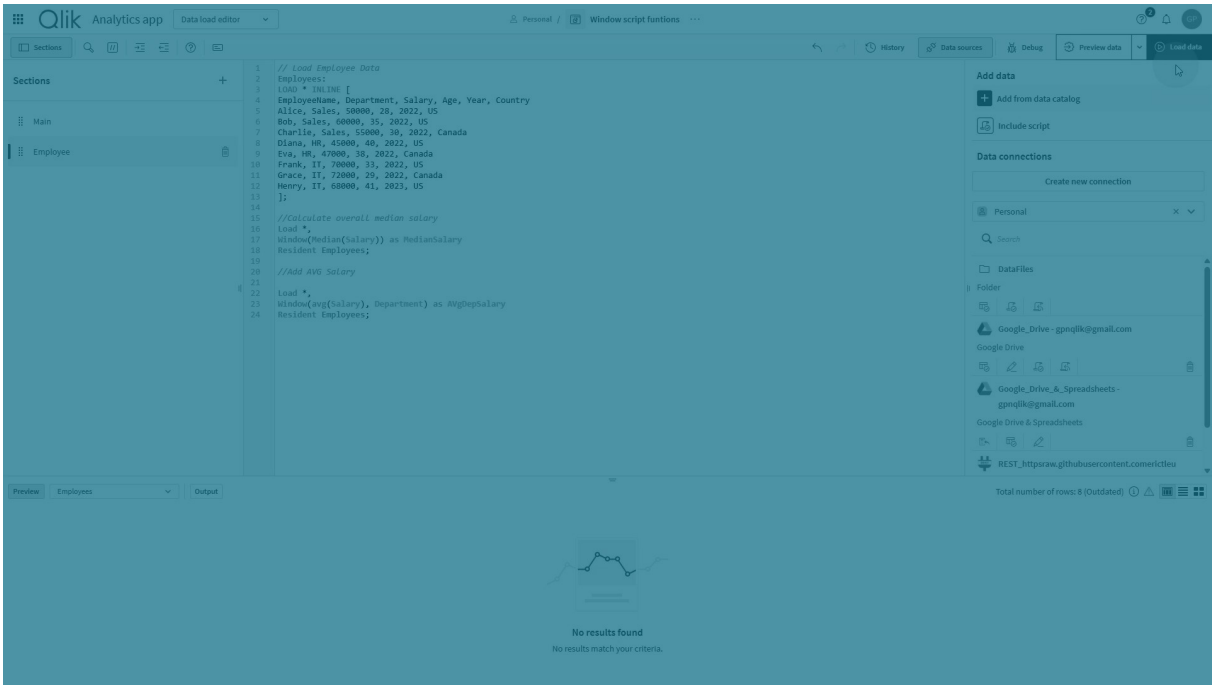
```
WRank ([mode[, fmt]])
```

Window

Window() utför beräkningar från flera rader i ett delavsnitt (eller "fönster") i en komplett tabell och producerar ett enda värde för varje rad.

Du kan använda **Window**-funktionerna för att utföra åtgärder som:

- Jämföra ett individuellt talvärde i en rad med medelvärdet, maxvärdet eller minimivärdet i kolumnen.
- Beräkna placeringen för ett enskilt värde, antingen i kolumnen eller i hela tabellen.



Window-funktionen ändrar inte antalet poster i tabellen men kan fortfarande utföra liknande uppgifter, t.ex. aggregering, relationella och intervallfunktioner.

Funktionen **Window** måste ingå i LOAD-satsen i den tabell du arbetar med, eller hänvisa till tabellen, t.ex. med en Resident LOAD. Se till exempel *Exempel - lägga till ett fält som innehåller en aggregering: (page 1895)*.

Namnet kan utelämnas, men detta resulterar i att fältnamnet sätts till hela funktionsuttrycket **Window**. Till exempel *AvgDepSalary* i exemplet nedan:

```
[AvgSalary]:
Load
       *,
       Window(avg(Salary), Department) as AvgDepSalary
Resident Employees;
```

> *Vi rekommenderar att du ger **Window**-funktionen ett fältnamn.*

Window har stöd för allmänna funktioner, till exempel avrundning eller grundläggande numeriska operationer. Exempel:

```
Load *, Round(Window(Sum(Salary),Department)) as SumDepSalary
Load *, Window(Sum(Salary),Department) + 5 as SumDepSalary
```

__Window__-funktionen söker igenom hela tabellen. När det gäller minnesanvändning är kostnaden för denna skanning proportionerlig mot antalet fält i tabellen. En tabell med ett fåtal fält, förutom de som __Window__-funktionen arbetar med, är alltid att föredra framför en tabell med många fält.

Vanligtvis förbrukar en __Window__-funktion 2-4 gånger så mycket minne och tar 2-4 gånger längre tid att läsa in jämfört med en liknande skriptfunktion som innehåller samma aggregering, en `where`-sats (filter), en `Group by`-sats (partition) och en `Order by`-sats (sort).

Syntax:

```
Window (main_expr, [partition1, partition2, ...], [sort_type, [sort_expr],
[filter_expr], [start_expr, end_expr]])
```

Datatyp för returen: Ett värde per rad, som kan kombineras med eller användas i ett annat uttryck, eller läggas till direkt i den resulterande tabell som skapas av LOAD-satsen.

Argument:

Argument

Argument	Beskrivning

main_expr	Det huvudsakliga indatauttrycket beräknas och returneras av funktionen. Det måste vara ett uttryck som baseras på en aggregering, till exempel `Median(Salary)`. Exempel:
	`Window(Median(Salary)) as MedianSalary`
	Uttrycket kan också vara ett fältnamn utan att någon aggregering tillämpats. I det här fallet behandlar **Window** den som om **Only()**-funktionen har tillämpats på det fältet. Exempel:
	`Window(Salary) as WSalary`
	Uttryck som innehåller nästlade aggregeringsfunktioner eller nästlade fönsterfunktioner, t.ex. `Window(Sum(Avg(Salary)))` eller `Window(Sum(Window(Salary)))`, är inte tillåtna.

Argument	Beskrivning

partition 1, partition 2	Efter main_expr kan du definiera valfritt antal partitioner. Partitioner är fält som definierar vilka kombinationer som aggregeringarna ska tillämpas med. Exempel:
	`Window(Avg(Salary), Department) as AvgDepSalary`
	Detta beräknar den genomsnittliga lönen för varje avdelning.
	
	När flera partitionsparametrar anges kommer aggregeringen att tillämpas på varje rad med en unik kombination av partitionsfält. Exempel:
	`Window(Avg(Salary), Department, Country) as AvgDepSalaryPerCountry`
	Detta beräknar den genomsnittliga lönen för varje avdelning i varje land. Kopior av avdelningar i olika länder, t.ex. *Marknadsföring* i *Sverige* och *Marknadsföring* i *Kanada*, kommer att behandlas som olika avdelningar.
	
	Partitionering liknar den gruppering som uppnås genom **group by** klausulen. Partitioneringen minskar dock inte antalet poster jämfört med indatatabellen.

sort_type, [sort_expr]	Ange sorteringstyp och sorteringsuttryck (valfritt). Detta definierar den ordning i vilken aggregeringen utförs. sort_type är en konstant sträng och kan ha ett av tre värden: • ASC: stigande sortering. • DESC: fallande sortering. • NONE Om du definierar sort_type ASC eller DESC måste du definiera ett sorteringsuttryck. Med sort_type NONE ska inget sorteringsuttryck anges. Sorteringsuttrycket avgör ordningen för raderna i en partition. Exempel: `Window(RecNo(), Department, 'ASC', Year)` I exemplet ovan sorteras aggregationen i partitionen stigande av *Årsfältet*. Sorteringsparametrarna påverkar också den ordning i vilken ett glidande fönster bearbetar en tabell. Exempel finns i *Exempel - Lägga till ett fält med ett glidande fönster (page 1898)*.

Argument	Beskrivning
filter_expr	Lägg till ett filteruttryck (valfritt). Detta är ett booleskt uttryck som avgör huruvida posten ska inkluderas i beräkningen eller inte. Om filter_expr tillhandahålls ska en sorteringstyp anges. Om ingen sorteringstyp är definierad kan argumentet tolkas som en partitionsparameter i stället för ett sorteringsuttryck. Parametern filter_expr kan utelämnas helt eller så kan ett sanningsenligt värde anges. Båda dessa resulterar i att inga filter används så att inga rader utesluts. Exempel: `Window(avg(Salary), Department, 'ASC', Age, Country='US' Or Country='Canada') as AvgDepSalaryInUsOrCanada` Ovanstående **Window**-sats filtrerar bort alla rader där fältet *Country* inte är *US* eller *Canada*, vilket utesluter dessa rader från aggregeringen. Det kommer fortfarande att finnas utdatavärden för de rader som filtrerades bort, men precis som alla andra utdatarader kommer de bara att innehålla aggregerade resultat från rader som inte uteslutits av filtret.

start_expr,end_expr	Alternativt kan du ställa in argumentet för glidande fönster-funktionalitet. Detta begränsar ytterligare antalet rader som kan aggregeras tillsammans. Varje utdatarad kommer endast att innehålla (aggregerade) resultat från ett antal rader som föregår och följer på den aktuella raden. Ett glidande fönster kräver två argument (start_expr,end_expr). Om dessa argument anges ska sorteringstypen inte hoppas över. Annars kan argumenten tolkas som en partitionsparameter i stället för glidande fönster. Parametern start_expr,end_expr omfattar:
	
	• Startuttryck: antalet rader före den aktuella raden som ska inkluderas i fönstret.
	• Slututtryck: antalet rader efter den aktuella raden som ska inkluderas i fönstret.
	
