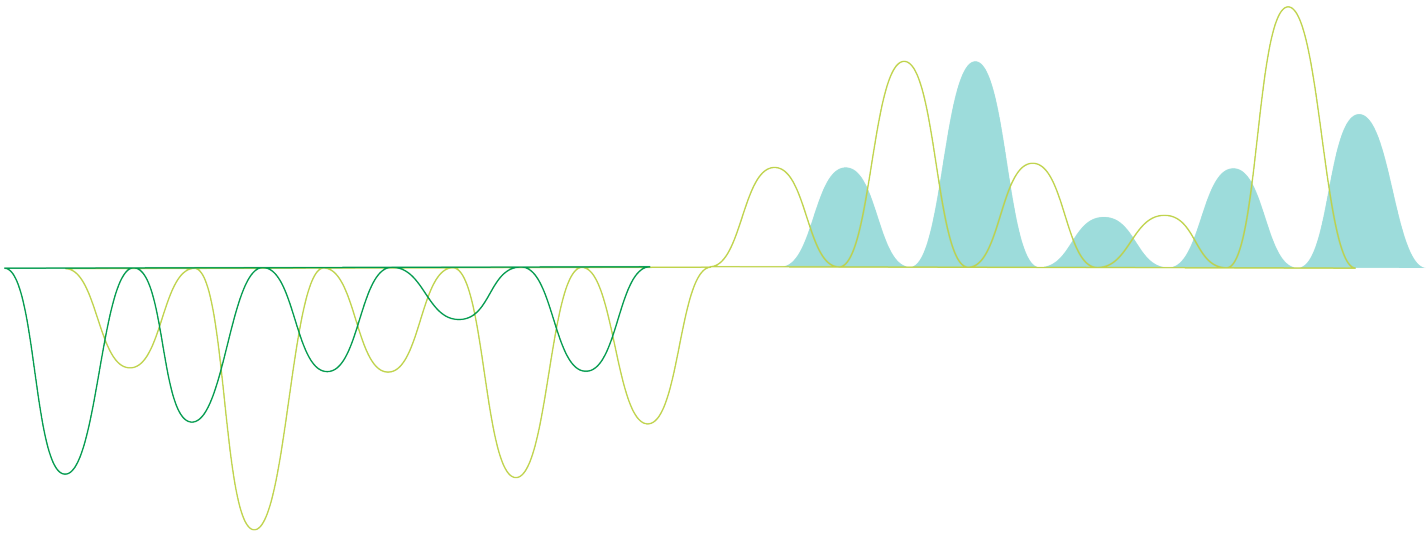
Sintaxis de script y funciones de gráficos

Qlik Sense®
February 2024
Copyright © 1993-aaaa} QlikTech International AB. Reservados todos los derechos.



© aaaa 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 ¿Qué es Qlik Sense? 16
1.1 ¿Qué se puede hacer en Qlik Sense? 16
1.2 ¿Cómo funciona Qlik Sense? 16
El modelo de apps 16
La experiencia asociativa 16
Colaboración y movilidad 16
1.3 ¿Cómo se puede instalar Qlik Sense? 17
Qlik Sense Desktop 17
Qlik Sense Enterprise 17
1.4 Cómo administrar y gestionar un sitio Qlik Sense 17
1.5 Ampliar Qlik Sense y adaptarlo a sus propias necesidades 17
Creación de extensiones y mashups 17
Crear clientes 17
Crear herramientas del servidor 17
Conexión con otras fuentes de datos 17
2 Descripción general de la sintaxis de script 18
2.1 Introducción a la sintaxis de script 18
2.2 ¿Qué es el formalismo Backus-Naur? 18
3 Sentencias de script y palabras clave 20
3.1 Sentencias de control de script 20
Descripción general de las sentencias de control de script 20
Call 22
Do..loop 23
End 24
Exit 24
Exit script 25
For..next 25
For each..next 27
If..then..elseif..else..end if 30
Next 31
Sub..end sub 31
Switch..case..default..end switch 33
To 34
3.2 Prefijos de script 34
Descripción general de los prefijos de script 34
Add 38
Buffer 39
Concatenate 41
Crosstable 47
First 56
Generic 58
Hierarchy 64
HierarchyBelongsTo 67
Inner 68
IntervalMatch 69
Join 73
Keep 83

Left ... 84
Correspondencia ... 85
Merge ... 86
NoConcatenate ... 91
Only ... 100
Outer ... 100
carga parcial ... 101
Replace ... 104
Right ... 106
Sample ... 107
Semantic ... 110
Unless ... 114
When ... 119
3.3 Sentencias habituales de script ... 126
Descripción general de las sentencias habituales de script ... 126
Alias ... 132
AutoNumber ... 133
Binary ... 136
Comment field ... 138
Comment table ... 139
Connect ... 139
Declare ... 141
Derive ... 144
Direct Query ... 145
Directory ... 150
Disconnect ... 151
Drop ... 152
Drop table ... 153
Execute ... 154
Field/Fields ... 155
FlushLog ... 155
Force ... 155
From ... 157
Load ... 158
Let ... 176
Loosen Table ... 176
Map ... 177
NullAsNull ... 178
NullAsValue ... 178
Qualify ... 179
Rem ... 180
Rename ... 181
Search ... 183
Section ... 184
Select ... 184
Set ... 187
Sleep ... 187
SQL ... 187

SQLColumns ... 188
SQLTables ... 189
SQLTypes ... 189
Star ... 191
Store ... 192
Table/Tables ... 197
Tag ... 198
Trace ... 198
Unmap ... 199
Unqualify ... 200
Untag ... 200
3.4 Directorio de trabajo ... 201
Directorio de trabajo de Qlik Sense Desktop ... 201
Directorio de trabajo de Qlik Sense ... 201
4 Trabajar con variables en el editor de carga de datos ... 202
4.1 General ... 202
4.2 Definir una variable ... 202
4.3 Eliminar una variable ... 203
4.4 Cargar un valor de variable como un valor de campo ... 203
4.5 Cálculo de variables ... 203
4.6 Variables de sistema ... 204
Descripción general de las variables de sistema ... 204
CreateSearchIndexOnReload ... 207
HidePrefix ... 207
HideSuffix ... 208
Include ... 208
OpenUrlTimeout ... 209
StripComments ... 210
Verbatim ... 210
4.7 Variables de manejo de valores ... 210
Descripción general de las variables de manejo de valores ... 210
NullDisplay ... 211
NullInterpret ... 211
NullValue ... 212
OtherSymbol ... 212
4.8 Variables de interpretación numérica ... 212
Formato de moneda ... 213
Formato numérico ... 213
Formato de tiempo ... 213
BrokenWeeks ... 215
DateFormat ... 216
DayNames ... 222
DecimalSep ... 226
FirstWeekDay ... 229
LongDayNames ... 233
LongMonthNames ... 236
MoneyDecimalSep ... 240

MoneyFormat 244
MoneyThousandSep 248
MonthNames 252
NumericalAbbreviation 258
ReferenceDay 258
ThousandSep 263
TimeFormat 269
TimestampFormat 270
4.9 Direct Discovery variables 273
Variables de sistema de Direct Discovery 273
Variables query banding de Teradata 274
Direct DiscoveryVariables de carácter de 275
Variables de interpretación numérica de Direct Discovery 276
4.10 Variables de error 277
Descripción general de las variables de error 277
ErrorMode 278
ScriptError 278
ScriptErrorCount 279
ScriptErrorList 279
5 Expresiones de script 280
6 Expresiones de gráfico 281
6.1 Definir el ámbito de agregación 281
6.2 Análisis de conjuntos 284
Expresiones de conjunto 284
Ejemplos 285
Conjuntos naturales 285
Identificadores de conjunto 288
Operadores de conjunto 289
Modificadores de conjunto 290
Expresiones de conjunto internas y externas 312
Tutorial - Crear una expresión de conjunto 314
Sintaxis para expresiones de conjuntos 324
6.3 Sintaxis general para expresiones de gráficos 324
6.4 Sintaxis general para agregaciones: 325
7 Operadores 326
7.1 Operadores de bit 326
7.2 Operadores lógicos 327
7.3 Operadores numéricos 327
7.4 Operadores relacionales 328
7.5 Operadores de cadena 330
& 330
like 330
8 Funciones de script y de gráfico 331
8.1 Conexiones analíticas para extensiones del lado del servidor (SSE) 331
8.2 Funciones de agregación 331
Uso de las funciones de agregación en el script de carga de datos 332

Uso de las funciones de agregación en expresiones de gráficos ... 332
Cómo se calculan las agregaciones ... 332
Agregación de campos clave ... 332
Funciones básicas de agregación ... 333
Funciones de agregación de contador ... 356
Funciones de agregación financiera ... 374
Funciones de agregación estadística ... 403
Funciones estadísticas de prueba ... 474
Funciones de agregación de cadena ... 543
Funciones para dimensiones sintéticas ... 556
Agregaciones anidadas ... 559
8.3 Aggr - función de gráfico ... 559
Ejemplos: Expresiones de gráfico que usan Aggr ... 562
8.4 Funciones de color ... 566
Funciones de colores predefinidos ... 568
ARGB ... 569
RGB ... 569
HSL ... 571
8.5 Funciones condicionales ... 572
Descripción general de las funciones condicionales ... 572
alt ... 573
class ... 574
coalesce ... 576
if ... 577
match ... 580
mixmatch ... 584
pick ... 587
wildmatch ... 587
8.6 Funciones de contador ... 590
Descripción general de las funciones de contador ... 590
autonumber ... 591
autonumberhash128 ... 594
autonumberhash256 ... 596
IterNo ... 598
RecNo ... 599
RowNo ... 600
RowNo - función de gráfico ... 601
8.7 Funciones de fecha y hora ... 604
Descripción general de las funciones de fecha y hora ... 604
addmonths ... 613
addyears ... 623
age ... 630
converttolocaltime ... 632
day ... 636
dayend ... 642
daylightsaving ... 650
dayname ... 651
daynumberofquarter ... 653

daynumberofyear 659
daystart 666
firstworkdate 673
GMT 675
hour 679
inday 682
indaytotime 691
inlunarweek 701
inlunarweektodate 713
inmonth 725
inmonths 733
inmonthstodate 747
inmonthtodate 760
inquarter 770
inquartertodate 783
inweek 796
inweektodate 812
inyear 826
inyeartodate 839
lastworkdate 852
localtime 861
lunarweekend 865
lunarweekname 876
lunarweekstart 889
makedate 901
maketime 907
makeweekdate 914
minute 922
month 928
monthend 934
monthname 944
monthsend 951
monthsname 964
monthsstart 977
monthstart 990
networkdays 1000
now 1010
quarterend 1017
quartername 1031
quarterstart 1043
second 1054
setdateyear 1060
setdateyearmonth 1062
timezone 1064
today 1064
UTC 1070
week 1070
weekday 1087

weekend 1096
weekname 1108
weekstart 1123
weekyear 1135
year 1145
yearend 1152
yearname 1164
yearstart 1176
yeartodate 1188
8.8 Funciones exponenciales y logarítmicas 1204
8.9 Funciones de campo 1205
Funciones de contador 1205
Funciones de campo y selección 1206
GetAlternativeCount - función de gráfico 1206
GetCurrentSelections - función de gráfico 1207
GetExcludedCount - función de gráfico 1209
GetFieldSelections - función de gráfico 1210
GetNotSelectedCount - función de gráfico 1213
GetObjectDimension - función de gráfico 1213
GetObjectField - función de gráfico 1214
GetObjectMeasure - función de gráfico 1215
GetPossibleCount - función de gráfico 1215
GetSelectedCount - función de gráfico 1217
8.10 Funciones de archivo 1218
Visión global de las funciones de archivo 1218
Attribute 1220
ConnectString 1229
FileBaseName 1229
FileDir 1230
FileExtension 1230
FileName 1230
FilePath 1231
FileSize 1231
FileTime 1232
GetFolderPath 1233
QvdCreateTime 1234
QvdFieldName 1235
QvdNoOfFields 1236
QvdNoOfRecords 1237
QvdTableName 1238
8.11 Funciones financieras 1239
Visión global de las funciones financieras 1240
BlackAndSchole 1240
FV 1241
nPer 1242
Pmt 1243
PV 1244
Rate 1245

8.12 Funciones de formato ... 1246
Descripción general de las funciones de formato ... 1246
ApplyCodepage ... 1247
Date ... 1248
Dual ... 1250
Interval ... 1252
Money ... 1253
Num ... 1254
Time ... 1257
Timestamp ... 1258
8.13 Funciones numéricas generales ... 1259
Descripción general de las funciones numéricas generales ... 1260
Funciones de combinación y permutación ... 1260
Funciones de módulo ... 1261
Funciones de paridad ... 1261
Funciones de redondeo ... 1261
BitCount ... 1262
Ceil ... 1262
Combin ... 1263
Div ... 1264
Even ... 1264
Fabs ... 1265
Fact ... 1265
Floor ... 1266
Fmod ... 1267
Frac ... 1268
Mod ... 1268
Odd ... 1269
Permut ... 1269
Round ... 1270
Sign ... 1271
8.14 Funciones geoespaciales ... 1272
Descripción general de las funciones geoespaciales ... 1272
GeoAggrGeometry ... 1274
GeoBoundingBox ... 1275
GeoCountVertex ... 1275
GeoGetBoundingBox ... 1276
GeoGetPolygonCenter ... 1276
GeoInvProjectGeometry ... 1277
GeoMakePoint ... 1278
GeoProject ... 1278
GeoProjectGeometry ... 1279
GeoReduceGeometry ... 1280
8.15 Funciones de interpretación ... 1281
Descripción general de las funciones de interpretación ... 1281
Date# ... 1283
Interval# ... 1284
Money# ... 1284

Num# 1286
Text 1287
Time# 1287
Timestamp# 1288
8.16 Funciones inter-registro 1289
Funciones de fila 1290
Funciones de columna 1290
Funciones de campo 1291
Funciones de la tabla pivotante 1291
Funciones inter-registro en el script de carga de datos 1292
Above - función de gráfico 1293
Below - función de gráfico 1297
Bottom - función de gráfico 1301
Column - función de gráfico 1305
Dimensionality - función de gráfico 1307
Exists 1308
FieldIndex 1312
FieldValue 1314
FieldValueCount 1315
LookUp 1317
NoOfRows - función de gráfico 1319
Peek 1321
Previous 1328
Top - función de gráfico 1330
SecondaryDimensionality - función de gráfico 1334
After - función de gráfico 1334
Before - función de gráfico 1336
First - función de gráfico 1337
Last - función de gráfico 1338
ColumnNo - función de gráfico 1339
NoOfColumns - función de gráfico 1340
8.17 Funciones lógicas 1340
8.18 Funciones de correspondencia 1341
Descripción general de las funciones de correspondencia 1341
ApplyMap 1342
MapSubstring 1343
8.19 Funciones matemáticas 1345
8.20 Funciones NULL 1346
Vista general de las funciones NULL 1346
EmptyIsNull 1346
IsNull 1347
NULL 1348
8.21 Funciones de rango 1349
Funciones de rango básicas 1349
Funciones de rango de contador 1350
Funciones de rango estadísticas 1350
Funciones de rango financieras 1351
RangeAvg 1352

RangeCorrel ... 1354
RangeCount ... 1356
RangeFractile ... 1358
RangeIRR ... 1360
RangeKurtosis ... 1362
RangeMax ... 1363
RangeMaxString ... 1365
RangeMin ... 1367
RangeMinString ... 1369
RangeMissingCount ... 1370
RangeMode ... 1372
RangeNPV ... 1374
RangeNullCount ... 1375
RangeNumericCount ... 1377
RangeOnly ... 1378
RangeSkew ... 1379
RangeStdev ... 1380
RangeSum ... 1382
RangeTextCount ... 1384
RangeXIRR ... 1386
RangeXNPV ... 1387
8.22 Funciones relacionales ... 1389
Funciones de ranking ... 1390
Funciones de agrupamiento ... 1390
Funciones de descomposición de series de tiempo ... 1391
Rank - función de gráfico ... 1392
HRank - función de gráfico ... 1396
Optimizar con k-means: Un ejemplo del mundo real ... 1398
KMeans2D - función de gráfico ... 1407
KMeansND - función de gráfico ... 1422
KMeansCentroid2D - función de gráfico ... 1437
KMeansCentroidND - función de gráfico ... 1438
STL_Trend - función de gráfico ... 1439
STL_Seasonal - función de gráfico ... 1441
STL_Residual - función de gráfico ... 1443
Tutorial: Descomposición de series de tiempo en Qlik Sense ... 1445
8.23 Funciones de distribución estadística ... 1450
Descripción general de las funciones de distribución estadística ... 1450
BetaDensity ... 1453
BetaDist ... 1453
BetaInv ... 1453
BinomDist ... 1454
BinomFrequency ... 1454
BinomInv ... 1455
ChiDensity ... 1455
ChiDist ... 1456
ChiInv ... 1456
FDensity ... 1457

FDist 1457
FInv 1458
GammaDensity 1459
GammaDist 1459
GammaInv 1460
NormDist 1460
NormInv 1461
PoissonDist 1462
PoissonFrequency 1462
PoissonInv 1462
TDensity 1463
TDist 1463
TInv 1464
8.24 Funciones de cadena 1464
Descripción general de las funciones de cadena 1465
Capitalize 1468
Chr 1469
Evaluate 1469
FindOneOf 1470
Hash128 1471
Hash160 1472
Hash256 1472
Index 1473
IsJson 1474
JsonGet 1475
JsonSet 1476
KeepChar 1477
Left 1478
Len 1479
LevenshteinDist 1479
Lower 1481
LTrim 1481
Mid 1482
Ord 1483
PurgeChar 1484
Repeat 1485
Replace 1485
Right 1486
RTrim 1487
SubField 1488
SubStringCount 1491
TextBetween 1492
Trim 1493
Upper 1494
8.25 Funciones de sistema 1494
Descripción general de las funciones de sistema 1495
EngineVersion 1497
GetSysAttr 1498

InObject - función de gráfico ... 1498
IsPartialReload ... 1502
ObjectId - función de gráfico ... 1502
ProductVersion ... 1505
StateName - función de gráfico ... 1506
8.26 Funciones de tabla ... 1506
Vista general de las funciones de tabla ... 1506
FieldName ... 1508
FieldNumber ... 1509
NoOfFields ... 1509
NoOfRows ... 1510
8.27 Funciones trigonométricas e hiperbólicas ... 1510
8.28 Funciones de ventana ... 1512
Window - función de script ... 1513
WRank - función de script ... 1520
9 Restricción de acceso al sistema de archivos ... 1527
9.1 Aspectos de seguridad relativos a la conexión con conexiones de datos ODBC y OLE DB basadas en archivos ... 1527
9.2 Limitaciones en el modo estándar ... 1527
Variables de sistema ... 1527
Sentencias de script habituales ... 1529
Sentencias de control de script ... 1530
Funciones de archivo ... 1531
Funciones de sistema ... 1533
9.3 Deshabilitar el modo estándar ... 1533
Qlik Sense ... 1534
Qlik Sense Desktop ... 1534
10 Secuencias de script a nivel de gráfico ... 1535
10.1 Sentencias de control ... 1535
Descripción general de las sentencias de control del modificador de gráfico ... 1535
Call ... 1537
Do..loop ... 1538
End ... 1539
Exit ... 1539
Exit script ... 1539
For..next ... 1540
For each..next ... 1541
If..then..elseif..else..end if ... 1544
Next ... 1545
Sub..end sub ... 1545
Switch..case..default..end switch ... 1547
To ... 1548
10.2 Prefijos ... 1548
Descripción general de los prefijos modificadores de gráficos ... 1548
Add ... 1548
Replace ... 1549
10.3 Sentencias habituales ... 1549

Descripción general de las sentencias habituales modificadoras de gráficos 1549
Load 1550
Let 1555
Set 1556
Put 1556
HCValue 1557
11 Funciones y sentencias de QlikView no admitidas en Qlik Sense 1558
11.1 Sentencias de script no admitidas en Qlik Sense 1558
11.2 Funciones no admitidas en Qlik Sense 1558
11.3 Prefijos no admitidos en Qlik Sense 1558
12 Funciones y sentencias no recomendadas en Qlik Sense 1559
12.1 Sentencias de script no recomendadas en Qlik Sense 1559
12.2 Parámetros de sentencias de script no recomendados en Qlik Sense 1559
12.3 Funciones no recomendadas en Qlik Sense 1561
El cualificador ALL 1561

1 ¿Qué es Qlik Sense?

Qlik Sense es una plataforma de análisis de datos. Con Qlik Sense podemos analizar datos y descubrir cosas por nuestra cuenta. Podemos compartir los conocimientos y analizar datos en equipo o en toda la empresa u organización. Qlik Sense nos permite formular y responder nuestras propias preguntas, así como seguir nuestro propio camino de investigación e indagación en los datos. Qlik Sense permite a cualquier equipo llegar a las decisiones de manera conjunta.

1.1 ¿Qué se puede hacer en Qlik Sense?

La mayoría de productos de Business Intelligence (BI) ayudan a las personas a responder preguntas que ya se comprenden de antemano. Pero ¿qué ocurre con las preguntas que se nos van ocurriendo sobre la marcha? ¿Ese tipo de preguntas que surgen tras leer un informe o visualizar un gráfico? Con la experiencia asociativa de Qlik Sense, podemos hacer todas las preguntas que se nos ocurran y responderlas al instante una tras otra, avanzando por nuestra propia ruta hacia el conocimiento. Con Qlik Sense podemos explorar los datos libremente, mediante simples clics de ratón, aprendiendo y profundizando en cada etapa del camino y descubriendo nuevas rutas de exploración basadas en nuestros propios descubrimientos.

1.2 ¿Cómo funciona Qlik Sense?

Qlik Sense genera vistas de la información sobre la marcha. Qlik Sense no requiere informes predefinidos o estáticos ni que el usuario dependa de otros departamentos o usuarios, tan solo hacemos clic y aprendemos cosas. Cada vez que un usuario hace clic, Qlik Sense responde al instante, actualizando cada visualización y vista de Qlik Sense en la app con un nuevo conjunto de datos recién calculados y visualizaciones específicas según lo que le interesa al usuario.

El modelo de apps

En lugar de desplegar y gestionar complejas aplicaciones de negocio, podemos crear nuestras propias apps de Qlik Sense y reutilizarlas, modificarlas o compartirlas con otros. El modelo de apps nos permite formular y dar respuesta a todas nuestras preguntas espontáneas, sin tener que recurrir a un experto para que elabore un nuevo informe o visualización.

La experiencia asociativa

Qlik Sense gestiona automáticamente todas las relaciones de los datos y presenta la información al usuario mediante una codificación de colores **green/white/gray**. Las selecciones se muestran de color verde, los datos asociados se visualizan en blanco y los datos excluidos (no asociados) se ven de color gris. Este feedback instantáneo anima a los usuarios a pensar en nuevas preguntas y continuar explorando y descubriendo cosas.

Colaboración y movilidad

Qlik Sense permite además colaborar con otros colegas siempre que lo necesite y sin importar dónde se encuentren. Todas las capacidades de Qlik Sense, incluida la experiencia asociativa y la colaboración, están

disponibles en dispositivos móviles. Con Qlik Sense, podemos formular y dar respuesta a nuestras propias preguntas sobre la marcha, seguir haciendo preguntas posteriores, implicar a amigos y colegas, en cualquier lugar en que nos encontremos.

1.3 ¿Cómo se puede instalar Qlik Sense?

Hay dos versiones de Qlik Sense para instalar, Qlik Sense Desktop y Qlik Sense Enterprise.

Qlik Sense Desktop

Se trata de una versión fácil de instalar, para un único usuario, que normalmente se instala en un ordenador local.

Qlik Sense Enterprise

Esta versión se utiliza para instalar sitios Qlik Sense. Un sitio es un conjunto de una o más máquinas de servidor conectadas a un repositorio lógico común o a un nodo central.

1.4 Cómo administrar y gestionar un sitio Qlik Sense

Con la consola (Consola de gestión de Qlik) se pueden configurar, gestionar y supervisar sitios Qlik Sense de forma fácil e intuitiva. Se pueden gestionar las licencias, el acceso y las normas de seguridad, configurar los nodos y las conexiones de origen de datos y sincronizar el contenido y los usuarios, entre muchas otras actividades y recursos.

1.5 Ampliar Qlik Sense y adaptarlo a sus propias necesidades

Qlik Sense le ofrece unas API y SDK flexibles para desarrollar sus propias extensiones y adaptar e integrar Qlik Sense para diferentes propósitos, como por ejemplo:

Creación de extensiones y mashups

Puede llevar a cabo actividades de desarrollo web con JavaScript para crear extensiones que constituyen una visualización personalizada en las apps Qlik Sense, o bien utilizar APIs de mashups para crear sitios web con contenido de Qlik Sense.

Crear clientes

Puede crear clientes en .NET e integrar objetos de Qlik Sense en sus propias aplicaciones. También puede crear clientes nativos en cualquier lenguaje de programación que admita la comunicación WebSocket mediante el protocolo de cliente de Qlik Sense.

Crear herramientas del servidor

Con las APIs de servicios y de directorio de usuarios puede crear su propia herramienta para administrar y gestionar los sitios Qlik Sense.

Conexión con otras fuentes de datos

Puede crear conectores de Qlik Sense para recuperar datos de fuentes de datos personalizados.

2 Descripción general de la sintaxis de script

2.1 Introducción a la sintaxis de script

En un script, se define el nombre de la fuente de datos, los nombres de las tablas y los nombres de los campos incluidos en la lógica. Además, están definidos en el script los campos definidos en los derechos de acceso. Un script se compone de una serie de sentencias que se ejecutan de manera consecutiva.

La sintaxis de la línea de comandos de Qlik Sense y la sintaxis de script se describen en una notación denominada Formalismo Backus-Naur, o código BNF.

Las primeras líneas de código ya se generan al crearse un nuevo archivo de Qlik Sense. Los valores por defecto de estas variables de interpretación numérica se derivan de las configuraciones regionales del sistema operativo.

El script consta de una serie de sentencias de script y palabras clave que se ejecutan de manera consecutiva. Todas las sentencias de script deben terminar con un punto y coma "**;**".

Puede usar expresiones y funciones en las sentencias **LOAD** para transformar los datos que se han cargado.

Para un archivo de tabla con comas, tabulaciones o puntos y coma como delimitadores, se puede usar una sentencia **LOAD**. Por defecto, una sentencia **LOAD** cargará todos los campos del archivo.

Se puede acceder a las bases de datos generales a través de los conectores de bases de datos ODBC o OLE DB. Aquí se utilizan sentencias SQL estándar. La sintaxis SQL aceptada difiere entre los distintos drivers ODBC.

Además, podemos acceder también a otras fuentes de datos utilizando conectores personalizados.

2.2 ¿Qué es el formalismo Backus-Naur?

La sintaxis de la línea de comandos de Qlik Sense y la sintaxis de script se describen en una notación denominada Formalismo Backus-Naur, o código BNF.

La siguiente tabla ofrece una lista de los símbolos utilizados en el código BNF, con una descripción de cómo se interpretan:

Símbolos

Símbolo	**Descripción**
|	OR lógico: se puede usar el símbolo en cualquiera de los dos lados.
( )	Paréntesis que definen la precedencia: se utilizan para estructurar la sintaxis BNF.
[ ]	Los corchetes indican que los elementos que encierran son opcionales.
{ }	Llaves: los elementos incluidos entre llaves pueden repetirse ninguna o más veces.
Símbolo	Una categoría sintáctica no concluyente que puede dividirse en otros símbolos. Por ejemplo, combinaciones de los de arriba, otros símbolos no concluyentes, cadenas de texto, etc.

Símbolo	Descripción
::=	Marca el comienzo de un bloque que define un símbolo.
LOAD	Un símbolo final que consiste en una cadena de texto. Debe escribirse tal cual en el script.

Todos los símbolos terminales se imprimen en estilo **bold face**. Por ejemplo, “(” debería interpretarse como un paréntesis que define la precedencia, mientras que “**(**” debería interpretarse como un carácter que ha de imprimirse en el script.

Ejemplo:

La descripción de la sentencia alias es:

alias fieldname **as** aliasname { , fieldname **as** aliasname}

Esto debe interpretarse como la cadena de texto "alias", seguida de un nombre de campo arbitrario, seguido por la cadena de texto "as", seguida por un nombre de alias arbitrario. Se puede dar cualquier cantidad de combinaciones adicionales de "fieldname as alias", separadas por comas.

Por ej. las siguientes sentencias son correctas:

```
alias a as first;

alias a as first, b as second;

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

Y estas sentencias no son correctas:

```
alias a as first b as second;

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

3 Sentencias de script y palabras clave

El script de Qlik Sense se compone de diversas sentencias. Una sentencia puede ser de dos tipos, una sentencia normal de script o una sentencia de control de script. Ciertas sentencias pueden ir precedidas de prefijos.

Las sentencias más comunes se utilizan habitualmente para manipular datos de varias formas. Estas sentencias pueden escribirse sobre cualquier número de filas en el script y deben terminar siempre en punto y coma ";".

Las sentencias de control en cambio se suelen emplear para controlar el flujo de ejecución del script. Hay que mantener cada cláusula de una sentencia de control dentro de una línea en el script. Estas cláusulas pueden terminar en punto y coma, o en un final de línea.

La aplicación de prefijos es posible con sentencias habituales, pero nunca con las sentencias de control. Los prefijos **when** y **unless** pueden usarse no obstante como sufijos para algunas cláusulas de control específicas.

En la siguiente sección hallará una lista ordenada alfabéticamente con todas las sentencias de script, sentencias de control y prefijos.

Todas las palabras clave del script pueden escribirse con cualquier combinación de caracteres en mayúscula o minúscula. Los nombres de campo y de variable utilizados en las sentencias, por supuesto, son sensibles a mayúsculas.

3.1 Sentencias de control de script

El script de Qlik Sense se compone de diversas sentencias. Una sentencia puede ser de dos tipos, una sentencia normal de script o una sentencia de control de script.

Las sentencias de control en cambio se suelen emplear para controlar el flujo de ejecución del script. Cada cláusula de una sentencia de control debe hallarse dentro de una línea de script y puede acabar en punto y coma o un final de línea.

Los prefijos nunca se aplican a las sentencias de control, con las excepciones de los prefijos **when** y **unless** que se pueden usar con algunas sentencias de control específicas.

Todas las palabras clave del script pueden escribirse con cualquier combinación de caracteres en mayúscula o minúscula.

Descripción general de las sentencias de control de script

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Call

La sentencia de control **call** invoca una subrutina que debe ir definida por una sentencia **sub** anterior.

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

Do..loop

La sentencia de control **do..loop** es una construcción de iteración de script que ejecuta una o varias sentencias hasta que se cumple una condición lógica.

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

Exit script

Esta sentencia de control detiene la ejecución del script. Puede insertarse en cualquier parte del script.

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

For each ..next

La sentencia de control **for each..next** es una construcción de iteración de script que ejecuta una o varias sentencias para cada valor en una lista separada por comas. Las sentencias dentro del bucle incluidas entre **for** y **next** se ejecutarán para cada valor de la lista.

```
For each..next var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

For..next

La sentencia de control **for..next** es una construcción de iteración de script con un contador. Las sentencias dentro del bucle incluidas entre **for** y **next** se ejecutarán para cada valor de la variable de contador entre los límites alto y bajo especificados.

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

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

If..then

La sentencia de control **if..then** es una construcción de selección de script que obliga a la ejecución del script a seguir diferentes rutas dependiendo de una o varias condiciones lógicas.

*Dado que la sentencia **if..then** es una sentencia de control y como tal finaliza con un punto y coma o un final de línea, cada una de sus cuatro cláusulas posibles (**if..then**, **elseif..then**, **else** y **end if**) no debe superar el límite de una línea.*

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

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

Sub

La sentencia de control **sub..end sub** define una subrutina que puede invocarse desde una sentencia **call**.

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

Switch

La sentencia de control **switch** es una construcción de selección de script que obliga a la ejecución de script a seguir diferentes rutas dependiendo del valor de una expresión.

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

Call

La sentencia de control **call** invoca una subrutina que debe ir definida por una sentencia **sub** anterior.

Sintaxis:

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

Argumentos:

Argumentos

Argumento	Descripción
name	El nombre de la subrutina.
paramlist	Una lista separada por comas de los parámetros que se habrán de enviar a la subrutina. Cada elemento de la lista puede ser un nombre de campo, una variable o una expresión arbitraria.

La subrutina llamada por una sentencia **call** debe definirse mediante una sentencia **sub** que se encuentre anteriormente durante la ejecución del script.

Los parámetros se copian en la subrutina y, si el parámetro en la sentencia **call** es una variable y no una expresión, se copia nuevamente al salir de la subrutina.

Limitaciones:

- Puesto que la sentencia **call** es una sentencia de control y, como tal, finaliza con un punto y coma o un final de línea, no debe superar el límite de una línea.
- Cuando define una subrutina con `Sub..end sub` dentro de una sentencia de control, por ejemplo `if..then`, solo puede llamar a la subrutina desde dentro de esa misma sentencia de control.

Ejemplo:

Este ejemplo enumera todos los archivos relacionados con Qlik en una carpeta y sus subcarpetas, y almacena la información de archivos en una tabla. Supongamos que ha creado una conexión de datos con el nombre Apps en la carpeta.

La subrutina DoDir se llama con la referencia a la carpeta, 'lib://Apps', como parámetro. Dentro de la subrutina, hay una llamada recursiva, `Call DoDir (Dir)`, que hace que la función busque archivos de manera recursiva en subcarpetas.

```
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://Apps')
```

Do..loop

La sentencia de control **do..loop** es una construcción de iteración de script que ejecuta una o varias sentencias hasta que se cumple una condición lógica.

Sintaxis:

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

*Dado que la sentencia **do..loop** es una sentencia de control y como tal finaliza con un punto y coma o un final de línea, cada una de sus tres posibles cláusulas (**do**, **exit do** y **loop**) no debe superar el límite de una línea.*

Argumentos:

Argumentos

Argumento	Descripción
condition	Una expresión lógica que devuelve True o False.
statements	Es cualquier grupo de una o varias sentencias de script de Qlik Sense.
while / until	La cláusula condicional **while** o **until** solo debe aparecer una vez en cualquier sentencia **do..loop**, es decir, o bien después de **do** o después de **loop**. Cada condición se interpreta sólo la primera vez que se encuentra pero se evalúa cada vez que se encuentra en el bucle.
exit do	Si se encuentra una cláusula **exit do** dentro del bucle, la ejecución del script se transferirá a la primera sentencia después de la cláusula **loop** que denota el final del bucle. Una cláusula **exit do** puede volverse condicional mediante el uso opcional de un sufijo **when** o **unless**.

Ejemplo:

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

Set a=1;

Do while a<10

LOAD * from file$(a).csv;

Let a=a+1;

Loop
```

End

La palabra clave de script **End** se utiliza para concluir cláusulas **If**, **Sub** y **Switch**.

Exit

La palabra clave de script **Exit** forma parte de la sentencia **Exit Script**, pero también se puede usar para salir de cláusulas **Do**, **For** o **Sub**.

Exit script

Esta sentencia de control detiene la ejecución del script. Puede insertarse en cualquier parte del script.

Sintaxis:

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

Puesto que la sentencia **exit script** es una sentencia de control y, como tal, finaliza con un punto y coma o un final de línea, no debe superar el límite de una línea.

Argumentos:

Argumentos

Argumento	Descripción
condition	Una expresión lógica que devuelve True o False.
when / unless	Una sentencia **exit script** puede volverse condicional mediante el uso opcional de **when** o una cláusula **unless**.

Ejemplos:

```
//Exit script
Exit Script;

//Exit script when a condition is fulfilled
Exit Script when a=1
```

For..next

La sentencia de control **for..next** es una construcción de iteración de script con un contador. Las sentencias dentro del bucle incluidas entre **for** y **next** se ejecutarán para cada valor de la variable de contador entre los límites alto y bajo especificados.

Sintaxis:

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

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

Las expresiones *expr1*, *expr2* y *expr3* solo se evalúan la primera vez que se entra en el bucle. El valor de la variable contador puede ser modificado por sentencias dentro del bucle, pero no es una buena práctica de programación.

Si se encuentra una cláusula **exit for** dentro del bucle, la ejecución del script se transferirá a la primera sentencia después de la cláusula **next** que denota el final del bucle. Una cláusula **exit for** puede volverse condicional mediante el uso opcional de un sufijo **when** o **unless**.

*Dado que la sentencia **for..next** es una sentencia de control y como tal finaliza con un punto y coma o un final de línea, cada una de sus tres posibles cláusulas (**for..to..step**, **exit for** y **next**) no debe superar el límite de una línea.*

Argumentos:

Argumentos

Argumento	Descripción
counter	Es un nombre de variable. Si se especifica *counter* después de **next** debe ser el mismo nombre de variable que el que se encuentra después del correspondiente **for**.
expr1	Una expresión que determina el primer valor de la variable *counter* para la que se debe ejecutar el bucle.
expr2	Una expresión que determina el último valor de la variable *counter* para el que se debe ejecutar el bucle.
expr3	Una expresión que determina el valor que indica el incremento de la variable *counter* cada vez que se ha ejecutado el bucle.
condition	una expresión lógica que devuelve True o False.
statements	Es cualquier grupo de una o varias sentencias de script de Qlik Sense.

Example 1: Cargar una secuencia de archivos

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

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

Example 2: Cargar un número aleatorio de archivos

En este ejemplo, se supone que hay archivos de datos *x1.csv, x3.csv, x5.csv, x7.csv* y *x9.csv*. La carga se detiene en un punto aleatorio usando la condición `if rand( )<0.5 then`.

```
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

La sentencia de control **for each..next** es una construcción de iteración de script que ejecuta una o varias sentencias para cada valor en una lista separada por comas. Las sentencias dentro del bucle incluidas entre **for** y **next** se ejecutarán para cada valor de la lista.

Sintaxis:
La sintaxis especial hace posible generar listas con los nombres de archivo y directorio en el directorio actual.

```
for each var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

Argumentos:

Argumentos

Argumento	Descripción
var	Es un nombre de variable de script que adquirirá un nuevo valor de lista para cada ejecución del bucle. Si se especifica **var** después de **next** debe ser el mismo nombre de variable que el que se encuentra después del correspondiente **for each**.

El valor de la variable **var** se puede modificar mediante sentencias dentro del bucle, pero no es una buena práctica de programación.

Si se encuentra una cláusula **exit for** dentro del bucle, la ejecución del script se transferirá a la primera sentencia después de la cláusula **next** que denota el final del bucle. Una cláusula **exit for** puede volverse condicional mediante el uso opcional de un sufijo **when** o **unless**.

*Dado que la sentencia **for each..next** es una sentencia de control y como tal finaliza con un punto y coma o un final de línea, cada una de sus tres posibles cláusulas (**for each**, **exit for** y **next**) no debe superar el límite de una línea.*

Sintaxis:

```
list := item { , item }
```

```
item := constant | (expression) | filelist mask | dirlist mask |
fieldvaluelist mask
```

Argumentos

Argumento	Descripción
constant	Es cualquier número o cadena. Obsérvese que una cadena introducida directamente en el script debe ir entre comillas simples. Una cadena sin entrecomillado simple se interpretará como una variable y entonces se utilizará el valor de dicha variable. Los números no tienen que ir entre comillas simples.
expression	Es una expresión cualquiera.
mask	Una máscara de nombre de archivo o carpeta que puede incluir cualquier carácter válido de nombre de archivo, así como los caracteres comodín estándar, ***** y **?**. Puede utilizar rutas de archivos absolutas o lib://.
condition	Una expresión lógica que devuelve True o False.
statements	Es cualquier grupo de una o varias sentencias de script de Qlik Sense.
filelist mask	Esta sintaxis produce una lista de todos los archivos incluidos en el directorio actual, separados por coma, que coincidan con la máscara de nombre de archivo. *Este argumento admite únicamente conexiones de biblioteca en modo estándar.*
dirlist mask	Esta sintaxis produce una lista con todas las carpetas de la carpeta actual (separadas por comas) que coincidan con la máscara de nombre de archivo. *Este argumento admite únicamente conexiones de biblioteca en modo estándar.*
fieldvaluelist mask	Esta sintaxis se repite a lo largo de los valores de un campo ya cargado en Qlik Sense.

*El Qlik Conectores de proveedores de almacenamiento web y otras conexiones de DataFiles no admiten máscaras de filtro que usen los caracteres comodín (***** y **?**).*

Example 1: Cargar una lista de archivos

```
// 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: Crear una lista de archivos en el disco

Este ejemplo carga una lista de todos los campos Qlik Sense relacionados en una carpeta.

```
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: Se repite a lo largo de los valores de un campo

Este ejemplo recorre toda la lista de valores cargados de FIELD y genera un nuevo campo, NEWFIELD. Por cada valor de FIELD, se crearán dos NEWFIELD registros.

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

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

La tabla resultante tiene el siguiente aspecto:

Example table

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

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

La sentencia de control **if..then** es una construcción de selección de script que obliga a la ejecución del script a seguir diferentes rutas dependiendo de una o varias condiciones lógicas.

Las sentencias de control se emplean habitualmente para controlar el flujo de ejecución del script. En una expresión de gráfico, utilice la función condicional **if** en su lugar.

Sintaxis:

```
If  condition then

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

Dado que la sentencia **if..then** es una sentencia de control y como tal finaliza con un punto y coma o un final de línea, cada una de sus cuatro cláusulas posibles (**if..then**, **elseif..then**, **else** y **end if**) no debe superar el límite de una línea.

Argumentos:

Argumentos

Argumento	Descripción
condition	Una expresión lógica que puede evaluarse como True o False.
statements	Es cualquier grupo de una o varias sentencias de script de Qlik Sense.

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

La palabra clave de script **Next** se utiliza para cerrar bucles **For**.

Sub..end sub

La sentencia de control **sub..end sub** define una subrutina que puede invocarse desde una sentencia **call**.

Sintaxis:

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

Los argumentos se copian en la subrutina y, si el parámetro real correspondiente en la sentencia **call** es un nombre de una variable, se copia nuevamente al salir de la subrutina.

Si una subrutina tiene parámetros más formales que los parámetros reales que pasan por una sentencia **call**, los parámetros adicionales se inicializarán en NULL y se podrán usar como variables locales dentro de la subrutina.

Argumentos:

Argumentos

Argumento	Descripción
name	El nombre de la subrutina.
paramlist	Una lista separada por comas con los nombres de variables de los parámetros formales de la subrutina. Estos pueden utilizarse como cualquier variable dentro de la subrutina.
statements	Es cualquier grupo de una o varias sentencias de script de Qlik Sense.

Limitaciones:

- Dado que la sentencia **sub** es una sentencia de control y como tal finaliza con un punto y coma o un final de línea, cada una de sus dos cláusulas (**sub** y **end sub**) no debe superar el límite de una línea.
- Cuando define una subrutina con `Sub..end sub` dentro de una sentencia de control, por ejemplo `if..then`, solo puede llamar a la subrutina desde dentro de esa misma sentencia de control.

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: - transferencia de parámetros

```
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)
```
El resultado de lo anterior será que, localmente, dentro de la subrutina, A se inicializará en 1, B se inicializará en 4 y C se inicializará en NULL.

Al salir de la subrutina, la variable global A obtendrá 2 como valor (copiado de la subrutina). El segundo parámetro real "(X+1)*2" no será copiado puesto que no es una variable. Por último, la variable global C no se verá afectada por la llamada de la subrutina.

Switch..case..default..end switch

La sentencia de control **switch** es una construcción de selección de script que obliga a la ejecución de script a seguir diferentes rutas dependiendo del valor de una expresión.

Sintaxis:

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

> *Dado que la sentencia **switch** es una sentencia de control y como tal finaliza con un punto y coma o un final de línea, cada una de sus cuatro cláusulas posibles (**switch**, **case**, **default** y **end switch**) no debe superar el límite de una línea.*

Argumentos:

Argumentos

Argumento	Descripción
expression	Es una expresión cualquiera.
valuelist	Una lista de valores separados por comas, con los que se compara el valor de expresión. La ejecución del script continuará con las sentencias del primer grupo que se haya hallado que contienen un valor en listavalores igual al valor de expresión. Cada valor de listavalores puede ser una expresión cualquiera. Si no se encuentra correspondencia en ninguna cláusula **case**, se ejecutarán las declaraciones bajo la cláusula **default**, si se especifica.
statements	Es cualquier grupo de una o varias sentencias de script de Qlik Sense.

Ejemplo:

```
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

La palabra clave de script **To** se utiliza en diversas sentencias de script.

3.2 Prefijos de script

La aplicación de prefijos es posible con sentencias habituales, pero nunca con las sentencias de control. Los prefijos **when** y **unless** pueden usarse no obstante como sufijos para algunas cláusulas de control específicas.

Todas las palabras clave del script pueden escribirse con cualquier combinación de caracteres en mayúscula o minúscula. Los nombres de campo y de variable utilizados en las sentencias, por supuesto, son sensibles a mayúsculas.

Descripción general de los prefijos de script

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Add

El prefijo **Add** se puede añadir a cualquier sentencia **LOAD** o **SELECT** en el script para especificar que debe agregar registros a otra tabla. También especifica que esta sentencia debe ejecutarse en una carga parcial. El prefijo **Add** también se puede usar en una sentencia **Map**.

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

```
Add [ Only ] mapstatement
```

Buffer

Los archivos QVD se pueden crear y mantener automáticamente a través del prefijo **buffer**. Este prefijo se puede utilizar en la mayoría de las sentencias **LOAD** y **SELECT** de scripts. Indica que los archivos QVD se utilizan para almacenar en caché/búfer el resultado de la sentencia.

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

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

Concatenate

Si dos tablas que deben concatenarse tienen diferentes conjuntos de campos, aún puede forzarse su concatenación mediante el prefijo **Concatenate**.

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

Crosstable

El prefijo de carga **crosstable** se utiliza para transponer datos estructurados de una "tabla cruzada" o "tabla pivotante". Los datos estructurados de esta manera se encuentran habitualmente cuando se trabaja con fuentes de hojas de cálculo. El resultado y el objetivo del prefijo de carga **crosstable** es transponer dichas estructuras a un equivalente de tabla normal orientada a columnas, ya que esta estructura generalmente es más adecuada para el análisis en Qlik Sense.

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

First

El prefijo **First** en una sentencia **LOAD** o **SELECT (SQL)** se usa para cargar un número máximo definido de registros desde una tabla de origen de datos.

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

Generic

El prefijo de carga **Generic** permite la conversión de datos modelados de entidad-atributo-valor (EAV) en una estructura de tabla relacional normalizada tradicional. El modelado EAV se denomina alternativamente "modelado de datos genéricos" o "esquema abierto".

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

Hierarchy

El prefijo **hierarchy** se utiliza para transformar una tabla jerárquica padre-hijo en una tabla que sea útil en un modelo de datos Qlik Sense. Se puede poner frente a una sentencia **LOAD** o **SELECT** y usará el resultado de la sentencia de carga como entrada para una transformación de tabla.

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

HierarchBelongsTo

Este prefijo se utiliza para transformar una tabla jerárquica padre-hijo en una tabla que sea útil en un modelo de datos Qlik Sense. Se puede poner frente a una sentencia **LOAD** o **SELECT** y usará el resultado de la sentencia de carga como datos de entrada para una transformación de tabla.

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

Inner

Los prefijos **join** y **keep** pueden ir precedidos por el prefijo **inner**.

Si se utiliza antes de **join** especifica que debería usarse un inner join. La tabla resultante contendrá por tanto combinaciones de valores de campo de las dos tablas originales donde los valores de campos de enlace se representan en ambas tablas. Si se utiliza antes de **keep**, especifica que ambas tablas de datos sin procesar deberán reducirse a su intersección común antes de ser almacenadas en Qlik Sense.

.

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

IntervalMatch

El prefijo **IntervalMatch** se utiliza para crear una tabla que coincida con valores numéricos discretos con uno o más intervalos numéricos y, opcionalmente, que coincida con los valores de una o varias claves adicionales.

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

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

Join

El prefijo **join** une la tabla cargada con una tabla existente o con la última tabla de datos creada.

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

Keep

El prefijo **keep** es similar al prefijo **join**. Al igual que el prefijo **join**, compara la tabla cargada con una tabla ya existente o con la última tabla de datos creada anteriormente, pero en lugar de unir la tabla cargada con una tabla existente, tiene el efecto de reducir una o ambas tablas antes de que se almacenen en Qlik Sense, en función de la intersección de datos de la tabla. La comparación final equivale a aplicar un join natural entre todos los campos comunes. La diferencia está en que las dos tablas no se unen, sino que se almacenan en Qlik Sense como dos tablas independientes de distinto nombre.

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

Left

Los prefijos **Join** y **Keep** pueden ir precedidos por el prefijo **left**.

Si se usan antes de **join** especifica que deberá usarse un left join. La tabla resultante solo contendrá combinaciones de valores de campo de las tablas iniciales donde los valores de campos de enlace se representan en la primera tabla. Si se usa antes de **keep**, especifica que la segunda tabla de datos sin procesar debe reducirse a su intersección común con la primera tabla, antes de ser almacenada en Qlik Sense.

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

Mapping

El prefijo **mapping** se utiliza para crear una tabla de correspondencia que se puede usar, por ejemplo, para reemplazar valores de campo y nombres de campo durante la ejecución de script.

```
Correspondencia  ( loadstatement | selectstatement )
```

Merge

El prefijo **Merge** se puede añadir a cualquier sentencia **LOAD** o **SELECT** en el script para especificar que la tabla cargada debe fusionarse en otra tabla. También especifica que esta sentencia debe ejecutarse en una carga parcial.

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

NoConcatenate

El prefijo **NoConcatenate** obliga a que dos tablas cargadas con conjuntos de campos idénticos se traten como dos tablas internas separadas, cuando de lo contrario se concatenarían automáticamente.

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

Outer

El prefijo explícito **Join** puede ir precedido por el prefijo **Outer** para especificar un outer join. En un outer join se generan todas las combinaciones entre las dos tablas. La tabla resultante contendrá por tanto combinaciones de valores de campo de las dos tablas originales donde los valores de campos de enlace se representan en una o ambas tablas. La palabra clave **Outer** es opcional y es el tipo de unión predeterminado que se usa cuando no se especifica un prefijo join.

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

Partial reload

Una carga completa siempre comienza eliminando todas las tablas en el modelo de datos existente y luego ejecuta el script de carga.

Una carga parcial *carga parcial (page 101)* no hace esto. En su lugar, mantiene todas las tablas en el modelo de datos y después ejecuta solo las sentencias **Load** y **Select** precedidas por un prefijo **Add**, **Merge** o **Replace**. Otras tablas de datos no se ven afectadas por el comando. El argumento **only** indica que la sentencia debe ejecutarse solo durante cargas parciales y debe ignorarse durante cargas completas. La tabla siguiente resume la ejecución de instrucciones para cargas parciales y completas.

Replace

El prefijo **Replace** se puede añadir a cualquier sentencia **LOAD** o **SELECT** en el script para especificar que la tabla cargada debe reemplazar a otra tabla. También especifica que esta sentencia debe ejecutarse en una carga parcial. El prefijo **Replace** también se puede usar en una sentencia **Map**.

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

```
Replace [only] mapstatement
```

Right

Los prefijos **Join** y **Keep** pueden ir precedidos por el prefijo **right**.

Si se usa antes de **join**, especifica que se debe usar un right join. La tabla resultante contendrá solo combinaciones de valores de campo de las dos tablas donde los valores de campos de enlace se representan en la segunda tabla. Si se usa antes de **keep**, especifica que la primera tabla de datos sin procesar debe reducirse a su intersección común con la segunda tabla antes de ser almacenada en Qlik Sense.

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

Sample

El prefijo **sample** para una sentencia **LOAD** o **SELECT** se utiliza para cargar una muestra aleatoria de registros desde el origen de datos.

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

Semantic
Las tablas que contienen relaciones entre registros se pueden cargar mediante un prefijo **semantic**. Pueden ser por ejemplo referencias dentro de una tabla, donde un registro apunte a otro, como por ej. a un predecesor o antepasado.

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

Unless
El prefijo y sufijo **unless** se utiliza para crear una cláusula condicional que determina si una sentencia o cláusula de salida debería evaluarse o no. Puede verse como una alternativa compacta a la sentencia **if..end if** completa.

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

When
El prefijo y sufijo **when** se utiliza para crear una cláusula condicional que determina si una sentencia o cláusula de salida debería ejecutarse o no. Puede verse como una alternativa compacta a la sentencia **if..end if** completa.

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

Add

El prefijo **Add** se puede añadir a cualquier sentencia **LOAD** o **SELECT** en el script para especificar que debe agregar registros a otra tabla. También especifica que esta sentencia debe ejecutarse en una carga parcial. El prefijo **Add** también se puede usar en una sentencia **Map**.

Para que la carga parcial funcione correctamente, la app debe abrirse con datos antes de que se active una carga parcial.

Realice una carga parcial con el botón **Cargar**. También puede usar Qlik Engine JSON API.

Sintaxis:

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

```
Add [only] mapstatement
```

Durante una recarga normal (no parcial), la construcción **Add LOAD** funcionará como una sentencia **LOAD** normal. Los registros se generarán y almacenarán en una tabla.

Si se usa el prefijo **Concatenate** o si existe una tabla con el mismo conjunto de campos, los registros se agregarán a la tabla existente relevante. De lo contrario, la construcción **Add LOAD** creará una nueva tabla.

Una carga parcial hará lo mismo. La única diferencia es que la construcción **Add LOAD** nunca creará una nueva tabla. Siempre existe una tabla relevante de la ejecución del script anterior a la que se deben agregar los registros.

No comprueba duplicados. Por lo tanto, una sentencia que utilice el prefijo **Add** a menudo incluirá un calificador distinct o una cláusula where que proteja de los duplicados.

La sentencia **Add Map...Using** hace que la asignación se produzca también durante la ejecución parcial del script.

Argumentos:

Argumentos

Argumento	Descripción
only	Un cualificador opcional que indica que la sentencia solo debe ejecutarse durante las cargas parciales. Y debe ser ignorada durante las recargas normales (no parciales).

Ejemplos y resultados:

Ejemplo	Resultado

`Tab1:`
 	
`LOAD Name, Number`
`FROM Persons.csv;`
 	
`Add LOAD Name, Number`
`FROM newPersons.csv;`
	Durante la recarga normal, los datos se cargan desde *Persons.csv* y se almacenan en la tabla Tab1 de Qlik Sense. Los datos de *NewPersons.csv* se concatenan entonces con la misma tabla de Qlik Sense.
	
	Durante la recarga parcial, los datos se cargan desde *NewPersons.csv* y se anexan a la tabla Tab1 de Qlik Sense. No se hace comprobación de duplicados.

`Tab1:`
 	
`SQL SELECT Name,`
`Number FROM`
`Persons.csv;`
 	
`Add LOAD Name, Number`
`FROM NewPersons.csv`
`where not exists`
`(Name);`
	Se realiza una comprobación de duplicados mirando si Name está en los datos de la tabla que se han cargado anteriormente.
	
	Durante la recarga normal, los datos se cargan desde *Persons.csv* y se almacenan en la tabla Tab1 de Qlik Sense. Los datos de *NewPersons.csv* se concatenan entonces con la misma tabla de Qlik Sense.
	
	Durante la recarga parcial, los datos se cargan desde *NewPersons.csv*, los cuales se anexan a la tabla Tab1 de Qlik Sense. Se realiza una comprobación de duplicados mirando si Name está en los datos de la tabla que se han cargado anteriormente.

`Tab1:`
 	
`LOAD Name, Number`
`FROM Persons.csv;`
 	
`Add Only LOAD Name,`
`Number FROM`
`NewPersons.csv where`
`not exists(Name);`
	Durante la recarga normal, los datos se cargan desde *Persons.csv* y se almacenan en la tabla Tab1 de Qlik Sense. Se descarta la carga de sentencia *NewPersons.csv*.
	
	Durante la recarga parcial, los datos se cargan desde *NewPersons.csv*, los cuales se anexan a la tabla Tab1 de Qlik Sense. Se realiza una comprobación de duplicados mirando si Name está en los datos de la tabla que se han cargado anteriormente.

Buffer

Los archivos QVD se pueden crear y mantener automáticamente a través del prefijo **buffer**. Este prefijo se puede utilizar en la mayoría de las sentencias **LOAD** y **SELECT** de scripts. Indica que los archivos QVD se utilizan para almacenar en caché/búfer el resultado de la sentencia.

Sintaxis:

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

Si no se utiliza ninguna opción, el búfer de QVD que crea la primera ejecución del script se utilizará de forma indefinida.

El archivo de buffer se almacena en la subcarpeta *Buffers*, normalmente *C:\ProgramData\Qlik\Sense\Engine\Buffers* (instalación del servidor) o *C:\Users\{user}\Documents\Qlik\Sense\Buffers* (Qlik Sense Desktop).

El nombre del archivo QVD es un nombre calculado, un hash hexadecimal de 160 bits de toda la sentencia **LOAD** o **SELECT** siguiente y otra información discriminatoria. Esto significa que el buffer de QVD se mostrará como no válido por cualquier cambio que se produzca en la siguiente sentencia de **LOAD** o **SELECT**.

Los búferes de QVD se quitarán normalmente cuando ya no se mencionen en ninguna parte a lo largo de una ejecución completa del script de la app que lo ha creado o cuando la app que lo ha creado ya no exista.

Argumentos:

Argumentos

Argumento	Descripción
incremental	La opción de incremental activa la capacidad de leer solo parte de un archivo subyacente. El tamaño anterior del archivo se almacena en el encabezado de XML en el archivo de QVD. Esto resulta de especial utilidad con archivos de registro. Todos los registros cargados en alguna ocasión anterior se leen desde el archivo de QVD mientras que los registros nuevos siguientes se leen desde el origen original y por último se crea un archivo de QVD actualizado. La opción incremental solo se puede usar con sentencias **LOAD** y archivos de texto. La carga incremental no se puede utilizar cuando se modifican o eliminan datos antiguos.
stale [after] amount [(días | horas)]	amount es un número que especifica el período de tiempo. Se pueden emplear decimales. Si se omite la unidad se interpreta como days. La opción de stale after se utiliza normalmente con fuentes de bases de datos donde no haya ninguna fecha-hora sencilla en los datos originales. En su lugar, especifique cuánto tiempo se puede usar la captura de QVD. Con la palabra stale tras la cláusula simplemente se establece un período de tiempo a partir de la hora de creación del buffer de QVD tras el cual se dejará de considerar válido. Antes de esa hora, el buffer de QVD se utilizará como origen de los datos, tras lo cual se utilizará el origen de datos original. El archivo del búfer de QVD se actualizará automáticamente y se iniciará un nuevo período.

Limitaciones:

Existen varias limitaciones. La más notable es que debe haber un archivo **LOAD** o una sentencia de **SELECT** en el centro de una sentencia compleja.

Example 1:

```
Buffer SELECT * from MyTable;
```

Example 2:

```
Buffer (stale after 7 days) SELECT * from MyTable;
```

Example 3:

```
Buffer (incremental) LOAD * from MyLog.log;
```

Concatenate

`Concatenate` es un prefijo de carga de script que permite agregar un conjunto de datos a una tabla en memoria ya existente. A menudo se utiliza para agregar diferentes conjuntos de datos de transacciones a una sola tabla central de hechos, o para crear conjuntos de datos de referencia comunes de un tipo específico que se originan en múltiples fuentes. Es similar en funcionalidad a un operador SQL UNION.

La tabla resultante de una operación `concatenate` contendrá el conjunto de datos original con las nuevas filas de datos añadidas al final de esa tabla. Las tablas de origen y de destino pueden tener diferentes campos presentes. Cuando los campos sean diferentes, la tabla resultante se ampliará para representar el resultado combinado de todos los campos presentes tanto en la tabla de origen como en la tabla de destino.

Sintaxis:

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

Argumentos

Argumento	Descripción
`tablename`	El nombre de una tabla existente. La tabla nombrada será el destino de la operación `Concatenate` y cualquier registro de datos cargado se agregará a esa tabla. Si no se usa el parámetro `tablename`, la tabla de destino será la última tabla cargada antes de esta sentencia.
`loadstatement/selectstatement`	El argumento `loadstatement/selectstatement` que sigue al argumento `tablename` se concatenarán a la tabla especificada.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo de función

Ejemplo	Resultado
`Concatenate (Transactions) Load …. ;`	Los datos cargados en la instrucción de carga debajo del prefijo `Concatenate` se agregarán a la tabla en memoria existente denominada `Transactions` (suponiendo que una tabla llamada `Transactions` se haya cargado antes de este punto en el script de carga).

Ejemplo 1: agregar varios conjuntos de datos a una tabla de destino con el prefijo de carga Concatenate

Script de carga y resultados

Vista general

En este ejemplo, cargará dos scripts en orden secuencial.

- El primer script de carga contiene un conjunto de datos inicial con fechas y cantidades que se envía a una tabla denominada `Transactions`.
- El segundo script de carga contiene:
    - Un segundo conjunto de datos que se agrega al conjunto de datos inicial mediante el uso del prefijo `Concatenate`. Este conjunto de datos tiene un campo adicional, `type`, que no está en el conjunto de datos inicial.
    - El prefijo `Concatenate`.

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

Primer script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

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

Tabla de resultados del primer script de carga

id	fecha	cantidad
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

La tabla muestra el conjunto de datos inicial.

Segundo script de carga

Abra el editor de carga de datos y agregue el script de carga a continuación.

```
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
];
```

Resultados

Cargue los datos y vaya a la hoja. Cree este campo como una dimensión.

- `type`

Tabla de resultados del segundo script de carga

id	fecha	cantidad	tipo
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	-
3758	10/01/2018	164.27	Internas
3759	10/03/2018	384.00	Externo
3760	10/06/2018	25.82	Internas
3761	10/09/2018	312.00	Internas
3762	10/15/2018	4.56	Internas
3763	10/16/2018	90.24	Internas
3764	10/18/2018	19.32	Externo

Tenga en cuenta los valores nulos en el campo `type` para los primeros siete registros cargados donde no se había definido `type`.

Ejemplo 2: agregar varios conjuntos de datos a una tabla de destino mediante concatenación implícita

Script de carga y resultados

Vista general

Un caso de uso típico para agregar datos implícitamente es cuando carga varios archivos de datos estructurados de manera idéntica y desea agregarlos todos a una tabla de destino.

Por ejemplo, mediante el uso de `wildcards` en nombres de archivo con sintaxis como:

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

o en bucles usando construcciones como:

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

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

next file
```

La concatenación implícita tendrá lugar entre dos tablas que se cargan con campos con nombres idénticos, incluso si no están definidos uno tras otro en el script. Esto puede provocar que los datos se añadan involuntariamente a las tablas. Si no desea agregar una tabla secundaria con campos idénticos de esta manera, utilice el prefijo de carga `NoConcatenate`*. Cambiar el nombre de la tabla con una etiqueta de nombre de tabla alternativa no es suficiente para evitar que se produzca una concatenación implícita. Para más información, vea NoConcatenate (page 91).*

En este ejemplo, cargará dos scripts en orden secuencial.

- El primer script de carga contiene un conjunto de datos inicial con cuatro campos que se envía a una tabla denominada `Transactions`.
- El segundo script de carga contiene un conjunto de datos con los mismos campos que el primer conjunto de datos.

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

Primer script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

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

Tabla de resultados del primer script de carga

id	fecha	tipo	cantidad
3758	10/01/2018	Internas	164.27
3759	10/03/2018	Externo	384.00
3760	10/06/2018	Internas	25.82
3761	10/09/2018	Internas	312.00
3762	10/15/2018	Internas	4.56
3763	10/16/2018	Internas	90.24
3764	10/18/2018	Externo	19.32

La tabla muestra el conjunto de datos inicial.

Segundo script de carga

Abra el editor de carga de datos y agregue el script de carga a continuación.

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

Resultados

Cargue los datos y vaya a la hoja.

Tabla de resultados del segundo script de carga

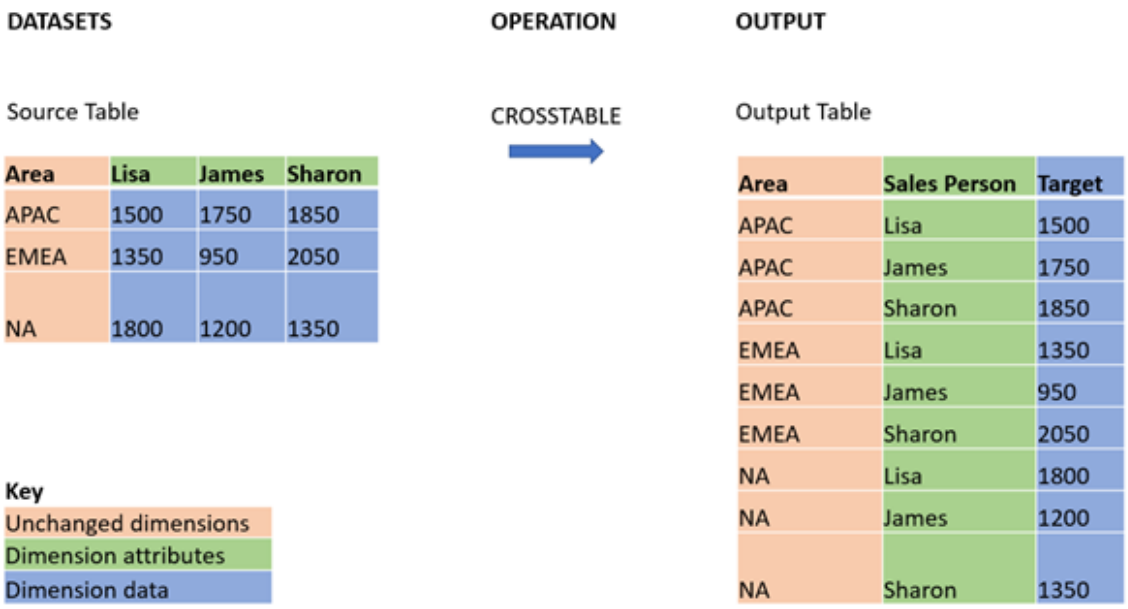
id	fecha	tipo	cantidad
3758	10/01/2018	Internas	164.27
3759	10/03/2018	Externo	384.00
3760	10/06/2018	Internas	25.82
3761	10/09/2018	Internas	312.00
3762	10/15/2018	Internas	4.56
3763	10/16/2018	Internas	90.24
3764	10/18/2018	Externo	19.32
3765	11/03/2018	Internas	129.40
3766	11/05/2018	Externo	638.50

El segundo conjunto de datos se concatenó implícitamente en el conjunto de datos inicial porque tenían campos idénticos.

Crosstable

El prefijo de carga **crosstable** se utiliza para transponer datos estructurados de una "tabla cruzada" o "tabla pivotante". Los datos estructurados de esta manera se encuentran habitualmente cuando se trabaja con fuentes de hojas de cálculo. El resultado y el objetivo del prefijo de carga **crosstable** es transponer dichas estructuras a un equivalente de tabla normal orientada a columnas, ya que esta estructura generalmente es más adecuada para el análisis en Qlik Sense.

Ejemplo de datos estructurados como una tabla cruzada y su estructura equivalente después de una transformación de tabla cruzada



Sintaxis:

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

Argumentos

Argumento	Descripción
attribute field name	El nombre del campo de salida deseado que describe la dimensión orientada horizontalmente que se transpondrá (la fila de encabezado).
data field name	El nombre del campo de salida deseado que describe los datos orientados horizontalmente de la dimensión que se transpondrá (la matriz de valores de datos debajo de la fila del encabezado).
n	El número de campos cualificadores, o dimensiones no modificadas, que preceden a la tabla que se va a transformar a una forma genérica. El valor predeterminado es 1.

Esta función de script se relaciona con las siguientes funciones:

Funciones relacionadas

Función	Interacción
Generic (page 58)	Un prefijo de carga de transformación que toma un conjunto de datos estructurados entidad-atributo-valor y lo transforma en una estructura de tabla relacional normal, separando cada atributo encontrado en un nuevo campo o columna de datos.

Ejemplo 1: transformar datos de ventas pivotados (simple)

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el primer script de carga a continuación, en una nueva pestaña.

El primer script de carga contiene un conjunto de datos al que se aplicará el prefijo de script `crosstable` más adelante, con la sección de aplicación `crosstable` descomentada. Esto significa que se utilizó la sintaxis de comentarios para deshabilitar esta sección en el script de carga.

El segundo script de carga es el mismo que el primero, pero con la aplicación de `crosstable` no comentado (habilitado al eliminar la sintaxis de comentarios). Los scripts se muestran de esta manera para resaltar el valor de esta función de script en la transformación de datos.

Primer script de carga (función no aplicada)

```
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;
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

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

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

Tabla de resultados

Producto	Ene 2021	Feb 2021	Mar 2021	Abr 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

Este script permite la creación de una tabla cruzada con una columna para cada mes y una fila por producto. En su formato actual, estos datos no son fáciles de analizar. Sería mucho mejor tener todos los números en un campo y todos los meses en otro, en una tabla de tres columnas. La siguiente sección explica cómo hacer esta transformación en la tabla cruzada.

Segundo script de carga (función aplicada)

Descomente el script eliminando el //. El script de carga deberá presentar el siguiente aspecto:

```
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;
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- Product
- Month
- Sales

Tabla de resultados

Producto	Month (Mes)	Ventas
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

Una vez que se ha aplicado el prefijo de script, la tabla cruzada se transforma en una tabla simple con una columna para `Month` y otra para `Sales`. Esto mejora la legibilidad de los datos.

Ejemplo 2: transformar datos de objetivos de ventas pivotados en una estructura de tabla vertical (intermedio)

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada Targets (Objetivos).
- El prefijo de carga `crosstable`, que transpone los nombres de las personas de ventas dinámicas en un

campo propio, etiquetado `Sales Person`.

- Los datos de objetivos de ventas asociados, que se estructuran en un campo llamado `Target`.

Script de carga

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

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `Area`
- `Sales Person`

Añada esta medida:

```
=Sum(Target)
```

Tabla de resultados

Área	Vendedor	=Sum(Target)
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
N/D	James	1200
N/D	Lisa	1800
N/D	Sharon	1350

Si desea replicar la visualización de datos como la tabla de entrada dinámica, puede crear una tabla pivotante equivalente en una hoja.

Haga lo siguiente:

1. Copie y pegue la tabla que acaba de crear en la hoja.
2. Arrastre el objeto gráfico de la **Tabla pivotante** a la parte superior de la copia de la tabla recién creada. Seleccione **Convertir**.
3. Haga clic en ✔ **Edición finalizada**.
4. Arrastre el campo `Sales Person` desde el estante de la columna vertical al estante de la columna horizontal.

La tabla siguiente muestra los datos en su forma de tabla inicial, como se indica en Qlik Sense:

Tabla de resultados originales, como se muestra en Qlik Sense

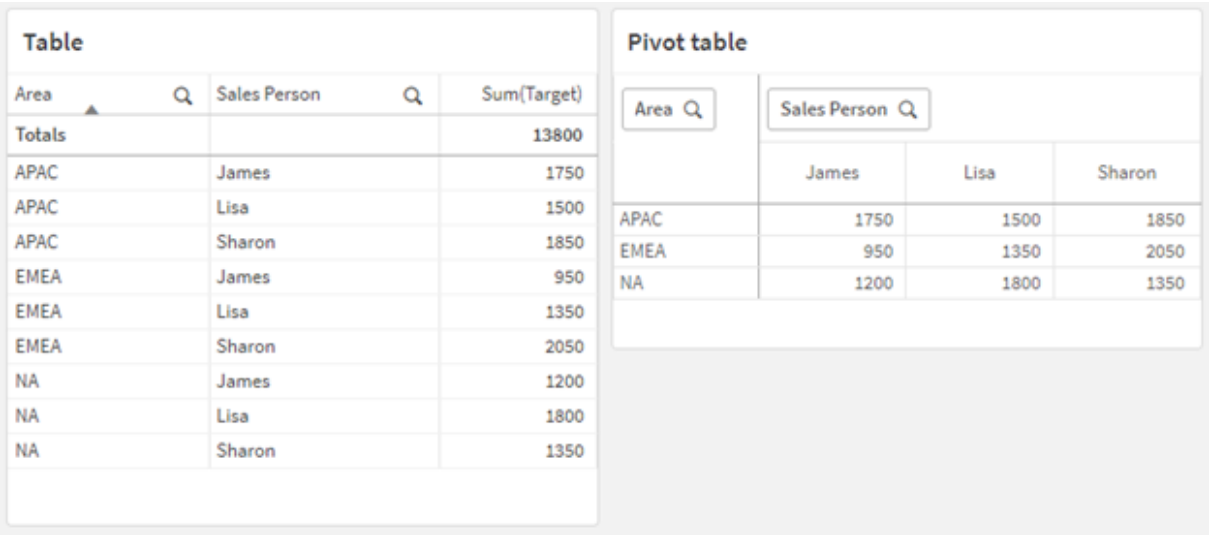
Área	**Vendedor**	**=Sum(Target)**
Totales	-	13800
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
N/D	James	1200
N/D	Lisa	1800
N/D	Sharon	1350

La tabla pivotante equivalente es similar a la siguiente, con la columna para el nombre de cada vendedor contenida dentro de la fila más grande para `Sales Person`:

Tabla pivotante equivalente con el campo `Sales Person` pivotado horizontalmente

Área	**James**	**Lisa**	**Sharon**
APAC	1750	1500	1850
EMEA	950	1350	2050
N/D	1350	1350	1350

Ejemplo de datos que se muestran como una tabla y una tabla pivotante equivalente con el campo `Sales Person` pivotado horizontalmente



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

Ejemplo 3: transformar datos de objetivos y ventas dinámicas en una estructura de tabla vertical (avanzado)

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que representa datos de ventas y objetivos, organizados por área y mes del año. Esto se carga en una tabla llamada `SalesAndTargets`.
- El prefijo de carga `crosstable`. Esto se usa para despivotar la dimensión `Month Year` en un campo específico, así como para transponer la matriz de ventas y las cantidades objetivo en un campo específico llamado `Amount`.
- Una conversión del campo de texto `Month Year` a una fecha adecuada, usando la función de conversión de texto a fecha `date#`. Este campo `Month Year` convertido a fecha se vuelve a unir a la tabla `SalesAndTarget` a través de un prefijo de carga `Join`.

Script de carga

```
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;
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `Area`
- `Month Year`

Cree la siguiente medida, con la etiqueta `Actual`:

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

Cree también la siguiente medida, con la etiqueta `Target`:

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

Tabla de resultados (cortada)

Área	Mes Año	Actual	Objetivo
APAC	Jan-22	435	425
APAC	Feb-22	434	425
APAC	Mar-22	397	425
APAC	Apr-22	404	425
APAC	May-22	458	425
APAC	Jun-22	447	425
APAC	Jul-22	413	425
APAC	Aug-22	458	425
APAC	Sep-22	385	425
APAC	Oct-22	421	425
APAC	Nov-22	448	425

Área	Mes Año	Actual	Objetivo
APAC	Dec-22	397	425
EMEA	Jan-22	363.5	362.5
EMEA	Feb-22	359.5	362.5

Si desea replicar la visualización de datos como la tabla de entrada dinámica, puede crear una tabla pivotante equivalente en una hoja.

Haga lo siguiente:

1. Copie y pegue la tabla que acaba de crear en la hoja.
2. Arrastre el objeto gráfico de la **Tabla pivotante** a la parte superior de la copia de la tabla recién creada. Seleccione **Convertir**.
3. Haga clic en ✔ **Edición finalizada**.
4. Arrastre el campo `Month Year` desde el estante de la columna vertical al estante de la columna horizontal.
5. Arrastre el elemento `Values` desde el estante de la columna horizontal al estante de la columna vertical.

La tabla siguiente muestra los datos en su forma de tabla inicial, como se indica en Qlik Sense:

Tabla original de resultados (cortada), como se muestra en Qlik Sense

Área	Mes Año	Actual	Objetivo
Totales	-	13812	13950
APAC	Jan-22	435	425
APAC	Feb-22	434	425
APAC	Mar-22	397	425
APAC	Apr-22	404	425
APAC	May-22	458	425
APAC	Jun-22	447	425
APAC	Jul-22	413	425
APAC	Aug-22	458	425
APAC	Sep-22	385	425
APAC	Oct-22	421	425
APAC	Nov-22	448	425
APAC	Dec-22	397	425

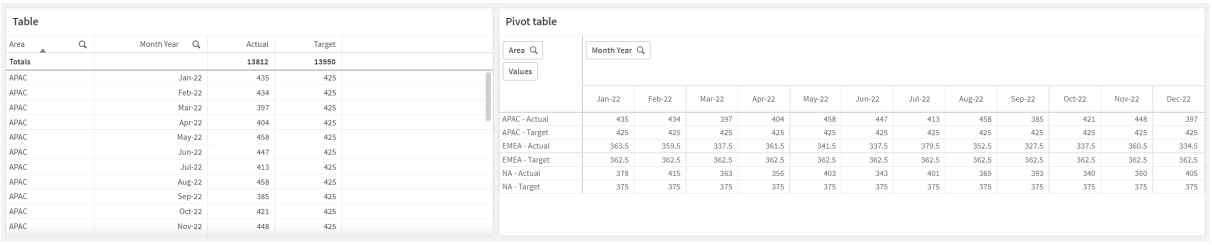
Área	Mes Año	Actual	Objetivo
EMEA	Jan-22	363.5	362.5
EMEA	Feb-22	359.5	362.5

La tabla pivotante equivalente es similar a la siguiente, con la columna para el nombre de cada mes del año individual contenida dentro de la fila más grande de `Month Year`:

Tabla pivotante equivalente (cortada) con el campo `Month Year` pivotado horizontalmente

Area (Valores)	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 - Actual	435	434	397	404	458	447	413	458	385	421	448	397
APAC - Objetivo	425	425	425	425	425	425	425	425	425	425	425	425
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
EMEA - Objetivo	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5
N/D - Actual	378	415	363	356	403	343	401	365	393	340	360	405
N/D - Objetivo	375	375	375	375	375	375	375	375	375	375	375	375

Ejemplo de datos que se muestran como una tabla y una tabla pivotante equivalente con el campo `Month Year` pivotado horizontalmente



First

El prefijo `First` en una sentencia SQL `LOAD` o `SELECT` se usa para cargar un número máximo definido de registros desde una tabla de origen de datos. Un caso típico de uso del prefijo `First` es cuando desea recuperar un pequeño subconjunto de registros desde un paso de carga de datos muy extenso o lento. Tan pronto como se haya cargado el número "`n`" de registros definido, el paso de carga finaliza prematuramente y el resto de la ejecución del script continúa

con normalidad.

Sintaxis:

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

Argumentos

Argumento	Descripción
n	Una expresión arbitraria que devuelve un entero indicando el número máximo de registros que se ha de leer. n puede ir también entre paréntesis: (n).
loadstatement | selectstatement	La instrucción load statement/select statement que sigue al argumento n definirá la tabla especificada que debe cargarse con el número máximo de registros establecido.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia SET DateFormat de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
FIRST 10 LOAD * from abc.csv;	Este ejemplo recuperará las primeras diez líneas de un archivo de Excel.
FIRST (1) SQL SELECT * from Orders;	Este ejemplo recuperará la primera línea seleccionada del conjunto de datos Orders.

Ejemplo: cargar las primeras cinco filas

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de fechas de las dos primeras semanas de 2020.
- La variable `First` que le indica a la aplicación que solo cargue los primeros cinco registros.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue `Date` como campo y `sum(sales)` como medida:

Tabla de resultados

Fecha	sum(Sales)
01/01/2020	6000
01/02/2020	3000
01/03/2020	6000
01/04/2020	8000
01/05/2020	5000

El script solo carga los primeros cinco registros de la tabla `Sales`.

Generic

El prefijo de carga **Generic** permite la conversión de datos modelados de entidad-atributo-valor (EAV) en una estructura de tabla relacional normalizada tradicional. El modelado EAV se denomina alternativamente "modelado de datos genéricos" o "esquema abierto".

Ejemplo de datos modelados EAV y una tabla relacional desnormalizada equivalente

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

Ejemplo de datos modelados EAV y un conjunto equivalente de tablas relacionales normalizadas

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

Si bien es técnicamente posible cargar y analizar datos modelados de EAV en Qlik, a menudo es más fácil trabajar con una estructura de datos relacional tradicional equivalente.

Sintaxis:

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

Estos temas le ayudarán a trabajar con esta función:

Temas relacionados

Tema	Descripción
Crosstable (page 47)	El prefijo de carga `Crosstable` transforma los datos que están orientados horizontalmente en datos orientados verticalmente. Desde una perspectiva puramente funcional, realiza la transformación opuesta al prefijo de carga `Generic`, aunque los prefijos suelen servir para casos de uso completamente diferentes.
Bases de datos genéricas en *Gestión de datos*	Los modelos de datos estructurados de EAV se describen con más detalle aquí.

Ejemplo 1: transformar datos estructurados de EAV con el prefijo de carga genérico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene un conjunto de datos que se carga en una tabla denominada Transactions. El conjunto de datos incluye un campo de fecha. Se utiliza la definición de `MonthNames` predeterminada.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `Color`.

Añada esta medida:

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

Ahora puede inspeccionar la cantidad de productos por color.

Tabla de resultados

Color	**=Count([Product ID])**
Marrón	4
Blanco	2

Observe la forma del modelo de datos, donde cada atributo se ha desglosado en una tabla aparte, nombrada de acuerdo con la etiqueta de la tabla de destino original `Product`. Cada tabla tiene el atributo como sufijo. Un ejemplo de esto es `Product.Color`. Los registros de salida de atributos de producto resultantes están asociados por `Product ID`.

Representación de los resultados del Visor del modelo de datos

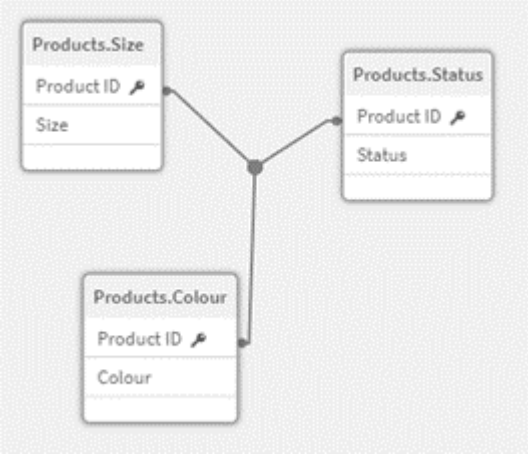


Tabla de registros resultante: Productos.Estado

ID de producto	Estado
13	Descontinuado
2	Descontinuado

Tabla de registros resultante: Products.Size

ID de producto	Tamaño
13	13-15
20	16-18
45	16-18

Tabla de registros resultante: Productos.Color

ID de producto	Color
13	Marrón
5	Marrón
44	Marrón
45	Marrón
20	Blanco
2	Blanco

Ejemplo 2: analizar datos estructurados de EAV sin el prefijo de carga genérico

Script de carga y expresión de gráfico

Vista general

Este ejemplo muestra cómo analizar datos estructurados de EAV en su forma original.

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene un conjunto de datos que se carga en una tabla denominada `Products` en una estructura EAV.

En este ejemplo, seguimos contando productos por atributo de color. Para analizar los datos estructurados de esta manera, deberá aplicar un filtrado a nivel de expresión de productos que lleven el valor de Atributo `Color`.

Además, los atributos individuales no están disponibles para seleccionar como dimensiones o campos, lo que dificulta determinar cómo crear visualizaciones efectivas.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `Value`.

Cree la siguiente medida:

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

Ahora puede inspeccionar la cantidad de productos por color.

Tabla de registros resultante: Productos.Estado

Valor	=Count({<Attribute={'Color'}>} [Product ID])
Marrón	4
Blanco	2

Ejemplo 3: desnormalizar las tablas de salida resultantes de una carga genérica (avanzada)

Script de carga y expresión de gráfico

Vista general

En este ejemplo, mostramos cómo la estructura de datos normalizada producida por el prefijo de carga `Generic` se puede desnormalizar nuevamente en una tabla de dimensiones `Product` consolidada. Esta es una técnica de modelado avanzada que se puede emplear como parte del ajuste del rendimiento del modelo de datos.

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

Script de carga

```
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;
```

Resultados

Abra el visor del modelo de datos y observe la forma del modelo de datos resultante. Solo hay una tabla desnormalizada presente. Es una combinación de las tres tablas de salida intermedias: `Products.Size`, `Products.Status` y `Products.Color`.

Modelo de datos internos resultante

Productos
ID de producto
Estado
Color
Tamaño

Tabla de registros resultante: Productos

ID de producto	**Estado**	**Color**	**Tamaño**
13	Descontinuado	Marrón	13-15
20	-	Blanco	16-18
2	Descontinuado	Blanco	-
5	-	Marrón	-
44	-	Marrón	-
45	-	Marrón	16-18

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `Color`.

Añada esta medida:

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

Tabla de resultados

Color	**=Count([Product ID])**
Marrón	4
Blanco	2

Hierarchy

El prefijo **hierarchy** se utiliza para transformar una tabla jerárquica padre-hijo en una tabla que sea útil en un modelo de datos Qlik Sense. Se puede poner frente a una sentencia **LOAD** o **SELECT** y usará el resultado de la sentencia de carga como entrada para una transformación de

tabla.

El prefijo crea una tabla de nodos expandidos, la cual contiene normalmente el mismo número de registros que una tabla de entrada, pero además, cada nivel de la jerarquía se almacena en un campo aparte. El campo path puede utilizarse en una estructura arbórea.

Sintaxis:

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

La tabla de entrada debe ser una tabla de nodos adyacentes. Las tablas de nodos adyacentes son tablas en las que cada registro se corresponde con un nodo y tiene un campo que contiene una referencia al nodo padre. En este tipo de tabla, el nodo se almacena en un único registro, pero puede tener un número indeterminado de hijos. La tabla puede por supuesto contener campos adicionales que describan atributos para los nodos.

El prefijo crea una tabla de nodos expandidos, la cual contiene normalmente el mismo número de registros que una tabla de entrada, pero además, cada nivel de la jerarquía se almacena en un campo aparte. El campo path puede utilizarse en una estructura arbórea.

Normalmente, la tabla de entrada tiene exactamente un registro por nodo y en dicho caso la tabla resultante contendrá el mismo número de registros. No obstante, a veces hay nodos con múltiples padres, esto es, un nodo viene representado por varios registros en la tabla de entrada. Si éste es el caso, la tabla resultante podrá contener un número mayor de registros que la tabla de entrada.

Todos los nodos con un id paterno que no se encuentre en la columna nodeid (incluidos aquellos nodos que hayan perdido su id de padre) se considerarán como raíces. Asimismo, únicamente se cargarán los nodos que posean una conexión a un nodo raíz – directa o indirecta – evitándose con esto las referencias circulares.

Los campos adicionales que contengan el nombre del nodo padre, la ruta del nodo y la profundidad del nodo se podrán crear sin problema.

Argumentos:

Argumentos

Argumento	Descripción
NodeID	El nombre del campo que contiene el ID de nodo. Este campo debe existir en la tabla de entrada.
ParentID	El nombre del campo que contiene el ID de nodo del nodo padre. Este campo debe existir en la tabla de entrada.
NodeName	El nombre del campo que contiene el nombre del nodo. Este campo debe existir en la tabla de entrada.
ParentName	Una cadena utilizada para nombrar el nuevo campo **ParentName**. Si se omite, dicho campo no se creará.

Argumento	Descripción
ParentSource	Es el nombre del campo que contiene el nombre del nodo empleado para crear la ruta al nodo. Se trata de un parámetro opcional. Si se omite, se utilizará **NodeName**.
PathName	Una cadena que se utiliza para nombrar el nuevo campo **Path**, que contiene la ruta de la raíz al nodo. Se trata de un parámetro opcional. Si se omite, dicho campo no se creará.
PathDelimiter	Una cadena utilizada como delimitador en el nuevo campo **Path**. Se trata de un parámetro opcional. Si se omite, se utilizará "/".
Depth	Una cadena utilizada para nombrar el nuevo campo **Depth**, que contiene la profundidad del nodo en la jerarquía. Se trata de un parámetro opcional. Si se omite, dicho campo no se creará.

Ejemplo:

```
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

Este prefijo se utiliza para transformar una tabla jerárquica padre-hijo en una tabla que sea útil en un modelo de datos Qlik Sense. Se puede poner frente a una sentencia **LOAD** o **SELECT** y usará el resultado de la sentencia de carga como datos de entrada para una transformación de tabla.

El prefijo crea una tabla que contiene todas las relaciones hijo-antepasado de la jerarquía. Los campos de antepasados pueden de esta forma utilizarse para seleccionar árboles enteros en la jerarquía. La tabla resultante contiene normalmente varios registros por nodo.

Sintaxis:

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

La tabla de entrada debe ser una tabla de nodos adyacentes. Las tablas de nodos adyacentes son tablas en las que cada registro se corresponde con un nodo y tiene un campo que contiene una referencia al nodo padre. En este tipo de tabla, el nodo se almacena en un único registro, pero puede tener un número indeterminado de hijos. La tabla puede por supuesto contener campos adicionales que describan atributos para los nodos.

El prefijo crea una tabla que contiene todas las relaciones hijo-antepasado de la jerarquía. Los campos de antepasados pueden de esta forma utilizarse para seleccionar árboles enteros en la jerarquía. La tabla resultante contiene normalmente varios registros por nodo.

Se puede crear un campo adicional que contenga la diferencia de profundidad entre los nodos.

Argumentos:

Argumentos

Argumento	Descripción
NodeID	El nombre del campo que contiene el ID de nodo. Este campo debe existir en la tabla de entrada.
ParentID	El nombre del campo que contiene el ID de nodo del nodo padre. Este campo debe existir en la tabla de entrada.
NodeName	El nombre del campo que contiene el nombre del nodo. Este campo debe existir en la tabla de entrada.
AncestorID	Una cadena que se emplea para nombrar el nuevo campo de ID del antepasado, el cual contiene el ID del nodo antepasado.
AncestorName	Es una cadena que se emplea para nombrar el nuevo campo del antepasado, el cual contiene el nombre del nodo antepasado.
DepthDiff	Una cadena utilizada para nombrar el nuevo campo **DepthDiff**, que contiene la profundidad del nodo en la jerarquía relativa al nodo antepasado. Se trata de un parámetro opcional. Si se omite, dicho campo no se creará.

Ejemplo:

```
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

Los prefijos **join** y **keep** pueden ir precedidos por el prefijo **inner**. Si se utiliza antes de **join** especifica que debería usarse un inner join. La tabla resultante contendrá por tanto combinaciones de valores de campo de las dos tablas originales donde los valores de campos

de enlace se representan en ambas tablas. Si se utiliza antes de **keep**, especifica que ambas tablas de datos sin procesar deberán reducirse a su intersección común antes de ser almacenadas en Qlik Sense.

Sintaxis:

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

Argumentos:

Argumentos

Argumento	Descripción
tablename	Tabla designada que debe compararse con la tabla cargada.
loadstatement o selectstatement	La sentencia **LOAD** o **SELECT** para la tabla cargada.

Ejemplo

Script de carga

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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 ];
```

Resultado

Tabla resultante

Column1	Column2	Column3
A	B	C
1	aa	xx

Explicación

Este ejemplo muestra el resultado de Inner Join cuando solo se unen los valores presentes tanto en la primera tabla (izquierda) como en la segunda (derecha).

IntervalMatch

El prefijo **IntervalMatch** se utiliza para crear una tabla que coincida con valores numéricos discretos con uno o más intervalos numéricos y, opcionalmente, que coincida con los valores de una o varias claves adicionales.

Sintaxis:

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

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

El prefijo **IntervalMatch** debe colocarse antes de una sentencia **LOAD** o **SELECT** que carga los intervalos. El campo que contiene los puntos de datos discretos (Time en el ejemplo inferior) y las claves adicionales ya deben haberse cargado en Qlik Sense antes de la sentencia con el prefijo **IntervalMatch**. El prefijo no lee por sí mismo este campo desde la tabla de la base de datos. El prefijo transforma la tabla cargada de intervalos y claves en una tabla que contiene una columna adicional: los puntos de datos numéricos discretos. También amplía el número de registros de forma que la nueva tabla contiene un registro por combinación posible de puntos de datos discretos, intervalo y valor del campo(s) clave.

Los intervalos pueden solaparse y los valores discretos se enlazarán con todos los intervalos coincidentes.

Cuando el prefijo IntervalMatch se amplía con los campos clave, se utiliza para crear una tabla que haga coincidir valores numéricos discretos con uno o más intervalos numéricos, mientras que al mismo tiempo coincida con los valores de una o varias teclas adicionales.

Para evitar que se ignoren los límites de intervalos no definidos, puede ser necesario permitir que los valores NULL se asignen a otros campos que constituyen los límites inferior o superior del intervalo. Esto puede gestionarlo la sentencia **NullAsValue** o una prueba explícita que reemplaza los valores NULL por un valor numérico mucho antes o después de cualquiera de los puntos de datos numéricos discretos.

Argumentos:

Argumentos

Argumento	Descripción
matchfield	Es el campo que contiene los valores numéricos discretos que se van a enlazar con los intervalos.
keyfield	Son campos que contienen los atributos adicionales que se van a comparar en la transformación.
loadstatement orselectstatement	Debe dar como resultado una tabla, en la que el primer campo contiene el límite inferior de cada intervalo, el segundo campo contiene el límite superior de cada intervalo y, en el caso de usar la asignación de claves, el tercer campo y cualquier campo subsiguiente contienen los campos clave presentes en la sentencia **IntervalMatch**. Los intervalos están siempre cerrados, es decir, los puntos finales están incluidos en el intervalo. En caso de tener límites no numéricos, no se considera el intervalo (se descarta como indefinido).

Example 1:

En las dos tablas a continuación, la primera enumera una serie de eventos discretos y la segunda define las horas de inicio y finalización de distintos pedidos. Mediante el prefijo **IntervalMatch** es posible conectar de forma lógica las dos tablas a fin de hallar por ej. qué pedidos se vieron afectados por incidencias y qué pedidos fueron procesados por qué turnos.

```
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;
```

La tabla **OrderLog** contiene ahora una columna adicional: *Time*. El número de registros también se expande.

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: (usando keyfield)

Mismo ejemplo que el anterior, añadiendo *ProductionLine* como un campo clave.

```
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;
```

Ahora se podría crear un cuadro de tabla como éste:

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

El prefijo **join** une la tabla cargada con una tabla existente o con la última tabla de datos creada.

El efecto de unir datos es ampliar la tabla de destino con un conjunto adicional de campos o atributos, es decir, aquellos que aún no están presentes en la tabla de destino. Cualquier nombre de campo común entre el conjunto de datos de origen y la tabla de destino se utiliza para determinar cómo asociar los nuevos registros entrantes. Esto se conoce comúnmente como una "unión o join natural". Una operación de unión de Qlik puede dar lugar a que la tabla de destino resultante tenga más o menos registros de los que tenía al principio, según la singularidad de la asociación de unión y el tipo de unión empleado.

Hay cuatro tipos de uniones:

Left join

Left joins son el tipo de unión más común. Por ejemplo, si tiene un conjunto de datos de transacciones y le gustaría combinarlo con un conjunto de datos de referencia, normalmente usaría un `Left Join`. Primero cargaría la tabla de transacciones, luego cargaría el conjunto de datos de referencia a la vez que lo une mediante un prefijo `Left Join` en la tabla de transacciones ya cargada. Un `Left Join` mantendría todas las transacciones tal como están y agregaría los campos de datos de referencia complementarios donde se encuentra un resultado coincidente.

Inner join

Cuando tiene dos conjuntos de datos en los que solo le importan los resultados en los que hay una asociación coincidente, considere usar un `Inner Join`. Esto eliminará todos los registros tanto de los datos de origen cargados como de la tabla de destino si no se encuentra ninguna coincidencia. Como resultado, esto puede dejar su tabla de destino con menos registros que antes de que se llevara a cabo la operación de unión.

Outer join

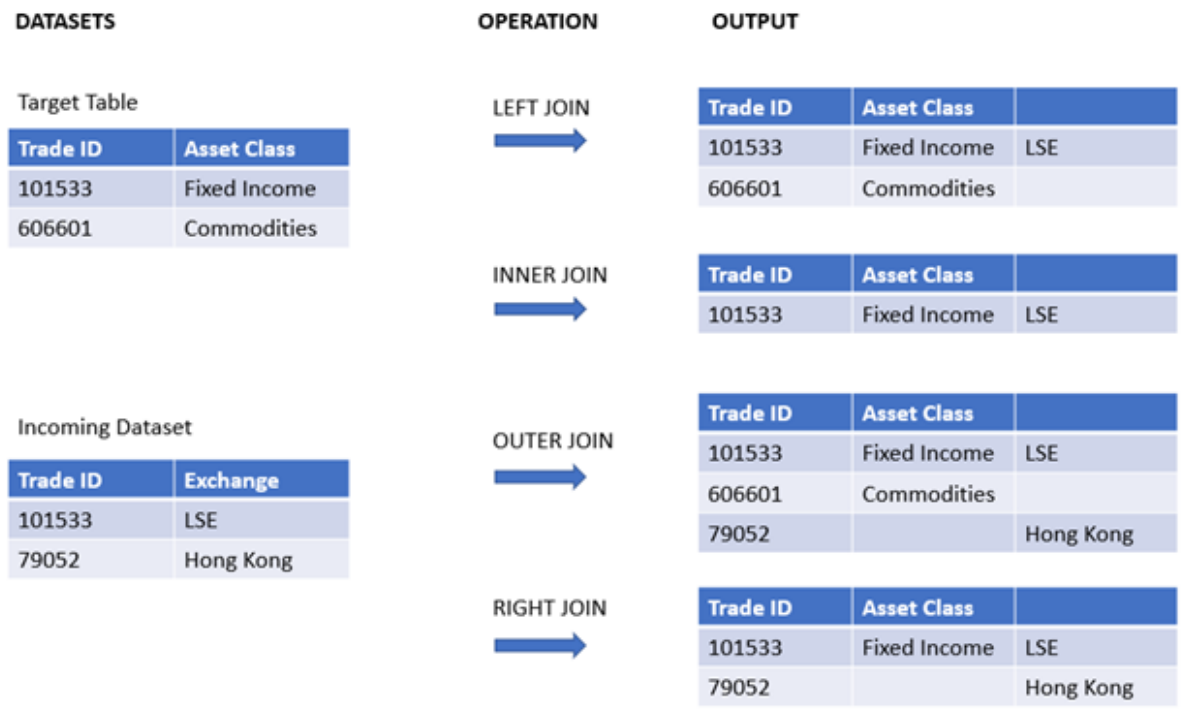
Cuando necesite conservar tanto los registros de destino como todos los registros entrantes, utilice un `Outer Join`. Cuando no se encuentra ningún resultado, todavía se conserva cada conjunto de registros, mientras que los campos del lado opuesto de la combinación permanecerán vacíos (nulos).

Si se omite la palabra clave "type", el tipo de unión predeterminado será una unión externa (outer join).

Right join

Este tipo de unión mantiene todos los registros a punto de cargarse, a la vez que reduce los registros de la tabla de destino de la unión a solo aquellos registros en los que hay una coincidencia de asociación en los registros entrantes. Este es un tipo de unión de nicho que a veces se usa como un medio para reducir una tabla de registros ya cargada previamente a un subconjunto requerido.

Conjuntos de resultados a modo de ejemplo de diferentes tipos de operaciones de unión



Si no hay nombres de campo en común entre el origen y el destino de una operación de unión, la unión dará como resultado un producto cartesiano de todas las filas; esto se denomina "unión cruzada".

Conjunto de resultados a modo de ejemplo de una operación de "cross join"

Sintaxis:

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

Argumentos

Argumento	Descripción
tablename	Tabla designada que debe compararse con la tabla cargada.
loadstatement o selectstatement	La sentencia **LOAD** o **SELECT** para la tabla cargada.

Estos temas le ayudarán a trabajar con esta función:

Temas relacionados

Tema	Descripción
Combinar tablas con Join y Keep en *Gestión de datos*	Este tema proporciona una explicación más detallada de los conceptos de “unir” y “mantener” conjuntos de datos.
Keep (page 83)	El prefijo de carga `Keep` es similar al prefijo `Join`, pero no combina los conjuntos de datos de origen y de destino. En su lugar, recorta cada conjunto de datos según el tipo de operación adoptada (inner, outer, left, o right).

Ejemplo 1 - Left join: Enriquecer una tabla de destino con un conjunto de datos de referencia

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que representa registros de cambios, el cual se carga en una tabla denominada `Changes`. Incluye un campo clave de ID de estado.
- Un segundo conjunto de datos que representa los estados de cambio, el cual se carga y combina con los registros de cambio originales uniéndolos con un prefijo left join `Join`.

Este left join garantiza que los registros de cambios permanezcan intactos mientras se agregan atributos de estado donde se encuentre un resultado coincidente en los registros de estado entrantes en función de un ID de estado común.

Script de carga

```
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');
```

Resultados

Abra el visor del modelo de datos y observe la forma del modelo de datos. Solo hay una tabla desnormalizada presente. Es una combinación de todos los registros de cambios originales, con los atributos de estado coincidentes unidos en cada registro de cambios.

Modelo de datos internos resultante

Cambios
ID de cambio
ID de estado
Fecha de inicio programada
Fecha de finalización programada
Impacto de negocio
Estado
Subestado

Si expande la ventana de vista previa en el visor del modelo de datos, verá una parte de este conjunto de resultados completo organizado en una tabla:

Vista previa de la tabla de cambios en el visor del modelo de datos

ID de cambio	**ID de estado**	**Fecha de inicio programada**	**Fecha de finalización programada**	**Impacto de negocio**	**Estado**	**Subestado**
10030	4	19/01/2022	23/02/2022	Ningún valor que mostrar	Cerrado	Cancelado
10031	3	20/01/2022	25/03/2022	Baja	Cerrado	Finalizado
10015	3	04/01/2022	15/02/2022	Baja	Cerrado	Finalizado
10103	1	02/04/2022	29/05/2022	Medio	Abrir	No iniciado
10116	1	15/04/2022	24/04/2022	Ningún valor que mostrar	Abrir	No iniciado
10134	1	03/05/2022	08/07/2022	Baja	Abrir	No iniciado
10264	1	10/09/2022	17/10/2022	Medio	Abrir	No iniciado
10040	1	29/01/2022	22/04/2022	Ningún valor que mostrar	Abrir	No iniciado
10323	1	08/11/2022	26/11/2022	Alta	Abrir	No iniciado
10187	2	25/06/2022	24/08/2022	Baja	Abrir	Iniciado

ID de cambio	ID de estado	Fecha de inicio programada	Fecha de finalización programada	Impacto de negocio	Estado	Subestado
10185	2	23/06/2022	08/09/2022	Ningún valor que mostrar	Abrir	Iniciado
10220	2	28/07/2022	06/09/2022	Ningún valor que mostrar	Abrir	Iniciado
10326	2	11/11/2022	05/12/2022	Ningún valor que mostrar	Abrir	Iniciado
10138	2	07/05/2022	03/08/2022	Ningún valor que mostrar	Abrir	Iniciado
10334	2	19/11/2022	06/02/2023	Baja	Abrir	Iniciado

Dado que la quinta fila de la tabla Estado (ID. de estado: "5", Estado: "Cerrado", Subestado: "Obsoleto") no corresponde a ninguno de los registros de la tabla Cambios, la información de esta fila no aparece en el conjunto de resultados anterior.

Vuelva al Editor de carga de datos. Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `Status`.

Añada esta medida:

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

Ahora puede inspeccionar la cantidad de cambios por estado.

Tabla de resultados

Estado	=Count([Change ID])
Abrir	12
Cerrado	3

Ejemplo 2 – Inner join: Combinar solo registros coincidentes

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que representa registros de cambios, el cual se carga en una tabla denominada `Changes`.

- Un segundo conjunto de datos que representa los registros de cambios que se originan en el sistema de origen `JIRA`. Esto se carga y se combina con los registros originales uniéndolos con un prefijo de carga `Inner Join`.

Este `Inner Join` garantiza que solo se mantengan los cinco registros de cambios que se encuentran en ambos conjuntos de datos.

Script de carga

```
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');
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `Source System`
- `Change ID`

- `Business Impact`

Ahora puede revisar los cinco registros resultantes. La tabla resultante de un `Inner Join` solo incluirá registros con información coincidente en ambos conjuntos de datos.

Tabla de resultados

Sistema de origen	ID de cambio	Impacto de negocio
JIRA	10030	Ningún valor que mostrar
JIRA	10134	Baja
JIRA	10220	Ningún valor que mostrar
JIRA	10323	Alta
JIRA	10334	Baja

Ejemplo 3 – Outer join: Combinar conjuntos de registros superpuestos

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que representa registros de cambios, el cual se carga en una tabla denominada `Changes`.
- Un segundo conjunto de datos que representa los registros de cambios que se originan en el sistema de origen `JIRA`, el cual se carga y se combina con los registros originales uniéndolos con un prefijo de carga `Outer Join`.

Esto garantiza que se conserven todos los registros de cambios superpuestos de ambos conjuntos de datos.

Script de carga

```
// 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');
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

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

Ahora puede revisar los 10 registros resultantes.

Tabla de resultados

Sistema de origen	ID de cambio	Impacto de negocio
JIRA	10030	Ningún valor que mostrar
JIRA	10134	Baja
JIRA	10220	Ningún valor que mostrar
JIRA	10323	-
JIRA	10334	Baja
JIRA	10597	-
-	10015	Baja
-	10031	Baja
-	10040	Ningún valor que mostrar
-	10138	Ningún valor que mostrar

Ejemplo 4 – Right join: Recortar una tabla de destino por un conjunto de datos maestro secundario

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que representa registros de cambios, el cual se carga en una tabla denominada `Changes`.
- Un segundo conjunto de datos que representa los registros de cambios que se originan en el sistema de origen `Teamwork`. Esto se carga y se combina con los registros originales al unirlo con un prefijo de carga `Right Join`.

Esto garantiza que solo se mantengan los registros de cambios `Teamwork`, sin perder ningún registro `Teamwork` si la tabla de destino no tiene ningún `Change ID`.

Script de carga

```
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');
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

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

Ahora puede revisar los cinco registros resultantes.

Tabla de resultados

Sistema de origen	ID de cambio	Impacto de negocio
Teamwork	10015	Baja
Teamwork	10031	Baja
Teamwork	10040	Ningún valor que mostrar
Teamwork	10103	Medio
Teamwork	50231	-

Keep

El prefijo **keep** es similar al prefijo **join**. Al igual que el prefijo **join**, compara la tabla cargada con una tabla ya existente o con la última tabla de datos creada anteriormente, pero en lugar de unir la tabla cargada con una tabla existente, tiene el efecto de reducir una o ambas tablas antes de que se almacenen en Qlik Sense, en función de la intersección de datos de la tabla. La comparación final equivale a aplicar un join natural entre todos los campos comunes. La diferencia está en que las dos tablas no se unen, sino que se almacenan en Qlik Sense como dos tablas independientes de distinto nombre.

Sintaxis:

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

El prefijo **keep** debe ir precedido por uno de los prefijos **inner**, **left** o **right**.

El prefijo explícito **join** en el lenguaje de script de Qlik Sense realiza una unión completa (full join) de las dos tablas. El resultado es una sola tabla. En muchos casos los joins dan como resultado unas tablas muy grandes. Una de las principales características de Qlik Sense es su capacidad de hacer asociaciones entre múltiples tablas en lugar de unirlas (mediante join). Esto permite ahorrar mucho espacio en la memoria e incrementar la velocidad de procesamiento, lo que se traduce en una enorme flexibilidad. Los joins explícitos deben evitarse por lo general en los scripts de Qlik Sense. La funcionalidad keep se ha diseñado para reducir el número de casos en los que se tengan que usar joins explícitos.

Argumentos:

Argumentos

Argumento	Descripción
tablename	Tabla designada que debe compararse con la tabla cargada.
loadstatement o selectstatement	La sentencia **LOAD** o **SELECT** para la tabla cargada.

Ejemplo:

```
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

Los prefijos **Join** y **Keep** pueden ir precedidos por el prefijo **left**.

Si se usan antes de **join** especifica que deberá usarse un left join. La tabla resultante solo contendrá combinaciones de valores de campo de las tablas iniciales donde los valores de campos de enlace se representan en la primera tabla. Si se usa antes de **keep**, especifica que la segunda tabla de datos sin procesar debe reducirse a su intersección común con la primera tabla, antes de ser almacenada en Qlik Sense.

¿Estaba buscando la función de cadena por el mismo nombre? Vea: Left (page 1478)

Sintaxis:

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

Argumentos:

Argumentos

Argumento	Descripción
tablename	Tabla designada que debe compararse con la tabla cargada.
loadstatement o selectstatement	La sentencia **LOAD** o **SELECT** para la tabla cargada.

Ejemplo

Script de carga

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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 ];
```

Resultado

Tabla resultante

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

Explicación

Este ejemplo muestra el resultado de Left Join cuando solo se unen los valores presentes en la primera tabla (izquierda).

Correspondencia

El prefijo **mapping** se utiliza para crear una tabla de correspondencia que se puede usar, por ejemplo, para reemplazar valores de campo y nombres de campo durante la ejecución de script.

Sintaxis:

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

El prefijo **mapping** se puede colocar delante de una o una sentencia **LOAD** o **SELECT** y almacenará el resultado de la sentencia de carga como una tabla de correspondencia. Mapping ofrece una manera eficaz de sustituir valores de campo durante la ejecución del script, por ej. reemplazando US, U.S. o América por USA. Una tabla de enlace se compone de dos columnas, la primera contiene los valores de comparación y la segunda contiene los valores de correspondencia deseados. Las tablas de correspondencia se almacenan temporalmente en la memoria y se eliminan automáticamente tras la ejecución de script.

Se puede acceder al contenido de la tabla de correspondencia usando, por ejemplo, la sentencia **Map ... Using**, la sentencia **Rename Field**, la función **Applymap()** o la función **Mapsubstring()**.

Ejemplo:

En este ejemplo, se ha cargado una lista de comerciales con un código de país que representa su país de residencia. Se utiliza una tabla que asigna un código de país a un país para reemplazar el código de país por el nombre del país. Solo tres países están definidos en la tabla de asignación, otros códigos de país están asignados a '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';
```

La tabla resultante tiene el siguiente aspecto:

Mapping table

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

Merge

El prefijo **Merge** se puede añadir a cualquier sentencia **LOAD** o **SELECT** en el script para especificar que la tabla cargada debe fusionarse en otra tabla. También especifica que esta sentencia debe ejecutarse en una carga parcial.

El caso de uso típico es cuando carga un registro de cambios y desea usarlo para aplicar inserts, updates, y deletes a una tabla existente.

Para que la carga parcial funcione correctamente, la app debe abrirse con datos antes de que se active una carga parcial.

Realice una carga parcial con el botón **Cargar**. También puede usar Qlik Engine JSON API.

Sintaxis:

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

Argumentos:

Argumentos

Argumento	Descripción
only	Un cualificador opcional que indica que la sentencia solo debe ejecutarse durante las cargas parciales. La sentencia se ignora durante las cargas normales (no parciales).
SequenceNoField	El nombre del campo que contiene una fecha-hora o un número de secuencia que define el orden de las operaciones.
SequenceNoVar	El nombre de la variable a la que se le asigna el valor máximo de SequenceNoField de la tabla que se fusiona.
ListOfKeys	Una lista de nombres de campo separados por comas que especifican la clave principal.
Operation	El primer campo de la sentencia de carga debe contener la siguiente operación como una cadena de texto: "Insert", "Update" o "Delete". También se aceptan: "i", "u" y "d".

Funcionalidad general

Durante una carga normal (no parcial), la construcción **Merge LOAD** funciona como una sentencia **Load** normal, pero con la funcionalidad adicional de eliminar registros obsoletos más antiguos y registros marcados para su eliminación. El primer campo de la sentencia **Load** debe contener información sobre la operación: Insert, Update o Delete.

Para cada registro cargado, el identificador de registro se compara con los registros cargados previamente y solo se mantendrá el último registro (según el número de secuencia). Si el último registro está marcado con Delete, no se conservará ninguno.

Tabla destino

La tabla que se ha de modificar viene determinada por el conjunto de campos. Si ya existe una tabla con el mismo conjunto de campos (excepto el primer campo; la operación), esta será la tabla relevante para modificar. Alternativamente, se puede usar un prefijo **Concatenate** para especificar la tabla. Si no se determina la tabla de destino, el resultado de la construcción **Merge LOAD** se almacena en una nueva tabla.

Si se usa el prefijo Concatenate, la tabla resultante tiene un conjunto de campos correspondientes a la unión de la tabla existente y la entrada a la fusión. Por lo tanto, la tabla de destino puede obtener más campos que el registro de cambios que se usa como entrada para la fusión.

Una carga parcial hace lo mismo que una carga completa. Con la diferencia de que una carga parcial rara vez crea una nueva tabla. A menos que haya utilizado la cláusula **Only**, siempre existe una tabla de destino con el mismo conjunto de campos de la ejecución del script anterior.

Número de secuencia

El registro de cambios cargado es un registro acumulado, es decir, contiene cambios que ya se han cargado, el parámetro SequenceNoVar se puede utilizar en una cláusula **Where** para limitar la cantidad de datos de entrada. Se podría hacer luego que el **Merge LOAD** cargue solo registros en los que el campo SequenceNoField sea mayor que SequenceNoVar. Al finalizar, **Merge LOAD** asigna un nuevo valor al SequenceNoVar con el valor máximo que se ve en el campo SequenceNoField.

Operaciones

La instrucción **Merge LOAD** puede tener menos campos que la tabla de destino. Las diferentes operaciones tratan los campos faltantes de manera diferente:

Insertar: Los campos que faltan en **Merge LOAD**, pero que existen en la tabla de destino, obtienen un NULL en la tabla de destino.

Eliminar: Los campos que faltan no afectan al resultado. Los registros relevantes se eliminan de todos modos.

Actualizar: Los campos listados en **Merge LOAD** se actualizan en la tabla de destino. Los campos que faltan no se modifican. Esto significa que las dos instrucciones siguientes no son idénticas:

- Merge on Key Concatenate Load 'U' as Operation, Key, F1, Null() as F2 From ...;
- Merge on Key Concatenate Load 'U' as Operation, Key, F1 From ...;

La primera instrucción actualiza los registros enumerados y cambia F2 a NULL. La segunda no cambia F2, sino que deja los valores en la tabla de destino.

Ejemplos

Ejemplo 1: Combinación simple con la tabla especificada

En este ejemplo, se carga una tabla inline denominada Persons, con tres filas. **Merge** luego cambia la tabla de la siguiente manera:

- Añade la fila *Mary, 4*.
- Elimina la fila *Steven, 3*.
- Asigna el número *5* a *Jake*.

La variable *LastChangeDate* se establece en el valor máximo en la columna *ChangeDate* después de que se ejecute **Merge**.

Script de carga

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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
];
```

Resultado

Antes del **Merge Load**, la tabla resultante aparece de la siguiente manera:

Resulting table

Name	**Number**
Jake	3
Jill	2
Steven	3

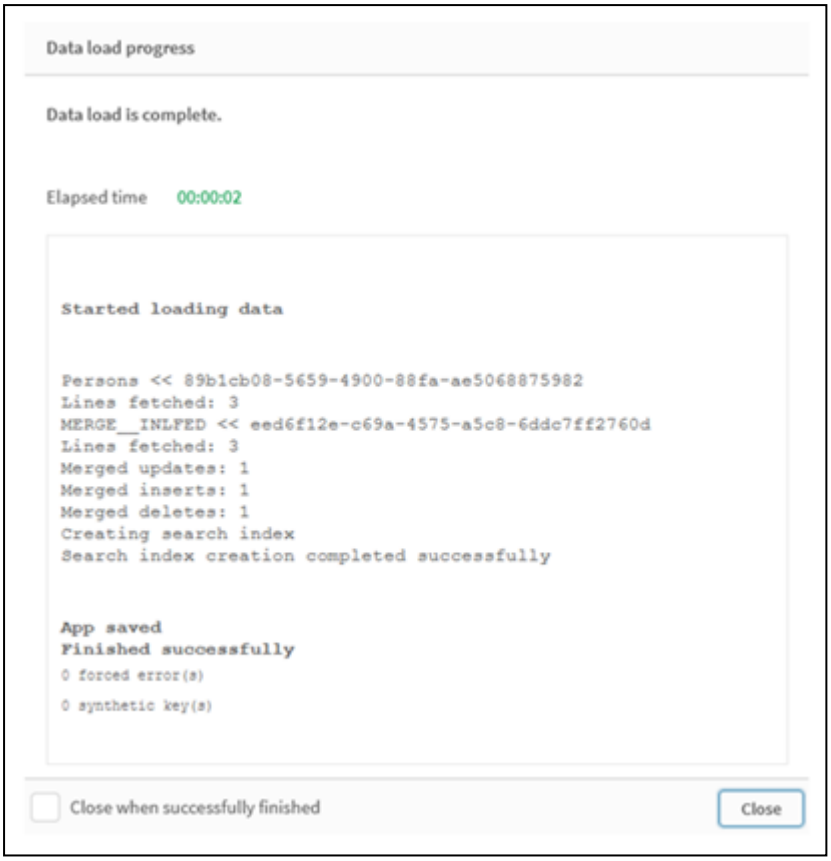
Después de la carga de **Merge Load**, la tabla aparece de la siguiente manera:

Resulting table

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

Cuando se cargan los datos, el cuadro de diálogo **Progreso de la carga de datos** muestra las operaciones que se realizan:

El cuadro de diálogo de progreso de carga de datos



Ejemplo 2: Script de carga de datos con campos que faltan

En este ejemplo, se cargan los mismos datos que arriba, pero ahora con una identificación para cada persona.

Merge cambia la tabla de la siguiente manera:

- Añade la fila *Mary, 4*.
- Elimina la fila *Steven, 3*.
- Asigna el número *5* a *Jake*.
- Asigna el número *6* a *Jill*.

Script de carga

Aquí utilizamos dos instrucciones **Merge Load**, una para "Insertar" y "Eliminar", y otra para "Actualizar".

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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
];
```

Resultado

Después de la carga de **Merge Load**, la tabla aparece de la siguiente manera:

Resulting table

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

Tenga en cuenta que la segunda instrucción **Merge** no incluye el campo **Name** y, como consecuencia, los nombres no han cambiado.

Ejemplo 3: Script de carga de datos: carga parcial mediante una cláusula Where con ChangeDate

En el ejemplo siguiente, el argumento **Only** especifica que el comando **Merge** solo se ejecutará durante una carga parcial. Las actualizaciones se filtrarán en función del LastChangeDate capturado previamente. Una vez que **Merge** haya finalizado, a la variable LastChangeDate se le asignará el valor máximo de la columna ChangeDate procesada durante la fusión.

Script de carga

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

NoConcatenate

El prefijo **NoConcatenate** obliga a que dos tablas cargadas con conjuntos de campos idénticos se traten como dos tablas internas separadas, cuando de lo contrario se concatenarían automáticamente.

Sintaxis:

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

De forma predeterminada, si se carga una tabla que contiene un número idéntico de campos y nombres de campo que coinciden con los de una tabla cargada anteriormente en el script, Qlik Sense concatenará automáticamente estas dos tablas. Esto sucederá incluso si la segunda tabla tiene un nombre diferente.

Sin embargo, si se incluye el prefijo de script `NoConcatenate` antes de la instrucción de carga o la instrucción de selección de la segunda tabla, estas dos tablas se cargarán por separado.

Un uso típico de `NoConcatenate` es cuando necesitamos crear una copia temporal de una tabla para realizar algunas transformaciones temporales en esa copia, mientras conservamos una copia de los datos originales. `NoConcatenate` garantiza que se pueda hacer esa copia sin tener que volver a agregarla implícitamente a la tabla de origen.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo de función

Ejemplo	**Resultado**
`Source: LOAD A,B from file1.csv; CopyOfSource: NoConcatenate LOAD A,B resident Source;`	Se carga una tabla con `A` y `B` como medidas. Se carga una segunda tabla con los mismos campos utilizando la variable `NoConcatenate`.

Ejemplo 1: concatenación implícita

Script de carga y resultados

Vista general

En este ejemplo, agregará dos scripts de carga en orden secuencial.

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos inicial con fechas y cantidades que se envía a una tabla denominada `Transactions`.

Primer script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

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

Primera tabla de resultados

id	fecha	cantidad
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

Segundo script de carga

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un segundo conjunto de datos con campos idénticos se envía a una tabla denominada `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
];
```

Resultados

Cargue los datos y vaya a la tabla.

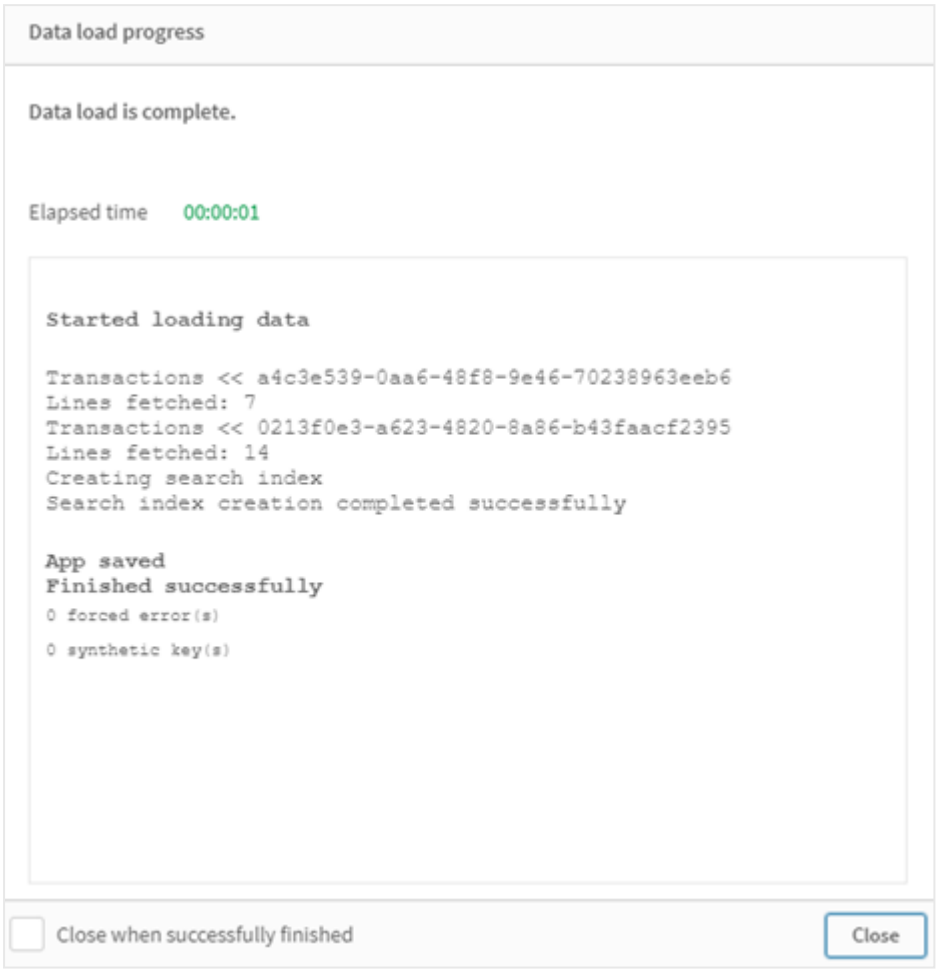
Segunda tabla de resultados

id	fecha	cantidad
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
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

Cuando se ejecuta el script, la tabla `Sales` se concatena implícitamente en la tabla existente `Transactions`, debido a que los dos conjuntos de datos comparten una cantidad idéntica de campos, con nombres de campo idénticos. Esto sucede a pesar de que la segunda etiqueta de nombre de la tabla intenta nombrar el conjunto de resultados `'Sales'`.

Puede ver que el conjunto de datos de Ventas está concatenado implícitamente observando el registro de **progreso de la carga de datos**.

Puede ver que el conjunto de datos de Ventas está concatenado implícitamente observando el registro de progreso de la carga de datos.



Ejemplo 2: escenario de caso de uso

Script de carga y resultados

Vista general

En este escenario de caso de uso, tiene:

- Un conjunto de datos de transacciones con:
    - id
    - fecha
    - importe (en libras esterlinas)
- Una tabla de divisas con:
    - Tasas de conversión de USD a GBP
- Un segundo conjunto de datos de transacciones con:

- id
- fecha
- cantidad (en USD)

Ahora cargará cinco scripts en orden secuencial.

- El primer script de carga contiene un conjunto de datos inicial con fechas y cantidades en GBP que se envía a una tabla denominada `Transactions`.
- El segundo script de carga contiene:
  - Un segundo conjunto de datos con fechas y cantidades en USD que se envía a una tabla denominada `Transactions_in_USD`.
  - El prefijo `noconcatenate` que se coloca antes de la instrucción de carga del conjunto de datos `Transactions_in_USD` para evitar la concatenación implícita.
- El tercer script de carga contiene el prefijo `join` que se usará para crear una tasa de cambio de moneda entre GBP y USD en la tabla `Transactions_in_USD`.
- El cuarto script de carga contiene el prefijo `concatenate` que agregará `Transactions_in_USD` a la tabla `Transactions` inicial.
- El quinto script de carga contiene la instrucción `drop table` que eliminará la tabla `Transactions_in_USD`, sus datos se han concatenado en la tabla `Transactions`.

Primer script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

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

Resultados del primer script de carga

id	fecha	cantidad
1	12/30/2018	23.56

id	fecha	cantidad
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

La tabla muestra el conjunto de datos inicial con los importes en libras esterlinas.

Segundo script de carga

```
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
];
```

Resultados

Cargue los datos y vaya a la tabla.

Resultados del segundo script de carga

id	fecha	cantidad
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
8	01/01/2019	164.27
9	01/03/2019	384.00
10	01/06/2019	25.82

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

Verá que se ha agregado el segundo conjunto de datos de la tabla `Transactions_in_USD`.

Tercer script de carga

Este script de carga une una tasa de cambio de moneda de USD a GBP a la tabla `Transactions_in_USD`.

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

Resultados

Cargue los datos y vaya al visor del modelo de datos. Seleccione la tabla `Transactions_in_USD` y verá que cada registro tiene un valor de campo "tasa" de 0.7.

Cuarto script de carga

Usando la carga residente, este script de carga concatenará la tabla `Transactions_in_USD` con la tabla `Transactions` después de convertir las cantidades a USD.

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

Resultados

Cargue los datos y vaya a la tabla. Verá nuevas entradas con los importes en GBP de las líneas ocho a catorce.

Resultados del cuarto script de carga

id	fecha	cantidad
1	12/30/2018	23.56
2	12/07/2018	556.31
3	12/16/2018	5.75
4	12/22/2018	125.00

id	fecha	cantidad
5	12/22/2018	484.21
6	12/22/2018	59.18
7	12/23/2018	177.42
8	01/01/2019	114.989
8	01/01/2019	164.27
9	01/03/2019	268.80
9	01/03/2019	384.00
10	01/06/2019	18.074
10	01/06/2019	25.82
11	01/09/2019	218.40
11	01/09/2019	312.00
12	01/15/2019	3.192
12	01/15/2019	4.56
13	01/16/2019	63.168
13	01/16/2019	90.24
14	01/18/2019	13.524
14	01/18/2019	19.32

Quinto script de carga

Este script de carga eliminará las entradas duplicadas de la cuarta tabla de resultados del script de carga, dejando solo las entradas con cantidades en GBP.

```
drop tables Transactions_in_USD;
```

Resultados

Cargue los datos y vaya a la tabla.

Resultados del quinto script de carga

id	fecha	cantidad
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

id	fecha	cantidad
6	12/22/2018	59.18
7	12/23/2018	177.42
8	01/01/2019	114.989
9	01/03/2019	268.80
10	01/06/2019	18.074
11	01/09/2019	218.40
12	01/15/2019	3.192
13	01/16/2019	63.168
14	01/18/2019	13.524

Tras cargar el quinto script de carga, la tabla de resultados muestra las catorce transacciones que existían en ambos conjuntos de datos de transacciones; sin embargo, las transacciones 8-14 han visto sus cantidades convertidas a GBP.

Si eliminamos el prefijo `NoConcatenate` que se usó antes de `Transactions_in_USD` en el segundo script de carga, el script fallará con el error: “Tabla "`Transactions_in_USD`" no encontrada”. Esto se debe a que la tabla `Transactions_in_USD` se habría concatenado automáticamente en la tabla `Transactions` original.

Only

La palabra clave de script **Only** se utiliza como función de agregación o como parte de la sintaxis en los prefijos de carga parcial **Add**, **Replace** y **Merge**.

Outer

El prefijo explícito **Join** puede ir precedido por el prefijo **Outer** para especificar un outer join. En un outer join se generan todas las combinaciones entre las dos tablas. La tabla resultante contendrá por tanto combinaciones de valores de campo de las dos tablas originales donde los valores de campos de enlace se representan en una o ambas tablas. La palabra clave **Outer** es opcional y es el tipo de unión predeterminado que se usa cuando no se especifica un prefijo join.

Sintaxis:

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

Argumentos:

Argumentos

Argumento	Descripción
tablename	Tabla designada que debe compararse con la tabla cargada.
loadstatement o selectstatement	La sentencia **LOAD** o **SELECT** para la tabla cargada.

Ejemplo

Script de carga

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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 ];
```

Tabla resultante

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

Explicación

En este ejemplo, las dos tablas, Table1 y Table2, se fusionan en una sola tabla denominada Table1. En casos como este, el prefijo **outer** a menudo se usa para unir varias tablas en una sola tabla a fin de realizar agregaciones sobre los valores de una sola tabla.

carga parcial

Una carga completa siempre comienza eliminando todas las tablas en el modelo de datos existente y luego ejecutando el script de carga.

Una carga parcial no hace esto. En su lugar, mantiene todas las tablas en el modelo de datos y después ejecuta solo las sentencias **Load** y **Select** precedidas por un prefijo **Add**, **Merge** o **Replace**. Otras tablas de datos no se ven afectadas por el comando. El argumento **only** indica que la sentencia debe ejecutarse solo durante cargas parciales y debe ignorarse durante cargas completas. La tabla siguiente resume la ejecución de instrucciones para cargas parciales y completas.

Sentencia	carga completa	carga parcial
Load ...	La instrucción se ejecutará	La instrucción no se ejecutará
Añadir/Reemplazar/Fusionar Cargar...	La instrucción se ejecutará	La instrucción se ejecutará
Añadir/Reemplazar/Fusionar/Cargar...	La instrucción no se ejecutará	La instrucción se ejecutará

Las cargas parciales tienen varias ventajas en comparación con las cargas completas:

- Son más rápidas, porque solo es necesario cargar los datos recientemente modificados. Con conjuntos de datos muy extensos la diferencia es significativa.
- Se consume menos memoria porque se cargan menos datos.
- Es más fiable, porque las consultas a los datos de origen se ejecutan más rápido, lo que reduce el riesgo de problemas de red.

Para que la carga parcial funcione correctamente, la app debe abrirse con datos antes de que se active una carga parcial.

Realice una carga parcial con el botón **Cargar**. También puede usar Qlik Engine JSON API.

Limitaciones

Una carga parcial fallará si hay comandos con referencias a tablas que existieron durante la carga completa, pero no durante la carga parcial.

Ejemplo

Comandos a modo de ejemplo

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

Donde <Table_removed_after_full_reload> es una tabla que existía en la carga completa, pero no en la carga parcial.

Solución

Como solución, puede rodear el comando con la siguiente sentencia if:

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

Una carga parcial puede eliminar valores de los datos. Sin embargo, esto no se reflejará en la lista de valores distintos, que es una tabla mantenida internamente. Así, tras una carga parcial, la lista contendrá todos los valores distintos que hayan existido en el campo desde la última recarga completa, que pueden ser más de los que existen actualmente tras la recarga parcial. Esto afecta al resultado de las funciones FieldValueCount() y FieldValue(). La función FieldValueCount() podría devolver un número mayor que el número actual de valores de campo.

Ejemplo

Ejemplo 1

Script de carga

Agregue el script de ejemplo a su app y realice una carga parcial. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

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

Resultado

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

Explicación

La instrucción solo se ejecuta durante una carga parcial. Si se omite el prefijo "distinct", la cuenta del campo **Num** aumentará con cada carga parcial subsiguiente.

Ejemplo 2

Script de carga

Agregue el script de ejemplo a su app. Realice una carga completa y vea el resultado. A continuación, realice una carga parcial y vea el resultado. Para ver los resultados, agregue los campos enumerados en la columna de resultados a una hoja de su app.

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

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

Resultado

Output table after full reload

ID	Value
1	1
2	2
3	3
4	4

ID	Value
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

Explicación

La primera tabla se carga durante una carga completa y la segunda tabla simplemente reemplaza a la primera tabla durante una carga parcial.

Replace

La palabra clave de script **Replace** se utiliza como una función de cadena o como prefijo en la carga parcial.

Replace

El prefijo **Replace** se puede añadir a cualquier sentencia **LOAD** o **SELECT** en el script para especificar que la tabla cargada debe reemplazar a otra tabla. También especifica que esta sentencia debe ejecutarse en una carga parcial. El prefijo **Replace** también se puede usar en una sentencia **Map**.

Para que la carga parcial funcione correctamente, la app debe abrirse con datos antes de que se active una carga parcial.

Realice una carga parcial con el botón **Cargar**. También puede usar Qlik Engine JSON API.

Sintaxis:

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

```
Replace [only] mapstatement
```

Durante una recarga normal (no parcial), la construcción **Replace LOAD** funcionará como una sentencia **LOAD** normal pero irá precedida por una **Drop Table**. Primero se eliminará la tabla anterior, luego se generarán los registros y se almacenarán como una tabla nueva.

Si se usa el prefijo **Concatenate** o si existe una tabla con el mismo conjunto de campos, esta será la tabla relevante que hay que modificar. De lo contrario, no habrá tabla que eliminar y la construcción **Replace LOAD** será idéntica a un **LOAD** normal.

Una carga parcial hará lo mismo. La única diferencia es que siempre hay una tabla de la ejecución del script anterior para eliminar. La construcción **Replace LOAD** siempre eliminará primero la tabla anterior y luego creará una nueva.

La sentencia **Replace Map...Using** hace que la asignación se produzca también durante la ejecución parcial del script.

Argumentos:

Argumentos

Argumento	Descripción
only	Un cualificador opcional que indica que la sentencia solo debe ejecutarse durante las cargas parciales. Y debe ser ignorada durante las recargas normales (no parciales).

Ejemplos y resultados:

Ejemplo	Resultado
`Tab1: Replace LOAD * from File1.csv;`	Durante la recarga normal y parcial, la tabla Tab1 de Qlik Sense se descarta inicialmente. A partir de ahí se cargan nuevos datos de File1.csv y se almacenan en Tab1.
`Tab1: Replace only LOAD * from File1.csv;`	Durante una recarga normal, esta sentencia se ignora. Durante la recarga parcial, cualquier tabla Qlik Sense previamente nombrada Tab1 se descarta inicialmente. A partir de ahí se cargan nuevos datos de File1.csv y se almacenan en Tab1.

Ejemplo	Resultado
`Tab1: LOAD a,b,c from File1.csv; Replace LOAD a,b,c from File2.csv;`	Durante la recarga normal, el archivo File1.csv se lee primero en la tabla Tab1 de Qlik Sense, pero después se descarta de inmediato y es reemplazada por nuevos datos cargados desde File2.csv. Se pierden todos los datos de File1.csv. Durante la carga parcial, la tabla Tab1 completa de Qlik Sense se descarta inicialmente. A partir de ahí es reemplazada por nuevos datos cargados desde File2.csv.
`Tab1: LOAD a,b,c from File1.csv; Replace only LOAD a,b,c from File2.csv;`	Durante la recarga normal, los datos se cargan desde File1.csv y se almacenan en la tabla Tab1 de Qlik Sense. File2.csv se descarta. Durante la recarga parcial, la tabla Tab1 completa de Qlik Sense se descarta inicialmente. A partir de ahí es reemplazada por nuevos datos cargados desde File2.csv. Se pierden todos los datos de File1.csv.

Right

Los prefijos **Join** y **Keep** pueden ir precedidos por el prefijo **right**.

Si se usa antes de **join**, especifica que se debe usar un right join. La tabla resultante contendrá solo combinaciones de valores de campo de las dos tablas donde los valores de campos de enlace se representan en la segunda tabla. Si se usa antes de **keep**, especifica que la primera tabla de datos sin procesar debe reducirse a su intersección común con la segunda tabla antes de ser almacenada en Qlik Sense.

¿Estaba buscando la función de cadena por el mismo nombre? Vea: Right (page 1486)

Sintaxis:

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

Argumentos:

Argumentos

Argumento	Descripción
tablename	Tabla designada que debe compararse con la tabla cargada.
loadstatement o selectstatement	La sentencia **LOAD** o **SELECT** para la tabla cargada.

Ejemplo

Script de carga

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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 ];
```

Resultado

Tabla resultante

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

Explicación

Este ejemplo muestra el resultado de Right Join cuando solo se unen los valores de la segunda tabla (derecha).

Sample

El prefijo **sample** para una sentencia **LOAD** o **SELECT** se utiliza para cargar una muestra aleatoria de registros desde el origen de datos.

Sintaxis:

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

La expresión que se evalúa no define el porcentaje de registros del conjunto de datos que se cargarán en la aplicación de Qlik Sense, sino la probabilidad de que cada registro que se lea se cargue en la aplicación. En otras palabras, especificar un valor `p = 0.5` no significa que se cargará el 50% del número total de registros, sino que para cada registro habrá un 50% de posibilidades de que se cargue en la aplicación de Qlik Sense.

Argumentos

Argumento	Descripción
p	Una expresión arbitraria que devuelve un número mayor que 0 e inferior o igual a 1. El número indica la probabilidad de que se lea un determinado registro. Todos los registros se leerán pero solo algunos de ellos se cargarán en Qlik Sense.

Cuándo se utiliza

La muestra es útil cuando desea muestrear datos provenientes de una tabla grande, para comprender la naturaleza de los datos, la distribución o el contenido de los campos. Como trae un subconjunto de datos, las cargas de datos son más rápidas, lo que permite probar más rápido los scripts. A diferencia de `First`, la función `Sample` trae datos de toda la tabla, en lugar de limitarse a las primeras filas. Esto puede proporcionar una representación más precisa de los datos en algunos casos.

Los ejemplos siguientes muestran dos posibles usos del prefijo de script `Sample`:

```
Sample 0.15 SQL SELECT * from Longtable;

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

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: muestra de una tabla inline

Script de carga y resultados

Vista general

En este ejemplo, la secuencia de script carga un conjunto de datos de muestra de un conjunto de datos que contiene siete registros en una tabla denominada `Transactions` a partir de una tabla inline.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `amount`

Agregue la siguiente medida:

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

Tabla de resultados

id	fecha	=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

En la iteración de la carga utilizada en este ejemplo, se leyeron los siete registros, pero solo se cargaron cuatro registros en la tabla de datos. Cualquier carga de repetición podría dar como resultado un número diferente y un conjunto diferente de registros que se cargan en la aplicación.

Ejemplo 2: muestra de una tabla generada automáticamente

Script de carga y resultados

Vista general

En este ejemplo, usar `Autogenerate` crea un conjunto de datos de 100 registros con los campos `date`, `id` y `amount`. Sin embargo se utiliza el prefijo `Sample`, con un valor de 0,1.

Script de carga

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

Autogenerate(100);
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `amount`

Agregue la siguiente medida:

Tabla de resultados

id	fecha	=Sum(amount)
48	9/28/2013	763

id	fecha	=Sum(amount)
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

En la iteración de la carga utilizada en este ejemplo, se cargaron siete registros del conjunto de datos creado. Una vez más, cualquier carga de repetición podría dar como resultado un número diferente y un conjunto diferente de registros que se cargan en la aplicación.

Semantic

El prefijo de carga `semantic` crea un tipo especial de campo que se puede usar en Qlik Sense para conectar y administrar datos relacionales, como estructuras de árbol, datos estructurados padre-hijo con autorreferencia y/o datos que se pueden describir como un gráfico.

Tenga en cuenta que la carga semántica (`semantic`) puede funcionar de manera similar a los prefijos *Hierarchy (page 64)* y *HierarchyBelongsTo (page 67)*. Los tres prefijos se pueden usar como bloques de construcción en soluciones front-end efectivas para atravesar datos relacionales.

Sintaxis:

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

Una carga semántica espera una entrada que tenga exactamente tres o cuatro campos de ancho con una definición estricta de lo que representa cada campo ordenado, como se muestra en la siguiente tabla:

Campos de carga semántica

Nombre del campo	Descripción del campo
1er campo:	Esta etiqueta es una representación del primero de dos objetos entre los que existe una relación.
2º campo:	Esta etiqueta se usará para describir la relación "en adelante" entre el primer y el segundo objeto. Si el primer objeto es un objeto secundario y el segundo objeto es un elemento principal, puede crear una pestaña de relación que indique "principal" o "padre de" como si estuviera siguiendo la relación de elemento secundario a elemento principal.
3er campo:	Esta etiqueta es una representación del segundo de dos objetos entre los que existe una relación.
4º campo:	Este campo es opcional. Esta etiqueta describe la relación "hacia atrás" o "inversa" entre el primer y el segundo objeto. Si el primer objeto es un hijo y el segundo objeto es un padre,

Nombre del campo	Descripción del campo
	una pestaña de relación podría indicar "hijo" o "hijo de" como si estuviera siguiendo la relación de padre a hijo. Si no agrega un cuarto campo, la etiqueta del segundo campo se usará para describir la relación en cualquier dirección. En ese caso, se agrega automáticamente un símbolo de flecha como parte de la etiqueta.

El siguiente código es un ejemplo del prefijo `semantic`.

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

Está permitido y es una práctica habitual etiquetar el tercer campo de la misma manera que el primer campo. Esto crea una búsqueda autorreferenciada, de modo que puede seguir los objetos hasta los objetos relacionados, a un paso de relación cada vez. Si el tercer campo no lleva el mismo nombre, entonces el resultado final será una simple búsqueda de uno o varios objetos a su(s) vecino (s) relacional(es) directo(s) a solo un paso, lo que es un resultado de poco uso práctico.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Funciones relacionadas

Funciones	Interacción
Hierarchy (page 64)	El prefijo de carga Hierarchy se usa para dividir y organizar nodos en estructuras de datos de tipo padre-hijo y otras estructuras de datos similares a gráficos y transformarlos en tablas.
HierarchyBelongsTo (page 67)	El prefijo de carga HierarchyBelongsTo se usa para ubicar y organizar los antepasados de padre-hijo y otras estructuras de datos similares a gráficos y transformarlos en tablas.

Ejemplo: crear un campo especial para conectar relaciones utilizando el prefijo semántico

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que representa registros de relaciones geográficas, el cual se carga en una tabla denominada `GeographyTree`.
    - Cada entrada tiene un ID al principio de la línea y un ParentID al final de la línea.
- El prefijo `semantic` que agregará un campo de comportamiento especial etiquetado con `Relation`.

Script de carga

```
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;
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones.

- `Id`
- `Geography`

Luego, cree un panel de filtro con `Relation` como dimensión. Haga clic en **Edición finalizada**.

Tabla de resultados

Id	**Geografía**
1	Mundo
2	Europa
3	Asia
4	América del Norte
5	Sudamérica
6	UK
7	Alemania
8	Suecia
9	Corea del Sur
10	Corea del Norte
11	China
12	Londres
13	Birmingham

Panel de filtrado

Relación
Hijo
Padre

Haga clic en **Europe** desde la dimensión `Geography` de la tabla y clic en **Child** desde la dimensión `Relation` en el panel de filtrado. Tenga en cuenta el resultado esperado en la tabla:

Tabla de resultados que muestra los "hijos" de Europa

Id	**Geografía**
6	UK
7	Alemania
8	Suecia

Hacer clic en **Child** de nuevo mostrará lugares que son "hijos" del Reino Unido, un paso más abajo.

Tabla de resultados que muestra los "hijos" de Reino Unido

Id	Geografía
12	Londres
13	Birmingham

Unless

El prefijo y sufijo **unless** se utiliza para crear una cláusula condicional que determina si una sentencia o cláusula de salida debería evaluarse o no. Puede verse como una alternativa compacta a la sentencia **if..end if** completa.

Sintaxis:

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

La sentencia **statement** o **exitstatement** solo se ejecutará si **condition** se evalúa como False.

El prefijo **unless** se puede usar en sentencias que ya tienen una o varias sentencias adicionales, incluyendo los prefijos **when** o **unless**.

Argumentos

Argumento	Descripción
condition	Una expresión lógica que devuelve True o False.
statement	Cualquier sentencia de script de Qlik Sense, excepto las sentencias de control.
exitstatement	Una cláusula **exit for**, **exit do** o **exit sub** o una sentencia **exit script**.

Cuándo se utiliza

La sentencia `Unless` devuelve un resultado booleano. Por lo general, este tipo de función se utilizará como condición cuando el usuario desee cargar o excluir partes del script de forma condicional.

Las líneas siguientes muestran tres ejemplos de cómo se puede utilizar la función `Unless`:

```
exit script unless A=1;

unless A=1 LOAD * from myfile.csv;

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

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: el prefijo Unless

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- La creación de la variable A, a la que se le da un valor de 1.
- Un conjunto de datos que se carga en una tabla denominada Transacciones, a menos que la variable A = 2.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- id
- date
- amount

Tabla de resultados

id	fecha	cantidad
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

Como a la variable A se le asigna el valor de 1 al comienzo del script, se evalúa la condición que sigue al prefijo `Unless`, devolviendo un resultado de `FALSE`. Como resultado, el script continúa ejecutando la sentencia `Load`. En la tabla de resultados se pueden ver todos los registros de la tabla `Transactions`.

Si el valor de esta variable se establece en 2, no se cargarán datos en el modelo de datos.

Ejemplo 2: el sufijo Unless

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga comienza cargando un conjunto de datos inicial en una tabla denominada `Transactions`. Después el script finaliza, a menos que haya menos de 10 registros en la tabla `Transactions`.

Si esta condición no da como resultado la finalización del script, se concatena un conjunto adicional de transacciones en la tabla `Transactions` y se repite este proceso.

Script de carga

```
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 ;
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- id
- date
- amount

Tabla de resultados

id	fecha	cantidad
1	08/30/2018	23.56
2	09/07/2018	556.31
3	09/16/2018	5.75
4	09/22/2018	125.00

id	fecha	cantidad
5	09/22/2018	484.21
6	09/22/2018	59.18
7	09/23/2018	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

Hay siete registros en cada uno de los tres conjuntos de datos del script de carga.

El primer conjunto de datos (con los `id` de transacción 1 a 7) se carga en la aplicación. La condición `Unless` evalúa si hay menos de 10 filas en la tabla `Transactions`. Esto devuelve `TRUE` y, por lo tanto, el segundo conjunto de datos (con los `id` de transacción 8 a 14) se carga en la aplicación. La segunda condición `Unless` evalúa si hay menos de 10 registros en la tabla `Transactions`. Esto devuelve `FALSE` y por lo tanto el script finaliza.

Ejemplo 3: Múltiples prefijos Unless

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

En este ejemplo, un conjunto de datos que contiene una transacción se crea como una tabla llamada `Transactions`. Luego se activa un bucle "for", en el que se evalúan dos sentencias unless anidadas:

1. A menos que haya más de 100 registros en la tabla `Transactions`
2. A menos que el número de registros en la tabla `Transactions` sea un múltiplo de 6

Si estas condiciones son `FALSE`, se generan y concatenan otros siete registros en la tabla existente `Transactions`. Este proceso se repite hasta que una de las dos transacciones devuelve un valor de `TRUE`.

Script de carga

```
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
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:`id`.

Tabla de resultados

id
0
1
2
3
4
5
+30 filas más

Las sentencias anidadas unless que ocurren en el bucle "for" devuelven lo siguiente:

1. ¿Hay más de 100 filas en la tabla `Transactions`?
2. ¿El número total de registros en la tabla `Transactions` es un múltiplo de 6?

Cada vez que ambas sentencias unless devuelvan un valor de `FALSE`, se generan y concatenan otros siete registros en la tabla `Transactions` existente.

Estas sentencias devuelven un valor de `FALSE` cinco veces, momento en el que hay un total de 36 filas de datos en la tabla `Transactions`.

Después de esto, la segunda sentencia `Unless` devuelve un valor de `TRUE` y, por lo tanto, la sentencia de carga que sigue a esto ya no se ejecutará.

When

El prefijo y sufijo **when** se utiliza para crear una cláusula condicional que determina si una sentencia o cláusula de salida debería ejecutarse o no. Puede verse como una alternativa compacta a la sentencia **if..end if** completa.

Sintaxis:

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

Tipo de datos que devuelve: Booleano

En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.

Las instrucciones **statement** o **exitstatement** solo se ejecutarán si la condición se evalúa como verdadera, TRUE.

El prefijo `When` se puede usar en sentencias que ya tienen una o varias sentencias adicionales, incluyendo los prefijos `When` o `Unless`.

Cuándo se utiliza

La sentencia `When` devuelve un resultado booleano. Por lo general, este tipo de función se utilizará como condición cuando el usuario desee cargar o excluir partes de un script.

Argumentos

Argumento	Descripción
condition	Una expresión lógica que devuelve TRUE o FALSE
statement	Cualquier sentencia de script de Qlik Sense, excepto las sentencias de control.
exitstatement	Una cláusula **exit for**, **exit do** o **exit sub** o una sentencia **exit script**.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
`exit script when A=1;`	Cuando la sentencia `A=1` devuelve TRUE, el script se detendrá.
`when A=1 LOAD * from myfile.csv;`	Cuando la sentencia `A=1` devuelve TRUE, el archivo `myfile.csv` se cargará.
`when A=1 unless B=2 drop table Tab1;`	Cuando la sentencia `A=1` devuelve TRUE, y si `B=2` devuelve FALSE, entonces se descartará la tabla `Tab1`.

Ejemplo 1: el prefijo When

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos con fechas y cantidades que se envía a una tabla denominada "`Transactions`".
- La sentencia `Let` que establece que la variable `A` se ha creado y tiene el valor de `1`.
- La condición `When` que porporciona la condición de que si `A` es igual a `1`, entonces el script continuará cargándose.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

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

Tabla de resultados

id	fecha	cantidad
1	08/30/2018	23.56
2	09/07/2018	556.31

id	fecha	cantidad
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

Como a la variable `A` se le asigna el valor de `1` al comienzo del script, se evalúa la condición que sigue al prefijo `When`, devolviendo un resultado de TRUE. Debido a que devuelve un resultado TRUE, el script continúa ejecutando la instrucción de carga. Se pueden ver todos los registros de la tabla de resultados.

Si el valor de esta variable se estableciera en cualquier valor que no fuera igual a `1`, no se cargarían datos en el modelo de datos.

Ejemplo 2: el sufijo When

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Tres conjuntos de datos con fechas y cantidades que se envían a una tabla denominada "`Transactions`".
    - El primer conjunto de datos contiene las transacciones 1-7.
    - El segundo conjunto de datos contiene las transacciones 8-14.
    - El tercer conjunto de datos contiene las transacciones 15-21.
- Una condición `When` que determina si la tabla "`Transactions`" contiene más de diez filas. Si alguna de las sentencias `When` se evalúa como TRUE, el script de carga se detendrá. Esta condición se coloca al final de cada uno de los tres conjuntos de datos.

Script de carga

```
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 ;
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- id
- date
- amount

Tabla de resultados

id	fecha	cantidad
1	08/30/2018	23.56
2	09/07/2018	556.31
3	09/16/2018	5.75

id	fecha	cantidad
4	09/22/2018	125.00
5	09/22/2018	484.21
6	09/22/2018	59.18
7	09/23/2018	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

Hay siete transacciones en cada uno de los tres conjuntos de datos. El primer conjunto de datos contiene las transacciones 1-7 y se carga en la aplicación. La condición `When` que sigue a esta instrucción de carga se evalúa como FALSE porque hay menos de diez filas en la tabla "`Transactions`". El script de carga continúa con el siguiente conjunto de datos.

El segundo conjunto de datos contiene las transacciones 8-14 y se carga en la aplicación. La segunda condición `When` devuelve TRUE porque hay más de diez filas en la tabla "`Transactions`". Por lo tanto, el script termina.

Ejemplo 3: múltiples prefijos When

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se crea un conjunto de datos que contiene una única transacción como una tabla llamada "`Transactions`".
- Un bucle `For` que se activa contiene dos condiciones `When` anidadas que evalúan si:
    1. Hay menos de 100 registros en la tabla "`Transactions`".
    2. El número de registros en la tabla "`Transactions`" no es un múltiplo de 6.

Script de carga

```
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
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

- `id`

La tabla de resultados solo muestra los primeros cinco ID de transacción, pero el script de carga crea 36 filas y luego finaliza una vez que se cumple la condición `When`.

Tabla de resultados

id
0
1
2
3
4
5
+30 filas más

Las condiciones anidadas `When` en el ciclo `For` evalúan las siguientes preguntas:

- ¿Hay menos de 100 filas en la tabla "`Transactions`"?
- ¿El número total de registros en la tabla "`Transactions`" no es un múltiplo de 6?

Cada vez que ambas condiciones `When` devuelven un valor de TRUE, se generan y concatenan otros siete registros en la tabla "`Transactions`" existente.

Las condiciones `When` devuelven un valor TRUE cinco veces. En ese punto, hay un total de 36 filas de datos en la tabla "`Transactions`".

Cuando se crean 36 filas de datos en la tabla "`Transactions`", la segunda instrucción `When` devuelve un valor de FALSE y, por lo tanto, la instrucción de carga que sigue a esta ya no se ejecutará.

3.3 Sentencias habituales de script

Las sentencias más comunes se utilizan habitualmente para manipular datos de varias formas. Estas sentencias pueden escribirse sobre cualquier número de filas en el script y deben terminar siempre en punto y coma ";".

Todas las palabras clave del script pueden escribirse con cualquier combinación de caracteres en mayúscula o minúscula. Los nombres de campo y de variable utilizados en las sentencias, por supuesto, son sensibles a mayúsculas.

Descripción general de las sentencias habituales de script

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Alias
La sentencia **alias** se utiliza para establecer un alias según el cual un campo se renombrará cada vez que aparezca en adelante en el script.

```
Alias fieldname as aliasname {,fieldname as aliasname}
```

Autonumber
Esta sentencia crea un valor entero único por cada valor distinto evaluado en un campo hallado durante la ejecución de script.

```
AutoNumber fields [Using namespace] ]
```

Binary
La sentencia **binary** se utiliza para cargar los datos desde otro documento QlikView, incluidos los datos de Section access.

```
Binary  [path] filename
```

comment
Ofrece una forma de mostrar los comentarios del campo (metadatos) desde bases de datos y hojas de cálculo. Los nombres de campo que no estén presentes en la app se ignorarán. Si hubiera múltiples nombres de un mismo campo, se empleará el último valor.

```
Comment field *fieldlist using mapname
Comment field fieldname with comment
```

comment table
Ofrece una forma de mostrar los comentarios de una tabla (metadatos) desde bases de datos u hojas de cálculo.

```
Comment table tablelist using mapname
Comment table tablename with comment
```

Connect

Esta funcionalidad no está disponible en Qlik Sense SaaS.

La sentencia **CONNECT** se utiliza para definir el acceso de Qlik Sense a una base de datos general mediante la interfaz OLE DB/ODBC. Para ODBC, primero se debe especificar la fuente de datos utilizando el administrador ODBC.

```
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
```

Declare

La sentencia **Declare** se utiliza para crear definiciones de campos, donde puede definir relaciones entre campos o funciones. Un conjunto de definiciones de campos puede servir para generar automáticamente campos derivados, los cuales se pueden utilizar como dimensiones. Por ejemplo, podemos crear una definición de calendario y utilizarla para generar dimensiones relacionadas, como por ej. año, mes, semana y día, a partir de un campo fecha.

```
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> ]
```

Derive

La sentencia **Derive** se utiliza para generar campos derivados basados en una definición de campo creada con una sentencia **Declare**. Puede especificar para qué campos de datos derivar campos, o bien derivarlos explícita o implícitamente basándose en etiquetas de campos.

```
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

La sentencia **DIRECT QUERY** permite enlazar tablas mediante una conexión ODBC u OLE DB, usando la función Direct Discovery.

```
Direct Query [path]
```

Directory

La sentencia **Directory** define qué directorio buscar en los archivos de datos en sentencias **LOAD** posteriores, hasta que se haga una nueva sentencia **Directory**.

```
Directory [path]
```

Disconnect

La sentencia **Disconnect** pone fin a la conexión ODBC/OLE DB/personalizada. Esta sentencia es opcional.

```
Disconnect
```

drop field

Se pueden eliminar uno o varios campos Qlik Sense del modelo de datos y, por lo tanto, de la memoria, en cualquier momento durante la ejecución del script, mediante el uso de una sentencia **drop field**. La propiedad "distinta" de una tabla se elimina después de una sentencia **drop field**.

*Tanto **drop field** como **drop fields** son formas permitidas sin diferencia alguna en su efecto. Si no se especifica tabla alguna, el campo se eliminará de todas las tablas en las que aparece.*

```
Drop field  fieldname [ , fieldname2 ...] [from tablename1 [ , tablename2
...]]
drop fields fieldname [ , fieldname2 ...] [from tablename1 [ , tablename2
...]]
```

drop table

Se pueden eliminar una o varias tablas internas de Qlik Sense del modelo de datos y por tanto de la memoria, en cualquier momento durante la ejecución del script, mediante una sentencia **drop table**.

*Las formas **drop table** y **drop tables** se aceptan ambas.*

```
Drop table  tablename [, tablename2 ...]
drop tables[ tablename [, tablename2 ...]
```

Execute

La sentencia **Execute** se utiliza para ejecutar otros programas a la vez que Qlik Sense está cargando datos. Por ejemplo, para hacer las conversiones que sean necesarias.

```
Execute commandline
```

FlushLog

La sentencia **FlushLog** obliga a Qlik Sense a escribir el contenido del búfer de script en el archivo de registro de script.

```
FlushLog
```

Force

La sentencia **force** obliga a Qlik Sense a interpretar los nombres de campo y valores de campo de las sentencias **LOAD** y **SELECT** posteriores como escritas solo con letras mayúsculas, solo con letras minúsculas, como suelen ir siempre o tal como aparecen (una combinación de ambas). Esta sentencia permite asociar valores de campo de tablas según distintas convenciones.

```
Force ( capitalization | case upper | case lower | case mixed )
```

LOAD

La sentencia **LOAD** carga campos desde un archivo, desde datos definidos en el script, desde una tabla previamente cargada, desde una página web, desde el resultado de una sentencia **SELECT** posterior, o generando los datos automáticamente. También se puede cargar datos desde conexiones analíticas.

```
 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)]
```

Let

La sentencia **let** es un complemento a la sentencia **set**, utilizada para definir variables de script. La sentencia **let**, al contrario que la sentencia **set**, evalúa la expresión a la derecha del signo "=" en tiempo de ejecución de script antes de que se asigne a la variable.

```
Let variablename=expression
```

Loosen Table

Una o más tablas de datos internas de Qlik Sense se pueden declarar como parcialmente desconectadas durante la ejecución del script mediante el uso de una sentencia **Loosen Table**. Cuando una tabla está parcialmente desconectada, todas las asociaciones entre los valores de campo de la tabla se eliminan. Se puede obtener un efecto similar cargando cada campo de la tabla parcialmente desconectada como tablas independientes, no conectadas. La desconexión parcial puede ser útil durante las pruebas para aislar temporalmente distintas partes de la estructura de datos. Una tabla parcialmente desconectada se identifica en el visor de tablas por las líneas de puntos. El uso de una o más sentencias **Loosen Table** en el script hará que Qlik Sense no tenga en cuenta cualquier configuración de tablas parcialmente desconectadas antes de la ejecución del script.

```
tablename [ , tablename2 ...]
Loosen Tables tablename [ , tablename2 ...]
```

Map ... using

La sentencia **map ... using** se utiliza para asignar un determinado valor de campo o expresión a los valores de una tabla de correspondencia específica. La tabla de correspondencia se crea mediante la sentencia **Mapping**.

```
Map  *fieldlist Using  mapname
```

NullAsNull

La sentencia **NullAsNull** deshabilita la conversión de valores NULL a valores de cadena previamente establecidos por una sentencia **NullAsValue**.

```
NullAsNull *fieldlist
```

NullAsValue

La sentencia **NullAsValue** especifica para qué campos debería convertirse NULL en un valor.

```
NullAsValue *fieldlist
```

Qualify

La sentencia **Qualify** se utiliza para activar la calificación de los nombres de campo, es decir, los nombres de campo obtendrán el nombre de la tabla como un prefijo.

```
Qualify *fieldlist
```

Rem

La sentencia **rem** se utiliza para insertar comentarios u observaciones en el script o para desactivar temporalmente las sentencias del script sin eliminarlas.

```
Rem string
```

Rename Field

Esta función de script renombra uno o varios campos de Qlik Sense tras haberlos cargado.

```
Rename field (using mapname|oldname to newname{ , oldname to newname })
```

```
Rename Fields (using mapname | oldname to newname{ , oldname to newname })
```

Rename Table

Esta función de script renombra una o varias tablas internas de Qlik Sense tras haberlas cargado.

```
Rename table  (using mapname | oldname to newname{ , oldname to newname })
Rename Tables (using mapname | oldname to newname{ , oldname to newname })
```

Section

Con la sentencia **section**, es posible definir si las sentencias **LOAD** y **SELECT** posteriores deberían considerarse como datos o como una definición de los derechos de acceso.

```
Section (access | application)
```

Select

La selección de campos desde una fuente de datos ODBC o proveedor OLE DB se realiza mediante sentencias SQL **SELECT** estándar. No obstante, si las sentencias **SELECT** se aceptan depende del controlador ODBC o proveedor OLE DB utilizado.

```
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 ]
```

Set

La sentencia **set** se utiliza para definir variables de script. Éstas pueden servir para sustituir cadenas, rutas, unidades de disco, etc.

```
Set variablename=string
```

Sleep

La sentencia **sleep** pausa la ejecución del script por un tiempo especificado.

```
Sleep n
```

SQL

La sentencia **SQL** permite enviar un comando arbitrario SQL a través de una conexión ODBC u OLE DB.

```
SQL sql_command
```

SQLColumns

La sentencia **sqlcolumns** devuelve un conjunto de campos que describen las columnas de una fuente de datos ODBC o OLE DB, a las que se ha realizado un **connect**.

```
SQLColumns
```

SQLTables

La sentencia **sqltables** devuelve un conjunto de campos que describen las tablas de una fuente de datos ODBC o OLE DB, a las que se ha realizado un **connect**.

```
SQLTables
```

SQLTypes

La sentencia **sqltypes** devuelve un conjunto de campos que describen los tipos de una fuente de datos ODBC o OLE DB, a los que se ha realizado un **connect**.

```
SQLTypes
```

Star

La cadena utilizada para representar el conjunto de todos los valores de un campo en la base de datos se puede establecer mediante la sentencia **star**. Afecta a las sentencias **LOAD** y **SELECT** subsiguientes.

```
Star is [ string ]
```

Store

La sentencia **Store** crea un archivo QVD, Parquet, CSV o TXT.

```
Store [ *fieldlist from] table into filename [ format-spec ];
```

Tag
Esta sentencia de script proporciona una forma de asignar etiquetas a uno o más campos o tablas. Si se intenta etiquetar un campo o una tabla que no está presente en la app, se ignorará el etiquetado. Si hubiera múltiples nombres de un mismo campo o etiqueta, se empleará el último valor.

```
Tag[field|fields] fieldlist with tagname
Tag [field|fields] fieldlist using mapname
Tag table tablelist with tagname
```

Trace
La sentencia **trace** escribe una cadena en la ventana de **Progreso de ejecución de script** y en el archivo de registro de script, cuando se utiliza. Es muy útil cuando se desea efectuar una depuración. Utilizando expansiones $ de variables que se calculan antes de la sentencia **trace**, puede personalizar el mensaje.

```
Trace string
```

Unmap
La sentencia **Unmap** desactiva la asignación de valores de campo especificada por una sentencia **Map ... Using** anterior para los campos cargados posteriormente.

```
Unmap *fieldlist
```

Unqualify
La sentencia **Unqualify** se utiliza para desactivar la calificación de los nombres de campo que la sentencia **Qualify** activó previamente.

```
Unqualify *fieldlist
```

Untag
Esta sentencia de script proporciona una manera de eliminar etiquetas de campos o tablas. Si se intenta eliminar la etiqueta de un campo o una tabla que no está presente en la app, se ignorará la eliminación de la etiqueta.

```
Untag[field|fields] fieldlist with tagname
Tag [field|fields] fieldlist using mapname
Tag table tablelist with tagname
```

Alias

La sentencia **alias** se utiliza para establecer un alias según el cual un campo se renombrará cada vez que aparezca en adelante en el script.