	Om du exempelvis vill inkludera de 3 föregående raderna, aktuell rad och nästa följande rad:
	
	`Window(concat(Text(Salary),'-'), Department, 'NONE', 1, -3, 1) as WSlidingSalaryDepartment`
	För att indikera alla föregående rader eller alla efterföljande rader kan du använda funktionen **Unbounded()**. Om du exempelvis vill inkludera alla föregående rader, aktuell rad och nästa följande rad:
	
	`Window(concat(Text(Salary),'-'), Department, 'NONE', 1, UNBOUNDED(), 1) as WSlidingSalaryDepartment`
	Om du exempelvis vill inkludera den tredje raden från den aktuella raden och alla efterföljande rader:
	
	`Window(concat(Text(Salary),'-'), Department, 'NONE', 1, 3, UNBOUNDED()) as WSlidingSalaryDepartment`
	I exemplen ovan anges sorteringstypen NONE så att skriptet kan skilja mellan partitionsargument och de parametrar som kommer efter sorteringstyp.

Exempel - lägga till ett fält som innehåller en aggregering:

Exempel: lägga till ett fält som innehåller en aggregering:

Laddningsskript

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. Skapa tabellen nedan i Qlik Sense för att se resultatet.

```
[Employees]:
Load
       *,
       Window(avg(Salary), Department) as AvgDepSalary
Inline [
       Name, Department, Salary, Age, Year, Country
       David, Sales, 11, 32, 2019, Canada
       Sara, Sales, 22, 25, 2022, Sweden
       Nils, Marketing, 33, 18, 2022, Sweden
       Patrick, Marketing, 44, 43, 2022, Canada
       Anna, Sales, 55, 33, 2023, Canada
       Johan, Sales, 66, 26, 2025, Sweden
       Olle, Marketing, 77, 19, 2023, Sweden
       Bonnie, Marketing, 88, 44, 2023, US
];
```

Resultat

Resultat för att lägga till ett fält som innehåller en aggregering

Name	Department	Salary	Age	Year	Country	AvgDepSalary
David	Sales	11	32	2019	Canada	39
Sara	Sales	22	25	2022	Sweden	39
Nils	Marketing	33	18	2022	Sweden	61
Patrick	Marketing	44	43	2022	Canada	61
Anna	Sales	55	33	2023	Canada	39
Johan	Sales	66	26	2025	Sweden	39
Olle	Marketing	77	19	2023	Sweden	61
Bonnie	Marketing	88	44	2023	US	61

Exempel - lägga till ett fält som innehåller en aggregering som filtrerats för specifika värden

Exempel: lägga till ett fält som innehåller en aggregering som filtrerats för specifika värden

Laddningsskript

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. Skapa tabellen nedan i Qlik Sense för att se resultatet.

```
[Employees]:
Load
*,
Window(avg(Salary), Department, 'NONE', Country = 'Sweden') as AvgDepSalarySweden
Inline [
       Name, Department, Salary, Age, Year, Country
       David, Sales, 11, 32, 2019, Canada
       Sara, Sales, 22, 25, 2022, Sweden
       Nils, Marketing, 33, 18, 2022, Sweden
       Patrick, Marketing, 44, 43, 2022, Canada
       Anna, Sales, 55, 33, 2023, Canada
       Johan, Sales, 66, 26, 2025, Sweden
       Olle, Marketing, 77, 19, 2023, Sweden
       Bonnie, Marketing, 88, 44, 2023, US
];
```

Resultat

Resultat för att lägga till ett fält som innehåller en aggregering som filtrerats för specifika värden

Name	Department	Salary	Age	Year	Country	AvgDepSalarySweden
David	Sales	11	32	2019	Canada	44
Sara	Sales	22	25	2022	Sweden	44
Nils	Marketing	33	18	2022	Sweden	55
Patrick	Marketing	44	43	2022	Canada	55
Anna	Sales	55	33	2023	Canada	44
Johan	Sales	66	26	2025	Sweden	44
Olle	Marketing	77	19	2023	Sweden	55
Bonnie	Marketing	88	44	2023	US	55

Exempel - Lägga till ett fält med ett glidande fönster

Exempel: lägga till ett fält med ett glidande fönster

Laddningsskript

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. Skapa tabellen nedan i Qlik Sense för att se resultatet.

```
[Employees]:
Load
       *,
       Window(concat(Text(Salary),'-'), Department, 'NONE', -1, 1) as SlidingWindow
Inline [
       Name, Department, Salary, Age, Year, Country
       David, Sales, 11, 32, 2019, Canada
       Sara, Sales, 22, 25, 2022, Sweden
       Nils, Marketing, 33, 18, 2022, Sweden
       Patrick, Marketing, 44, 43, 2022, Canada
       Anna, Sales, 55, 33, 2023, Canada
       Johan, Sales, 66, 26, 2025, Sweden
       Olle, Marketing, 77, 19, 2023, Sweden
       Bonnie, Marketing, 88, 44, 2023, US
];
```

Resultat

Resultat för att lägga till ett fält som innehåller en aggregering som filtrerats för specifika värden

Name	Department	Salary	Age	Year	Country	SlidingWindow
David	Sales	11	32	2019	Canada	11-22
Sara	Sales	22	25	2022	Sweden	11-22-55
Nils	Marketing	33	18	2022	Sweden	33-44
Patrick	Marketing	44	43	2022	Canada	33-44-77
Anna	Sales	55	33	2023	Canada	22-55-66
Johan	Sales	66	26	2025	Sweden	55-66
Olle	Marketing	77	19	2023	Sweden	44-77-88
Bonnie	Marketing	88	44	2023	US	77-88

Exempel – Lägga till ett fält med ett glidande fönster

Exempel: lägga till ett fält med ett glidande fönster

Laddningsskript

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. Skapa tabellen nedan i Qlik Sense för att se resultatet.

```
[Employees]:
Load
       *,
       Window(concat(Text(Salary),'-'), Department, 'ASC', Age, -1, 1) as SortedSlidingWindow
Inline [
       Name, Department, Salary, Age, Year, Country
       David, Sales, 11, 32, 2019, Canada
       Sara, Sales, 22, 25, 2022, Sweden
       Nils, Marketing, 33, 18, 2022, Sweden
       Patrick, Marketing, 44, 43, 2022, Canada
       Anna, Sales, 55, 33, 2023, Canada
       Johan, Sales, 66, 26, 2025, Sweden
       Olle, Marketing, 77, 19, 2023, Sweden
       Bonnie, Marketing, 88, 44, 2023, US
];
```

Resultat

Resultat för att lägga till ett fält som innehåller en aggregering som filtrerats för specifika värden

Name	Department	Salary	Age	Year	Country	SortedSlidingWindow
David	Sales	11	32	2019	Canada	11-55-66
Sara	Sales	22	25	2022	Sweden	22-66
Nils	Marketing	33	18	2022	Sweden	33-77-88
Patrick	Marketing	44	43	2022	Canada	44-77
Anna	Sales	55	33	2023	Canada	11-55
Johan	Sales	66	26	2025	Sweden	11-22-66
Olle	Marketing	77	19	2023	Sweden	33-44-77
Bonnie	Marketing	88	44	2023	US	33-88

För att bättre förstå hur partitionerna och sorteringsordningen påverkar aggregeringarna i ett glidande fönster kan du sortera tabellen stigande efter namn.

Exempel - lägga till ett fält som innehåller en aggregering som filtrerats för specifika värden

Name	Department	Salary	Age	Year	Country	SortedSlidingWindow
Anna	Sales	55	33	2023	Canada	11-55

Name	Department	Salary	Age	Year	Country	SortedSlidingWindow
Bonnie	Marketing	88	44	2023	US	33-88
David	Sales	11	32	2019	Canada	11-55-66
Johan	Sales	66	26	2025	Sweden	11-22-66
Nils	Marketing	33	18	2022	Sweden	33-77-88
Olle	Marketing	77	19	2023	Sweden	33-44-77
Patrick	Marketing	44	43	2022	Canada	44-77
Sara	Sales	22	25	2022	Sweden	22-66

I raden för *Johan* har aggregeringen inkluderat lönevärdena för både *David* och *Sara.* Det här är föregående och följande rader inom partitionen *Försäljningsavdelningen* när de sorteras efter ålder.

För ett annat exempel som visar vilka rader som ingår, ändra uttrycket **Windows** från `Window(concat(Text(Salary),'-'), Department, 'ASC', Age, -1, 1)` till `Window(concat(Text(Name),'-'), Department, 'ASC', Age, -1, 1)` och ladda om. Värdena i SortedSlidingWindow i tabellen nedan ger namnet från föregående och följande rader inom partitionen *Försäljning*.