Sintaxis:

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

Argumentos:

Argumentos

Argumento	Descripción
fieldname	El nombre del campo en sus datos fuente
aliasname	Un nombre alias que desee utilizar en vez

Ejemplos y resultados:

Ejemplo	Resultado
`Alias ID_N as NameID;`	
`Alias A as Name, B as Number, C as Date;`	Los cambios de nombre definidos mediante esta sentencia se utilizan en todas las sentencias **SELECT** y **LOAD** posteriores. Se puede definir un nuevo alias para un nombre de campo mediante una nueva sentencia **alias** en cualquier posición posterior en el script.

AutoNumber

Esta sentencia crea un valor entero único por cada valor distinto evaluado en un campo hallado durante la ejecución de script.

También puede usar la función de *autonumber (page 591)* dentro de una sentencia **LOAD**, aunque se presentarán algunas limitaciones cuando quiera usar una carga optimizada. Puede crear una carga optimizada si carga los datos desde un archivo **QVD** primero y después usa la sentencia **AutoNumber** para convertir los valores en claves de símbolo.

Sintaxis:

```
AutoNumber *fieldlist [Using namespace] ]
```

Argumentos:

Argumentos

Argumento	Descripción
*fieldlist	Una lista separada por comas de los campos en los que los valores se deben reemplazar por un valor entero único. Puede usar los caracteres comodín ? y * en los nombres de campos para incluir todos los campos que tengan campos coincidentes. También puede usar * para incluir todos los campos. Tiene que entrecomillar los nombres de campos cuando se usen caracteres comodín.
namespace	**Using** `namespace` es opcional. Puede usar esta opción si quiere crear un espacio de nombres donde los valores idénticos de campos distintos compartan la misma clave. Si no usa esta opción, todos los campos tendrán un índice de clave independiente.

Limitaciones:

Cuando tenga varias sentencias **LOAD** en el script, tendrá que colocar la sentencia **AutoNumber** tras la sentencia **LOAD** final.

Ejemplo: script con AutoNumber

Ejemplo de script

En este ejemplo, los datos se cargan primero sin la sentencia **AutoNumber**. A continuación, se agrega la instrucción **AutoNumber** para mostrar el efecto.

Datos utilizados en el ejemplo

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear el ejemplo de script a continuación. Deje la sentencia **AutoNumber** sin comentar por ahora.

```
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;
```

Crear visualizaciones

Cree dos visualizaciones de tabla en una hoja de Qlik Sense. Agregue **KeyToOtherTable**, **Región**, **Año**, **Mes** y **Ventas** como dimensiones para la primera tabla. Agregue **KeyToOtherTable**, **Región**, **Año**, **Mes** y **Ventas** como dimensiones para la segunda tabla.

Resultado

Tabla RegionSales

KeyToOtherTable	Region	Year	Month	Sales
North|2014|June	North	2014	June	127
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

Tabla 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

Explicación

El ejemplo muestra un campo compuesto **KeyToOtherTable** que vincula las dos tablas. **AutoNumber** no se utiliza. Tenga en cuenta la longitud de los valores **KeyToOtherTable**.

Agregar la sentencia AutoNumber

Elimine el comentario de la sentencia **AutoNumber** en el script de carga:

```
AutoNumber KeyToOtherTable;
```

Resultado

Tabla 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

KeyToOtherTable	Region	Year	Month	Sales
4	South	2013	May	367
4	South	2014	June	645

Tabla 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

Explicación

Los valores del campo **KeyToOtherTable** se han reemplazado por valores enteros únicos y, como resultado, se ha reducido la longitud de los valores de campo, conservando así la memoria. Los campos clave en ambas tablas se ven afectados por **AutoNumber** y las tablas permanecen vinculadas. El ejemplo es breve y con fines de demostración, pero tendría todo el sentido en una tabla con un número elevado de filas.

Binary

La sentencia **binary** se utiliza para cargar los datos de otra app de Qlik Sense o documento QlikView, incluidos los datos de la sección de acceso. Otros elementos de la app no se incluyen, por ejemplo, hojas, historias, visualizaciones, elementos maestros o variables.

Solo se permite una sentencia **binary** en el script. La sentencia **binary** debe ser la primera sentencia del script, incluso antes de las sentencias SET normalmente ubicadas al inicio del script.

Sintaxis:

```
binary [path] filename
```

Argumentos:

Argumentos

Argumento	Descripción
path	La ruta al archivo que debería remitir a una conexión de datos de carpetas. Esto es necesario si el archivo no está situado en el directorio de trabajo de Qlik Sense. **Ejemplo: *'lib://Table Files/'*** En el modo de elaboración de scripts de legado, se admiten también los diguientes formatos de ruta: • Absoluta **Ejemplo: *c:**data*** • relativa a la app que contiene esta línea de script. **Ejemplo: *data***
filename	El nombre del archivo, incluida la extensión de archivo .qvw o .qvf.

Limitaciones:

No puede usar **binary** para cargar datos desde una app en la misma instalación de Qlik Sense Enterprise refiriéndose al ID de la app. Solo puede cargar desde un archivo *.qvf*.

Ejemplos

Cadena	Descripción
`Binary lib://DataFolder/customer.qvw;`	En este ejemplo, el archivo debe estar ubicado en la conexión de datos de la **Carpeta**. Esta puede ser, por ejemplo, una carpeta que su administrador haya creado en el servidor de Qlik Sense. Haga clic en **Crear nueva conexión** en el editor de carga de datos y luego seleccione **Carpeta** en **Ubicaciones de archivo**.
`Binary customer.qvf;`	En este ejemplo, el archivo debe estar ubicado en el directorio de trabajo de Qlik Sense.
`Binary c:\qv\customer.qvw;`	Este ejemplo que utiliza una ruta de archivo absoluta sólo funcionará en el modo de script de legado.

Comment field

Ofrece una forma de mostrar los comentarios del campo (metadatos) desde bases de datos y hojas de cálculo. Los nombres de campo que no estén presentes en la app se ignorarán. Si hubiera múltiples nombres de un mismo campo, se empleará el último valor.

Sintaxis:

```
comment [fields] *fieldlist using mapname
```

```
comment [field] fieldname with comment
```

La tabla de enlace empleada deberá tener dos columnas, la primera con los nombres de campo y la segunda con los comentarios.

Argumentos:

Argumentos

Argumento	Descripción
**fieldlist*	Es una lista separada por comas con los campos que se han de comentar. Usar * como campo indica la totalidad de campos. Se permiten los caracteres comodín * y ? en nombres de campo. Puede que sea necesario entrecomillar los nombres de campo cuando se empleen caracteres comodín.
mapname	El nombre de una tabla de correspondencia previamente leída en una sentencia mapping **LOAD** o mapping **SELECT**.
fieldname	El nombre del campo que debería comentarse.
comment	El comentario que deberá añadirse al campo.

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

Ofrece una forma de mostrar los comentarios de una tabla (metadatos) desde bases de datos u hojas de cálculo.

Los nombres de tabla que no estén presentes en la app se ignorarán. Si hubiera múltiples nombres de una misma tabla, se empleará el último valor. La palabra clave se puede utilizar para leer los comentarios desde una fuente de datos.

Sintaxis:

```
comment [tables] tablelist using mapname
comment [table] tablename with comment
```

Argumentos:

Argumentos

Argumento	Descripción
tablelist	(table{,table})
mapname	El nombre de una tabla de correspondencia previamente leída en una sentencia mapping **LOAD** o mapping **SELECT**.
tablename	El nombre de la tabla que debería comentarse.
comment	El comentario que deberá añadirse a la tabla.

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

La sentencia **CONNECT** se utiliza para definir el acceso de Qlik Sense a una base de datos general mediante la interfaz OLE DB/ODBC. Para ODBC, primero se debe especificar la fuente de datos utilizando el administrador ODBC.

Esta funcionalidad no está disponible en Qlik Sense SaaS.

Esta sentencia admite únicamente conexiones de datos de carpetas en modo estándar.

Sintaxis:

```
ODBC CONNECT TO connect-string
OLEDB CONNECT TO connect-string
CUSTOM CONNECT TO connect-string
LIB CONNECT TO connection
```

Argumentos:

Argumentos

Argumento	Descripción

connect-string	`connect-string ::= datasourcename { ; conn-spec-item }`
	La cadena de conexión es el nombre de la fuente de datos y una lista opcional de uno o más elementos de especificación de conexión. Si el nombre de la fuente de datos contienen espacios en blanco, o si se lista cualquiera de los elementos de especificación de la conexión, la cadena de conexión deberá ir entre comillas.
	
	datasourcename debe ser un origen de datos ODBC definido o una cadena que define un proveedor de OLE DB.
	
	`conn-spec-item ::=`**`DBQ=`**`database_specifier |`**`DriverID=`**`driver_specifier |`**`UID=`**`userid |`**`PWD=`**`password`
	
	Los elementos posibles de especificación de conexión pueden variar según las diferentes bases de datos. En algunas bases de datos, es posible que haya otros elementos más aparte de los señalados arriba. Para OLE DB, algunos de los elementos específicos de conexión son obligatorios y no opcionales.

connection	El nombre de una conexión de datos almacenada en el editor de carga de datos.

Si la conexión **ODBC** se coloca antes de **CONNECT**, se usará la interfaz ODBC; si no, se usará OLE DB.

Usar **LIB CONNECT TO** conecta con una base de datos utilizando una conexión de datos almacenada que se creó en el editor de carga de datos.

Example 1:

```
ODBC CONNECT TO 'Sales
DBQ=C:\Program Files\Access\Samples\Sales.mdb';
```

La fuente de datos definida a través de esta sentencia es utilizada por sentencias **Select (SQL)** subsiguientes, hasta que se hace una nueva sentencia **CONNECT**.

Example 2:

```
LIB CONNECT TO 'DataConnection';
```

Connect32

Esta sentencia se usa de la misma manera que la sentencia **CONNECT**, pero obliga al uso de un proveedor ODBC/OLE DB de 32 bits. No aplicable a connect personalizada.

Connect64

Esta sentencia se usa de la misma manera que la sentencia **CONNECT**, pero obliga al uso de un proveedor de 64 bits. No aplicable a connect personalizada.

Declare

La sentencia **Declare** se utiliza para crear definiciones de campos, donde puede definir relaciones entre campos o funciones. Un conjunto de definiciones de campos puede servir para generar automáticamente campos derivados, los cuales se pueden utilizar como dimensiones. Por ejemplo, podemos crear una definición de calendario y utilizarla para generar dimensiones relacionadas, como por ej. año, mes, semana y día, a partir de un campo fecha.

Puede usar **Declare** para configurar una nueva definición de campo o para crear una definición de campo basada en una definición ya existente.

Configurar una nueva definición de campo

Sintaxis:

```
definition_name:

Declare [Field[s]] Definition [Tagged tag_list ]

[Parameters parameter_list ]

Fields field_list
```

Argumentos:

Argumento	Descripción
definition_name	El nombre de la definición de campo, acabado en dos puntos. *No utilice autoCalendar como nombre para las definiciones de campo, ya que este nombre se reserva para las plantillas de calendario generadas automáticamente.* **Ejemplo:** `Calendar:`
tag_list	Una lista de etiquetas separadas por comas para aplicar a campos derivados de la definición de campo. La aplicación de etiquetas es opcional, pero si no aplica etiquetas que se usen para especificar un criterio de ordenación, como $date, $numeric o $text, el campo derivado se ordenará por orden de carga como criterio predeterminado. **Ejemplo:** `'$date'Thank you for bringing this to our attention, and apologies for the inconvenience.`
parameter_list	Una lista de parámetros separados por comas. Un parámetro se define como `name=value` y se le asigna un valor de inicio, el cual puede ignorarse cuando se reutiliza una definición de campo. Opcional. **Ejemplo:** `first_month_of_year = 1`
field_list	Una lista de campos separados por comas para generar cuando se utilice la definición de campo. Un campo se define en forma de `<expression>` **`As`** `field_name` **`tagged`** `tag`. Utilice `$1` para hacer referencia al campo de datos desde el que se deben generar los campos derivados. **Ejemplo:** `Year($1) As Year tagged ('$numeric')`

Ejemplo:

```
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')
;
```

El calendario ya está definido y puede aplicarlo a los campos de fecha que se han cargado, en este caso OrderDate y ShippingDate, utilizando una cláusula **Derive**.

Reutilizar una definición de campo previa

Sintaxis:

```
<definition name>:

Declare [Field][s] Definition

Using <existing_definition>

[With <parameter_assignment> ]
```

Argumentos:

Argumento	Descripción
definition_name	El nombre de la definición de campo, acabado en dos puntos. **Ejemplo:** `MyCalendar:`
existing_definition	La definición de campo para reutilizar cuando se cree la nueva definición de campo. La nueva definición de campo funcionará de la misma manera que la definición en la que se basa, con la excepción de que si usa parameter_assignment se usará para cambiar un valor empleado en las expresiones de campo. **Ejemplo:** `Using Calendar`
parameter_assignment	Una lista de asignaciones de parámetros separadas por comas. Una asignación de parámetro se define como `name=value` e ignora el valor de parámetro establecido en la definición del campo base. Opcional. **Ejemplo:** `first_month_of_year = 4`

Ejemplo:

En este ejemplo reutilizamos la definición de calendario que se creó en el ejemplo anterior. En este caso deseamos usar un año fiscal que comience en abril. Esto se logra asignando el valor 4 al parámetro first_ month_of_year, lo que afectará al campo DayNumberOfYear que se define.

El ejemplo asume que utilizamos los datos de muestra y la definición de campo del ejemplo anterior.

```
MyCalendar:
DECLARE FIELD DEFINITION USING Calendar WITH first_month_of_year=4;



DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING MyCalendar;
```

Cuando haya recargado el script de datos, los campos generados estarán disponibles en el editor de hojas, con los nombres OrderDate.MyCalendar.* y ShippingDate.MyCalendar.*.

Derive

La sentencia **Derive** se utiliza para generar campos derivados basados en una definición de campo creada con una sentencia **Declare**. Puede especificar para qué campos de datos derivar campos, o bien derivarlos explícita o implícitamente basándose en etiquetas de campos.

Sintaxis:

```
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
```

Argumentos:

Argumentos

Argumento	Descripción
definition	Nombre de la definición de campo que utilizar cuando derivamos campos. **Ejemplo:** `Calendar`
field_list	Una lista de campos de datos separados por comas desde la que deberían generarse los campos derivados, basado en la definición de campo. Los campos de datos deben ser campos que ya hayamos cargado en el script. **Ejemplo:** `OrderDate, ShippingDate`

Argumento	Descripción
tag_list	Una lista de etiquetas separadas por comas. Se generarán campos derivados para todos los campos de datos con cualquiera de las etiquetas listadas. La lista de etiquetas debe estar entre corchetes. **Ejemplo:** `('$date','$timestamp')`

Ejemplos:

- Derivar campos para campos de datos específicos.
  En este caso especificamos los campos OrderDate y ShippingDate.
  `DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING Calendar;`
- Derivar campos para todos los campos con una etiqueta específica.
  En este caso derivamos campos basados en Calendar para todos los campos con una etiqueta `$date`.
  `DERIVE FIELDS FROM EXPLICIT TAGS ('$date') USING Calendar;`
- Derivar campos para todos los campos con una etiqueta de definición de campo.
  En este caso derivamos campos para todos los campos de datos con la misma etiqueta de la definición de campo Calendar, que en este caso es `$date`.
  `DERIVE FIELDS FROM IMPLICIT TAG USING Calendar;`

Direct Query

La sentencia **DIRECT QUERY** permite enlazar tablas mediante una conexión ODBC u OLE DB, usando la función Direct Discovery.

Sintaxis:

```
DIRECT QUERY DIMENSION fieldlist [MEASURE fieldlist] [DETAIL fieldlist] FROM
tablelist
[WHERE where_clause]
```

Las palabras clave **DIMENSION**, **MEASURE** y **DETAIL** pueden utilizarse en cualquier orden.

Las palabras clave **DIMENSION** y **FROM** son obligatorias en todas las sentencias **DIRECT QUERY**. La palabra clave **FROM** debe aparecer después de la palabra clave **DIMENSION**.

Los campos especificados directamente tras la palabra clave **DIMENSION** se cargan en la memoria y pueden utilizarse para crear asociaciones entre los datos en memoria y los datos de Direct Discovery.

*La sentencia **DIRECT QUERY** no puede contener cláusulas **DISTINCT** o **GROUP BY**.*

Utilizando la palabra clave **MEASURE** puede definir campos de los que Qlik Sense es consciente a un “nivel meta”. Los datos reales de un campo de medida residen solo en la base de datos durante el proceso de carga de datos, y se recuperan conforme a las expresiones de gráfico que se utilicen en una visualización.

Normalmente, los campos con valores discretos que se emplearán como dimensiones deberían cargarse con la palabra clave **DIMENSION**, mientras que los datos numéricos que se utilizarán solo en las agregaciones, deberían seleccionarse con la palabra clave **MEASURE**.

Los campos **DETAIL** ofrecen información o detalles, como campos de comentario, que un usuario puede querer mostrar en un cuadro de tabla que descienda a los detalles. Los campos **DETAIL** no pueden utilizarse en expresiones de gráfico.

Por diseño, la sentencia **DIRECT QUERY** es neutra en cuanto a la fuente de datos para fuentes de datos que admiten SQL. Por esa razón, la misma sentencia **DIRECT QUERY** puede utilizarse en distintas bases de datos SQL sin cambiar. Direct Discovery genera consultas adecuadas para la base de datos según se necesiten.

Se puede emplear la sintaxis nativa de la fuente de datos cuando el usuario conozca la base de datos que desea consultar y desee explotar extensiones específicas de la base de datos en SQL. La sintaxis nativa que admite la fuente de datos:

- Como expresiones de campos en cláusulas **DIMENSION** y **MEASURE**
- Como el contenido de la cláusula **WHERE**

Ejemplos:

```
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"')
```

Los siguientes términos se utilizan como palabras clave y por lo tanto no pueden utilizarse como nombres de columnas o campos sin ir entre comillas: and, as, detach, detail, dimension, distinct, from, in, is, like, measure, native, not, or, where

Argumentos:

Argumento	Descripción
fieldlist	Una lista separada por comas de especificaciones de campo, *fieldname {, fieldname}*. Una especificación de campo puede ser un nombre de campo, en cuyo caso se utiliza el mismo nombre para el nombre de la columna de la base de datos y el nombre del campo Qlik Sense. O una especificación de campo puede ser un "campo alias," en cuyo caso se da un nombre de campo Qlik Sense a una expresión de base de datos o nombre de columna.
tablelist	es el nombre de una tabla o vista de la base de datos desde la que se cargan los datos. Por lo general se trata de vistas que contienen un JOIN efectuado en la base de datos.

Argumento	Descripción

where_clause	La sintaxis completa de las cláusulas **WHERE** de bases de datos no se define aquí, pero se permiten la mayoría de "expresiones relacionales" de SQL, incluido el uso de llamadas a funciones, el operador **LIKE** de cadenas, **IS NULL** e **IS NOT NULL**, e **IN. BETWEEN** no se incluye.
	
	NOT es un operador unario, a diferencia de un modificador o determinadas palabras clave.
	
	Ejemplos:
	
	`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)`
	Este último ejemplo no puede escribirse como:
	
	`WHERE X NOT in (1,2,3)`

Ejemplo:

En este ejemplo se usa tabla de base de datos denominada TableName, que contiene los campos Dim1, Dim2, Num1, Num2 y Num3.Dim1 y Dim2 se cargarán en el conjunto de datos de Qlik Sense.

```
DIRECT QUERY DIMENSTION Dim1, Dim2 MEASURE Num1, Num2, Num3 FROM TableName ;
```

Dim1 y Dim2 estarán disponibles para su uso como dimensiones. Num1, Num2 y Num3 estarán disponibles para agregaciones. Dim1 y Dim2 estarán también disponibles para agregaciones. El tipo de agregaciones para el que se pueden utilizar Dim1 y Dim2 depende de sus tipos de datos. Por ejemplo, en muchos casos los campos **DIMENSION** contienen datos de cadenas como nombres o números de cuenta. Dichos campos no pueden sumarse pero sí contarse: count(Dim1): `count(Dim1)`.

*Las sentencias **DIRECT QUERY** se escriben directamente en el editor de script. Para simplificar la construcción de sentencias **DIRECT QUERY**, podemos generar una sentencia **SELECT** desde una conexión de datos, y a continuación editar el script generado para cambiarlo a una sentencia **DIRECT QUERY**.*
*Por ejemplo, la sentencia **SELECT**:*

```
SQL SELECT
 SalesOrderID,
 RevisionNumber,
 OrderDate,
 SubTotal,
 TaxAmt
FROM MyDB.Sales.SalesOrderHeader;
```

*podría cambiarse por la siguiente sentencia **DIRECT QUERY**:*

```
DIRECT QUERY
 DIMENSION
 SalesOrderID,
 RevisionNumber

MEASURE
 SubTotal,
 TaxAmt

DETAIL
OrderDate

FROM MyDB.Sales.SalesOrderHeader;
```

Listas de campos en Direct Discovery

Una lista de campos es una lista separada por comas de especificaciones de campos, *fieldname {, fieldname}*. Una especificación de campo puede ser un nombre de campo, en cuyo caso se utiliza el mismo nombre para el nombre de la columna de la base de datos y el nombre del campo. O una especificación de campo puede ser un "campo alias," en cuyo caso se da un nombre de campo Qlik Sense a una expresión de base de datos o nombre de columna.

Los nombres de campo pueden ser o nombres simples o nombres entrecomillados. Un nombre simple comienza con un carácter alfabético Unicode y va seguido de cualquier combinación de caracteres alfabéticos o numéricos o guiones bajos. Los nombres entrecomillados comienzan con unas comillas dobles y contienen cualquier secuencia de caracteres. Si un nombre entrecomillado contiene comillas dobles, esas comillas dobles se representan empleando dos signos de dobles comillas adyacentes.

Los nombres de campo en Qlik Sense son sensibles a mayúsculas. Los nombres de campo de las bases de datos pueden ser o no sensibles a mayúsculas, dependiendo de la base de datos concreta. Una consulta Direct Discovery preserva el caso de todos los identificadores de campos y alias. En el siguiente ejemplo, el alias "MyState" se utiliza internamente para almacenar los datos de la columna de la base de datos "STATEID".

```
DIRECT QUERY Dimension STATEID as MyState Measure AMOUNT from SALES_TABLE;
```

Esto difiere del resultado de una sentencia **SQL Select** con un alias. Si el alias no se entrecomilla explícitamente, el resultado por defecto es el de una columna devuelta por la base de datos destino. En el ejemplo siguiente, la instrucción **SQL Select** a una base de datos Oracle crea "MYSTATE," con todas las letras mayúsculas, como el alias interno de Qlik Sense, aunque el alias se especifica como un caso mixto. La sentencia **SQL Select** utiliza el nombre de columna que devuelve la base de datos, que en el caso de Oracle es todo en mayúsculas.

```
SQL Select STATEID as MyState, STATENAME from STATE_TABLE;
```

Para evitar este comportamiento, use la sentencia LOAD para especificar el alias.

```
Load STATEID as MyState, STATENAME;
SQL Select STATEID, STATEMENT from STATE_TABLE;
```

En este ejemplo, la columna "STATEID" es almacenada internamente por Qlik Sense como "MyState".

Se permiten la mayoría de expresiones escalares de bases de datos como especificaciones de campos. Las llamadas a funciones también se pueden utilizar en especificaciones de campos. Las expresiones pueden contener constantes que son booleanas, numéricas, o cadenas contenidas en signos de entrecomillado simple (la inclusión de signos de entrecomillado simple viene representada por signos de entrecomillado adyacentes).

Ejemplos:

```
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;
```

Direct Discovery no permite el uso de agregaciones en sentencias **LOAD**. Si se utilizaran agregaciones, los resultados serían impredecibles. No debe utilizarse una sentencia **LOAD** como la siguiente:

```
DIRECT QUERY DIMENSION stateid, SUM(amount*7) AS MultiFirst MEASURE amount FROM sales_table;
```

SUM no debe estar en la sentencia **LOAD**.

Direct Discovery tampoco admite funciones de Qlik Sense en sentencias **Direct Query**. Por ejemplo, la siguiente especificación para un campo **DIMENSION** devuelve un error cuando el campo "Mth" se usa como una dimensión en una visualización:

```
month(ModifiedDate) as Mth
```

Directory

La sentencia **Directory** define qué directorio buscar en los archivos de datos en sentencias **LOAD** posteriores, hasta que se haga una nueva sentencia **Directory**.

Sintaxis:

```
Directory[path]
```

Si la sentencia **Directory** se emite sin un **path** o se deja fuera, Qlik Sense buscará en el directorio de trabajo de Qlik Sense.

Argumentos:

Argumentos

Argumento	Descripción
path	Un texto que puede interpretarse como la ruta al archivo data. La ruta es la ruta del archivo, que puede ser: • Absoluta **Ejemplo: *c:\data*** • relativa al directorio de trabajo de la app Qlik Sense. **Ejemplo: *data*** • Dirección URL (HTTP o FTP), que apunta a una ubicación en Internet o una intranet. **Ejemplo: *http://www.qlik.com***

Ejemplos:

```
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

La sentencia **Disconnect** pone fin a la conexión ODBC/OLE DB/personalizada. Esta sentencia es opcional.

Sintaxis:

```
Disconnect
```

La conexión terminará automáticamente cuando se ejecute una nueva sentencia **connect** o cuando finalice la ejecución del script.

Ejemplo:

```
Disconnect;
```

Drop

La palabra clave de script **Drop** se puede utilizar para eliminar tablas o campos de la base de datos.

Drop field

Se pueden eliminar uno o varios campos Qlik Sense del modelo de datos y, por lo tanto, de la memoria, en cualquier momento durante la ejecución del script, mediante el uso de una sentencia **drop field**. La propiedad "distinta" de una tabla se elimina después de una sentencia **drop field**.

*Tanto **drop field** como **drop fields** son formas permitidas sin diferencia alguna en su efecto. Si no se especifica tabla alguna, el campo se eliminará de todas las tablas en las que aparece.*

Sintaxis:

```
Drop field fieldname { , fieldname2 ...} [from tablename1 { , tablename2
...}]
Drop fields fieldname { , fieldname2 ...} [from tablename1 { , tablename2
...}]
```

Ejemplos:

```
Drop field A;
Drop fields A,B;
Drop field A from X;
Drop fields A,B from X,Y;
```

Drop table

Se pueden eliminar una o varias tablas internas de Qlik Sense del modelo de datos y por tanto de la memoria, en cualquier momento durante la ejecución del script, mediante una sentencia **drop table**.

Sintaxis:

```
drop table  tablename {, tablename2 ...}
drop tables tablename {, tablename2 ...}
```

*Las formas **drop table** y **drop tables** se aceptan ambas.*

Los elementos siguientes se perderán como resultado de esto:

- Las tabla(s) en sí.
- Todos los campos que no formen parte de las restantes tablas.
- Los valores de campo en los campos, los cuales vienen exclusivamente desde las tablas eliminadas.

Ejemplos y resultados:

Ejemplo	Resultado
`drop table Orders, Salesmen, T456a;`	Esta línea da como resultado que las tres tablas sean eliminadas de la memoria.

`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;`
	Una vez creada la tabla *Tab2*, se prescinde de la tabla *Tab1*.

Drop table

Se pueden eliminar una o varias tablas internas de Qlik Sense del modelo de datos y por tanto de la memoria, en cualquier momento durante la ejecución del script, mediante una sentencia **drop table**.

Sintaxis:

```
drop table  tablename {, tablename2 ...}
drop tables tablename {, tablename2 ...}
```

*Las formas **drop table** y **drop tables** se aceptan ambas.*

Los elementos siguientes se perderán como resultado de esto:

- Las tabla(s) en sí.
- Todos los campos que no formen parte de las restantes tablas.
- Los valores de campo en los campos, los cuales vienen exclusivamente desde las tablas eliminadas.

Ejemplos y resultados:

Ejemplo	Resultado
`drop table Orders, Salesmen, T456a;`	Esta línea da como resultado que las tres tablas sean eliminadas de la memoria.

Ejemplo	Resultado
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;	Una vez creada la tabla *Tab2*, se prescinde de la tabla *Tab1*.

Execute

La sentencia **Execute** se utiliza para ejecutar otros programas a la vez que Qlik Sense está cargando datos. Por ejemplo, para hacer las conversiones que sean necesarias.

Esta funcionalidad no está disponible en Qlik Sense SaaS.

Esta sentencia no se admite en modo estándar.

Sintaxis:

```
execute commandline
```

Argumentos:

Argumentos

Argumento	Descripción
commandline	Es un texto que puede ser interpretado por el sistema operativo como una línea de comando. Puede consignar una ruta de archivo absoluta o una ruta de carpeta lib://.

Si desea usar **Execute** deben darse las siguientes condiciones:

- Debe estar en modo de legado (aplicable a Qlik Sense y Qlik Sense Desktop).
- Necesita configurar OverrideScriptSecurity en 1 en *Settings.ini* (aplicable para Qlik Sense). *Settings.ini* está ubicado en *C:\ProgramData\Qlik\Sense\Engine* y generalmente es un archivo vacío.

*Si configura OverrideScriptSecurity para que habilite **Execute**, cualquier usuario podrá ejecutar archivos en el servidor. Por ejemplo, un usuario podrá adjuntar un archivo ejecutable a una app y luego ejecutar el archivo en el script de carga de datos.*

Haga lo siguiente:

1. Haga una copia de *Settings.ini* y ábralo en un editor de texto.
2. Verifique que el archivo incluye *[Parámetros 7]* en la primera línea.
3. Inserte una nueva línea y escriba *OverrideScriptSecurity=1*.
4. Inserte una línea vacía al final del script.
5. Guarde el archivo.
6. Sustituya *Settings.ini* por su archivo editado.
7. Reinicie Qlik Sense Engine Service (QES).

Si Qlik Sense se está ejecutando como un servicio, algunos comandos puede que no se comporten de la forma esperada.

Ejemplo:

```
Execute C:\Program Files\Office12\Excel.exe;
Execute lib://win\notepad.exe // win is a folder connection referring to c:\windows
```

Field/Fields

Las palabras clave de script **Field** y **Fields** se utilizan en sentencias **Declare**, **Derive**, **Drop**, **Comment**, **Rename** y **Tag/Untag**.

FlushLog

La sentencia **FlushLog** obliga a Qlik Sense a escribir el contenido del búfer de script en el archivo de registro de script.

Sintaxis:

```
FlushLog
```

El contenido del buffer se escribe en el archivo .log de registro. Este comando puede ser útil para depurar, puesto que recibiremos datos que de otro modo se podrían haber perdido en una ejecución fallida de script.

Ejemplo:

```
FlushLog;
```

Force

La sentencia **force** obliga a Qlik Sense a interpretar los nombres de campo y valores de campo de las sentencias **LOAD** y **SELECT** posteriores como escritas solo con letras mayúsculas, solo con letras minúsculas, como suelen ir siempre o tal como aparecen (una combinación de ambas). Esta sentencia permite asociar valores de campo de tablas según distintas convenciones.

Sintaxis:

```
Force ( capitalization | case upper | case lower | case mixed )
```

Si no se especifica nada, se asume force case mixed. La sentencia force es válida hasta que se incluya una nueva sentencia force.

La sentencia **force** no tiene efecto en la sección de acceso: todos los valores de campo cargados son insensibles a mayúsculas y minúsculas.

Ejemplos y resultados

Ejemplo	Resultado

Este ejemplo muestra cómo obligar a poner en mayúscula inicial:
`FORCE Capitalization;`
`Capitalization:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	La tabla **Capitalization** contiene los siguientes valores:
	`Ab`
	`Cd`
	`Ef`
	`Gh`
	Todos los valores están en mayúscula inicial.

Este ejemplo muestra cómo obligar a poner en mayúsculas:
`FORCE Case Upper;`
`CaseUpper:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	La tabla **CaseUpper** contiene los siguientes valores:
	`AB`
	`CD`
	`EF`
	`GH`
	Todos los valores están en mayúsculas.

Ejemplo	Resultado

Este ejemplo muestra cómo obligar a poner en minúsculas:
`FORCE Case Lower;`
`CaseLower:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	La tabla **CaseLower** contiene los siguientes valores:
	`ab`
	`cd`
	`ef`
	`gh`
	Todos los valores están en minúsculas.

Este ejemplo muestra cómo obligar a poner en una combinación de mayúsculas y minúsculas:
`FORCE Case Mixed;`
`CaseMixed:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	La tabla **CaseMixed** contiene los siguientes valores:
	`ab`
	`Cd`
	`eF`
	`GH`
	Todos los valores están tal y como aparecen en el script.

Vea también:

From

La palabra clave de script **From** se utiliza en sentencias **Load** para hacer referencia a un archivo y en sentencias **Select** para hacer referencia a una vista o tabla de base de datos.

Load

La sentencia **LOAD** carga campos desde un archivo, desde datos definidos en el script, desde una tabla previamente cargada, desde una página web, desde el resultado de una sentencia **SELECT** posterior, o generando los datos automáticamente. También se pueden cargar datos desde conexiones analíticas.

Sintaxis:

```
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 ]
```

Argumentos:

Argumentos

Argumento	Descripción
distinct	Puede usar **distinct** como un predicado si solo desea cargar registros únicos. Si hay registros duplicados, se cargará la primera instancia. Si está usando loads precedentes necesita colocar **distinct** en la primera sentencia load, puesto que **distinct** solo afecta a la tabla de destino.

Argumento	Descripción
fieldlist	*fieldlist ::= (* | field{**,** * | field })* Una lista de los campos que se van a cargar. Usar * como una lista de campos indica todos los campos de la tabla. *field ::= (fieldref | expression) [***as** *aliasname]* La definición de campo debe contener siempre una referencia literal a un campo existente, o a una expresión. *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [* ***I*** | ***U*** | ***R*** | ***B*** | **T**]) *fieldname* es un texto que es idéntico a un nombre de campo de la tabla. Tenga en cuenta que el nombre de campo debe ir entre comillas dobles rectas o corchetes si contiene por ejemplo espacios. A veces los nombres de campo no están disponibles de forma explícita. Entonces se usa una nomenclatura diferente: *@fieldnumber* representa el número de campo en un archivo de tabla delimitada. Debe ser un entero positivo precedido por "@". La numeración se hace siempre desde 1 hasta el número de campos. *@startpos*:*endpos* representa las posiciones de inicio y final de un campo en un archivo con registros de longitud fija. Las posiciones deben ser ambas números enteros positivos. Los dos números deben estar precedidos por "@" y separados por dos puntos. La numeración se hace siempre desde 1 hasta el número de posiciones. En el último campo, **n** se usa como posición final. • Si *@startpos*:*endpos* va seguido inmediatamente por los caracteres **I** o **U**, los bytes leídos se interpretarán como un binario firmado (**I**) o un entero no firmado (**U**) (orden de bytes de Intel). El número de posiciones leídas debe ser 1, 2 o 4. • Si *@startpos*:*endpos* va inmediatamente seguido por el carácter **R**, los bytes leídos se interpretarán como un número real binario (IEEE de 32 bits o coma flotante de 64 bits). El número de las posiciones leídas debe ser 4 u 8. • Si *@startpos*:*endpos* va inmediatamente seguido por el carácter **B**, los bytes leídos se interpretarán como números BCD (Binary Coded Decimal) según el estándar COMP-3. Se puede especificar cualquier número de bytes. *expression* puede ser una función numérica o una función de cadena basada en uno o varios campos más de la misma tabla. Para más información, vea la sintaxis de las expresiones. **as** se usa para asignar un nuevo nombre al campo.

Argumento	Descripción
from	**from** se usa si los datos deben cargarse desde un archivo usando una carpeta o una conexión de datos de archivos web. *file ::= [path] filename* **Ejemplo: *'lib://Table Files/'*** Si la ruta se omite, Qlik Sense busca el archivo en el directorio especificado por la sentencia **Directory**. Si no hay sentencia **Directory**, Qlik Sense busca en el directorio de trabajo, *C:\Users\{user}\Documents\Qlik\Sense\Apps*. *En una instalación de servidor de Qlik Sense, el directorio de trabajo se especifica en Qlik Sense Repository Service, por defecto es C:\ProgramData\Qlik\Sense\Apps.* El *filename* puede contener los DOS caracteres comodín estándar (* y ?). Esto provocará la carga de todos los archivos en el directorio especificado. *format-spec ::= (fspec-item { , fspec-item })* La especificación de formato consiste en una lista de varios elementos de caracterización, entre paréntesis. **Modo de script de legado** En el modo de elaboración de scripts de legado, se admiten también los diguientes formatos de ruta: • Absoluta **Ejemplo: *c:\data*** • relativa al directorio de trabajo de la app Qlik Sense. **Ejemplo: *data*** • Dirección URL (HTTP o FTP), que apunta a una ubicación en Internet o una intranet. **Ejemplo: *http://www.qlik.com*** •

Argumento	Descripción
from_field	**from_field** se usa si los datos se deben cargar desde un campo previamente cargado. *fieldassource::=***(***tablename, fieldname****)*** El campo es el nombre del *tablename* y *fieldname* previamente cargados. *format-spec ::= (fspec-item {, fspec-item })* La especificación de formato consiste en una lista con varios elementos de especificación entre paréntesis. Para más información, vea *Elementos de especificación de formato (page 168)*. ***from_field** solo admite comas como delimitador de listas al separar campos en las tablas.*
inline	**inline** se usa si los datos se deben escribir dentro del script y no se cargan desde un archivo. *data ::= [text]* Los datos introducidos mediante una cláusula **inline** deben estar entre comillas dobles o entre corchetes. El texto en su interior se interpreta de la misma manera que el contenido de un archivo. Por lo tanto, cuando inserte una nueva línea en un archivo de texto, también debe hacerlo en el texto de una cláusula **inline**, es decir, pulsando la tecla Intro al escribir la secuencia de script. El número de columnas viene definido por la primera línea. *format-spec ::= (fspec-item {, fspec-item })* La especificación de formato consiste en una lista con varios elementos de especificación entre paréntesis. Para más información, vea *Elementos de especificación de formato (page 168)*.
resident	**resident** se usa si los datos se deben cargar desde una tabla previamente cargada. *table label* es una etiqueta que precede a la sentencia o sentencias **LOAD** o **SELECT** que crearon la tabla original. La etiqueta debe ir seguida de dos puntos al final de la línea.
autogenerate	**autogenerate** se utiliza si los datos deben ser generados automáticamente por Qlik Sense. *size ::= number* *Number* es un número entero que indica la cantidad de registros que se generarán. La lista de campos no debe contener expresiones que requieran datos de una fuente de datos externa o una tabla previamente cargada, a menos que nos refiramos a un único valor de campo de una tabla previamente cargada con la función **Peek**.

Argumento	Descripción
extension	Puede cargar datos desde conexiones analíticas. Necesita usar la cláusula **extension** para llamar a una función definida en el complemento plugin (SSE) de extensión del lado del servidor o evaluar un script. Puede enviar una única tabla al complemento SSE y devuelve una sola tabla de datos. Si el complemento plugin no especifica los nombres de los campos que se devuelven, los campos se denominarán Field1, Field2 y así sucesivamente. `Extension pluginname.functionname( tabledescription );` • Cargar datos utilizando una función en un complemento plugin SSE *tabledescription ::= (table { ,tablefield})* Si no indica campos de tabla, los campos se usarán en orden de carga. • Cargar los datos evaluando un script en un complemento plugin SSE *tabledescription ::= (script, table { ,tablefield})* **Manejo del tipo de datos en la definición del campo de la tabla** Los tipos de datos se detectan automáticamente en las conexiones analíticas. Si los datos no tienen valores numéricos y al menos una cadena de texto no nula, el campo se considera texto. En cualquier otro caso, se considera numérico. Puede forzar el tipo de datos encerrando un nombre de campo en **Cadena()** o **Mixto()**. • **Cadena()** obliga al campo a ser de texto. Si el campo es numérico, se extrae la parte de texto del valor dual, no se realiza ninguna conversión. • **Mixto()** obliga al campo a ser dual. **Cadena()** o **Mixto()** no se pueden usar fuera de las definiciones de campo de la tabla de **extensión** y no puede usar otras funciones Qlik Sense en una definición de campo de tabla. **Más sobre las conexiones analíticas** Debe configurar las conexiones analíticas antes de poder usarlas.
where	**where** es una cláusula utilizada para indicar si un registro debe incluirse en la selección o no. La selección se incluye si *criterion* es True. *criterion* es una expresión lógica
while	**while** es una cláusula utilizada para indicar si un registro debe leerse repetidamente. Se lee el mismo registro siempre y cuando el *criterion* sea True. Para ser útil, una cláusula **while** debe incluir por lo general la función **IterNo( )**. *criterion* es una expresión lógica

Argumento	Descripción
group by	**group by** es una cláusula que sirve para definir sobre qué campos deben agregarse (agruparse) los datos. Los campos de agrupación deberán incluirse de alguna manera en las expresiones cargadas. Ningún otro campo más que los de agrupación deberá emplearse fuera de las funciones de agregación en las expresiones cargadas. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	**order by** es una cláusula utilizada para clasificar los registros de una tabla residente antes de que la sentencia **load** los procese. La tabla residente puede ordenarse por más de un campo en orden ascendente o descendente. La ordenación se hace principalmente por valores numéricos y secundariamente por valor de cotejo nacional. Esta cláusula solo puede utilizarse cuando al fuente de datos es una tabla residente. Los campos de ordenación especifican por qué campos está ordenada la tabla residente. El campo puede especificarse por su nombre o por su número en la tabla residente (el primer número de campo es el 1). orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } *sortorder* es o bien *asc* ascendente o *desc* descendente. Si no se especifica *sortorder*, se asume *asc*. *fieldname*, *path*, *filename* y *aliasname* son cadenas de texto que representan lo que sus respectivos nombres implican. Cualquier campo de la tabla fuente se puede usar como *fieldname*. Sin embargo, los campos creados a través de la cláusula (*aliasname*) están fuera del alcance y no se pueden usar dentro de la misma sentencia **load**.

Si no se proporciona ninguna fuente de datos por medio de una cláusula **from**, **inline**, **resident**, **from_fieldextensión** o una cláusula **autogenerate**, los datos se cargarán a partir del resultado de la sentencia **SELECT** o **LOAD** inmediatamente posterior. Dicha sentencia posterior no debería llevar ningún prefijo.

Ejemplos:

Cargar diferentes formatos de archivo
Cargar un archivo de datos delimitados con las opciones predefinidas:

```
LOAD * from data1.csv;
```

Cargar un archivo de datos delimitado desde una conexión de biblioteca (DataFiles):

```
LOAD * from 'lib://DataFiles/data1.csv';
```

Cargar todos los archivos de datos delimitados desde una conexión de biblioteca (DataFiles):

```
LOAD * from 'lib://DataFiles/*.csv';
```

Cargar un archivo delimitado, especificando la coma como delimitador y con etiquetas incrustadas:

```
LOAD * from 'c:\userfiles\data1.csv' (ansi, txt, delimiter is ',', embedded labels);
```

Cargar un archivo delimitado, especificando el tabulador como delimitador y con etiquetas incrustadas:

```
LOAD * from 'c:\userfiles\data2.txt' (ansi, txt, delimiter is '\t', embedded labels);
```

Cargar un archivo dif con encabezados integrados:

```
LOAD * from file2.dif (ansi, dif, embedded labels);
```

Cargar tres campos desde un archivo de registro de longitud fija sin cabeceras:

```
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);
```

Cargar un archivo QVX, especificando una ruta absoluta:

```
LOAD * from C:\qdssamples\xyz.qvx (qvx);
```

Cargar archivos web
Cargar desde la URL predeterminada configurada en la conexión de datos del archivo web:

```
LOAD * from [lib://MyWebFile];
```

Cargar desde una URL específica e ignorar la URL configurada en la conexión de datos del archivo web:

```
LOAD * from [lib://MyWebFile] (URL is 'http://localhost:8000/foo.bar');
```

Cargar desde una URL específica configurada en una variable usando la expansión de signo de dólar:

```
SET dynamicURL = 'http://localhost/foo.bar';

LOAD * from [lib://MyWebFile] (URL is '$(dynamicURL)');
```

Seleccionar ciertos campos, renombrar y calcular campos
Cargar solo tres campos específicos desde un archivo delimitado:

```
LOAD FirstName, LastName, Number from data1.csv;
```

Renombrar el primer campo como A y el segundo campo como B cuando se carga un archivo sin etiquetas:

```
LOAD @1 as A, @2 as B from data3.txt (ansi, txt, delimiter is '\t', no labels);
```

Cargar Name como una concatenación de FirstName, un carácter de espacio y LastName:

```
LOAD FirstName&' '&LastName as Name from data1.csv;
```

Cargar Quantity, Price y Value (el producto de Quantity y Price):

```
LOAD Quantity, Price, Quantity*Price as Value from data1.csv;
```

Seleccionar ciertos registros
Cargar solo registros únicos, los registros duplicados se descartarán:

```
LOAD distinct FirstName, LastName, Number from data1.csv;
```

Cargar solo registros donde el campo Litres tenga un valor superior a cero:

```
LOAD * from Consumption.csv where Litres>0;
```

Cargue datos que no residan en un archivo y datos autogenerados.
Cargar una tabla con datos inline, dos campos denominados CatID y Category:

```
LOAD * Inline

[CatID, Category

0,Regular

1,Occasional

2,Permanent];
```

Cargar una tabla con datos inline, tres campos denominados UserID, Password y Access:

```
LOAD * Inline [UserID, Password, Access

A, ABC456, User

B, VIP789, Admin];
```

Cargar una tabla con 10.000 filas. El campo A contendrá el número del registro leído (1,2,3,4,5 ...) y el campo B contendrá un número aleatorio entre 0 y 1:

```
LOAD RecNo( ) as A, rand( ) as B autogenerate(10000);
```

El paréntesis tras autogenerate se permite pero no es obligatorio.

Cargar datos de una tabla previamente cargada
Primero cargamos un archivo de tabla delimitado y lo denominamos tab1:

```
tab1:

SELECT A,B,C,D from 'lib://DataFiles/data1.csv';
```

Cargar campos de la tabla ya cargada tab1 como tab2:

```
tab2:

LOAD A,B,month(C),A*B+D as E resident tab1;
```

Cargar campos de la tabla ya cargada tab1 pero solo los registros donde A sea mayor que B:

```
tab3:
```

```
LOAD A,A+B+C resident tab1 where A>B;
```

Cargar campos de la tabla ya cargada tab1 ordenados por A:

```
LOAD A,B*C as E resident tab1 order by A;
```

Cargar campos de la tabla ya cargada tab1, ordenados por el primer campo, luego el segundo campo:

```
LOAD A,B*C as E resident tab1 order by 1,2;
```

Cargar campos de la tabla ya cargada tab1 ordenados por C descendente, luego B en orden ascendente y luego el primer campo por orden descendente:

```
LOAD A,B*C as E resident tab1 order by C desc, B asc, 1 desc;
```

Cargar datos de archivos previamente cargados
Cargue el campo Types de la tabla previamente cargada Characters como A:

```
LOAD A from_field (Characters, Types);
```

Cargar datos de una tabla subsiguiente (load precedente)
Cargar A, B y los campos calculados X y Y desde Table1 que se carga en la sentencia **SELECT** subsiguiente:

```
LOAD A, B, if(C>0,'positive','negative') as X, weekday(D) as Y;

SELECT A,B,C,D from Table1;
```

Agrupar datos
Cargar campos agrupados (agregados) por ArtNo:

```
LOAD ArtNo, round(Sum(TransAmount),0.05) as ArtNoTotal from table.csv group by ArtNo;
```

Cargar campos agrupados (agregados) por Week y ArtNo:

```
LOAD Week, ArtNo, round(Avg(TransAmount),0.05) as WeekArtNoAverages from table.csv group by
Week, ArtNo;
```

Leer un registro de forma repetida
En este ejemplo tenemos un archivo de entrada Grades.csv que contiene las calificaciones de cada alumno condensadas en un campo:

```
Student,Grades

Mike,5234

John,3345
```

```
Pete,1234

Paul,3352
```

Las calificaciones, en una escala del 1 al 5, representan las asignaturas Math, English, Science y History. Podemos separar las en valores aparte leyendo cada registro varias veces con una cláusula **while**, usando la función **IterNo( )** como contador. En cada lectura, la calificación se extrae con la función **Mid** y se almacena en Grade, y la asignatura se selecciona usando la función **pick** y se almacena en Subject. La cláusula final **while** contiene la prueba para verificar si se han leído todas las calificaciones (cuatro por alumno en este caso), lo que significa que se debe leer el siguiente registro del alumno.

```
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));
```

El resultado es una tabla que contiene los siguientes datos:

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

Cargar desde conexiones analíticas

Se utilizan los siguientes datos de muestra.

```
Values:
Load
 Rand() as A,
 Rand() as B,
 Rand() as C
AutoGenerate(50);
```

Cargar datos utilizando una función

En estos ejemplos, suponemos que tenemos un complemento, un plugin de conexión analítica llamado *P* que contiene una función personalizada *Calculate(Parameter1, Parameter2)*. La función devuelve la tabla *Resultados* que contiene los campos *Field1* y *Field2*.

```
Load * Extension P.Calculate( Values{A, C} );
```
Cargar todos los campos que se devuelven al enviar los campos A y C a la función.

```
Load Field1 Extension P.Calculate( Values{A, C} );
```
Cargar solo el campo Field1 al enviar los campos A y C a la función.

```
Load * Extension P.Calculate( Values );
```
Cargar todos los campos que se devuelven al enviar los campos A y B a la función. Como los campos no se especifican, A y B se utilizan ya que son los primeros en orden en la tabla.

```
Load * Extension P.Calculate( Values {C, C});
```
Cargar todos los campos que se devuelven al enviar el campo C a ambos parámetros de la función.

```
Load * Extension P.Calculate( Values {String(A), Mixed(B)});
```
Cargar todos los campos que se devuelven al enviar el campo A forzado como una cadena y B forzado como un valor numérico a la función.

Cargar los datos evaluando un script

```
Load A as A_echo, B as B_echo Extension R.ScriptEval( 'q;', Values{A, B} );
```
Cargar la tabla devuelta por el script q al enviar los valores de A y B.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{A, B} );
```
Cargar la tabla devuelta por el script almacenado en la variable My_R_Script al enviar los valores de A y B.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{B as D, *} );
```
Cargar la tabla devuelta por el script almacenado en la variable My_R_Script al enviar los valores de B renombrados como D, A y C. Usar * envía los campos restantes sin referencia.

La extensión de archivo de las conexiones de DataFiles distingue entre mayúsculas y minúsculas. Por ejemplo: .qvd.

Elementos de especificación de formato

Cada elemento de especificación de formato define una determinada propiedad del archivo de tabla:

fspec-item ::= [**ansi** | **oem** | **mac** | **UTF-8** | **Unicode** | **txt** | **fix** | **dif** | **biff** | **ooxml** | **html** | **xml** | **kml** | **qvd** | **qvx** | **parquet** | **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`]

Character set

Character set es un especificador de archivo para la sentencia **LOAD** que define el conjunto de caracteres utilizado en el archivo.

Los especificadores **ansi**, **oem** y **mac** ya se usaron en QlikView y aún funcionan. Sin embargo, no se generarán al crear la sentencia **LOAD** con Qlik Sense.

Sintaxis:

```
utf8 | unicode | ansi | oem | mac | codepage is
```

Argumentos:

Argumentos

Argumento	Descripción
utf8	Juego de caracteres UTF-8
unicode	Juego de caracteres Unicode
ansi	Windows, página de código 1252
oem	DOS, OS/2, AS400 y otros
mac	Página de código 10000
codepage is	Con el especificador **codepage** , se puede utilizar cualquier página de código Windows como *N* .

Limitaciones:

La conversión del conjunto de caracteres **oem** no se ha implementado en macOS. Si no se especifica nada, se asume la página de código 1252 de Windows.

Ejemplo:

```
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)
```

Vea también:

Load (page 158)

Formato de tabla

El formato de la tabla es un especificador de archivo para la sentencia **LOAD** que define el tipo de archivo. Si no se especifica nada, se asume *.txt*.

Tipos de formato de tabla

Tipo	Descripción
txt	En un archivo de texto delimitado, las columnas de la tabla están separadas por un carácter delimitador.
fix	En un archivo de registro de longitud fija, cada campo tiene un número fijo de caracteres. Por lo general, muchos archivos de registro de longitud fija contienen registros separados por un avance de línea, pero hay opciones más avanzadas para especificar el tamaño de registro en bytes o para expandirse por más de una línea con **Record is**. *Si los datos contienen caracteres de múltiples bytes, los saltos de campo podrían alinearse mal porque el formato se basa en una longitud fija en bytes.*
dif	En un archivo *.dif*, (Data Interchange Format) se utiliza un formato especial para definir la tabla.
biff	Qlik Sense también puede interpretar datos en archivos Excel estándar mediante el formato *biff*. (Binary Interchange File Format).
ooxml	Excel 2007 y versiones posteriores utilizan el formato ooxml *.xslx*.
html	Si la tabla es parte de una página o archivo html, debería usarse html.
xml	xml (Extensible Markup Language) es un lenguaje de marcado habitual que se utiliza para representar estructuras de datos en un formato de texto.
qvd	El formato *qvd* es el formato propietario de archivos de QVD, exportado desde una app de Qlik Sense.
qvx	*qvx* es un formato de archivo/stream para resultados de alto rendimiento en Qlik Sense.
parquet	Apache Parquet es un formato de almacenamiento en columnas, altamente eficiente para almacenar y consultar grandes conjuntos de datos.

Delimiter is

Para archivos de tablas delimitados, se puede especificar un delimitador arbitrario mediante el especificador **delimiter is**. Este especificador solo tiene sentido en archivos .txt delimitados.

Sintaxis:

```
delimiter is char
```

Argumentos:

Argumentos

Argumento	Descripción
char	Especifica un solo carácter de los caracteres 127 ASCII.

Se pueden aplicar además los siguientes valores:

Valores opcionales

Valor	Descripción
'\t'	representa un signo de tabulación, con o sin comillas.
'\\'	representa un carácter de barra invertida (\ ).
'spaces'	representa todas las combinaciones de uno o más espacios. Los caracteres no imprimibles con un valor ASCII inferior a 32, con la excepción de CR y LF, se interpretarán como espacios.

Si no se especifica nada, se asume **delimiter is ','**.

Ejemplo:

```
LOAD * from a.txt (utf8, txt, delimiter is ',' , embedded labels);
```

Vea también:

Load (page 158)

No eof

El especificador **no eof** se usa para descartar el carácter de fin de archivo al cargar archivos delimitados **.txt**.

Sintaxis:

```
no eof
```

Si se utiliza el especificador **no eof**, los caracteres con el punto de código 26, que de otro modo denota el final del archivo, se ignoran y pueden ser parte de un valor de campo.

Este especificador solo tiene sentido en archivos delimitados de texto.

Ejemplo:

```
LOAD * from a.txt (txt, utf8, embedded labels, delimiter is ' ', no eof);
```

Vea también:

Load (page 158)

Labels

Labels es un especificador de archivos para la sentencia **LOAD** que define dónde se pueden encontrar los nombres de campos en un archivo.

Sintaxis:

```
embedded labels|explicit labels|no labels
```

Los nombres de campo pueden encontrarse en sitios diferentes en el archivo. Si el primer registro contiene los nombres de los campos, se debe usar **embedded labels**. Si no hay nombres de campo que buscar, se debe usar **no labels**. En archivos *dif*, a veces se usa una sección de encabezado aparte con nombres de campo explícitos. En tal caso, se debe usar **explicit labels**. Si no se especifica nada, se asume **embedded labels**, también para archivos *dif*.

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)
```

Vea también:

Load (page 158)

Header is

Determina el tamaño de la cabecera de los archivos de tabla. Se puede especificar una longitud de cabecera arbitraria mediante el especificador **header is**. Una cabecera es una sección de texto no utilizada por Qlik Sense.

Sintaxis:

```
header is n
```

```
header is line
```

```
header is n lines
```

La longitud del encabezado se puede dar en bytes (**header is n**), o en líneas (**header is line** o **header is n lines**). **n** debe ser un entero positivo, que representa la longitud del encabezado. Si no se especifica, se asume **header is 0**. El especificador **header is** solo es relevante para archivos de tabla.

Ejemplo:

Este es un ejemplo de una tabla de fuente de datos que contiene una línea de texto de cabecera que Qlik Sense no debería interpretar como datos.

```
*Header line
Col1,Col2
```

```
a,B
c,D
```

Usando el especificador **header is 1 lines**, la primera línea no se cargará como datos. En el ejemplo, el especificador **embedded labels** le dice a Qlik Sense que interprete la primera línea no excluida como que contiene etiquetas de campo.

```
LOAD Col1, Col2
FROM 'lib://files/header.txt'
(txt, embedded labels, delimiter is ',', msq, header is 1 lines);
```

El resultado es una tabla con dos campos, Col1 y Col2.

Vea también:

Load (page 158)

Record is

Para archivos de longitud de registro fijo, la longitud de registro debe especificarse mediante el especificador **record is**.

Sintaxis:

```
Record is n
Record is line
Record is n lines
```

Argumentos:

Argumentos

Argumento	Descripción
n	Especifica la longitud de registro en bytes.
line	Especifica la longitud de registro en una línea.
n lines	Especifica la longitud de registro en líneas donde n es un entero positivo que representa la longitud de registro.

Limitaciones:

El especificador **record is** solo es aplicable a archivos **fix**.

Vea también:

Load (page 158)

Quotes

Quotes es un especificador de archivo para la sentencia **LOAD** que define si se pueden usar comillas y la prioridad entre comillas y separadores. Es solo para archivos de texto.

Sintaxis:

```
no quotes
```

```
msq
```

Si se omite el especificador, se usan las comillas estándar, es decir, las comillas " " o ' ' se pueden usar, pero solo si son el primer y último carácter no en blanco de un valor de campo.

Argumentos:

Argumentos

Argumento	Descripción
no quotes	Se utiliza si los símbolos de entrecomillado no se aceptan en un archivo de texto.
msq	Sirve para especificar un estilo de comillas modernas, que permite un contenido de múltiples líneas en los campos. Los campos que contengan caracteres de final de línea deben ir entre comillas dobles. Una limitación de la opción msq es que si un carácter de comillas dobles (") aparece por sí solo como primer o último carácter en el contenido de un campo, se interpretará como el inicio o el final del contenido de múltiples líneas, lo cual puede dar lugar a resultados inesperados en el conjunto de datos cargado. En este caso conviene aplicar las comillas estándar, omitiendo el especificador.

XML

Opciones válidas de Este especificador de script se usa cuando se cargan archivos xml. para el especificador **XML** se enumeran en la sintaxis.

No puede cargar archivos DTD en Qlik Sense.

Sintaxis:

```
xmlsimple
```

Vea también:

Load (page 158)

KML

Este especificador de script se usa cuando se cargan archivos KML para utilizar en una visualización de mapa.

Sintaxis:

```
kml
```

El archivo KML puede representar datos de área (por ejemplo, países o regiones), representados mediante polígonos, o bien datos de puntos (por ejemplo, ciudades o lugares) representados por puntos en forma de [long, lat].

URL is

Este especificador de script se usa para establecer la URL de una conexión de datos de archivos web al cargar un archivo web.

Sintaxis:

```
URL is string
```

Argumentos:

Argumentos

Argumento	Descripción
string	Especifica la URL del archivo que se ha de cargar. Esto anulará la URL establecida en la conexión de archivo web que se esté utilizando.

Limitaciones:

El especificador **URL is** solo es aplicable a archivos web. Necesita usar una conexión existente de datos de archivos web.

Vea también:

Load (page 158)

userAgent is

Este especificador de script se usa para establecer el agente de usuario del navegador al cargar un archivo web.

Sintaxis:

```
userAgent is string
```

Argumentos:

Argumentos

Argumento	Descripción
string	Especifica la cadena del agente de usuario del navegador. Esto anulará el agente de usuario del navegador predeterminado "Mozilla/5.0".

Limitaciones:

El especificador **userAgent is** solo es aplicable a archivos web.

Vea también:

Load (page 158)

Let

La sentencia **let** es un complemento a la sentencia **set**, utilizada para definir variables de script. La sentencia **let**, al contrario que la sentencia **set**, evalúa la expresión a la derecha del signo "=" en tiempo de ejecución de script antes de que se asigne a la variable.

Sintaxis:

```
Let variablename=expression
```

Ejemplos y resultados:

Ejemplo	Resultado

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	`$(x)` se evaluará como ' 3+4 '
	
	`$(y)` se evaluará como ' 7 '
	
	`$(z)` se evaluará como ' 8 '
	
	Observe la diferencia entre las sentencias **Set** y **Let**. La sentencia **Set** asigna la cadena '3+4' a la variable, mientras que la sentencia **Let** evalúa la cadena y asigna 7 a la variable.

Let T=now( );	`$(T)` recibirá el valor de la hora actual.

Loosen Table

Una o más tablas de datos internas de Qlik Sense se pueden declarar como parcialmente desconectadas durante la ejecución del script mediante el uso de una sentencia **Loosen Table**. Cuando una tabla está parcialmente desconectada, todas las asociaciones entre los valores de campo de la tabla se eliminan. Se puede obtener un efecto similar cargando cada campo de la tabla parcialmente desconectada como tablas independientes, no conectadas. La desconexión parcial puede ser útil durante las pruebas para aislar temporalmente distintas partes de la estructura de datos. Una tabla parcialmente desconectada se identifica en el visor de tablas por las líneas de puntos. El uso de una o más sentencias **Loosen Table** en el script hará que Qlik Sense no tenga en cuenta cualquier configuración de tablas parcialmente desconectadas antes de la ejecución del script.

Sintaxis:

```
Loosen Tabletablename [ , tablename2 ...]
```

```
Loosen Tablestablename [ , tablename2 ...]
```

Se puede usar sintaxis: **Loosen Table** o **Loosen Tables**.

*Cuando Qlik Sense encuentra referencias circulares en la estructura de datos que no pueden romperse por tablas declaradas como parcialmente desconectadas de forma interactiva o explícita en el script, se obligará a una o más tablas adicionales a ser parcialmente desconectadas hasta que no queden referencias circulares. Cuando esto ocurra, el cuadro de diálogo **Advertencia Interación** lanza una advertencia.*

Ejemplo:

```
Tab1:

SELECT * from Trans;

Loosen Table Tab1;
```

Map

La sentencia **map ... using** se utiliza para asignar un determinado valor de campo o expresión a los valores de una tabla de correspondencia específica. La tabla de correspondencia se crea mediante la sentencia **Mapping**.

Sintaxis:

```
Map fieldlist Using  mapname
```

La correspondencia o asignación automática (mapeo) se realiza para los campos cargados después de la sentencia **Map ... Using** y hasta el final del script o hasta que encuentra una sentencia **Unmap**.

La asignación (o mapeo) es lo último que se hace en la cadena de eventos, cuando ya falta poco para que el campo se almacene en la tabla interna de Qlik Sense. Esto significa que la correspondencia o mapeo no se realiza cada vez que se encuentra un nombre de campo como parte de una expresión sino más bien cuando el valor se almacena bajo un nombre de campo en la tabla interna. Si se requiere un mapeo en el nivel de expresión, debe usarse la función **Applymap()** en su lugar.

Argumentos:

Argumentos

Argumento	Descripción
fieldlist	Una lista de campos separados por coma, que debe hacerse corresponder desde este punto del script. Usar * como campo indica la totalidad de campos. Se permiten los caracteres comodín * y ? en nombres de campo. Puede que sea necesario entrecomillar los nombres de campo cuando se empleen caracteres comodín.
mapname	El nombre de una tabla de correspondencia previamente leída en una sentencia **mapping load** o **mapping select**.

Ejemplos y resultados:

Ejemplo	Resultado
Map Country Using Cmap;	Habilita el mapeo del campo Country usando Cmap.
Map A, B, C Using X;	Habilita el mapeo del campo A, B y C usando map X.
Map * Using GenMap;	Habilita el mapeo de todos los campos usando GenMap.

NullAsNull

La sentencia **NullAsNull** deshabilita la conversión de valores NULL a valores de cadena previamente establecidos por una sentencia **NullAsValue**.

Sintaxis:

```
NullAsNull *fieldlist
```

La sentencia **NullAsValue** funciona como un conmutador y puede activarse o desactivarse varias veces en el script, ya sea usando un **NullAsValue** o una sentencia **NullAsNull**.

Argumentos:

Argumentos

Argumento	Descripción
*fieldlist	Una lista separada por comas de los campos para los que **NullAsNull** se debe activar. Usar * como campo indica la totalidad de campos. Se permiten los caracteres comodín * y ? en nombres de campo. Puede que sea necesario entrecomillar los nombres de campo cuando se empleen caracteres comodín.

Ejemplo:

```
NullAsNull A,B;
LOAD A,B from x.csv;
```

NullAsValue

La sentencia **NullAsValue** especifica para qué campos debería convertirse NULL en un valor.

Sintaxis:

```
NullAsValue *fieldlist
```

De manera predeterminada, Qlik Sense considera los valores NULL como entidades que faltan o no están definidas. No obstante, determinados contextos de base de datos implican que los valores NULL deben considerarse como valores especiales en lugar de simplemente valores perdidos. El hecho de que normalmente no se permita a los valores NULL enlazar con otros valores NULL puede suspenderse por medio de la sentencia **NullAsValue**.

La sentencia **NullAsValue** funciona como un conmutador y operará en las sentencias posteriores de carga. Se puede deshabilitar de nuevo mediante la sentencia **NullAsNull**.

Argumentos:

Argumentos

Argumento	Descripción
*fieldlist	Una lista separada por comas de los campos para los que **NullAsValue** se debe activar. Usar * como campo indica la totalidad de campos. Se permiten los caracteres comodín * y ? en nombres de campo. Puede que sea necesario entrecomillar los nombres de campo cuando se empleen caracteres comodín.

Ejemplo:

```
NullAsValue A,B;
Set NullValue = 'NULL';
LOAD A,B from x.csv;
```

Qualify

La sentencia **Qualify** se utiliza para activar la calificación de los nombres de campo, es decir, los nombres de campo obtendrán el nombre de la tabla como un prefijo.

Sintaxis:

```
Qualify *fieldlist
```

La unión automática entre campos con el mismo nombre en diferentes tablas se puede suspender mediante la sentencia **qualify**, que califica el nombre del campo con su nombre de tabla. Si están calificados, se renombrarán el/los nombre(s) del campo cuando se encuentre en una tabla. El nuevo nombre tendrá la forma: *tablename.fieldname*. *Tablename* es equivalente a la etiqueta de la tabla actual, o, si no hay ninguna etiqueta, al nombre que aparece después de **from** en sentencias **LOAD** y **SELECT**.

La calificación se realizará para todos los campos cargados después de la sentencia **qualify**.

Por defecto, al iniciar la ejecución de un script, está desactivada la calificación. La calificación de un nombre de campo se puede activar en cualquier momento usando una sentencia **qualify**. La calificación se puede desactivar en cualquier momento usando una sentencia **Unqualify**.

*La sentencia **qualify** no debe usarse junto con una recarga parcial.*

Argumentos:

Argumentos

Argumento	Descripción
*fieldlist	Es un listado de campos separados por comas en los que se ha de aplicar la calificación. Usar * como campo indica la totalidad de campos. Se permiten los caracteres comodín * y ? en nombres de campo. Puede que sea necesario entrecomillar los nombres de campo cuando se empleen caracteres comodín.

Example 1:

```
Qualify B;

LOAD A,B from x.csv;

LOAD A,B from y.csv;
```

Las dos tablas **x.csv** y **y.csv** se asocian únicamente mediante **A**. Tres campos dará como resultado: A, x.B, y.B.

Example 2:

En una base de datos con la que no esté familiarizado, puede que quiera empezar asegurándose de que sólo uno o unos pocos campos estén asociados, como se muestra en el ejemplo:

```
qualify *;

unqualify TransID;

SQL SELECT * from tab1;

SQL SELECT * from tab2;

SQL SELECT * from tab3;
```

Solo se usará **TransID** para asociaciones entre las tablas *tab1*, *tab2* y *tab3*.

Rem

La sentencia **rem** se utiliza para insertar comentarios u observaciones en el script o para desactivar temporalmente las sentencias del script sin eliminarlas.

Sintaxis:

```
Rem string
```

Todo lo que entre **rem** y el siguiente punto y coma **;** se considera un comentario.

Hay disponibles dos métodos alternativos para hacer comentarios en el script:

1. Es posible crear un comentario en cualquier parte del script, excepto entre dos comillas, colocando la sección en cuestión entre **/*** y ***/**.
2. Al escribir **//** en la secuencia de script, todo el texto que sigue a la derecha en la misma fila se convierte en un comentario. (Tenga en cuenta la excepción //: que se puede usar como parte de una dirección de Internet).

Argumentos:

Argumentos

Argumento	Descripción
string	Es un texto cualquiera.

Ejemplo:

```
Rem ** This is a comment **;
/* This is also a comment */
// This is a comment as well
```

Rename

La palabra clave de script **Rename** se puede utilizar para cambiar el nombre de tablas o campos que ya se han cargado.

Rename field

Esta función de script renombra uno o varios campos de Qlik Sense tras haberlos cargado.

> *No se recomienda utilizar el mismo nombre para un campo y una función en Qlik Sense*

Se puede usar sintaxis: **rename field** o **rename fields**.

Sintaxis:

```
Rename Field (using mapname | oldname to newname{ , oldname to newname })
```

```
Rename Fields (using mapname | oldname to newname{ , oldname to newname })
```

Argumentos:

Argumento	Descripción
mapname	El nombre de una tabla de enlace previamente cargada que contiene uno o más pares de tablas antiguas y nuevas.
oldname	El nombre del antiguo campo.
newname	El nombre del nuevo campo.

Limitaciones:

No puede renombrar dos campos de forma que contengan el mismo nombre.

Example 1:

```
Rename Field XAZ0007 to Sales;
```

Example 2:

```
FieldMap:

Mapping SQL SELECT oldnames, newnames from datadictionary;

Rename Fields using FieldMap;
```

Rename table

Esta función de script renombra una o varias tablas internas de Qlik Sense tras haberlas cargado.

Se puede usar sintaxis: **rename table** o **rename tables**.

Sintaxis:

```
Rename Table (using mapname | oldname to newname{ , oldname to newname })
Rename Tables (using mapname | oldname to newname{ , oldname to newname })
```

Argumentos:

Argumentos

Argumento	Descripción
mapname	El nombre de una tabla de enlace previamente cargada que contiene uno o más pares de tablas antiguas y nuevas.
oldname	El nombre de la tabla antigua.
newname	El nombre de la nueva tabla.

Limitaciones:

Dos tablas con distinto nombre no pueden renombrarse con un mismo nombre. El script generará un error si tratamos de renombrar un campo con el mismo nombre de una tabla previa.

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

La sentencia **Search** se utiliza para incluir o excluir campos en la búsqueda inteligente.

Sintaxis:

```
Search Include *fieldlist
Search Exclude *fieldlist
```

Puede usar varias sentencias Search para refinar su selección de campos que incluir. Las sentencias se evalúan de arriba a abajo.

Argumentos:

Argumentos

Argumento	Descripción
*fieldlist	Lista separada por comas de los campos que se deben incluir o excluir de las búsquedas en la búsqueda inteligente. Usar * como campo indica la totalidad de campos. Se permiten los caracteres comodín * y ? en nombres de campo. Puede que sea necesario entrecomillar los nombres de campo cuando se empleen caracteres comodín.

Ejemplo:

Ejemplos de búsqueda

Sentencia	Descripción
`Search Include *;`	Incluya todos los campos en sus búsquedas en la búsqueda inteligente.
`Search Exclude [*ID];`	Excluya todos los campos que terminen en ID de sus búsquedas en la búsqueda inteligente.
`Search Exclude '*ID';`	Excluya todos los campos que terminen en ID de sus búsquedas en la búsqueda inteligente.
`Search Include ProductID;`	Incluya el campo ProductID en sus búsquedas en la búsqueda inteligente.

El resultado combinado de estas tres sentencias, en esta secuencia, es que todos los campos que terminan en ID excepto ProductID se excluyen de las búsquedas en la búsqueda inteligente.

Section

Con la sentencia **section**, es posible definir si las sentencias **LOAD** y **SELECT** posteriores deberían considerarse como datos o como una definición de los derechos de acceso.

Sintaxis:

```
Section (access | application)
```

Si no se especifica nada, se asume **section application**. La definición de **section** es válida hasta que se realiza una nueva sentencia **section**.

Ejemplo:

```
Section access;

Section application;
```

Select

La selección de campos desde una fuente de datos ODBC o proveedor OLE DB se realiza mediante sentencias SQL **SELECT** estándar. No obstante, si las sentencias **SELECT** se aceptan depende del controlador ODBC o proveedor OLE DB utilizado. El uso de la sentencia **SELECT** requiere una conexión de datos abierta a la fuente.

Sintaxis:

```
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 ]
```

A veces se pueden concatenar varias sentencias **SELECT** en una sola mediante el uso de un operador **union**:

```
selectstatement Union selectstatement
```

La sentencia **SELECT** es interpretada por el driver ODBC o proveedor OLE DB, así que podrían producirse desviaciones de la sintaxis SQL general según las capacidades de los drivers ODBC o del proveedor OLE DB, por ejemplo:

- **as** a veces no se permite, es decir, *aliasname* debe ir inmediatamente después de *fieldname*.
- **as** a veces es obligatorio si se usa un *aliasname*.
- **distinct**, **as**, **where**, **group by**, **order by** o **union** a veces no son compatibles.
- El driver ODBC a veces no acepta todos los símbolos de comillas enunciados anteriormente.

*¡Esta no es una descripción completa de la sentencia SQL **SELECT**! Por ej. las sentencias **SELECT** se pueden anidar, se pueden hacer varias uniones en una sentencia **SELECT**, la cantidad de funciones permitidas en las expresiones a veces es muy grande, etc.*

Argumentos:

Argumentos

Argumento	Descripción
distinct	**distinct** se utiliza un predicado si las combinaciones duplicadas de valores en los campos seleccionados solo se deben cargar una vez.
distinctrow	**distinctrow** es un predicado utilizado si los registros duplicados en la tabla de origen solo deberían cargarse una vez.
fieldlist	**`fieldlist ::= (*| field ) {, field }`** Una lista de los campos que se van a seleccionar. Usar * como una lista de campos indica todos los campos de la tabla. **`fieldlist ::= field {, field }`** Una lista de uno o más campos, separados por comas. **`field ::= ( fieldref| expression ) [as aliasname ]`** La expresión puede ser una función numérica o de cadena basada en uno o varios campos. Algunos de los operadores y funciones generalmente aceptados son: +, -, *, /, & (concatenación de cadenas), sum(fieldname), count(fieldname), avg(fieldname)(average), month(fieldname), etc. Consulte la documentación del driver ODBC para más información. **`fieldref ::= [ tablename. ] fieldname`** **tablename** y **fieldname** son cadenas de texto idénticas a lo que implican. Deben estar entre comillas dobles rectas si contienen por ej. espacios. La cláusula **as** se usa para asignar un nuevo nombre al campo.
from	**`tablelist ::= table {, table }`** La lista de las tablas de las que se van a seleccionar los campos. **`table ::= tablename [ [as ] aliasname ]`** **tablename** puede ir o no entre comillas.

Argumento	Descripción
where	**where** es una cláusula utilizada para indicar si un registro debe incluirse en la selección o no. **criterion** es una expresión lógica que a veces puede ser muy compleja. Algunos de los operadores aceptados son: operadores y funciones numéricas, **=**, **<>** o **#(** no igual), **>**, **>=**,**<**, **<=**, **and**, **or**, **not**, **exists**, **some**, **all**, **in** y también nuevas sentencias **SELECT**. Consulte la documentación del driver ODBC o proveedor OLE DB para más información.
group by	**group by** es una cláusula utilizada para agregar (agrupar) varios registros en uno. Dentro de un grupo, para un determinado campo, todos los registros deben tener el mismo valor, o el campo sólo podrá utilizarse desde dentro de una expresión, p.ej. como una suma o una media. La expresión basada en uno o varios campos se define en la expresión del símbolo de campo.
having	**having** es una cláusula que se usa para calificar grupos de una manera similar a como se usa la cláusula **where** para calificar registros.
order by	**order by** es una cláusula que se utiliza para indicar el criterio de ordenación de la tabla resultante de la sentencia **SELECT**.
join	**join** es un cualificador que indica si se deben unir varias tablas en una. Los nombres de campo y los nombres de tabla deben estar entre comillas si contienen espacios en blanco o letras de los juegos de caracteres nacionales. Cuando el script es generado automáticamente por Qlik Sense, el símbolo de comillas utilizado es el preferido del driver ODBC o el proveedor de OLE DB, especificados en la definición de la fuente de datos de la sentencia **Connect**.

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

La sentencia **set** se utiliza para definir variables de script. Éstas pueden servir para sustituir cadenas, rutas, unidades de disco, etc.

Sintaxis:

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Sleep

La sentencia **sleep** pausa la ejecución del script por un tiempo especificado.

Sintaxis:

```
Sleep n
```

Argumentos:

Argumento	Descripción
n	Indicado en milisegundos, donde *n* es un número entero positivo no mayor que 3600000 (es decir, 1 hora). El valor puede ser una expresión.

Example 1:

```
Sleep 10000;
```

Example 2:

```
Sleep t*1000;
```

SQL

La sentencia **SQL** permite enviar un comando arbitrario SQL a través de una conexión ODBC u OLE DB.

Sintaxis:

```
SQL sql_command
```

Enviar sentencias SQL que actualizan la base de datos dará un error si Qlik Sense ha abierto la conexión ODBC en modo de solo lectura.

La sintaxis:

```
SQL SELECT * from tab1;
```
se permite, y es la sintaxis preferida para **SELECT**, por razones de consistencia. El prefijo SQL, no obstante, seguirá siendo opcional para sentencias **SELECT**.

Argumentos:

Argumento	Descripción
sql_command	Un comando SQL válido.

Example 1:

```
SQL leave;
```

Example 2:

```
SQL Execute <storedProc>;
```

SQLColumns

La sentencia **sqlcolumns** devuelve un conjunto de campos que describen las columnas de una fuente de datos ODBC o OLE DB, a las que se ha realizado un **connect**.

Sintaxis:

```
SQLcolumns
```

Los campos se pueden combinar con los campos generados por los comandos **sqltables** y **sqltypes** y para dar una buena visión general de una base de datos determinada. Los doce campos estándar son:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

COLUMN_NAME

DATA_TYPE

TYPE_NAME

PRECISION

LENGTH

SCALE

RADIX

NULLABLE

REMARKS

Para una descripción detallada de estos campos, consulte un manual de referencia de ODBC.

Ejemplo:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLcolumns;
```

Algunos controladores ODBC puede que no admitan este comando. Algunos controladores ODBC pueden producir campos adicionales.

SQLTables

La sentencia **sqltables** devuelve un conjunto de campos que describen las tablas de una fuente de datos ODBC o OLE DB, a las que se ha realizado un **connect**.

Sintaxis:

```
SQLTables
```

Los campos se pueden combinar con los campos generados por los comandos **sqlcolumns** y **sqltypes** y para dar una buena visión general de una base de datos determinada. Los cinco campos estándar son:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

TABLE_TYPE

REMARKS

Para una descripción detallada de estos campos, consulte un manual de referencia de ODBC.

Ejemplo:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTables;
```

Algunos controladores ODBC puede que no admitan este comando. Algunos controladores ODBC pueden producir campos adicionales.

SQLTypes

La sentencia **sqltypes** devuelve un conjunto de campos que describen los tipos de una fuente de datos ODBC o OLE DB, a los que se ha realizado un **connect**.

Sintaxis:

```
SQLTypes
```

Los campos se pueden combinar con los campos generados por los comandos **sqlcolumns** y **sqltables** y para dar una buena visión general de una base de datos determinada. Los quince campos estándar son:

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

Para una descripción detallada de estos campos, consulte un manual de referencia de ODBC.

Ejemplo:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTypes;
```

Algunos controladores ODBC puede que no admitan este comando. Algunos controladores ODBC pueden producir campos adicionales.

Star

La cadena utilizada para representar el conjunto de todos los valores de un campo en la base de datos se puede establecer mediante la sentencia **star**. Afecta a las sentencias **LOAD** y **SELECT** subsiguientes.

Sintaxis:

```
Star is[ string ]
```

Argumentos:

Argumentos

Argumento	Descripción
string	Es un texto cualquiera. Observe que la cadena debe estar entre comillas si contiene espacios en blanco. Si no se especifica nada, se asume **star is;**, es decir, no hay un símbolo de estrella disponible, a menos que se especifique explícitamente. Esta definición es válida hasta que se realice una nueva sentencia **star**.

No se recomienda el uso de la sentencia **Star is** en la parte de datos del script (en la **Sección de aplicación**) si se utiliza sección de acceso. Sin embargo, el carácter de asterisco es totalmente compatible con los campos protegidos en la parte de **Sección de acceso** del script. En este caso, no necesita usar la sentencia explícita **Star is** ya que esta está siempre implícita en la sección de acceso.

Limitaciones

- No puede usar el carácter de asterisco con campos clave; es decir, campos que vinculan tablas.
- No puede usar el carácter de asterisco con ningún campo afectado por la sentencia **Unqualify**, ya que esto puede afectar a los campos que vinculan las tablas.
- No puede usar el carácter de asterisco con tablas no lógicas, por ejemplo, tablas info-load o mapping-load.
- Cuando el carácter de asterisco se usa en un campo reductor (un campo que se vincula a los datos) en la sección de acceso, representa los valores enumerados en este campo en la sección de acceso. No representa otros valores que puedan existir en los datos pero que no se enumeran en la sección de acceso.
- No puede usar el carácter de asterisco con campos afectados por ninguna forma de reducción de datos fuera del área de la **Sección de acceso**.

Ejemplo

El ejemplo inferior es un extracto de un script de carga de datos que contiene sección de acceso.

```
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

];
```

Lo siguiente es de aplicación:

- El signo *Star* es *.
- El usuario *ADMIN* ve todos los campos. Nada se omite.
- El usuario *USER1* no puede ver el campo *SALES*.
- El usuario *USER2* no puede ver el campo *WAREHOUSE* .
- El usuario *USER3* no puede ver el campo *EMPLOYEES*.
- El usuario *USER4* se agrega dos veces a la solución para OMITIR dos campos para este usuario: *SALES* y *WAREHOUSE*.
- El usuario *USER5* tiene un "*" añadido, lo que significa que todos los campos que figuran en OMIT no están disponibles, es decir, el usuario *USER5* no puede ver los campos *SALES*, *WAREHOUSE* y *EMPLOYEES* pero este usuario sí puede ver el campo *ORDERS*.

Store

La sentencia **Store** crea un archivo QVD, Parquet, CSV o TXT.

Sintaxis:

```
Store [ fieldlist from] table into filename [ format-spec ];
```

La sentencia creará un archivo denominado explícitamente QVD, Parquet o archivo de texto.

La sentencia solo puede exportar campos desde una tabla de datos. Si tuviéramos que exportar campos de varias tablas, debemos hacer previamente un join explícito en el script para crear la tabla de datos que se ha de exportar.

Los valores de texto se exportan al archivo CSV en formato UTF-8. Se puede especificar un delimitador, vea **LOAD**. La sentencia **store** a un CSV no admite exportación BIFF.

En algunos casos, con datos que no están bien formados, los campos estarán entre comillas dobles para garantizar que los datos se interpreten correctamente. Esto puede suceder, por ejemplo, cuando el campo contiene caracteres como comillas, comas, espacios o saltos de línea.

Argumentos:

Argumentos de la sentencia Store

Argumento	Descripción
*fieldlist::= (* | field) { , field })*	Una lista de los campos que se van a seleccionar. Usar un asterisco * indica la totalidad de campos. *field::= fieldname [***as** *aliasname]* *fieldname* es un texto que es idéntico a un nombre de campo en *table*. (Tenga en cuenta que el nombre de campo debe ir entre comillas dobles rectas o corchetes si contiene por ejemplo espacios u otros caracteres no estándar.) *aliasname* es un nombre alternativo para el campo que se utilizará en el archivo QVD o CSV resultante.
table	Es una tabla etiquetada en el script, ya cargada, que se usará como fuente de datos.

Argumento	Descripción
filename	El nombre del archivo destino, incluyendo una ruta válida a una conexión de datos de carpetas. **Ejemplo: *'lib://Table Files/target.qvd'*** En el modo de elaboración de scripts de legado, se admiten también los diguientes formatos de ruta: • Absoluta **Ejemplo: *c:\data\sales.qvd*** • relativa al directorio de trabajo de la app Qlik Sense. **Ejemplo: *data\sales.qvd*** Si se omite la ruta, Qlik Sense almacena el archivo en el directorio especificado por la sentencia **Directory**. Si no hay sentencia **Directory**, Qlik Sense almacena el archivo en el directorio de trabajo, *C:\Users\{user}\Documents\Qlik\Sense\Apps*. •

Argumento	Descripción
*format-spec ::=***(** (**txt** | **qvd** | **parquet**), **compression is** *codec***)**	Puede establecer la especificación de formato en cualquiera de estos formatos de archivo. Si se omite la especificación de formato, se asume **qvd**. • **txt** para archivos CSV y TXT. • **qvd** para archivos QVD. • **parquet** para archivos Parquet. Si utiliza **parquet**, también puede establecer qué códec de compresión usar con **compression is**. Si no especifica el códec de compresión con compresión con **compression is**, se utiliza `snappy`. Están disponibles las siguientes opciones de compresión: • `uncompressed` • `snappy` • `gzip` • `lz4` • `brotli` • `zstd` • `lz4_hadoop` Ejemplo: `Store mytable into [lib://DataFiles/myfile.parquet] (parquet, compression is lz4);`

Ejemplos:

```
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 myfile.parquet (parquet);

Store * from mytable into 'lib://FolderConnection/myfile.qvd';
```

La extensión de archivo de las conexiones de DataFiles distingue entre mayúsculas y minúsculas. Por ejemplo: .qvd.

Almacenamiento en archivos parquet

Parquet es un formato de archivo fuertemente tipado, en el que cada campo contiene un único tipo específico de datos (como in32, double, timestamp o text). Qlik Sense almacena datos internos como un dual escrito libremente, donde los datos de diferentes fuentes se pueden mezclar en los mismos campos. Como solo se puede almacenar una parte del dual en cada campo de Parquet, es importante saber qué contiene cada campo. De forma predeterminada, Qlik Sense utiliza el tipo de campo para determinar cómo se debe almacenar el campo. Al almacenar datos en archivos Parquet en un formato específico, debe especificar qué tipo de datos son sus campos al cargarlos. Si trata de almacenar datos en campos incompatibles en un archivo Parquet, como números en un campo de texto o texto en un campo de marca de tiempo, terminará con valores nulos.

Al cargar datos que desea almacenar en Parquet, es posible cambiar el comportamiento predeterminado. Puede darles formato para cambiar su tipo de datos, o bien etiquetarlos para obligar a unos tipos de columnas específicos en Parquet.

Formateado de datos para almacenamiento en Parquet

Puede utilizar las funciones de formato de Qlik Sense para clasificar sus datos. Por ejemplo, **Text()**, **Num()**, **Interval()** o **Timestamp()** pueden aplicar formatos de datos al almacenar datos en Parquet. Qlik Sense puede almacenar datos en casi 20 tipos de datos según los atributos de campo y las etiquetas de campo automáticas. Para más información, vea *Funciones de interpretación (page 1281)*.

Ejemplo: Formateado de datos con Num() y Text()

El ejemplo siguiente demuestra la preparación de datos para su almacenamiento en Parquet. **Num()** se aplica al campo numérico. **Text()** se aplica tanto a texto como a mixto. En el caso de formato mixto, **Text()** evita que se trate como un campo numérico en Parquet y que los valores de texto se cambien a valores nulos.

```
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://DataFiles/Tmp.parquet] (parquet);
```

Etiquetado de datos para almacenamiento en Parquet

Puede etiquetar sus datos con etiquetas `$parquet` para obligar a tener unos tipos de columnas específicos al almacenar datos en Parquet. Cada tipo de datos se puede aplicar agregando la etiqueta de control correspondiente. Por ejemplo, para almacenar un campo como INT32 en Parquet, etiquételo con `$parquet-int32` en el script de carga. Dependiendo del tipo de datos, se almacenará la cadena o la representación numérica de los datos duales.

Las siguientes etiquetas de control de Parquet se pueden utilizar para etiquetar campos con el fin de almacenarlos en archivos Parquet.

Etiquetas de control de Parquet

Etiqueta de control	Dual	Tipo físico	Tipo lógico	Tipo convertido
$parquet-boolean	Número	BOOLEAN	NONE	NONE
$parquet-int32	Número	INT32	NONE	NONE
$parquet-int64	Número	INT64	NONE	NONE
$parquet-float	Número	FLOAT	NONE	NONE
$parquet-double	Número	DOUBLE	NONE	NONE
$parquet-bytearray	Cadena	BYTE_ARRAY	NONE	UTF8
$parquet-bytearrayfix	Número	FIXED_LEN_BYTE_ARRAY	NONE	DECIMAL
$parquet-decimal	Número	INT64	DECIMAL	DECIMAL
$parquet-date	Número	INT32	DATE	DATE
$parquet-time	Número	INT64	TIME	TIME_MICROS
$parquet-timestamp	Número	INT64	TIMESTAMP	TIMESTAMP_MICROS
$parquet-string	Cadena	BYTE_ARRAY	STRING	UTF8
$parquet-enum	Cadena	BYTE_ARRAY	ENUM	ENUM
$parquet-interval	Número	FIXED_LEN_BYTE_ARRAY	INTERVAL	INTERVAL
$parquet-json	Cadena	BYTE_ARRAY	JSON	JSON
$parquet-bson	Cadena	BYTE_ARRAY	BSON	BSON
$parquet-uuid	Cadena	FIXED_LEN_BYTE_ARRAY	UUID	NONE

Ejemplo: Etiquetado de datos para almacenamiento en Parquet

En este ejemplo, se utilizan dos etiquetas para definir los datos de Parquet. El campo *num* va etiquetado con `$parquet-int32` para definirlo como un campo numérico que se establecerá como INT32 en Parquet.

```
Data:
LOAD * INLINE [
num, text,
123.321, abc
456.654, def
789.987, ghi
];
TAG num WITH '$parquet-int32';
STORE Data INTO [lib://DataFiles/Tmp.parquet] (parquet);
```

Table/Tables

Las palabras clave de script **Table** y **Tables** se utilizan en sentencias **Drop**, **Comment** y **Rename**, así como también como especificador de formato en sentencias **Load**.

Tag

Esta sentencia de script proporciona una forma de asignar etiquetas a uno o más campos o tablas. Si se intenta etiquetar un campo o una tabla que no está presente en la app, se ignorará el etiquetado. Si hubiera múltiples nombres de un mismo campo o etiqueta, se empleará el último valor.

Sintaxis:

```
Tag [field|fields] fieldlist with tagname
```

```
Tag [field|fields] fieldlist using mapname
```

```
Tag table tablelist with tagname
```

Argumentos

Argumento	Descripción
fieldlist	Uno o varios campos que deben etiquetarse, en una lista separada por comas.
mapname	El nombre de una tabla de correspondencia previamente leída por una sentencia **mapping Load** o **mapping Select**.
tablelist	Una lista separada por comas de las tablas que deben etiquetarse.
tagname	Es el nombre de la etiqueta que debería aplicarse al campo.

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
tag fields using tagmap;
```

Example 2:

```
tag field Alpha with 'MyTag2';
```

Trace

La sentencia **trace** escribe una cadena en la ventana de **Progreso de ejecución de script** y en el archivo de registro de script, cuando se utiliza. Es muy útil cuando se desea efectuar una depuración. Utilizando expansiones $ de variables que se calculan antes de la sentencia **trace**, puede personalizar el mensaje.

Sintaxis:

```
Trace string
```

Example 1:

La instrucción siguiente se puede utilizar justo después de la instrucción Load que carga la tabla "Main" (Principal).

```
Trace Main table loaded;
```
Esto mostrará el texto "Main table loaded" (Tabla principal cargada) en el cuadro de diálogo de ejecución del script y en el archivo de registro.

Example 2:

La instrucción siguiente se puede utilizar justo después de la instrucción Load que carga la tabla "Main" (Principal).

```
Let MyMessage = NoOfRows('Main') & ' rows in Main table';

Trace $(MyMessage);
```
Esto mostrará un texto que muestra el número de filas en el cuadro de diálogo de ejecución del script y en el archivo de registro, por ejemplo, '265.391 filas en la tabla principal'.

Unmap

La sentencia **Unmap** desactiva la asignación de valores de campo especificada por una sentencia **Map ... Using** anterior para los campos cargados posteriormente.

Sintaxis:

```
Unmap *fieldlist
```

Argumentos:

Argumentos

Argumento	Descripción
*fieldlist	es una lista separada por comas de los campos que ya no deberían enlazarse a partir de este punto del script. Usar * como campo indica la totalidad de campos. Se permiten los caracteres comodín * y ? en nombres de campo. Puede que sea necesario entrecomillar los nombres de campo cuando se empleen caracteres comodín.

Ejemplos y resultados:

Ejemplo	Resultado
Unmap Country;	Deshabilita la correspondencia del campo Country.
Unmap A, B, C;	Deshabilita la correspondencia de los campos A, B y C.
Unmap * ;	Impide enlaces de todos los campos.

Unqualify

La sentencia **Unqualify** se utiliza para desactivar la calificación de los nombres de campo que la sentencia **Qualify** activó previamente.

Sintaxis:

```
Unqualify *fieldlist
```

Argumentos:

Argumentos

Argumento	Descripción
*fieldlist	Es un listado de campos separados por comas en los que se ha de aplicar la cualificación. Usar * como campo indica la totalidad de campos. Se permiten los caracteres comodín * y ? en nombres de campo. Puede que sea necesario entrecomillar los nombres de campo cuando se empleen caracteres comodín. Consulte la documentación de la sentencia **Qualify** para más información.

Example 1:

En una base de datos con la que no esté familiarizado, puede que quiera empezar asegurándose de que sólo uno o unos pocos campos estén asociados, tal como se muestra en el ejemplo:

```
qualify *;
unqualify TransID;
SQL SELECT * from tab1;
SQL SELECT * from tab2;
SQL SELECT * from tab3;
```

Primero la cualificación está activada para todos los campos.
Luego se desactiva la cualificación para**TransID**.
Solo se usará **TransID** para asociaciones entre las tablas *tab1*, *tab2* y *tab3*. Todos los demás campos se cualificarán con el nombre de la tabla.

Untag

Esta sentencia de script proporciona una manera de eliminar etiquetas de campos o tablas. Si se intenta eliminar la etiqueta de un campo o una tabla que no está presente en la app, se ignorará la eliminación de la etiqueta.

Sintaxis:

```
Untag [field|fields] fieldlist with tagname
```

```
Untag [field|fields] fieldlist using mapname
```

```
Untag table tablelist with tagname
```

Argumentos:

Argumentos

Argumento	Descripción
fieldlist	Uno o varios campos cuyas etiquetas deben eliminarse, en una lista separada por comas.
mapname	El nombre de una tabla de correspondencia previamente leída en una sentencia mapping **LOAD** o mapping **SELECT**.
tablelist	Una lista separada por comas de las tablas que deben desetiquetarse.
tagname	Es el nombre de la etiqueta que debería eliminarse del campo.

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
Untag fields using tagmap;
```

Example 2:

```
Untag field Alpha with MyTag2;
```

3.4 Directorio de trabajo

Si estamos remitiendo a un archivo en una sentencia de script y se omite la ruta, Qlik Sense busca el archivo en el orden siguiente:

1. El directorio especificado por una sentencia **Directory** (solo es compatible con el modo de script heredado).
2. Si no hay sentencia **Directory**, Qlik Sense busca en el directorio de trabajo.

Directorio de trabajo de Qlik Sense Desktop

En Qlik Sense Desktop, el directorio de trabajo es *C:\Users\{user}\Documents\Qlik\Sense\Apps*.

Directorio de trabajo de Qlik Sense

En una instalación de servidor de Qlik Sense, el directorio de trabajo se especifica en Qlik Sense Repository Service, por defecto es *C:\ProgramData\Qlik\Sense\Apps*. Consulte la ayuda de Consola de gestión de Qlik para obtener más información.

4 Trabajar con variables en el editor de carga de datos

Una variable en Qlik Sense es un contenedor que almacena un valor estático o un cálculo, por ejemplo un valor numérico o alfanumérico. Cuando utilice la variable en la app, cualquier cambio efectuado en la variable se aplica en todos los lugares en que se use la variable. Puede definir variables en el panel de variables o en el script, usando el Editor de carga de datos. Establece el valor de una variable usando las sentencias **Let** o **Set** en el script de carga de datos.

También puede trabajar con las variables de Qlik Sense desde el panel de variables cuando editamos una hoja.

4.1 General

Si el primer carácter del valor de una variable es un signo igual ' = ', Qlik Sense tratará de evaluar el valor como una fórmula (o expresión de Qlik Sense) y a continuación mostrará o devolverá el resultado en lugar del texto mismo de la fórmula.

Cuando se utiliza una variable, ésta es sustituida por su valor. Las variables se pueden utilizar en el script para expansión de signo dólar y en varias sentencias de control. Esto puede resultar de gran utilidad cuando la misma cadena se repite muchas veces en el script, por ejemplo, una ruta.

Hay algunas variables de sistema especiales que Qlik Sense fija al comienzo de la ejecución de script, independientemente de cuáles fueran sus valores previos.

4.2 Definir una variable

Las variables brindan la capacidad de almacenar valores estáticos o el resultado de un cálculo. Al definir una variable, utilice la siguiente sintaxis:

```
set variablename = string
```

o bien

```
let variable = expression
```

La sentencia **Set** se utiliza para la asignación de cadenas. Asigna el texto a la derecha del signo igual a la variable. La sentencia **Let** evalúa una expresión a la derecha del signo igual en el tiempo de ejecución del script y asigna el resultado de la expresión a la variable.

Las variables son sensibles a mayúsculas.

No se recomienda utilizar el mismo nombre para un campo y una función en Qlik Sense

Ejemplos:

`set x = 3 + 4; //` la variable obtendrá la cadena "3 + 4" como valor.

`let x = 3 + 4; //` devuelve 7 como valor.

`set x = Today(); //` devuelve "Today()" como valor.

`let x = Today(); //` devuelve la fecha de hoy como valor, por ejemplo, "27/9/2021".

4.3 Eliminar una variable

Si elimina una variable del script y vuelve a cargar los datos, la variable permanecerá en la app. Si desea eliminar completamente la variable de la app, también debe eliminarla del diálogo de variables.

4.4 Cargar un valor de variable como un valor de campo

Si desea cargar un valor de una variable como un valor de campo en una sentencia **LOAD** y el resultado de la expansión dólar es de texto en vez de numérico o una expresión, entonces necesita encerrar la variable expandida entre comillas simples.

Ejemplo:

El ejemplo a continuación carga la variable de sistema que contiene las lista de errores de script en una tabla. Puede observar que la expansión de ScriptErrorCount en la cláusula **If** no requiere comillas, mientras que la expansión de ScriptErrorList sí requiere comillas.

```
IF $(ScriptErrorCount) >= 1 THEN

   LOAD '$(ScriptErrorList)' AS Error AutoGenerate 1;
END IF
```

4.5 Cálculo de variables

Hay varias maneras de utilizar variables con valores calculados en Qlik Sense y el resultado depende de cómo se defina y de cómo se denomine en una expresión.

En este ejemplo, cargamos algunos datos inline:

```
LOAD * INLINE [
    Dim, Sales
    A, 150
    A, 200
    B, 240
    B, 230
    C, 410
    C, 330
];
```

Vamos a definir dos variables:

```
Let vSales  = 'Sum(Sales)' ;
```

```
Let vSales2  = '=Sum(Sales)' ;
```

En la segunda variable, añadimos un signo igual antes de la expresión. De este modo la variable se calculará antes de que se expanda y se evalúe la expresión.

Si utiliza la variable vSales tal como está, por ejemplo, en una medida, el resultado será la cadena Sum(Sales), es decir, no se realizará ningún cálculo.

Si agrega una expansión de signo de dólar y llama a $(vSales) en la expresión, la variable se expandirá y se mostrará la suma de Sales.

Por último, si llama a $(vSales2), la variable se calculará antes de expandirse. Esto significa que el resultado mostrado será la suma total de Sales. La diferencia entre usar =$(vSales) y =$(vSales2) como expresiones de medida se ve en este cuadro que muestra los resultados:

Resultados

Dim	$(vSales)	$(vSales2)
A	350	1560
B	470	1560
C	740	1560

Como se puede observar, $(vSales) da como resultado la suma parcial de un valor de dimensión, mientras que $(vSales2) da como resultado la suma total.

Están disponibles las siguientes variables de script:

- *Variables de error (page 277)*
- *Variables de interpretación numérica (page 212)*
- *Variables de sistema (page 204)*
- *Variables de manejo de valores (page 210)*

4.6 Variables de sistema

Las variables de sistema, algunas de las cuales son definidas por el propio sistema, ofrecen información sobre el sistema y la app Qlik Sense.

Descripción general de las variables de sistema

Algunas de las funciones se describen a continuación tras la vista genérica. Para esas funciones, puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

CreateSearchIndexOnReload

Esta variable define si deben crearse archivos indexados de búsqueda durante la recarga de datos.

```
CreateSearchIndexOnReload
```

Floppy
Devuelve la letra de la primera unidad de disco que encuentra, normalmente *a:*. Esta es una variable definida por el sistema.

```
Floppy
```

Esta variable no es posible en modo estándar.

CD
Devuelve la letra de la primera unidad de CD-ROM que encuentre. Si no encuentra ningún CD-ROM, devuelve *c:*. Esta es una variable definida por el sistema.

```
CD
```

Esta variable no es posible en modo estándar.

HidePrefix
Todos los nombres de campo que comiencen por esta cadena de texto, se ocultarán de la misma manera que los campos del sistema. Esta es una variable definida por el usuario.

```
HidePrefix
```

HideSuffix
Todos los nombres de campo que terminen con esta cadena de texto se ocultarán de la misma forma que los campos de sistema. Esta es una variable definida por el usuario.

```
HideSuffix
```

Include
La variable **Include/Must_Include** especifica un archivo que contiene texto que debe incluirse en el script y evaluarse como código de script. No se utiliza para añadir datos. Puede almacenar partes de su código de script en un archivo de texto aparte y reutilizarlo en diversas apps. Esta es una variable definida por el usuario.

```
$(Include=filename)
$(Must_Include=filename)
```

OpenUrlTimeout
Esta variable define el tiempo de espera en segundos que Qlik Sense deberá respetar al obtener datos de fuentes URL (por ejemplo, HTML páginas web). Si se omite, el tiempo establecido será de unos 20 minutos.

```
OpenUrlTimeout
```

QvPath
Devuelve la cadena de búsqueda al ejecutable de Qlik Sense: Esta es una variable definida por el sistema.

```
QvPath
```

Esta variable no es posible en modo estándar.

QvRoot
Devuelve el directorio raíz del ejecutable de Qlik Sense: Esta es una variable definida por el sistema.

```
QvRoot
```

Esta variable no es posible en modo estándar.

QvWorkPath
Devuelve la cadena de búsqueda a la app actual de Qlik Sense. Esta es una variable definida por el sistema.

```
QvWorkPath
```

Esta variable no es posible en modo estándar.

QvWorkRoot
Devuelve el directorio raíz de la app actual de Qlik Sense. Esta es una variable definida por el sistema.

```
QvWorkRoot
```

Esta variable no es posible en modo estándar.

StripComments
Si esta variable se define en 0, se inhibirá la eliminación de los comentarios /*..*/ y // en el script. Si no se define esta variable, las líneas de comentario se ejecutarán siempre.

```
StripComments
```

Verbatim
Normalmente todos los valores de campo son despojados automáticamente de sus caracteres precedentes y posteriores vacíos (ASCII 32) antes de ser cargados en la base de datos de Qlik Sense. Si esta variable se configura en 1, no se eliminarán estos caracteres vacíos. Los caracteres del tabulador (ASCII 9) y el espacio fijo (ANSI 160) nunca sufren esta acción.

```
Verbatim
```

WinPath
Devuelve la cadena de exploración a Windows. Esta es una variable definida por el sistema.

```
WinPath
```

Esta variable no es posible en modo estándar.

WinRoot
Devuelve el directorio raíz de Windows. Esta es una variable definida por el sistema.

```
WinRoot
```

Esta variable no es posible en modo estándar.

CollationLocale
Especifica qué datos locales utilizar para el criterio de ordenación y la correspondencia de búsquedas. El valor es un nombre de tipo cultural relativo a un dato local, por ejemplo 'en-US'.Esta es una variable definida por el sistema.

```
CollationLocale
```

CreateSearchIndexOnReload

Esta variable define si deben crearse archivos indexados de búsqueda durante la recarga de datos.

Sintaxis:

```
CreateSearchIndexOnReload
```

Se puede definir si los archivos indexados de búsqueda deben crearse durante la recarga de datos, o si deben crearse tras la primera solicitud de búsqueda del usuario. La ventaja de crear archivos indexados de búsqueda durante la recarga de datos es que se evita el tiempo de espera que experimenta el primer usuario que realiza una búsqueda. Esto debe sopesarse en función del tiempo de recarga de datos requerido por la creación del índice de búsqueda.

Si se omite esta variable, los archivos indexados de búsqueda no se crearán durante la recarga de datos.

Para apps de sesión, los archivos indexados de búsqueda no se crearán durante la recarga de datos, independientemente de la configuración de esta variable.

Example 1: Crear campos indexados de búsqueda durante la recarga de datos

```
set CreateSearchIndexOnReload=1;
```

Example 2: Crear campos indexados de búsqueda tras la primera solicitud de búsqueda

```
set CreateSearchIndexOnReload=0;
```

HidePrefix

Todos los nombres de campo que comiencen por esta cadena de texto, se ocultarán de la misma manera que los campos del sistema. Esta es una variable definida por el usuario.

Sintaxis:

```
HidePrefix
```

Ejemplo:

```
set HidePrefix='_' ;
```

Si se usa esta sentencia, los nombres de campo que comiencen con guión bajo no se mostrarán en las listas de nombres de campo cuando los campos del sistema estén ocultos.

HideSuffix

Todos los nombres de campo que terminen con esta cadena de texto se ocultarán de la misma forma que los campos de sistema. Esta es una variable definida por el usuario.

Sintaxis:

```
HideSuffix
```

Ejemplo:

```
set HideSuffix='%';
```

Si se emplea esta sentencia, los nombres de campo que terminen con un signo de porcentaje no se mostrarán en las listas de nombres de campo cuando los campos del sistema estén ocultos.

Include

La variable **Include/Must_Include** especifica un archivo que contiene texto que debe incluirse en el script y evaluarse como código de script. No se utiliza para añadir datos. Puede almacenar partes de su código de script en un archivo de texto aparte y reutilizarlo en diversas apps. Esta es una variable definida por el usuario.

Esta variable admite únicamente conexiones de datos de carpetas en modo estándar.

Sintaxis:

```
$(Include=filename)
```

```
$(Must_Include=filename)
```

Hay dos versiones de la variable:

- **Include** no genera un error si no encuentra el archivo, fallará en silencio.
- **Must_Include** genera un error si no encuentra el archivo.

Si no se especifica ninguna ruta, el nombre del archivo estará relacionado con el directorio de trabajo de la app Qlik Sense. También puede especificar una ruta de archivo absoluta o una ruta a una conexión de carpeta lib://. No ponga un carácter de espacio antes o después del signo igual.

*La construcción **set Include** =filename no es aplicable.*

Ejemplos:

```
$(Include=abc.txt);
```

```
$(Must_Include=lib://DataFiles/abc.txt);
```

Limitaciones

Compatibilidad cruzada limitada entre archivos codificados en UTF-8 en Windows frente a Linux.

Es opcional utilizar UTF-8 con BOM (marca de orden de bytes). BOM puede interferir con el uso de UTF-8 en software que no espera bytes que no sean ASCII al comienzo de un archivo, pero que de otra manera podría manejar el flujo de texto.

- Los sistemas Windows usan BOM en UTF-8 para identificar que un archivo está codificado en UTF-8, a pesar de que no hay ambigüedad en el almacenamiento de bytes.
- Unix/Linux usa UTF-8 para Unicode, pero no usa la lista de materiales, ya que esto interfiere con la sintaxis de los archivos de comando.

Esto tiene algunas implicaciones para Qlik Sense.

- En Windows, cualquier archivo que comience con una lista de materiales UTF-8 se considera un archivo de secuencia de comandos UTF-8. De lo contrario, se asume la codificación ANSI.
- De lo contrario, se asume la codificación ANSI. Es por eso que el UTF-8 funciona aunque no contiene una lista de materiales.

Como resultado, no se puede garantizar la portabilidad. No siempre es posible crear un archivo en Windows que pueda ser interpretado por Linux y viceversa. No hay compatibilidad cruzada entre los dos sistemas con respecto a los archivos codificados en UTF-8 debido al manejo diferente de la lista de materiales.

OpenUrlTimeout

Esta variable define el tiempo de espera en segundos que Qlik Sense deberá respetar al obtener datos de fuentes URL (por ejemplo, HTML páginas web). Si se omite, el tiempo establecido será de unos 20 minutos.

Sintaxis:

```
OpenUrlTimeout
```

Ejemplo:

```
set OpenUrlTimeout=10;
```

StripComments

Si esta variable se define en 0, se inhibirá la eliminación de los comentarios /*..*/ y // en el script. Si no se define esta variable, las líneas de comentario se ejecutarán siempre.

Sintaxis:

```
StripComments
```

Algunos drivers de bases de datos utilizan /*..*/ como sugerencias de optimización en sentencias **SELECT**. Si este es el caso, los comentarios no deben eliminarse antes de enviar la sentencia **SELECT** al driver de la base de datos.

Se recomienda que esta variable se establezca en 1 inmediatamente por detrás de la(s) sentencia(s) donde se necesite.

Ejemplo:

```
set StripComments=0;
SQL SELECT * /* <optimization directive> */ FROM Table ;
set StripComments=1;
```

Verbatim

Normalmente todos los valores de campo son despojados automáticamente de sus caracteres precedentes y posteriores vacíos (ASCII 32) antes de ser cargados en la base de datos de Qlik Sense. Si esta variable se configura en 1, no se eliminarán estos caracteres vacíos. Los caracteres del tabulador (ASCII 9) y el espacio fijo (ANSI 160) nunca sufren esta acción.

Sintaxis:

```
Verbatim
```

Ejemplo:

```
set Verbatim = 1;
```

4.7 Variables de manejo de valores

Esta sección describe las variables que se utilizan para la gestión de valores NULL y otros valores.

Descripción general de las variables de manejo de valores

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

NullDisplay
El símbolo definido sustituirá todos los valores NULL de ODBC y conectores en el nivel más bajo de datos. Esta es una variable definida por el usuario.

```
NullDisplay
```

NullInterpret
El símbolo definido se interpretará como NULL cuando aparece en un archivo de texto, archivo Excel o una sentencia inline. Esta es una variable definida por el usuario.

```
NullInterpret
```

NullValue
Si se utiliza la sentencia **NullAsValue**, el símbolo definido sustituirá a todos los valores NULL en los campos **NullAsValue** especificados con la cadena especificada.

```
NullValue
```

OtherSymbol
Define que un símbolo se trate como "todos los demás valores" antes de una sentencia **LOAD/SELECT**. Esta es una variable definida por el usuario.

```
OtherSymbol
```

NullDisplay

El símbolo definido sustituirá todos los valores NULL de ODBC y conectores en el nivel más bajo de datos. Esta es una variable definida por el usuario.

Sintaxis:

```
NullDisplay
```

Ejemplo:

```
set NullDisplay='<NULL>';
```

NullInterpret

El símbolo definido se interpretará como NULL cuando aparece en un archivo de texto, archivo Excel o una sentencia inline. Esta es una variable definida por el usuario.

Sintaxis:

```
NullInterpret
```

Ejemplos:

```
set NullInterpret=' ';
set NullInterpret =;
```

no devolverá valores NULL por valores en blanco de Excel, sino por un archivo de texto CSV.

```
set NullInterpret ='';
```

devolverá valores NULL por valores en blanco en Excel.

NullValue

Si se utiliza la sentencia **NullAsValue**, el símbolo definido sustituirá a todos los valores NULL en los campos **NullAsValue** especificados con la cadena especificada.

Sintaxis:

```
NullValue
```

Ejemplo:

```
NullAsValue Field1, Field2;
set NullValue='<NULL>';
```

OtherSymbol

Define que un símbolo se trate como "todos los demás valores" antes de una sentencia **LOAD/SELECT**. Esta es una variable definida por el usuario.

Sintaxis:

```
OtherSymbol
```

Ejemplo:

```
set OtherSymbol='+';
LOAD * inline
[X, Y
a, a
b, b];
LOAD * inline
[X, Z
a, a
+, c];
```

El valor de campo Y='b' enlazará ahora con Z='c' mediante el otro símbolo.

4.8 Variables de interpretación numérica

Las variables de interpretación numérica son variables definidas por el sistema. Las variables vienen incluidas en la parte superior del script de carga y aplican los ajustes de formato numérico en el momento de la ejecución del script. Se pueden borrar, editar o duplicar.

Las variables de interpretación numérica se generan de forma automática conforme a la configuración actual del sistema operativo al crear una nueva app. En Qlik Sense Desktop, esto va de acuerdo con la configuración del sistema operativo del PC. En Qlik Sense, es según el sistema operativo del servidor donde Qlik Sense esté instalado. Si el servidor de Qlik Cloud al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Qlik Sense Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Formato de moneda

MoneyDecimalSep
El separador decimal definido reemplaza el símbolo decimal de moneda establecido en la configuración regional.

```
MoneyDecimalSep
```

MoneyFormat
El símbolo definido reemplaza el símbolo de moneda establecido en la configuración regional.

```
MoneyFormat
```

MoneyThousandSep
El separador de miles definido reemplaza el símbolo de agrupación de dígitos de moneda establecido en la configuración regional.

```
MoneyThousandSep
```

Formato numérico

DecimalSep
El separador decimal definido reemplaza el símbolo decimal establecido en la configuración regional.

```
DecimalSep
```

ThousandSep
El separador de miles definido reemplaza al símbolo de agrupación de dígitos del sistema operativo (configuración regional).

```
ThousandSep
```

NumericalAbbreviation
La abreviatura numérica establece qué abreviatura usar para los prefijos de escala de numerales, por ejemplo M para mega o un millón (10^6) y µ para micro (10^{-6}).

```
NumericalAbbreviation
```

Formato de tiempo

DateFormat
Esta variable de entorno define el formato de fecha utilizado como predeterminado en la aplicación. El formato se utiliza tanto para interpretar como para formatear fechas. Si la variable no está definida, el formato de fecha de la configuración regional del sistema operativo se obtendrá cuando se ejecute el script.

```
DateFormat
```

TimeFormat
El formato definido reemplaza el formato de hora del sistema operativo (configuración regional).

```
TimeFormat
```

TimestampFormat
El formato definido reemplaza los formatos de fecha y hora del sistema operativo (configuración Regional).

```
TimestampFormat
```

MonthNames
El formato definido reemplaza la convención de nombres de meses establecida en la configuración regional.

```
MonthNames
```

LongMonthNames
El formato definido reemplaza la convención de nombres largos de meses establecida en la configuración regional.

```
LongMonthNames
```

DayNames
El formato definido reemplaza la convención de nombres de los días la semana establecida en la configuración regional.

```
DayNames
```

LongDayNames
El formato definido reemplaza la convención de nombres largos de los días la semana establecida en la configuración regional.

```
LongDayNames
```

FirstWeekDay
Un entero que define qué día se utilizará como primer día de la semana.

```
FirstWeekDay
```

BrokenWeeks
Esta configuración define si las semanas están divididas o no.

```
BrokenWeeks
```

ReferenceDay
La configuración define qué día de enero se establece como día de referencia para definir la semana 1.

```
ReferenceDay
```

FirstMonthOfYear
El parámetro define qué mes usar como primer mes del año, lo cual puede servir para definir años financieros que utilicen un desplazamiento mensual, por ejemplo, con inicio el 1 de abril.

Este parámetro actualmente no se usa pero se reserva para un uso futuro.

Parámetros válidos son 1 (enero) a 12 (diciembre). El parámetro por defecto es 1.

Sintaxis:

```
FirstMonthOfYear
```

Ejemplo:

```
Set FirstMonthOfYear=4; //Sets the year to start in April
```

BrokenWeeks

Esta configuración define si las semanas están divididas o no.

Sintaxis:

```
BrokenWeeks
```

En Qlik Sense, la configuración regional se obtiene cuando se crea la app y la configuración correspondiente se almacena en el script como variables de entorno.

Un desarrollador de aplicaciones norteamericano a menudo incluye `Set BrokenWeeks=1;` en el script, lo que corresponde a semanas interrumpidas. Un desarrollador de aplicaciones europeo a menudo incluye `Set BrokenWeeks=0;` en el script, lo que corresponde a semanas ininterrumpidas.

Semanas ininterrumpidas significa que:

- En algunos años, la semana 1 comienza en diciembre y en otros años, la última semana del año anterior continúa hasta enero.
- Según la ISO 8601, la semana 1 siempre tiene al menos 4 días en enero. En Qlik Sense, esto se puede configurar usando la variable `ReferenceDay`.

Semanas interrumpidas significa que:

- La última semana del año nunca continúa hasta enero.
- La semana 1 empieza el 1 de enero y, en la mayoría de los casos, no es una semana completa.

Se pueden utilizar los siguientes valores:

- 0 (= se utilizan semanas ininterrumpidas)
- 1 (= se utilizan semanas interrumpidas)

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las

fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos:

Si desea configuraciones ISO para semanas y números de semana, asegúrese de tener lo siguiente en el script:

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;   //(use unbroken weeks)
Set ReferenceDay=4;
```

Si desea la configuración de EE. UU., asegúrese de tener lo siguiente en el script:

```
Set FirstWeekDay=6;
Set BrokenWeeks=1;   //(use broken weeks)
Set ReferenceDay=1;
```

DateFormat

Esta variable de entorno define el formato de fecha utilizado como predeterminado en la app y por fecha, que devuelve funciones como `date()` y `date#()`. El formato se utiliza para interpretar y dar formato a fechas. Si la variable no está definida, el formato de fecha establecido por su configuración regional se obtendrá al ejecutar el script.

Sintaxis:

```
DateFormat
```

Ejemplos de la función DateFormat

Ejemplo	Resultado
`Set DateFormat='M/D/YY'; //(US format)`	Este uso de la función `DateFormat` define la fecha en el formato de EE. UU., mes/día/año.
`Set DateFormat='DD/MM/YY'; //(UK date format)`	Este uso de la función `DateFormat` define la fecha en el formato de UK, día/mes/año.
`Set DateFormat='YYYY/MM/DD'; //(ISO date format)`	Este uso de la función `DateFormat` define la fecha en el formato ISO, año/mes/día.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las

fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: valor predeterminado de las variables del sistema

Script de carga y resultados

General

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de fechas.
- La función `DateFormat`, que utilizará el formato de fecha de EE. UU.

En este ejemplo, se carga un conjunto de datos en una tabla denominada "`Transactions`". Incluye un campo `date`. Se utiliza la definición de `DateFormat`. Este patrón se utilizará para la conversión implícita de texto a fecha cuando se carguen las fechas de texto.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `month`

Cree esta medida:

```
=sum(amount)
```

Tabla de resultados

date	mes	=sum(amount)
01/01/2022	Ene	1000
02/01/2022	Feb	2123
03/01/2022	Mar	4124
04/01/2022	Abr	2431

La definición de `DateFormat` MM/DD/AAAA se utiliza para la conversión implícita de texto a fechas, por lo que el campo `date` se interpreta correctamente como una fecha. Se utiliza el mismo formato para mostrar la fecha, como se muestra en la tabla de resultados.

Ejemplo 2: cambiar la variable del sistema

Script de carga y resultados

General

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- El mismo conjunto de datos del ejemplo anterior.
- La función `DateFormat`, que utilizará el formato "DD/MM/AAAA".

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `month`

Cree esta medida:

```
=sum(amount)
```

Tabla de resultados

date	mes	=sum(amount)
01/01/2022	Ene	1000
02/01/2022	Ene	2123
03/01/2022	Ene	4124
04/01/2022	Ene	2431

Debido a que la definición de `DateFormat` se estableció en "DD/MM/AAAA", puede ver que los dos dígitos tras el primer símbolo "/" se han interpretado como el mes, lo que da como resultado que todos los registros sean del mes de enero.

Ejemplo 3: interpretación de fechas

Script de carga y resultados

General

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos con fechas en formato numérico.
- La variable `DateFormat`, que utilizará el formato "DD/MM/AAAA".
- La variable `date()`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `month`

Cree esta medida:

```
=sum(amount)
```

Tabla de resultados

date	mes	=sum(amount)
06/03/2022	Jun	1000
06/04/2022	Jun	2123
06/05/2022	Jun	4124
06/07/2022	Jun	2431

En el script de carga, la función `date()` sirve para convertir la fecha numérica en un formato de fecha. Si no se proporciona un formato específico como segundo argumento en la función, se utiliza el formato `DateFormat`. Esto da como resultado que el campo de fecha use el formato "MM/DD/AAAA".

Ejemplo 4: formato de fecha extranjera

Script de carga y resultados

General

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de fechas.
- La variable `DateFormat`, que utiliza el formato "DD/MM/AAAA" pero no está comentada por barras diagonales.

Script de carga

```
// 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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `month`

Cree esta medida:

```
=sum(amount)
```

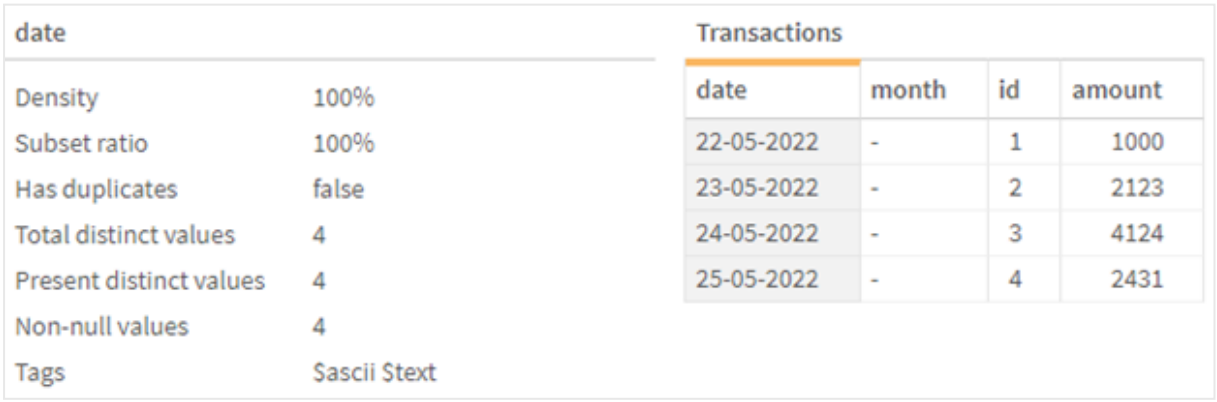
Tabla de resultados

date	mes	=sum(amount)
22-05-2022	-	1000
23-05-2022	-	2123
24-05-2022	-	4124
25-05-2022	-	2431

En el script de carga inicial, el `DateFormat` que se utiliza es el predeterminado "MM/DD/AAAA". Debido a que el campo `date` en el conjunto de datos de las transacciones no tiene este formato, el campo no se interpreta como una fecha. Esto se muestra en la tabla de resultados donde los valores del campo `month` son nulos.

Puede verificar los tipos de datos interpretados en el Visor del modelo de datos inspeccionando las propiedades de "Etiquetas" del campo `date`.

Vista previa de la tabla `Transactions`. Tenga en cuenta las "Etiquetas" del campo `date` que indican que los datos de entrada textuales no se han convertido implícitamente en una fecha/hora.

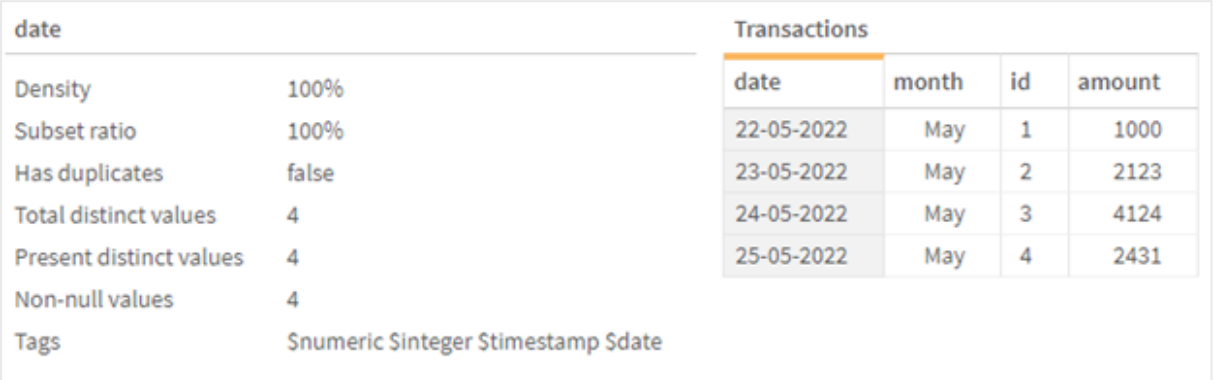


Esto se puede resolver habilitando la variable de sistema `DateFormat`:

```
// SET DateFormat='DD/MM/YYYY';
```

Elimine las barras diagonales dobles y vuelva a cargar los datos.

Vista previa de la tabla `Transactions`. Tenga en cuenta las "Etiquetas" del campo `date` que indican que los datos de entrada de texto se han convertido implícitamente en una marca de fecha/hora.



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

El formato definido reemplaza la convención de nombres de los días la semana establecida en la configuración regional.

Sintaxis:

`DayNames`

Al modificar la variable, se requiere un punto y coma ; para separar los valores individuales.

Ejemplos de la función DayName

Ejemplo de función	Definición de resultado
`Set DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';`	Este uso de la función DayNames define los nombres de los días en su forma abreviada.
`Set DayNames='M;Tu;W;Th;F;Sa;Su';`	Este uso de la función DayNames define los nombres de los días por sus primeras letras.

La función `DayNames` se utiliza a menudo en combinación con las siguientes funciones:

Funciones relacionadas

Función	Interacción
weekday (page 1087)	Función de script que devuelve `DayNames` como valores de campo.
Date (page 1248)	Función de script que devuelve `DayNames` como valores de campo.
LongDayNames (page 233)	Valores en forma larga de los nombres de días `DayNames`.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: valor predeterminado de las variables del sistema

Script de carga y resultados

Vista general

En este ejemplo, las fechas en el conjunto de datos se establecen en el formato MM/DD/AAAA.

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos con fechas, que se cargará en una tabla denominada `Transactions`.
- Un campo `date`.
- La definición de `DayNames` predeterminada.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- dayname

Cree esta medida:

```
sum(amount)
```

Tabla de resultados

date	dayname	sum(amount)
01/01/2022	Sáb	1000
02/01/2022	Mar	2123
03/01/2022	Mar	4124
04/01/2022	Vie	2431

En el script de carga, la función WeekDay se utiliza con el campo date como el argumento proporcionado. En la tabla de resultados, el resultado de salida de esta función WeekDay muestra los días de la semana en el formato de la definición de DayNames.

Ejemplo 2: cambiar la variable del sistema

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña. Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, al comienzo del script, la definición de DayNames se modifica para usar los días de la semana abreviados en afrikáans.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `dayname`

Cree esta medida:

```
sum(amount)
```

Tabla de resultados

date	dayname	sum(amount)
01/01/2022	Sa	1000
02/01/2022	Di	2123
03/01/2022	Di	4124
04/01/2022	Vr	2431

En la tabla de resultados, el resultado de salida de esta función `WeekDay` muestra los días de la semana en el formato de la definición de `DayNames`.

Es importante recordar que si se modifica el idioma de los días de la semana `DayNames` como se ha hecho en este ejemplo, `LongDayNames` todavía contendría los días de la semana en inglés. Esto debería modificarse también por tanto si ambas variables se utilizan en la aplicación.

Ejemplo 3: función de fecha

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos con fechas, que se cargará en una tabla denominada `Transactions`.
- Un campo `date`.
- La definición de `DayNames` predeterminada.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `dayname`

Cree esta medida:

```
sum(amount)
```

Tabla de resultados

date	dayname	sum(amount)
01/01/2022	Sáb	1000
02/01/2022	Mar	2123
03/01/2022	Mar	4124
04/01/2022	Vie	2431

Se utiliza la definición de `DayNames` predeterminada. En el script de carga, la función `Date` se utiliza con el campo `date` como el primer argumento. El segundo argumento es `WWW`. Este formato convierte el resultado en los valores almacenados en la definición de `DayNames`. Esto se muestra en la información de salida de la tabla de resultados.

DecimalSep

El separador decimal definido reemplaza al símbolo decimal establecido en la configuración regional.

Qlik Sense interpreta automáticamente el texto como números cada vez que se encuentra un patrón numérico reconocible. Las variables de sistema `ThousandSep` y `DecimalSep` determinar la composición de los patrones aplicados al analizar texto como números. Las variables `ThousandSep` y `DecimalSep` establecen el patrón de formato de número predeterminado al visualizar contenido numérico en gráficos y tablas frontales. Es decir, impacta directamente en las opciones de **Formato numérico** para cualquier expresión frontal.

Suponiendo que tenemos un separador de miles de coma "," y un separador decimal de ".", estos son ejemplos de patrones que se convertirían implícitamente en valores numéricos equivalentes:

0,000.00

0000.00

0,000

Aquí tiene ejemplos de patrones que permanecerían sin cambios como texto; es decir, no convertidos a numérico:

0.000,00

0,00

Sintaxis:

DecimalSep

Ejemplos de funciones

Ejemplo	Resultado
`Set DecimalSep='.';`	Establece "." Como el separador decimal.
`Set DecimalSep=',';`	Establece "," Como el separador decimal.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo: efecto de establecer variables de separador numérico en diferentes datos de entrada

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de sumas y fechas con las sumas establecidas en diferentes patrones de formato.

- Una tabla denominada `Transactions`.
- La variable `DecimalSep` que está establecida en `"."`.
- La variable `ThousandSep` que está establecida en `","`.
- La variable `delimiter` que se establece como el '|' carácter para separar los diferentes campos en una línea.

Script de carga

```
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 '|');
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `amount`.

Cree esta medida:

```
=sum(amount)
```

Tabla de resultados

Amount	**=Sum(amount)**
Totales	20814.7086
1.000-45	
3.756,178	
4124,35	
23.344	23.344
2431.36	2431.36

Amount	=Sum(amount)	
	2431.84	2431.84
	3,454.356	3454.356
	3554.284	3554.284
	4132.5246	4132.5246
	4,787	4787

Cualquier valor que no se interprete como número permanece como texto y se alinea a la izquierda de forma predeterminada. Todos los valores convertidos correctamente se alinean a la derecha y conservan el formato de entrada original.

La columna de expresiones muestra el equivalente numérico, que tiene un formato predeterminado con solo un separador decimal ".". Esto se puede anular con la configuración desplegable **Formato numérico** en la configuración de la expresión.

FirstWeekDay

Un entero que define qué día se utilizará como primer día de la semana.

Sintaxis:

```
FirstWeekDay
```

El lunes es el primer día de la semana según ISO 8601, el estándar internacional para la representación de fechas y horas. El lunes también se usa como el primer día de la semana en varios países, por ejemplo, en el Reino Unido, España, Francia, Alemania y Suecia.

Pero en otros países, como Estados Unidos y Canadá, se considera que el domingo es el comienzo de la semana.

En Qlik Sense, la configuración regional se obtiene cuando se crea la app y la configuración correspondiente se almacena en el script como variables de entorno.

Un desarrollador de aplicaciones norteamericano a menudo obtiene `Set FirstWeekDay=6;` en el script, lo que corresponde al domingo. Un desarrollador de aplicaciones europeo a menudo obtiene `Set FirstWeekDay=0;` en el script, lo que corresponde al lunes.

Valores que se pueden establecer para FirstWeekDay

Valor	Día
0	Lunes
1	Martes
2	Miércoles
3	Jueves

Valor	Día
4	Viernes
5	Sábado
6	Domingo

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos:

Si desea configuraciones ISO para semanas y números de semana, asegúrese de tener lo siguiente en el script:

```
Set FirstWeekDay=0;  // Monday as first week day
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

Si desea la configuración de EE. UU., asegúrese de tener lo siguiente en el script:

```
Set FirstWeekDay=6;  // Sunday as first week day
Set BrokenWeeks=1;
Set ReferenceDay=1;
```

Ejemplo 1: usar el valor predeterminado (script)

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

En este ejemplo, el script de carga usa el valor predeterminado de la variable de sistema Qlik Sense, `FirstWeekDay=6`. Estos datos contienen datos de los primeros 14 días de 2020.

Script de carga

```
// 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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `week`
- `weekday`

Tabla de resultados

Fecha	semana	weekday
01/01/2021	1	Mié
01/02/2021	1	Jue
01/03/2021	1	Vie
01/04/2021	1	Sáb
01/05/2021	2	Dom
01/06/2020	2	Lun
01/07/2020	2	Mar
01/08/2020	2	Mié
01/09/2020	2	Jue

Fecha	semana	weekday
01/10/2020	2	Vie
01/11/2020	2	Sáb
01/12/2020	3	Dom
01/13/2020	3	Lun
01/14/2020	3	Mar

Debido a que se está utilizando la configuración predeterminada, la variable del sistema `FirstWeekDay` se establece en 6. En la tabla de resultados, cada nueva semana se puede ver a partir del domingo (el 5 y 12 de enero).

Ejemplo 2: cambiar la variable FirstWeekDay (script)

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

En este ejemplo, los datos contienen datos de los primeros 14 días de 2020. Al comienzo del script, establecemos la variable `FirstWeekDay` en 3.

Script de carga

```
// 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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `week`
- `weekday`

Tabla de resultados

Fecha	semana	weekday
01/01/2021	52	Mié
01/02/2021	1	Jue
01/03/2021	1	Vie
01/04/2021	1	Sáb
01/05/2021	1	Dom
01/06/2020	1	Lun
01/07/2020	1	Mar
01/08/2020	1	Mié
01/09/2020	2	Jue
01/10/2020	2	Vie
01/11/2020	2	Sáb
01/12/2020	2	Dom
01/13/2020	2	Lun
01/14/2020	2	Mar

Como la variable del sistema `FirstWeekDay` está establecida en 3, el primer día de cada semana será el jueves. En la tabla de resultados podemos ver que cada nueva semana comenzará a partir del jueves (2 y 9 de enero).

LongDayNames

El formato definido reemplaza la convención de nombres largos de los días la semana establecida en la configuración regional.

Sintaxis:

`LongDayNames`

El ejemplo siguiente de la función `LongDayNames` define los nombres completos de los días:

```
Set LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';
```

Al modificar la variable, se requiere un punto y coma ; para separar los valores individuales.

La función LongDayNames se puede usar en combinación con la función *Date (page 1248)*, que devuelve DayNames como valores de campo.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia SET DateFormat de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: valor predeterminado de las variables del sistema

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos con fechas, que se cargará en una tabla denominada Transactions.
- Un campo date.
- La definición de LongDayNames predeterminada.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- dayname

Cree esta medida:

```
=sum(amount)
```

Tabla de resultados

date	dayname	=sum(amount)
01/01/2022	Sábado	1000
02/01/2022	Martes	2123
03/01/2022	Martes	4124
04/01/2022	Viernes	2431

En el script de carga, para crear un campo denominado dayname, la función Date se utiliza con el campo date como el primer argumento. El segundo argumento de la función es el formato WWWW.

Usar este formato convierte los valores del primer argumento en el nombre de día completo correspondiente que se establece en la variable LongDayNames. En la tabla de resultados, los valores de campo de nuestro campo creado dayname muestran esto.

Ejemplo 2: cambiar la variable del sistema

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo. Sin embargo, al comienzo del script, la definición de LongDayNames se modifica para usar los días de la semana en español.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `dayname`

Cree esta medida:

```
=sum(amount)
```

Tabla de resultados

date	dayname	=sum(amount)
01/01/2022	Sábado	1000
02/01/2022	Martes	2123
03/01/2022	Martes	4124
04/01/2022	Viernes	2431

En el script de carga, la variable `LongDayNames` se modifica para listar los días de la semana en español.

Después cree un campo denominado `dayname`, que es la función `Date` empleada con el campo `date` como el primer argumento.

El segundo argumento de la función es el formato `WWWW`. Usar este formato Qlik Sense convierte los valores del primer argumento en el nombre de día completo correspondiente establecido en la variable `LongDayNames`.

En la tabla de resultados, los valores de campo de nuestro campo creado `dayname` muestran los días de la semana escritos en español y con sus nombres completos.

LongMonthNames

El formato definido reemplaza la convención de nombres largos de meses establecida en la configuración regional.

Sintaxis:

`LongMonthNames`

Al modificar la variable, se debe utilizar `;` para separar los valores individuales.

El ejemplo siguiente de la función LongMonthNames define los nombres de los meses en su totalidad:

```
Set
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';
```

La función LongMonthNames se utiliza a menudo en combinación con las siguientes funciones:

Funciones relacionadas

Función	Interacción
Date (page 1248)	Función de script que devuelve DayNames como valores de campo.
LongDayNames (page 233)	Valores en forma larga de los nombres de días DayNames.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia SET DateFormat de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: valor predeterminado de las variables del sistema

Script de carga y resultados

General

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada Transactions.
- Un campo date.
- La definición de LongMonthNames predeterminada.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones.

- `date`
- `monthname`

Cree esta medida

```
=sum(amount)
```

Tabla de resultados

date	monthname	sum(amount)
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
01/08/2022	Enero	3554.28
01/09/2022	Enero	3756.17
01/10/2022	Enero	3454.35

Se utiliza la definición de `LongMonthNames` predeterminada. En el script de carga, para crear un campo denominado `month`, la función `Date` se utiliza con el campo `date` como el primer argumento. El segundo argumento de la función es el formato `MMMM`.

Usar este formato Qlik Sense convierte los valores del primer argumento en el nombre de mes completo correspondiente establecido en la variable `LongMonthNames`. En la tabla de resultados, los valores de campo de nuestro campo creado `month` muestran esto.

Ejemplo 2: cambiar la variable del sistema

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Transactions`.
- Un campo `date`.
- La variable `LongMonthNames` que se modifica para usar los días de la semana abreviados en español.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue `sum(amount)` como medida y estos campos como dimensiones:

- `date`
- `monthname`

Cree esta medida:

```
=sum(amount)
```

Tabla de resultados

date	monthname	sum(amount)
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
01/08/2022	Enero	3554.28
01/09/2022	Enero	3756.17
01/10/2022	Enero	3454.35

En el script de carga, la variable `LongMonthNames` se modifica para listar los meses del año en español. Después, para crear un campo denominado `monthname`, se utiliza la función `Date` con el campo `date` como el primer argumento. El segundo argumento de la función es el formato `MMMM`.

Usar este formato Qlik Sense convierte los valores del primer argumento en el nombre de mes completo correspondiente establecido en la variable `LongMonthNames`. En la tabla de resultados, los valores de campo de nuestro campo creado `monthname` muestran esto.

MoneyDecimalSep

El separador decimal definido reemplaza el símbolo decimal de moneda establecido en la configuración regional.

> *De manera predeterminada, Qlik Sense muestra los números y el texto de forma diferente en los gráficos de tablas. Los números se alinean a la derecha y el texto se alinea a la izquierda. Esto facilita la búsqueda de problemas de conversión de texto a número. Cualquier tabla en esta página que muestre resultados de Qlik Sense usará este formato.*

Sintaxis:

```
MoneyDecimalSep
```

Las aplicaciones de Qlik Sense interpretarán los campos de texto que se ajustan a este formato como valores monetarios. El campo de texto debe contener el símbolo de moneda que se define en la variable del sistema `MoneyFormat`. `MoneyDecimalSep` es particularmente útil cuando se manejan fuentes de datos recibidas de múltiples configuraciones regionales diferentes.

El siguiente ejemplo muestra un posible uso de la variable de sistema `MoneyDecimalSep`:

```
Set MoneyDecimalSep='.';
```

Esta función se utiliza a menudo junto con las siguientes funciones:

Funciones relacionadas

Función	Interacción
MoneyFormat	En casos de interpretación de campos de texto, el símbolo MoneyFormat se utilizará como parte de la interpretación. Para el formato de números, Qlik Sense utilizará el formato MoneyFormat en los objetos de gráfico.
MoneyThousandSep	En casos de interpretación de campos de texto, también se debe adherir a la función MoneyThousandSep.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia SET DateFormat de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1 - Notación de separador decimal de punto MoneyDecimalSep (.)

Script de carga y resultados

General

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada Transactions.
- Datos proporcionados que tienen su campo monetario en formato de texto con un punto "." utilizado como separador decimal. Cada registro también lleva el prefijo de un símbolo "$", excepto el último registro, que tiene el prefijo de un símbolo "£".

Tenga en cuenta que la variable de sistema MoneyFormat define el dólar "$" como la moneda predeterminada.

Script de carga

```
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'
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:`amount`.

Agregue las siguientes medidas:

- `isNum(amount)`
- `sum(amount)`

Revise los resultados a continuación, demostrando la interpretación correcta de todos los valores en dólares "$" únicamente.

Tabla de resultados

cantidad	=isNum(amount)	=Sum(amount)
Totales	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	$2814.32

La tabla de resultados anterior muestra cómo el campo `amount` se ha interpretado correctamente para todos los valores prefijados en dólares ($), mientras que el prefijo de la libra `amount` no se ha convertido a un valor monetario.

Ejemplo 2 - Notación de coma (,) MoneyDecimalSep

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Transactions`.
- Datos proporcionados que tienen su campo monetario en formato de texto con una coma "," utilizada como separador decimal. Cada registro también va precedido por un símbolo "$", a excepción del último registro, que utiliza erróneamente el separador decimal de punto ".".

Tenga en cuenta que la variable de sistema `MoneyFormat` define el dólar "$" como la moneda predeterminada.

Script de carga

```
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'
];
```

Resultados

Texto de párrafo para Resultados.

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:`amount`.

Agregue las siguientes medidas:

- `isNum(amount)`
- `sum(amount)`

Revise los resultados a continuación, demostrando la interpretación correcta de todos los valores, excepto en la cantidad en la que el separador decimal utiliza el punto "." como notación. En ese caso, debería haberse utilizado una coma en su lugar.

Tabla de resultados

cantidad	=isNum(amount)	=Sum(amount)
Totales	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	$2814.32

MoneyFormat

Esta variable del sistema define el patrón de formato utilizado por Qlik para la traducción automática de texto a número, donde el número tiene como prefijo un símbolo monetario. También define cómo se mostrarán en los objetos del gráfico las medidas cuyas propiedades de Formato de número están establecidas en "Moneda".

El símbolo definido como parte del patrón de formato en la variable del sistema `MoneyFormat` reemplaza al símbolo de moneda establecido por su configuración regional.

De manera predeterminada, Qlik Sense muestra los números y el texto de forma diferente en los gráficos de tablas. Los números se alinean a la derecha y el texto se alinea a la izquierda. Esto facilita la búsqueda de problemas de conversión de texto a número. Cualquier tabla en esta página que muestre resultados de Qlik Sense usará este formato.

Sintaxis:

`MoneyFormat`

```
Set MoneyFormat='$ #,##0.00; ($ #,##0.00)';
```

Este formato se mostrará en los objetos del gráfico cuando la propiedad de un campo numérico `Number Formatting` se establezca en `Money`. Además, cuando los campos de texto numéricos son interpretados por Qlik Sense, si el símbolo de moneda del campo de texto coincide con el del símbolo definido en la variable `MoneyFormat`, Qlik Sense interpretará este campo como un valor monetario.

Esta función se utiliza a menudo junto con las siguientes funciones:

Funciones relacionadas

Función	Interacción
MoneyDecimalSep (page 240)	Para el formato de números, se utilizará el formato `MoneyDecimalSep` en el formato de los objetos.
MoneyThousandSep (page 248)	Para el formato de números, se utilizará el formato `MoneyThousandSep` en el formato de los objetos.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: MoneyFormat

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene un conjunto de datos que se carga en una tabla denominada Transactions. Se utiliza la definición de `MoneyFormat` predeterminada.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `amount`

Añada esta medida:

```
=Sum(amount)
```

En **Formato numérico**, seleccione **Moneda** para configurar `Sum(amount)` como valor de moneda.

Tabla de resultados

fecha	Amount	=Sum(amount)
Totales		$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
01/08/2022	$35545828255	$35545828255.00
01/09/2022	$37565817436	$37565817436.00
01/10/2022	$3454343566	$3454343566.00

Se utiliza la definición de `MoneyFormat` predeterminada. Esto presenta el siguiente aspecto: `$###0.00;-$###0.00`. En la tabla de resultados, el formato del campo `amount` muestra el símbolo de moneda y se han incluido el punto decimal y los lugares decimales.

Ejemplo 2: MoneyFormat con separador de miles y formatos de entrada mixtos

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de formato de entrada mixto, que se carga en una tabla denominada `Transactions` con separadores de miles y separadores decimales intercalados.
- Se modifica la definición de `MoneyFormat` para incluir una coma como separador de miles.
- Una de las filas de datos delimitada erróneamente con comas separadoras de miles en los lugares incorrectos. Observe cómo esta cantidad se deja como texto y no se puede interpretar como un número.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `amount`

Añada esta medida:

```
=Sum(amount)
```

En **Formato numérico**, seleccione **Moneda** para configurar `Sum(amount)` como valor de moneda.

Tabla de resultados

fecha	Amount	=Sum(amount)
Totales		$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	$24313884663	$24,313,884,663.00
01/07/2022	$545883436	$545,883,436.00
01/08/2022	$35545828255	$35,545,828,255.00
01/09/2022	$37565817436	$37,565,817,436.00
01/10/2022	$3454343566	$3,454,343,566.00

Al comienzo del script, la variable del sistema `MoneyFormat` se modifica para incluir una coma como separador de miles. En la tabla Qlik Sense, se puede ver que el formato incluye este separador. Además, la fila con el separador erróneo no se ha interpretado correctamente y permanece como texto. Por eso no contribuye a la sumatoria del importe.

MoneyThousandSep

El separador de miles definido reemplaza el símbolo de agrupación de dígitos de moneda establecido en la configuración regional.

De manera predeterminada, Qlik Sense muestra los números y el texto de forma diferente en los gráficos de tablas. Los números se alinean a la derecha y el texto se alinea a la izquierda. Esto facilita la búsqueda de problemas de conversión de texto a número. Cualquier tabla en esta página que muestre resultados de Qlik Sense usará este formato.

Sintaxis:

```
MoneyThousandSep
```

Las aplicaciones de Qlik Sense interpretarán los campos de texto que se ajustan a este formato como valores monetarios. El campo de texto debe contener el símbolo de moneda que se define en la variable del sistema `MoneyFormat`. `MoneyThousandSep` es particularmente útil cuando se manejan fuentes de datos recibidas de múltiples configuraciones regionales diferentes.

El siguiente ejemplo muestra un posible uso de la variable de sistema `MoneyThousandSep`:

```
Set MoneyDecimalSep=',';
```

Esta función se utiliza a menudo junto con las siguientes funciones:

Funciones relacionadas

Función	Interacción
`MoneyFormat`	En casos de interpretación de campos de texto, el símbolo `MoneyFormat` se utilizará como parte de la interpretación. Para el formato de números, Qlik Sense utilizará el formato `MoneyFormat` en los objetos de gráfico.
`MoneyDecimalSep`	En casos de interpretación de campos de texto, también se debe adherir a la función `MoneyDecimalSep`.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: notación de coma (,) de MoneyThousandSep

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Transactions`.
- Datos proporcionados que tienen su campo monetario en formato de texto con una coma empleada como separador de miles. Cada registro también va precedido por un símbolo "$".

Tenga en cuenta que la variable de sistema `MoneyFormat` define el dólar "$" como la moneda predeterminada.

Script de carga

```
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'
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:`amount`.

Agregue las siguientes medidas:

- `isNum(amount)`
- `sum(amount)`

Revise los resultados a continuación. La tabla demuestra la interpretación correcta de todos los valores usando la notación coma "," como separador de miles.

El campo `amount` se ha interpretado correctamente para todos los valores, con la excepción de un valor que usaba un punto "." como separador de miles.

Tabla de resultados

cantidad	=isNum(amount)	=Sum(amount)
Totales	0	$161645330590.00

cantidad	=isNum(amount)	=Sum(amount)
$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	$35545828255.00
$37,565,817,436	-1	$37565817436.00

Ejemplo 2: notación de punto (.) de MoneyThousandSep

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Transactions`.
- Datos proporcionados que tienen su campo monetario en el formato de texto con un punto empleado como separador de miles. Cada registro también va precedido por un símbolo "$".

Tenga en cuenta que la variable de sistema `MoneyFormat` define el dólar "$" como la moneda predeterminada.

Script de carga

```
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'
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:`amount`.

Agregue las siguientes medidas:

- `isNum(amount)`
- `sum(amount)`

Revise los resultados a continuación, demostrando la interpretación correcta de todos los valores usando la notación de punto "." como separador de miles.

El campo `amount` se ha interpretado correctamente para todos los valores, con la excepción de un valor que usaba una coma "," como separador de miles.

Tabla de resultados

cantidad	=isNum(amount)	=Sum(amount)
Totales	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
$24.313.884.663	-1	$24313884663.00
$35.545.828.255	-1	$35545828255.00
$37.565.817.436	-1	$37565817436.00

MonthNames

El formato definido reemplaza la convención de nombres de meses establecida en la configuración regional.

Sintaxis:

```
MonthNames
```

Al modificar la variable, se debe utilizar ; para separar los valores individuales.

Ejemplos de funciones

Ejemplo	Resultados
`Set MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';`	Este uso de la función `MonthNames` define los nombres de los meses en inglés y en su forma abreviada.
`Set MonthNames='Enero;Feb;Marzo;Abr;Mayo;Jun;Jul;Agosto;Set;Oct;Nov;Dic';`	Este uso de la función `MonthNames` define los nombres de los meses en español y en su forma abreviada.

La función `MonthNames` se puede utilizar en combinación con las siguientes funciones:

Funciones relacionadas

Función	Interacción
month (page 928)	Función de script que devuelve los valores definidos en `MonthNames` como valores de campo
Date (page 1248)	Función de script que devuelve los valores definidos en `MonthNames` como valores de campo basándose en un argumento de formato proporcionado
LongMonthNames (page 236)	Valores en forma larga de `MonthNames`

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: valor predeterminado de las variables del sistema

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Transactions`.
- Un campo `date`.
- La definición de `MonthNames` predeterminada.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `monthname`

Cree esta medida:

```
=sum(amount)
```

Tabla de resultados

date	monthname	sum(amount)
01/01/2022	Ene	1000.45

date	monthname	sum(amount)
01/02/2022	Ene	2123.34
01/03/2022	Ene	4124.35
01/04/2022	Ene	2431.36
01/05/2022	Ene	4787.78
01/06/2022	Ene	2431.84
01/07/2022	Ene	2854.83
01/08/2022	Ene	3554.28
01/09/2022	Ene	3756.17
01/10/2022	Ene	3454.35

Se utiliza la definición de `MonthNames` predeterminada. En el script de carga, la función `Month` se utiliza con el campo `date` como el argumento proporcionado.

En la tabla de resultados, el resultado de esta función `Month` muestra los meses del año en el formato definido en `MonthNames`.

Ejemplo 2: cambiar la variable del sistema

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Transactions`.
- Un campo `date`.
- La variable `MonthNames` que se modifica para usar los nombres de los meses abreviados en español.

Script de carga

```
                            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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `monthname`

Cree esta medida:

```
=sum(amount)
```

Tabla de resultados

date	monthname	sum(amount)
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
01/08/2022	Enero	3554.28
01/09/2022	Enero	3756.17
01/10/2022	Enero	3454.35

En el script de carga, primero se modifica la variable `MonthNames` para listar los meses del año abreviados en español. La función `Month` se utiliza con el campo `date` como el argumento proporcionado.

En la tabla de resultados, el resultado de esta función `Month` muestra los meses del año en el formato definido en `MonthNames`.

Es importante recordar que si se modifica el idioma de la variable `MonthNames`, como se ha hecho en este ejemplo, la variable `LongMonthNames` todavía contendría los meses del año en inglés. La variable `LongMonthNames` debería modificarse si ambas variables se utilizan en la aplicación.

Ejemplo 3: función de fecha

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Transactions`.
- Un campo `date`.
- La definición de `MonthNames` predeterminada.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `monthname`

Cree esta medida:

```
=sum(amount)
```

Tabla de resultados

date	monthname	sum(amount)
01/01/2022	Ene	1000.45
01/02/2022	Ene	2123.34
01/03/2022	Ene	4124.35
01/04/2022	Ene	2431.36

date	monthname	sum(amount)
01/05/2022	Ene	4787.78
01/06/2022	Ene	2431.84
01/07/2022	Ene	2854.83
01/08/2022	Ene	3554.28
01/09/2022	Ene	3756.17
01/10/2022	Ene	3454.35

Se utiliza la definición de `MonthNames` predeterminada. En el script de carga, la función `Date` se utiliza con el campo `date` como el primer argumento. El segundo argumento es `MMM`.

Usar este formato Qlik Sense convierte los valores del primer argumento en el nombre de mes correspondiente establecido en la variable `MonthNames`. En la tabla de resultados, los valores de campo de nuestro campo creado `month` muestran esto.

NumericalAbbreviation

La abreviatura numérica establece qué abreviatura usar para los prefijos de escala de numerales, por ejemplo M para mega o un millón (10^6) y µ para micro (10^{-6}).

Sintaxis:

```
NumericalAbbreviation
```

Puede configurar la variable `NumericalAbbreviation` como una cadena que contenga una lista de pares de definición de abreviatura, delimitada por punto y coma. Cada par de definición de abreviatura debe contener la escala (el exponente en base decimal) y la abreviatura separada por dos puntos, por ejemplo, `6:M` para un millón.

El valor predeterminado es `'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'`.

Ejemplos:

Esta configuración cambiará el prefijo de mil a t y el prefijo de mil millones a B. Esto sería útil para aplicaciones financieras donde se esperan abreviaturas como t$, M$ y 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

El parámetro define qué día de enero establecer como día de referencia para definir la semana 1. En otras palabras, esta configuración prescribe cuántos días de la semana 1 deben ser fechas de enero.

Sintaxis:

```
ReferenceDay
```

ReferenceDay establece cuántos días se incluyen en la primera semana del año. ReferenceDay se puede establecer en cualquier valor entre 1 y 7. Cualquier valor fuera del rango 1–7 se interpreta como el punto medio de la semana (4), lo que equivale a establecer ReferenceDay en 4.

Si no selecciona un valor para la configuración de ReferenceDay, se mostrará el valor predeterminado ReferenceDay=0, que se interpretará como el punto medio de la semana (4), como se ve en la tabla de valores ReferenceDay, a continuación.

La función ReferenceDay se utiliza a menudo en combinación con las siguientes funciones:

Funciones relacionadas

Variable	Interacción
BrokenWeeks (page 215)	Si la app de Qlik Sense funciona con semanas ininterrumpidas, se aplicará la configuración de la variable ReferenceDay. Sin embargo, si se utilizan semanas divididas, la semana 1 comenzará el 1 de enero y terminará como dicte la configuración de la variable FirstWeekDay e ignorará la etiqueta ReferenceDay.
FirstWeekDay (page 229)	Un entero que define qué día se utilizará como primer día de la semana.

Qlik Sense permite establecer los siguientes valores para ReferenceDay:

Valores de ReferenceDay

Valor	Reference day
0 (por defecto)	4 de enero
1	1 de enero
2	2 de enero
3	3 de enero
4	4 de enero
5	5 de enero
6	6 de enero
7	7 de enero

En el siguiente ejemplo, ReferenceDay = 3 define el 3 de enero como el día de referencia:

```
SET ReferenceDay=3; //(set January 3 as the reference day)
```

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia SET DateFormat de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos:

Si desea configuraciones ISO para semanas y números de semana, asegúrese de tener lo siguiente en el script:

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;    // Jan 4th is always in week 1
```

Si desea la configuración de EE. UU., asegúrese de tener lo siguiente en el script:

```
Set FirstWeekDay=6;
Set BrokenWeeks=1;
Set ReferenceDay=1;     // Jan 1st is always in week 1
```

Ejemplo 1: Script de carga usando el valor predeterminado; ReferenceDay=0

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- La variable `ReferenceDay` que está establecida en `0`.
- La variable `BrokenWeeks` que está establecida en `0` obliga a la app a usar semanas ininterrumpidas.
- Un conjunto de datos de fechas que abarca desde finales de 2019 hasta principios de 2020.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `week`
- `weekday`

Tabla de resultados

date	**semana**	**weekday**
12/27/2019	52	Vie
12/28/2019	52	Sáb
12/29/2019	1	Dom
12/30/2019	1	Lun
12/31/2019	1	Mar
01/01/2020	1	Mié
01/02/2020	1	Jue
01/03/2020	1	Vie
01/04/2020	1	Sáb
01/05/2020	2	Dom
01/06/2020	2	Lun
01/07/2020	2	Mar
01/08/2020	2	Mié
01/09/2020	2	Jue
01/10/2020	2	Vie
01/11/2020	2	Sáb

La semana 52 concluye el sábado 28 de diciembre. Debido a que `ReferenceDay` requiere que el 4 de enero se incluya en la semana 1, la semana 1 comienza el 29 de diciembre y concluye el sábado 4 de enero.

Ejemplo: la variable ReferenceDay establecida en 5

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- La variable `ReferenceDay` que está establecida en `5`.
- La variable `BrokenWeeks` que está establecida en `0` obliga a la app a usar semanas ininterrumpidas.
- Un conjunto de datos de fechas que abarca desde finales de 2019 hasta principios de 2020.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `week`

- weekday

Tabla de resultados

date	semana	weekday
12/27/2019	52	Vie
12/28/2019	52	Sáb
12/29/2019	53	Dom
12/30/2019	53	Lun
12/31/2019	53	Mar
01/01/2020	53	Mié
01/02/2020	53	Jue
01/03/2020	53	Vie
01/04/2020	53	Sáb
01/05/2020	1	Dom
01/06/2020	1	Lun
01/07/2020	1	Mar
01/08/2020	1	Mié
01/09/2020	1	Jue
01/10/2020	1	Vie
01/11/2020	1	Sáb

La semana 52 concluye el sábado 28 de diciembre. La variable `BrokenWeeks` obliga a la aplicación a usar semanas ininterrumpidas. El valor del día de referencia de `5` requiere que el 5 de enero se incluya en la semana 1.

Sin embargo, esto es ocho días después de la conclusión de la semana 52 del año anterior. Por lo tanto, la semana 53 comienza el 29 de diciembre y concluye el 4 de enero. La semana 1 comienza el domingo 5 de enero.

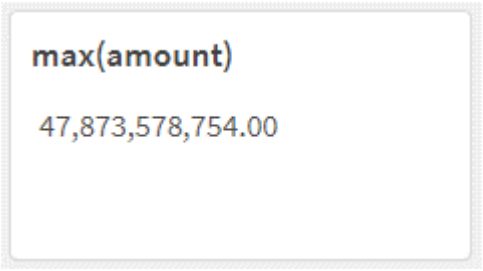
ThousandSep

El separador de miles definido reemplaza al símbolo de agrupación de dígitos del sistema operativo (configuración regional).

Sintaxis:

```
ThousandSep
```

El objeto Qlik Sense utilizando la variable `ThousandSep` *(con el separador de miles)*



Las aplicaciones de Qlik Sense interpretan los campos de texto que se ajustan a este formato como valores numéricos. Este formato se mostrará en los objetos del gráfico cuando la propiedad de Formato numérico de un campo numérico se establezca en **Número**.

`ThousandSep` es útil cuando se manejan fuentes de datos recibidas de múltiples configuraciones regionales.

> *Si la variable* `ThousandSep` *se modifica después de que los objetos ya se hayan creado y formateado en la aplicación, el usuario deberá volver a formatear cada campo relevante deseleccionando y luego volviendo a seleccionar la propiedad de* ***Formato numérico Número****.*

Los ejemplos siguientes muestran posibles usos de la variable de sistema `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)
```

Estos temas le ayudarán a trabajar con esta función:

Temas relacionados

Tema	Descripción
DecimalSep (page 226)	En los casos de interpretación de campos de texto, también se deben respetar los ajustes del separador decimal proporcionados por esta función. Para el formato de números, Qlik Sense utilizará **DecimalSep** cuando sea necesario.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las

fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: variables del sistema predeterminadas

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Transactions`.
- Se utiliza la definición de la variable `ThousandSep` predeterminada.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:`date`.
2. Agregue la siguiente medida:
   `=sum(amount)`
3. En el panel de propiedades, en **Datos**, seleccione la medida.
4. En **Formato numérico**, seleccione **Número**.

Ajustar el formato numérico para una medida de gráfico

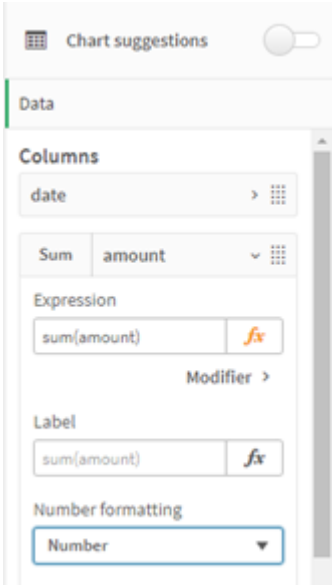


Tabla de resultados

fecha	**=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
01/08/2022	35,545,828,255.00
01/09/2022	37,565,817,436.00
01/10/2022	3,454,343,566.00

En este ejemplo, se utiliza la definición de `ThousandSep` predeterminada, que se establece en formato de coma (","). En la tabla de resultados, el formato del campo de cantidad muestra una coma entre las agrupaciones de miles.

Ejemplo 2: cambiar la variable del sistema

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- El mismo conjunto de datos del primer ejemplo, que se carga en una tabla denominada `Transactions`.
- Modificación de la definición de `ThousandSep`, al inicio del script, para mostrar un carácter "*" como separador de miles. Este es un ejemplo extremo y se utiliza únicamente para demostrar la funcionalidad de la variable.

La modificación empleada en este ejemplo es extrema y no se usa habitualmente, pero se muestra aquí para indicar la funcionalidad de la variable.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:`date`.
2. Agregue la siguiente medida:
   `=sum(amount)`
3. En el panel de propiedades, en **Datos**, seleccione la medida.
4. En **Formato numérico**, seleccione **Personalizado**.

Tabla de resultados

fecha	=sum(amount)
01/01/2022	10*000*000*441.00
01/02/2022	21*237*492*432.00

fecha	=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
01/08/2022	35*545*828*255.00
01/09/2022	37*565*817*436.00
01/10/2022	3*454*343*566.00

Al comienzo del script, la variable del sistema `ThousandSep` se modifica para convertirse en un asterisco "*". En la tabla de resultados, el formato del campo de cantidad muestra un asterisco "*" en las agrupaciones de miles.

Ejemplo 3: interpretación de texto

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Transactions`.
- Datos que tienen su campo numérico en formato de texto con una coma como separador de miles.
- Uso de la variable de sistema `ThousandSep` predefinida.

Script de carga

```
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'
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:`date`.
2. Agregue la siguiente medida:
   `=sum(amount)`
3. En el panel de propiedades, en **Datos**, seleccione la medida.
4. En **Formato numérico**, seleccione **Número**.
5. Agregue la siguiente medida para evaluar si el campo de cantidad es o no un valor numérico:
   `=isnum(amount)`

Tabla de resultados

fecha	=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
01/09/2022	37,565,817,436.00	-1
01/10/2022	3*454*343*566.00	-1

Una vez que los datos se han cargado, podemos ver que Qlik Sense ha interpretado el campo de cantidad como un valor numérico, dado que los datos se ajustan a la variable `ThousandSep`. Esto se demuestra con la función `isnum()`, que evalúa cada entrada a -1 o `TRUE`.