Resultat för att lägga till ett fält som innehåller en aggregering som filtrerats för specifika värden

Name	Department	Salary	Age	Year	Country	SortedSlidingWindow
Anna	Sales	55	33	2023	Canada	Anna-David
Bonnie	Marketing	88	44	2023	US	Bonnie-Nils
David	Sales	11	32	2019	Canada	Anna-David-Johan
Johan	Sales	66	26	2025	Sweden	David-Johan-Sara
Nils	Marketing	33	18	2022	Sweden	Bonnie-Nils-Olle
Olle	Marketing	77	19	2023	Sweden	Nils-Olle-Patrick
Patrick	Marketing	44	43	2022	Canada	Olle-Patrick
Sara	Sales	22	25	2022	Sweden	Johan-Sara

Begränsningar

Window har följande begränsningar:

- **Window** är en resursintensiv funktion, framför allt avseende minnesförbrukning.
- **Window** stöds inte i Qlik Sense Mobile.
- Diagramuttrycken har inte stöd för **Window**.
- Du kan inte nästla **Window**-funktioner inuti andra **Window**-funktioner.
- **Window** kan inte användas inuti en aggregeringsfunktion.
- **Window** kan inte användas i en LOAD-sats med en group by-sats.
- **Window** måste kunna söka genom hela tabellen.

- **WRank()**, **RecNo()** och **RowNo()** kan inte användas tillsammans med **Window** när glidande fönster-funktionaliteten används.
- När funktionen med glidande fönster används ökar beräkningstiden med fönstrets storlek. Stora tabeller i kombination med stora fönsterstorlekar bör i allmänhet undvikas.

WRank

WRank() utvärderar raderna i diagrammet i en tabell i laddningsskriptet, och visar för varje rad den relativa placeringen av fältet som utvärderas i laddningsskriptet. Vid utvärdering av tabellen, jämför funktionen resultatet med resultaten på de andra raderna som innehåller den aktuella partitionen och returnerar rangordningen för den aktuella raden inom segmentet.

Partitioner i en tabell

WRank kan bara användas i en **Window**-funktion. **Window**-funktionen måste innehålla en sorteringstyp och ett sorteringsuttryck. Rangordningen tillämpas på sorteringsuttrycket.

Syntax:

```
WRank ([mode[, fmt]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
mode	Anger funktionsresultatets talrepresentation (valfritt).
fmt	Anger funktionsresultatets textrepresentation (valfritt).

Rangordningen returneras som ett dualt värde, som i det fall där varje rad har en unik rangordning är ett heltal mellan 1 och antalet rader i den aktuella partitionen.

Om flera rader har samma rangordningsnummer, kan text- och talrepresentationerna för gruppen kontrolleras med parametrarna **mode** och **fmt**.

mode

Det första argumentet, **mode**, kan ta följande värden:

mode-värden

Värde	Beskrivning
0 (standard)	Om alla rangordningsnummer inom gruppen är lägre än det mellersta värdet i den totala rangordningen, får alla rader i gruppen det lägsta rangordningsnumret inom gruppen. Om alla rangordningsnummer inom gruppen är högre än det mellersta värdet i den totala rangordningen, får alla rader i gruppen det högsta rangordningsnumret inom gruppen. Om rangordningsnumren inom gruppen spänner över det mellersta värdet i den totala rangordningen, får alla rader i gruppen ett värde som motsvarar medelvärdet av det högsta och det lägsta rangordningsnumret i hela partitionen.
1	Lägsta rangordningstalet ges åt alla rader.
2	Medelvärdet ges som rangordningstal åt alla rader.
3	Högsta rangordningstalet ges åt alla rader.
4	Lägsta rangordningstalet på första raden, därefter ökning med ett för varje rad

fmt

Det andra argumentet, **fmt**, kan ta följande värden:

fmt-värden

Värde	Beskrivning
0 (standard)	Lägsta värdet - högsta värdet på alla rader (exempelvis 3 - 4).
1	Lägsta värdet på alla rader.
2	Lägsta värdet på första raden, tom sträng på övriga rader.

Ordningen på rader inom **mode** 4 och **fmt**2 bestäms av laddningsordningen som angivits för tabellfälten.

Exempel - Lägga till ett rangordnat fält

Exempel: lägga till ett rangordnat fält

Laddningsskript

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. Skapa tabellen nedan i Qlik Sense för att se resultatet.

```
Transactions:
Load
*,
Window(WRank(0),customer_id, 'Desc', transaction_amount) as TransactionRanking;
```

```
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

Resultat

Resultat för att lägga till ett rangordnat fält

transaktions-id	transaktionsdatum	transaktionsbelopp	transaktionsantal	Kund-id	storlek	färgkod	Transaktionsrankning
3750	20180830	23.56	2	2038593	L	Röd	4-4
3751	20180907	556,31	6	203521	M	Orange	1-1
3752	20180916	5,75	1	5646471	S	Blå	2-2
3754	20180922	484,21	13	049681	XS	Röd	1-1
3756	20180922	59,18	2	2038593	M	Blå	3-3
3753	20180922	125,00	7	3036491	L	Svart	1-1
3757	20180923	177,42	21	203521	XL	Svart	3-3
3758	20180924	153.42	14	2038593	L	Röd	2-2
3759	20180925	7.42	5	203521	M	Orange	4-4

transaktions-id	transaktionsdatum	transaktionsbelopp	transaktionsantal	Kund-id	storlek	färgkod	Transaktionsrankning
3760	20180925	80.12	18	5646471	M	Blå	1-1
3763	20180926	63.55	12	049681	S	Röd	2-2
3761	20180926	3.42	7	3036491	XS	Svart	2-2
3764	20180927	325.95	8	203521	XL	Svart	2-2
3763	20180927	177.56	10	2038593	L	Blå	1-1

Exempel - lägga till ett rangordnat fält med fmt för att erhålla ett ensiffrigt resultat

Exempel: lägga till ett rangordnat fält med fmt för att erhålla ett ensiffrigt resultat

Laddningsskript

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. Skapa tabellen nedan i Qlik Sense för att se resultatet.

```
Transactions:
Load
*,Window(WRank(0,1),customer_id, 'Desc', transaction_amount) as TransactionRanking;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

Resultat

Resultat för att lägga till ett rangordnat fält med fmt för att erhålla ett ensiffrigt resultat

transaktions-ID	transaktionsdatum	transaktionsbelopp	transaktionskvantitet	kund-id	storlek	färgkod	Transaktionsrankning
3750	20180830	23.56	2	2038593	L	Röd	4
3751	20180907	556,31	6	203521	M	Orange	1
3752	20180916	5,75	1	5646471	S	Blå	2
3754	20180922	484,21	13	049681	XS	Röd	1
3756	20180922	59,18	2	2038593	M	Blå	3
3753	20180922	125,00	7	3036491	L	Svart	1
3757	20180923	177,42	21	203521	XL	Svart	3
3758	20180924	153,42	14	2038593	L	Röd	2
3759	20180925	7.42	5	203521	M	Orange	4
3760	20180925	80.12	18	5646471	M	Blå	1
3763	20180926	63.55	12	049681	S	Röd	2
3761	20180926	3.42	7	3036491	XS	Svart	2
3764	20180927	325.95	8	203521	XL	Svart	2
3763	20180927	177.56	10	2038593	L	Blå	1

Exempel - Lägga till ett rangordnat fält med flera partitioner

Exempel: lägga till ett rangordnat fält med flera partitioner

Laddningsskript

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. Skapa tabellen nedan i Qlik Sense för att se resultatet.

```
Transactions:
Load
*,Window(WRank(0,1),customer_id, size, color_code, 'Desc', transaction_amount) as
TransactionRanking;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

Resultat

Resultat för att lägga till ett rangordnat fält med fmt för att erhålla ett ensiffrigt resultat

transaktions_id	transaktionsdatum	transaktions_belopp	transaktions_kvantitet	kund_id	storlek	färgkod	Transaktionsrankning
3750	20180830	23.56	2	2038593	L	Röd	2
3751	20180907	556,31	6	203521	M	Orange	1
3752	20180916	5,75	1	5646471	S	Blå	1
3754	20180922	484,21	13	049681	XS	Röd	1

transaktions_id	transaktionsdatum	transaktions_belopp	transaktions_kvantitet	kund_id	storlek	färgkod	Transaktionsrankning
3756	20180922	59,18	2	2038593	M	Blå	1
3753	20180922	125,00	7	3036491	L	Svart	1
3757	20180923	177,42	21	203521	XL	Svart	2
3758	20180924	153,42	14	2038593	L	Röd	1
3759	20180925	7,42	5	203521	M	Orange	2
3760	20180925	80.12	18	5646471	M	Blå	1
3763	20180926	63.55	12	049681	S	Röd	1
3761	20180926	3.42	7	3036491	XS	Svart	1
3764	20180927	325,95	8	203521	XL	Svart	1
3763	20180927	177,56	10	2038593	L	Blå	1

Begränsningar

WRank har följande begränsningar:

- Om ditt fmt-värde är 0 och du vill använda textdelen av det duala resultatet för **WRank** måste du använda **Text()** med **Window(WRank)**. Till exempel: `Text(Window(WRank(0), Unit, 'DESC', Age)) as UnitWRankedByAgeText`.