En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.

TimeFormat

El formato definido reemplaza el formato de hora del sistema operativo (configuración regional).

Sintaxis:

```
TimeFormat
```

Ejemplo:

```
Set TimeFormat='hh:mm:ss';
```

TimestampFormat

El formato definido reemplaza los formatos de fecha y hora del sistema operativo (configuración Regional).

Sintaxis:

```
TimestampFormat
```

Ejemplo:

Los ejemplos siguientes utilizan *1983-12-14T13:15:30Z* como datos de indicación de tiempo para mostrar los resultados de diferentes sentencias **SET TimestampFormat**. El formato de fecha empleado es **YYYYMMDD** y el formato de hora es **h:mm:ss TT**. El formato de fecha se especifica en la sentencia **SET DateFormat** y el formato de hora en la sentencia **SET TimeFormat**, en la parte superior del script de carga de datos.

Resultados

Ejemplo	Resultado
`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 1:15:30 PM
`SET TimestampFormat='YYYY-MM-DD hh:mm:ss[.fff] TT';`	1983-12-14 01:15:30

Ejemplos: Script de carga

Ejemplo: Script de carga
En el primer script de carga se utiliza *SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT'*. En el segundo script de carga, el formato de fecha-hora cambia a *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'*. Los diferentes resultados muestran cómo funciona la sentencia **SET TimeFormat** con diferentes formatos de datos de tiempo.

La tabla siguiente muestra el conjunto de datos que se utiliza en los scripts de carga a continuación. La segunda columna de la tabla muestra el formato de cada indicación de tiempo en el conjunto de datos. Las primeras cinco indicaciones de tiempo siguen las reglas ISO 8601 pero la sexta no lo hace.

Conjunto de datos

Tabla que muestra los datos de tiempo utilizados y el formato para cada indicación de tiempo en el conjunto de datos.

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

En el **Editor de carga de datos**, cree una nueva sección y luego agregue el script de ejemplo y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

Script de carga

```
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
];
```

Resultados

Tabla de Qlik Sense que muestra los resultados de la variable de interpretación TimestampFormat utilizada en el script de carga. La última indicación de tiempo en el conjunto de datos no devuelve una fecha correcta.

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	-

El siguiente script de carga utiliza el mismo conjunto de datos. Sin embargo, utiliza *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'* para coincidir con el formato no ISO 8601 de la sexta marca de tiempo.

En el **Editor de carga de datos**, reemplace el script del ejemplo anterior por el que está debajo y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

Script de carga

```
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
];
```

Resultados

Tabla de Qlik Sense que muestra los resultados de la variable de interpretación TimestampFormat utilizada en el script de carga.

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

4.9 Direct Discovery variables

Variables de sistema de Direct Discovery

DirectCacheSeconds

Podemos poner un límite de caché a los resultados de la consulta efectuada a Direct Discovery en gráficos. Una vez que se haya alcanzado este límite, Qlik Sense borrará la caché cuando se hagan nuevas consultas de Direct Discovery. Qlik Sense consulta a la fuente de datos las selecciones efectuadas y crea la caché de nuevo según el límite de tiempo designado. El resultado de cada combinación de selecciones se envía a caché por separado. Es decir, la caché se actualiza con cada selección de manera independiente, de modo que una selección actualiza la caché solo para los campos seleccionados y una segunda selección actualiza la caché en los campos que interesa actualizar. Si la segunda selección incluye campos que se actualizaron en la primera selección, no se actualizarán en la caché de nuevo si no se ha alcanzado el límite de envío a caché.

La caché de Direct Discovery no se aplica a las visualizaciones de **Tabla**. Las selecciones de tabla consultan la fuente de datos constantemente.

El valor límite debe fijarse en segundos. El límite de la caché por defecto es de 1800 segundos (30 minutos).

El valor utilizado para **DirectCacheSeconds** es el valor establecido en el momento en que se ejecuta la sentencia **DIRECT QUERY**. El valor no puede cambiarse en tiempo de ejecución.

Ejemplo:

```
SET DirectCacheSeconds=1800;
```

DirectConnectionMax

Podemos hacer llamadas paralelas, asíncronas, a la base de datos, empleando la función de conexión directa. La sintaxis de script de carga para configurar la capacidad de conexión directa es la siguiente:

```
SET DirectConnectionMax=10;
```

El parámetro numérico especifica el número máximo de conexiones a la base de datos que el código Direct Discovery debería emplear mientras actualiza los objetos de una hoja. El parámetro predeterminado es 1.

Esta variable debería utilizarse con cuidado. Configurarlo en más de 1 se sabe que causa problemas cuando se conecta a Microsoft SQL Server.

DirectUnicodeStrings
Direct Discovery puede admitir la selección de datos Unicode extendidos mediante el uso del formato SQL estándar para literales de cadenas de caracteres extendidos (N'<cadena extendida>'), tal como lo requieren algunas bases de datos (especialmente SQL Server). El uso de esta sintaxis se puede habilitar para Direct Discovery con la variable de script **DirectUnicodeStrings**.

Fijar esta variable en 'true' (verdadero) habilitará el uso del marcador de caracteres “N” del estándar ANSI frente a los literales de cadena. No todas las bases de datos admiten este estándar. El parámetro predeterminado es 'false', 'falso'.

DirectDistinctSupport
Cuando se selecciona el valor de un campo **DIMENSION**en un objeto Qlik Sense, se genera una consulta para la base de datos fuente. Cuando la consulta requiere agrupamiento, Direct Discovery usa la palabra clave **DISTINCT** para seleccionar solo valores únicos. Algunas bases de datos, no obstante, requieren la palabra clave **GROUP BY**. Configure **DirectDistinctSupport** en 'false' para generar **GROUP BY** en vez de **DISTINCT** en consultas de valores únicos.

```
SET DirectDistinctSupport='false';
```
Si se fija DirectDistinctSupported como verdadero, entonces se utiliza **DISTINCT**. Si no se fija, el comportamiento predeterminado supone utilizar **DISTINCT**.

DirectEnableSubquery
En escenarios de tablas múltiples de alta cardinalidad, es posible generar subconsultas en la consulta SQL en lugar de generar una cláusula IN grande. Esto se activa configurando **DirectEnableSubquery** en 'true'. El valor predeterminado es 'false'.

*Cuando **DirectEnableSubquery** está habilitado, no podemos cargar tablas que no estén en modo Direct Discovery.*

```
SET DirectEnableSubquery='true';
```

Variables query banding de Teradata

Teradata query banding es una función que permite a las aplicaciones empresariales colaborar con la base de datos Teradata subyacente a fin de proporcionar una mejor contabilidad, priorización y administración de la carga de trabajo. Utilizando query banding podemos incluir metadatos, como por ejemplo unas credenciales de usuario, en una consulta.

Hay dos variables disponibles, ambas son cadenas que se evalúan y envían a la base de datos.

SQLSessionPrefix

Esta cadena se envía al crear una conexión a la base de datos.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR SESSION;';
```

Si **OSuser()**, por ejemplo, devuelve *WA\sbt*, esto devolverá `SET QUERY_BAND = 'Who=WA\sbt;' FOR SESSION;`, que se envía a la base de datos cuando se crea la conexión.

SQLQueryPrefix

Esta cadena se envía con cada consulta en particular.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR TRANSACTION;';
```

Direct DiscoveryVariables de carácter de

DirectFieldColumnDelimiter

Podemos fijar el carácter utilizado como delimitador de campos en sentencias **Direct Query** para bases de datos que requieran un carácter distinto de la coma como delimitador de campos. El carácter especificado debe ir rodeado de símbolos de entrecomillado simple en la sentencia **SET**.

```
SET DirectFieldColumnDelimiter= '|'
```

DirectStringQuoteChar

Podemos especificar un carácter para utilizarlo como símbolo de entrecomillado en una consulta generada. La opción predefinida es un entrecomillado simple. El carácter especificado debe ir rodeado de símbolos de entrecomillado simple en la sentencia **SET**.

```
SET DirectStringQuoteChar= '"';
```

DirectIdentifierQuoteStyle

Podemos especificar que se utilice entrecomillado no ANSI en los identificadores de las consultas generadas. Hoy por hoy, el único entrecomillado no ANSI disponible es GoogleBQ. El valor predeterminado es ANSI. Se pueden usar mayúsculas, minúsculas y una combinación de mayúsculas y minúsculas (ANSI, ansi, Ansi).

```
SET DirectIdentifierQuoteStyle="GoogleBQ";
```

Por ejemplo, el entrecomillado ANSI se emplea en la sentencia **SELECT** a continuación:

```
SELECT [Quarter] FROM [qvTest].[sales] GROUP BY [Quarter]
```

Cuando **DirectIdentifierQuoteStyle** está fijado en "GoogleBQ", la sentencia **SELECT** utilizará las comillas del siguiente modo:

```
SELECT [Quarter] FROM [qvTest.sales] GROUP BY [Quarter]
```

DirectIdentifierQuoteChar

Podemos especificar un carácter para controlar el entrecomillado de identificadores en una consulta generada. Esto se puede fijar o bien en un carácter (como un símbolo de entrecomillado doble) o dos (como un par de corchetes). La opción predeterminada es un entrecomillado doble.

```
SET DirectIdentifierQuoteChar='[]';
SET DirectIdentifierQuoteChar='``';
SET DirectIdentifierQuoteChar=' ';
```

```
SET DirectIdentifierQuoteChar='""';
```

DirectTableBoxListThreshold
Cuando se utilizan campos Direct Discovery en una visualización de **Tabla**, se establece un umbral o límite para limitar el número de filas mostradas. El límite por defecto es de 1000 registros. El parámetro del límite predefinido se puede modificar configurando la variable **DirectTableBoxListThreshold** en el script de carga. Por ejemplo:

```
SET DirectTableBoxListThreshold=5000;
```
El parámetro del límite se aplica únicamente a visualizaciones de **Tabla** que contengan campos de Direct Discovery. Las visualizaciones de **Tabla** que contienen campos solo en memoria no están limitadas por el parámetro **DirectTableBoxListThreshold**.

No se mostrará ningún campo en la visualización de **Tabla** hasta que la selección contenga menos registros que el límite del umbral.

Variables de interpretación numérica de Direct Discovery

DirectMoneyDecimalSep
El separador decimal definido reemplaza el símbolo decimal de la moneda en la sentencia SQL generada para cargar datos usando Direct Discovery. Este carácter debe coincidir con el carácter utilizado en **DirectMoneyFormat**.

El valor predefinido es `'.'`

Ejemplo:

```
Set DirectMoneyDecimalSep='.';
```

DirectMoneyFormat
El símbolo definido reemplaza el formato de moneda en la sentencia SQL generada para cargar datos usando Direct Discovery. El símbolo de moneda para el separador de miles no debería incluirse.

El valor predefinido es `'#.0000'`

Ejemplo:

```
Set DirectMoneyFormat='#.0000';
```

DirectTimeFormat
El formato de hora definido reemplaza al formato de hora de la sentencia SQL generada para cargar datos usando Direct Discovery.

Ejemplo:

```
Set DirectTimeFormat='hh:mm:ss';
```

DirectDateFormat
El formato de fecha definido reemplaza al formato de fecha de la sentencia SQL generada para cargar datos usando Direct Discovery.

Ejemplo:

```
Set DirectDateFormat='MM/DD/YYYY';
```

DirectTimeStampFormat
El formato definido reemplaza al formato de fecha y hora de la sentencia SQL generada para cargar los datos que utilizan Direct Discovery.

Ejemplo:

```
Set DirectTimestampFormat='M/D/YY hh:mm:ss[.fff]';
```

4.10 Variables de error

Los valores de las cinco variables en total sobrevivirán a la ejecución del script. La primera variable, ErrorMode, es un dato de entrada del usuario y las últimas tres son generadas por Qlik Sense con información sobre los errores en el script.

Descripción general de las variables de error

Cada variable se describe con más detalle después de la descripción general. También puede hacer clic en el nombre de la variable en la sintaxis para acceder inmediatamente a los detalles de esa variable específica.

Consulte la ayuda en línea de Qlik Sense para obtener más detalles sobre la variable.

ErrorMode
Esta variable de error determina qué acción llevará a cabo Qlik Sense cuando se produzca un error durante la ejecución del script.

```
ErrorMode
```

ScriptError
Esta variable de error devuelve el código de error de la última sentencia ejecutada en el script.

```
ScriptError
```

ScriptErrorCount
Esta variable de error devuelve el número total de sentencias que han producido errores durante la actual ejecución de script. Esta variable siempre se pone a 0 al principio de la ejecución del script.

```
ScriptErrorCount
```

ScriptErrorList
Esta variable de error contiene una lista concatenada de todos los errores de script que hayan ocurrido durante la última ejecución de script. Cada error va separado por una línea.

```
ScriptErrorList
```

ErrorMode

Esta variable de error determina qué acción llevará a cabo Qlik Sense cuando se produzca un error durante la ejecución del script.

Sintaxis:

```
ErrorMode
```

Argumentos:

Argumentos

Argumento	Descripción
ErrorMode=1	El parámetro por defecto. La ejecución de script se interrumpirá y se instará al usuario a actuar (en modo no por lotes).
ErrorMode =0	Qlik Sense simplemente ignorará el error y continuará la ejecución del script con la siguiente sentencia de script.
ErrorMode =2	Qlik Sense disparará un mensaje de error "Falló la ejecución de script..." en el momento justo en que se produce el fallo, sin instar al usuario a actuar de antemano.

Ejemplo:

```
set ErrorMode=0;
```

ScriptError

Esta variable de error devuelve el código de error de la última sentencia ejecutada en el script.

Sintaxis:

```
ScriptError
```

Esta variable será reestablecida a 0 tras cada sentencia de script ejecutada correctamente. Si ocurre un error, será configurada según un código de error interno de Qlik Sense. Los códigos de error son valores duales con un componente numérico y otro de texto. Los siguientes códigos existen:

Códigos de error de script

Código de error	Descripción
0	Sin error. El texto de valor dual está vacío.
1	Error general.
2	Error de sintaxis.
3	Error general: ODBC.
4	Error general: OLE DB.

Código de error	Descripción
5	Error general en la base de datos personalizada.
6	Error general: XML.
7	Error general: HTML.
8	No se encontró el archivo.
9	No se encontró la base de datos.
10	No se ha encontrado la tabla.
11	No se encontró el campo.
12	Archivo con formato incorrecto.
16	Error semántico.

Ejemplo:

```
set ErrorMode=0;

LOAD * from abc.qvf;

if ScriptError=8 then

exit script;

//no file;

end if
```

ScriptErrorCount

Esta variable de error devuelve el número total de sentencias que han producido errores durante la actual ejecución de script. Esta variable siempre se pone a 0 al principio de la ejecución del script.

Sintaxis:

```
ScriptErrorCount
```

ScriptErrorList

Esta variable de error contiene una lista concatenada de todos los errores de script que hayan ocurrido durante la última ejecución de script. Cada error va separado por una línea.

Sintaxis:

```
ScriptErrorList
```

5 Expresiones de script

Las expresiones se pueden utilizar tanto en sentencias **LOAD** como en sentencias **SELECT**. La sintaxis y las funciones aquí descritas se aplican a la sentencia **LOAD**, y no a la sentencia **SELECT**, ya que esta última es interpretada por el controlador ODBC y no por Qlik Sense. No obstante, la mayoría de controladores ODBC suelen ser capaces de interpretar una serie de funciones descritas a continuación.

Las expresiones se componen de funciones, campos y operadores, combinados en una sintaxis.

Todas las expresiones de un script de Qlik Sense devuelven un número y/o una cadena, según corresponda. Las funciones lógicas y los operadores devuelven 0 para False y -1 para True. Las conversiones de número a cadena y viceversa están implícitas. Los operadores lógicos y las funciones interpretan 0 como False y todo lo demás como True.

La sintaxis general para una expresión es la siguiente:

Sintaxis general

Expresión	**Campos**	**Operador**
expression ::= (constant	constant	|
expression ::= (constant	fieldref	|
expression ::= (constant	operator1 expression	|
expression ::= (constant	expression operator2 expression	|
expression ::= (constant	function	|
expression ::= (constant	(expression)	)

donde:

- **constant** es una cadena (un texto, una fecha o una hora) entre comillas simples o un número. Las constantes se escriben sin separadores de miles y con un punto decimal como separador decimal.
- **fieldref** es un nombre de campo de la tabla cargada.
- **operator1** es un operador unitario (que funciona en una expresión, la de la derecha).
- **operator2** es un operador binario (que funciona en dos expresiones, una a cada lado).
- **function ::= functionname(parameters)**

- **parameters ::= expression { , expression }**

El número y los tipos de parámetros no son aleatorios. Dependen de la función empleada.

Las expresiones y funciones pueden por tanto anidarse libremente, y siempre y cuando la expresión devuelva un valor interpretable, Qlik Sense no emitirá ningún mensaje de error.

6 Expresiones de gráfico

Una expresión de gráfico (visualización) es una combinación de funciones, campos, operadores matemáticos (+ * / =) y otras medidas. Las expresiones se utilizan para procesar los datos de una app y producir un resultado visible en una visualización. Las expresiones no se limitan al uso en medidas. Podemos crear visualizaciones más potentes y dinámicas, con expresiones para títulos, subtítulos, pies de página e incluso dimensiones.

Esto implica, por ejemplo, que en lugar de que el título de una visualización sea texto estático, puede estar formado por una expresión cuyo resultado varíe en función de las selecciones realizadas.

Para obtener información más detallada sobre las funciones de script y las funciones de gráfico, consulte la ayuda online de Sintaxis de script y funciones de gráficos.

6.1 Definir el ámbito de agregación

Normalmente, hay dos factores que determinan los registros que se utilizan para definir el valor de agregación de una expresión. Cuando se trabaja con visualizaciones, estos factores son los siguientes:

- El valor de dimensión (si la agregación se realiza en una expresión de gráfico)
- Las selecciones

Juntos, estos factores definen el ámbito de la agregación. Pueden producirse situaciones en las que le interese que en el cálculo se omita la selección, las dimensiones o ambas. En las funciones de gráfico puede lograr eso utilizando el cualificador TOTAL, el análisis de conjuntos o una combinación de ambos.

Agregación: Método y descripción

Método	Descripción
El cualificador TOTAL	Utilizar el cualificador total en la función de agregación, ignora el valor de dimensión. La agregación se realizará en todos los valores de campo posibles. El cualificador **TOTAL** puede ir seguido de una lista de uno o más nombres de campo entre paréntesis angulares. Estos nombres de campo deberían ser un subgrupo de las variables de dimensión del gráfico. En este caso, el cálculo se realiza ignorando todas las variables de dimensión del gráfico excepto las listadas, es decir, que devolverá un valor por cada combinación de valores de campo de los campos de dimensión listados. También los campos que no constituyan actualmente una dimensión de un gráfico pueden incluirse en la lista. Esto puede resultar útil en el caso de dimensiones de grupo, en las que los campos de dimensión no son fijos. Listar todas las variables del grupo hará que la función opere correctamente cuando el nivel jerárquico varíe.
Análisis de conjuntos	Si utiliza el análisis de conjuntos dentro de su agregación, se ignora la selección. La agregación se realizará en todos los valores repartidos por las dimensiones.
Cualificador TOTAL y análisis de conjuntos	Utilizar el cualificador **TOTAL** y el análisis de conjuntos dentro de su agregación, hace que se ignore la selección y se descarten las dimensiones.

El cualificador ALL	Utilizar el cualificador **ALL** dentro de su agregación descarta la selección y las dimensiones. Se puede lograr el equivalente con la sentencia de análisis de conjuntos {1} y el cualificador **TOTAL** :
	`=sum(All Sales)`
	`=sum({1} Total Sales)`

Ejemplo: Cualificador TOTAL

En el ejemplo siguiente se muestra cómo se puede utilizar TOTAL para calcular una proporción relativa. Suponiendo que se ha seleccionado Q2, usar TOTAL calcula la suma de todos los valores omitiendo las dimensiones.

Ejemplo: Cualificador 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%

*Para mostrar los números como porcentaje, en el panel de propiedades, en la medida que desea que se muestre como valor de porcentaje, en **Formato numérico**, seleccione **Número** y, desde **Formato**, elija **Simple** junto con uno de los formatos de %.*

Ejemplo: Análisis de conjuntos

En el ejemplo siguiente se muestra cómo se puede utilizar el análisis de conjuntos para comparar conjuntos de datos antes de efectuar cualquier selección. Suponiendo que se ha seleccionado Q2, usar el análisis de conjuntos con la definición del conjunto {1} calcula la suma de todos los valores omitiendo cualquier selección pero dividida por las dimensiones.

Ejemplo: Análisis de conjuntos

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%

Ejemplo: Cualificador TOTAL y análisis de conjuntos

El ejemplo siguiente muestra cómo se pueden combinar el análisis de conjuntos y el cualificador TOTAL para comparar conjuntos de datos antes de efectuar cualquier selección y en todas las dimensiones. Suponiendo que se haya seleccionado Q2, usar el análisis de conjuntos con la definición del conjunto {1} y el cualificador TOTAL calcula la suma de todos los valores descartando cualquier selección y descartando las dimensiones.

Ejemplo: Cualificador TOTAL y análisis de conjuntos

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%

Datos utilizados en los ejemplos:

```
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 Análisis de conjuntos

Cuando realiza una selección en una app, define un subconjunto de registros en los datos. Las funciones de agregación, como `Sum()`, `Max()`, `Min()`, `Avg()` y `Count()` se calculan basándose en este subconjunto.

En otras palabras, su selección define el ámbito de la agregación; define el conjunto de registros sobre los que se realizan los cálculos.

El análisis de conjuntos ofrece una forma de definir un ámbito que es diferente del conjunto de registros definido por la selección actual. Este nuevo ámbito también puede considerarse como una selección alternativa.

Esto puede resultar útil si desea comparar la selección actual con un valor particular, por ejemplo, el valor del año pasado o la cuota de mercado global.

Expresiones de conjunto

Las expresiones de conjunto se pueden utilizar dentro y fuera de las funciones de agregación y van encerradas entre llaves.

Ejemplo: Expresión de conjunto interna

```
Sum( {$<Year={2021}>} Sales )
```

Ejemplo: Expresión de conjunto externa

```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

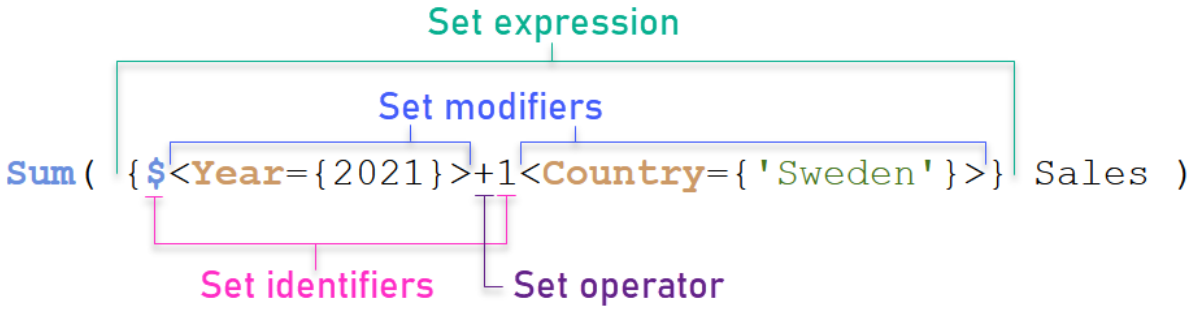
Una expresión de conjunto consta de una combinación de los siguientes elementos:

- **Identificadores**. Un identificador de conjunto representa una selección, definida en otro lugar. También representa un conjunto específico de registros en los datos. Podría ser la selección actual, una selección de un marcador o una selección de un estado alternativo. Una expresión de conjunto simple consiste en un identificador único y exclusivo, como el signo dólar, {$}, que significa todos los registros en la selección actual.
  Ejemplos: `$`, `1`, `BookMark1`, `State2`
- **Operadores**. Un operador de conjunto se puede utilizar para crear uniones, diferencias o intersecciones entre diferentes identificadores de conjunto. De esta forma, puede crear un subconjunto o un superconjunto de las selecciones definidas por los identificadores del conjunto.
  Ejemplos: `+`, `-`, `*`, `/`

- **Modificadores**. Se puede agregar un modificador de conjunto al identificador de conjunto para cambiar su selección. Un modificador también se puede usar por sí solo y luego modificará el identificador predeterminado. Un modificador debe ir incluido entre corchetes angulares <…>. Ejemplos:`<Year={2020}><Supplier={ACME}>`

Los elementos se combinan para formar expresiones de conjunto.

Elementos en una expresión de conjunto



La expresión de conjunto anterior, por ejemplo, se construye a partir de la agregación `Sum(Sales)`.

El primer operando devuelve las ventas del año `2021` de la selección actual, lo cual se indica mediante el identificador de conjunto `$` y el modificador que contiene la selección del año `2021`. El segundo operando devuelve las ventas `Sales` de Suecia: `Sweden` e ignora la selección actual, lo cual viene indicado por el identificador de conjunto `1`.

Por último, la expresión devuelve un conjunto que se compone de los registros que pertenecen a cualquiera de los dos operandos de conjunto, tal como lo indica el operador de conjunto `+`.

Ejemplos

Los ejemplos que combinan los elementos de la expresión de conjunto anterior están disponibles en los siguientes temas:

Conjuntos naturales

Por lo general, una expresión de conjunto representa tanto un conjunto de registros en el modelo de datos como una selección que define este subconjunto de datos. En este caso, el conjunto se denomina conjunto natural.

Los identificadores de conjuntos, con o sin modificadores de conjuntos, siempre representan conjuntos naturales.

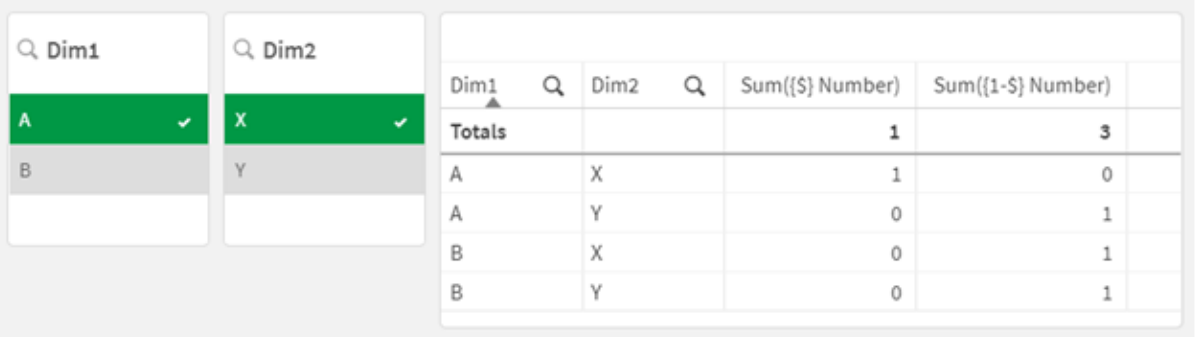
Sin embargo, una expresión de conjunto que utiliza operadores de conjunto también representa un subconjunto de los registros, pero por lo general aún no se puede describir mediante una selección de valores de campo. Tal expresión no es un conjunto natural.

Por ejemplo, el conjunto dado por `{1-$}` no siempre puede definirse mediante una selección. Por lo tanto, no es un conjunto natural. Esto se puede mostrar cargando los siguientes datos, agregándolos a una tabla y luego haciendo selecciones usando paneles de filtrado.

```
Load * Inline
[Dim1, Dim2, Number
A, X, 1
A, Y, 1
B, X, 1
B, Y, 1];
```

Al hacer selecciones para `Dim1` y `Dim2`, obtiene la vista que se muestra en la siguiente tabla.

Tabla con conjuntos naturales y no naturales.



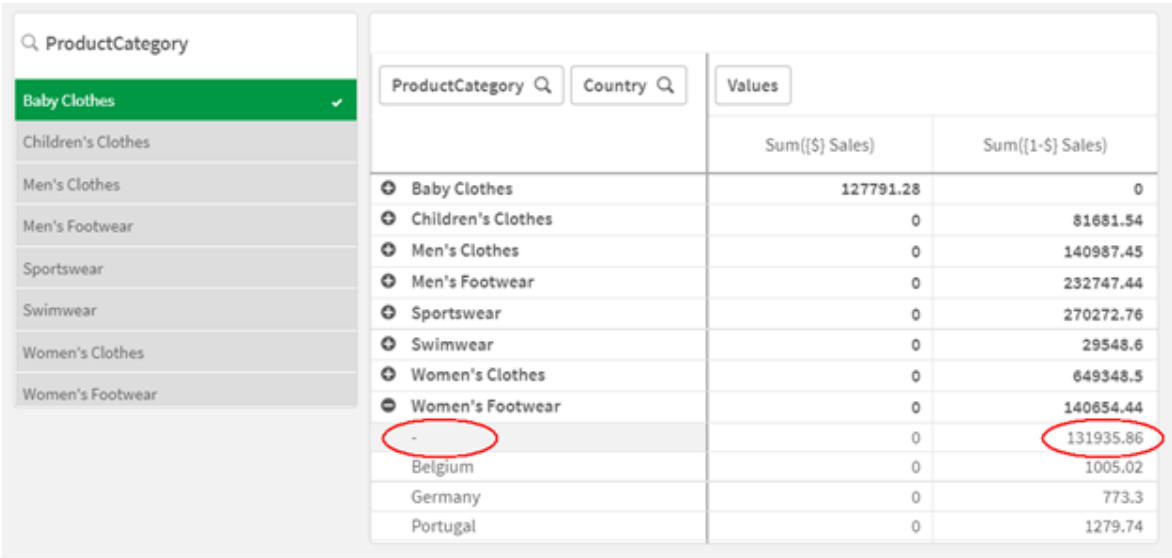
La expresión de conjunto en la primera medida utiliza un conjunto natural: corresponde a la selección que se hace `{$}`.

La segunda medida es diferente. Utiliza `{1-$}`. No es posible hacer una selección que corresponda a este conjunto, por lo que es un conjunto no natural.

Esta distinción tiene varias consecuencias:

- Los modificadores de conjuntos solo se pueden aplicar a identificadores de conjuntos. No se pueden aplicar a una expresión de conjunto arbitraria. Por ejemplo, no es posible utilizar una expresión de conjunto como:
  `{ (BM01 * BM02) <Field={x,y}> }`
  Aquí, los paréntesis normales (redondeados) implican que la intersección entre `BM01` y `BM02` debe evaluarse antes de aplicar el modificador de conjunto. La razón es que no hay ningún conjunto de elementos que se pueda modificar.
- No puede usar conjuntos no naturales dentro de las funciones de elementos `P()` y `E()`. Estas funciones devuelven un conjunto de elementos, pero no es posible deducir el conjunto de elementos de un conjunto no natural.
- Una medida que utiliza un conjunto no natural no siempre se puede atribuir al valor dimensional correcto si el modelo de datos tiene muchas tablas. Por ejemplo, en el siguiente gráfico, algunas cifras de ventas excluidas se atribuyen a Country, mientras otras tienen NULL como Country.

Gráfico con un conjunto no natural



El hecho de que la asignación se realice correctamente o no depende del modelo de datos. En este caso, el número no se puede asignar si pertenece a un país que está excluido por la selección.

Identificador	Descripción
1	Representa el conjunto completo de todos los registros de la aplicación, independientemente de las selecciones realizadas.
$	Representa los registros de la selección actual. La expresión de conjunto **{$}** es, por lo tanto, equivalente a no indicar una expresión de conjunto.
$1	Representa la selección anterior. $2 representa la selección anterior menos una, y así sucesivamente.
$_1	Representa la selección siguiente (avance). $_2 representa la siguiente selección menos una, y así sucesivamente.
BM01	Puede utilizar cualquier ID o nombre de marcador.
MyAltState	Puede referirse a las selecciones efectuadas en un estado alterno por su nombre de estado.

Ejemplo	Resultado
sum ({1} Sales)	Devuelve el total de ventas de la app, descartando las selecciones pero no la dimensión.
sum ({$} Sales)	Devuelve las ventas para la selección actual, es decir, lo mismo que sum(Sales).
sum ({$1} Sales)	Devuelve las ventas de la selección anterior.
sum ({BM01} Sales)	Devuelve las ventas para el marcador llamado *BM01*.

Ejemplo	Resultado
sum({$<OrderDate = DeliveryDate>} Sales)	Devuelve las ventas de la selección actual donde OrderDate = DeliveryDate.
sum({1<Region = {US}>} Sales)	Devuelve las ventas de la región USA, descartando la selección actual.
sum({$<Region = >} Sales)	Devuelve las ventas de la selección, pero con la selección de *Region* eliminada.
sum({<Region = >} Sales)	Devuelve lo mismo que el ejemplo anterior. Cuando se omite el conjunto para modificar, se supone $.
sum({$<Year={2000}, Region={“U*”}>} Sales)	Devuelve las ventas de la selección actual, pero con nuevas selecciones tanto en *Year* como en *Region*.

Identificadores de conjunto

Un identificador de conjunto representa un conjunto de registros en los datos; ya sea todos los datos o un subconjunto de los datos. Es el conjunto de registros definidos por una selección. Podría ser la selección actual, todos los datos (sin selección), una selección de un marcador o una selección de un estado alternativo.

En el ejemplo `Sum( {$<Year = {2009}>} Sales )`, el identificador es el signo dólar: `$`. Esto representa la selección actual. También representa todos los registros posibles. Luego, este conjunto puede ser alterado por la parte modificadora de la expresión del conjunto: se agrega la selección `2009` en `Year`.

En una expresión de conjunto más compleja, se pueden usar dos identificadores junto con un operador para formar una unión, una diferencia o una intersección de los dos conjuntos de registros.

La tabla siguiente muestra algunos identificadores comunes.

Ejemplos con identificadores comunes

Identificador	Descripción
`1`	Representa el conjunto completo de todos los registros de la aplicación, independientemente de las selecciones realizadas.
`$`	Representa los registros de la selección actual en el estado predeterminado. La expresión de conjunto `{$}` por lo tanto, suele ser el equivalente a no indicar una expresión de conjunto.
`$1`	Representa la selección anterior en el estado predeterminado. `$2` representa la selección anterior, menos uno, y así sucesivamente.
`$_1`	Representa la siguiente selección (hacia adelante). `$_2` representa la siguiente selección, menos una, y así sucesivamente.
`BM01`	Puede utilizar cualquier ID o nombre de marcador.
`AltState`	Puede hacer referencia a un estado alterno por su nombre de estado.
`AltState::BM01`	Un marcador contiene las selecciones de todos los estados y puede hacer referencia a un marcador específico calificando el nombre del marcador.

La siguiente tabla muestra ejemplos con diferentes identificadores.

Ejemplos con distintos identificadores

Ejemplo	Resultado
`Sum ({1} Sales)`	Devuelve el total de ventas de la app, descartando las selecciones pero no la dimensión.
`Sum ({$} Sales)`	Devuelve las ventas para la selección actual, es decir, lo mismo que `Sum(Sales)`.
`Sum ({$1} Sales)`	Devuelve las ventas de la selección anterior.
`Sum ({BM01} Sales)`	Devuelve las ventas para el marcador llamado `BM01`.

Operadores de conjunto

Los operadores de conjuntos se utilizan para incluir, excluir o intersectar conjuntos de datos. Todos los operadores emplean conjuntos como operandos y devuelven un conjunto como resultado.

Puede utilizar operadores de conjuntos en dos situaciones diferentes:

- Para realizar una operación de conjunto en identificadores de conjunto, que representan conjuntos de registros en datos.
- Para realizar una operación de conjunto en los conjuntos de elementos, en los valores de campo o dentro de un modificador de conjunto.

La tabla siguiente muestra los operadores que se pueden usar en expresiones de conjuntos.

Operadores

Operador	Descripción
+	Unión. Esta operación binaria devuelve un conjunto compuesto por los registros o elementos que pertenecen a cualquiera de los dos operandos de conjunto.
-	Exclusión. Esta operación binaria devuelve un conjunto formado por los registros o elementos que pertenecen al primero, pero no al segundo, de los operandos de conjunto. Además, cuando se emplee como operador unario, devuelve el otro conjunto complementario.
*	Intersección. Esta operación binaria devuelve un conjunto compuesto por los registros o elementos que pertenecen a los dos operandos de conjunto.
/	Diferencia simétrica `(XOR)`. Esta operación binaria devuelve un conjunto formado por los registros o elementos que pertenecen a cualquiera de los dos operandos del conjunto, pero no a ambos.

La tabla siguiente muestra ejemplos con operadores.

Ejemplos con operadores

Ejemplo	Resultado
`Sum ({1-$} Sales)`	Devuelve las ventas de todo lo excluido por la selección actual.
`Sum ({$*BM01} Sales)`	Devuelve las ventas de la intersección entre la selección y el marcador #160;`BM01`.
`Sum ({-($+BM01)} Sales)`	Devuelve las ventas excluidas por la selección y el marcador `BM01`.
`Sum ({$<Year={2009}>+1<Country={'Sweden'}>} Sales)`	Devuelve las ventas del año 2009 asociadas con las selecciones actuales y agrega el conjunto completo de datos asociados con el país `Sweden` en todos los años.
`Sum ({$<Country={"S*"}+{"*land"}>} Sales)`	Devuelve las ventas de los países que comienzan por `S` o terminan por `land`.

Modificadores de conjunto

Las expresiones de conjunto se usan para definir el ámbito de un cálculo. La parte central de la expresión de conjunto es el modificador de conjunto que especifica una selección. Este se utiliza para modificar la selección del usuario, o la selección en el identificador de conjunto, y el resultado define un nuevo ámbito para el cálculo.

El modificador de conjunto consta de uno o varios nombres de campo, cada uno de ellos seguido por una selección que debería efectuarse en el campo. El modificador va entre paréntesis angulares: `< >`

Por ejemplo:

- `Sum ( {$<Year = {2015}>} Sales )`
- `Count ( {1<Country = {Germany}>} distinct OrderID )`
- `Sum ( {$<Year = {2015}, Country = {Germany}>} Sales )`

Conjuntos de elementos

Un conjunto de elementos se puede definir utilizando lo siguiente:

- Una lista de valores
- Una búsqueda
- Una referencia a otro campo
- Una función establecida

Si se omite la definición del conjunto de elementos, el modificador de conjunto borrará cualquier selección en este campo. Por ejemplo:

```
Sum( {$<Year = >} Sales )
```

Ejemplos: Expresiones de gráfico para modificadores de conjunto basados en conjuntos de elementos

Ejemplos: expresiones de gráfico

Script de carga

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear los ejemplos de expresión del gráfico a continuación.

```
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
];
```

Expresiones de gráfico

Cree una tabla en una hoja de Qlik Sense con las siguientes expresiones de gráfico.

Tabla: Modificadores de conjunto basados en conjuntos de elementos

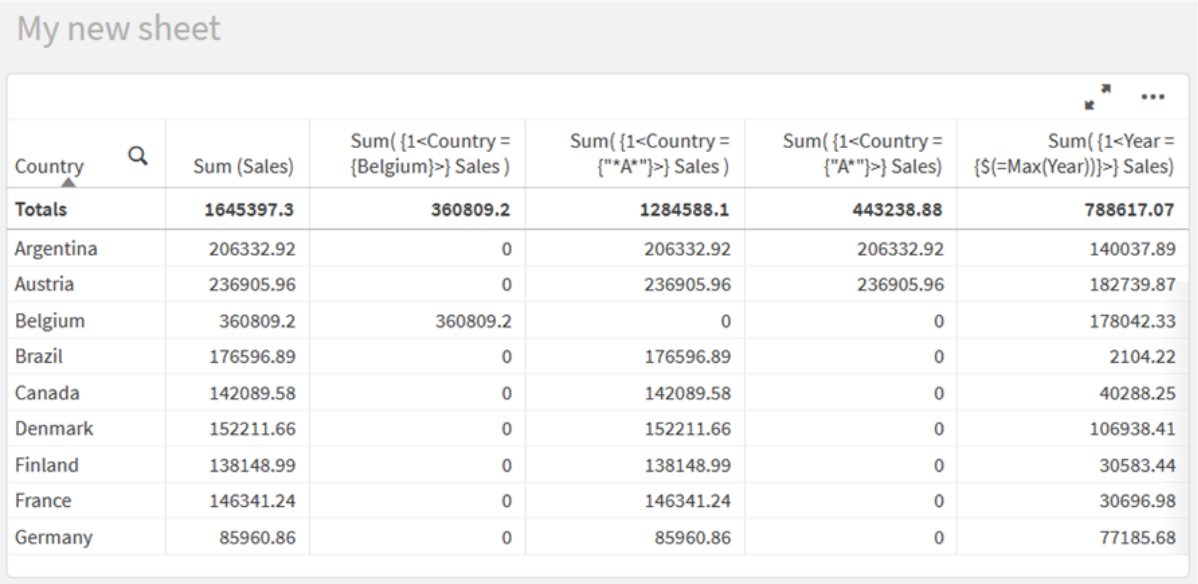
País	**Sum(Sales)**	**Sum ({1<Country= {Belgium}>} Sales)**	**Sum ({1<Country= {"*A*"}>} Sales)**	**Sum ({1<Country= {"A*"}>} Sales)**	**Sum({1<Year= {$(=Max (Year))}>} Sales)**
Totales	1645397.3	360809.2	1284588.1	443238.88	788617.07
Argentina	206332.92	0	206332.92	206332.92	140037.89
Austria	236905.96	0	236905.96	236905.96	182739.87
Bélgica	360809.2	360809.2	0	0	178042.33

País	Sum(Sales)	Sum ({1<Country= {Belgium}>} Sales)	Sum ({1<Country= {"*A*"}>} Sales)	Sum ({1<Country= {"A*"}>} Sales)	Sum({1<Year= {$(=Max (Year))}>} Sales)
Brasil	176596.89	0	176596.89	0	2104.22
Canadá	142089.58	0	142089.58	0	40288.25
Dinamarca	152211.66	0	152211.66	0	106938.41
Finlandia	138148.99	0	138148.99	0	30583.44
Francia	146341.24	0	146341.24	0	30696.98
Alemania	85960.86	0	85960.86	0	77185.68

Explicación

- Dimensiones:
  - `Country`
- Medidas:
  - `Sum(Sales)`
    Suma de `Sales` sin una expresión de conjunto.
  - `Sum({1<Country={Belgium}>}Sales)`
    Seleccione `Belgium` y luego sume las correspondientes ventas `Sales`.
  - `Sum({1<Country={"*A*"}>}Sales)`
    Seleccione todos los países que tienen una `A` y luego sume las correspondientes ventas `Sales`.
  - `Sum({1<Country={"A*"}>}Sales)`
    Seleccione todos los países que comiencen por una `A` y luego sume las correspondientes ventas `Sales`.
  - `Sum({1<Year={$(=Max(Year))}>}Sales)`
    Calcule el `Max(Year)`, que es `2015`, y luego sume las correspondientes `Sales`.

Modificadores de conjunto basados en conjuntos de elementos


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

Valores listados

El ejemplo más común de un conjunto de elementos es uno que se basa en una lista de valores de campo encerrados entre corchetes. Por ejemplo:

- `{$<Country = {Canada, Germany, Singapore}>}`
- `{$<Year = {2015, 2016}>}`

Las llaves interiores definen el conjunto de elementos. Los valores individuales están separados por comas.

Comillas y distinción entre mayúsculas y minúsculas

Si los valores contienen espacios en blanco o caracteres especiales, los valores deben estar entrecomillados. Las comillas simples serán una coincidencia literal que distinga entre mayúsculas y minúsculas con un solo valor de campo. Las comillas dobles implican una coincidencia que no distingue entre mayúsculas y minúsculas con uno o varios valores de campo. Por ejemplo:

- `<Country = {'New Zealand'}>`
  Solo coincide con `New Zealand`.
- `<Country = {"New Zealand"}>`
  Solo coincide con `New Zealand`, `NEW ZEALAND` y `new zealand`.

Las fechas deben ir entre comillas y utilizar el formato de fecha del campo en cuestión. Por ejemplo:

- `<ISO_Date = {'2021-12-31'}>`
- `<US_Date = {'12/31/2021'}>`
- `<UK_Date = {'31/12/2021'}>`

Las comillas dobles pueden sustituirse por corchetes o por comillas oblicuas.

Búsquedas

Los conjuntos de elementos también se pueden crear mediante búsquedas. Por ejemplo:

- `<Country = {"C*"}>`
- `<Ingredient = {"*garlic*"}>`
- `<Year = {">2015"}>`
- `<Date = {">12/31/2015"}>`

Los comodines se pueden utilizar en búsquedas de texto: Un asterisco (*) representa cualquier número de caracteres y un signo de interrogación (?) representa un solo carácter. Los operadores relacionales se pueden utilizar para definir búsquedas numéricas.

Siempre debe utilizar comillas dobles para las búsquedas. Las búsquedas no distinguen entre mayúsculas y minúsculas.

Expansiones de signo dólar

Las expansiones de signo dólar son necesarias si desea usar un cálculo dentro del conjunto de elementos. Por ejemplo, si desea ver solo el último año posible, puede usar:

```
<Year = {$(=Max(Year))}>
```

Valores seleccionados en otros campos

Los modificadores pueden basarse en los valores seleccionados de otro campo. Por ejemplo:

```
<OrderDate = DeliveryDate>
```

Este modificador tomará los valores seleccionados de `DeliveryDate` y los aplicará como una selección en `OrderDate`. Si hay muchos valores distintos, más de varios cientos, entonces esta operación consume mucha CPU y debería evitarse.

Funciones del conjunto de elementos

Los elementos se pueden basar también en las funciones de conjunto `P()` (valores posibles) y `E()` (valores excluidos).

Por ejemplo, si desea seleccionar los países en los que se ha vendido el producto `Cap`, puede utilizar:

```
<Country = P({1<Product={Cap}>} Country)>
```

Del mismo modo, si desea seleccionar los países en los que el producto `Cap` no se ha vendido, puede utilizar:

```
<Country = E({1<Product={Cap}>} Country)>
```

Modificadores de conjunto con búsquedas

Puede crear conjuntos de elementos mediante búsquedas con modificadores de conjunto.

Por ejemplo:

- `<Country = {"C*"}>`
- `<Year = {">2015"}>`
- `<Ingredient = {"*garlic*"}>`

Las búsquedas siempre deben incluirse entre comillas dobles, corchetes o comillas oblicuas. Puede utilizar una lista con una mezcla de cadenas literales (comillas simples) y búsquedas (comillas dobles). Por ejemplo:

```
<Product = {'Nut', "*Bolt", Washer}>
```

Búsquedas de texto

Se pueden utilizar comodines y otros símbolos en las búsquedas de texto:

- Un asterisco (*) representará cualquier número de caracteres.
- Un signo de interrogación (?) representará un solo carácter.
- Un acento circunflejo (^) marcará el comienzo de una palabra.

Por ejemplo:

- `<Country = {"C*", "*land"}>`
  Dame todos los países que comienzan por `C` o terminan en `land`.
- `<Country = {"*^z*"}>`
  Esto devolverá todos los países que tengan una palabra que comience por `z`, por ejemplo, `New Zealand`.

Búsquedas numéricas

Puede realizar búsquedas numéricas utilizando estos operadores relacionales: >, >=, <, <=

Una búsqueda numérica siempre comienza con uno de estos operadores. Por ejemplo:

- `<Year = {">2015"}>`
  Dame 2016 y los años posteriores.
- `<Date = {">=1/1/2015<1/1/2016"}>`
  Dame todas las fechas durante 2015. Tenga en cuenta la sintaxis para describir un intervalo de tiempo entre dos fechas. El formato de fecha debe coincidir con el formato de fecha del campo en cuestión.

Búsquedas de expresiones

Puede utilizar búsquedas de expresiones para realizar búsquedas más avanzadas. Después se evalúa una agregación para cada valor de campo en el campo de búsqueda. Se seleccionan todos los valores para los que la expresión de búsqueda devuelve verdadero.

Una búsqueda de expresiones siempre comienza con un signo igual: =

Por ejemplo:

```
<Customer = {"=Sum(Sales)>1000"}>
```

Esto devolverá todos los clientes con un valor de ventas superior a 1000. `Sum(Sales)` se calcula sobre la selección actual. Esto significa que si tiene una selección en otro campo, por ejemplo, en el campo `Product`, obtendrá los clientes que cumplieron la condición de venta para los productos seleccionados únicamente.

Si desea que la condición sea independiente de la selección, debe usar el análisis de conjuntos dentro de la cadena de búsqueda. Por ejemplo:

```
<Customer = {"=Sum({1} Sales)>1000"}>
```

Las expresiones después del signo igual se interpretarán como un valor booleano. Esto significa que si se evalúa como otra cosa, cualquier número distinto de cero se interpretará como verdadero, mientras que el cero y las cadenas de texto se interpretarán como falso.

Comillas

Utilice comillas cuando las cadenas de búsqueda contengan espacios en blanco o caracteres especiales. Las comillas simples implican una coincidencia literal, que distingue entre mayúsculas y minúsculas, con un solo valor de campo. Las comillas dobles implican una búsqueda que no distingue entre mayúsculas y minúsculas y que potencialmente coincide con varios valores de campo.

Por ejemplo:

- `<Country = {'New Zealand'}>`
  Dame solo `New Zealand`.
- `<Country = {"New Zealand"}>`
  Dame `New Zealand`, `NEW ZEALAND` y `new zealand`

Las comillas dobles pueden sustituirse por corchetes o por comillas oblicuas.

> *En versiones anteriores de Qlik Sense, no había distinción entre comillas simples y comillas dobles y todas las cadenas entre comillas se trataban como búsquedas. Para mantener la compatibilidad con versiones anteriores, las apps creadas con versiones anteriores de Qlik Sense continuarán funcionando como lo hicieron en versiones anteriores. Las apps creadas con Qlik Sense de noviembre de 2017 o posterior respetarán la diferencia entre los dos tipos de comillas.*

Ejemplos: Expresiones de gráfico para modificadores de conjunto con búsquedas

Ejemplos: expresiones de gráfico

Script de carga

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear los ejemplos de expresión del gráfico a continuación.

```
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];
```

Ejemplo 1: Expresiones de gráfico con búsquedas de texto

Cree una tabla en una hoja de Qlik Sense con las siguientes expresiones de gráfico.

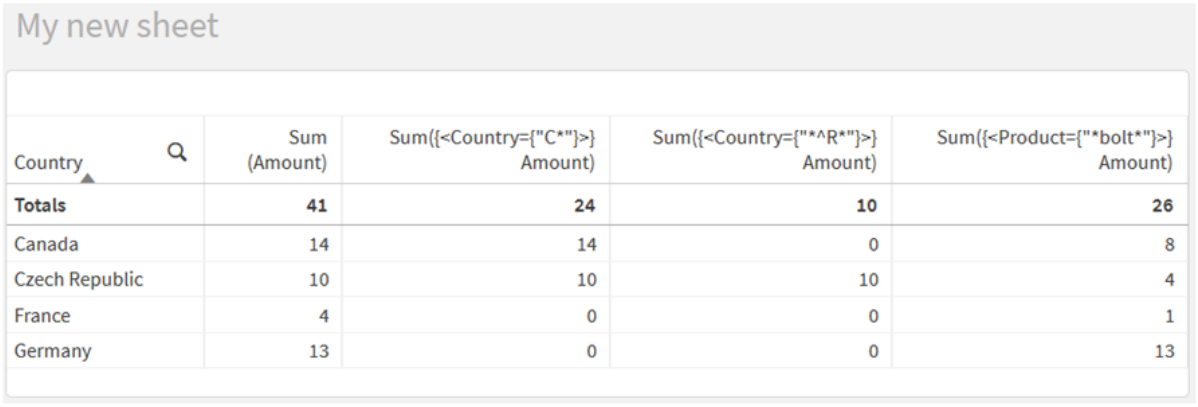
Tabla: Modificadores de conjunto con búsquedas de texto

País	Sum (Amount)	Sum({<Country={"C*"}>} Amount)	Sum({<Country={"*^R*"}>} Amount)	Sum({<Product={"*bolt*"}>} Amount)
Totales	**41**	**24**	**10**	**26**
Canadá	14	14	0	8
República Checa	10	10	10	4
Francia	4	0	0	1
Alemania	13	0	0	13

Explicación

- Dimensiones:
  - `Country`
- Medidas:
  - `Sum(Amount)`
    Suma de `Amount` sin una expresión de conjunto.
  - `Sum({<Country={"C*"}>}Amount)`
    Sum `Amount` de todos los países que comienzan por `C`, como `Canada` y `Czech Republic`.
  - `Sum({<Country={"*^R*"}>}Amount)`
    Sum `Amount` de todos los países que tienen una palabra que comienza por `R`, como `Czech Republic`.
  - `Sum({<Product={"*bolt*"}>}Amount)`
    Sum `Amount` de todos los productos que contienen la cadena de texto `bolt`, como `Bolt` y `Anchor bolt`.

Modificadores de conjunto con búsquedas de texto



Country	Sum (Amount)	Sum({<Country={"C*"}>} Amount)	Sum({<Country={"*^R*"}>} Amount)	Sum({<Product={"*bolt*"}>} Amount)
Totals	**41**	**24**	**10**	**26**
Canada	14	14	0	8
Czech Republic	10	10	10	4
France	4	0	0	1
Germany	13	0	0	13

Ejemplo 2: Expresiones de gráfico con búsquedas numéricas

Cree una tabla en una hoja de Qlik Sense con las siguientes expresiones de gráfico.

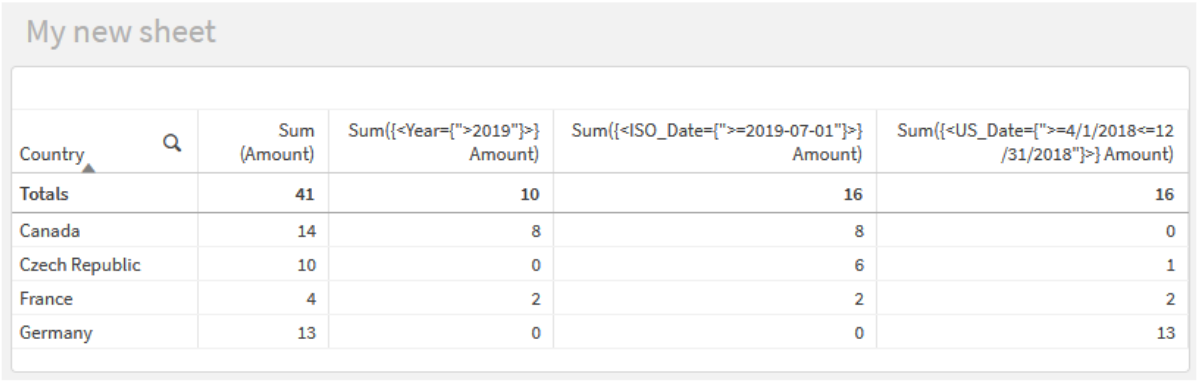
Tabla: Modificadores de conjunto con búsquedas numéricas

País	Sum (Amount)	Sum({<Year={">2019"}>} Amount)	Sum({<ISO_Date={">=2019-07-01"}>} Amount)	Sum({<US_Date={">=4/1/2018<=12/31/2018"}>} Amount)
Totales	**41**	**10**	**16**	**16**
Canadá	14	8	8	0
República Checa	10	0	6	1
Francia	4	2	2	2
Alemania	13	0	0	13

Explicación

- Dimensiones:
  - `Country`
- Medidas:
  - `Sum(Amount)`
    Suma de `Amount` sin una expresión de conjunto.
  - `Sum({<Year={">2019"}>}Amount)`
    Sum `Amount` para todos los años después de 2019.
  - `Sum({<ISO_Date={">=2019-07-01"}>}Amount)`
    Sum `Amount` para todas las fechas en o después de `2019-07-01`. El formato de la fecha en la búsqueda debe coincidir con el formato del campo.
  - `Sum({<US_Date={">=4/1/2018<=12/31/2018"}>}Amount)`
    Sum `Amount` para todas las fechas de `4/1/2018` a `12/31/2018`, incluidas las fechas de inicio y finalización. El formato de las fechas en la búsqueda debe coincidir con el formato del campo.

Modificadores de conjunto con búsquedas numéricas



Country	Sum (Amount)	Sum({<Year={">2019"}>} Amount)	Sum({<ISO_Date={">=2019-07-01"}>} Amount)	Sum({<US_Date={">=4/1/2018<=12/31/2018"}>} Amount)
Totals	**41**	**10**	**16**	**16**
Canada	14	8	8	0
Czech Republic	10	0	6	1
France	4	2	2	2
Germany	13	0	0	13

Ejemplo 3: Expresiones de gráfico con búsquedas de expresiones

Cree una tabla en una hoja de Qlik Sense con las siguientes expresiones de gráfico.

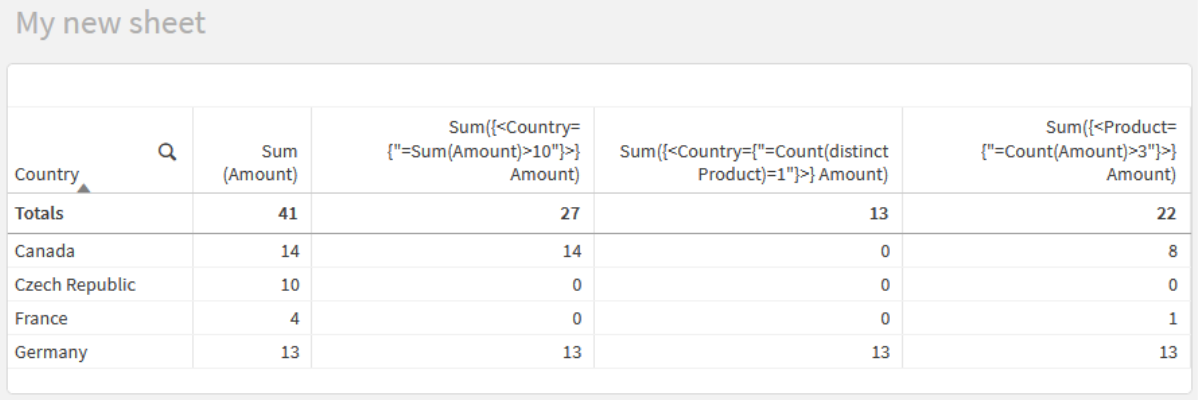
Table - Set modifiers with expression searches

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

Explicación

- Dimensiones:
  - `Country`
- Medidas:
  - `Sum(Amount)`
    Suma de `Amount` sin una expresión de conjunto.
  - `Sum({<Country={"=Sum(Amount)>10"}>}Amount)`
    Sum `Amount` para todos los países que tienen una suma agregada de `Amount` mayor que 10.
  - `Sum({<Country={"=Count(distinct Product)=1"}>}Amount)`
    Sum `Amount` para todos los países que están asociados con exactamente un producto distinto.
  - `Sum({<Product={"=Count(Amount)>3"}>}Amount)`
    Sum `Amount` para todos los países que tienen más de tres transacciones en los datos.

Modificadores de conjunto con búsquedas de expresiones


My new sheet

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

Ejemplos	Resultados
sum({$–1<Product = {“*Internal*”, “*Domestic*”}>} Sales)	Devuelve las ventas de la selección actual, excluyendo las operaciones relativas a productos con la cadena "Internal" o "Domestic" en el nombre del producto.
sum({$<Customer = {“=Sum ({1<Year = {2007}>} Sales) > 1000000”}>} Sales)	Devuelve las ventas de la selección actual, pero con una nueva selección en el campo "Customer": solo los clientes que durante 2007 tuvieron un total de ventas superior a 1.000.000.

Modificadores de conjunto con expansiones de signo dólar

Las expansiones de signo dólar son construcciones que se calculan antes de analizar y evaluar la expresión. A continuación, el resultado se inserta en la expresión, en lugar de la `$(…)`. El cálculo de la expresión se realiza posteriormente utilizando el resultado de la expansión de dólar.

El editor de expresiones muestra una vista previa de la expansión de dólar para que pueda verificar qué devuelve su expansión de signo dólar.

Vista previa de la expansión de signo dólar en el editor de expresiones

Utilice expansiones de signo dólar cuando desee usar un cálculo dentro de su conjunto de elementos.

Por ejemplo, si desea ver únicamente el último año posible, puede usar la siguiente construcción:

```
<Year = {$(=Max(Year))}>
```

`Max(Year)` se calcula primero y el resultado se insertaría en la expresión en lugar de la `$(…)`.

El resultado tras la expansión del dólar será una expresión como la siguiente:

```
<Year = {2021}>
```

La expresión dentro de la expansión del dólar se calcula en función de la selección actual. Esto significa que si tiene una selección en otro campo, el resultado de la expresión se verá afectado.

Si desea que el cálculo sea independiente de la selección, utilice el análisis de conjuntos dentro de la expansión del dólar. Por ejemplo:

```
<Year = {$(=Max({1} Year))}>
```

Cadenas de texto

Cuando desee que la expansión de dólar dé como resultado una cadena de texto, se aplican las reglas de entrecomillado normales. Por ejemplo:

```
<Country = {'$(=FirstSortedValue(Country,Date)'}>
```

El resultado tras la expansión del dólar será una expresión como la siguiente:

```
<Country = {'New Zealand'}>
```

Obtendrá un error de sintaxis si no usa las comillas.

Números

Cuando desee que la expansión de dólar dé como resultado un número, asegúrese de que la expansión tenga el mismo formato que el campo. Esto significa que a veces es necesario ajustar la expresión en una función de formato.

Por ejemplo:

```
<Amount = {$(=Num(Max(Amount), '###0.00'))}>
```

El resultado tras la expansión del dólar será una expresión como la siguiente:

```
<Amount = {12362.00}>
```

Utilice un hash para obligar a la expansión a que use siempre un punto decimal y no un separador de miles. Por ejemplo:

```
<Amount = {$(#=Max(Amount))}>
```

Fechas

Cuando desee que la expansión de dólar dé como resultado una fecha, asegúrese de que la expansión tenga el formato correcto. Esto significa que a veces es necesario ajustar la expresión en una función de formato.

Por ejemplo:

```
<Date = {'$(=Date(Max(Date)))'}>
```

El resultado tras la expansión del dólar será una expresión como la siguiente:

```
<Date = {'12/31/2015'}>
```

Al igual que con las cadenas de texto, debe utilizar las comillas correctas.

Un caso de uso muy habitual es desear que el cálculo se limite al último mes (o año). En ese caso puede usar una búsqueda numérica en combinación con la función `AddMonths()`.

Por ejemplo:

```
<Date = {">=$(=AddMonths(Today(),-1))"}>
```

El resultado tras la expansión del dólar será una expresión como la siguiente:

```
<Date = {">=9/31/2021"}>
```

Esto seleccionará todos los eventos que ocurrieron el último mes.

Ejemplo: Expresiones de gráfico para modificadores de conjunto con expansiones de signo dólar

Ejemplo: expresiones de gráfico

Script de carga

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear los ejemplos de expresión del gráfico a continuación.

```
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];
```

Expresiones de gráfico con expansiones de signo dólar

Cree una tabla en una hoja de Qlik Sense con las siguientes expresiones de gráfico.

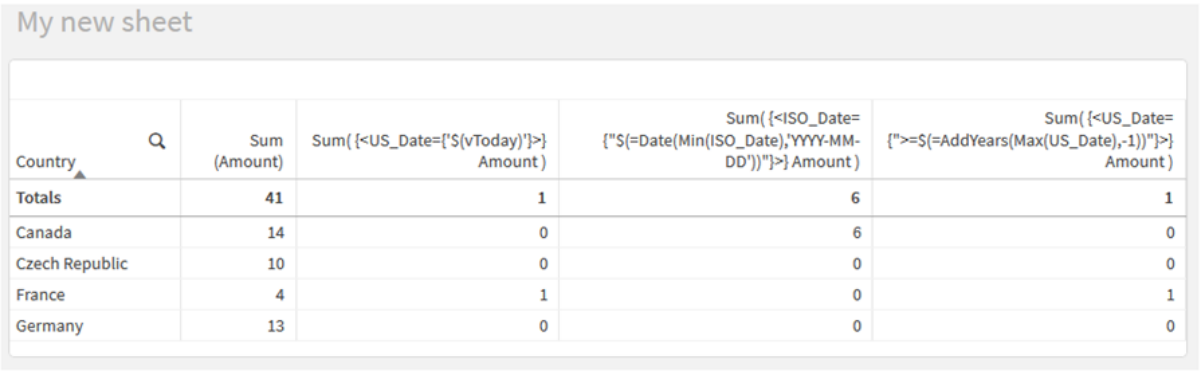
Tabla: Modificadores de conjunto con expansiones de signo dólar

País	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)
Totales	**41**	**1**	**6**	**1**
Canadá	14	0	6	0
República Checa	10	0	0	0
Francia	4	1	0	1
Alemania	13	0	0	0

Explicación

- Dimensiones:
  - `Country`
- Medidas:
  - `Sum(Amount)`
    Suma de `Amount` sin una expresión de conjunto.
  - `Sum({<US_Date={'$(vToday)'}>}Amount)`
    Sum `Amount` para todos los registros en los que la `US_Date` es la misma que en la variable `vToday`.
  - `Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>}Amount)`
    Sum `Amount` para todos los registros en los que la `ISO_Date` es la misma que la primera `ISO_Date` posible (más pequeña). La función `Date()` es necesaria para garantizar que el formato de la fecha coincida con el del campo.
  - `Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>}Amount)`
    Sum `Amount` para todos los registros que tienen una `US_Date` después o en la misma fecha de un año antes que la última `US_Date` posible (mayor). La función `AddYears()` devolverá una fecha en el formato especificado por la variable `DateFormat` y este debe coincidir con el formato del campo `US_Date`.

Modificadores de conjunto con expansiones de signo dólar



Ejemplos	Resultados
sum({$<Year = {$(#vLastYear)}>} Sales)	Devuelve las ventas del año anterior en relación con la selección actual. Aquí se utiliza una variable vLastYear, que contiene el año relevante, en una expansión de signo de dólar.
sum({$<Year = {$(#=Only(Year)-1)}>} Sales)	Devuelve las ventas del año anterior en relación con la selección actual. Aquí se utiliza una expansión signo dólar para calcular el año anterior.

Modificadores de conjunto con operadores de conjunto

Los operadores de conjuntos se utilizan para incluir, excluir o intersectar diferentes conjuntos de elementos. Combinan los diferentes métodos para definir conjuntos de elementos.

Los operadores son los mismos que los utilizados para los identificadores de conjuntos.

Operadores

Operador	Descripción
+	Unión. Esta operación binaria devuelve un conjunto compuesto por los registros o elementos que pertenecen a cualquiera de los dos operandos de conjunto.
-	Exclusión. Esta operación binaria devuelve un conjunto formado por los registros o elementos que pertenecen al primero, pero no al segundo, de los operandos de conjunto. Además, cuando se emplee como operador unario, devuelve el otro conjunto complementario.
*	Intersección. Esta operación binaria devuelve un conjunto compuesto por los registros o elementos que pertenecen a los dos operandos de conjunto.
/	Diferencia simétrica (XOR). Esta operación binaria devuelve un conjunto formado por los registros o elementos que pertenecen a cualquiera de los dos operandos del conjunto, pero no a ambos.

Por ejemplo, los dos modificadores siguientes definen el mismo conjunto de valores de campo:

- `<Year = {1997, "20*"}>`
- `<Year = {1997} + {"20*"}>`

Ambas expresiones seleccionan 1997 y los años que empiezan por 20. En otras palabras, esta es la unión de las dos condiciones.

Los operadores de conjuntos también permiten definiciones más complejas. Por ejemplo:

```
<Year = {1997, "20*"} - {2000}>
```

Esta expresión seleccionará los mismos años que los anteriores, pero además excluirá el año 2000.

.

Ejemplos: Expresiones de gráfico para modificadores de conjuntos con operadores de conjuntos

Ejemplos: expresiones de gráfico

Script de carga

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear los ejemplos de expresión del gráfico a continuación.

```
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];
```

Expresiones de gráfico

Cree una tabla en una hoja de Qlik Sense con las siguientes expresiones de gráfico.

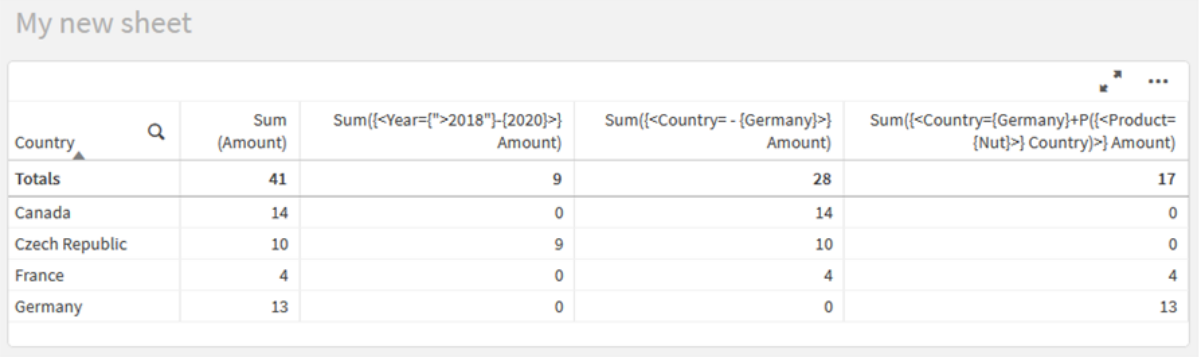
Tabla: Modificadores de conjunto con operadores de conjunto

País	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum({<Country=-{Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>}Country)>} Amount)
Totales	**41**	**9**	**28**	**17**
Canadá	14	0	14	0
República Checa	10	9	10	0
Francia	4	0	4	4
Alemania	13	0	0	13

Explicación

- Dimensiones:
  - `Country`
- Medidas:
  - `Sum(Amount)`
    Suma de `Amount` sin una expresión de conjunto.
  - `Sum({<Year={">2018"}-{2020}>}Amount)`
    Sum `Amount` para todos los años después de 2018, excepto 2020.
  - `Sum({<Country=-{Germany}>}Amount)`
    Sum `Amount` para todos los países excepto `Germany`. Tenga en cuenta el operador de exclusión unario.
  - `Sum({<Country={Germany}+P({<Product={Nut}>}Country)>}Amount)`
    Sum `Amount` para `Germany` y todos los países asociados con el producto `Nut`.

Modificadores de conjunto con operadores de conjunto



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

Ejemplos	Resultados
sum({$<Product = Product + {OurProduct1} – {OurProduct2} >} Sales)	Devuelve las ventas de la selección actual, pero con el producto "OurProduct1" agregado a la lista de productos seleccionados y "OurProduct2" eliminado de la lista de productos seleccionados.
sum({$<Year = Year + ({“20*”,1997} – {2000}) >} Sales)	Devuelve las ventas de la selección actual, pero con selecciones adicionales en el campo “Year”: 1997 y todos los que comienzan por "20", excepto el 2000. Observe que si se incluye 2000 en la selección actual, se incluirá todavía tras la modificación.
sum({$<Year = (Year + {“20*”,1997}) – {2000} >} Sales)	Devuelve prácticamente lo mismo que el anterior, pero en este caso se excluirá el 2000, incluso si estuviera inicialmente incluido en la selección actual. El ejemplo muestra la importancia de utilizar en ocasiones los paréntesis, que ayudan a definir un orden de prioridad.

Ejemplos	Resultados
sum({$<Year = {“*”} – {2000}, Product = {“*bearing*”} >} Sales)	Devuelve las ventas de la selección actual, pero con una nueva selección en "Year": todos los años excepto 2000; y solo para los productos que contienen la cadena "bearing" (rodamiento).

Modificadores de conjunto con operadores de conjunto implícitos

La forma estándar de escribir selecciones en un modificador de conjunto es usar un signo igual. Por ejemplo:

```
Year = {">2015"}
```

La expresión a la derecha del signo igual en el modificador de conjunto se denomina un conjunto de elementos. Define un conjunto de valores de campo distintos, en otras palabras, una selección.

Esta notación define una nueva selección, sin tener en cuenta la selección actual en el campo. Entonces, si el identificador del conjunto contiene una selección en este campo, la selección anterior será reemplazada por la del conjunto de elementos.

Cuando desee basar su selección en la selección actual del campo, debe utilizar una expresión diferente.

Por ejemplo, si desea respetar la selección anterior y agregar el requisito de que el año sea posterior a 2015, puede escribir lo siguiente:

```
Year = Year * {">2015"}
```

El asterisco es un operador de conjunto que define una intersección, por lo que obtendrá la intersección entre la selección actual en `Year` y el requisito adicional de que el año sea posterior a `2015`. Una forma alternativa de escribir esto es la siguiente:

```
Year *= {">2015"}
```

Es decir, el operador de asignación (`*=`) define implícitamente una intersección.

De manera similar, las uniones implícitas, las exclusiones y las diferencias simétricas se pueden definir utilizando los siguientes signos: `+=`, `-=`, `/=`

Ejemplos: Expresiones de gráfico para modificadores de conjuntos con operadores de conjuntos implícitos

Ejemplos: expresiones de gráfico

Script de carga

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear los ejemplos de expresión del gráfico a continuación.

```
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];
```

Expresiones de gráfico con operadores de conjunto implícitos

Cree una tabla en una hoja de Qlik Sense con las siguientes expresiones de gráfico.

Seleccione `Canada` y `Czech Republic` de una lista de países.

Tabla: expresiones de gráfico con operadores de conjunto implícitos

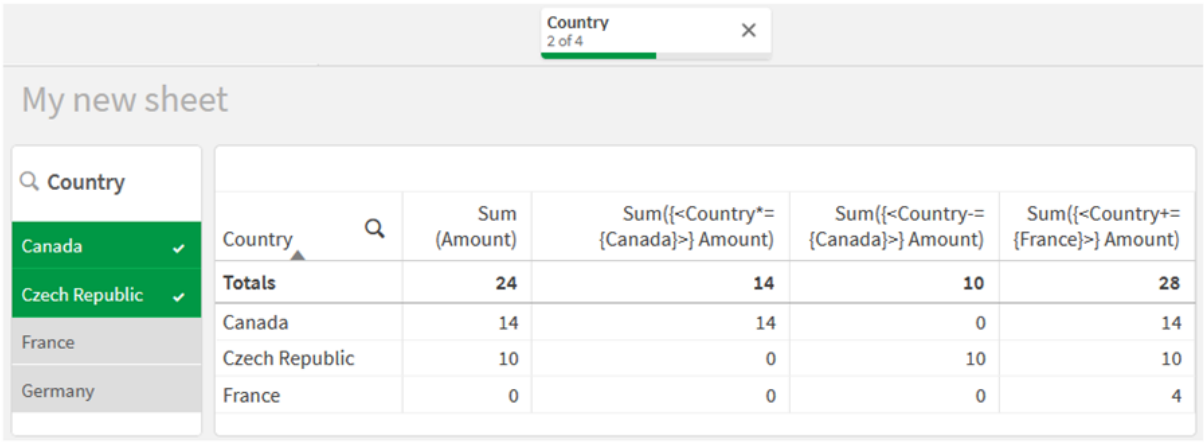
País	Sum (Amount)	Sum({<Country*= {Canada}>} Amount)	Sum({<Country-= {Canada}>} Amount)	Sum({<Country+= {France}>} Amount)
Totales	**24**	**14**	**10**	**28**
Canadá	14	14	0	14
República Checa	10	0	10	10
Francia	0	0	0	4

Explicación

- Dimensiones:
  - `Country`
- Medidas:
  - `Sum(Amount)`
    Sum `Amount` para la selección actual. Observe que solo `Canada` y `Czech Republic` tienen valores distintos de cero.
  - `Sum({<Country*={Canada}>}Amount)`
    Sum `Amount` para la selección actual, que intersecta con el requisito de que `Country` sea `Canada`. Si `Canada` no forma parte de la selección del usuario, la expresión de conjunto devuelve un conjunto vacío y la columna tendrá `0` en todas las filas.

- `Sum({<Country-={Canada}>}Amount)`
  Sum `Amount` para la selección actual, pero primero se debe excluir `Canada` de la selección de `Country`. Si `Canada` no forma parte de la selección del usuario, la expresión de conjunto no cambiará ningún número.
- `Sum({<Country+={France}>}Amount)`
  Sum `Amount` para la selección actual, pero primero se debe añadir `France` a la selección de `Country`. Si `France` ya forma parte de la selección del usuario, la expresión de conjunto no cambiará ningún número.

Modificadores de conjunto con operadores de conjunto implícitos.



Ejemplos	Resultados
sum({$<Product += {OurProduct1, OurProduct2} >} Sales)	Devuelve las ventas de la selección actual, pero utilizando una unión implícita para agregar los productos ''OurProduct1" y "OurProduct2'' a la lista de productos seleccionados.
sum({$<Year += {“20*”,1997} – {2000} >} Sales)	Devuelve las ventas de la selección actual, pero utilizando una unión implícita para añadir un determinado número de años a la selección: 1997 y todos los que comiencen por "20", excepto el 2000. Observe que si se incluye 2000 en la selección actual, se incluirá todavía tras la modificación. Igual que `<Year=Year + ({“20*”,1997}-{2000})>`.
sum({$<Product *= {OurProduct1} >} Sales)	Devuelve las ventas de la selección actual, pero solo para la intersección de los productos actualmente seleccionados y el producto OurProduct1.

Modificadores de conjunto que utilizan funciones de conjunto

A veces necesitamos definir un conjunto de valores de campo empleando una definición de conjunto anidada. Por ejemplo, es posible que desee seleccionar todos los clientes que han adquirido un producto específico, sin seleccionar el producto.

En tales casos, use las funciones del conjunto de elementos `P()` y `E()`. Estas devuelven los conjuntos de elementos de valores posibles y valores excluidos de un campo, respectivamente. Dentro de los paréntesis, puede especificar el campo en cuestión y una expresión de conjunto que defina el ámbito. Por ejemplo:

```
P({1<Year = {2021}>} Customer)
```

Esto devolverá el conjunto de clientes que tuvieron transacciones en 2021. Luego puede usar esto en un modificador de conjunto. Por ejemplo:

```
Sum({<Customer = P({1<Year = {2021}>} Customer)>} Amount)
```

Esta expresión de conjunto seleccionará a estos clientes, pero no restringirá la selección a 2021.

Estas funciones no podrán emplearse en otras expresiones.

Además, solo se pueden usar conjuntos naturales dentro de las funciones del conjunto de elementos. Es decir, un conjunto de registros que se pueden definir mediante una simple selección.

Por ejemplo, el conjunto dado por `{1-$}` no puede definirse siempre mediante una selección y, por lo tanto, no es un conjunto natural. El uso de estas funciones en conjuntos no naturales arrojará resultados inesperados.

Ejemplos: Expresiones de gráfico para modificadores de conjunto que utilizan funciones de conjunto

Ejemplos: expresiones de gráfico

Script de carga

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear los ejemplos de expresión del gráfico a continuación.

```
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];
```

Expresiones de gráfico

Cree una tabla en una hoja de Qlik Sense con las siguientes expresiones de gráfico.

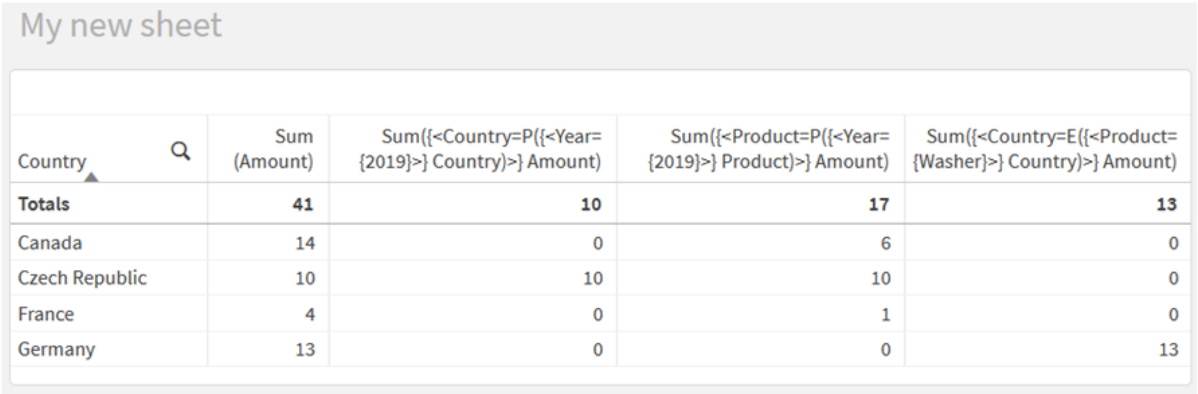
Tabla: Modificadores de conjunto que utilizan funciones de conjunto

País	Sum (Amount)	Sum({<Country=P ({<Year= {2019}>}Country)>} Amount)	Sum({<Product=P ({<Year= {2019}>}Product)>} Amount)	Sum({<Country=E ({<Product= {Washer}>}Country)>} Amount)
Totales	**41**	**10**	**17**	**13**
Canadá	14	0	6	0
República Checa	10	10	10	0
Francia	4	0	1	0
Alemania	13	0	0	13

Explicación

- Dimensiones:
  - `Country`
- Medidas:
  - `Sum(Amount)`
    Suma de `Amount` sin una expresión de conjunto.
  - `Sum({<Country=P({<Year={2019}>} Country)>} Amount)`
    Sum `Amount` para los países que están asociados con el año `2019`. Sin embargo, no limitará el cálculo a `2019`.
  - `Sum({<Product=P({<Year={2019}>} Product)>} Amount)`
    Sum `Amount` para los productos que están asociados con el año `2019`. Sin embargo, no limitará el cálculo a `2019`.
  - `Sum({<Country=E({<Product={Washer}>} Country)>} Amount)`
    Sum `Amount` para los países que no están asociados con el producto `Washer`.

Modificadores de conjunto que utilizan funciones de conjunto



Ejemplos	Resultados
sum({$<Customer = P({1<Product= {'Shoe'}>} Customer)>} Sales)	Devuelve las ventas de la selección actual, pero solo de los clientes que han comprado alguna vez el producto "Shoe". La función del elemento P() aquí devuelve una lista de posibles clientes; aquellos que están implicados por la selección "Shoe" en el campo Product.
sum({$<Customer = P({1<Product= {'Shoe'}>})>} Sales)	Igual que el anterior. Si se omite el campo en la función de elemento, la función devolverá los valores posibles del campo especificados en la asignación externa.
sum({$<Customer = P({1<Product= {'Shoe'}>} Supplier)>} Sales)	Devuelve las ventas de la selección actual, pero solo de los clientes que han suministrado alguna vez el producto "Shoe", es decir, que el cliente es también proveedor. La función de elemento P() aquí devuelve una lista de posibles proveedores; aquellos implicados por la selección "Shoe" en el campo Product. La lista de proveedores se utiliza como una selección en el campo. Customer.
sum({$<Customer = E({1<Product= {'Shoe'}>})>} Sales)	Devuelve las ventas de la selección actual, pero solo de los clientes que nunca han comprado el producto "Shoe". La función de elemento E() aquí devuelve una lista de clientes excluidos; aquellos que están excluidos por la selección "Shoe" en el campo Product.

Expresiones de conjunto internas y externas

Las expresiones de conjunto se utilizan dentro y fuera de las funciones de agregación y van entre llaves.

Cuando utiliza una expresión de conjunto dentro de una función de agregación, puede presentar el siguiente aspecto:

Ejemplo: Expresión de conjunto interna

```
Sum( {$<Year={2021}>} Sales )
```

Utilice una expresión de conjunto fuera de la función de agregación si tiene expresiones con múltiples agregaciones y desea evitar escribir la misma expresión de conjunto en cada función de agregación.

Si utiliza una expresión de conjunto externa, debe colocarse al comienzo del área.

Ejemplo: Expresión de conjunto externa

```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

Si utiliza una expresión de conjunto fuera de la función de agregación, también puede aplicarla en medidas maestras actuales.

Ejemplo: Expresión de conjunto externa aplicada a la medida maestra

```
{<Year={2021}>} [Master Measure]
```

Una expresión de conjunto utilizada fuera de las funciones de agregación afecta a toda la expresión, a menos que esté entre corchetes, en ese caso los corchetes definen el alcance. En el ejemplo siguiente de ámbito léxico, la expresión de conjunto solo se aplica a la agregación que va dentro de los corchetes.

Ejemplo: Ámbito léxico

```
( {<Year={2021}>} Sum(Amount) / Count(distinct Customer) ) – Avg(CustomerSales)
```

Reglas

Alcance léxico

La expresión de conjunto afecta a toda la expresión, a menos que esté encerrada entre corchetes. Si es así, los corchetes definen el alcance léxico.

Posición

La expresión de conjunto debe colocarse al comienzo del ámbito léxico.

Contexto

El contexto es la selección que es relevante para la expresión. Tradicionalmente, el contexto siempre ha sido el estado predeterminado de la selección actual. Pero si un objeto está establecido en un estado alterno, el contexto es el estado alterno de la selección actual.

También puede definir un contexto en forma de una expresión de conjunto externa.

Herencia

Las expresiones de conjunto internas tienen prioridad sobre las expresiones de conjunto externas. Si la expresión de conjunto interna contiene un identificador de conjunto, reemplaza el contexto. De lo contrario, el contexto y la expresión de conjunto se fusionarán.

- `{$<SetExpression>}`: ignora la expresión de conjunto externa
- `{<SetExpression>}`: se fusiona con la expresión de conjunto externa

Asignación de conjunto de elementos

La asignación del conjunto de elementos determina cómo se fusionarán las dos selecciones. Si se utiliza un signo igual normal, la selección en la expresión del conjunto interno tiene prioridad. De lo contrario, se utilizará el operador de conjunto implícito.

- `{<Field={value}>}`: esta selección interna reemplaza cualquier selección externa en "`Field`".
- `{<Field+={value}>}`: esta selección interna se fusiona con la selección externa en "`Field`", utilizando el operador de unión.
- `{<Field*={value}>}`: esta selección interna se fusiona con la selección externa en "`Field`", utilizando el operador de intersección.

Herencia en varios pasos

La herencia se puede dar en varios pasos. Ejemplos:

- Selección actual → `Sum(Amount)`
  La función de agregación utilizará el contexto, que aquí es la selección actual.
- Selección actual → `{<Set1>} Sum(Amount)`
  `Set1` heredará su contexto de la selección actual y el resultado será el contexto para la función de agregación.
- Selección actual → `{<Set1>} ({<Set2>} Sum(Amount))`
  `Set2` heredará su contexto de `Set1`, que a su vez lo hereda de la selección actual, y el resultado será el contexto para la función de agregación.

La función Aggr()

La función `Aggr()` crea una agregación anidada que tiene dos agregaciones independientes. En el ejemplo a continuación, se calcula `Count()` para cada valor de `Dim`, y la matriz resultante se agrega mediante la función `Sum()`.

Ejemplo:

`Sum(Aggr(Count(X),Dim))`
`Count()` es la agregación interna y `Sum()` la agregación externa.

- La agregación interna no hereda ningún contexto de la agregación externa.
- La agregación interna hereda el contexto de la función `Aggr()`, que puede contener una expresión de conjunto.
- Tanto la función `Aggr()` como la función de agregación externa heredan el contexto de una expresión de conjunto externa.

Tutorial - Crear una expresión de conjunto

Puede crear expresiones de conjunto en Qlik Sense para apoyar el análisis de datos. En este contexto, el análisis a menudo se denomina análisis de conjuntos. El análisis de conjuntos ofrece una forma de definir un ámbito que es diferente del conjunto de registros definido por la selección actual en una app.

Lo que aprenderá

Este tutorial proporciona las expresiones de datos y gráficos para crear expresiones de conjuntos utilizando modificadores, identificadores y operadores de conjuntos.

¿A quién va dirigido este tutorial?

Este tutorial va dirigido a desarrolladores de aplicaciones que se sientan cómodos trabajando con el editor de script y las expresiones de gráficos.

Lo que debe hacer antes de comenzar

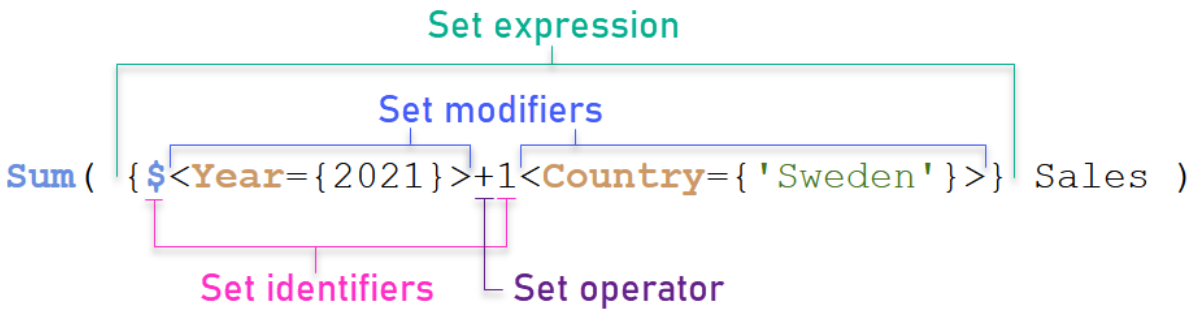
Una asignación de acceso profesional a Qlik Sense Enterprise, que le permite cargar datos y crear aplicaciones.

- Análisis de conjuntos. Parte 1: Introducción para principiantes
- Análisis de conjuntos. Parte 2:

Elementos en una expresión de conjunto

Las expresiones de conjunto se incluyen en una función de agregación, como `Sum()`, `Max()`, `Min()`, `Avg()` o `Count()`. Las expresiones de conjunto se construyen a partir de bloques de construcción conocidos como elementos. Estos elementos son modificadores, identificadores y operadores de conjuntos.

Elementos en una expresión de conjunto



La expresión de conjunto anterior, por ejemplo, se construye a partir de la agregación `Sum(Sales)`. La expresión de conjunto se incluye entre corchetes exteriores: `{ }`

El primer operando de la expresión es: `$<Year={2021}>`

Este operando devuelve las ventas del año `2021` para la selección actual. El modificador, `<Year={2021}>`, contiene la selección del año 2021. El identificador del conjunto `$` indica que la expresión del conjunto se basa en la selección actual.

El segundo operando en la expresión es: `1<Country={'Sweden'}>`

Este operando devuelve Sales para Sweden. El modificador, `<Country={'Sweden'}>`, contiene la selección del país Sweden. El identificador de conjunto `1` indica que se ignorarán las selecciones realizadas en la app.

Por último, el operador de conjunto `+` indica que la expresión devuelve un conjunto que consiste en los registros que pertenecen a cualquiera de los dos operandos del conjunto.

Tutorial: Crear una expresión de conjunto

Complete los siguientes procedimientos para crear las expresiones de conjunto que se muestran en este tutorial.

Crear una nueva app y cargar datos

Haga lo siguiente:

1. Cree una nueva app.
2. Haga clic en **Editor de script**. También puede, si lo desea, hacer clic en **Preparar** > **Editor de carga de datos** en la barra de navegación.
3. Cree una nueva sección en el **Editor de carga de datos**.

4. Copie los siguientes datos y péguelos en la nueva sección: *Datos del tutorial de expresión de conjunto (page 323)*
5. Haga clic en **Cargar datos**. Los datos se cargan como una carga inline.

Crear expresiones de conjunto con modificadores

El modificador de conjunto consta de uno o varios nombres de campo, cada uno de ellos seguido por una selección que debería efectuarse en el campo. El modificador va entre paréntesis angulares. Por ejemplo, en esta expresión de conjunto:

```
Sum ( {<Year = {2015}>} Sales )
```

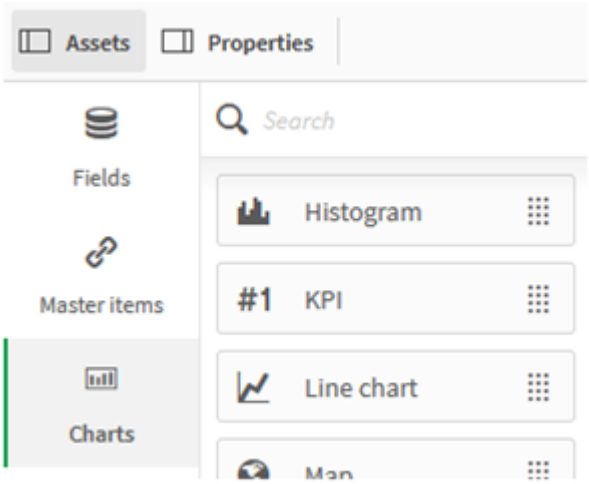
El modificador es:

```
<Year = {2015}>
```

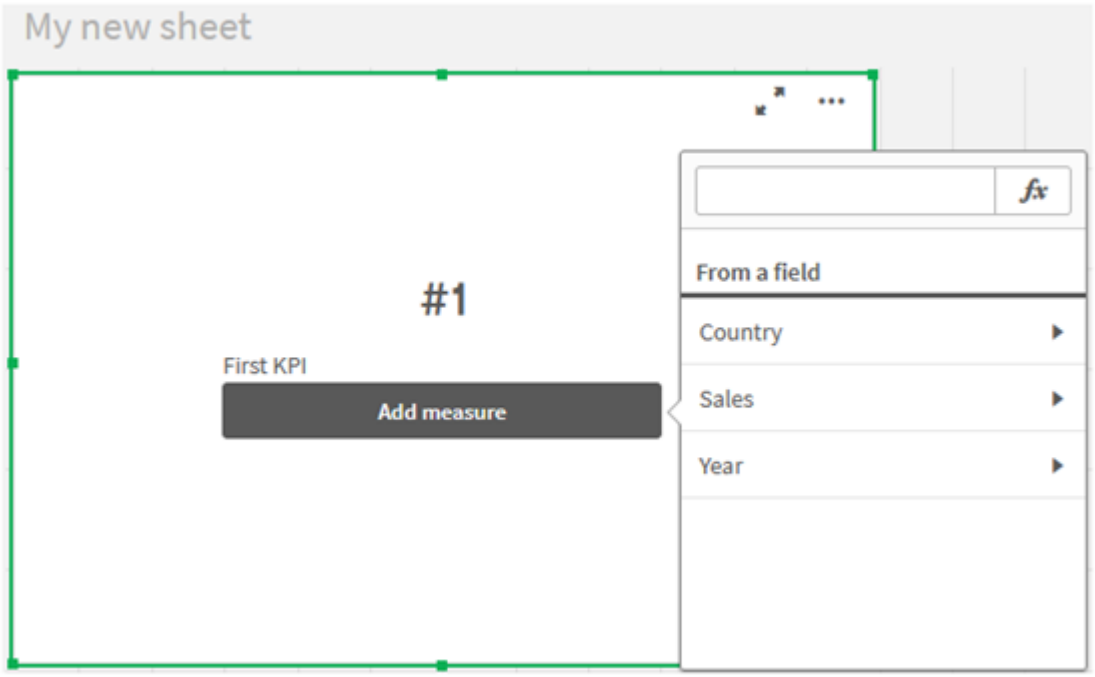
Este modificador especifica que se seleccionarán los datos del año 2015. Las llaves entra las que se incluye el modificador indican una expresión establecida.

Haga lo siguiente:

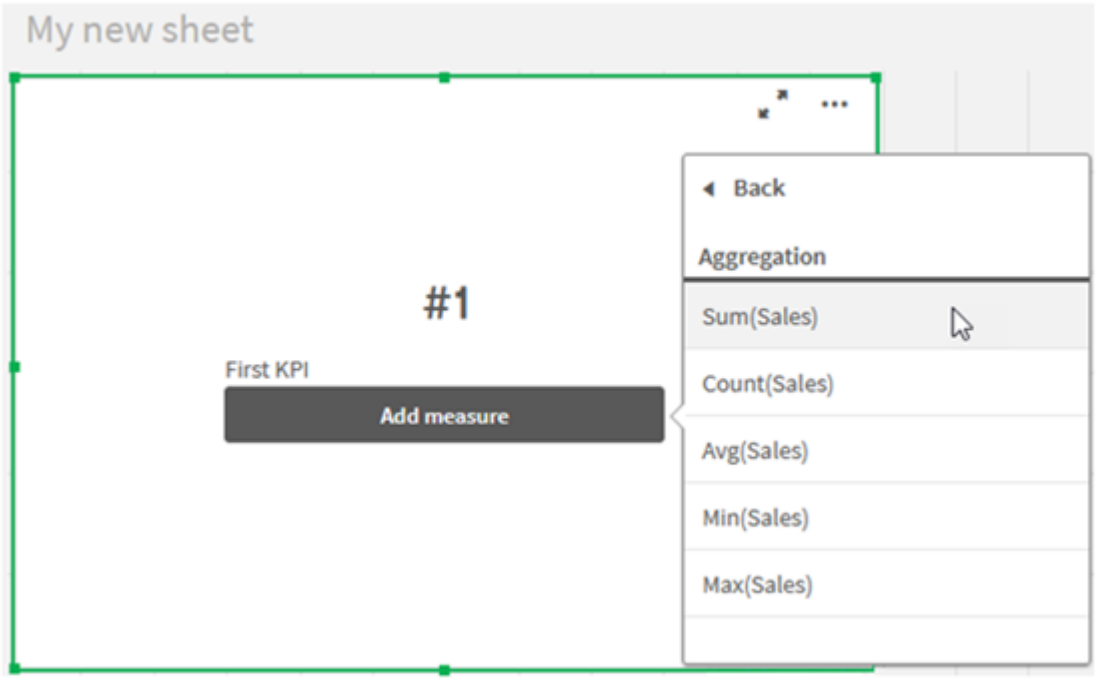
1. En una hoja, abra el panel de **Activos** desde la barra de navegación y después haga clic en **Gráficos**.



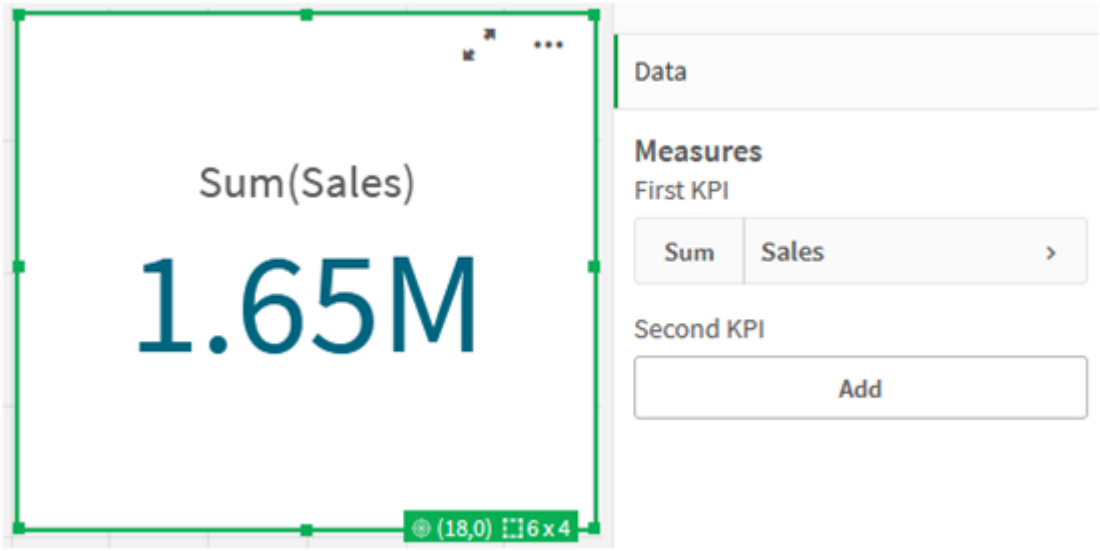
2. Arrastre un **KPI** a la hoja y después haga clic en **Añadir medida**.



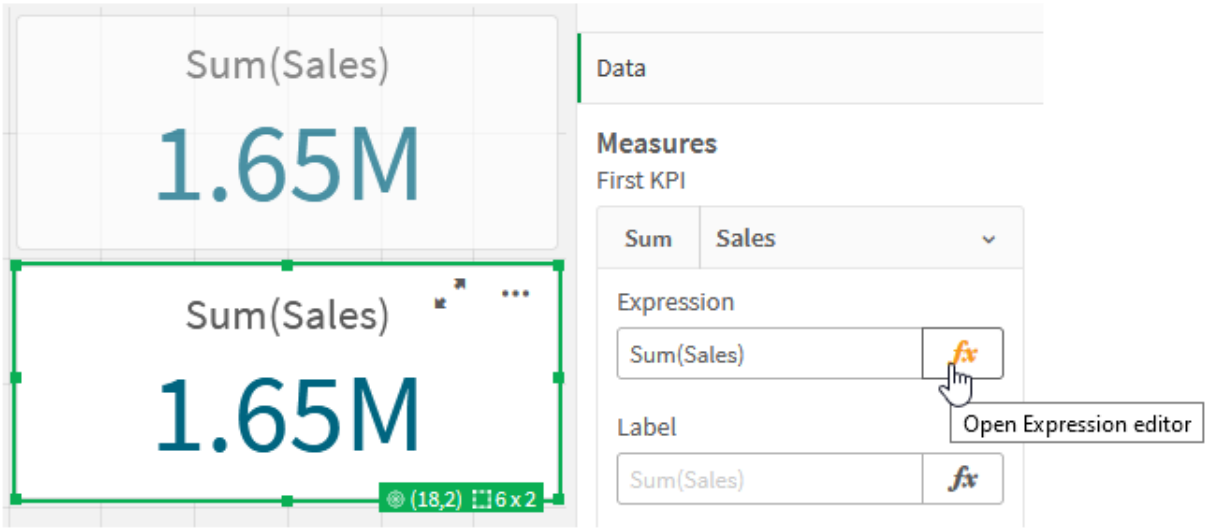
3. Haga clic en `Sales` y después seleccione `Sum(Sales)` para la agregación.



El KPI muestra la suma de las ventas de todos los años.

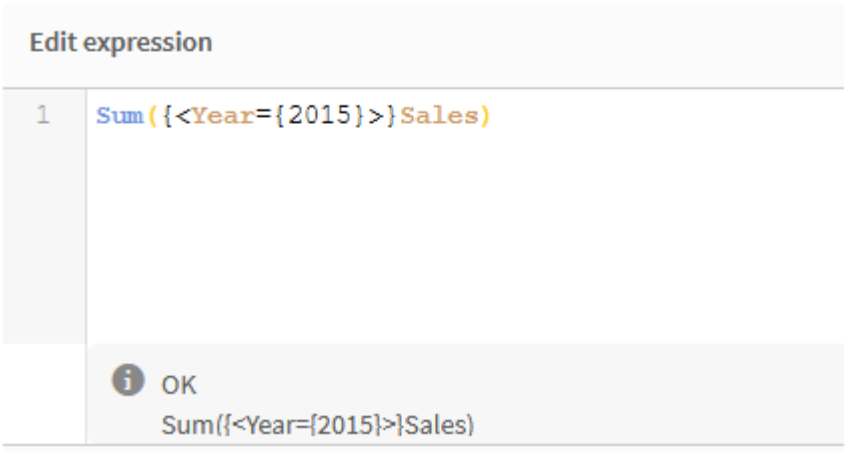


4. Copie y pegue el KPI para crear un nuevo KPI.
5. Haga clic en el nuevo KPI y después haga clic en **Sales**, debajo de **Medidas** y luego en **Abrir el editor de expresiones**.

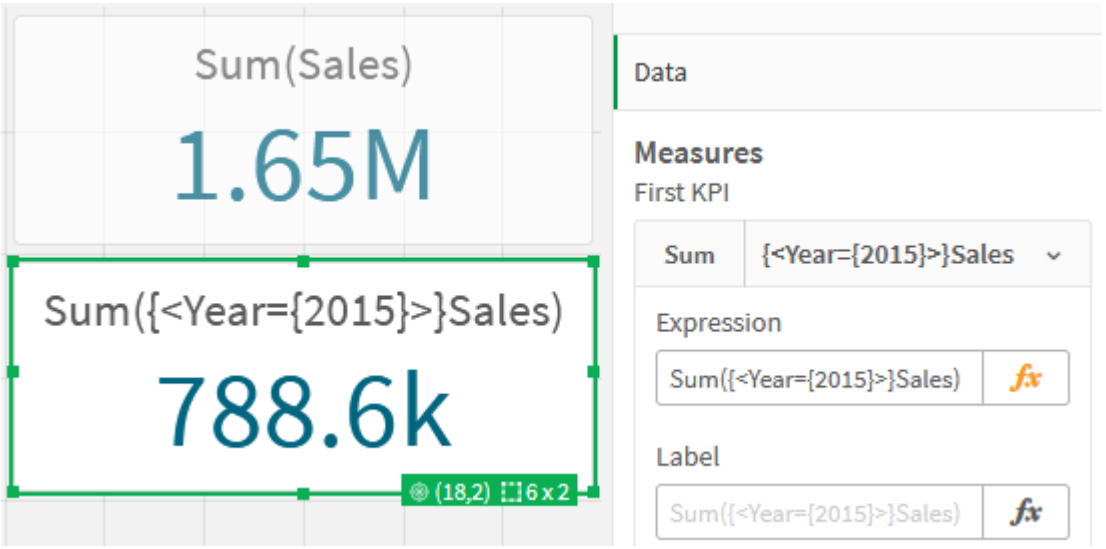


El editor de expresiones se abre con la agregación `Sum(Sales)`.

6. En el editor de expresiones, cree una expresión para sumar Sales para 2015 solo:
   i. Agregue llaves para indicar una expresión de conjunto: `Sum({}Sales)`
   i. Agregue paréntesis angulares para indicar un modificador de conjunto: `Sum({<>}Sales)`
   ii. En los paréntesis angulares, agregue el campo que se ha de seleccionar, en este caso el campo es `Year`, seguido de un signo igual. A continuación, incluya 2015 entre llaves. El modificador de conjunto resultante es: `{<Year={2015}>}`.
   La expresión completa es:
   `Sum({<Year={2015}>}Sales)`

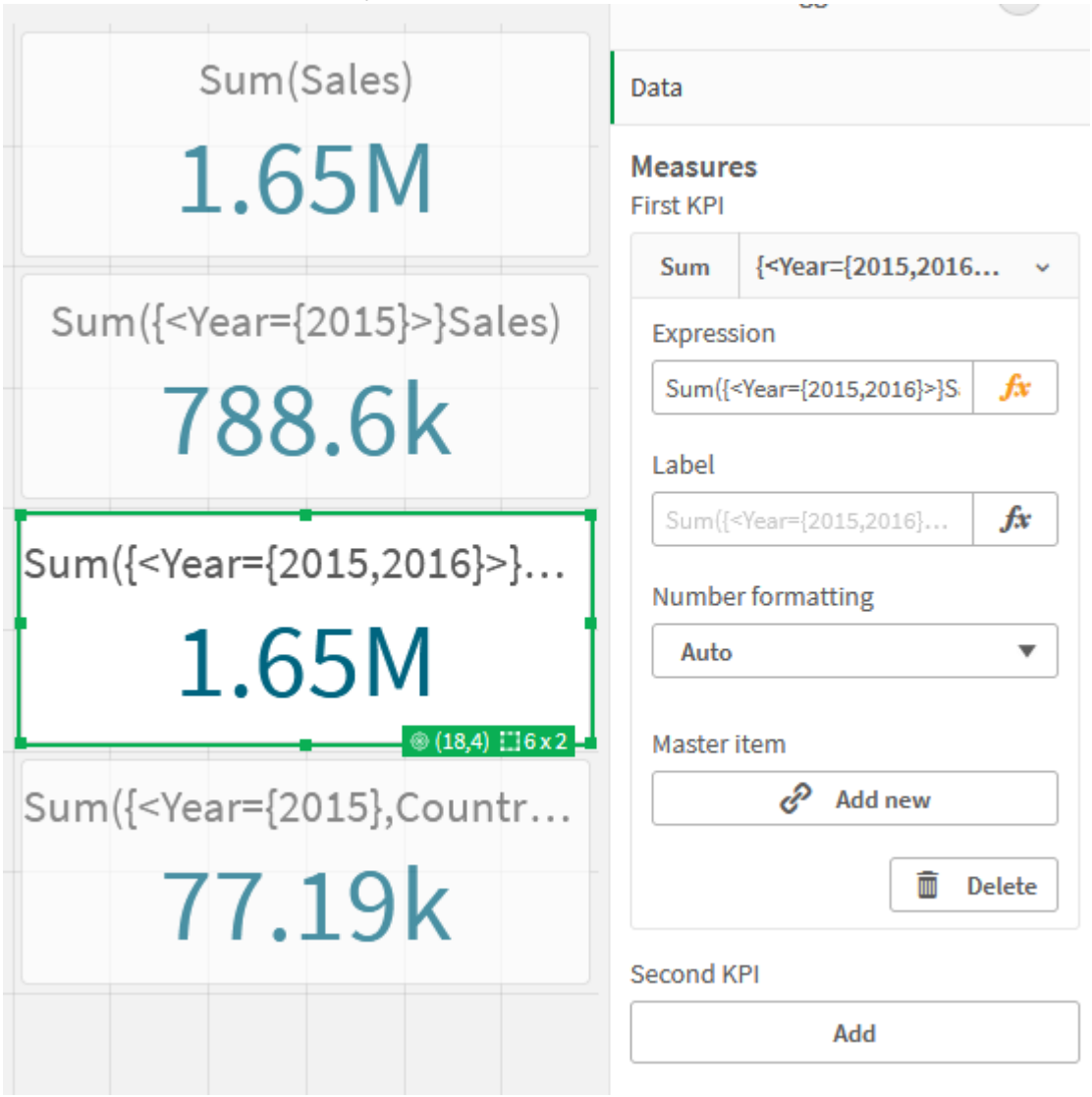


   iii. Haga clic en **Aplicar** para guardar la expresión y cerrar el editor de expresiones. La suma de Sales para 2015 se muestra en el KPI.



7. Cree dos KPI más con las siguientes expresiones:
   `Sum({<Year={2015,2016}>}Sales)`
   El modificador arriba es `<Year={2015,2016}>`. La expresión devolverá la suma de Sales para 2015 y 2016.
   `Sum({<Year={2015},Country={'Germany'}>} Sales)`
   El modificador arriba es `<Year={2015}, Country={'Germany'}>`. La expresión devolverá la suma de Sales para 2015, cuando 2015 se cruce con Germany.

KPI que utilizan modificadores de conjuntos



Agregar identificadores de conjunto

Las expresiones de conjunto anteriores utilizarán las selecciones actuales como base, porque no se utilizó un identificador. A continuación agregue identificadores para especificar el comportamiento cuando se realizan las selecciones.

Haga lo siguiente:

En su hoja, copie las siguientes expresiones de conjunto que utilizan operadores:

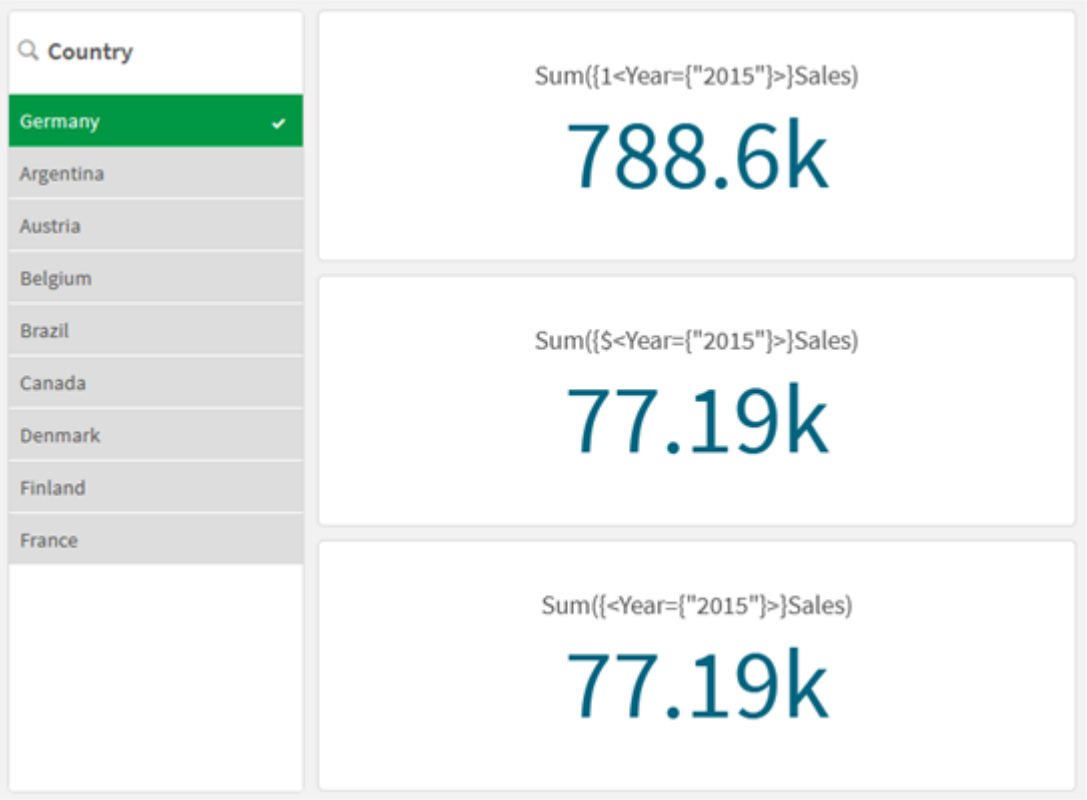
```
Sum({$<Year={"2015"}>}Sales)
```

El identificador `$` basará la expresión establecida en las selecciones actuales realizadas en los datos. Este también es el comportamiento predeterminado cuando no se utiliza un identificador.

```
Sum({1<Year={"2015"}>}Sales)
```

El identificador 1 hará que la agregación de `Sum(Sales)` en 2015 ignore la selección actual. El valor de la agregación no cambiará cuando el usuario haga otras selecciones. Por ejemplo, cuando Germany se selecciona a continuación, el valor de la suma agregada de 2015 no cambia.

KPI que utilizan modificadores e identificadores de conjuntos



Agregar operadores

Los operadores de conjunto sirven para incluir, excluir o intersectar conjuntos de datos. Todos los operadores emplean conjuntos como operandos y devuelven un conjunto como resultado.

Puede utilizar operadores de conjuntos en dos situaciones diferentes:

- Para realizar una operación de conjunto en identificadores de conjunto, que representan conjuntos de registros en datos.
- Para realizar una operación de conjunto en los conjuntos de elementos, en los valores de campo o dentro de un modificador de conjunto.

Haga lo siguiente:

En su hoja, cree o copie la siguiente expresión de conjunto:

```
Sum({$<Year={2015}>+1<Country={'Germany'}>}Sales)
```

El operador del signo más `(+)` produce una unión de los conjuntos de datos de 2015 y Germany. Como se explicó anteriormente con los identificadores de conjunto, el identificador de signo dólar `($)` significa que se usarán las selecciones actuales para el primer operando, `<Year={2015}>`, se respetarán. El identificador `1` significa que la selección se ignorará para el segundo operando, `<Country={'Germany'}>`.

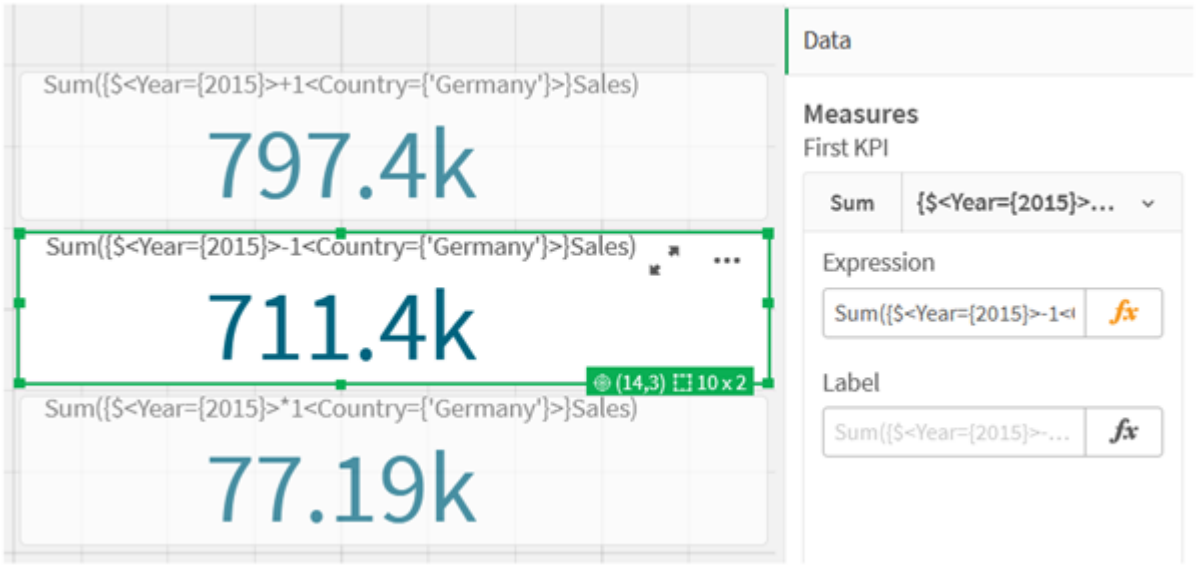
KPI que utilizan el operador de signo más (+)

También puede usar un signo menos `(-)` para obtener un conjunto de datos que se compone de los registros que pertenecen a 2015 pero no a Germany. O utilice un asterisco `(*)` para obtener un conjunto que se compone de los registros que pertenecen a ambos conjuntos.

```
Sum({$<Year={2015}>-1<Country={'Germany'}>}Sales)
```

```
Sum({$<Year={2015}>*1<Country={'Germany'}>}Sales)
```

KPI que utilizan operadores



Datos del tutorial de expresión de conjunto

Script de carga

Cargue los datos siguientes como una carga inline y luego cree las expresiones de gráfico del tutorial.

```
//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
];
```

Sintaxis para expresiones de conjuntos

La sintaxis completa (sin incluir el uso opcional de corchetes estándar para definir la prioridad) se describe mediante el formalismo Backus-Naur:

```
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 Sintaxis general para expresiones de gráficos

La siguiente estructura de sintaxis general se puede utilizar para expresiones de gráficos, con muchos parámetros opcionales:

```
expression ::= ( constant | expressionname | operator1 expression | expression operator2
expression | function | aggregation function | (expression ) )
```

donde:

constant es una cadena (un texto, una fecha o una hora) entre comillas simples o un número. Las constantes se escriben sin separador de miles y con un punto decimal como separador decimal.

expressionname es el nombre (etiqueta) de otra expresión en el mismo gráfico.

operator1 es un operador unitario (que funciona en una expresión, la de la derecha).

operator2 es un operador binario (que funciona en dos expresiones, una a cada lado).

```
function ::= functionname ( parameters )
parameters ::= expression { , expression }
```
El número y los tipos de parámetros no son aleatorios. Dependen de la función empleada.

```
aggregationfunction ::= aggregationfunctionname ( parameters2 )
parameters2 ::= aggrexpression { , aggrexpression }
```
El número y los tipos de parámetros no son aleatorios. Dependen de la función empleada.

6.4 Sintaxis general para agregaciones:

La siguiente estructura de sintaxis general se puede utilizar para agregaciones, con muchos parámetros opcionales:

```
aggrexpression ::= ( fieldref | operator1 aggrexpression | aggrexpression operator2
aggrexpression | functioninaggr | ( aggrexpression ) )
```
fieldref es un nombre de campo.

```
functionaggr ::= functionname ( parameters2 )
```
Las expresiones y funciones pueden por tanto anidarse libremente, siempre y cuando **fieldref** esté incluido siempre dentro de exactamente una función de agregación y siempre que la expresión devuelva un valor interpretable, Qlik Sense no emitirá ningún mensaje de error.

7 Operadores

Esta sección describe los operadores que se pueden utilizar en Qlik Sense. Hay dos tipos de operadores:

- Operadores unarios (toman solo un operando)
- Operadores binarios (toman dos operandos)

La mayoría de los operadores son binarios.

Se pueden definir los siguientes operadores:

- Operadores de bit
- Operadores lógicos
- Operadores numéricos
- Operadores relacionales
- Operadores de cadena

7.1 Operadores de bit

Todos los operadores de bit convierten (truncan) los operandos en enteros con signo (32 bits) y devuelven el resultado de la misma forma. Todas las operaciones se realizan bit a bit. Si un operando no puede interpretarse como un número, la operación devolverá NULL.

Operadores de bit

Operador	**Nombre completo**	**Descripción**
bitnot	Bit inverso.	Operador unitario. La operación devuelve el inverso del operando ejecutado bit a bit. **Ejemplo:** `bitnot 17` devuelve -18
bitand	Bit and.	La operación devuelve el AND lógico de los operandos ejecutados bit a bit. **Ejemplo:** `17 bitand 7` devuelve 1
bitor	Bit or.	La operación devuelve el OR lógico de los operandos ejecutados bit a bit. **Ejemplo:** `17 bitor 7` devuelve 23

Operador	Nombre completo	Descripción
bitxor	Bit or exclusivo.	La operación devuelve el OR lógico exclusivo de los operandos, ejecutado bit a bit. **Ejemplo:** `17 bitxor 7` devuelve 22
>>	Bit right shift.	La operación devuelve el primer operando desplazado un paso a la derecha. El número de pasos se define en el segundo operando. **Ejemplo:** `8 >> 2` devuelve 2
<<	Bit left shift.	La operación devuelve el primer operando desplazado a la izquierda. El número de pasos se define en el segundo operando. **Ejemplo:** `8 << 2` devuelve 32

7.2 Operadores lógicos

Todos los operadores lógicos interpretan los operandos de forma lógica y devuelven True (-1) o False (0) como resultado.

Operadores lógicos

Operador	Descripción
not	Lógica inversa. Uno de los pocos operadores unitarios. La operación devuelve la lógica inversa del operando.
and	And lógico. La operación devuelve el and lógico de los operandos.
or	Or lógico. La operación devuelve el or lógico de los operandos.
Xor	Or lógico exclusivo. La operación devuelve el or lógico exclusivo de los operandos. Es decir, como el or lógico, pero con la diferencia de que el resultado es False si ambos operandos son True.

7.3 Operadores numéricos

Todos los operadores numéricos usan los valores numéricos de los operandos y devuelven un valor numérico como resultado.

Operadores numéricos

Operador	Descripción
+	Signo para número positivo (operador unitario) o suma aritmética. La operación binaria devuelve la suma de los dos operandos.
-	Signo para número negativo (operador unitario) o substracción aritmética. La operación unitaria devuelve el operando multiplicado por -1, y la operación binaria la diferencia entre los dos operadores.
*	Multiplicación aritmética. La operación devuelve el producto de los dos operandos.
/	División aritmética. La operación devuelve el resto entre dos operandos.

7.4 Operadores relacionales

Todos los operadores relacionales comparan los valores de los operandos y devuelven True (-1) o False (0) como resultado. Todos los operadores relacionales son binarios.

Operadores relacionales

Operador	Descripción
<	Menor que. Se hace una comparación numérica si ambos operadores pueden ser interpretados numéricamente. La operación devuelve el valor lógico de la evaluación de la comparación.
<=	Menor o igual que. Se hace una comparación numérica si ambos operadores pueden ser interpretados numéricamente. La operación devuelve el valor lógico de la evaluación de la comparación.
>	Mayor que. Se hace una comparación numérica si ambos operadores pueden ser interpretados numéricamente. La operación devuelve el valor lógico de la evaluación de la comparación.
>=	Mayor o igual que. Se hace una comparación numérica si ambos operadores pueden ser interpretados numéricamente. La operación devuelve el valor lógico de la evaluación de la comparación.
=	Igual. Se hace una comparación numérica si ambos operadores pueden ser interpretados numéricamente. La operación devuelve el valor lógico de la evaluación de la comparación.
<>	Distinto de. Se hace una comparación numérica si ambos operadores pueden ser interpretados numéricamente. La operación devuelve el valor lógico de la evaluación de la comparación.

Operador	Descripción
precedes	A diferencia del operador **<**, no se intenta hacer una interpretación numérica de los valores del argumento antes de la comparación. La operación devuelve verdadero si el valor de la izquierda del operador tiene una representación de texto, la cual en una comparación de cadena, viene antes de la representación de texto del valor a la derecha. **Ejemplo:** `'1 ' precedes ' 2'` devuelve FALSE `' 1' precedes ' 2'` devuelve TRUE puesto que el valor ASCII de un espacio (' ') es de menos valor que el valor ASCII de un número. Compare esto con: `'1 ' < ' 2'` devuelve TRUE `' 1' < ' 2'` devuelve TRUE
follows	A diferencia del operador **<**, no se intenta hacer una interpretación numérica de los valores del argumento antes de la comparación. La operación devuelve verdadero si el valor a la izquierda del operador tiene una representación de texto, la cual, en una comparación de cadena, viene después de la representación de texto del valor a la derecha. **Ejemplo:** `' 2' follows '1 '` devuelve FALSE ' 2' follows ' 1' devuelve TRUE puesto que el valor ASCII de un espacio (' ') es de menos valor que el valor ASCII de un número. Compare esto con: `' 2' > ' 1'` devuelve TRUE `' 2' > '1 '` devuelve TRUE

7.5 Operadores de cadena

Hay dos operadores de cadena. Uno utiliza los valores cadena de los operandos y devuelve una cadena como resultado. El otro compara los operandos y devuelve un valor booleano que indica la correspondencia.

&

Concatenación de cadenas. La operación devuelve una cadena de texto, que consiste en los dos operandos, uno tras otro.

Ejemplo:

`'abc' & 'xyz'` devuelve 'abcxyz'

like

Comparación de cadenas con caracteres comodín. La operación devuelve un valor booleano True (-1) si la cadena anterior al operador coincide con la cadena posterior al operador. La segunda cadena puede contener los caracteres comodín * (cualquier número de caracteres arbitrarios) o ? (un carácter arbitrario).

Ejemplo:

`'abc' like 'a*'` devuelve True (-1)

`'abcd' like 'a?c*'` devuelve True (-1)

`'abc' like 'a??bc'` devuelve False (0)

8 Funciones de script y de gráfico

Transforme datos y agréguelos con scripts de carga de datos y expresiones de gráfico.

Muchas funciones se pueden utilizar de idéntica manera tanto en scripts de carga de datos como en expresiones de gráficos, pero hay algunas excepciones:

- Algunas funciones solo se pueden utilizar en scripts de carga de datos, vienen especificadas como: función de script.
- Algunas funciones solo se pueden utilizar en expresiones de gráficos, vienen especificadas como: función de gráfico.
- Algunas funciones se pueden utilizar tanto en scripts de carga de datos como en expresiones de gráficos, pero hay diferencias en los parámetros y aplicación. Estas se describen aparte, en otros temas, especificadas como función de script o función de gráfico.

8.1 Conexiones analíticas para extensiones del lado del servidor (SSE)

Las funciones habilitadas por conexiones analíticas solo estarán visibles si se han configurado las conexiones analíticas y se ha iniciado Qlik Sense.

Las conexiones analíticas se configuran en la consola QMC, vea el tema "Crear una conexión analítica" en la guía Gestionar sitios Qlik Sense.

En Qlik Sense Desktop, las conexiones analíticas se configuran editando el archivo *settings.ini*, véase el tema " Configurar conexiones analíticas en Qlik Sense Desktop" en la guía Qlik Sense Desktop.

8.2 Funciones de agregación

La familia de funciones conocida como funciones de agregación, consiste en funciones que toman varios valores de campo como entrada y devuelven un solo resultado por grupo, donde la agrupación se define mediante una dimensión de gráfico o una cláusula **group by** en la sentencia de script.

Entre las funciones de agregación se incluyen **Sum()**, **Count()**, **Min()**, **Max()** y muchas más.

La mayoría de funciones de agregación pueden utilizarse tanto en el script de carga de datos como en las expresiones de gráficos, pero la sintaxis difiere.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Al nombrar una entidad, evite asignar el mismo nombre a más de un campo, variable o medida. Existe un orden estricto de precedencia para resolver conflictos entre entidades con nombres idénticos. Este orden se refleja en cualquier objeto o contexto en el que se utilicen estas entidades. Este orden de precedencia es el siguiente:

- Dentro de una agregación, un campo tiene prioridad sobre una variable. Los nombres de las medidas no son relevantes en las agregaciones y no se priorizan.
- Fuera de una agregación, una etiqueta de medida tiene prioridad sobre una variable, que a su vez tiene prioridad sobre un nombre de campo.
- Además, fuera de una agregación, se puede reutilizar una medida haciendo referencia a su etiqueta, a menos que la etiqueta sea de hecho una etiqueta calculada. En esa situación, la medida pierde importancia para reducir el riesgo de autorreferencia y, en este caso, el nombre siempre se interpretará primero como una etiqueta de medida, en segundo lugar como un nombre de campo y en tercer lugar como un nombre de variable.

Uso de las funciones de agregación en el script de carga de datos

Las funciones de agregación solo se pueden utilizar dentro de sentencias **LOAD** y **SELECT**.

Uso de las funciones de agregación en expresiones de gráficos

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Una función de agregación agrega sobre el conjunto de registros posibles definidos por la selección. Se puede definir no obstante un conjunto alternativo de registros utilizando una expresión de conjunto en análisis de conjuntos.

Cómo se calculan las agregaciones

Una agregación recorre los registros de una tabla específica, agregando los registros en la misma. Por ejemplo, **Count**(<Field>) contará el número de registros en la tabla donde reside <Field>. Si desea agregar solo los valores de campo distintos, debe usar la cláusula **distinct**, por ej. **Count**(**distinct** <Field>).

Si la función de agregación contiene campos de diferentes tablas, la función de agregación recorrerá los registros del producto cruzado de las tablas de los campos constituyentes. Esto tiene una penalización en el rendimiento y por esta razón deben evitarse dichas agregaciones, especialmente cuando tenemos grandes cantidades de datos.

Agregación de campos clave

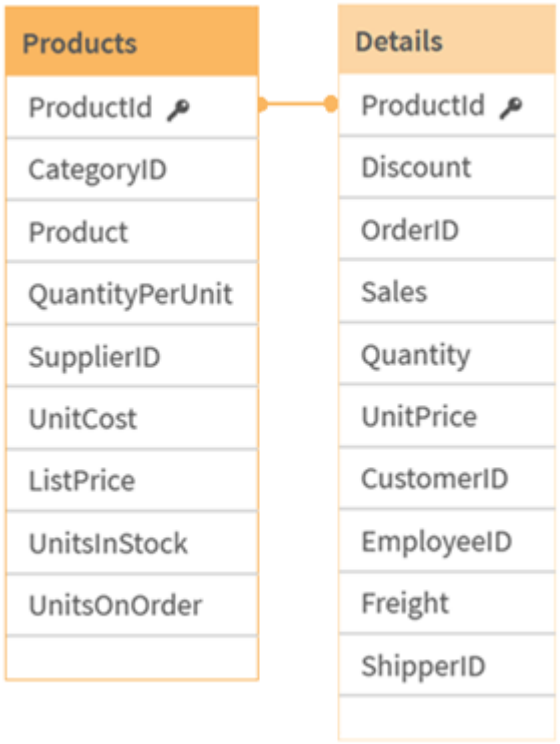
La forma en que se calculan las agregaciones significa que no puede agregar campos clave porque no está claro qué tabla debe usarse para la agregación. Por ejemplo, si el campo <Key> vincula dos tablas, no está claro si **Count**(<Key>) debe devolver el número de registros de la primera o la segunda tabla.

Sin embargo, si utiliza la cláusula **distinct**, la agregación está bien definida y se puede calcular.

Por lo tanto, si usa un campo clave dentro de una función de agregación sin la cláusula **distinct**, Qlik Sense devolverá un número que puede que no tenga sentido. La solución es usar la cláusula **distinct** o usar una copia de la clave, una copia que reside en una tabla solamente.

Por ejemplo, en las siguientes tablas, ProductID es la clave entre las tablas.

La clave ProductID entre las tablas Products y Details



Count(ProductID) se puede contar, o bien en la tabla Products (que tiene solo un registro por producto; ProductID es la clave principal), o bien se puede contar en la tabla Details (que probablemente tenga varios registros por producto). Si desea contar la cantidad de productos distintos, debe usar Count(distinct ProductID). Si desea contar el número de filas en una tabla específica, no debe usar la clave.

Funciones básicas de agregación

Descripción general de las funciones básicas de agregación

Las funciones básicas de agregación son un grupo de las funciones de agregación más comunes.

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Funciones básicas de agregación en el script de carga de datos

FirstSortedValue

FirstSortedValue() devuelve el valor de la expresión especificada en **value** que corresponde al resultado de ordenar el argumento **sort_weight**, por ejemplo, el nombre del producto con el precio unitario más bajo. El enésimo valor según el criterio de ordenación, se puede especificar en **rank**. Si más de un valor resultante comparten el mismo **sort_weight** para el **rank** especificado, la función devuelve NULL. Los valores ordenados se repiten en una serie de registros, según lo definido por una cláusula **group by**, o se agregan en todo el conjunto de datos si no se define ninguna cláusula **group by**.

```
FirstSortedValue ([ distinct ] expression, sort_weight [, rank ])
```

Max

Max() halla el valor numérico más alto de los datos agregados en la expresión, según lo definido por una cláusula **group by**. Especificando un **rank** n, se puede hallar el valor enésimo más alto.

```
Max ( expression[, rank])
```

Min

Min() devuelve el valor numérico más bajo de los datos agregados en la expresión, según lo definido por una cláusula **group by**. Especificando un **rank** n, se puede hallar el valor enésimo más bajo.

```
Min ( expression[, rank])
```

Mode

Mode() devuelve el valor más común, el valor de la moda, de los datos agregados en la expresión, definidos en una cláusula **group by**. La función **Mode()** puede devolver valores numéricos y también valores de texto.

```
Mode (expression )
```

Only

Only() devuelve un valor si hay un solo resultado posible de los datos agregados. Si los registros contienen solo un valor, entonces devuelve ese valor, de lo contrario devuelve NULL. Utilice la cláusula **group by** para evaluar múltiples registros. La función **Only()** puede devolver valores numéricos y de texto.

```
Only (expression )
```

Sum

Sum() calcula el total de los valores agregados en la expresión, según lo definido en una cláusula **group by**.

```
Sum ([distinct]expression)
```

Funciones básicas de agregación en expresiones de gráficos

Las funciones de agregación en gráficos solo pueden emplearse sobre campos en expresiones de gráficos. La expresión del argumento de una función de agregación no deberá contener otra función de agregación.

FirstSortedValue

FirstSortedValue() devuelve el valor de la expresión especificada en **value** que corresponde al resultado de ordenar el argumento **sort_weight**, por ejemplo, el nombre del producto con el precio unitario más bajo. El enésimo valor según el criterio de ordenación, se puede especificar en **rank**. Si más de un valor resultante

comparten el mismo **sort_weight** para el **rank** especificado, la función devuelve NULL.

```
FirstSortedValue - función de gráfico([{SetExpression}] [DISTINCT] [TOTAL
[<fld {,fld}>]] value, sort_weight [,rank])
```

Max

Max() encuentra el valor más alto de los datos agregados. Especificando un **rank** n, se puede hallar el valor enésimo más alto.

```
Max - función de gráficoMax() encuentra el valor más alto de los datos
agregados. Especificando un rank n, se puede hallar el valor enésimo más
alto. Puede que también desee echar un vistazo a FirstSortedValue y rangemax,
que tienen una funcionalidad similar a la función Max. Max([{SetExpression}]
[TOTAL [<fld {,fld}>]] expr [,rank])
numérico ArgumentosArgumentoDescripciónexprLa expresión o el campo que
contiene los datos que se han de medir.rankEl valor predeterminado de rank es
1, que se corresponde con el valor más alto. Especificando rank como 2,
devuelve el segundo valor más alto. Si rank es 3, devuelve el tercer valor
más alto, y así sucesivamente.SetExpressionDe forma predeterminada, la
función de agregación agregará sobre el conjunto de registros posibles
definidos por la selección. Se puede definir un conjunto alternativo de
registros mediante una expresión de análisis de conjuntos. TOTALSi la palabra
TOTAL aparece antes de los argumentos de la función, el cálculo se realiza
sobre todos los valores posibles dadas las selecciones actuales y no solo
aquellas que pertenecen al valor dimensional actual, es decir, no tiene en
cuenta las dimensiones del gráfico. Usar TOTAL [<fld {.fld}>], donde al
cualificador TOTAL le sigue una lista de uno o más nombres de campo como un
subconjunto de las variables de dimensión del gráfico, crea un subconjunto de
los valores totales posibles. DatosCustomerProductUnitSalesUnitPrice
AstridaAA416AstridaAA1015AstridaBB99BetacabBB510BetacabCC220BetacabDD-
25CanutilityAA815CanutilityCC-19Ejemplos y resultadosEjemplosResultadosMax
(UnitSales)10, porque este es el valor más alto en UnitSales.El valor de un
pedido se calcula a partir del número de unidades vendidas en (UnitSales)
multiplicado por el precio por unidad.Max(UnitSales*UnitPrice)150, porque
este es el valor más alto del resultado de calcular todos los valores
posibles de (UnitSales)*(UnitPrice).Max(UnitSales, 2)9, que es el segundo
valor más alto.Max(TOTAL UnitSales)10, porque el cualificador TOTAL significa
que se encuentra el valor más alto posible, sin tener en cuenta las
dimensiones del gráfico. Para un gráfico con Customer como dimensión, el
cualificador TOTAL garantizará que se devuelva el máximo valor en todo el
conjunto de datos, en lugar del máximo UnitSales para cada cliente.Haga la
selección Customer B.Max({1} TOTAL UnitSales)15, independientemente de la
selección realizada, porque la expresión Set Analysis {1} define el conjunto
de registros que se han de evaluar como ALL, sin importar qué selección se
realice.Datos utilizados en los ejemplos:ProductData:LOAD * inline
[Customer|Product|UnitSales|UnitPriceAstrida|AA|4|16Astrida|AA|10|15Astrida|B
```

```
B|9|9Betacab|BB|5|10Betacab|CC|2|20Betacab|DD||25Canutility|AA|8|15Canutility
|CC||19] (delimiter is '|'); FirstSortedValue RangeMax (
[{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Min

Min() encuentra el valor más bajo de los datos agregados. Especificando un **rank** n, se puede hallar el valor enésimo más bajo.

```
Min - función de gráfico([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr [,rank])
```

Mode

Mode() encuentra el valor que más veces aparece, el valor de la moda, en los datos agregados. La función **Mode()** puede procesar valores de texto y también valores numéricos.

```
Mode - función de gráfico ({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Only

Only() devuelve un valor si hay un solo resultado posible de los datos agregados. Por ejemplo, buscar el único producto en el que el precio por unidad sea 9 devolverá NULL si más de un producto tiene un precio por unidad de 9.

```
Only - función de gráfico([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr)
```

Sum

Sum() calcula el total de los valores proporcionados por la expresión o campo en todos los datos agregados.

```
Sum - función de gráfico([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr])
```

FirstSortedValue

FirstSortedValue() devuelve el valor de la expresión especificada en **value** que corresponde al resultado de ordenar el argumento **sort_weight**, por ejemplo, el nombre del producto con el precio unitario más bajo. El enésimo valor según el criterio de ordenación, se puede especificar en **rank**. Si más de un valor resultante comparten el mismo **sort_weight** para el **rank** especificado, la función devuelve NULL. Los valores ordenados se repiten en una serie de registros, según lo definido por una cláusula **group by**, o se agregan en todo el conjunto de datos si no se define ninguna cláusula **group by**.

Sintaxis:

```
FirstSortedValue ([ distinct ] value, sort-weight [, rank ])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
value Expression	La función encuentra el valor de la expresión **value** que corresponde al resultado de la ordenación **sort_weight**.
sort-weight Expression	La expresión que contiene los datos que se han de ordenar. Se encuentra el primer valor (el más bajo) de **sort_weight**, a partir del cual se determina el valor correspondiente de la expresión **value**. Si coloca un signo menos delante de **sort_weight**, la función devuelve el último valor ordenado (el más alto) en su lugar.
rank Expression	Al indicar un **rank** "n" mayor que 1, se obtiene el enésimo valor ordenado.
distinct	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

Para tener el mismo aspecto que en la columna inferior de resultados, en el panel de propiedades, bajo Ordenar, cambie de Auto a Personalizado, a continuación deseleccione el orden numérico y alfabético.

Ejemplos de script

Ejemplo	Resultado
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 La función ordena UnitSales de menor a mayor, buscando el valor de Customer con el valor más pequeño de UnitSales, el orden más pequeño. Porque CC corresponde al pedido más pequeño (valor de UnitSales = 2) para el cliente Astrida. AA corresponde al pedido más pequeño (4) para el cliente Betacab, AA corresponde al pedido más pequeño (8) para el cliente Canutility y DD corresponde al pedido más pequeño (10) para el cliente Divadip..
Dado que la tabla **Temp** se carga como en el ejemplo anterior: LOAD Customer,FirstSortedValue(Product, -UnitSales) as MyProductWithLargestOrderByCustomer Resident Temp Group By Customer;	Customer MyProductWithLargestOrderByCustomer Astrida AA Betacab DD Canutility CC Divadip - Un signo menos precede al argumento sort_weight, por lo que la función ordena el más grande primero. Porque AA corresponde al pedido más grande (valor de UnitSales:18) para el cliente Astrida, DD corresponde al pedido más grande (12) para el cliente Betacab y CC corresponde al pedido más grande (13) para el cliente Canutility. Hay dos valores idénticos para el pedido más grande (16) para el cliente Divadip, por lo tanto, esto produce un resultado nulo.

Ejemplo	Resultado

Dado que la tabla **Temp** se carga como en el ejemplo anterior:
`LOAD Customer,FirstSortedValue(distinct Product, -UnitSales) as MyProductWithSmallestOrderByCustomer Resident Temp Group By Customer;`
	`Customer MyProductWithLargestOrderByCustomer`
	`Astrida AA`
	`Betacab DD`
	`Canutility CC`
	`Divadip AA`
	Esto es lo mismo que en el ejemplo anterior, excepto en que se usa el cualificador distinct. Esto hace que el resultado duplicado de Divadip se descarte, lo que permite devolver un valor no nulo.

FirstSortedValue - función de gráfico

FirstSortedValue() devuelve el valor de la expresión especificada en **value** que corresponde al resultado de ordenar el argumento **sort_weight**, por ejemplo, el nombre del producto con el precio unitario más bajo. El enésimo valor según el criterio de ordenación, se puede especificar en **rank**. Si más de un valor resultante comparten el mismo **sort_weight** para el **rank** especificado, la función devuelve NULL.

Sintaxis:

```
FirstSortedValue([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] value, sort_weight [,rank])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
value	Campo de salida. La función encuentra el valor de la expresión **value** que corresponde al resultado de la ordenación **sort_weight**.
sort_weight	Campo de entrada. La expresión que contiene los datos que se han de ordenar. Se encuentra el primer valor (el más bajo) de **sort_weight**, a partir del cual se determina el valor correspondiente de la expresión **value**. Si coloca un signo menos delante de **sort_weight**, la función devuelve el último valor ordenado (el más alto) en su lugar.
rank	Al indicar un **rank** "n" mayor que 1, se obtiene el enésimo valor ordenado.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.

Argumento	Descripción
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Ejemplos y resultados:

Datos

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

Ejemplos y resultados

Ejemplo	Resultado
`firstsortedvalue (Product, UnitPrice)`	BB, que es el `Product` con el `UnitPrice` más bajo (9).
`firstsortedvalue (Product, UnitPrice, 2)`	BB, que es el `Product` con el segundo `UnitPrice` más bajo (10).
`firstsortedvalue (Customer, -UnitPrice, 2)`	Betacab, que es el `Customer` con el `Product` que tiene el segundo `UnitPrice` más alto (20).
`firstsortedvalue (Customer, UnitPrice, 3)`	NULL, porque hay dos valores de `Customer` (Astrida y Canutility) con el mismo rank (tercer valor más bajo) `UnitPrice`(15). Utilice el cualificador distinct para asegurarse de que no se produzcan resultados nulos inesperados.
`firstsortedvalue (Customer, -UnitPrice*UnitSales, 2)`	Canutility, que es el `Customer` con el segundo valor de pedido de venta de `UnitPrice` más alto, multiplicado por `UnitSales` (120).

Datos utilizados en los ejemplos:

```
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
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Max

Max() halla el valor numérico más alto de los datos agregados en la expresión, según lo definido por una cláusula **group by**. Especificando un **rank** n, se puede hallar el valor enésimo más alto.

Sintaxis:

```
Max ( expr [, rank])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr Expression	La expresión o el campo que contiene los datos que se han de medir.
rank Expression	El valor predeterminado de **rank** es 1, que se corresponde con el valor más alto. Especificando **rank** como 2, devuelve el segundo valor más alto. Si **rank** es 3, devuelve el tercer valor más alto, y así sucesivamente.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

Para tener el mismo aspecto que en la columna inferior de resultados, en el panel de propiedades, bajo Ordenar, cambie de Auto a Personalizado, a continuación deseleccione el orden numérico y alfabético.

Ejemplo:

```
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;
```

Tabla resultante

Cliente	MyMax
Astrida	18
Betacab	5
Canutility	8

Ejemplo:

Dado que la tabla **Temp** se carga como en el ejemplo anterior:

```
LOAD Customer, Max(UnitSales,2) as MyMaxRank2 Resident Temp Group By Customer;
```

Tabla resultante

Cliente	MyMaxRank2
Astrida	10
Betacab	4
Canutility	-

Max - función de gráfico

Max() encuentra el valor más alto de los datos agregados. Especificando un **rank** n, se puede hallar el valor enésimo más alto.

*Puede que también desee echar un vistazo a **FirstSortedValue** y **rangemax**, que tienen una funcionalidad similar a la función **Max**.*

Sintaxis:

```
Max([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
rank	El valor predeterminado de **rank** es 1, que se corresponde con el valor más alto. Especificando **rank** como 2, devuelve el segundo valor más alto. Si **rank** es 3, devuelve el tercer valor más alto, y así sucesivamente.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Ejemplos y resultados:

Datos

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

Ejemplos y resultados

Ejemplos	Resultados
`Max(UnitSales)`	10, porque este es el valor más alto en `UnitSales`.
El valor de un pedido se calcula a partir del número de unidades vendidas en (`UnitSales`) multiplicado por el precio por unidad. `Max(UnitSales*UnitPrice)`	150, porque este es el valor más alto del resultado de calcular todos los valores posibles de (`UnitSales`)*(`UnitPrice`).
`Max(UnitSales, 2)`	9, que es el segundo valor más alto.
`Max(TOTAL UnitSales)`	10, porque el cualificador TOTAL significa que se encuentra el valor más alto posible, sin tener en cuenta las dimensiones del gráfico. Para un gráfico con Customer como dimensión, el cualificador TOTAL garantizará que se devuelva el máximo valor en todo el conjunto de datos, en lugar del máximo UnitSales para cada cliente.
Haga la selección `Customer B`. `Max({1} TOTAL UnitSales)`	15, independientemente de la selección realizada, porque la expresión Set Analysis {1} define el conjunto de registros que se han de evaluar como ALL, sin importar qué selección se realice.

Datos utilizados en los ejemplos:

```
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
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Vea también:

- *FirstSortedValue - función de gráfico (page 339)*
- *RangeMax (page 1363)*

Min

Min() devuelve el valor numérico más bajo de los datos agregados en la expresión, según lo definido por una cláusula **group by**. Especificando un **rank** n, se puede hallar el valor enésimo más bajo.

Sintaxis:

```
Min ( expr [, rank])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr Expression	La expresión o el campo que contiene los datos que se han de medir.
rank Expression	El valor predeterminado de **rank** es 1, que se corresponde con el valor más bajo. Especificando **rank** como 2, devuelve el segundo valor más bajo. Si **rank** es 3, devuelve el tercer valor más bajo, y así sucesivamente.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

Para tener el mismo aspecto que en la columna inferior de resultados, en el panel de propiedades, bajo Ordenar, cambie de Auto a Personalizado, a continuación deseleccione el orden numérico y alfabético.

Ejemplo:

```
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;
```

Tabla resultante

Cliente	MyMin
Astrida	2
Betacab	4
Canutility	8

Ejemplo:

Dado que la tabla **Temp** se carga como en el ejemplo anterior:

```
LOAD Customer, Min(UnitSales,2) as MyMinRank2 Resident Temp Group By Customer;
```

Tabla resultante

Cliente	MyMinRank2
Astrida	9
Betacab	5
Canutility	-

Min - función de gráfico

Min() encuentra el valor más bajo de los datos agregados. Especificando un **rank** n, se puede hallar el valor enésimo más bajo.

*Puede que también desee echar un vistazo a **FirstSortedValue** y **rangemin**, que tienen una funcionalidad similar a la función **Min**.*

Sintaxis:

```
Min({[SetExpression] [TOTAL [<fld {,fld}>]]} expr [,rank])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
rank	El valor predeterminado de **rank** es 1, que se corresponde con el valor más bajo. Especificando **rank** como 2, devuelve el segundo valor más bajo. Si **rank** es 3, devuelve el tercer valor más bajo, y así sucesivamente.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Ejemplos y resultados:

Datos

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

La función Min() debe devolver un valor que no sea NULL del conjunto de valores dados por la expresión, si lo hay. Así que, en los ejemplos, como hay valores NULL en los datos, la función devuelve el primer valor que no sea NULL de la expresión.

Ejemplos y resultados

Ejemplos	Resultados
`Min(UnitSales)`	2, porque este es el valor más bajo no NULL en `UnitSales`.
El valor de un pedido se calcula a partir del número de unidades vendidas en (`UnitSales`) multiplicado por el precio por unidad. `Min(UnitSales*UnitPrice)`	40, porque este es el valor más bajo no NULL resultado de calcular todos los valores posibles de (`UnitSales`)*(`UnitPrice`).
`Min(UnitSales, 2)`	4, que es el segundo valor más bajo (tras los valores NULL).
`Min(TOTAL UnitSales)`	2, porque el cualificador TOTAL implica que se encuentra el valor más bajo posible, descartando las dimensiones del gráfico. Para un gráfico con Customer como dimensión, el cualificador TOTAL garantizará que se devuelva el valor mínimo en todo el conjunto de datos, en lugar del mínimo UnitSales para cada cliente.
Haga la selección Customer B. `Min({1} TOTAL UnitSales)`	2, que es independiente de la selección de Customer B. La expresión de análisis de conjuntos Set Analysis {1} define el conjunto de registros que debe evaluarse como ALL, sin importar qué selección se realice.

Datos utilizados en los ejemplos:

```
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
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Vea también:

- *FirstSortedValue - función de gráfico (page 339)*
- *RangeMin (page 1367)*

Mode

Mode() devuelve el valor más común, el valor de la moda, de los datos agregados en la expresión, definidos en una cláusula **group by**. La función **Mode()** puede devolver valores numéricos y también valores de texto.

Sintaxis:

```
Mode ( expr)
```

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
expr Expression	La expresión o el campo que contiene los datos que se han de medir.

Limitaciones:

Si más de un valor ocurre con la misma frecuencia, devuelve NULL.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

Para tener el mismo aspecto que en la columna inferior de resultados, en el panel de propiedades, bajo Ordenar, cambie de Auto a Personalizado, a continuación deseleccione el orden numérico y alfabético.

Ejemplos de script

Ejemplo	Resultado
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 porque AA es el único producto vendido más de una vez.

Mode - función de gráfico

Mode() encuentra el valor que más veces aparece, el valor de la moda, en los datos agregados. La función **Mode()** puede procesar valores de texto y también valores numéricos.

Sintaxis:

Mode({[SetExpression] [**TOTAL** [<fld {,fld}>]]} expr**)**

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Ejemplos y resultados:

Datos

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

Ejemplos y resultados

Ejemplos	Resultados
`Mode(UnitPrice)` Haga la selección `Customer A`.	15, porque este es el valor que se da con más frecuencia en `UnitSales`. Devuelve NULL (-). Ningún valor único ocurre más a menudo que otro.
`Mode(Product)` Haga la selección `Customer A`	AA, porque este es el valor que se da con más frecuencia en `Product`. Devuelve NULL (-). Ningún valor único ocurre más a menudo que otro.
`Mode (TOTAL UnitPrice)`	15, porque el cualificador TOTAL significa que el valor más habitual sigue siendo 15, incluso sin tener en cuenta las dimensiones del gráfico.
Haga la selección `Customer B`. `Mode({1} TOTAL UnitPrice)`	15, independientemente de la selección realizada, porque la expresión Set Analysis {1} define el conjunto de registros que se han de evaluar como ALL, sin importar qué selección se realice.

Datos utilizados en los ejemplos:

```
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
Canutility|AA|8|15
Canutility|CC||19
```

```
] (delimiter is '|');
```

Vea también:

- *Avg - función de gráfico (page 411)*
- *Median - función de gráfico (page 450)*

Only

Only() devuelve un valor si hay un solo resultado posible de los datos agregados. Si los registros contienen solo un valor, entonces devuelve ese valor, de lo contrario devuelve NULL. Utilice la cláusula **group by** para evaluar múltiples registros. La función **Only()** puede devolver valores numéricos y de texto.

Sintaxis:

```
Only ( expr )
```

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
expr Expression	La expresión o el campo que contiene los datos que se han de medir.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

Para tener el mismo aspecto que en la columna inferior de resultados, en el panel de propiedades, bajo Ordenar, cambie de Auto a Personalizado, a continuación deseleccione el orden numérico y alfabético.

```
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;
```

Tabla resultante

Cliente	MyUniqIDCheck
Astrida	1 porque solo el cliente Astrida tiene registros completos que incluyen CustomerID.

Only - función de gráfico

Only() devuelve un valor si hay un solo resultado posible de los datos agregados. Por ejemplo, buscar el único producto en el que el precio por unidad sea 9 devolverá NULL si más de un producto tiene un precio por unidad de 9.

Sintaxis:

```
Only([{SetExpression}] [TOTAL [<fld {,fld}>]] expr)
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Utilice Only() cuando desee un resultado NULL si hay varios valores posibles en los datos de muestra.

Ejemplos y resultados:

Datos

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16

Customer	Product	UnitSales	UnitPrice
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

Ejemplos y resultados

Ejemplos	Resultados
`Only({<UnitPrice={9}>} Product)`	BB, porque este es el único `Product` que tiene un `UnitPrice` de '9'.
`Only({<Product={DD}>} Customer)`	Betacab, porque es el único `Customer` que vende un `Product` denominado «DD».
`Only({<UnitPrice={20}>} UnitSales)`	El número de `UnitSales` donde `UnitPrice` es 20 es 2, porque solo hay un valor de `UnitSales` en el que `UnitPrice` =20.
`Only({<UnitPrice={15}>} UnitSales)`	NULL, porque hay dos valores de `UnitSales` donde el `UnitPrice` =15.

Datos utilizados en los ejemplos:

```
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
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Sum

Sum() calcula el total de los valores agregados en la expresión, según lo definido en una cláusula **group by**.

Sintaxis:

```
sum ( [ distinct] expr)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
distinct	Si la palabra **distinct** figura antes de la expresión, todos los duplicados se descartan.
expr Expression	La expresión o el campo que contiene los datos que se han de medir.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

Para tener el mismo aspecto que en la columna inferior de resultados, en el panel de propiedades, bajo Ordenar, cambie de Auto a Personalizado, a continuación deseleccione el orden numérico y alfabético.

```
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;
```

Tabla resultante

Cliente	MySum
Astrida	39
Betacab	9
Canutility	8

Sum - función de gráfico

Sum() calcula el total de los valores proporcionados por la expresión o campo en todos los datos agregados.

Sintaxis:

```
Sum([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan. *Aunque se admite el cualificador DISTINCT, utilícelo solo con extrema precaución porque puede inducir a error al lector al pensar que se muestra un valor total cuando se han omitido algunos datos.*
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Ejemplos y resultados:

Datos

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

Ejemplos y resultados

Ejemplos	Resultados
`Sum(UnitSales)`	38. El total de los valores de `UnitSales`.
`Sum(UnitSales*UnitPrice)`	505. El total de `UnitPrice` multiplicado por `UnitSales` agregado.
`Sum (TOTAL UnitSales*UnitPrice)`	505 para todas las filas de la tabla, así como el total, porque el cualificador TOTAL significa que la suma sigue siendo 505, sin tener en cuenta las dimensiones del gráfico.
Haga la selección `Customer B`. `Sum({1} TOTAL UnitSales*UnitPrice)`	505, independientemente de la selección realizada, porque la expresión Set Analysis {1} define el conjunto de registros que se han de evaluar como ALL, sin importar qué selección se realice.

Datos utilizados en los ejemplos:

```
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
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Funciones de agregación de contador

Las funciones de agregación de contador devuelven diversos tipos de cuentas de una expresión que se repite a lo largo de un número de registros en un script de carga de datos o un número de valores en una dimensión de un gráfico.

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Funciones de agregación de contador en el script de carga de datos

Count

Count() devuelve el número de valores agregados en la expresión, según lo definido por una cláusula **group by**.

```
Count ([distinct ] expression | * )
```

MissingCount

MissingCount() devuelve el número de valores perdidos agregados en la expresión, según lo definido por una cláusula **group by**.

```
MissingCount ([ distinct ] expression)
```

NullCount

NullCount() devuelve el número de valores NULL agregados en la expresión, según lo definido por una cláusula **group by**.

```
NullCount ([ distinct ] expression)
```

NumericCount

NumericCount() devuelve el número de valores numéricos hallados en la expresión, según lo definido por una cláusula **group by**.

```
NumericCount ([ distinct ] expression)
```

TextCount

TextCount() devuelve el número de valores de campo no numéricos agregados en la expresión, según lo definido por una cláusula **group by**.

```
TextCount ([ distinct ] expression)
```

Funciones de agregación de contador en expresiones de gráficos

Se pueden utilizar las siguientes funciones de agregación de contador en gráficos.

Count

Count() se utiliza para agregar el número de valores, de texto y numéricos, en cada dimensión del gráfico.

```
Count - función de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

MissingCount

MissingCount() se utiliza para agregar el número de valores perdidos en cada dimensión del gráfico. Los valores perdidos son todos valores no numéricos.

```
MissingCount - función de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]] expr)
```

NullCount

NullCount() se utiliza para agregar el número de valores NULL en cada dimensión del gráfico.

```
NullCount - función de gráfico({[SetExpression][DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

NumericCount

NumericCount() agrega el número de valores numéricos en cada dimensión del gráfico.

```
NumericCount - función de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

TextCount

TextCount() se utiliza para agregar el número de valores de campo que no son numéricos en cada dimensión del gráfico.

```
TextCount - función de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

Count

Count() devuelve el número de valores agregados en la expresión, según lo definido por una cláusula **group by**.

Sintaxis:

```
Count( [distinct ] expr)
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
distinct	Si la palabra **distinct** figura antes de la expresión, todos los duplicados se descartan.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

Para tener el mismo aspecto que en la columna inferior de resultados, en el panel de propiedades, bajo Ordenar, cambie de Auto a Personalizado, a continuación deseleccione el orden numérico y alfabético.

Ejemplos de script

Ejemplo	Resultado
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 Mientras la dimensión Customer esté incluida en la tabla en la hoja, de lo contrario el resultado de OrdersByCustomer es 3, 2.
Dado que la tabla **Temp** se carga como en el ejemplo anterior: LOAD Count(OrderNumber) as TotalOrderNumber Resident Temp;	TotalOrderNumber 10
Dado que la tabla **Temp** se carga como en el primer ejemplo: LOAD Count(distinct OrderNumber) as TotalOrderNumber Resident Temp;	TotalOrderNumber 8 Porque hay dos valores de OrderNumber con el mismo valor, 1, y un valor nulo.

Count - función de gráfico

Count() se utiliza para agregar el número de valores, de texto y numéricos, en cada dimensión del gráfico.

Sintaxis:

```
Count({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.

Argumento	Descripción
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Ejemplos y resultados:

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
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	1	25	25
Canutility	AA	3	8	15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

En los ejemplos siguientes se da por sentado que se han seleccionado todos los clientes, excepto si se indica lo contrario.

Ejemplos y resultados

Ejemplo	Resultado
`Count(OrderNumber)`	10, porque hay 10 campos que podrían tener un valor para OrderNumber, y se cuentan todos los registros, incluso los vacíos. *"0" cuenta como un valor y no una celda vacía. Sin embargo, si una medida agrega a 0 para una dimensión esa dimensión no se incluirá en los gráficos.*
`Count(Customer)`	10, porque Count devuelve el número de veces que aparece en todos los campos.
`Count(DISTINCT [Customer])`	4, porque al usar el calificador Distinct, Count solo devuelve resultados que aparecen una vez.
Siempre y cuando el cliente Canutility esté seleccionado `Count(OrderNumber)/Count({1} TOTAL OrderNumber)`	0,2 porque la expresión devuelve el número de pedidos del cliente seleccionado como un porcentaje de pedidos de la totalidad de clientes. En este caso 2 / 10.
Puesto que los clientes Astrida y Canutility están seleccionados `Count(TOTAL <Product> OrderNumber)`	5 porque ese es el número de pedidos de producto realizados solo para los clientes seleccionados y se cuentan las celdas vacías.

Datos utilizados en los ejemplos:

```
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 '|');
```

MissingCount

MissingCount() devuelve el número de valores perdidos agregados en la expresión, según lo definido por una cláusula **group by**.

Sintaxis:

```
MissingCount ( [ distinct ] expr)
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
expr Expression	La expresión o el campo que contiene los datos que se han de medir.
distinct	Si la palabra **distinct** figura antes de la expresión, todos los duplicados se descartan.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

Para tener el mismo aspecto que en la columna inferior de resultados, en el panel de propiedades, bajo Ordenar, cambie de Auto a Personalizado, a continuación deseleccione el orden numérico y alfabético.

Ejemplos de script

Ejemplo	Resultado
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 La segunda sentencia da: TotalMissingCount 3 en una tabla con esa dimensión.

Ejemplo	Resultado
Dado que la tabla **Temp** se carga como en el ejemplo anterior: `LOAD MissingCount(distinct OrderNumber) as TotalMissingCountDistinct Resident Temp;`	`TotalMissingCountDistinct` 1 Porque solo hay un OrderNumber, un valor perdido.

MissingCount - función de gráfico

MissingCount() se utiliza para agregar el número de valores perdidos en cada dimensión del gráfico. Los valores perdidos son todos valores no numéricos.

Sintaxis:

```
MissingCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Ejemplos y resultados:

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16

Customer	Product	OrderNumber	UnitSales	Unit Price
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	DD	3		25

Ejemplos y resultados

Ejemplo	Resultado
`MissingCount([OrderNumber])`	3, porque 3 de los 10 campos OrderNumber están vacíos *"0" cuenta como un valor y no una celda vacía. Sin embargo, si una medida agrega a 0 para una dimensión esa dimensión no se incluirá en los gráficos.*
`MissingCount ([OrderNumber])/MissingCount ({1} Total [OrderNumber])`	La expresión devuelve el número de pedidos incompletos del cliente seleccionado como una fracción de pedidos incompletos de todos los clientes. Hay un total de 3 valores de OrderNumber que faltan para todos los clientes. Así que para cada Customer al que le falte un valor de Product el resultado es 1/3.

Datos utilizados en el ejemplo:

```
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 '|');
```

NullCount

NullCount() devuelve el número de valores NULL agregados en la expresión, según lo definido por una cláusula **group by**.

Sintaxis:

```
NullCount ( [ distinct ] expr)
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
expr Expression	La expresión o el campo que contiene los datos que se han de medir.
distinct	Si la palabra **distinct** figura antes de la expresión, todos los duplicados se descartan.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

Para tener el mismo aspecto que en la columna inferior de resultados, en el panel de propiedades, bajo Ordenar, cambie de Auto a Personalizado, a continuación deseleccione el orden numérico y alfabético.

Ejemplos de script

Ejemplo	Resultado
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 La segunda sentencia da: TotalNullCount 1 en una tabla con esa dimensión, dado que solo un registro contiene un valor null.

NullCount - función de gráfico

NullCount() se utiliza para agregar el número de valores NULL en cada dimensión del gráfico.

Sintaxis:

```
NullCount({[SetExpression][DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
set_ expression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.

Argumento	Descripción
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Ejemplos y resultados:

Ejemplos y resultados

Ejemplo	Resultado
NullCount ([OrderNumber])	1 porque hemos introducido un valor nulo utilizando NullInterpret en la sentencia **LOAD** inline.