9 Behörighetskontroll för filsystem

Av säkerhetsskäl har Qlik Sense i standardläge inte stöd för sökvägar i dataladdningsskriptet eller funktioner och variabler som exponerar filsystemet.

Eftersom sökvägar hanterades i QlikView är det dock möjligt att inaktivera standardläget och använda det bakåtkompatibla läget om du vill återanvända QlikView-dataladdningsskript.

Att inaktivera standardläge kan skapa en säkerhetsrisk genom att exponera filsystemet.

Inaktivera standardläge (page 1913)

9.1 Säkerhetsaspekter vid anslutning till filbaserade ODBC- och OLE DB-dataanslutningar

ODBC- och OLE DB-dataanslutningar som använder filbaserade drivrutiner kommer att visa sökvägen till den anslutna datafilen i kopplingssträngen. Sökvägen kan exponeras när anslutningen redigeras, i dialogrutan för urval av data eller i vissa SQL-frågor. Detta gäller både i standardläge och bakåtkompatibelt läge.

Om att visa sökvägen till datafilen inte är önskvärt, rekommenderas att ansluta till datafilen med hjälp av en mappdatakoppling om det går.

9.2 Begränsningar i standardläge

Flera satser, variabler och funktioner kan inte användas eller har begränsningar i standardläget. Användning av satser som inte stöds i dataladdningsskriptet ger ett fel när det körs. Felmeddelanden återfinns i skriptloggfilen. Användning av variabler och funktioner som inte stöds ger inga felmeddelanden eller loggfilsposter. Funktionen returnerar i stället NULL.

Det finns ingen indikation på att en variabel, sats eller funktion inte stöds när du redigerar dataladdningsskriptet.

Systemvariabler

Systemvariabler

Variabel	Standardläge	Bakåtkompatibelt läge	Definition
Floppy	Stöds ej	Stöds	Returnerar enhetsbeteckningen för den första diskettenhet som påträffas, vanligen *a:*.

Variabel	Standardläge	Bakåtkompatibelt läge	Definition
CD	Stöds ej	Stöds	Returnerar bokstaven på den första cd-rom-enhet som påträffas. Om ingen cd-rom-enhet påträffas, returneras *c:*.
QvPath	Stöds ej	Stöds	Returnerar sökströngen till Qlik Sense-programfilen.
QvRoot	Stöds ej	Stöds	Returnerar rotkatalogen till Qlik Sense-programfilen.
QvWorkPath	Stöds ej	Stöds	Returnerar sökströngen till den aktuella Qlik Sense applikationen.
QvWorkRoot	Stöds ej	Stöds	Returnerar rotmappen till den aktuella Qlik Sense appen.
WinPath	Stöds ej	Stöds	Returnerar sökströngen till Windows.
WinRoot	Stöds ej	Stöds	Returnerar Windows rotmapp.
$(include=...)	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Variabeln **Include/Must_Include** specificerar en fil som innehåller text som ska inkluderas i skriptet och utvärderas som skriptkod. Den används inte för att lägga till data. Du kan spara delar av din skriptkod i en separat textfil och återanvända den i flera applikationer. Detta är en användardefinierad variabel.

Vanliga skriptsatser

Vanliga skriptsatser

Sats	Standardläge	Bakåtkompatibelt läge	Definition
Binary	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Satsen **binary** används för att ladda data från en annan applikation.
Connect	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	**CONNECT**-satsen används för att ange Qlik Sense-åtkomst till en allmän databas via OLE DB/ODBC-gränssnittet. För ODBC måste datakällan först anges med hjälp av ODBC-administratören.
Directory	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	**Directory**-satsen anger i vilken katalog datafilerna ska sökas i följande **LOAD**-satser tills en ny **Directory**-sats anges.
Execute	Stöds ej	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	**Execute**-satsen använda för att köra andra program medan Qlik Sense laddar data, exempelvis för att göra de konverteringar som krävs.
LOAD from ...	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	**LOAD**-satsen laddar fält från en fil, direkt från data i skriptet, från en tidigare inläst tabell, från en webbsida, från resultatet av en efterföljande **SELECT**-sats eller genom att generera data automatiskt.

Sats	Standardläge	Bakåtkompatibelt läge	Definition
Store into ...	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	**Store**-satsen skapar en QVD-, Parquet-,CSV- eller TXT-fil.

Kontrollsatser i skriptet

Kontrollsatser i skriptet

Sats	Standardläge	Bakåtkompatibelt läge	Definition
For each... filelist mask/dirlist mask	Indata som stöds: Sökväg som använder bibliotekskoppling Returnerade utdata: Bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem Returnerade utdata: Bibliotekskoppling eller filsystemssökväg, beroende på indata	Syntaxen filelist mask skapar en kommaavgränsad lista över alla filer i den aktuella katalog som matchar **filelist mask**. Syntaxen **dirlist mask** skapar en kommaavgränsad lista över alla mappar i den aktuella mappen som matchar filnamnsmasken.

Filfunktioner

Filfunktioner

Funktion	Standardläge	Bakåtkompatibelt läge	Definition
Attribute()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Returnerar metataggarnas värde från olika mediafiler som text.
ConnectString()	Returnerad utdata: Kopplingsnamn för bibliotek	Biblioteksanslutningens namn eller faktiska anslutning, beroende på indata	Returnerar den aktiva kopplingssträngen för ODBC- eller OLE DB-anslutningar.

Funktion	Standardläge	Bakåtkompatibelt läge	Definition
FileDir()	Returnerade utdata: Bibliotekskoppling	Returnerade utdata: Bibliotekskoppling eller filsystemssökväg, beroende på indata	**FileDir**-funktionen returnerar en textsträng som anger sökvägen till katalogen som innehåller den tabellfil som håller på att läsas in.
FilePath()	Returnerade utdata: Bibliotekskoppling	Returnerade utdata: Bibliotekskoppling eller filsystemssökväg, beroende på indata	**FilePath**-funktionen returnerar en textsträng som anger den kompletta sökvägen till den tabellfil som håller på att läsas in.
FileSize()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	**FileSize**-funktionen returnerar ett heltal som anger storleken i byte för filen filename eller, om inget filename angetts, för den tabellfil som håller på att läsas in.
FileTime()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	**FileTime**-funktionen returnerar en tidsmarkör i UTC-format för den senaste ändringen av en angiven fil. Om ingen fil anges returnerar funktionen en tidsmarkör i UTC för den senaste ändringen av den tabellfil som för närvarande är inläst.

Funktion	Standardläge	Bakåtkompatibelt läge	Definition
GetFolderPath()	Stöds ej	Returnerade utdata: Absolut sökväg	Funktionen **GetFolderPath** returnerar värdet av funktionen Microsoft Windows *SHGetFolderPath*. Den här funktionen tar som indata namnet på en Microsoft Windows -mapp och returnerar den fullständiga sökvägen till mappen.
QvdCreateTime()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Denna skriptfunktion returnerar XML-huvudets tidsstämpel från en QVD-fil, om någon sådan finns, annars returneras NULL. I tidsmarkören anges tiden i UTC.
QvdFieldName()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Denna skriptfunktion returnerar namnet på fält nummer **fieldno** i en QVD-fil. Om fältet inte finns returneras NULL.
QvdNoOfFields()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Denna skriptfunktion returnerar antalet fält i en QVD-fil.
QvdNoOfRecords()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Denna skriptfunktion returnerar aktuellt antal poster i en QVD-fil.
QvdTableName()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Denna skriptfunktion returnerar namnet på tabellen i en QVD-fil.

Systemfunktioner

Systemfunktioner

Funktion	Standardläge	Bakåtkompatibelt läge	Definition
DocumentPath()	Stöds ej	Returnerade utdata: Absolut sökväg	Denna funktion returnerar en sträng som innehåller den fullständiga sökvägen till den aktuella Qlik Sense applikationen.

9.3 Inaktivera standardläge

Du kan inaktivera standardläge, eller med andra ord, ställa in bakåtkompatibelt läge, om du vill återanvända QlikView-dataladdningsskriptet som hänvisar till absoluta eller relativa filsökvägar samt bibliotekskopplingar.

Att inaktivera standardläge kan skapa en säkerhetsrisk genom att exponera filsystemet.

Qlik Sense

För Qlik Sense kan standardläget inaktiveras i QMC med hjälp av egenskapen **Standardläge**.

Qlik Sense Desktop

I Qlik Sense Desktop kan du ställa in standardläge/bakåtkompatibelt läge i *Settings.ini*.

Om du installerade Qlik Sense Desktop i den förvalda installationsplatsen, då finns *Settings.ini* i *C:\Users\{user}\Documents\Qlik\Sense\Settings.ini*. Om du installerade Qlik Sense Desktop i en mapp som du valt, då finns *Settings.ini* i mappen *Engine* i installationens sökväg.

Gör följande:

1. Öppna *Settings.ini* i en textredigerare.
2. Ändra *StandardReload=1* till *StandardReload=0*.
3. Spara filen och starta Qlik Sense Desktop.

Qlik Sense Desktop körs nu i bakåtkompatibelt läge.

Inställningar

Följande inställningar är tillgängliga för StandardReload:

- 1 (standardläge)
- 0 (bakåtkompatibelt läge)

10 Skript på diagramnivå

När du ändrar diagramdata använder du en underuppsättning av Qlik Sense-skriptet som består av ett antal satser. En sats kan vara antingen en vanlig skriptsats eller en kontrollsats. Vissa satser kan föregås av prefix.

Vanliga satser används normalt för att modifiera data på ett eller annat sätt. Dessa satser kan skrivas på ett valfritt antal rader i skriptet och måste alltid avslutas med ett semikolon: ";".

Kontrollsatser används normalt för att kontrollera skriptexekveringens flöde. Varje tillägg i en kontrollsats måste hållas inom en och samma rad i skriptet och avslutas med antingen semikolon eller radslut.

Prefix kan sättas framför vissa vanliga satser, men aldrig framför kontrollsatser.

Alla nyckelord i skriptet kan skrivas med antingen versaler eller gemener. Fält- och variabelnamn är dock skiftlägeskänsliga.

I det här delavsnittet kan du hitta en alfabetisk lista över alla skriptsatser, kontrollsatser och prefix som finns tillgängliga i underuppsättningen av skriptet som används vid ändring av diagramdata.

10.1 Kontrollsatser

När du ändrar diagramdata använder du en underuppsättning av Qlik Sense-skriptet som består av ett antal satser. En sats kan vara antingen en vanlig skriptsats eller en kontrollsats.

Kontrollsatser används normalt för att kontrollera skriptexekveringens flöde. Varje tillägg i en kontrollsats måste hållas inom en och samma rad i skriptet och avslutas med antingen semikolon eller radslut.

Prefix används aldrig för kontrollsatser.

Alla nyckelord i skriptet kan skrivas med antingen versaler eller gemener.

Översikt över kontrollsatser för diagrammodifierare

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Call

Kontrollsatsen **call** anropar en subrutin som måste vara definierad av en **sub**-sats på en tidigare plats i skriptet.

```
Call name ( [ paramlist ])
```

Do..loop

Kontrollsatsen **do..loop** är en konstruktion för skriptiteration som exekverar en eller flera satser tills ett logiskt villkor uppfylls.

```
Do-loop [ ( while | until ) condition ] [statements]
[exit do [ ( when | unless ) condition ] [statements]
loop [ ( while | until ) condition ]
```

End

Skriptnyckelordet **End** används för att stänga **If**-, **Sub**- och **Switch**-satser.

Exit

Skriptnyckelordet **Exit** är en del av **Exit Script**-satsen, men kan även användas för att lämna**Do**-, **For**- eller **Sub**-tillägg.

Exit script

Denna kontrollsats avbryter skriptexekveringen. Satsen får förekomma var som helst i skriptet.

```
Exit script[ (when | unless) condition ]
```

For..next

Kontrollsatsen **for..next** är en konstruktion för skriptiteration med en räknevariabel. Satserna inom slingan som innesluts av **for** och **next** exekveras för varje värde i räknevariabeln mellan angivna övre och undre gränser.

```
For..next  counter = expr1 to expr2 [ stepexpr3 ]

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

For each ..next

Kontrollsatsen **for each..next** är en konstruktion för skriptiteration som exekverar en eller flera satser för varje värde i en kommaavgränsad lista. Satserna inom slingan som innesluts av **for** och **next** exekveras för varje värde i listan.

```
For each..next var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

If..then

Kontrollsatsen **if..then** är en skriptvalskonstruktion som gör att skriptexekveringen slår in på olika vägar beroende på ett eller flera logiska villkor.

*Eftersom **if..then**-satsen är en kontrollsats, och som sådan slutar med ett semikolon eller radslut, måste var och en av dess fyra möjliga satser (**if..then**, **elseif..then**, **else** och **end if**) rymmas på en egen rad i skriptet.*

```
If..then..elseif..else..end if condition then

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

Next

Skriptnyckelordet **Next** används för att stänga **For**-slingor.

Sub

Kontrollsatsen **sub..end sub** definierar en subrutin som kan anropas från en **call**-sats.

```
Sub..end sub name [ ( paramlist )] statements end sub
```

Switch

Satsen **switch** är en konstruktion för att göra val i skriptet som tvingar skriptexekveringen att följa olika vägar beroende på värdet hos ett uttryck.

```
Switch..case..default..end switch expression {case valuelist [ statements ]}
[default statements] end switch
```

To

Skriptnyckelordet **To** används i flera skriptsatser.

Call

Kontrollsatsen **call** anropar en subrutin som måste vara definierad av en **sub**-sats på en tidigare plats i skriptet.

Syntax:

```
Call name ( [ paramlist ])
```

Argument:

Argument

Argument	Beskrivning
name	Subrutinens namn.
paramlist	En kommaavgränsad lista över de faktiska parametrar som ska skickas till den underordnade rutinen. Varje parameter i listan kan vara ett fältnamn, ett variabelnamn eller ett godtyckligt uttryck.

Den underordnade rutinen som anropas med hjälp av en **call**-sats måste vara definierad i en **sub**-sats som påträffats tidigare under skriptexekveringen.

Parametrarna kopieras in i subrutinen. Om parametern i **call**-satsen är ett variabelnamn, inte ett uttryck, kopieras dess värde dessutom tillbaka ut igen när subrutinen avslutas.

Begränsningar:

- Eftersom **call**-satsen är en kontrollsats och som sådan slutar med ett semikolon eller ett radslut, får den inte korsa en linjegräns.
- När du definierar en underordnad rutin med `Sub..end sub` inuti en kontrollsats, till exempel `if..then`, kan du bara anropa den underordnade rutinen inifrån samma kontrollsats.

Do-loop

Kontrollsatsen **do..loop** är en konstruktion för skriptiteration som exekverar en eller flera satser tills ett logiskt villkor uppfylls.

Syntax:

```
Do [ ( while | until ) condition ] [statements]
[exit do [ ( when | unless ) condition ] [statements]
loop[ ( while | until ) condition ]
```

*Eftersom **do..loop**-satsen är en kontrollsats, och som sådan slutar med ett semikolon eller radslut, måste var och en av dess tre möjliga komponenter (**do**, **exit do** och **loop**) rymmas på en egen rad i skriptet.*

Argument:

Argument

Argument	Beskrivning
condition	Ett logiskt uttryck som utvärderas till True eller False.

Argument	Beskrivning
statements	Valfri grupp av en eller flera Qlik Sense-skriptsatser.
while / until	Villkorssatsen **while** eller **until** får bara förekomma en gång i en **do..loop**-sats, dvs. antingen efter **do** eller efter **loop**. Varje villkor tolkas bara första gången det påträffas, men utvärderas för varje gång det förekommer i slingan.
exit do	Om en **exit do**-sats påträffas i slingan flyttas skriptexekveringen till den första satsen efter **loop**-satsen som markerar slutet på slingan. En **exit do**-sats kan göras villkorad genom det valbara användandet av ett **when**- eller **unless**-suffix.

End

Skriptnyckelordet **End** används för att stänga **If**-, **Sub**- och **Switch**-satser.

Exit

Skriptnyckelordet **Exit** är en del av **Exit Script**-satsen, men kan även användas för att lämna**Do**-, **For**- eller **Sub**-tillägg.

Exit script

Denna kontrollsats avbryter skriptexekveringen. Satsen får förekomma var som helst i skriptet.

Syntax:

```
Exit Script [ (when | unless) condition ]
```

Eftersom **exit script**-satsen är en kontrollsats och som sådan slutar med ett semikolon eller ett radslut, får den inte korsa en linjegräns.

Argument:

Argument

Argument	Beskrivning
condition	Ett logiskt uttryck som utvärderas till True eller False.
when / unless	En **exit script**-sats kan göras villkorad genom det valbara användandet av en **when**- eller **unless**-sats.

Exempel:

```
//Exit script
Exit Script;

//Exit script when a condition is fulfilled
Exit Script when a=1
```

For..next

Kontrollsatsen **for..next** är en konstruktion för skriptiteration med en räknevariabel. Satserna inom slingan som innesluts av **for** och **next** exekveras för varje värde i räknevariabeln mellan angivna övre och undre gränser.

Syntax:

```
For counter = expr1 to expr2 [ step expr3 ]

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

Uttrycken *expr1*, *expr2* och *expr3* utvärderas bara första gången slingan påbörjas. Värdet på räknevariabeln kan ändras av satser inom slingan, men detta brukar inte göras inom programmering.

Om en **exit for**-sats påträffas i slingan flyttas skriptexekveringen till den första satsen efter **next**-satsen som markerar slutet på slingan. En **exit for**-sats kan göras villkorad genom det valbara användandet av ett **when**- eller **unless**-suffix.

> *Eftersom **for..next**-satsen är en kontrollsats, och som sådan slutar med ett semikolon eller radslut, måste var och en av dess tre möjliga komponenter (**for..to..step**, **exit for** och **next**) rymmas på en egen rad i skriptet.*

Argument:

Argument

Argument	Beskrivning
counter	Ett variabelnamn. Om *counter* anges efter **next**, måste variabelnamnet överensstämma med det variabelnamn som angivits efter motsvarande **for**.
expr1	Ett uttryck som anger det första värdet i *counter* för vilket slingan ska exekveras.
expr2	Ett uttryck som anger det sista värdet i *counter* för vilket slingan ska exekveras.
expr3	Ett uttryck som anger hur många steg värdet i *counter* ska öka varje gång slingan exekverats.
condition	Ett logiskt uttryck som utvärderas till True eller False.
statements	Valfri grupp av en eller flera Qlik Sense-skriptsatser.