Datos utilizados en el ejemplo:

```
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=;
```

NumericCount

NumericCount() devuelve el número de valores numéricos hallados en la expresión, según lo definido por una cláusula **group by**.

Sintaxis:

```
NumericCount ( [ distinct ] expr)
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
expr Expression	La expresión o el campo que contiene los datos que se han de medir.
distinct	Si la palabra **distinct** figura antes de la expresión, todos los duplicados se descartan.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

Para tener el mismo aspecto que en la columna inferior de resultados, en el panel de propiedades, bajo Ordenar, cambie de Auto a Personalizado, a continuación deseleccione el orden numérico y alfabético.

Ejemplo de script

Ejemplo	Resultado
`LOAD NumericCount(OrderNumber) as TotalNumericCount Resident Temp;`	La segunda sentencia da: TotalNumericCount 7 en una tabla con esa dimensión.
Dado que la tabla **Temp** se carga como en el ejemplo anterior: `LOAD NumericCount(distinct OrderNumber) as TotalNumericCountDistinct Resident Temp;`	TotalNumericCountDistinct 6 Debido a que hay un OrderNumber que duplica a otro, el resultado es 6 no duplicados.

Ejemplo:

```
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;
```

Tabla resultante

Cliente	NumericCountByCustomer
Astrida	3
Betacab	2
Canutility	0
Divadip	2

NumericCount - función de gráfico

NumericCount() agrega el número de valores numéricos en cada dimensión del gráfico.

Sintaxis:

```
NumericCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
set_ expression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.

Argumento	Descripción
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Ejemplos y resultados:

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15
Astrida	BB	4	9	1
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	DD	3		25

En los ejemplos siguientes se da por sentado que se han seleccionado todos los clientes, excepto si se indica lo contrario.

Ejemplos y resultados

Ejemplo	Resultado
`NumericCount ([OrderNumber])`	7, porque tres de los 10 campos de OrderNumber están vacíos. *"0" cuenta como un valor y no una celda vacía. Sin embargo, si una medida agrega a 0 para una dimensión esa dimensión no se incluirá en los gráficos.*

Ejemplo	Resultado
`NumericCount ([Product])`	0, porque todos los nombres de productos están en texto. Normalmente puede utilizar esto para comprobar que a los campos de texto no se les ha dado contenido numérico.
`NumericCount (DISTINCT [OrderNumber])/Count (DISTINCT [OrderNumber)]`	Cuenta el número de números de pedido numéricos distintos y lo divide por el número de números de pedido numéricos y no numéricos. Esto será 1 si todos los valores de campo son numéricos. Normalmente puede utilizar esto para comprobar que todos los valores de campo son numéricos. En el ejemplo, hay 7 valores numéricos distintos para OrderNumber 8 numéricos distintos y no numéricos, por lo que la expresión devuelve 0,875.

Datos utilizados en el ejemplo:

```
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 '|');
```

TextCount

TextCount() devuelve el número de valores de campo no numéricos agregados en la expresión, según lo definido por una cláusula **group by**.

Sintaxis:

```
TextCount ( [ distinct ] expr)
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
expr Expression	La expresión o el campo que contiene los datos que se han de medir.
distinct	Si la palabra **distinct** figura antes de la expresión, todos los duplicados se descartan.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

Para tener el mismo aspecto que en la columna inferior de resultados, en el panel de propiedades, bajo Ordenar, cambie de Auto a Personalizado, a continuación deseleccione el orden numérico y alfabético.

Ejemplo:

```
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;
```

Tabla resultante

Cliente	**ProductTextCount**
Astrida	3
Betacab	3
Canutility	2
Divadip	2

Ejemplo:

```
LOAD Customer,TextCount(OrderNumber) as OrderNumberTextCount Resident Temp Group By Customer;
```

Tabla resultante

Cliente	**OrderNumberTextCount**
Astrida	0
Betacab	1
Canutility	2
Divadip	0

TextCount - función de gráfico

TextCount() se utiliza para agregar el número de valores de campo que no son numéricos en cada dimensión del gráfico.

Sintaxis:

```
TextCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Ejemplos y resultados:

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15
Astrida	BB	4	9	1
Betacab	BB	6	5	10
Betacab	CC	5	2	20
Betacab	DD			25

Customer	Product	OrderNumber	UnitSales	Unit Price
Canutility	AA			15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

Ejemplos y resultados

Ejemplo	Resultado
`TextCount ([Product])`	10 porque los 10 campos en Product son texto. *"0" cuenta como un valor y no una celda vacía. Sin embargo, si una medida agrega a 0 para una dimensión esa dimensión no se incluirá en los gráficos. Las celdas vacías se evalúan como no textuales y TextCount no las cuenta.*
`TextCount ([OrderNumber])`	3 porque las celdas vacías se cuentan. Normalmente podemos utilizar esto para verificar que no se haya dado valores de texto a campos numéricos o que son distintos de cero.
`TextCount (DISTINCT [Product])/Count ([Product)]`	Cuenta todo el número de valores de texto distintos de Product (4) y lo divide por el número total de valores en Product (10). El valor por defecto es 0,4.

Datos utilizados en el ejemplo:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|1|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 '|');
```

Funciones de agregación financiera

En esta sección se describen las funciones de agregación para operaciones financieras relacionadas con los pagos y el flujo de caja.

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Funciones financieras de agregación en el script de carga de datos

IRR

IRR() devuelve la tasa interna de retorno agregada de una serie de flujos de efectivo representados por los números en la expresión que se repite por una serie de registros según lo definido por una cláusula group by.

```
IRR (expression)
```

XIRR

XIRR() devuelve la tasa interna de rendimiento agregada (anual) de una planificación de flujos de efectivo (no necesariamente periódicos) representados por números pareados en **pmt** y **date** que se repiten a lo largo de un número de registros, según lo definido por una cláusula group by. Todos los pagos son descontados según una base de un año de 365 días.

```
XIRR (valueexpression, dateexpression )
```

NPV

La función de script **NPV()** toma una tasa de descuento y varios valores ordenados por período. Las entradas (ingresos) son positivas y las salidas (pagos futuros) se supone que son valores negativos para estos cálculos. Estos ocurren al final de cada período.

```
NPV(rate, expression)
```

XNPV

XNPV() devuelve el valor actual neto agregado de una planificación de flujos de efectivo (no necesariamente periódicos) representado por números pareados en **pmt** y **date**. Todos los pagos son descontados según una base de un año de 365 días.

```
XNPV (rate, valueexpression, dateexpression)
```

Funciones financieras de agregación en expresiones de gráficos

Se pueden utilizar las siguientes funciones de agregación financiera en gráficos.

IRR

IRR() devuelve la tasa interna de retorno agregada de una serie de flujos de efectivo representados por los números en la expresión proporcionados por **value** que se repiten por las dimensiones del gráfico.

```
 IRR - función de gráfico([TOTAL [<fld {,fld}>]] value)
```

NPV

NPV() devuelve el valor presente neto agregado de una inversión en función de una tasa **discount_rate** por período y una serie de pagos futuros (valores negativos) e ingresos (valores positivos), representados por los números en **value**, que se repiten por las dimensiones del gráfico. Se asume que los pagos e ingresos se producen al final de cada período.

```
NPV - función de gráfico([TOTAL [<fld {,fld}>]] discount_rate, value)
```

XIRR

XIRR() devuelve la tasa interna de rendimiento agregada (anual) de una planificación de flujos de efectivo (no necesariamente periódicos) representados por números pareados en las expresiones proporcionadas por **pmt** y **date**, que se repiten a lo largo de las dimensiones del gráfico. Todos los pagos son descontados según una base de un año de 365 días.

```
XIRR - función de gráfico([TOTAL [<fld {,fld}>]] pmt, date)
```

XNPV

XNPV() devuelve el valor presente neto agregado de una planificación de flujos de efectivo (no necesariamente periódicos) representados por números pareados de las expresiones dadas por **pmt** y **date**, que se repiten por las dimensiones del gráfico. Todos los pagos son descontados según una base de un año de 365 días.

```
XNPV - función de gráfico([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

IRR

IRR() devuelve la tasa interna de retorno agregada de una serie de flujos de efectivo representados por los números en la expresión que se repite por una serie de registros según lo definido por una cláusula group by.

Estos flujos de efectivo no tienen por qué ser constantes y parejos, como lo serían en una anualidad. Sin embargo, sí que han de producirse a intervalos regulares, como por ejemplo, en períodos mensuales o anuales. La tasa interna de devolución es el último tipo de interés recibido para una inversión consistente en pagos (valores negativos) e ingresos (valores positivos) que se suceden durante períodos regulares. La función necesita al menos un valor positivo y uno negativo para ser evaluada.

Esta función utiliza una versión simplificada del método de Newton para calcular la tasa interna de retorno (TIR).

Sintaxis:

```
IRR(value)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	La expresión o el campo que contiene los datos que se han de medir.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos se descartan.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

Ejemplos y resultados:

Ejemplos y resultados

Ejemplo	Año	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 - función de gráfico

IRR() devuelve la tasa interna de retorno agregada de una serie de flujos de efectivo representados por los números en la expresión proporcionados por **value** que se repiten por las dimensiones del gráfico.

Estos flujos de efectivo no tienen por qué ser constantes y parejos, como lo serían en una anualidad. Sin embargo, sí que han de producirse a intervalos regulares, como por ejemplo, en períodos mensuales o anuales. La tasa interna de devolución es el último tipo de interés recibido para una inversión consistente en pagos (valores negativos) e ingresos (valores positivos) que se suceden durante períodos regulares. La función necesita al menos un valor positivo y uno negativo para ser evaluada.

Esta función utiliza una versión simplificada del método de Newton para calcular la tasa interna de retorno (TIR).

Sintaxis:

```
IRR([TOTAL [<fld {,fld}>]] value)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	La expresión o el campo que contiene los datos que se han de medir.

Argumento	Descripción
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Los valores de texto, valores NULL y valores perdidos se descartan.

Ejemplos y resultados:

Ejemplos y resultados

Ejemplo	Resultado
`IRR (Payments)`	0.1634 Se supone que los pagos son periódicos por naturaleza, por ejemplo, mensuales. *El campo Fecha se usa en el ejemplo XIRR donde los pagos pueden no ser periódicos, siempre y cuando especifique las fechas en las que se realizaron dichos pagos.*

Datos utilizados en los ejemplos:

```
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 '|');
```

Vea también:

- *XIRR - función de gráfico (page 390)*
- *Aggr - función de gráfico (page 559)*

NPV

La función de script **NPV()** toma una tasa de descuento y varios valores ordenados por período. Las entradas (ingresos) son positivas y las salidas (pagos futuros) se supone que son valores negativos para estos cálculos. Estos ocurren al final de cada período.

El valor actual neto, VAN (o NPV por sus siglas en inglés), se utiliza para calcular el valor total actual de una corriente futura de flujos de caja. Para calcular el VAN (o NPV), tenemos que estimar los flujos de caja futuros de cada período y determinar el tipo de descuento correcto. La función de script **NPV()** toma una tasa de descuento y múltiples valores ordenados por período. Las entradas (ingresos) son positivas y las salidas (pagos futuros) se supone que son valores negativos para estos cálculos. Estos ocurren al final de cada período.

Sintaxis:

```
NPV(discount_rate, value)
```

Tipo de datos que devuelve: numérico. De forma predeterminada, el resultado recibirá el formato de moneda.

La fórmula para calcular el valor actual neto es:

$$NPV = \sum_{t=1}^{n} \frac{R_t}{(1+i)^t}$$

donde:

- R_t = Entradas y salidas netas de efectivo durante un solo periodo `t`
- `i` = Tasa de descuento o rendimiento que podría obtenerse en inversiones alternativas
- `t` = Número de períodos del temporizador

Argumentos

Argumento	Descripción
discount_rate	**discount_rate** es la tasa porcentual de descuento aplicada. Un valor de 0,1 indicaría una tasa de descuento del 10%.
value	Este campo contiene valores para múltiples períodos ordenados por período. Se supone que el primer valor es el flujo de caja al final del período 1, y así sucesivamente.

Limitaciones:

La función `NPV()` tiene las siguientes limitaciones:

- Los valores de texto, valores NULL y valores perdidos se descartan.
- Los valores del flujo de caja deben estar en orden ascendente por período.

Cuándo se utiliza

NPV() es una función financiera utilizada para verificar la rentabilidad del proyecto y para deducir otras medidas. Esta función es útil cuando los flujos de caja están disponibles como datos sin procesar.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia SET DateFormat de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: pago único (script)

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de un proyecto y su flujo de caja durante un período, que se carga en una tabla denominada CashFlow.
- Una carga residente de la tabla CashFlow, que se utiliza para calcular el campo NPV del proyecto en una tabla denominada NPV.
- Una tasa de descuento codificada del 10%, que se utiliza en el cálculo del NPV.
- Una sentencia Group By, que se utiliza para agrupar todos los pagos del proyecto.

Script de carga

```
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;
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- PrjId
- NPV

Tabla de resultados

PrjId	NPV
1	$909.09

Para recibir un pago único de $1000 al final de un período, a una tasa de descuento del 10% por período, el NPV es igual a $1000 dividido por (1 + tasa de descuento). El NPV efectivo es igual a $909,09

Ejemplo 2: múltiples pagos (script)

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de un proyecto y su flujo de caja durante varios períodos, que se carga en una tabla denominada CashFlow.
- Una carga residente de la tabla CashFlow, que se utiliza para calcular el campo NPV del proyecto en una tabla denominada NPV.
- Una tasa de descuento codificada del 10% (0,1) se utiliza en el cálculo del NPV.
- Una sentencia Group By, que se utiliza para agrupar todos los pagos del proyecto.

Script de carga

```
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;
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- PrjId
- NPV

Tabla de resultados

PrjId	NPV
1	$1735.54

Para recibir pagos de $1000 al final de dos períodos, a una tasa de descuento del 10% por período, el NPV efectivo es igual a $1735,54.

Ejemplo 3: múltiples pagos (script)

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Tasas de descuento para dos proyectos, que se cargan en una tabla denominada `Project`.
- Flujos de caja para múltiples períodos, para cada proyecto por ID de proyecto e ID de período. Este ID de período podría usarse para ordenar los registros en caso de que los datos no estén ordenados.
- La combinación de `NoConcatenate`, cargas residentes y la función `Left Join` para crear una tabla temporal, `tmpNPV`. La tabla combina los registros de las tablas `Project` y `CashFlow` en una tabla plana. Esta tabla tendrá tasas de descuento repetidas para cada período.
- Una carga residente de la tabla `tmpNPV`, que se usa para calcular el campo NPV para cada proyecto en una tabla denominada `NPV`.
- La tasa de descuento de valor único asociada a cada proyecto. Esto se recupera usando la función `only()` y se usa en el cálculo del NPV de cada proyecto.
- Una instrucción `Group By`, que se utiliza para agrupar todos los pagos de cada proyecto por ID de proyecto.

Para evitar que se carguen datos sintéticos o redundantes en el modelo de datos, la tabla `tmpNPV` se elimina al final del script.

Script de carga

```
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;
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- PrjId
- NPV

Tabla de resultados

PrjId	NPV
1	$2486.85
2	$2042.12

El Proyecto ID 1 espera recibir pagos de $1000 al final de tres períodos, a una tasa de descuento del 10% por período. Por lo tanto, el NPV efectivo es igual a $2486,85.

El Proyecto ID 2 espera dos pagos de $500 y dos pagos adicionales de $1000 en cuatro períodos a una tasa de descuento del 15%. Por lo tanto, el NPV efectivo es igual a $2042,12.

Ejemplo 4: ejemplo de rentabilidad de proyecto (script)

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Tasas de descuento e inversiones iniciales (período 0) para dos proyectos, cargados en una tabla denominada `Project`.
- Flujos de caja para múltiples períodos, para cada proyecto por ID de proyecto e ID de período. Este ID de período podría usarse para ordenar los registros en caso de que los datos no estén ordenados.
- La combinación de `NoConcatenate`, cargas residentes y la función `Left Join` para crear una tabla temporal, `tmpNPV`. La tabla combina los registros de las tablas `Project` y `CashFlow` en una tabla plana. Esta tabla tendrá tasas de descuento repetidas para cada período.
- La tasa de descuento de valor único asociada a cada proyecto, que se recupera mediante la función `only()` y se utiliza en el cálculo del NPV para cada proyecto.
- Se usa una carga residente de la tabla tmpNPV para calcular el campo NPV para cada proyecto en una tabla denominada NPV.
- Se crea un campo adicional que divide el NPV por la inversión inicial de cada proyecto para calcular el índice de rentabilidad del proyecto.
- Se utiliza una sentencia group by, que agrupa por ID de proyecto, para agrupar todos los pagos de cada proyecto.

Para evitar que se carguen datos sintéticos o redundantes en el modelo de datos, la tabla `tmpNPV` se elimina al final del script.

Script de carga

```
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;
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- PrjId
- NPV

Cree la siguiente medida:

```
=only(Profitability_Index)
```

Tabla de resultados

PrjId	NPV	=only(Profitability_Index)
1	$87039.82	0.87
2	$123513.71	1.24

El proyecto ID 1 tiene un NPV efectivo de 87.039,82 dólares y una inversión inicial de 100.000 dólares. Por lo tanto, el índice de rentabilidad es igual a 0,87. Como es inferior a 1, el proyecto no es rentable.

El proyecto ID 2 tiene un NPV efectivo de 123.513,71 dólares y una inversión inicial de 100.000 dólares. Por lo tanto, el índice de rentabilidad es igual a 1,24. Como es mayor que 1, el proyecto es rentable.

NPV - función de gráfico

NPV() devuelve el valor presente neto agregado de una inversión en función de una tasa **discount_rate** por período y una serie de pagos futuros (valores negativos) e ingresos (valores positivos), representados por los números en **value**, que se repiten por las dimensiones del gráfico. Se asume que los pagos e ingresos se producen al final de cada período.

Sintaxis:

```
NPV([TOTAL [<fld {,fld}>]] discount_rate, value)
```

Tipo de datos que devuelve: numérico De forma predeterminada, el resultado recibirá el formato de moneda.

Argumentos:

Argumentos

Argumento	Descripción
discount_rate	**discount_rate** es la tasa porcentual de descuento aplicada.
value	La expresión o el campo que contiene los datos que se han de medir.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles. El cualificador **TOTAL** puede ir seguido de una lista de uno o más nombres de campo entre paréntesis angulares. Estos nombres de campo deberían ser un subgrupo de las variables de dimensión del gráfico. En este caso, el cálculo se realiza ignorando todas las variables de dimensión del gráfico excepto las listadas, es decir, que devolverá un valor por cada combinación de valores de campo de los campos de dimensión listados. También los campos que no constituyan actualmente una dimensión de un gráfico pueden incluirse en la lista. Esto puede resultar útil en el caso de dimensiones de grupo, en las que los campos de dimensión no son fijos. Listar todas las variables del grupo hará que la función opere correctamente cuando el nivel jerárquico varíe.

Limitaciones:

discount_rate y **value** no deben contener funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Los valores de texto, valores NULL y valores perdidos se descartan.

Ejemplos y resultados:

Ejemplos y resultados

Ejemplo	Resultado
`NPV(Discount, Payments)`	-$540.12

Datos utilizados en los ejemplos:

```
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 '|');
```

Vea también:

- *XNPV - función de gráfico (page 400)*
- *Aggr - función de gráfico (page 559)*

XIRR

XIRR() devuelve la tasa interna de rendimiento agregada (anual) de una planificación de flujos de efectivo (no necesariamente periódicos) representados por números pareados en **pmt** y **date** que se repiten a lo largo de un número de registros, según lo definido por una cláusula group by. Todos los pagos son descontados según una base de un año de 365 días.

La funcionalidad XIRR de Qlik (funciones **XIRR()** y **RangeXIRR()**) utiliza la siguiente ecuación, resolviendo para el valor `Rate`, a fin de determinar el valor XIRR correcto:

```
XNPV(Rate, pmt, date) = 0
```

La ecuación se resuelve utilizando una versión simplificada del método de Newton.

Sintaxis:

```
XIRR(pmt, date )
```

Tipo de datos que devuelve: numérico

Argumentos

Argumento	Descripción
pmt	Pagos. La expresión o campo que contiene los flujos de caja correspondientes a la programación de pagos proporcionada en **date**.
date	La expresión o campo que contiene la planificación de fechas correspondientes a los pagos de efectivo proporcionados en **pmt**.

Al trabajar con esta función, se aplican las siguientes limitaciones:

- Los valores de texto, valores NULL y valores perdidos en una o ambas partes de un par de datos harán que se descarte el par de datos completo.
- Esta función requiere al menos un pago negativo válido y al menos un pago positivo válido (con las fechas válidas correspondientes). Si no se proporcionan estos pagos, devuelve un valor NULL.

Estos temas le ayudarán a trabajar con esta función:

- *XNPV (page 394)*: Utilice esta función para calcular el valor actual neto agregado para una programación de flujos de efectivo.
- *RangeXIRR (page 1386)*: **RangeXIRR()** es el equivalente de la función range para la función **XIRR()**.

En diferentes versiones de Qlik Sense administrado por el cliente, hay variaciones en el algoritmo que utiliza esta función. Para más información sobre actualizaciones recientes del algoritmo, consulte el artículo de ayuda Reparación y actualización de la función XIRR.

Ejemplo

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Datos de transacciones para una serie de flujos de efectivo.
- El uso de la función **XIRR()** para calcular la tasa interna de rendimiento anual de estos flujos de efectivo.

Script de carga

```
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;
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `Year`
- `XIRR2013`

Tabla de resultados

Año	XIRR2013
2013	0.5385

Interpretación del valor de retorno XIRR

La funcionalidad XIRR generalmente se utiliza para analizar una inversión, donde hay un pago saliente (negativo) al principio, y luego una serie de pagos de ingresos más pequeños (positivos) más adelante. Aquí tiene un ejemplo simplificado con solo un pago negativo y uno positivo:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|-100
2024-01-01|110
] (delimiter is '|');
```

Hacemos un pago inicial de 100 y recuperamos 110 después de exactamente un año. Esto representa una tasa de rendimiento del 10% anual. `XIRR(Payments, Date)` devuelve un valor de 0,1.

El valor de retorno de la funcionalidad XIRR puede ser positivo o negativo. En el caso de una inversión, un resultado negativo indicaría que la inversión es una pérdida. La cantidad de ganancia o pérdida se puede calcular simplemente haciendo una suma agregada sobre el campo de pagos.

En el ejemplo anterior, estamos prestando nuestro dinero por un año. La tasa de rendimiento se puede considerar como interés. También es posible usar la funcionalidad de XIRR cuando estamos del otro lado de la transacción (por ejemplo, si somos el prestatario en lugar del prestamista).

Vea este ejemplo:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|100
2024-01-01|-110
] (delimiter is '|');
```

Es igual que el primer ejemplo, pero invertido. Aquí, tomamos prestados 100 por un año y lo devolvemos con un 10% de interés. En este ejemplo, el cálculo de XIRR devuelve 0,1 (10 %), el mismo valor que en el primer ejemplo.

Tenga en cuenta que en el primer ejemplo recibimos una ganancia de 10 y en el segundo ejemplo experimentamos una pérdida de 10, pero el valor de rendimiento de la funcionalidad XIRR es positivo para ambos ejemplos. Esto se debe a que la funcionalidad XIRR calcula el interés oculto en la transacción, sin importar de qué lado se encuentre en la transacción.

Limitaciones con múltiples soluciones

La funcionalidad XIRR de Qlik se define mediante la siguiente ecuación, en la que se resuelve el valor de `Rate`:

```
XNPV(Rate, pmt, date) = 0
```

A veces es posible que esta ecuación tenga más de una solución. Esto se conoce como el "problema de múltiples TIR" y es ocasionado por un flujo de efectivo no normal (también llamado flujo de efectivo no convencional). El siguiente script de carga muestra un ejemplo de ello:

```
Cashflow:
LOAD * inline [
Date|Payments
2021-01-01|-200
2022-01-01|500
2023-01-01|-250
] (delimiter is '|');
```

En este ejemplo, hay una solución negativa y una solución positiva. (`Rate` = -0,3 y `Rate` = 0,8). **XIRR()** devolverá 0,8.

Cuando la funcionalidad XIRR de Qlik busca una solución, comienza en `Rate` = 0 y aumenta la tasa en pasos hasta que encuentra una solución. Si hay más de una solución positiva, devolverá la primera que encuentre. Si no puede encontrar una solución positiva, reiniciará `Rate` de nuevo en cero y comenzará a buscar una solución en la dirección negativa.

Tenga en cuenta que se garantiza que una corriente de flujo de efectivo "normal" tiene una sola solución. El flujo de efectivo "normal" significa que todos los pagos con el mismo signo (positivo o negativo) están en un grupo continuo.

Vea también:

- *XNPV (page 394)*
- *RangeXIRR (page 1386)*
- XIRR function Fix and Update

XIRR - función de gráfico

XIRR() devuelve la tasa interna de rendimiento agregada (anual) de una planificación de flujos de efectivo (no necesariamente periódicos) representados por números pareados en las expresiones proporcionadas por **pmt** y **date**, que se repiten a lo largo de las dimensiones del gráfico. Todos los pagos son descontados según una base de un año de 365 días.

La funcionalidad XIRR de Qlik (funciones **XIRR()** y **RangeXIRR()**) utiliza la siguiente ecuación, resolviendo para el valor `Rate`, a fin de determinar el valor XIRR correcto:

```
XNPV(Rate, pmt, date) = 0
```

La ecuación se resuelve utilizando una versión simplificada del método de Newton.

Sintaxis:

```
XIRR([TOTAL [<fld {,fld}>]] pmt, date)
```

Tipo de datos que devuelve: numérico

Argumentos

Argumento	Descripción
pmt	Pagos. La expresión o campo que contiene los flujos de caja correspondientes a la programación de pagos proporcionada en **date**.
date	La expresión o campo que contiene la planificación de fechas correspondientes a los pagos de efectivo proporcionados en **pmt**.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Al trabajar con esta función, se aplican las siguientes limitaciones:

- **pmt** y **date** no deben contener funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.
- Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.
- Esta función requiere al menos un pago negativo válido y al menos un pago positivo válido (con las fechas válidas correspondientes). Si no se proporcionan estos pagos, devuelve un valor NULL.

Estos temas le ayudarán a trabajar con esta función:

- *XNPV - función de gráfico (page 400)*: Utilice esta función para calcular el valor actual neto agregado para una programación de flujos de efectivo.
- *RangeXIRR (page 1386)*: **RangeXIRR()** es el equivalente de la función range para la función **XIRR()**.

En diferentes versiones de Qlik Sense administrado por el cliente, hay variaciones en el algoritmo que utiliza esta función. Para más información sobre actualizaciones recientes del algoritmo, consulte el artículo de ayuda Reparación y actualización de la función XIRR.

Ejemplo

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene transacciones de flujos de efectivo.
- La información se almacena en una tabla llamada `Cashflow`.

Script de carga

```
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 '|');
```

Resultados

Haga lo siguiente:

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue el siguiente cálculo como medida:

```
=XIRR(Payments, Date)
```

Tabla de resultados

=XIRR(Payments, Date)
0.5385

Interpretación del valor de retorno XIRR

La funcionalidad XIRR generalmente se utiliza para analizar una inversión, donde hay un pago saliente (negativo) al principio, y luego una serie de pagos de ingresos más pequeños (positivos) más adelante. Aquí tiene un ejemplo simplificado con solo un pago negativo y uno positivo:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|-100
2024-01-01|110
] (delimiter is '|');
```

Hacemos un pago inicial de 100 y recuperamos 110 después de exactamente un año. Esto representa una tasa de rendimiento del 10% anual. `XIRR(Payments, Date)` devuelve un valor de 0,1.

El valor de retorno de la funcionalidad XIRR puede ser positivo o negativo. En el caso de una inversión, un resultado negativo indicaría que la inversión es una pérdida. La cantidad de ganancia o pérdida se puede calcular simplemente haciendo una suma agregada sobre el campo de pagos.

En el ejemplo anterior, estamos prestando nuestro dinero por un año. La tasa de rendimiento se puede considerar como interés. También es posible usar la funcionalidad de XIRR cuando estamos del otro lado de la transacción (por ejemplo, si somos el prestatario en lugar del prestamista).

Vea este ejemplo:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|100
2024-01-01|-110
] (delimiter is '|');
```

Es igual que el primer ejemplo, pero invertido. Aquí, tomamos prestados 100 por un año y lo devolvemos con un 10% de interés. En este ejemplo, el cálculo de XIRR devuelve 0,1 (10 %), el mismo valor que en el primer ejemplo.

Tenga en cuenta que en el primer ejemplo recibimos una ganancia de 10 y en el segundo ejemplo experimentamos una pérdida de 10, pero el valor de rendimiento de la funcionalidad XIRR es positivo para ambos ejemplos. Esto se debe a que la funcionalidad XIRR calcula el interés oculto en la transacción, sin importar de qué lado se encuentre en la transacción.

Limitaciones con múltiples soluciones

La funcionalidad XIRR de Qlik se define mediante la siguiente ecuación, en la que se resuelve el valor de `Rate`:

```
XNPV(Rate, pmt, date) = 0
```

A veces es posible que esta ecuación tenga más de una solución. Esto se conoce como el "problema de múltiples TIR" y es ocasionado por un flujo de efectivo no normal (también llamado flujo de efectivo no convencional). El siguiente script de carga muestra un ejemplo de ello:

```
Cashflow:
LOAD * inline [
Date|Payments
2021-01-01|-200
2022-01-01|500
2023-01-01|-250
] (delimiter is '|');
```

En este ejemplo, hay una solución negativa y una solución positiva. (`Rate` = -0,3 y `Rate` = 0,8). **XIRR()** devolverá 0,8.

Cuando la funcionalidad XIRR de Qlik busca una solución, comienza en `Rate` = 0 y aumenta la tasa en pasos hasta que encuentra una solución. Si hay más de una solución positiva, devolverá la primera que encuentre. Si no puede encontrar una solución positiva, reiniciará `Rate` de nuevo en cero y comenzará a buscar una solución en la dirección negativa.

Tenga en cuenta que se garantiza que una corriente de flujo de efectivo "normal" tiene una sola solución. El flujo de efectivo "normal" significa que todos los pagos con el mismo signo (positivo o negativo) están en un grupo continuo.

Vea también:

IRR - función de gráfico (page 377)
Aggr - función de gráfico (page 559)
XIRR function Fix and Update

XNPV

XNPV() devuelve el valor actual neto agregado de una planificación de flujos de efectivo (no necesariamente periódicos) representado por números pareados en **pmt** y **date**. Todos los pagos son descontados según una base de un año de 365 días.

Sintaxis:

```
XNPV(discount_rate, pmt, date)
```

Tipo de datos que devuelve: numérico

De forma predeterminada, el resultado recibirá el formato de moneda.

La fórmula para calcular el XNPV se muestra a continuación:

Fórmula de agregación de XNPV

$$XNPV = \sum_{i=1}^{n} \frac{P_i}{(1+rate)^{(di-d1)/365}}$$

donde:

- P_i = Entradas y salidas de efectivo netas durante un solo periodo `i`
- d_1 = la primera fecha de pago
- d_i = la `i`ª fecha de pago
- `rate` = la tasa de descuento

El valor actual neto, VAN (NPV en inglés), se utiliza para calcular el valor total actual de una corriente futura de flujos de efectivo dada una tasa de descuento. Para calcular el XNPV, necesitamos estimar los flujos de efectivo futuros con las fechas correspondientes. Después de esto, para cada pago, aplicamos la tasa de descuento compuesta basada en la fecha del pago.

Realizar la agregación XNPV sobre una serie de pagos es similar a realizar una agregación Sum sobre esos pagos. La diferencia es que cada cantidad es modificada (o “descontada”) de acuerdo a la tasa de descuento elegida (similar a la tasa de interés) y a cuánto tiempo está el pago en el futuro. Realizar el XNPV con el

parámetro **discount_rate** configurado en cero hará que XNPV sea equivalente a una operación Sum (los pagos no se modificarán antes de sumarlos). Por lo general, cuanto más se acerque a cero la tasa de descuento **discount_rate**, más similar será el resultado de XNPV al de una agregación Sum.

Argumentos

Argumento	Descripción
discount_rate	**discount_rate** es la tasa anual a la que se deben descontar los pagos. Un valor de 0,1 indicaría una tasa de descuento del 10%.
pmt	Pagos. La expresión o campo que contiene los flujos de caja correspondientes a la programación de pagos proporcionada en **date**. Se da por sentado que los valores positivos son entradas de efectivo y los valores negativos son salidas de efectivo. ***XNPV()*** *no descuenta el flujo de caja inicial ya que siempre ocurrirá en la fecha de inicio. Los pagos posteriores se descuentan sobre una base anual de 365 días. Esto es diferente de* ***NPV()****, donde también se descuenta el primer pago.*
date	La expresión o campo que contiene la planificación de fechas correspondientes a los pagos de efectivo proporcionados en **pmt**. El primer valor se utiliza como fecha de inicio para calcular las compensaciones de los flujos de caja futuros.

Al trabajar con esta función, se aplican las siguientes limitaciones:

- Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Cuándo se utiliza

- `XNPV()` se utiliza en modelos financieros para calcular el valor actual neto (VAN o NPV en inglés) de una oportunidad de inversión.
- Debido a su mayor precisión, se prefiere XNPV sobre NPV para todo tipo de modelos financieros.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: pago único (script)

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de un proyecto y su flujo de caja durante un año, cargado en una tabla denominada `CashFlow`. La fecha inicial para el cálculo se establece en el 1 de julio de 2022, con un flujo de caja neto de 0. Después de un año, se produce un flujo de caja de $1000.
- Una carga residente de la tabla `CashFlow`, que se utiliza para calcular el campo `XNPV` del proyecto en una tabla denominada `XNPV`.
- Se utiliza una tasa de descuento codificada del 10% (0,1) en el cálculo del XNPV.
- Se utiliza una sentencia `Group By` para agrupar todos los pagos del proyecto.

Script de carga

```
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;
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `PrjId`
- `XNPV`

Tabla de resultados

PrjId	XNPV
1	$909.09

Según la fórmula, el valor XNPV para el primer registro es 0 y para el segundo registro, el valor XNPV es $909,09 . Por lo tanto, el XNPV total es de $909,09 .

Ejemplo 2: múltiples pagos (script)

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de un proyecto y su flujo de caja durante un año, cargado en una tabla denominada `CashFlow`.
- Una carga residente de la tabla `CashFlow`, que se utiliza para calcular el campo `XNPV` del proyecto en una tabla denominada `XNPV`.
- Se utiliza una tasa de descuento codificada del 10% (0,1) en el cálculo del XNPV.
- Se utiliza una sentencia `Group By` para agrupar todos los pagos del proyecto.

Script de carga

```
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;
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `PrjId`
- `XNPV`

Tabla de resultados

PrjId	XNPV
1	$1322.21

En este ejemplo, se recibe un pago de $1000 al final del primer año y un pago de $500 al final del segundo año. Con una tasa de descuento del 10% por período, el XNPV efectivo es igual a $1322,21.

Tenga en cuenta que solo la primera fila de datos debe hacer referencia a la fecha base para los cálculos. Para el resto de las filas, el orden no es importante, ya que el parámetro de fecha se usa para calcular el período transcurrido.

Ejemplo 3 – Múltiples pagos y flujos de caja irregulares (script)

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Tasas de descuento para dos proyectos en una tabla llamada `Project`.
- Flujos de caja de múltiples períodos para cada proyecto por ID de proyecto y fechas. Se utiliza el campo `Dates` para calcular la duración durante la cual se aplica la tasa de descuento al flujo de caja. Aparte del primer registro (flujo de caja inicial y fecha), el orden de los registros no es importante y cambiarlo no debería afectar los cálculos.
- Mediante una combinación de `NoConcatenate`, cargas residentes y la función `Left Join`, se crea una tabla temporal, `tmpNPV`que combina los registros de las tablas `Project` y `CashFlow` en una tabla plana. Esta tabla tendrá tasas de descuento repetidas para cada flujo de caja.
- Una carga residente de la tabla `tmpNPV`, que se utiliza para calcular el campo `XNPV` para cada proyecto en una tabla denominada `XNPV` .
- La tasa de descuento de valor único asociada a cada proyecto, que se recupera mediante la función `only()` y se utiliza en el cálculo del `XNPV` para cada proyecto.
- Se utiliza una sentencia `Group By`, que agrupa por ID de proyecto, para agrupar todos los pagos y fechas correspondientes para cada proyecto.
- Para evitar que se carguen datos sintéticos o redundantes en el modelo de datos, la tabla `tmpXNPV` se elimina al final del script.

Script de carga

```
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;
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- PrjId
- XNPV

Tabla de resultados

PrjId	XNPV
1	$1735.54
2	$278.36

El proyecto ID 1 tiene un flujo de caja inicial de $0 el 1 de julio de 2021. Hay dos pagos de $1000 que se han de recibir al final de dos años subsiguientes, a una tasa de descuento del 10% por período. Por lo tanto, el NPV efectivo es igual a $1735,54.

El proyecto ID 2 tiene una salida inicial de $1000 (por lo tanto, el signo negativo) el 1 de julio de 2020. Después de dos años, se espera un pago de $500. Después de tres años, se espera un pago de $500. Finalmente, el 1 de julio de 2024 se espera un pago de $1000. Con la tasa de descuento del 15%, el XNPV efectivo es igual a $278,36.

Vea también:

Drop table (page 153)

- *group by (page 163)*
- *Join (page 73)*
- *Max (page 341)*
- *NoConcatenate (page 91)*
- *NPV - función de gráfico (page 386)*
- *Only (page 351)*

XNPV - función de gráfico

XNPV() devuelve el valor presente neto agregado de una planificación de flujos de efectivo (no necesariamente periódicos) representados por números pareados de las expresiones dadas por **pmt** y **date**, que se repiten por las dimensiones del gráfico. Todos los pagos son descontados según una base de un año de 365 días.

Sintaxis:

```
XNPV([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

Tipo de datos que devuelve: numérico

De forma predeterminada, el resultado recibirá el formato de moneda.

La fórmula para calcular el XNPV se muestra a continuación:

Fórmula de agregación de XNPV

$$XNPV = \sum_{i=1}^{n} \frac{P_i}{(1+rate)^{(di-d1)/365}}$$

donde:

- P_i = Entradas y salidas de efectivo netas durante un solo periodo i
- d_1 = la primera fecha de pago
- d_i = la i^a fecha de pago
- `rate` = la tasa de descuento

El valor actual neto, VAN (NPV en inglés), se utiliza para calcular el valor total actual de una corriente futura de flujos de efectivo dada una tasa de descuento. Para calcular el XNPV, necesitamos estimar los flujos de efectivo futuros con las fechas correspondientes. Después de esto, para cada pago, aplicamos la tasa de descuento compuesta basada en la fecha del pago.

Realizar la agregación XNPV sobre una serie de pagos es similar a realizar una agregación Sum sobre esos pagos. La diferencia es que cada cantidad es modificada (o “descontada”) de acuerdo a la tasa de descuento elegida (similar a la tasa de interés) y a cuánto tiempo está el pago en el futuro. Realizar el XNPV con el parámetro **discount_rate** configurado en cero hará que XNPV sea equivalente a una operación Sum (los pagos no se modificarán antes de sumarlos). Por lo general, cuanto más se acerque a cero la tasa de descuento **discount_rate**, más similar será el resultado de XNPV al de una agregación Sum.

Argumentos

Argumento	Descripción
discount_rate	**discount_rate** es la tasa anual a la que se deben descontar los pagos. Un valor de 0,1 indicaría una tasa de descuento del 10%.
pmt	Pagos. La expresión o campo que contiene los flujos de caja correspondientes a la programación de pagos proporcionada en **date**. Se da por sentado que los valores positivos son entradas de efectivo y los valores negativos son salidas de efectivo. ***XNPV()*** *no descuenta el flujo de caja inicial ya que siempre ocurrirá en la fecha de inicio. Los pagos posteriores se descuentan sobre una base anual de 365 días. Esto es diferente de* ***NPV()****, donde también se descuenta el primer pago.*
date	La expresión o campo que contiene la planificación de fechas correspondientes a los pagos de efectivo proporcionados en **pmt**. El primer valor se utiliza como fecha de inicio para calcular las compensaciones de los flujos de caja futuros.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Al trabajar con esta función, se aplican las siguientes limitaciones:

- **discount_rate**, **pmt** y **date** no deben contener funciones de agregación, a menos que dichas agregaciones internas contengan los cualificadores **TOTAL** o **ALL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.
- Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Cuándo se utiliza

- `XNPV()` se utiliza en modelos financieros para calcular el valor actual neto (VAN o NPV en inglés) de una oportunidad de inversión.
- Debido a su mayor precisión, se prefiere XNPV sobre NPV para todo tipo de modelos financieros.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene transacciones de flujos de efectivo.
- La información se almacena en una tabla llamada `Cashflow`.

Script de carga

```
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 '|');
```

Resultados

Haga lo siguiente:

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue el siguiente cálculo como medida:

```
=XNPV(0.09, Payments, Date)
```

Tabla de resultados

=XNPV(0.09, Payments, Date)
$3062.49

Vea también:

NPV - función de gráfico (page 386)
Aggr - función de gráfico (page 559)

Funciones de agregación estadística

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Funciones de agregación estadística en el script de carga de datos

Se pueden utilizar las siguientes funciones de agregación estadística en scripts.

Avg

Avg() halla el valor promedio de los datos agregados en la expresión a lo largo de un número de registros según lo definido por una cláusula **group by**.

```
Avg  ([distinct] expression)
```

Correl

Correl() devuelve el coeficiente de correlación agregado de una serie de coordenadas representadas por números pareados en x-expression y y-expression que se repiten por una serie de registros según lo definido por una cláusula **group by**.

```
Correl (x-expression, y-expression)
```

Fractile

Fractile() halla el valor que corresponde al fractil (cuantil) completo de los datos agregados de la expresión que se repite por un número de registros definido por una cláusula **group by**.

```
Fractile (expression, fractile)
```

FractileExc

FractileExc() halla el valor que corresponde al fractil (cuantil) exclusivo de los datos agregados de la expresión que se repite por un número de registros definido por una cláusula **group by**.

```
FractileExc (expression, fractile)
```

Kurtosis

Kurtosis() devuelve la kurtosis de los datos de la expresión iterada sobre una serie de registros según lo definido por una cláusula **group by**.

```
Kurtosis  ([distinct ] expression )
```

LINEST_B

LINEST_B() devuelve el valor b agregado (intercepción en y) de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por números pareados en x-expression y y-expression que se repiten por una serie de registros según lo definido por una cláusula **group by**.

```
LINEST_B (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_df

LINEST_DF() devuelve los grados de libertad agregados de una regresión lineal definida por la ecuación y=mx+b para una serie de coordenadas representadas por números pareados en x-expression y y-expression que se repiten por un número de registros definido por una cláusula **group by**.

```
LINEST_DF  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_f

Esta función de script devuelve el estadístico F agregado ($r^2/(1-r^2)$) de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por números pareados en x-expression y y-expression que se repiten a lo largo de una serie de registros, según lo definido por una cláusula **group by**.

```
LINEST_F (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_m

LINEST_M() devuelve el valor m agregado (pendiente) de una regresión lineal definida por la ecuación y=mx+b para una serie de coordenadas representadas por los números pareados en x-expression y y-expression que se repiten por un número de registros según lo definido por una cláusula **group by**.

```
LINEST_M  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_r2

LINEST_R2() devuelve el valor agregado r^2 (coeficiente de determinación) de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por números pareados en x-expression y y-expression que se repiten en una serie de registros según lo definido por una cláusula **group by**.

```
LINEST_R2 (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_seb

LINEST_SEB() devuelve el error estándar agregado del valor b de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados en x-expression y y-expression que se repiten a lo largo de una serie de registros según lo definido por una cláusula **group by**.

```
LINEST_SEB  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sem

LINEST_SEM() devuelve el error estándar agregado del valor m de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados en x-expression y y-expression que se repiten a lo largo de una serie de registros según lo definido por una cláusula **group by**.

```
LINEST_SEM  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sey

LINEST_SEY() devuelve el error estándar agregado de la estimación y de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados en x-expression y y-expression que se repiten por un número de registros según lo definido por una cláusula **group by**.

```
LINEST_SEY  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_ssreg

LINEST_SSREG() devuelve la suma de cuadrados de regresión agregada de una regresión lineal definida por la ecuación y=mx+b para una serie de coordenadas representadas por los números pareados en x-expression y y-expression que se repiten sobre un número de registros conforme a lo definido por una cláusula **group by**.

```
LINEST_SSREG  (y-expression, x-expression [, y0 [, x0 ]])
```

Linest_ssresid

LINEST_SSRESID() devuelve la suma residual agregada de cuadrados de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados en x-expression y y-expression que se repiten por un conjunto de registros según lo definido por una cláusula **group by**.

```
LINEST_SSRESID  (y-expression, x-expression [, y0 [, x0 ]])
```

Median

Median() devuelve la mediana agregada de los valores de la expresión que se repiten sobre un número de registros conforme a lo definido por una cláusula **group by**.

```
Median (expression)
```

Skew

Skew() devuelve la asimetría de la expresión en una serie de registros según lo definido por una cláusula **group by**.

```
Skew ([ distinct] expression)
```

Stdev

Stdev() devuelve la desviación estándar de los valores dados por la expresión en una serie de registros según lo definido por una cláusula **group by**.

```
Stdev ([distinct] expression)
```

Sterr

Sterr() devuelve el error estándar agregado (stdev/sqrt(n)) de una serie de valores representados por la expresión iterada sobre un número de registros, según lo definido por una cláusula **group by**.

```
Sterr  ([distinct] expression)
```

STEYX

STEYX() devuelve el error estándar agregado del valor y pronosticado para cada valor de x en la regresión de una serie de coordenadas representadas por los números pareados en x-expression y y-expression que se repiten a lo largo de un número de registros, según lo definido por una cláusula **group by**.

```
STEYX  (y-expression, x-expression)
```

Funciones de agregación estadística en expresiones de gráficos

Se pueden utilizar las siguientes funciones de agregación estadística en gráficos:

Avg

Avg() devuelve el promedio agregado de la expresión o campo que se repite por las dimensiones del gráfico.

```
Avg - función de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]}
expr)
```

Correl

Correl() devuelve el coeficiente de correlación agregado de dos conjuntos de datos. La función de correlación es una medida de la relación entre los conjuntos de datos y se agrega para pares de valores (x,y) que se repiten por las dimensiones del gráfico.

```
Correl - función de gráfico({[SetExpression] [TOTAL [<fld {, fld}>]]} value1,
value2 )
```

Fractile

Fractile() halla el valor que corresponde al fractil (cuantil) completo de los datos agregados en el rango proporcionado por la expresión que se repite por las dimensiones del gráfico.

```
Fractile - función de gráfico({[SetExpression] [TOTAL [<fld {, fld}>]]} expr,
fraction)
```

FractileExc

FractileExc() halla el valor que corresponde al fractil exclusivo (cuantil) de los datos agregados en el rango proporcionado por la expresión que se repite por las dimensiones del gráfico.

```
FractileExc - función de gráfico({[SetExpression] [TOTAL [<fld {, fld}>]]}
expr, fraction)
```

Kurtosis

Kurtosis() halla la kurtosis del rango de datos agregados en la expresión o campo que se repite por las dimensiones del gráfico.

```
Kurtosis - función de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld{,
fld}>]]} expr)
```

LINEST_b

LINEST_B() devuelve el valor b agregado (intercepción en y) de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por números pareados en las expresiones dadas por las expresiones **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

```
LINEST_R2 - función de gráfico({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_df

LINEST_DF() devuelve los grados agregados de libertad de una regresión lineal definida por la ecuación y=mx+b para una serie de coordenadas representadas por números pareados en las expresiones dadas por **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

```
LINEST_DF - función de gráfico({[SetExpression] [TOTAL [<fld{, fld}>]]} y_
value, x_value [, y0_const [, x0_const]])
```

LINEST_f

LINEST_F() devuelve el estadístico F agregado (r2/(1-r2)) de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por números pareados en las expresiones dadas por **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

```
LINEST_F - función de gráfico({[SetExpression] [TOTAL[<fld{, fld}>]]} y_
value, x_value [, y0_const [, x0_const]])
```

LINEST_m

LINEST_M() devuelve el valor agregado m (pendiente) de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por números pareados dados por las expresiones **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

```
LINEST_M - función de gráfico({[SetExpression] [TOTAL[<fld{, fld}>]]} y_
value, x_value [, y0_const [, x0_const]])
```

LINEST_r2

LINEST_R2() devuelve el valor agregado r2 (coeficiente de determinación) de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por números pareados dados por las expresiones **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

```
LINEST_R2 - función de gráfico({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_seb

LINEST_SEB() devuelve el error estándar agregado del valor b de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados proporcionados por las expresiones **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

```
LINEST_SEB - función de gráfico({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_sem

LINEST_SEM() devuelve el error estándar agregado del valor m de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados proporcionados por las expresiones **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

```
LINEST_SEM - función de gráfico([{set_expression}][ distinct ] [total [<fld
{,fld}>] ] y-expression, x-expression [, y0 [, x0 ]] )
```

LINEST_sey

LINEST_SEY() devuelve el error estándar agregado del cálculo y de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados proporcionados por las expresiones **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

```
LINEST_SEY - función de gráfico({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssreg

LINEST_SSREG() devuelve la suma de cuadrados de regresión agregada de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados proporcionados por las expresiones **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

```
LINEST_SSREG - función de gráfico({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssresid

LINEST_SSRESID() devuelve la suma residual agregada de cuadrados de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados de las expresiones dados por **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

```
LINEST_SSRESID - función de gráficoLINEST_SSRESID() devuelve la suma residual
agregada de cuadrados de una regresión lineal definida por la ecuación y=mx+b
de una serie de coordenadas representadas por los números pareados de las
expresiones dados por x_value y y_value, que se repiten por las dimensiones
del gráfico.  LINEST_SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{,
fld}>]] y_value, x_value[, y0_const[, x0_const]])
numérico ArgumentosArgumentoDescripcióny_valueLa expresión o campo que
contiene el rango de valores y que se han de medir.x_valueLa expresión o
campo que contiene el rango de valores x que se han de medir.y0, x0Se puede
establecer un valor y0 opcional que obligue a la línea de regresión a pasar a
través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible
forzar la línea de regresión para que pase a través de una sola coordenada
fija. A menos que se definan ambos, y0 y x0, la función requiere al menos dos
pares de datos válidos para calcular. Si se definen y0 y x0, un único par de
datos servirá. SetExpressionDe forma predeterminada, la función de agregación
agregará sobre el conjunto de registros posibles definidos por la selección.
Se puede definir un conjunto alternativo de registros mediante una expresión
de análisis de conjuntos. DISTINCTSi la palabra DISTINCT aparece antes de los
argumentos de la función, los duplicados resultantes de evaluar los
argumentos de la función se descartan. TOTALSi la palabra TOTAL aparece antes
de los argumentos de la función, el cálculo se realiza sobre todos los
valores posibles dadas las selecciones actuales y no solo aquellas que
pertenecen al valor dimensional actual, es decir, no tiene en cuenta las
dimensiones del gráfico. Usar TOTAL [<fld {.fld}>], donde al cualificador
TOTAL le sigue una lista de uno o más nombres de campo como un subconjunto de
las variables de dimensión del gráfico, crea un subconjunto de los valores
totales posibles.Se puede establecer un valor y0 opcional que obligue a la
línea de regresión a pasar a través del eje y en un punto determinado.
Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase
a través de una sola coordenada fija.  El parámetro de la función de
agregación no debe contener otras funciones de agregación, a menos que dichas
agregaciones internas contengan el cualificador TOTAL. Para agregaciones
anidadas más avanzadas, utilice la función avanzada Aggr, en combinación con
una dimensión especificada. Los valores de texto, valores NULL y los valores
```

```
perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.  An example of how to use linest functionsavg({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_value, x_value [, y0_const[, x0_const]])
```

Median

Median() devuelve el valor de la mediana del rango de valores agregados en la expresión que se repite por las dimensiones del gráfico.

```
Median - función de gráfico({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MutualInfo

MutualInfo calcula la información mutua (IM) entre dos campos o entre los valores agregados en **Aggr()**.

```
MutualInfo - función de gráfico{[SetExpression] [DISTINCT] [TOTAL target, driver [, datatype [, breakdownbyvalue [, samplesize ]]])
```

Skew

Skew() devuelve la asimetría agregada de la expresión o campo que se repite por las dimensiones del gráfico.

```
Skew - función de gráfico{[SetExpression] [DISTINCT] [TOTAL [<fld{ ,fld}>]]} expr)
```

Stdev

Stdev() halla la desviación estándar del rango de datos agregados en la expresión o campo que se repite por las dimensiones del gráfico.

```
Stdev - función de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} expr)
```

Sterr

Sterr() halla el valor del error estándar de la media, (stdev/sqrt(n)), de la serie de valores agregados en la expresión que se repite por las dimensiones del gráfico.

```
Sterr - función de gráfico({[SetExpression] [DISTINCT] [TOTAL[<fld{, fld}>]]} expr)
```

STEYX

STEYX() devuelve el error estándar agregado al predecir valores y para cada valor x de una regresión lineal dada por una serie de coordenadas representadas por números pareados en las expresiones dadas por **y_value** y **x_value**.

```
STEYX - función de gráfico{[SetExpression] [TOTAL [<fld{, fld}>]]} y_value, x_value)
```

Avg

Avg() halla el valor promedio de los datos agregados en la expresión a lo largo de un número de registros según lo definido por una cláusula **group by**.

Sintaxis:

```
Avg([DISTINCT] expr)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
DISTINCT	Si la palabra **distinct** figura antes de la expresión, todos los duplicados se descartan.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

Datos resultantes

Ejemplo	Resultado

`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`
	Esto puede verificarse en la hoja mediante la creación de una tabla que incluya la medida.
	`Sum(Sales)/12`

Dado que la tabla **Temp** se carga como en el ejemplo anterior:
 	
`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`
	Solo se cuentan los valores distintos. Divida el total por el número de valores no duplicados.

Avg - función de gráfico

Avg() devuelve el promedio agregado de la expresión o campo que se repite por las dimensiones del gráfico.

Sintaxis:

```
Avg([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Ejemplos y resultados:

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

Ejemplos de funciones

Ejemplo	Resultado
`Avg(Sales)`	Para una tabla que incluya la dimensión `Customer` y la medida `Avg([Sales])`, si se muestran **Totales**, el resultado es 2566.
`Avg([TOTAL (Sales))`	53.458333 para todos los valores de `Customer`, porque el cualificador TOTAL implica que las dimensiones se ignoran.
`Avg(DISTINCT (Sales))`	51.862069 para el total, porque usar el cualificador Distinct implica que solo se evalúan valores únicos de `Sales` por cada `Customer`.

Datos utilizados en los ejemplos:

```
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 '|');
```

Vea también:

Aggr - función de gráfico (page 559)

Correl

Correl() devuelve el coeficiente de correlación agregado de una serie de coordenadas representadas por números pareados en x-expression y y-expression que se repiten por una serie de registros según lo definido por una cláusula **group by**.

Sintaxis:

```
Correl(value1, value2)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value1, value2	Las expresiones o campos que contienen los dos conjuntos de muestra para los que se ha de medir el coeficiente de correlación.

Limitaciones:

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

Datos resultantes

Ejemplo	Resultado
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;	En una tabla con la dimensión `Correl_Salary`, se mostrará el resultado del cálculo de Correl() en el script de carga de datos: 0.9270611

Correl - función de gráfico

Correl() devuelve el coeficiente de correlación agregado de dos conjuntos de datos. La función de correlación es una medida de la relación entre los conjuntos de datos y se agrega para pares de valores (x,y) que se repiten por las dimensiones del gráfico.

Sintaxis:

```
Correl([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] value1, value2 )
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value1, value2	Las expresiones o campos que contienen los dos conjuntos de muestra para los que se ha de medir el coeficiente de correlación.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplo	Resultado
`Correl (Age, Salary)`	Para una tabla que incluya la dimensión `Employee name` y la medida `Correl(Age, Salary)`, el resultado es 0.9270611. El resultado solo se muestra para la celda de totales.

Ejemplo	Resultado
`Correl (TOTAL Age, Salary))`	0.927. Este resultado junto con los siguientes se muestran con tres decimales para una mejor legibilidad. Si crea un panel de filtrado con la dimensión Gender, y hace selecciones desde el mismo, verá el resultado 0.951 cuando Female esté seleccionado y 0.939 si Male está seleccionado. Esto se debe a que la selección excluye todos los resultados que no pertenecen al otro valor de Gender.
`Correl({1} TOTAL Age, Salary))`	0.927. Independiente de las selecciones. Esto es porque la expresión de conjunto {1} ignora todas las selecciones y dimensiones.
`Correl (TOTAL <Gender> Age, Salary))`	0.927 en la celda del total, 0.939 para todos los valores de Male y 0.951 para todos los valores de Female. Esto corresponde a los resultados de hacer las selecciones en un panel de filtrado basado en Gender.

Datos utilizados en los ejemplos:

```
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 '|');
```

Vea también:

Aggr - función de gráfico (page 559)

- *Avg - función de gráfico (page 411)*
- *RangeCorrel (page 1354)*

Fractile

Fractile() halla el valor que corresponde al fractil (cuantil) completo de los datos agregados de la expresión que se repite por un número de registros definido por una cláusula **group by**.

Puede usar FractileExc (page 420) para calcular el fractil exclusivo.

Sintaxis:

```
Fractile(expr, fraction)
```

Tipo de datos que devuelve: numérico

La función devuelve el valor correspondiente al rango definido por `rank = fraction * (N-1) + 1` donde `N` es el número de valores en `expr`. Si `rank` es un número no entero, se realiza una interpolación entre los dos valores más cercanos.

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o campo que contiene los datos que se deben utilizar al calcular el fractil.
fraction	Un número entre 0 y 1 correspondiente al percentil (cuantil expresado como fracción) que se debe calcular.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

Datos resultantes

Ejemplo	Resultado
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;	En una tabla con las dimensiones Type y MyFractile, los resultados de los cálculos de Fractile() en el script de carga de datos son: Type MyFractile Comparison 27.5 Observation 36

Fractile - función de gráfico

Fractile() halla el valor que corresponde al fractil (cuantil) completo de los datos agregados en el rango proporcionado por la expresión que se repite por las dimensiones del gráfico.

Puede usar FractileExc - función de gráfico (page 422) para calcular el fractil exclusivo.

Sintaxis:

```
Fractile([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr, fraction)
```

Tipo de datos que devuelve: numérico

La función devuelve el valor correspondiente al rango definido por `rank = fraction * (N-1) + 1` donde `N` es el número de valores en `expr`. Si `rank` es un número no entero, se realiza una interpolación entre los dos valores más cercanos.

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o campo que contiene los datos que se deben utilizar al calcular el fractil.
fraction	Un número entre 0 y 1 correspondiente al percentil (cuantil expresado como fracción) que se debe calcular.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Ejemplos y resultados:

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

Ejemplos de funciones

Ejemplo	Resultado
`Fractile (Sales, 0.75)`	Para una tabla que incluya la dimensión `Customer` y la medida `Fractile([Sales])`, si se muestran **Totales**, el resultado es 71.75. Esta es la razón en la distribución de valores de `Sales` de que el 75% de los valores se encuentren por debajo.
`Fractile (TOTAL Sales, 0.75))`	71.75 para todos los valores de `Customer`, porque el cualificador TOTAL implica que las dimensiones se ignoran.
`Fractile (DISTINCT Sales, 0.75)`	70 para el total, porque usar el cualificador DISTINCT implica que solo se evaluarán los valores únicos de `Sales` por cada `Customer`.

Datos utilizados en los ejemplos:

```
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 '|');
```

Vea también:

Aggr - función de gráfico (page 559)

FractileExc

FractileExc() halla el valor que corresponde al fractil (cuantil) exclusivo de los datos agregados de la expresión que se repite por un número de registros definido por una cláusula **group by**.

Puede usar Fractile (page 417) para calcular el fractil completo.

Sintaxis:

```
FractileExc(expr, fraction)
```

Tipo de datos que devuelve: numérico

La función devuelve el valor correspondiente al rango definido por `rank = fraction * (N+1)` donde `N` es el número de valores en `expr`. Si `rank` es un número no entero, se realiza una interpolación entre los dos valores más cercanos.

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o campo que contiene los datos que se deben utilizar al calcular el fractil.
fraction	Un número entre 0 y 1 correspondiente al percentil (cuantil expresado como fracción) que se debe calcular.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

Datos resultantes

Ejemplo	Resultado
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;	En una tabla con las dimensiones Type y MyFractile, los resultados de los cálculos de FractileExc() en el script de carga de datos son: Type MyFractile Comparison 28.5 Observation 38

FractileExc - función de gráfico

FractileExc() halla el valor que corresponde al fractil exclusivo (cuantil) de los datos agregados en el rango proporcionado por la expresión que se repite por las dimensiones del gráfico.

Puede usar Fractile - función de gráfico (page 418) para calcular el fractil completo.

Sintaxis:

```
FractileExc([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr,
fraction)
```

Tipo de datos que devuelve: numérico

La función devuelve el valor correspondiente al rango definido por `rank = fraction * (N+1)` donde `N` es el número de valores en `expr`. Si `rank` es un número no entero, se realiza una interpolación entre los dos valores más cercanos.

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o campo que contiene los datos que se deben utilizar al calcular el fractil.
fraction	Un número entre 0 y 1 correspondiente al percentil (cuantil expresado como fracción) que se debe calcular.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Ejemplos y resultados:

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

Ejemplos de funciones

Ejemplo	Resultado
`FractileExc (Sales, 0.75)`	Para una tabla que incluya la dimensión `Customer` y la medida `FractileExc([Sales])`, si se muestran **Totales**, el resultado es 75.25. Esta es la razón en la distribución de valores de `Sales` de que el 75% de los valores se encuentren por debajo.
`FractileExc (TOTAL Sales, 0.75))`	75,25 para todos los valores de `Customer`, porque el cualificador TOTAL implica que las dimensiones se ignoran.
`FractileExc (DISTINCT Sales, 0.75)`	73,50 para el total, porque usar el cualificador DISTINCT implica que solo se evaluarán los valores únicos de `Sales` por cada `Customer`.

Datos utilizados en los ejemplos:

```
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 '|');
```

Vea también:

Aggr - función de gráfico (page 559)

Kurtosis

Kurtosis() devuelve la kurtosis de los datos de la expresión iterada sobre una serie de registros según lo definido por una cláusula **group by**.

Sintaxis:

```
Kurtosis([distinct ] expr )
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
distinct	Si la palabra **distinct** figura antes de la expresión, todos los duplicados se descartan.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

Datos resultantes

Ejemplo	Resultado
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;	En una tabla con las dimensiones Type, MyKurtosis1 y MyKurtosis2 los resultados de los cálculos de Kurtosis() en el script de carga de datos son: Type MyKurtosis1 MyKurtosis2 Comparison -1.1612957 -1.4982366 Observation -1.1148768 -0.93540144

Kurtosis - función de gráfico

Kurtosis() halla la kurtosis del rango de datos agregados en la expresión o campo que se repite por las dimensiones del gráfico.

Sintaxis:

```
Kurtosis([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.

Argumento	Descripción
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Ejemplos y resultados:

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

Ejemplos de funciones

Ejemplo	Resultado
`Kurtosis (Value)`	Para una tabla que incluya la dimensión `Type` y la medida `Kurtosis(Value)`, si se muestran **Totales** en la tabla, y el formato numérico se fija en 3 cifras significativas, el resultado es 1.252. Para `Comparison` es 1.161 y para `Observation` es 1.115.
`Kurtosis (TOTAL Value))`	1.252 para todos los valores de `Type`, porque el cualificador TOTAL implica que se descartan las dimensiones.

Datos utilizados en los ejemplos:

```
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 '|');
```

Vea también:

Avg - función de gráfico (page 411)

LINEST_B

LINEST_B() devuelve el valor b agregado (intercepción en y) de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por números pareados en x-expression y y-expression que se repiten por una serie de registros según lo definido por una cláusula **group by**.

Sintaxis:

```
LINEST_B (y_value, x_value[, y0 [, x0 ]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.

Argumento	Descripción
y(0), x(0)	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.

Limitaciones:

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)

LINEST_B - función de gráfico

LINEST_B() devuelve el valor b agregado (intercepción en y) de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por números pareados en las expresiones dadas por las expresiones **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

Sintaxis:

```
LINEST_B([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [ , x0_const]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y0_const, x0_const	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. *A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.*

Argumento	Descripción
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)
Avg - función de gráfico (page 411)

LINEST_DF

LINEST_DF() devuelve los grados de libertad agregados de una regresión lineal definida por la ecuación y=mx+b para una serie de coordenadas representadas por números pareados en x-expression y y-expression que se repiten por un número de registros definido por una cláusula **group by**.

Sintaxis:

```
LINEST_DF (valor_y, valor_x[, y0 [, x0 ]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y(0), x(0)	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.

Limitaciones:

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)

LINEST_DF - función de gráfico

LINEST_DF() devuelve los grados agregados de libertad de una regresión lineal definida por la ecuación y=mx+b para una serie de coordenadas representadas por números pareados en las expresiones dadas por **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

Sintaxis:

```
LINEST_DF([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value [, y0_const [, x0_const]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.

Argumento	Descripción
y0, x0	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. *A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.*
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

- *Ejemplos de cómo usar funciones linest (page 470)*
- *Avg - función de gráfico (page 411)*

LINEST_F

Esta función de script devuelve el estadístico F agregado ($r^2/(1-r^2)$) de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por números pareados en x-expression y y-expression que se repiten a lo largo de una serie de registros, según lo definido por una cláusula **group by**.

Sintaxis:

```
LINEST_F (valor_y, valor_x[, y0 [, x0 ]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y(0), x(0)	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.

Limitaciones:

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)

LINEST_F - función de gráfico

LINEST_F() devuelve el estadístico F agregado (r2/(1-r2)) de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por números pareados en las expresiones dadas por **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

Sintaxis:

```
LINEST_F([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.

Argumento	Descripción
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y0, x0	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. *A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.*
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)
Avg - función de gráfico (page 411)

LINEST_M

LINEST_M() devuelve el valor m agregado (pendiente) de una regresión lineal definida por la ecuación y=mx+b para una serie de coordenadas representadas por los números pareados en x-expression y y-expression que se repiten por un número de registros según lo definido por una cláusula **group by**.

Sintaxis:

```
LINEST_M (valor_y, valor_x[, y0 [, x0 ]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y(0), x(0)	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.

Limitaciones:

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)

LINEST_M - función de gráfico

LINEST_M() devuelve el valor agregado m (pendiente) de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por números pareados dados por las expresiones **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

Sintaxis:

```
LINEST_M([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.

Argumento	Descripción
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y0, x0	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. *A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.*
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)
Avg - función de gráfico (page 411)

LINEST_R2

LINEST_R2() devuelve el valor agregado r^2 (coeficiente de determinación) de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por números pareados en x-expression y y-expression que se repiten en una serie de registros según lo definido por una cláusula **group by**.

Sintaxis:

```
LINEST_R2 (valor_y, valor_x[, y0 [, x0 ]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y(0), x(0)	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.

Limitaciones:

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)

LINEST_R2 - función de gráfico

LINEST_R2() devuelve el valor agregado r2 (coeficiente de determinación) de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por números pareados dados por las expresiones **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

Sintaxis:

```
LINEST_R2([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.

Argumento	Descripción
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y0, x0	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. *A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.*
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)
Avg - función de gráfico (page 411)

LINEST_SEB

LINEST_SEB() devuelve el error estándar agregado del valor b de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados en x-expression y y-expression que se repiten a lo largo de una serie de registros según lo definido por una cláusula **group by**.

Sintaxis:

```
LINEST_SEB (valor_y, valor_x[, y0 [, x0 ]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y(0), x(0)	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.

Limitaciones:

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)

LINEST_SEB - función de gráfico

LINEST_SEB() devuelve el error estándar agregado del valor b de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados proporcionados por las expresiones **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

Sintaxis:

```
LINEST_SEB([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.

Argumento	Descripción
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y0, x0	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. *A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.*
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)
Avg - función de gráfico (page 411)

LINEST_SEM

LINEST_SEM() devuelve el error estándar agregado del valor m de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados en x-expression y y-expression que se repiten a lo largo de una serie de registros según lo definido por una cláusula **group by**.

Sintaxis:

```
LINEST_SEM (valor_y, valor_x[, y0 [, x0 ]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y(0), x(0)	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.

Limitaciones:

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)

LINEST_SEM - función de gráfico

LINEST_SEM() devuelve el error estándar agregado del valor m de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados proporcionados por las expresiones **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

Sintaxis:

```
LINEST_SEM([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.

Argumento	Descripción
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y0, x0	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. *A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.*
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)
Avg - función de gráfico (page 411)

LINEST_SEY

LINEST_SEY() devuelve el error estándar agregado de la estimación y de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados en x-expression y y-expression que se repiten por un número de registros según lo definido por una cláusula **group by**.

Sintaxis:

```
LINEST_SEY (valor_y, valor_x[, y0 [, x0 ]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y(0), x(0)	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.

Limitaciones:

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)

LINEST_SEY - función de gráfico

LINEST_SEY() devuelve el error estándar agregado del cálculo y de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados proporcionados por las expresiones **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

Sintaxis:

```
LINEST_SEY([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.

Argumento	Descripción
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y0, x0	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. *A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.*
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)
Avg - función de gráfico (page 411)

LINEST_SSREG

LINEST_SSREG() devuelve la suma de cuadrados de regresión agregada de una regresión lineal definida por la ecuación y=mx+b para una serie de coordenadas representadas por los números pareados en x-expression y y-expression que se repiten sobre un número de registros conforme a lo definido por una cláusula **group by**.

Sintaxis:

```
LINEST_SSREG (valor_y, valor_x[, y0 [, x0 ]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y(0), x(0)	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.

Limitaciones:

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)

LINEST_SSREG - función de gráfico

LINEST_SSREG() devuelve la suma de cuadrados de regresión agregada de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados proporcionados por las expresiones **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

Sintaxis:

```
LINEST_SSREG([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.

Argumento	Descripción
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y0, x0	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. *A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.*
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)
Avg - función de gráfico (page 411)

LINEST_SSRESID

LINEST_SSRESID() devuelve la suma residual agregada de cuadrados de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados en x-expression y y-expression que se repiten por un conjunto de registros según lo definido por una cláusula **group by**.

Sintaxis:

```
LINEST_SSRESID (valor_y, valor_x[, y0 [, x0 ]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y(0), x(0)	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.

Limitaciones:

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)

LINEST_SSRESID - función de gráfico

LINEST_SSRESID() devuelve la suma residual agregada de cuadrados de una regresión lineal definida por la ecuación y=mx+b de una serie de coordenadas representadas por los números pareados de las expresiones dados por **x_value** y **y_value**, que se repiten por las dimensiones del gráfico.

Sintaxis:

```
LINEST_SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value,
x_value[, y0_const[, x0_const]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.

Argumento	Descripción
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.
y0, x0	Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija. *A menos que se definan ambos, y0 y x0, la función requiere al menos dos pares de datos válidos para calcular. Si se definen y0 y x0, un único par de datos servirá.*
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Se puede establecer un valor y0 opcional que obligue a la línea de regresión a pasar a través del eje y en un punto determinado. Indicando ambos y0 y x0 es posible forzar la línea de regresión para que pase a través de una sola coordenada fija.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Vea también:

Ejemplos de cómo usar funciones linest (page 470)
Avg - función de gráfico (page 411)

Median

Median() devuelve la mediana agregada de los valores de la expresión que se repiten sobre un número de registros conforme a lo definido por una cláusula **group by**.

Sintaxis:

```
Median (expr)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.

Ejemplo: Expresión de gráfico con mediana

Ejemplo: expresión de script

Script de carga

Cargue los siguientes datos inline y la expresión de script en el editor de carga de datos para este ejemplo.

```
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;
```

Crear una visualización

Cree una visualización de tabla en una hoja de Qlik Sense con **Letter** y **MyMedian** como dimensiones.

Resultado

Letter	MyMedian
A	3.5
B	8

Explicación

La mediana se considera el número "hacia la mitad" cuando los números se han ordenado de menor a mayor. Si el conjunto de datos tiene un número par de valores, la función devuelve el promedio de los dos valores intermedios. En este ejemplo, la mediana se calcula para cada conjunto de valores de **A** y **B**, que es 3,5 y 8,

respectivamente.

Median - función de gráfico

Median() devuelve el valor de la mediana del rango de valores agregados en la expresión que se repite por las dimensiones del gráfico.

Sintaxis:

```
Median([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Ejemplo: Expresión de gráfico con mediana

Ejemplo: expresión de gráfico

Script de carga

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear el ejemplo de expresión de gráfico a continuación.

```
Load RecNo() as RowNo, Letter, Number Inline
[Letter, Number
A,1
```

```
A,3
A,4
A,9
B,2
B,8
B,9];
```

Crear una visualización

Cree una visualización de tabla en una hoja de Qlik Sense con **Letter** como dimensión.

Expresión de gráfico

Añada la siguiente expresión a la tabla como medida:

```
Median(Number)
```

Resultado

Letter	Median(Number)
Totals	4
A	3.5
B	8

Explicación

La mediana se considera el número "hacia la mitad" cuando los números se han ordenado de menor a mayor. Si el conjunto de datos tiene un número par de valores, la función devuelve el promedio de los dos valores intermedios. En este ejemplo, la mediana se calcula para cada conjunto de valores de **A** y **B**, que es 3,5 y 8, respectivamente.

La mediana de **Totales** se calcula a partir de todos los valores, lo que equivale a 4.

Vea también:

Avg - función de gráfico (page 411)

MutualInfo - función de gráfico

MutualInfo calcula la información mutua (IM) entre dos campos o entre los valores agregados en **Aggr()**.

MutualInfo devuelve la información mutua agregada de dos conjuntos de datos. Esto permite el análisis de controladores clave entre un campo y un controlador potencial. La información mutua mide la relación entre los conjuntos de datos y se agrega para valores de pares (x, y) iterados sobre las dimensiones del gráfico. La información mutua se mide entre 0 y 1 y se puede formatear como un valor de percentil. **MutualInfo** se define mediante selecciones o mediante una expresión de conjunto.

MutualInfo permite diferentes tipos de análisis de IM:

- IM pareada: Calcula la IM entre un campo controlador y un campo destino.
- Desglose del controlador por valor: La información mutua IM se calcula entre los valores individuales de los campos en el controlador y los campos de destino.
- Selección de características: Use **MutualInfo** en un gráfico de cuadrícula para crear una matriz en la que todos los campos se comparen entre sí en función de la IM.

MutualInfo no necesariamente indica causalidad entre campos que comparten información mutua. Dos campos pueden compartir información mutua, pero pueden no ser iguales entre sí. Por ejemplo, al comparar las ventas de helados y la temperatura exterior, **MutualInfo** mostrará información mutua entre los dos. No indicará si es la temperatura exterior lo que impulsa las ventas de helados, lo cual es probable, o si es la venta de helados lo que impulsa la temperatura exterior, lo cual es poco probable.

Al calcular la información mutua, las asociaciones afectan a la correspondencia y la frecuencia de los valores de los campos que son de diferentes tablas.

Los valores devueltos para los mismos campos o selecciones pueden variar ligeramente. Esto se debe a que cada llamada a **MutualInfo** opera en una muestra seleccionada al azar y a la aleatoriedad inherente del algoritmo **MutualInfo**.

MutualInfo se puede aplicar a la función **Aggr()**.

Sintaxis:

```
MutualInfo({SetExpression}] [DISTINCT] [TOTAL] field1, field2 , datatype [,
breakdownbyvalue [, samplesize ]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
field1, field2	Las expresiones o campos que contienen los dos conjuntos de muestra para los que se ha de medir el coeficiente de información mutua.
datatype	Los tipos de datos que contienen el campo destino y el campo controlador, 1 o 'dd' para discreto:discreto 2 o 'cc' para continuo:continuo 3 o 'cd' para continuo:discreto 4 o 'dc' para discreto:continuo Los tipos de datos distinguen entre mayúsculas y minúsculas.

Argumento	Descripción
breakdownbyvalue	Un valor estático que corresponde a un valor en el controlador. Si se proporciona, el cálculo calculará la contribución de IM para ese valor. Puede usar **ValueList()** o **ValueLoop()**. Si se añade **Null()**, el cálculo computará el IM general para todos los valores del controlador. El desglose por valor requiere que el controlador contenga datos discretos.
samplesize	El número de valores para extraer muestras desde el destino y el controlador. Las muestras son aleatorias. **MutualInfo** requiere un tamaño de muestra mínimo de 80. De forma predeterminada, **MutualInfo** muestrea hasta 10.000 pares de datos ya que **MutualInfo** puede consumir demasiados recursos. Puede especificar un número mayor de pares de datos en el tamaño de la muestra. Si se agota el tiempo de espera de **MutualInfo**, reduzca el tamaño de la muestra.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

Ejemplos de funciones

Ejemplo	Resultado
`mutualinfo (Age, Salary, 1)`	Para una tabla que incluya la dimensión `Employee name` y la medida `mutualinfo(Age, Salary, 1)`, el resultado es 0.99820986. El resultado solo se muestra para la celda de totales.

Ejemplo	Resultado
`mutualinfo (TOTAL Age, Salary, 1, null(), 81)`	Si crea un panel de filtrado con la dimensión Gender y realiza selecciones desde el mismo, verá el resultado 0.99805677 cuando Female esté seleccionado y 0.99847373 si Male está seleccionado. Esto se debe a que la selección excluye todos los resultados que no pertenecen al otro valor de Gender.
`mutualinfo (TOTAL Age, Gender, 1, ValueLoop (25,35))`	0.68196996. Seleccionar cualquier valor de Gender cambiará esto a 0.
`mutualinfo ({1} TOTAL Age, Salary, 1, null())`	0.99820986. Esto es independiente de las selecciones. La expresión de conjunto {1} ignora todas las selecciones y dimensiones.

Datos utilizados en los ejemplos:

```
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() devuelve la asimetría de la expresión en una serie de registros según lo definido por una cláusula **group by**.

Sintaxis:

```
Skew([ distinct] expr)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
DISTINCT	Si la palabra **distinct** figura antes de la expresión, todos los duplicados se descartan.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Después construya una tabla simple con `Type` y *MySkew* como dimensiones.

Datos resultantes

Ejemplo	Resultado

`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;`
	Los resultados del cálculo Skew() son:
	• `Type` es `MySkew`
	• `Comparison` es 0.86414768
	• `Observation` es 0.32625351

Skew - función de gráfico

Skew() devuelve la asimetría agregada de la expresión o campo que se repite por las dimensiones del gráfico.

Sintaxis:

```
Skew([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Después cree una tabla simple con `Type` como dimensión y `Skew(Value)` como medida.

`Totals` debe estar habilitado en las propiedades de la tabla.

Ejemplo	Resultado
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 '|');	Los resultados del cálculo Skew(Value) son: • Total es 0.23522195 • Comparison es 0.86414768 • Observation es 0.32625351

Vea también:

Avg - función de gráfico (page 411)

Stdev

Stdev() devuelve la desviación estándar de los valores dados por la expresión en una serie de registros según lo definido por una cláusula **group by**.

Sintaxis:

```
Stdev([distinct] expr)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
distinct	Si la palabra **distinct** figura antes de la expresión, todos los duplicados se descartan.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Después construya una tabla simple con `Type` y `MyStdev` como dimensiones.

Datos resultantes

Ejemplo	Resultado

`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;`
	Los resultados del cálculo Stdev() son:
	• `Type` es `MyStdev`
	• `Comparison` es 14.61245
	• `Observation` es 12.507997

Stdev - función de gráfico

Stdev() halla la desviación estándar del rango de datos agregados en la expresión o campo que se repite por las dimensiones del gráfico.

Sintaxis:

```
Stdev([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Después cree una tabla simple con `Type` como dimensión y `Stdev(Value)` como medida.

`Totals` debe estar habilitado en las propiedades de la tabla.

Ejemplo	Resultado

`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 '|');`
	Los resultados del cálculo Stdev(Value) son:
	• `Total` es 15.47529
	• `Comparison` es 14.61245
	• `Observation` es 12.507997

Vea también:

- *Avg - función de gráfico (page 411)*
- *STEYX - función de gráfico (page 468)*

Sterr

Sterr() devuelve el error estándar agregado (stdev/sqrt(n)) de una serie de valores representados por la expresión iterada sobre un número de registros, según lo definido por una cláusula **group by**.

Sintaxis:

```
Sterr ([distinct] expr)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
distinct	Si la palabra **distinct** figura antes de la expresión, todos los duplicados se descartan.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos se descartan.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

Datos resultantes

Ejemplo	Resultado

`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;`
	En una tabla con las dimensiones `Type` y `MySterr`, los resultados del cálculo Sterr() en el script de carga de datos son:
	
	`Type MySterr`
	
	`Comparison 3.2674431`
	
	`Observation 2.7968733`

Sterr - función de gráfico

Sterr() halla el valor del error estándar de la media, (stdev/sqrt(n)), de la serie de valores agregados en la expresión que se repite por las dimensiones del gráfico.

Sintaxis:

```
Sterr([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Los valores de texto, valores NULL y valores perdidos se descartan.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Después cree una tabla simple con `Type` como dimensión y `Sterr(Value)` como medida.

`Totals` debe estar habilitado en las propiedades de la tabla.

Ejemplo	Resultado
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 '|');	Los resultados del cálculo Sterr(Value) son: • Total es 2.4468583 • Comparison es 3.2674431 • Observation es 2.7968733

Vea también:

- *Avg - función de gráfico (page 411)*
- *STEYX - función de gráfico (page 468)*

STEYX

STEYX() devuelve el error estándar agregado del valor y pronosticado para cada valor de x en la regresión de una serie de coordenadas representadas por los números pareados en x-expression y y-expression que se repiten a lo largo de un número de registros, según lo definido por una cláusula **group by**.

Sintaxis:

```
STEYX (y_value, x_value)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores y que se han de medir.
x_value	La expresión o campo que contiene el rango de valores x que se han de medir.

Limitaciones:

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

Datos resultantes

Ejemplo	Resultado
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;	En una tabla con la dimensión MySTEYX, el resultado del cálculo STEYX() en el script de carga de datos es 2.0714764.