For each..next

Kontrollsatsen **for each..next** är en konstruktion för skriptiteration som exekverar en eller flera satser för varje värde i en kommaavgränsad lista. Satserna inom slingan som innesluts av **for** och **next** exekveras för varje värde i listan.

Syntax:
En speciell syntax gör det möjligt att generera listor med namn på filer och mappar i den aktuella mappen.

```
for each var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

Argument:

Argument

Argument	Beskrivning
var	Namnet på en skriptvariabel som får ett nytt värde från lista för varje exekvering av slingan. Om **var** anges efter **next**, måste variabelnamnet överensstämma med det variabelnamn som angivits efter motsvarande **for each**.

Värdet på variabeln **var** kan ändras av satser inom slingan, men detta brukar inte göras inom programmering.

Om en **exit for**-sats påträffas i slingan flyttas skriptexekveringen till den första satsen efter **next**-satsen som markerar slutet på slingan. En **exit for**-sats kan göras villkorad genom det valbara användandet av ett **when**- eller **unless**-suffix.

> *Eftersom **for each..next**-satsen är en kontrollsats, och som sådan slutar med ett semikolon eller radslut, måste var och en av dess tre möjliga komponenter (**for each**, **exit for** och **next**) rymmas på en egen rad i skriptet.*

Syntax:

```
list := item { , item }

item := constant | (expression) | filelist mask | dirlist mask |
fieldvaluelist mask
```

Argument

Argument	Beskrivning
constant	Valfritt tal eller valfri sträng Observera att en sträng som skrivs direkt i skriptet måste omslutas av enkla citattecken. En sträng utan enkla citattecken tolkas som en variabel och då används värdet för variabeln. Tal behöver inte omslutas av enkla citattecken.
expression	Ett godtyckligt uttryck.
mask	En fil- eller katalognamnsmask som får innehålla valfria giltiga filnamnstecken samt standardjokertecknen * och ?. Du kan använda absoluta sökvägar eller sökvägar till lib://.
condition	Ett logiskt uttryck som utvärderas till True eller False.
statements	Valfri grupp av en eller flera Qlik Sense-skriptsatser.
filelist mask	Denna syntax skapar en kommaavgränsad lista över alla filer i den aktuella mappen som matchar filnamnsmasken. *Argumentet har enbart stöd för bibliotekskopplingar i standardläget.*
dirlist mask	Denna syntax skapar en kommaavgränsad lista över alla kataloger i den aktuella katalog som matchar mappnamnsmasken. *Argumentet har enbart stöd för bibliotekskopplingar i standardläget.*
fieldvaluelist mask	Syntaxen itererar genom värdena i ett fält som redan har laddats in i Qlik Sense.

Qlik Kopplingar till webblagringsleverantörer och andra DataFiles-kopplingar stöder inte filtermasker som använder jokertecken (och ?).*

Example 1: Ladda en fillista

```
// LOAD the files 1.csv, 3.csv, 7.csv and xyz.csv
for each a in 1,3,7,'xyz'
   LOAD * from file$(a).csv;
next
```

Example 2: Skapa en fillista på disk

Detta exempel laddar en lista över alla Qlik Sense-relaterade filer i en mapp.

```
sub DoDir (Root)
   for each Ext in 'qvw', 'qva', 'qvo', 'qvs', 'qvc', 'qvf', 'qvd'

      for each File in filelist (Root&'/*.' &Ext)
```

```
        LOAD
           '$(File)' as Name,
           FileSize( '$(File)' ) as Size,
           FileTime( '$(File)' ) as FileTime
        autogenerate 1;

     next File

  next Ext
  for each Dir in dirlist (Root&'/*' )

     call DoDir (Dir)

  next Dir

end sub

call DoDir ('lib://DataFiles')
```

Example 3: Itererar genom värdena i ett fält

Det här exemplet itererar genom listan med laddade värden för FIELD och genererar ett nytt fält, NEWFIELD. För varje värde i FIELD skapas två NEWFIELD-poster.

```
load * inline [
FIELD
one
two
three
];

FOR Each a in FieldValueList('FIELD')
LOAD '$(a)' &'-'&RecNo() as NEWFIELD AutoGenerate 2;
NEXT a
```

Den resulterande tabellen ser ut så här:

Example table

NEWFIELD
one-1
one-2
two-1
two-2
three-1
three-2

If..then..elseif..else..end if

Kontrollsatsen **if..then** är en skriptvalskonstruktion som gör att skriptexekveringen slår in på olika vägar beroende på ett eller flera logiska villkor.

Kontrollsatser används normalt för att kontrollera skriptexekveringens flöde. I ett diagramuttryck, använd den **if** villkorsfunktionen istället.

Syntax:

```
If  condition then

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

Eftersom **if..then**-satsen är en kontrollsats, och som sådan slutar med ett semikolon eller radslut, måste var och en av dess fyra möjliga satser (**if..then**, **elseif..then**, **else** och **end if**) rymmas på en egen rad i skriptet.

Argument:

Argument

Argument	Beskrivning
condition	Ett logiskt uttryck som kan utvärderas som True eller False.
statements	Valfri grupp av en eller flera Qlik Sense-skriptsatser.

Example 1:

```
if a=1 then
      LOAD * from abc.csv;

      SQL SELECT e, f, g from tab1;
end if
```

Example 2:

```
if a=1 then; drop table xyz; end if;
```

Example 3:

```
if x>0 then
      LOAD * from pos.csv;
elseif x<0 then
      LOAD * from neg.csv;
else
      LOAD * from zero.txt;
end if
```

Next

Skriptnyckelordet **Next** används för att stänga **For**-slingor.

Sub..end sub

Kontrollsatsen **sub..end sub** definierar en subrutin som kan anropas från en **call**-sats.

Syntax:

```
Sub name [ ( paramlist )] statements end sub
```

Argument kopieras in i subrutinen. Om motsvarande faktiska parameter i **call**-satsen är ett variabelnamn, kopieras de dessutom tillbaka ut igen när subrutinen avslutas.

Om en underordnad rutin har fler formella parametrar än antalet faktiska parametrar som skickas från en **call**-sats, initialiseras de extra parametrarna till NULL och kan användas som lokala variabler inom den underordnade rutinen.

Argument:

Argument

Argument	Beskrivning
name	Subrutinens namn.
paramlist	En kommaavgränsad lista över variabelnamn för den underordnade rutinens formella parametrar. Dessa kan användas som valfri variabel inuti subrutinen.
statements	Valfri grupp av en eller flera Qlik Sense-skriptsatser.

Begränsningar:

- Eftersom **sub**-satsen är en kontrollsats, och som sådan slutar med ett semikolon eller radslut, måste var och en av dess två möjliga satser (**sub** och **end sub**) rymmas på en egen rad i skriptet.

- När du definierar en underordnad rutin med `Sub..end sub` inuti en kontrollsats, till exempel `if..then`, kan du bara anropa den underordnade rutinen inifrån samma kontrollsats.

Example 1:

```
Sub INCR (I,J)

I = I + 1

Exit Sub when I < 10

J = J + 1

End Sub

Call INCR (X,Y)
```

Example 2: - parameteröverföring

```
Sub ParTrans (A,B,C)

A=A+1

B=B+1

C=C+1

End Sub

A=1

X=1

C=1

Call ParTrans (A, (X+1)*2)
```

Resultatet av exemplet ovan blir följande (lokalt, inuti subrutinen): A initialiseras till 1, B initialiseras till 4 och C initialiseras till NULL.

När man lämnar subrutinen får den globala variabeln A värdet 2 (kopieras tillbaka från subrutinen). Den andra faktiska parametern “(X+1)*2” kopieras inte tillbaka eftersom den inte är någon variabel. Slutligen påverkas den globala variabeln C inte av subrutinens anrop.

Switch..case..default..end switch

Satsen **switch** är en konstruktion för att göra val i skriptet som tvingar skriptexekveringen att följa olika vägar beroende på värdet hos ett uttryck.

Syntax:

```
Switch expression {case valuelist [ statements ]} [default statements] end switch
```

*Eftersom **switch**-satsen är en kontrollsats, och som sådan slutar med ett semikolon eller radslut, måste var och en av dess fyra möjliga satser (**switch**, **case**, **default** och **end switch**) rymmas på en egen rad i skriptet.*

Argument:

Argument

Argument	Beskrivning
expression	Ett godtyckligt uttryck.
valuelist	En kommaavgränsad lista av värden med vilka uttryckets värde jämförs. Skriptexekveringen fortsätter med satserna efter den första gruppen där värdet i valuelist motsvarar uttryckets värde. Varje värde i valuelist kan vara ett godtyckligt uttryck. Om det inte finns någon motsvarighet i någon **case**-sats, körs satserna under **default**, om de anges.
statements	Valfri grupp av en eller flera Qlik Sense-skriptsatser.

Exempel:

```
Switch I

Case 1

LOAD '$(I): CASE 1' as case autogenerate 1;

Case 2

LOAD '$(I): CASE 2' as case autogenerate 1;

Default

LOAD '$(I): DEFAULT' as case autogenerate 1;

End Switch
```

To

Skriptnyckelordet **To** används i flera skriptsatser.

10.2 Prefix

Prefix kan sättas framför vissa vanliga satser, men aldrig framför kontrollsatser.

Alla nyckelord i skriptet kan skrivas med antingen versaler eller gemener. Fält- och variabelnamn är dock skiftlägeskänsliga.

Översikt över prefix för diagrammodifierare

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Add

Add-prefixet kan adderas till valfri **LOAD**- eller **SELECT**-sats i skriptet för att specificera att det ska lägga till poster i en annan tabell. Det specificerar även att satsen ska köras i en delvis laddning. **Add**-prefixet kan även användas i en **Map**-sats.

```
Add [only] [Concatenate[(tablename )]] (loadstatement | selectstatement)
```

```
Add [ Only ] mapstatement
```

Replace

Prefixet **Replace** kan adderas till valfri **LOAD**- eller **SELECT**-sats i skriptet för att specificera att den laddade tabellen ska ersätta en annan tabell. Det specificerar även att satsen ska köras i en delvis laddning. **Replace**-prefixet kan även användas i en **Map**-sats.

```
Replace [only] [Concatenate[(tablename) ]] (loadstatement | selectstatement)
```

```
Replace [only] mapstatement
```

Add

I ett diagrammodifierande sammanhang används **Add**-prefixet med **LOAD** för att lägga till värden till tabellen *HC1*, som representerar hyperkuben som beräknas av Qliks associativa motor. Du kan ange en eller flera kolumner. Saknade värden fylls automatiskt i av Qliks associativa motor.

Syntax:

```
Add loadstatement
```

Exempel:

Det här exemplet lägger till två rader i kolumnerna *Datum* och *Försäljning* från inline-satsen

```
Add Load
x as Dates,
y as Sales
Inline
[
Dates,Sales
2001/09/1,1000
2001/09/10,-300
]
```

Replace

I ett diagrammodifierande sammanhang ändrar **Replace**-prefixet alla värden i tabellen *HC1* med ett beräknat värde som definieras av skriptet.

Syntax:

```
Replace loadstatement
```

Exempel:

Detta exempel skriver över alla värden i kolumnen `z` med summan av `x` och `y`.

```
Replace Load
x+y as z
Resident HC1;
```

10.3 Vanliga satser

Vanliga satser används normalt för att modifiera data på ett eller annat sätt. Dessa satser kan skrivas på ett valfritt antal rader i skriptet och måste alltid avslutas med ett semikolon: ";".

Alla nyckelord i skriptet kan skrivas med antingen versaler eller gemener. Fält- och variabelnamn är dock skiftlägeskänsliga.

Översikt över regelbundna satser för diagrammodifierare

Varje funktion beskrivs mer ingående efter översikten. Du kan även klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

LOAD

I ett diagrammodifierande sammanhang laddar **LOAD**-satsen ytterligare data till hyperkuben från data definierade i skriptet eller från en tidigare laddad tabell. Det går även att ladda data från analytiska kopplingar.

> *Satsen **LOAD** måste ha antingen prefixet **Replace** eller **Add**, annars kommer den att avvisas.*

```
Add | Replace Load [ distinct ] fieldlist

(

inline data [ format-spec ] |

resident table-label

) | extension pluginname.functionname([script] tabledescription)]
```

```
[ where criterion | while criterion ]
```

```
[ group by groupbyfieldlist ]
```

```
[order by orderbyfieldlist ]
```

Let

let-satsen är ett komplement till **set**-satsen som används för att definiera skriptvariabler. I motsats till **set**-satsen utvärderas **let**-satsens uttryck till höger om '=' när skriptet körs, innan det tilldelas variabeln.

```
Let variablename=expression
```

Set

Satsen **set** används för att definiera skriptvariabler. Dessa kan användas för att ersätta strängar, sökvägar, drivrutiner med mera.

```
Set variablename=string
```

Put

Satsen **Put** används för att ställa in ett numeriskt värde i hyperkuben.

HCValue

Funktionen **HCValue** används för att hämta värden i en rad i en angiven kolumn.

Load

I ett diagrammodifierande sammanhang laddar **LOAD**-satsen ytterligare data till hyperkuben från data definierade i skriptet eller från en tidigare laddad tabell. Det går även att ladda data från analytiska kopplingar.

*Satsen **LOAD** måste ha antingen prefixet **Replace** eller **Add**, annars kommer den att avvisas.*

Syntax:

```
Add | Replace LOAD fieldlist
```

```
(
```

```
inline data [ format-spec ] |
```

```
resident table-label
```

```
) | extension pluginname.functionname([script] tabledescription)]
```

```
[ where criterion | while criterion ]
```

```
[ group by groupbyfieldlist ]
```

```
[order by orderbyfieldlist ]
```

Argument:

Argument

Argument	Beskrivning
fieldlist	*fieldlist ::= (* | field{, * | field })* En lista på de fält som ska läsas in. Genom att använda * som fältlista anger man alla fält i tabellen. *field ::= (fieldref | expression)* [**as** *aliasname]* Fältdefinitionen måste alltid innehålla en litteral, en referens till ett befintligt fält eller ett uttryck. *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [**I** | **U** | **R** | **B** | **T**])* *fieldname* är en text som är identisk med fältnamnet i tabellen. Observera att fältnamnet måste omslutas av raka, dubbla citationstecken eller av hakparenteser om det innehåller exempelvis mellanslag. Ibland är fältnamn inte explicit tillgängliga. Då används en annan metod: *@fieldnumber* representerar fältnumret i en avgränsad tabellfil. Det måste vara ett positivt heltal som föregås av "@". Numreringen går alltid från 1 och upp till antalet fält. *@startpos*:*endpos* representerar första och sista teckenposition för fältet i en fixpostfil med fasta fältpositioner. Positionerna måste vara positiva heltal. De två talen måste föregås av "@" och avgränsas av kolon. Numreringen går alltid från 1 och upp till antalet positioner. I det sista fältet används **n** som slutposition. • Om*@startpos*:*endpos* omedelbart följs av tecknen **I** eller**U**, tolkas inlästa bytes som binärt signerade (**I**) eller osignerade (**U**) heltal (Intel byte order). Antalet positioner som läses måste vara 1, 2 eller 4. • Om *@startpos*:*endpos* omedelbart följs av tecknet **R**, tolkas inlästa bytes som ett binärt reellt tal (IEEE 32-bitars eller 64-bitars floating point). Antalet positioner som läses in måste vara 4 eller 8. • Om *@startpos*:*endpos* omedelbart följs av tecknet **B**, tolkas inlästa bytes som BCD (Binary Coded Decimal)-tal enligt COMP-3-standard. Valfritt antal bytes kan anges. *expression* kan vara en numerisk funktion eller en strängfunktion baserad på ett eller flera andra fält i samma tabell. För ytterligare information, se uttryckens syntax. **as** används för att döpa om fält.

Argument	Beskrivning
inline	**inline** används om data ska skrivas i skriptet, och inte läsas in från en fil. *data ::= [text]* Data som läses in med en **inline**-sats måste inledas och avslutas med dubbla citationstecken eller hakparenteser. Texten mellan dessa tolkas som om den vore skriven i en fil. Precis som du infogar en ny rad i en textfil bör du göra det även i texten i en **inline**-sats. Klicka på vanligt sätt på returtangenten när du skriver skriptet. Antalet kolumner definieras av den första raden. *format-spec ::= (fspec-item {, fspec-item })* Formatspecifikationen består av en lista med flera formatspecifikatorer inom parentes. Mer information finns i *Formatspecifikatorer (page 191)*.
resident	**resident** används om data ska läsas in från en tidigare inläst tabell. *table label* är en etikett som föregår **LOAD**-satsen som skapade den ursprungliga tabellen. Etiketten ska avslutas med kolon.
extension	Du kan ladda data från analytiska kopplingar. Du måste använda **extension**-satsen för att anropa en funktion som definierats i komplement på serversidan, (SSE) -instickspprogrammet, eller utvärdera ett skript. Du kan skicka en enstaka tabell till SSE-instickspprogrammet. En datatabell returneras. Fälten kallas Field1, Field2 och så vidare om instickspprogrammet inte specificerar namnen på de fält som returneras. `Extension pluginname.functionname( tabledescription );` • Ladda data med en funktion i ett SSE-instickspprogram *tabledescription ::= (table { ,tablefield})* Om du inte anger tabellfält används fälten i laddningsordning. • Ladda data genom att utvärdera ett skript i ett SSE-instickspprogram *tabledescription ::= (script, table { ,tablefield})* **Hantering av datatyp i tabellfältdefinitionen** Datatyper identifieras automatiskt i analytiska kopplingar. Om data inte har numeriska värden och minst en icke-NULL textsträng betraktas fältet som text. I alla andra fall betraktas det som numeriskt. Du kan tvinga datatypen genom att radbryta ett fältnamn med **String()** eller **Mixed()**. • **String()** tvingar fältet att vara text. Om fältet är numeriskt extraheras textdelen av det duala värdet. Ingen konvertering görs. • **Mixed()** tvingar fältet att vara dualt. **String()** eller **Mixed()** kan inte användas utanför **komplement**-tabellfältdefinitioner och du kan inte använda andra Qlik Sense-funktioner i en tabellfältdefinition.

Argument	Beskrivning
where	**where** är ett tillägg som används för att tala om huruvida en post ska inkluderas i valet eller inte. Valet inkluderas om *criterion* är True. *criterion* är ett logiskt uttryck.
while	**while** är en sats som används för att tala om när en post ska läsas in upprepade gånger. Samma post läses in så länge *criterion* är True. För att vara användbar måste en **while**-sats typiskt sett innehålla en **IterNo( )**-funktion. *criterion* är ett logiskt uttryck.
group by	**group by** används för att definiera över vilka fält data ska aggregeras (grupperas). Aggregeringsfälten ska på något sätt inlemmas i de uttryck som läses in. Inga andra fält än aggregeringsfälten får användas utanför aggregeringsfunktionerna i inläsningsuttrycken. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	**order by** är en sats som används för att sortera poster i en resident tabell innan de bearbetas av **load**-satsen. Den residenta tabellen kan sorteras efter ett eller flera fält, i stigande eller sjunkande ordning. Sortering görs i första hand efter talvärde, i andra hand efter nationell sorteringsspråkvariant. Satsen är endast användbar när datakällan är en resident tabell. Ordningsföljdsfälten anger efter vilket fält resident-tabellen sorteras. Ange fältets namn eller dess nummer i resident-tabellen (det första fältet får nummer 1). orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } *sortorder* är antingen *asc* för stigande eller *desc* för fallande. Om ingen *sortorder* anges, antas *asc*. *fieldname*, *path*, *filename* och *aliasname* är textsträngar som representerar det namnen antyder. Valfritt fält i källtabellen kan användas som *fieldname*. Däremot hamnar fält som har skapats med hjälp av as-satsen (*aliasname*) utanför och kan inte användas inuti samma **load**-sats.

Let

let-satsen är ett komplement till **set**-satsen som används för att definiera skriptvariabler. I motsats till **set**-satsen utvärderas **let**-satsens uttryck till höger om '=' när skriptet körs, innan det tilldelas variabeln.

Syntax:

```
Let variablename=expression
```

Exempel och resultat:

Exempel	Resultat

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	`$(x)` utvärderas som ' 3+4 '
	
	`$(y)` utvärderas som ' 7 '
	
	`$(z)` utvärderas som ' 8 '
	
	Observera skillnaden mellan **Set**- och **Let**-satserna. **Set**-satsen tilldelar strängen '3+4' till variabeln, medan **Let**-satsen utvärderar strängen och tilldelar variabeln värdet 7.

Let T=now( );	`$(T)` får ett värde som motsvarar aktuell tid.

Set

Satsen **set** används för att definiera skriptvariabler. Dessa kan användas för att ersätta strängar, sökvägar, drivrutiner med mera.

Syntax:

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Put

Satsen **put** används för att ange ett numeriskt värde i hyperkuben.

Åtkomst till kolumnerna kan göras via etiketter. Du kan också komma åt kolumner och rader efter deklarationsordning. Se exemplen nedan för mer information.

*Du kan inte ändra diagramdimensioner med satsen **put**.*

```
put column(position)=value
```

Åtkomst till kolumnerna kan göras via etiketter.

Det här exemplet anger värdet `1` i den första positionen i kolumnen med etiketten *Sales*.

```
Put Sales(1) = 1;
```

Du kan komma åt mått-kolumner efter deklarationsordning med formatet `#hc1.measure` för mått.

Det här exemplet anger värdet 1000 i den tionde positionen i den slutliga sorterade hyperkuben.

```
Put #hc1.measure.2(10) = 1000;
```

HCValue

Funktionen **HCValue** används för att hämta värden i en rad i en angiven kolumn.

Syntax:

```
HCValue(column,position)
```

Example 1:

Det här exemplet returnerar värdet vid den första positionen i kolumnen med etiketten ”Försäljning”.

```
HCValue(Sales,1)
```

Example 2:

Detta exempel returnerar värdet på den tionde positionen av den sorterade hyperkuben.

```
HCValue(#hc1.measure2,10)
```

Example 3:

Det här exemplet returnerar värdet på den femte raden i den tredje dimensionen.

```
HCValue(#hc1.dimension.3,5)
```

Om det inte finns några sådana dimensioner eller uttryck, i värde eller etiketter, returneras ett fel som indikerar att kolumnen inte hittades. Om indexet för kolumnen är utanför gränserna, returneras NULL.

11 QlikView-funktioner och -satser som inte stöds i Qlik Sense

De flesta funktioner och satser som kan användas i QlikView-laddningsskript och diagramuttryck stöds också i Qlik Sense, men det finns några undantag, som beskrivs här.

11.1 Skriptsatser som inte stöds i Qlik Sense

QlikView skriptsatser som inte stöds i Qlik Sense

Sats	Kommentarer
Command	Använd **SQL** i stället.
InputField	

11.2 Funktioner som inte stöds i Qlik Sense

I denna lista beskrivs QlikView-skript och diagramfunktioner som inte stöds i Qlik Sense.

- **GetCurrentField**
- **GetExtendedProperty**
- **Input**
- **InputAvg**
- **InputSum**
- **MsgBox**
- **NoOfReports**
- **ReportComment**
- **ReportId**
- **ReportName**
- **ReportNumber**

11.3 Prefix som inte stöds i Qlik Sense

I denna lista beskrivs QlikView-prefix som inte stöds i Qlik Sense.

- Bundle
- Image_Size
- Info

12 Funktioner och satser som inte rekommenderas i Qlik Sense

De flesta funktioner och satser som kan användas i QlikView-laddningsskript och diagramuttryck stöds även i Qlik Sense, men en del av dem rekommenderas inte för användning i Qlik Sense. De finns även funktioner och satser tillgängliga i tidigare versioner av Qlik Sense som har fasats ut.

Av kompatibilitetsskäl kommer de fortfarande att fungera som avsett, men man bör uppdatera koden i enlighet med rekommendationerna i det här avsnittet, eftersom de kan komma att tas bort i framtida versioner.

12.1 Skriptsatser som inte rekommenderas i Qlik Sense

Den här tabellen innehåller skriptsatser som inte rekommenderas för användning i Qlik Sense.

Skriptsatser som inte rekommenderas

Sats	Rekommendation
Command	Använd **SQL** i stället.
CustomConnect	Använd **Custom Connect** i stället.

12.2 Skriptsatsparametrar som inte rekommenderas i Qlik Sense

I den här tabellen beskrivs skriptsatsparametrar som inte rekommenderas för användning i Qlik Sense.

Skriptsatsparametrar som inte rekommenderas

Sats	Parametrar
Buffer	Använd **Incremental** i stället för: • **Inc** (rekommenderas ej) • **Incr** (rekommenderas ej)

Sats	Parametrar
LOAD	Följande parameternyckelord genereras av QlikView-filomvandlingsguider. Funktionerna behålls när data laddas in igen, men Qlik Sense tillhandahåller inte vägledning/guider för att generera satsen med dessa parametrar: • Bottom • Cellvalue • Col • Colmatch • Colsplit • Colxtr • Compound • Contain • Equal • Every • Expand • Filters • Intarray • Interpret • Length • Longer • Numerical • Pos • Remove • Rotate • Row • Rowcnd • Shorter

12.3 Funktioner som inte rekommenderas i Qlik Sense

I den här tabellen beskrivs skript- och diagramfunktioner som inte rekommenderas för användning i Qlik Sense.

Funktioner som inte rekommenderas

Funktion	Rekommendation
NumAvg **NumCount** **NumMax** **NumMin** **NumSum**	Använd Range-funktioner i stället. *Intervallfunktioner (page 1576)*
Color() **QliktechBlue** **QliktechGray**	Använd andra färgfunktioner i stället. **QliktechBlue()** kan ersättas med **RGB(8, 18, 90)** och **QliktechGray** kan ersättas med **RGB(158, 148, 137)** för att få samma färger. *Färgfunktioner (page 619)*
QlikViewVersion	Använd **EngineVersion** i stället. *EngineVersion (page 1874)*
ProductVersion	Använd **EngineVersion** i stället. *EngineVersion (page 1874)*
QVUser	
Year2Date	Använd **YearToDate** i stället.
Vrank	Använd **Rank** i stället.
WildMatch5	Använd **WildMatch** i stället.

Kvalificeraren ALL

I QlikView kan kvalificeraren **ALL** förekomma framför ett uttryck. Detta motsvaras av att använda **{1} TOTAL**. I ett sådant fall görs beräkningen över alla värden i fältet i dokumentet. Diagramdimensioner och aktuella urval beaktas inte. Samma värde returneras alltid oavsett det logiska tillståndet i dokumentet. Om kvalificeraren **ALL** används kommer ett set-uttryck inte att kunna användas eftersom kvalificeraren **ALL** i sig själv definierar ett set. För att kunna erbjuda bakåtkompatibilitet kommer kvalificeraren **ALL** fortfarande att fungera i denna Qlik Sense-version. Det är möjligt att den kommer att tas bort i framtida versioner.