STEYX - función de gráfico

STEYX() devuelve el error estándar agregado al predecir valores y para cada valor x de una regresión lineal dada por una serie de coordenadas representadas por números pareados en las expresiones dadas por **y_value** y **x_value**.

Sintaxis:

```
STEYX([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
y_value	La expresión o campo que contiene el rango de valores conocidos y que se medirán.
x_value	La expresión o campo que contiene el rango de valores conocidos x que se medirán.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

El parámetro de la función de agregación no debe contener otras funciones de agregación, a menos que dichas agregaciones internas contengan el cualificador **TOTAL**. Para agregaciones anidadas más avanzadas, utilice la función avanzada **Aggr**, en combinación con una dimensión especificada.

Los valores de texto, valores NULL y los valores perdidos en cualquiera o ambas partes de un par de datos dan como resultado que el par de datos completo no se tenga en cuenta.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Después construya una tabla simple con `KnownY` y `KnownX` como dimensión y `Steyx(KnownY,KnownX)` como medida.

`Totals` debe estar habilitado en las propiedades de la tabla.

Ejemplo	Resultado
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 '|');	El resultado del cálculo STEYX(KnownY,KnownX) es 2,071 (si el formato numérico está fijado en 3 decimales).

Vea también:

- *Avg - función de gráfico (page 411)*
- *Sterr - función de gráfico (page 464)*

Ejemplos de cómo usar funciones linest

Las funciones linest se utilizan para encontrar valores asociados con el análisis de regresión lineal. Esta sección describe cómo construir visualizaciones utilizando datos de muestra para hallar los valores de las funciones linest disponibles en Qlik Sense. Las funciones linest pueden utilizarse en el script de carga de datos y en expresiones de gráficos.

Consulte los temas de la función de gráfico y de script linest si desea descripciones de sintaxis y argumentos.

Expresiones de datos y script utilizadas en los ejemplos

Cargue los siguientes datos inline y expresiones de script en el editor de carga de datos para los ejemplos de linest() a continuación.

```
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;
```

Ejemplo 1: Expresiones de script que usan linest

Ejemplo: Expresiones de script

Cree una visualización a partir de los cálculos del script de carga de datos.

Cree una visualización de tabla en una hoja de Qlik Sense con los siguientes campos como columnas:

- Linest_B
- Linest_DF
- Linest_F

- Linest_M
- Linest_R2
- Linest_SEB
- Linest_SEM
- Linest_SEY
- Linest_SSREG
- Linest_SSRESID

Resultado

La tabla que contiene los resultados de los cálculos linest realizados en el script de carga de datos presentarán el siguiente aspecto:

Tabla de resultados

Linest_B	Linest_DF	Linest_F	Linest_M	Linest_R2	Linest_SEB
-35.047	18	20.788	8.605	0.536	22.607

Tabla de resultados

Linest_SEM	Linest_SEY	Linest_SSREG	Linest_SSRESID
1.887	48.666	49235.014	42631.186

Ejemplo 2: Expresiones de gráfico que usan linest

Ejemplo: Expresiones de gráfico

Cree una visualización de tabla en una hoja de Qlik Sense con los siguientes campos como dimensiones:

```
ValueList('Linest_b', 'Linest_df','Linest_f', 'Linest_m','Linest_r2','Linest_SEB','Linest_
SEM','Linest_SEY','Linest_SSREG','Linest_SSRESID')
```

Esta expresión utiliza la función de dimensiones sintéticas para crear etiquetas para las dimensiones con los nombres de las funciones linest. Puede cambiar la etiqueta a **Funciones Linest** para ahorrar espacio.

Añada la siguiente expresión a la tabla como medida.

```
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) )
```

Esta expresión muestra el valor del resultado de cada función linest comparado con el correspondiente nombre en la dimensión sintética. El resultado de `Linest_b(Y,X)` se muestra junto a **linest_b**, etc.

Resultado

Tabla de resultados

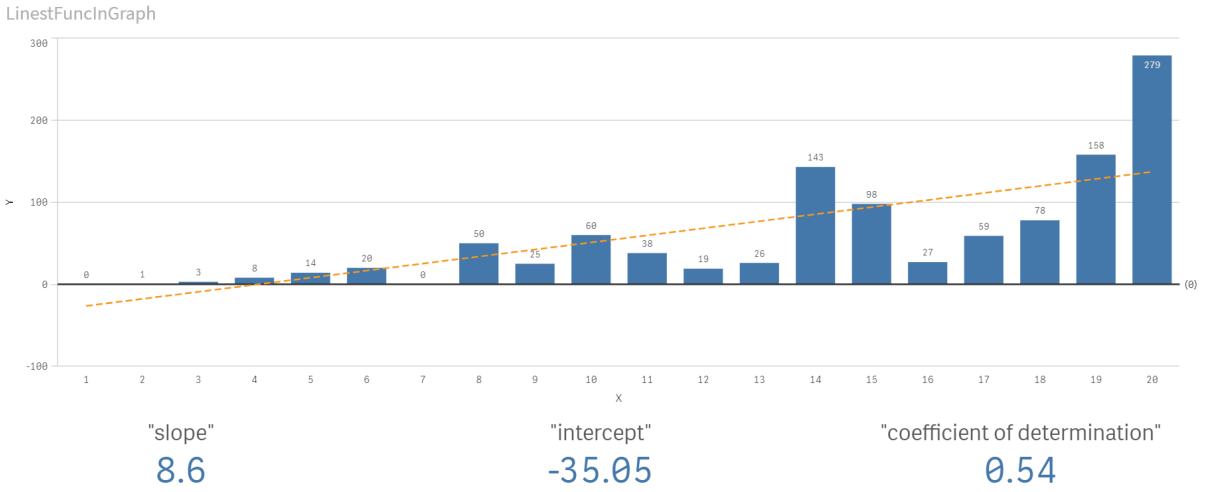
Linest functions	Linest function results
Linest_b	-35.047
Linest_df	18
Linest_f	20.788
Linest_m	8.605
Linest_r2	0.536
Linest_SEB	22.607
Linest_SEM	1.887
Linest_SEY	48.666
Linest_SSREG	49235.014
Linest_SSRESID	42631.186

Ejemplo 3: Expresiones de gráfico que usan linest

Ejemplo: Expresiones de gráfico

1. Cree una visualización de gráfico de barras en una hoja de Qlik Sense con **X** como dimensión e **Y** como medida.
2. Agregue una línea de tendencia lineal a la medida Y.
3. Agregue una visualización de KPI a la hoja.
    1. Agregue *pendiente* como etiqueta para el KPI.
    2. Agregue `sum(Linest_M)` como expresión para el KPI.
4. Agregue una segunda visualización de KPI a la hoja.
    1. Agregue *intersección* como etiqueta para el KPI.
    2. Agregue `Sum(Linest_B)` como expresión para el KPI.
5. Agregue una tercera visualización de KPI a la hoja.
    1. Agregue *coeficiente de determinación* como etiqueta para el KPI.
    2. Agregue `Sum(Linest_R2)` como expresión para el KPI.

Resultado



Explicación

El gráfico de barras muestra el trazado de los datos X e Y. Las funciones linest () relevantes proporcionan valores para la ecuación de regresión lineal en la que se basa la línea de tendencia, a saber, y = m * x + b. La ecuación utiliza el método de "mínimos cuadrados" para calcular una línea recta (línea de tendencia) al devolver una matriz que describe la línea que mejor se ajusta a los datos.

Los KPI muestran los resultados de las funciones linest() **sum(Linest_M)** para la pendiente y **sum(Linest_B)** para la intersección en Y, que son variables en la ecuación de regresión lineal, y el valor R2 agregado correspondiente para el coeficiente de determinación.

Funciones estadísticas de prueba

Las funciones de prueba estadística se pueden utilizar tanto en el script de carga de datos como en expresiones de gráficos, pero la sintaxis difiere.

Funciones de prueba Chi-2

Generalmente se utiliza en el estudio de variables cualitativas. Uno puede comparar las frecuencias observadas en una tabla de frecuencia de una dirección con frecuencias esperadas, o estudiar la conexión entre dos variables en una tabla de contingencias.

Funciones de prueba T

Las funciones de prueba t se utilizan para el examen estadístico de dos promedios de población. Una prueba t de dos muestras analiza si dos muestras son distintas y es muy habitual utilizarla cuando dos distribuciones normales tienen variaciones desconocidas y cuando un experimento utiliza un tamaño de muestra pequeño.

Funciones de prueba Z

Un análisis estadístico de dos promedios de población. Una prueba z de dos muestras analiza si dos muestras son distintas y es muy habitual cuando dos distribuciones normales tienen variaciones conocidas y cuando un experimento usa un gran tamaño de muestra.

Funciones de prueba Chi2

Generalmente se utiliza en el estudio de variables cualitativas. Uno puede comparar las frecuencias observadas en una tabla de frecuencia de una dirección con frecuencias esperadas, o estudiar la conexión entre dos variables en una tabla de contingencias.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.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Chi2Test_chi2

Chi2Test_chi2() devuelve el valor agregado de la prueba chi^2 de una o dos series de valores.

```
Chi2Test_chi2() devuelve el valor agregado de la prueba chi2 de una o dos series de valores.(col, row, actual_value[, expected_value])
```

Chi2Test_df

Chi2Test_df() devuelve el valor agregado df (grados de libertad) de la prueba chi ^{2} de una o dos series de valores.

```
Chi2Test_df() devuelve el valor agregado df (grados de libertad) de la prueba chi 2 de una o dos series de valores.(col, row, actual_value[, expected_value])
```

Chi2Test_p

Chi2Test_p() devuelve el valor p agregado de la prueba chi^2 (significación) de una o dos series de valores.

```
Chi2Test_p - función de gráfico(col, row, actual_value[, expected_value])
```

Vea también:

- *Funciones de prueba T (page 478)*
- *Funciones de prueba Z (page 515)*

Chi2Test_chi2

Chi2Test_chi2() devuelve el valor agregado de la prueba chi^2 de una o dos series de valores.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Todos Qlik Sense chi² Las funciones de prueba tienen los mismos argumentos.

Sintaxis:

```
Chi2Test_chi2(col, row, actual_value[, expected_value])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
col, row	La columna y fila especificadas en la matriz de valores que se están probando.
actual_value	El valor de datos observado en la columna **col** y fila **row** especificadas.
expected_value	El valor esperado para la distribución en la columna **col** y fila **row** especificada.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
Chi2Test_chi2( Grp, Grade, Count )
Chi2Test_chi2( Gender, Description, Observed, Expected )
```

Vea también:

Ejemplos de cómo usar funciones chi2-test en gráficos (page 531)
Ejemplos de cómo usar funciones chi2-test en el script de carga de datos (page 535)

Chi2Test_df

Chi2Test_df() devuelve el valor agregado df (grados de libertad) de la prueba chi ^{2} de una o dos series de valores.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Todos Qlik Sense chi² Las funciones de prueba tienen los mismos argumentos.

Sintaxis:

```
Chi2Test_df(col, row, actual_value[, expected_value])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
col, row	La columna y fila especificadas en la matriz de valores que se están probando.
actual_value	El valor de datos observado en la columna **col** y fila **row** especificadas.
expected_value	El valor esperado para la distribución en la columna **col** y fila **row** especificada.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
Chi2Test_df( Grp, Grade, Count )
Chi2Test_df( Gender, Description, Observed, Expected )
```

Vea también:

Ejemplos de cómo usar funciones chi2-test en gráficos (page 531)
Ejemplos de cómo usar funciones chi2-test en el script de carga de datos (page 535)

Chi2Test_p - función de gráfico

Chi2Test_p() devuelve el valor p agregado de la prueba chi^2 (significación) de una o dos series de valores. La prueba se puede realizar, bien en los valores de **actual_value**, comprobando las variaciones dentro de la matriz **col** y **row** especificada, o bien comparando los valores de **actual_value** con los correspondientes valores en **expected_value**, si se especifica.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Todos Qlik Sense chi^2 Las funciones de prueba tienen los mismos argumentos.

Sintaxis:

```
Chi2Test_p(col, row, actual_value[, expected_value])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
col, row	La columna y fila especificadas en la matriz de valores que se están probando.
actual_value	El valor de datos observado en la columna **col** y fila **row** especificadas.
expected_value	El valor esperado para la distribución en la columna **col** y fila **row** especificada.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
Chi2Test_p( Grp, Grade, Count )
Chi2Test_p( Gender, Description, Observed, Expected )
```

Vea también:

Ejemplos de cómo usar funciones chi2-test en gráficos (page 531)
Ejemplos de cómo usar funciones chi2-test en el script de carga de datos (page 535)

Funciones de prueba T

Las funciones de prueba t se utilizan para el examen estadístico de dos promedios de población. Una prueba t de dos muestras analiza si dos muestras son distintas y es muy habitual utilizarla cuando dos distribuciones normales tienen variaciones desconocidas y cuando un experimento utiliza un tamaño de muestra pequeño.

En los siguientes apartados, las funciones estadísticas de prueba t se agrupan conforme a la prueba de muestra de estudiante que se aplica a cada tipo de función.

Crear un informe t-test típico (page 537)

Dos pruebas t de muestras independientes
Las funciones descritas a continuación se aplican a dos pruebas t de estudiante de dos muestras independientes.

ttest_conf
TTest_conf devuelve el valor agregado del intervalo de confianza de la prueba t de dos muestras independientes.

TTest_conf devuelve el valor agregado del intervalo de confianza de la prueba t de dos muestras independientes. (grp, value [, sig[, eq_var]]**)**

ttest_df

TTest_df() devuelve el valor agregado de la prueba t de estudiante (grados de libertad) de dos series independientes de valores.

TTest_df() devuelve el valor agregado de la prueba t de estudiante (grados de libertad) de dos series independientes de valores. (grp, value [, eq_var**)**

ttest_dif

TTest_dif() es una función numérica que devuelve la diferencia de la media de la prueba t de estudiante agregada de dos series independientes de valores.

TTest_dif() es una función numérica que devuelve la diferencia de la media de la prueba t de estudiante agregada de dos series independientes de valores. (grp, value**)**

ttest_lower

TTest_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de dos series independientes de valores.

TTest_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de dos series independientes de valores. (grp, value [, sig[, eq_var]]**)**

ttest_sig

TTest_sig() devuelve el nivel de significación agregado de 2 colas de la prueba t de estudiante de dos series independientes de valores.

TTest_sig() devuelve el nivel de significación agregado de 2 colas de la prueba t de estudiante de dos series independientes de valores. (grp, value [, eq_var]**)**

ttest_sterr

TTest_sterr() devuelve el error estándar agregado de la prueba t de estudiante de la diferencia de media de dos series independientes de valores.

TTest_sterr() devuelve el error estándar agregado de la prueba t de estudiante de la diferencia de media de dos series independientes de valores. (grp, value [, eq_var]**)**

ttest_t

TTest_t() devuelve el valor t agregado de dos series independientes de valores.

TTest_t() devuelve el valor t agregado de dos series independientes de valores. (grp, value [, eq_var]**)**

ttest_upper
TTest_upper() devuelve el valor agregado del límite superior del intervalo de confianza de dos series independientes de valores.

```
TTest_upper() devuelve el valor agregado del límite superior del intervalo de confianza de dos series independientes de valores. (grp, value [, sig [, eq_var]])
```

Dos pruebas t ponderadas de muestras independientes
Las funciones siguientes se aplican a dos pruebas t de estudiante de dos muestras independientes, en las que la serie de los datos de entrada se suministra en un formato de dos columnas ponderadas.

ttestw_conf
TTestw_conf() devuelve el valor t agregado de dos series independientes de valores.

```
TTestw_conf() devuelve el valor t agregado de dos series independientes de valores. (weight, grp, value [, sig[, eq_var]])
```

ttestw_df
TTestw_df() devuelve el valor df (grados de libertad) agregado de la prueba t de estudiante de dos series independientes de valores.

```
TTestw_df() devuelve el valor df (grados de libertad) agregado de la prueba t de estudiante de dos series independientes de valores. (weight, grp, value [, eq_var])
```

ttestw_dif
TTestw_dif() devuelve la diferencia de medias agregada de la prueba t de estudiante de dos series independientes de valores.

```
TTestw_dif() devuelve la diferencia de medias agregada de la prueba t de estudiante de dos series independientes de valores. ( weight, grp, value)
```

ttestw_lower
TTestw_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de dos series independientes de valores.

```
TTestw_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de dos series independientes de valores. (weight, grp, value [, sig[, eq_var]])
```

ttestw_sig
TTestw_sig() devuelve el nivel de significación agregado de 2 colas de la prueba t de estudiante de dos series independientes de valores.

```
TTestw_sig() devuelve el nivel de significación agregado de 2 colas de la prueba t de estudiante de dos series independientes de valores. ( weight, grp, value [, eq_var])
```

ttestw_sterr

TTestw_sterr() devuelve el error estándar agregado de la prueba t de estudiante de la diferencia de media de dos series independientes de valores.

```
TTestw_sterr() devuelve el error estándar agregado de la prueba t de estudiante de la diferencia de media de dos series independientes de valores. (weight, grp, value [, eq_var])
```

ttestw_t

TTestw_t() devuelve el valor t agregado de dos series independientes de valores.

```
TTestw_t() devuelve el valor t agregado de dos series independientes de valores. (weight, grp, value [, eq_var])
```

ttestw_upper

TTestw_upper() devuelve el valor agregado del límite superior del intervalo de confianza de dos series independientes de valores.

```
TTestw_upper() devuelve el valor agregado del límite superior del intervalo de confianza de dos series independientes de valores. (weight, grp, value [, sig [, eq_var]])
```

Pruebas t de una muestra

Las funciones siguientes se aplican a pruebas t de estudiante de una muestra.

ttest1_conf

TTest1_conf() devuelve el valor de intervalo de confianza agregado de una serie de valores.

```
TTest1_conf() devuelve el valor de intervalo de confianza agregado de una serie de valores. (value [, sig])
```

ttest1_df

TTest1_df() devuelve el valor df (grados de libertad) agregado de la prueba t de estudiante de una serie de valores.

```
TTest1_df() devuelve el valor df (grados de libertad) agregado de la prueba t de estudiante de una serie de valores. (value)
```

ttest1_dif

TTest1_dif() devuelve la diferencia de medias agregada de la prueba t de estudiante de una serie de valores.

```
TTest1_dif() devuelve la diferencia de medias agregada de la prueba t de estudiante de una serie de valores. (value)
```

ttest1_lower

TTest1_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de una serie de valores.

```
TTest1_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de una serie de valores. (value [, sig])
```

ttest1_sig

TTest1_sig() devuelve el nivel de significación agregado de 2 colas de la prueba t de estudiante de una serie de valores.

```
TTest1_sig() devuelve el nivel de significación agregado de 2 colas de la prueba t de estudiante de una serie de valores. (value)
```

ttest1_sterr

TTest1_sterr() devuelve el error estándar agregado de la prueba t de estudiante de la diferencia de media de una serie de valores.

```
TTest1_sterr() devuelve el error estándar agregado de la prueba t de estudiante de la diferencia de media de una serie de valores. (value)
```

ttest1_t

TTest1_t() devuelve el valor agregado t de una serie de valores.

```
TTest1_t() devuelve el valor agregado t de una serie de valores. (value)
```

ttest1_upper

TTest1_upper() devuelve el valor agregado del límite superior del intervalo de confianza de una serie de valores.

```
TTest1_upper() devuelve el valor agregado del límite superior del intervalo de confianza de una serie de valores. (value [, sig])
```

Pruebas t ponderadas de una muestra

Las funciones siguientes se aplican a pruebas t de estudiante de una sola muestra en las que los datos de entrada se suministran en un formato de dos columnas ponderadas.

ttest1w_conf

TTest1w_conf() es una función **numérica** que devuelve el valor del intervalo de confianza agregado de una serie de valores.

```
TTest1w_conf() es una función numérica que devuelve el valor del intervalo de confianza agregado de una serie de valores. (weight, value [, sig])
```

ttest1w_df

TTest1w_df() devuelve el valor df (grados de libertad) agregado de la prueba t de estudiante de una serie de valores.

```
TTest1w_df() devuelve el valor df (grados de libertad) agregado de la prueba t de estudiante de una serie de valores. (weight, value)
```

ttest1w_dif

TTest1w_dif() devuelve la diferencia de medias agregada de la prueba t de estudiante de una serie de valores.

```
TTest1w_dif() devuelve la diferencia de medias agregada de la prueba t de estudiante de una serie de valores. (weight, value)
```

ttest1w_lower

TTest1w_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de una serie de valores.

```
TTest1w_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de una serie de valores. (weight, value [, sig])
```

ttest1w_sig

TTest1w_sig() devuelve el nivel de significación agregado de 2 colas de la prueba t de estudiante de una serie de valores.

```
TTest1w_sig() devuelve el nivel de significación agregado de 2 colas de la prueba t de estudiante de una serie de valores. (weight, value)
```

ttest1w_sterr

TTest1w_sterr() devuelve el error estándar agregado de la prueba t de estudiante de la diferencia de media de una serie de valores.

```
TTest1w_sterr() devuelve el error estándar agregado de la prueba t de estudiante de la diferencia de media de una serie de valores. (weight, value)
```

ttest1w_t

TTest1w_t() devuelve el valor agregado t de una serie de valores.

```
TTest1w_t() devuelve el valor agregado t de una serie de valores. ( weight, value)
```

ttest1w_upper

TTest1w_upper() devuelve el valor agregado del límite superior del intervalo de confianza de una serie de valores.

```
TTest1w_upper() devuelve el valor agregado del límite superior del intervalo de confianza de una serie de valores. (weight, value [, sig])
```

TTest_conf

TTest_conf devuelve el valor agregado del intervalo de confianza de la prueba t de dos muestras independientes.

Esta función se aplica a pruebas t de estudiante de muestras independientes.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest_conf ( grp, value [, sig [, eq_var]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTest_conf( Group, Value )
TTest_conf( Group, Value, Sig, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest_df

TTest_df() devuelve el valor agregado de la prueba t de estudiante (grados de libertad) de dos series independientes de valores.

Esta función se aplica a pruebas t de estudiante de muestras independientes.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest_df (grp, value [, eq_var])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTest_df( Group, Value )
TTest_df( Group, Value, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest_dif

TTest_dif() es una función numérica que devuelve la diferencia de la media de la prueba t de estudiante agregada de dos series independientes de valores.

Esta función se aplica a pruebas t de estudiante de muestras independientes.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest_dif (grp, value [, eq_var] )
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTest_dif( Group, Value )
TTest_dif( Group, Value, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest_lower

TTest_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de dos series independientes de valores.

Esta función se aplica a pruebas t de estudiante de muestras independientes.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest_lower (grp, value [, sig [, eq_var]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTest_lower( Group, Value )
TTest_lower( Group, Value, Sig, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest_sig

TTest_sig() devuelve el nivel de significación agregado de 2 colas de la prueba t de estudiante de dos series independientes de valores.

Esta función se aplica a pruebas t de estudiante de muestras independientes.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest_sig (grp, value [, eq_var])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTest_sig( Group, Value )
TTest_sig( Group, Value, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest_sterr

TTest_sterr() devuelve el error estándar agregado de la prueba t de estudiante de la diferencia de media de dos series independientes de valores.

Esta función se aplica a pruebas t de estudiante de muestras independientes.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest_sterr (grp, value [, eq_var])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTest_sterr( Group, Value )
TTest_sterr( Group, Value, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest_t

TTest_t() devuelve el valor t agregado de dos series independientes de valores.

Esta función se aplica a pruebas t de estudiante de muestras independientes.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest_t(grp, value[, eq_var])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
TTest_t( Group, Value, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest_upper

TTest_upper() devuelve el valor agregado del límite superior del intervalo de confianza de dos series independientes de valores.

Esta función se aplica a pruebas t de estudiante de muestras independientes.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest_upper (grp, value [, sig [, eq_var]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTest_upper( Group, Value )
TTest_upper( Group, Value, sig, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTestw_conf

TTestw_conf() devuelve el valor t agregado de dos series independientes de valores.

Esta función se aplica a dos pruebas t de estudiante independientes de dos muestras, en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTestw_conf (weight, grp, value [, sig [, eq_var]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTestw_conf( Weight, Group, Value )
TTestw_conf( Weight, Group, Value, sig, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTestw_df

TTestw_df() devuelve el valor df (grados de libertad) agregado de la prueba t de estudiante de dos series independientes de valores.

Esta función se aplica a dos pruebas t de estudiante independientes de dos muestras, en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTestw_df (weight, grp, value [, eq_var])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTestw_df( Weight, Group, Value )
TTestw_df( Weight, Group, Value, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTestw_dif

TTestw_dif() devuelve la diferencia de medias agregada de la prueba t de estudiante de dos series independientes de valores.

Esta función se aplica a dos pruebas t de estudiante independientes de dos muestras, en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTestw_dif (weight, grp, value)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTestw_dif( Weight, Group, Value )
TTestw_dif( Weight, Group, Value, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTestw_lower

TTestw_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de dos series independientes de valores.

Esta función se aplica a dos pruebas t de estudiante independientes de dos muestras, en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTestw_lower (weight, grp, value [, sig [, eq_var]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTestw_lower( Weight, Group, Value )
TTestw_lower( Weight, Group, Value, sig, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTestw_sig

TTestw_sig() devuelve el nivel de significación agregado de 2 colas de la prueba t de estudiante de dos series independientes de valores.

Esta función se aplica a dos pruebas t de estudiante independientes de dos muestras, en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTestw_sig ( weight, grp, value [, eq_var])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTestw_sig( Weight, Group, Value )
TTestw_sig( Weight, Group, Value, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTestw_sterr

TTestw_sterr() devuelve el error estándar agregado de la prueba t de estudiante de la diferencia de media de dos series independientes de valores.

Esta función se aplica a dos pruebas t de estudiante independientes de dos muestras, en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTestw_sterr (weight, grp, value [, eq_var])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTestw_sterr( Weight, Group, Value )
TTestw_sterr( Weight, Group, Value, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTestw_t

TTestw_t() devuelve el valor t agregado de dos series independientes de valores.

Esta función se aplica a dos pruebas t de estudiante independientes de dos muestras, en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
ttestw_t (weight, grp, value [, eq_var])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.

Argumento	Descripción
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTestw_t( Weight, Group, Value )
TTestw_t( Weight, Group, Value, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTestw_upper

TTestw_upper() devuelve el valor agregado del límite superior del intervalo de confianza de dos series independientes de valores.

Esta función se aplica a dos pruebas t de estudiante independientes de dos muestras, en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTestw_upper (weight, grp, value [, sig [, eq_var]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.

Argumento	Descripción
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTestw_upper( Weight, Group, Value )
TTestw_upper( Weight, Group, Value, sig, false )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1_conf

TTest1_conf() devuelve el valor de intervalo de confianza agregado de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1_conf (value [, sig ])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTest1_conf( Value )
TTest1_conf( Value, 0.005 )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1_df

TTest1_df() devuelve el valor df (grados de libertad) agregado de la prueba t de estudiante de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1_df (value)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
TTest1_df( Value )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1_dif

TTest1_dif() devuelve la diferencia de medias agregada de la prueba t de estudiante de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1_dif (value)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
TTest1_dif( Value )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1_lower

TTest1_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1_lower (value [, sig])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTest1_lower( value )
TTest1_lower( value, 0.005 )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1_sig

TTest1_sig() devuelve el nivel de significación agregado de 2 colas de la prueba t de estudiante de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1_sig (value)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
TTest1_sig( Value )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1_sterr

TTest1_sterr() devuelve el error estándar agregado de la prueba t de estudiante de la diferencia de media de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1_sterr (value)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
TTest1_sterr( Value )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1_t

TTest1_t() devuelve el valor agregado t de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1_t (value)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
TTest1_t( Value )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1_upper

TTest1_upper() devuelve el valor agregado del límite superior del intervalo de confianza de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1_upper (value [, sig])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTest1_upper( Value )
TTest1_upper( Value, 0.005 )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1w_conf

TTest1w_conf() es una función **numérica** que devuelve el valor del intervalo de confianza agregado de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1w_conf (weight, value [, sig ])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTest1w_conf( Weight, Value )
TTest1w_conf( Weight, Value, 0.005 )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1w_df

TTest1w_df() devuelve el valor df (grados de libertad) agregado de la prueba t de estudiante de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1w_df (weight, value)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
TTest1w_df( Weight, Value )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1w_dif

TTest1w_dif() devuelve la diferencia de medias agregada de la prueba t de estudiante de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1w_dif (weight, value)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
TTest1w_dif( Weight, Value )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1w_lower

TTest1w_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1w_lower (weight, value [, sig ])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTest1w_lower( Weight, Value )
TTest1w_lower( Weight, Value, 0.005 )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1w_sig

TTest1w_sig() devuelve el nivel de significación agregado de 2 colas de la prueba t de estudiante de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1w_sig (weight, value)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
TTest1w_sig( Weight, Value )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1w_sterr

TTest1w_sterr() devuelve el error estándar agregado de la prueba t de estudiante de la diferencia de media de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1w_sterr (weight, value)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
TTest1w_sterr( Weight, Value )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1w_t

TTest1w_t() devuelve el valor agregado t de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1w_t ( weight, value)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
TTest1w_t( Weight, Value )
```

Vea también:

Crear un informe t-test típico (page 537)

TTest1w_upper

TTest1w_upper() devuelve el valor agregado del límite superior del intervalo de confianza de una serie de valores.

Esta función se aplica a pruebas t de estudiante de una muestra en las que la serie de datos de entrada se suministra en un formato de dos columnas ponderadas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
TTest1w_upper (weight, value [, sig])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Las muestras que se han de evaluar. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
weight	Cada valor en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
TTest1w_upper( Weight, Value )
TTest1w_upper( Weight, Value, 0.005 )
```

Vea también:

Crear un informe t-test típico (page 537)

Funciones de prueba Z

Un análisis estadístico de dos promedios de población. Una prueba z de dos muestras analiza si dos muestras son distintas y es muy habitual cuando dos distribuciones normales tienen variaciones conocidas y cuando un experimento usa un gran tamaño de muestra.

Las funciones estadísticas de prueba z se agrupan conforme al tipo de datos de entrada que se aplica a la función.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Ejemplos de cómo usar funciones z-test (page 541)

Funciones de formato de una columna

Las siguientes funciones se aplican a pruebas z con series de datos de entrada simples:

ztest_conf

ZTest_conf() devuelve el valor z agregado de una serie de valores.

```
ZTest_conf() devuelve el valor z agregado de una serie de valores. (value [, sigma [, sig ])
```

ztest_dif

ZTest_dif() devuelve la diferencia de medias de la prueba z agregada de una serie de valores.

```
ZTest_dif() devuelve la diferencia de medias de la prueba z agregada de una serie de valores. (value [, sigma])
```

ztest_sig

ZTest_sig() devuelve el nivel de significación de 2 colas agregado de la prueba z de una serie de valores.

```
ZTest_sig() devuelve el nivel de significación de 2 colas agregado de la prueba z de una serie de valores. (value [, sigma])
```

ztest_sterr

ZTest_sterr() devuelve el error estándar agregado de la prueba z de la diferencia de medias de una serie de valores.

```
ZTest_sterr() devuelve el error estándar agregado de la prueba z de la diferencia de medias de una serie de valores. (value [, sigma])
```

ztest_z

ZTest_z() devuelve el valor z agregado de una serie de valores.

```
ZTest_z() devuelve el valor z agregado de una serie de valores. (value [, sigma])
```

ztest_lower

ZTest_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de dos series independientes de valores.

```
ZTest_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de dos series independientes de valores. (grp, value [, sig [, eq_var]])
```

ztest_upper

ZTest_upper() devuelve el valor agregado del límite superior del intervalo de confianza de dos series independientes de valores.

```
ZTest_upper() devuelve el valor agregado del límite superior del intervalo de confianza de dos series independientes de valores. (grp, value [, sig [, eq_var]])
```

Funciones de formato de dos columnas ponderadas

Las siguientes funciones se aplican a pruebas z donde la serie de datos de entrada se da en un formato de dos columnas ponderadas.

ztestw_conf

ZTestw_conf() devuelve el valor del intervalo de confianza z agregado de una serie de valores.

```
ZTestw_conf() devuelve el valor del intervalo de confianza z agregado de una serie de valores. (weight, value [, sigma [, sig]])
```

ztestw_dif

ZTestw_dif() devuelve la diferencia de medias de la prueba z agregada de una serie de valores.

```
ZTestw_dif() devuelve la diferencia de medias de la prueba z agregada de una serie de valores.  (weight, value [, sigma])
```

ztestw_lower

ZTestw_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de dos series independientes de valores.

```
ZTestw_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de dos series independientes de valores. (weight, value [, sigma])
```

ztestw_sig

ZTestw_sig() devuelve el nivel de significación de 2 colas agregado de la prueba z de una serie de valores.

```
ZTestw_sig() devuelve el nivel de significación de 2 colas agregado de la prueba z de una serie de valores. (weight, value [, sigma])
```

ztestw_sterr

ZTestw_sterr() devuelve el error estándar agregado de la prueba z de la diferencia de medias de una serie de valores.

```
ZTestw_sterr() devuelve el error estándar agregado de la prueba z de la diferencia de medias de una serie de valores. (weight, value [, sigma])
```

ztestw_upper

ZTestw_upper() devuelve el valor agregado del límite superior del intervalo de confianza de dos series independientes de valores.

```
ZTestw_upper() devuelve el valor agregado del límite superior del intervalo de confianza de dos series independientes de valores. (weight, value [, sigma])
```

ztestw_z

ZTestw_z() devuelve el valor z agregado de una serie de valores.

```
ZTestw_z() devuelve el valor z agregado de una serie de valores. (weight, value [, sigma])
```

ZTest_z

ZTest_z() devuelve el valor z agregado de una serie de valores.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
ZTest_z(value[, sigma])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Se supone una media de población de 0. Si desea que el test se realice conforme a otra media distinta, reste dicha media valor de los valores de muestra.
sigma	Si se conoce, la desviación estándar se puede indicar en **sigma**. Si se omite **sigma**, se utilizará la desviación estándar real de la muestra.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
ZTest_z( Value-TestValue )
```

Vea también:

Ejemplos de cómo usar funciones z-test (page 541)

ZTest_sig

ZTest_sig() devuelve el nivel de significación de 2 colas agregado de la prueba z de una serie de valores.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
ZTest_sig(value[, sigma])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Se supone una media de población de 0. Si desea que el test se realice conforme a otra media distinta, reste dicha media valor de los valores de muestra.
sigma	Si se conoce, la desviación estándar se puede indicar en **sigma**. Si se omite **sigma**, se utilizará la desviación estándar real de la muestra.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
ZTest_sig(Value-TestValue)
```

Vea también:

Ejemplos de cómo usar funciones z-test (page 541)

ZTest_dif

ZTest_dif() devuelve la diferencia de medias de la prueba z agregada de una serie de valores.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
ZTest_dif(value[, sigma])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Se supone una media de población de 0. Si desea que el test se realice conforme a otra media distinta, reste dicha media valor de los valores de muestra.
sigma	Si se conoce, la desviación estándar se puede indicar en **sigma**. Si se omite **sigma**, se utilizará la desviación estándar real de la muestra.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
ZTest_dif(Value-TestValue)
```

Vea también:

Ejemplos de cómo usar funciones z-test (page 541)

ZTest_sterr

ZTest_sterr() devuelve el error estándar agregado de la prueba z de la diferencia de medias de una serie de valores.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
ZTest_sterr(value[, sigma])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Se supone una media de población de 0. Si desea que el test se realice conforme a otra media distinta, reste dicha media valor de los valores de muestra.
sigma	Si se conoce, la desviación estándar se puede indicar en **sigma**. Si se omite **sigma**, se utilizará la desviación estándar real de la muestra.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
ZTest_sterr(Value-TestValue)
```

Vea también:

Ejemplos de cómo usar funciones z-test (page 541)

ZTest_conf

ZTest_conf() devuelve el valor z agregado de una serie de valores.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
ZTest_conf(value[, sigma[, sig]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Se supone una media de población de 0. Si desea que el test se realice conforme a otra media distinta, reste dicha media valor de los valores de muestra.
sigma	Si se conoce, la desviación estándar se puede indicar en **sigma**. Si se omite **sigma**, se utilizará la desviación estándar real de la muestra.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
ZTest_conf(Value-TestValue)
```

Vea también:

Ejemplos de cómo usar funciones z-test (page 541)

ZTest_lower

ZTest_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de dos series independientes de valores.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
ZTest_lower (grp, value [, sig [, eq_var]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
ZTest_lower( Group, Value )
ZTest_lower( Group, Value, sig, false )
```

Vea también:

Ejemplos de cómo usar funciones z-test (page 541)

ZTest_upper

ZTest_upper() devuelve el valor agregado del límite superior del intervalo de confianza de dos series independientes de valores.

Esta función se aplica a pruebas t de estudiante de muestras independientes.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
ZTest_upper (grp, value [, sig [, eq_var]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
ZTest_upper( Group, Value )
ZTest_upper( Group, Value, sig, false )
```

Vea también:

Ejemplos de cómo usar funciones z-test (page 541)

ZTestw_z

ZTestw_z() devuelve el valor z agregado de una serie de valores.

Esta función se aplica a las pruebas z en las que la serie de datos de entrada se proporciona en un formato ponderado de dos columnas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
ZTestw_z (weight, value [, sigma])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores deben ser devueltos por **value**. Se asume una media de 0 como muestra. Si desea que el test se realice conforme a otra media distinta, reste dicho valor de los valores de muestra.
weight	Cada valor de muestra en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.
sigma	Si se conoce, la desviación estándar se puede indicar en **sigma**. Si se omite **sigma**, se utilizará la desviación estándar real de la muestra.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
ZTestw_z( Weight, Value-TestValue)
```

Vea también:

Ejemplos de cómo usar funciones z-test (page 541)

ZTestw_sig

ZTestw_sig() devuelve el nivel de significación de 2 colas agregado de la prueba z de una serie de valores.

Esta función se aplica a las pruebas z en las que la serie de datos de entrada se proporciona en un formato ponderado de dos columnas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
ZTestw_sig (weight, value [, sigma])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores deben ser devueltos por **value**. Se asume una media de 0 como muestra. Si desea que el test se realice conforme a otra media distinta, reste dicho valor de los valores de muestra.
weight	Cada valor de muestra en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.
sigma	Si se conoce, la desviación estándar se puede indicar en **sigma**. Si se omite **sigma**, se utilizará la desviación estándar real de la muestra.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
ZTestw_sig( Weight, Value-TestValue)
```

Vea también:

Ejemplos de cómo usar funciones z-test (page 541)

ZTestw_dif

ZTestw_dif() devuelve la diferencia de medias de la prueba z agregada de una serie de valores.

Esta función se aplica a las pruebas z en las que la serie de datos de entrada se proporciona en un formato ponderado de dos columnas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
ZTestw_dif ( weight, value [, sigma])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores deben ser devueltos por **value**. Se asume una media de 0 como muestra. Si desea que el test se realice conforme a otra media distinta, reste dicho valor de los valores de muestra.
weight	Cada valor de muestra en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.
sigma	Si se conoce, la desviación estándar se puede indicar en **sigma**. Si se omite **sigma**, se utilizará la desviación estándar real de la muestra.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
ZTestw_dif( Weight, Value-TestValue)
```

Vea también:

Ejemplos de cómo usar funciones z-test (page 541)

ZTestw_sterr

ZTestw_sterr() devuelve el error estándar agregado de la prueba z de la diferencia de medias de una serie de valores.

Esta función se aplica a las pruebas z en las que la serie de datos de entrada se proporciona en un formato ponderado de dos columnas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
ZTestw_sterr (weight, value [, sigma])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores deben ser devueltos por **value**. Se asume una media de 0 como muestra. Si desea que el test se realice conforme a otra media distinta, reste dicho valor de los valores de muestra.
weight	Cada valor de muestra en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.
sigma	Si se conoce, la desviación estándar se puede indicar en **sigma**. Si se omite **sigma**, se utilizará la desviación estándar real de la muestra.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
ZTestw_sterr( Weight, Value-TestValue)
```

Vea también:

Ejemplos de cómo usar funciones z-test (page 541)

ZTestw_conf

ZTestw_conf() devuelve el valor del intervalo de confianza z agregado de una serie de valores.

Esta función se aplica a las pruebas z en las que la serie de datos de entrada se proporciona en un formato ponderado de dos columnas.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
ZTest_conf(weight, value[, sigma[, sig]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Se supone una media de población de 0. Si desea que el test se realice conforme a otra media distinta, reste dicha media valor de los valores de muestra.
weight	Cada valor de muestra en **value** se puede contar una o más veces de acuerdo con un valor de peso correspondiente en **weight**.
sigma	Si se conoce, la desviación estándar se puede indicar en **sigma**. Si se omite **sigma**, se utilizará la desviación estándar real de la muestra.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplo:

```
ZTestw_conf( Weight, Value-TestValue)
```

Vea también:

Ejemplos de cómo usar funciones z-test (page 541)

ZTestw_lower

ZTestw_lower() devuelve el valor agregado del límite inferior del intervalo de confianza de dos series independientes de valores.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
ZTestw_lower (grp, value [, sig [, eq_var]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
ZTestw_lower( Group, Value )
ZTestw_lower( Group, Value, sig, false )
```

Vea también:

Ejemplos de cómo usar funciones z-test (page 541)

ZTestw_upper

ZTestw_upper() devuelve el valor agregado del límite superior del intervalo de confianza de dos series independientes de valores.

Esta función se aplica a pruebas t de estudiante de muestras independientes.

Si la función se utiliza en el script de carga de datos, los valores se repiten a lo largo de varios registros definidos por una cláusula group by.

Si la función se utiliza en una expresión de gráfico, los valores se repiten a lo largo de las dimensiones del gráfico.

Sintaxis:

```
ZTestw_upper (grp, value [, sig [, eq_var]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
value	Los valores de muestra que se han de evaluar. Los valores de muestra deben agruparse lógicamente según lo especificado por exactamente dos valores en **group**. Si no se proporciona un nombre de campo para los valores de muestra en el script de carga, el campo se denominará automáticamente **Value**.
grp	El campo que contiene los nombres de cada uno de los dos grupos de muestra. Si no se proporciona un nombre de campo para el grupo en el script de carga, el campo recibirá automáticamente el nombre de **Type**.
sig	El nivel de significación de dos colas se puede especificar en **sig**. Si se omite, **sig** se establece en 0,025, lo que resulta en un intervalo de confianza del 95%.
eq_var	Si **eq_var** se especifica como False (0), se supondrán varianzas separadas de las dos muestras. Si **eq_var** se especifica como True (1), se supondrán varianzas iguales de las dos muestras.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos en el valor de la expresión harán que la función devuelva NULL como resultado.

Ejemplos:

```
ZTestw_upper( Group, Value )
ZTestw_upper( Group, Value, sig, false )
```

Vea también:

Ejemplos de cómo usar funciones z-test (page 541)

Ejemplos de funciones estadísticas de prueba

En esta sección aparecen ejemplos de funciones estadísticas de prueba aplicadas a gráficos, así como de scripts de carga de datos.

Ejemplos de cómo usar funciones chi2-test en gráficos

Las funciones chi2-test se utilizan para hallar valores asociados con el análisis estadístico de chi cuadrado.

Esta sección describe cómo construir visualizaciones utilizando datos de muestra para hallar los valores de las funciones de prueba de distribución de Chi al cuadrado disponibles en Qlik Sense. Consulte los temas individuales de la función de gráfico chi2-test si desea descripciones de la sintaxis y los argumentos.

Cargar los datos para las muestras

Hay tres conjuntos de datos de muestra que describen tres muestras estadísticas diferentes para cargarlas en el script.

Haga lo siguiente:

1. Cree una nueva app.

   Al cargar datos, introduzca lo siguiente:

   ```
   // 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. Haga clic en para cargar datos.

Crear las visualizaciones de la función de gráfico chi2-test

Ejemplo: Muestra 1

Haga lo siguiente:

1. En el editor de carga de datos, haga clic en para ir a la vista de app y después haga clic en la hoja que creó anteriormente.
   Se abre la vista de hoja.
2. Haga clic en **Editar hoja** para editar la hoja.
3. Desde **Gráficos** añada una tabla, y desde **Campos** añada Grp, Grade y Count como dimensiones.
   Esta tabla muestra los datos de muestra.
4. Añada otra tabla con la siguiente expresión como dimensión.
   `ValueList('p','df','Chi2')`
   Esto utiliza la función de dimensiones sintéticas para crear etiquetas para las dimensiones con los nombres de las tres funciones chi2-test.
5. Añada la siguiente expresión a la tabla como medida.
   ```
   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)))
   ```
   Esto tiene el efecto de colocar el valor resultante de cada función chi2-test en la tabla junto a su dimensión sintética asociada.
6. Defina el **Formato numérico** de la medida en **Número** y **3 Cifras significativas**.

> *En la expresión para la medida, podría usar la siguiente expresión en su lugar:* `Pick(Match(ValueList('p','df','Chi2'),'p','df','Chi2'),Chi2Test_p(Grp,Grade,Count),Chi2Test_df(Grp,Grade,Count),Chi2Test_Chi2(Grp,Grade,Count))`

Resultado:

La tabla resultante para las funciones chi2-test de los datos de la Muestra 1 contendrá los siguientes valores:

Tabla de resultados

p	df	Chi2
0.820	5	2.21

Ejemplo: Muestra 2

Haga lo siguiente:

1. En la hoja que estaba editando en el ejemplo Muestra 1, desde **Gráficos** agregue una tabla, y desde **Campos** agregue, Sex, Opinion y OpCount como dimensiones.
2. Haga una copia de la tabla de resultados de la Muestra 1 utilizando los comandos **Copiar** y **Pegar**. Edite la expresión en la medida y reemplace los argumentos de las tres funciones chi2-test por los nombres de los campos utilizados en los datos de la Muestra 2, por ejemplo: `Chi2Test_p (Sex,Opinion,OpCount)`.

Resultado:

La tabla resultante para las funciones chi2-test de los datos de la Muestra 2 contendrá los siguientes valores:

Tabla de resultados

p	df	Chi2
0.000309	2	16.2

Ejemplo: Muestra 3

Haga lo siguiente:

1. Cree dos tablas más de la misma manera que en los ejemplos de los datos para la Muestra 1 y Muestra 2. En la tabla de dimensiones, utilice los siguientes campos como dimensiones: Gender, Description, Actual y Expected.
2. En la tabla de resultados, use los nombres de los campos utilizados en los datos de la Muestra 3, por ejemplo: `Chi2Test_p(Gender,Description,Actual,Expected)`.

Resultado:

La tabla resultante para las funciones chi2-test de los datos de la Muestra 3 contendrá los siguientes valores:

Tabla de resultados

p	df	Chi2
0.000308	2	16.2

Ejemplos de cómo usar funciones chi2-test en el script de carga de datos

Las funciones chi2-test se utilizan para hallar valores asociados con el análisis estadístico de chi cuadrado. En esta sección se describe cómo utilizar las funciones de prueba de distribución de chi al cuadrado disponibles en Qlik Sense, en el script de carga de datos. Consulte los temas individuales de la función de gráfico chi2-test si desea descripciones de la sintaxis y los argumentos.

Este ejemplo utiliza una tabla que contiene el número de alumnos que obtienen una nota (A-F) para dos grupos de estudiantes (I y II).

Data table

Group	A	B	C	D	E	F
I	15	7	9	20	26	19
II	10	11	7	15	21	16

Cargar los datos de muestra

Haga lo siguiente:

1. Cree una nueva app.

   En el editor de carga de datos, introduzca lo siguiente:
   ```
   // 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. Haga clic en para cargar datos.

Ahora ya hemos cargado los datos de muestra.
   ```

Cargar los valores de la función chi2-test

Ahora cargaremos los valores chi2-test basados en los datos de muestra en una nueva tabla, agrupados por Grp.

Haga lo siguiente:

1. En el editor de carga de datos, añada lo siguiente al final del script:

```
// Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the top of the script.

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. Haga clic en para cargar datos.

Ahora ha cargado los valores chi2-test en una tabla denominada Chi2_table.

Resultados

Puede ver los valores chi2-test resultantes en el visor de modelo de datos, en **Vista previa**, deberán presentar el siguiente aspecto:

Results

Grp	chi2	df	p
I	16.00	5	0.007
II	9.40	5	0.094

Crear un informe t-test típico

Un informe t-test típico de estudiante puede incluir tablas con **Group Statistics** y resultados **Independent Samples Test**.

En las siguientes secciones construiremos estas tablas mediante funciones Qlik Senset-test aplicadas a dos grupos independientes de muestras, Observation y Comparison. Las tablas correspondientes para estas muestras tendrán el siguiente aspecto:

Estadísticas de los grupos

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Prueba de muestra independiente

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

Cargar los datos de muestra

Haga lo siguiente:

1. Cree una nueva app con una nueva hoja.
2. Introduzca lo siguiente en el editor de carga de script:
```
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 '|');
```
   En este script de carga se incluye **recno()** porque **crosstable** requiere tres argumentos. De modo que **recno()** simplemente proporciona un argumento adicional, en este caso un ID para cada fila. Sin el mismo, los valores de muestra de **Comparison** no se cargarían.
3. Haga clic en para cargar datos.

Crear la tabla Group statistics

Haga lo siguiente:

1. En el editor de carga de datos, haga clic en para ir a la vista de app y después haga clic en la hoja que creó anteriormente.
   Así se abre la vista de hoja.
2. Haga clic en **Editar hoja** para editar la hoja.
3. Desde **Gráficos**, agregue una tabla y desde **Campos**, agregue Type como una dimensión a la tabla.
4. Agregue las siguientes expresiones como medidas.

Expresiones de ejemplo

Etiqueta	Expresión
N	Count(Value)
Mean	Avg(Value)
Standard Deviation	Stdev(Value)
Standard Error Mean	Sterr(Value)

5. Haga clic en **Ordenar** y asegúrese de que Type esté en la parte superior de la lista de ordenación.

Resultado:

Una tabla Group statistics correspondiente a estas muestras tendrá el siguiente aspecto:

Estadísticas de los grupos

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Crear la tabla Independent sample test

Haga lo siguiente:

1. Haga clic en **Editar hoja** para editar la hoja.
2. Desde **Gráficos** agregue una tabla con la siguiente expresión como una dimensión a la tabla. `=ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1))` y dele la etiqueta Tipo.

3. Agregue las siguientes expresiones como medidas:

Expresiones de ejemplo

Etiqueta	Expresión
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))

Resultado:

Prueba de muestra independiente

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

Ejemplos de cómo usar funciones z-test

Las funciones z-test se usan para hallar valores asociados con el análisis estadístico de prueba z z-test para grandes muestras de datos, generalmente mayores de 30, y donde se conoce la varianza.

Esta sección describe cómo construir visualizaciones utilizando datos de muestra para hallar los valores de las funciones z-test disponibles en Qlik Sense. Consulte los temas individuales de la función de gráfico z-test si desea descripciones de la sintaxis y los argumentos.

Cargar los datos de muestra

Los datos de muestra utilizados aquí son los mismos que los utilizados en los ejemplos de las funciones t-test. El tamaño de los datos de la muestra normalmente se consideraría demasiado pequeño para el análisis de prueba z, pero es suficiente para ilustrar el uso de las diferentes funciones z-test en Qlik Sense.

Haga lo siguiente:

1. Cree una nueva app con una nueva hoja.

   Si creó una app para las funciones t-test, podría usarla y crear una nueva hoja para estas funciones.

2. En el editor de carga de datos, introduzca lo siguiente:

```
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 '|');
```

   En este script de carga se incluye **recno()** porque **crosstable** requiere tres argumentos. De modo que **recno()** simplemente proporciona un argumento adicional, en este caso un ID para cada fila. Sin el

mismo, los valores de muestra de **Comparison** no se cargarían.

3. Haga clic en para cargar datos.

Crear la tabla z-test

Haga lo siguiente:

1. En el editor de carga de datos, haga clic en para ir a la vista de app y después haga clic en la hoja que creó anteriormente.
   Se abre la vista de hoja.
2. Haga clic en **Editar hoja** para editar la hoja.
3. Desde **Gráficos** añada una tabla y desde **Campos** añada Type como dimensión.
4. Añada las siguientes expresiones a la tabla como medidas.

Expresiones de ejemplo

Etiqueta	Expresión
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)

*Puede que desee ajustar el formato numérico de las medidas para tener unos valores con sentido. La tabla será más fácil de leer si se establece el formato numérico en la mayoría de las medidas como **Número>Simple** en lugar de **Automático**. Pero para ZTest Sig, por ejemplo, utilice el formato de número: **Personalizado**, y luego ajuste el patrón de formato a **#.######**.*

Resultado:

La tabla resultante para las funciones z-test de los datos de muestra contendrá los siguientes valores:

z-test tabla de resultados

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

Crear la tabla z-testw

Las funciones z-testw son para usarlas cuando la serie de datos de entrada ocurre en formato ponderado de dos columnas. Las expresiones requieren un valor para el argumento weight.

Los ejemplos aquí utilizan el valor 2 en todo momento, pero podría usar una expresión, que definiría un valor para cada observación de weight.

Haga lo siguiente:

1. En el editor de carga de datos, haga clic en para ir a la vista de app y después haga clic en la hoja que creó anteriormente.
   Se abre la vista de hoja.
2. Haga clic en **Editar hoja** para editar la hoja.
3. Desde **Gráficos** añada una tabla y desde **Campos** añada Type como dimensión.
4. Añada las siguientes expresiones a la tabla como medidas.

Expresiones de ejemplo

Etiqueta	Expresión
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)

Utilice el mismo formato numérico que en el ejemplo de las funciones z-test.

Resultado:

La tabla resultante para las funciones z-testw contendrá los siguientes valores:

z-testw tabla de resultados

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

Funciones de agregación de cadena

En esta sección se describen funciones de agregación relativas a cadenas.

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Funciones de agregación de cadenas en el script de carga de datos

Concat

Concat() se utiliza para combinar valores de cadena. La función de script devuelve la concatenación de cadena agregada de todos los valores de la expresión iterada en una serie de registros, tal como se define en una cláusula **group by**.

```
Concat ([ distinct ] expression [, delimiter [, sort-weight]])
```

FirstValue
FirstValue() devuelve el valor que se cargó primero de los registros definidos por la expresión, ordenados por una cláusula **group by**.

Esta función solo está disponible como función de script.

```
FirstValue (expression)
```

LastValue
LastValue() devuelve el último valor que se cargó de los registros definidos por la expresión, ordenados por una cláusula **group by**.

Esta función solo está disponible como función de script.

```
LastValue (expression)
```

MaxString
MaxString() busca valores de cadenas de caracteres en la expresión y devuelve el último valor de texto ordenado alfabéticamente en varios registros, según lo definido por una cláusula **group by**.

```
MaxString (expression )
```

MinString
MinString() busca valores de cadenas de caracteres en la expresión y devuelve el primer valor de texto ordenado alfabéticamente en varios registros, según lo definido por una cláusula **group by**.

```
MinString (expression )
```

Funciones de agregación de cadenas en gráficos

Las siguientes funciones de gráficos están disponibles para agregación de cadenas en gráficos.

Concat
Concat() se utiliza para combinar valores de cadena. Esta función devuelve la concatenación agregada de cadenas de todos los valores de la expresión que se evalúan en cada dimensión.

```
Concat - función de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld{,
fld}>]] string[, delimiter[, sort_weight]])
```

MaxString
MaxString() busca valores de cadenas de caracteres en la expresión o campo y devuelve el último valor de texto en orden alfabético.

```
MaxString - función de gráfico({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MinString
MinString() busca valores de cadenas de caracteres en la expresión o campo y devuelve el primer valor de texto en orden alfabético.

```
MinString - función de gráfico({[SetExpression] [TOTAL [<fld {, fld}>]]}
expr)
```

Concat

Concat() se utiliza para combinar valores de cadena. La función de script devuelve la concatenación de cadena agregada de todos los valores de la expresión iterada en una serie de registros, tal como se define en una cláusula **group by**.

Sintaxis:

```
Concat ([ distinct ] string [, delimiter [, sort-weight]])
```

Tipo de datos que devuelve: cadena

Argumentos:

La expresión o campo que contiene la cadena que se ha de procesar.

Argumentos

Argumento	Descripción
string	La expresión o campo que contiene la cadena que se ha de procesar.
delimiter	Cada valor puede estar separado por la cadena que se encuentra en delimiter.
sort-weight	El orden de concatenación puede ir determinado por el valor de la dimensión **sort-weight**, si está presente, con la cadena correspondiente al valor más bajo que aparece primero en la concatenación.
distinct	Si la palabra **distinct** figura antes de la expresión, todos los duplicados se descartan.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

Ejemplos y resultados

Ejemplo	Resultado	Resultados una vez agregados a una hoja

`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

Dado que la tabla **TeamData** se carga como en el ejemplo anterior: `LOAD SalesGroup,Concat(distinct Team,'-') as TeamConcat2 Resident TeamData Group By SalesGroup;`	SalesGroup East West	TeamConcat2 Alpha-Beta-Delta-Gamma Epsilon-Eta-Theta-Zeta
Dado que la tabla **TeamData** se carga como en el ejemplo anterior: Como se agrega el argumento de **sort-weight**, los resultados se ordenan por el valor de la dimensión 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 - función de gráfico

Concat() se utiliza para combinar valores de cadena. Esta función devuelve la concatenación agregada de cadenas de todos los valores de la expresión que se evalúan en cada dimensión.

Sintaxis:

```
Concat({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} string[, delimiter [, sort_weight]])
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
string	La expresión o campo que contiene la cadena que se ha de procesar.
delimiter	Cada valor puede estar separado por la cadena que se encuentra en delimiter.
sort-weight	El orden de concatenación puede ir determinado por el valor de la dimensión **sort-weight**, si está presente, con la cadena correspondiente al valor más bajo que aparece primero en la concatenación.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si la palabra **DISTINCT** aparece antes de los argumentos de la función, los duplicados resultantes de evaluar los argumentos de la función se descartan.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Ejemplos y resultados:

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

Ejemplos de funciones

Ejemplo	Resultado
`Concat(Team)`	La tabla se construye a partir de las dimensiones SalesGroup y Amount, y las variaciones en la medida Concat(Team). Ignorando el resultado de los Totales, tenga en cuenta que aunque hay datos para ocho valores de Team distribuidos en dos valores de SalesGroup, el único resultado de la medida Concat(Team) que concatena más de un valor de cadena Team en la tabla es la fila que contiene la dimensión Amount, que da como resultado BetaGammaGamma. Esto es porque hay tres valores para Amount 20000 en los datos de entrada. Todos los demás resultados permanecen sin concatenar cuando la medida se extiende a lo largo de las dimensiones porque solo hay un valor de Team para cada combinación de SalesGroup y Amount.
`Concat (DISTINCT Team,', ')`	Beta, Gamma, porque el cualificador DISTINCT significa que el resultado Gamma duplicado no se tiene en cuenta. Además, el argumento delimitador se define como una coma seguida por un espacio.
`Concat (TOTAL <SalesGroup> Team)`	Todos los valores de cadena de todos los valores de Team se concatenan si se utiliza el cualificador TOTAL. Con la selección de campos <SalesGroup> especificada, esto divide los resultados en los dos valores de la dimensión SalesGroup. Para SalesGroupEast, los resultados son AlphaBetaDeltaGammaGamma. Para SalesGroupWest, los resultados son EpsilonEtaThetaZeta.
`Concat (TOTAL <SalesGroup> Team,';', Amount)`	Añadiendo el argumento para **sort-weight**: Amount, los resultados se ordenan por el valor de la dimensión. Amount. Los resultados son DeltaBetaGammaGammaAlpha y EtaEpsilonZEtaTheta.

Datos utilizados en el ejemplo:

```
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() devuelve el valor que se cargó primero de los registros definidos por la expresión, ordenados por una cláusula **group by**.

Esta función solo está disponible como función de script.

Sintaxis:

```
FirstValue ( expr)
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.

Limitaciones:

Si no encuentra ningún resultado, devuelve NULL.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

Datos resultantes

Ejemplo	Resultado	Resultados en una hoja

`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() devuelve el último valor que se cargó de los registros definidos por la expresión, ordenados por una cláusula **group by**.

Esta función solo está disponible como función de script.

Sintaxis:

```
LastValue ( expr )
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.

Limitaciones:

Si no encuentra ningún resultado, devuelve NULL.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

Para tener el mismo aspecto que en la columna inferior de resultados, en el panel de propiedades, bajo Ordenar, cambie de Auto a Personalizado, a continuación deseleccione el orden numérico y alfabético.

Ejemplo	Resultado	Resultado con la ordenación personalizada
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() busca valores de cadenas de caracteres en la expresión y devuelve el último valor de texto ordenado alfabéticamente en varios registros, según lo definido por una cláusula **group by**.

Sintaxis:

```
MaxString ( expr )
```

Tipo de datos que devuelve: dual

Argumentos:

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.

Limitaciones:

Si no encuentra ningún resultado, devuelve NULL.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

Ejemplo	Resultado	
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
Dado que la tabla **TeamData** se carga como en el ejemplo anterior y su script de carga de datos contiene la sentencia SET: 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 - función de gráfico

MaxString() busca valores de cadenas de caracteres en la expresión o campo y devuelve el último valor de texto en orden alfabético.

Sintaxis:

```
MaxString({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Limitaciones:

Si la expresión no contiene valores, devuelve una representación de cadena NULL.

Ejemplos y resultados:

Tabla de resultados

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

Ejemplos de funciones

Ejemplo	Resultado
`MaxString (Team)`	Hay tres valores de 20000 para la dimensión Amount: dos de Gamma (en diferentes fechas) y uno de Beta. El resultado de la medida MaxString (Team) es por tanto Gamma, porque este es el valor más alto en las cadenas ordenadas.
`MaxString (Date)`	2013/11/01 es el valor Date más alto de las tres asociadas con la dimensión Amount. Esto da por sentado que su script contiene la sentencia SET `SET DateFormat='YYYY-MM-DD';'`

Datos utilizados en el ejemplo:

```
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() busca valores de cadenas de caracteres en la expresión y devuelve el primer valor de texto ordenado alfabéticamente en varios registros, según lo definido por una cláusula **group by**.

Sintaxis:

```
MinString ( expr )
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.

Limitaciones:

Si no encuentra ningún resultado, devuelve NULL.

Ejemplos y resultados:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

Datos resultantes

Ejemplo	Resultado	
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
Dado que la tabla **TeamData** se carga como en el ejemplo anterior y su script de carga de datos contiene la sentencia SET: 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 - función de gráfico

MinString() busca valores de cadenas de caracteres en la expresión o campo y devuelve el primer valor de texto en orden alfabético.

Sintaxis:

```
MinString({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
TOTAL	Si la palabra **TOTAL** aparece antes de los argumentos de la función, el cálculo se realiza sobre todos los valores posibles dadas las selecciones actuales y no solo aquellas que pertenecen al valor dimensional actual, es decir, no tiene en cuenta las dimensiones del gráfico. Usar **TOTAL [<fld {.fld}>]**, donde al cualificador **TOTAL** le sigue una lista de uno o más nombres de campo como un subconjunto de las variables de dimensión del gráfico, crea un subconjunto de los valores totales posibles.

Ejemplos y resultados:

Datos de muestra

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

Ejemplos de funciones

Ejemplos	Resultados
`MinString (Team)`	Hay tres valores de 20000 para la dimensión Amount: dos de Gamma (en diferentes fechas) y uno de Beta. El resultado de la medida MinString (Team) es por tanto Beta, porque este es el primer valor en las cadenas ordenadas.
`MinString (Date)`	2013/11/01 es el primer valor de fecha Date de los tres asociados con la dimensión Amount. Esto da por sentado que su script contiene la sentencia SET `SET DateFormat='YYYY-MM-DD';'`

Datos utilizados en el ejemplo:

```
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 '|');
```

Funciones para dimensiones sintéticas

Una dimensión sintética se crea en la app a partir de los valores generados desde las funciones para dimensiones sintéticas y no directamente desde campos en el modelo de datos. Cuando se utilizan valores generados por una función de dimensiones sintéticas en un gráfico como dimensión calculada, se crea una dimensión sintética. Las dimensiones sintéticas permiten crear, por ejemplo, gráficos con dimensiones a partir de valores derivados de los datos, es decir, dimensiones dinámicas.

Las dimensiones sintéticas no se ven afectadas por las selecciones.

Se pueden utilizar las siguientes funciones de dimensiones sintéticas en gráficos.

ValueList

ValueList() devuelve un conjunto de valores listados que, cuando se usan en una dimensión calculada, formarán una dimensión sintética.

```
ValueList - función de gráfico (v1 {, Expression})
```

ValueLoop

ValueLoop() devuelve un conjunto de valores iterados que, cuando se usen en una dimensión calculada, formarán una dimensión sintética.

```
ValueLoop - función de gráfico(from [, to [, step ]])
```

ValueList - función de gráfico

ValueList() devuelve un conjunto de valores listados que, cuando se usan en una dimensión calculada, formarán una dimensión sintética.

*En los gráficos con una dimensión sintética creada con la función **ValueList**, es posible hacer referencia al valor de dimensión correspondiente a una celda de expresión específica replanteando la función **ValueList** con los mismos parámetros en la expresión del gráfico. La función, por supuesto, puede utilizarse en cualquier parte del diseño, pero aparte de su uso en las dimensiones sintéticas, solo tendrá sentido dentro de una función de agregación.*

Las dimensiones sintéticas no se ven afectadas por las selecciones.

Sintaxis:

```
ValueList(v1 {,...})
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
v1	Valor estático (suele ser una cadena, pero puede ser un número).
{,...}	Lista opcional de valores estáticos.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplo	Resultado
`ValueList ('Number of Orders', 'Average Order Size', 'Total Amount')`	Cuando se utiliza para crear una dimensión en una tabla, por ejemplo, esto da como resultado los tres valores de cadena como etiquetas de fila en la tabla. A continuación se puede hacer referencia a ellos en una expresión.
`=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) ))`	Esta expresión toma los valores de la dimensión creada y los referencia en una sentencia IF anidada como entrada a tres funciones de agregación: **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

Datos utilizados en los ejemplos:

`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 - función de gráfico

ValueLoop() devuelve un conjunto de valores iterados que, cuando se usen en una dimensión calculada, formarán una dimensión sintética.
Los valores generados comenzarán con el valor **from** y terminarán con el valor **to**, incluidos los valores intermedios en incrementos de paso.

*En los gráficos con una dimensión sintética creada con la función **ValueLoop**, es posible hacer referencia al valor de dimensión correspondiente a una celda de expresión específica replanteando la función **ValueLoop** con los mismos parámetros en la expresión del gráfico. La función, por supuesto, puede utilizarse en cualquier parte del diseño, pero aparte de su uso en las dimensiones sintéticas, solo tendrá sentido dentro de una función de agregación.*

Las dimensiones sintéticas no se ven afectadas por las selecciones.

Sintaxis:

```
ValueLoop(from [, to [, step ]])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumentos	Descripción
from	Valor inicial del conjunto de valores que se han de generar.
to	Valor final del conjunto de valores que se han de generar.
step	Tamaño del incremento entre valores.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplo	Resultado
`ValueLoop (1, 10)`	De este modo se crea una dimensión en una tabla, por ejemplo, que se puede utilizar para fines tales como la creación de etiquetas numeradas. Este ejemplo se traduce en valores numerados del 1 al 10. A continuación se puede hacer referencia a estos valores en una expresión.
`ValueLoop (2, 10,2)`	Este ejemplo da como resultado valores numerados 2, 4, 6, 8 y 10 porque el argumento step tiene un valor de 2.

Agregaciones anidadas

Puede que nos encontremos con situaciones en las que necesitemos aplicar una agregación al resultado de otra agregación. Esto se conoce con el nombre de agregaciones anidadas.

No puede anidar agregaciones en la mayoría de las expresiones de gráfico. Sin embargo, sí puede anidar agregaciones si utiliza el calificador **TOTAL** en la función de agregación interna.

No se permiten más de 100 niveles de anidación.

Agregaciones anidadas con el cualificador TOTAL

Ejemplo:

Desea calcular la suma del campo **Sales**, pero solo incluir transacciones con una **OrderDate** igual al año pasado. El último año se puede obtener mediante la función de agregación. **`Max(TOTAL Year (OrderDate))`**.

La agregación siguiente arrojaría el resultado deseado:

```
Sum(If(Year(OrderDate)=Max(TOTAL Year(OrderDate)), Sales))
```

Qlik Sense requiere la inclusión del calificador **TOTAL** en este tipo de anidamiento. Es necesario para la comparación deseada. Este tipo de necesidad de anidamiento es bastante común y debería emplearse siempre que sea posible.

Vea también:

Aggr - función de gráfico (page 559)

8.3 Aggr - función de gráfico

Aggr() devuelve un conjunto de valores para la expresión calculada sobre la dimensión o dimensiones indicadas. Por ejemplo, el valor máximo de ventas, por cliente, por región.

La función **Aggr** se utiliza para agregaciones anidadas, en las que su primer parámetro (la agregación interna) se calcula una vez por valor dimensional. Las dimensiones se especifican en el segundo parámetro (y los parámetros subsiguientes).

Además, la función **Aggr** debe incluirse en una función de agregación externa, utilizando la matriz de resultados de la función **Aggr** como entrada para la agregación en la que está anidada.

Sintaxis:

```
Aggr({SetExpression}[DISTINCT] [NODISTINCT ] expr, StructuredParameter{, StructuredParameter})
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
expr	Una expresión que consiste en una función de agregación. De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección.
StructuredParameter	StructuredParameter consiste en una dimensión y, opcionalmente, criterios de ordenación con el formato: `(Dimension(Sort-type, Ordering))` La dimensión es un único campo y no puede ser una expresión. La dimensión se utiliza para determinar la matriz de valores para los que se calcula la expresión Aggr. Si se incluyen criterios de ordenación, se ordena el conjunto de valores creados por la función Aggr, calculada para la dimensión. Esto es importante cuando el orden de clasificación afecta al resultado de la expresión en la que se incluye la función Aggr. Para ver detalles sobre cómo usar los criterios de ordenación, véase Añadir criterios de ordenación a la dimensión en el parámetro estructurado.
SetExpression	De forma predeterminada, la función de agregación agregará sobre el conjunto de registros posibles definidos por la selección. Se puede definir un conjunto alternativo de registros mediante una expresión de análisis de conjuntos.
DISTINCT	Si el argumento de la expresión va precedido por el cualificador **distinct** o si no se utiliza ningún cualificador en absoluto, cada combinación distinta de valores de dimensión generará un único valor de retorno. Esta es la forma habitual de hacer agregaciones: cada combinación distinta de valores de dimensión generará una línea en el gráfico.

Argumento	Descripción
NODISTINCT	Si el argumento de la expresión va precedido por el cualificador **nodistinct**, cada combinación de valores de dimensión podrá generar más de un valor de retorno, dependiendo de los valores subyacentes de la estructura de datos. Si solo hay una dimensión, la función **aggr** devolverá una matriz con el mismo número de elementos que filas hay en los datos fuente.

Las funciones básicas de agregación, como **Sum**, **Min** y **Avg**, devuelven un único valor numérico, mientras que la función Aggr() se puede comparar con crear un conjunto temporal de resultados (una tabla virtual), sobre el que se puede realizar otra agregación. Por ejemplo, calculando un valor de ventas promedio al sumar las ventas por cliente en una sentencia **Aggr()** y calcular después la media de la suma de resultados: **Avg(TOTAL Aggr(Sum(Sales),Customer))**.

Utilice la función Aggr() en dimensiones calculadas si desea crear agregaciones de gráfico anidadas en múltiples niveles.

Limitaciones:

Cada dimensión en una función Aggr() debe ser un único campo y no puede ser una expresión (dimensión calculada).

Añadir criterios de ordenación a la dimensión en el parámetro estructurado

En su forma básica, el argumento StructuredParameter en la sintaxis de la función Aggr es una única dimensión. La expresión: Aggr(Sum(Sales, Month)) halla el valor total de las ventas de cada mes. No obstante, cuando se incluye dentro de otra función de agregación puede dar resultados inesperados, a menos que se utilicen criterios de ordenación. Esto se produce porque algunas dimensiones pueden ordenarse numérica o alfabéticamente, etc.

En el argumento StructuredParameter de la función Aggr, puede especificar criterios de ordenación en la dimensión de su expresión. De esta forma, impondrá un orden de clasificación a la tabla virtual que produce la función Aggr.

El argumento StructuredParameter tiene la sintaxis siguiente:

```
(FieldName, (Sort-type, Ordering))
```

Los parámetros estructurados pueden anidarse:

```
(FieldName, (FieldName2, (Sort-type, Ordering)))
```

El tipo de clasificación puede ser: NUMERIC, TEXT, FREQUENCY o LOAD_ORDER.

Los tipos de ordenación asociados con cada tipo de clasificación son los siguientes:

Tipos de ordenación permitidos

Tipo de clasificación	Tipos de ordenación permitidos
NUMERIC	ASCENDING, DESCENDING o REVERSE
TEXT	ASCENDING, A2Z, DESCENDING, REVERSE o Z2A
FREQUENCY	DESCENDING, REVERSE o ASCENDING
LOAD_ORDER	ASCENDING, ORIGINAL, DESCENDING o REVERSE

Los tipos de ordenación REVERSE y DESCENDING son equivalentes.

Para el tipo de clasificación TEXT, los tipos de ordenación ASCENDING y A2Z son equivalentes, y DESCENDING, REVERSE y Z2A son equivalentes.

Para el tipo de clasificación LOAD_ORDER, los tipos de ordenación ASCENDING y ORIGINAL son equivalentes.

Ejemplos: Expresiones de gráfico que usan Aggr

Ejemplos: expresiones de gráfico

Ejemplo 1 de expresión de gráfico

Script de carga

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear el ejemplo de expresión de gráfico a continuación.

```
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 '|');
```

Expresión de gráfico

Cree una visualización de KPI en una hoja de Qlik Sense. Añada la siguiente expresión al KPI como medida:

```
Avg(Aggr(Sum(UnitSales*UnitPrice), Customer))
```

Resultado

376.7

Explicación

La expresión `Aggr(Sum(UnitSales*UnitPrice), Customer)` halla el valor total de ventas por **Customer**, y devuelve un conjunto de valores: 295, 715 y 120 por los tres valores de **Customer**.

Efectivamente, hemos construido una lista temporal de valores sin tener que crear una tabla o columna explícita que contenga dichos valores.

Estos valores se utilizan como datos de entrada en la función **Avg()** para hallar el valor promedio de las ventas, 376.7.

Ejemplo 2 de expresión de gráfico

Script de carga

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear el ejemplo de expresión de gráfico a continuación.

```
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 '|');
```

Expresión de gráfico

Cree una visualización de tabla en una hoja de Qlik Sense con **Customer**, **Product**, **UnitPrice** y **UnitSales** como dimensiones. Añada la siguiente expresión a la tabla como medida:

```
Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
```

Resultado

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

Explicación

Un conjunto de valores: 16, 16, 15, 15, 12, 12, 22, 22, 25, 15, 15 y 19. El cualificador **nodistinct** significa que el conjunto de valores contiene un elemento por cada fila de los datos fuente: cada una es el **UnitPrice** máximo por cada **Customer** y **Product**.

Ejemplo 3 de expresión de gráfico

Script de carga

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear el ejemplo de expresión de gráfico a continuación.

```
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;
```

Expresiones de gráfico

Cree una visualización de tabla en una hoja de Qlik Sense con **Year** y **Month** como dimensiones. Añada las siguientes expresiones a la tabla como medidas:

- `Sum(Sales)`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))` etiquetada como Aggr() estructurada en la tabla.

Resultado

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

Explicación

Este ejemplo muestra los valores agregados durante un período de doce meses para cada año en orden cronológico ascendente, por lo tanto, los parámetros estructurados (numéricos, ascendentes) forman parte de la expresión **Aggr()**. Se requieren dos dimensiones específicas como parámetros estructurados: **Year** y **Month**, ordenadas como (1) **Year** (numérico) y (2) **Month** (numérico). Estas dos dimensiones deben usarse en la visualización de tablas o gráficos. Esto es necesario para que la lista de dimensiones de la función **Aggr()** se corresponda con las dimensiones del objeto utilizado en la visualización.

Puede comparar la diferencia entre estas medidas en una tabla o en gráficos de líneas aparte:

- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year), (Month) ))`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))`

Debe quedar claro que solo la última expresión realiza la acumulación deseada de valores agregados.

Vea también:

Funciones básicas de agregación (page 333)

8.4 Funciones de color

Estas funciones se pueden utilizar en expresiones asociadas con la configuración y evaluación de las propiedades de color de los objetos gráficos, así como también en los scripts de carga de datos.

*Qlik Sense admite las funciones de color **Color()**, **qliktechblue** y **qliktechgray**para ofrecer una buena compatibilidad con versiones anteriores; sin embargo, no se recomienda su uso.*

ARGB

ARGB() se utiliza en expresiones para fijar o evaluar las propiedades de color de un objeto gráfico, donde el color viene definido por un componente de rojo **r**, un componente de verde **g** y un componente de azul **b**, con un factor alfa (opacidad) de **alpha**.

```
ARGB (alpha, r, g, b)
```

HSL

HSL() se utiliza en las expresiones para fijar o evaluar las propiedades de color de un objeto gráfico, donde el color se define mediante los valores de **hue**, **saturation** y **luminosity** entre 0 y 1.

```
HSL  (hue, saturation, luminosity)
```

RGB

RGB() devuelve un número entero correspondiente al código de color del color definido por los tres parámetros: el componente de rojo r, el componente de verde g y el componente de azul b. Estos componentes deben tener valores enteros entre 0 y 255. La función se puede utilizar en expresiones para establecer o evaluar las propiedades de color de un objeto de gráfico.

```
RGB  (r, g, b)
```

Colormix1

Colormix1() se usa en las expresiones para devolver una representación de color ARGB de un degradado de color, basándose en un valor entre 0 y 1.

```
Colormix1 (Value , ColorZero , ColorOne)
```

Value es un número real ente 0 y 1.

- Si Value = 0 devuelve ColorZero .
- Si Value = 1 devuelve ColorOne .
- Si 0 < Value< 1 devuelve el sombreado intermedio apropiado.

ColorZero es una representación válida de color RGB para que el color se asocie con el extremo inferior del intervalo.

ColorOne es una representación válida de color RGB para que el color se asocie con el extremo superior del intervalo.

Ejemplo:

```
Colormix1(0.5, red(), blue())
```
devuelve:

```
ARGB(255,64,0,64) (purple)
```

Colormix2
Colormix2() se usa en las expresiones para devolver una representación de color ARGB a partir de un degradado de dos colores, basado en un valor entre -1 y 1, con la posibilidad de especificar un color intermedio para la posición central (0).

```
Colormix2 (Value ,ColorMinusOne , ColorOne[ , ColorZero])
```
Value es un número real ente -1 y 1.

- Si Value = -1 devuelve el primer color.
- Si Value = -1 devuelve el segundo color.
- Si -1 < Value< 1 devuelve la mezcla de color adecuada.

ColorMinusOne es una representación válida de color RGB para que el color se asocie con el extremo inferior del intervalo.

ColorOne es una representación válida de color RGB para que el color se asocie con el extremo superior del intervalo.

ColorZero es una representación opcional válida de color RGB para que el color se asocie con el centro del intervalo.

SysColor
SysColor() devuelve la representación de color ARGB para el sistema de color de Windows nr, donde nr corresponde al parámetro a la función API **GetSysColor(nr)** de Windows.

```
SysColor (nr)
```

ColorMapHue
ColorMapHue() devuelve un valor ARGB de un color de un mapa de colores que varía el componente o matiz del modelo de color HSV. El mapa de color empieza con el color rojo, pasa por el amarillo, el verde, el cyan, el azul, el magenta y vuelve al rojo. x debe especificarse como un valor entre 0 y 1.

```
ColorMapHue (x)
```

ColorMapJet

ColorMapJet() devuelve un valor ARGB de un color de un mapa de colores que comienza con azul, pasa por cian, amarillo y naranja, y vuelve a rojo. x debe especificarse como un valor entre 0 y 1.

```
ColorMapJet (x)
```

Funciones de colores predefinidos

Las funciones siguientes pueden utilizarse en expresiones para colores predefinidos. Cada función devuelve una representación de color RGB.

Opcionalmente se puede dar un parámetro para el factor alfa, en cuyo caso devuelve una representación de color ARGB. El factor alfa 0 corresponde con una transparencia total, mientras que el factor alfa de 255 corresponde a una opacidad total. Si no se introduce un valor para alpha, se presupone 255.

Funciones de colores predefinidos

Función de color	**valor RGB**
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)

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	**Resultados**
`Blue()`	RGB(0,0,128)
`Blue(128)`	ARGB(128,0,0,128)

ARGB

ARGB() se utiliza en expresiones para fijar o evaluar las propiedades de color de un objeto gráfico, donde el color viene definido por un componente de rojo **r**, un componente de verde **g** y un componente de azul **b**, con un factor alfa (opacidad) de **alpha**.

Sintaxis:

```
ARGB(alpha, r, g, b)
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
alpha	Valor de transparencia en el rango 0 - 255. 0 corresponde a la transparencia total y 255 corresponde a la opacidad total.
r, g, b	Los valores de rojo, verde y azul. Un valor de 0 corresponde a ninguna aportación y un valor de 255 corresponde a una aportación total.

Todos los argumentos deben ser expresiones que devuelven enteros en el rango comprendido entre 0 y 255.

Si se va a interpretar el componente numérico y se le va a asignar el formato en notación hexadecimal, los valores de los componentes de color serán más fáciles de ver. Por ejemplo, verde claro tiene el número 4 278 255 360, que en notación hexadecimal es FF00FF00. Las dos primeras posiciones "FF" (255) denotan el canal **alpha**. Las dos siguientes posiciones ‘00’ denotan la cantidad de **rojo**, las dos siguientes posiciones ‘FF’ denotan la cantidad de **verde** y las dos posiciones finales ‘00’ denotan la cantidad de **azul**.

RGB

RGB() devuelve un número entero correspondiente al código de color del color definido por los tres parámetros: el componente de rojo r, el componente de verde g y el componente de azul b. Estos componentes deben tener valores enteros entre 0 y 255. La función se puede utilizar en expresiones para establecer o evaluar las propiedades de color de un objeto de gráfico.

Sintaxis:

```
RGB (r, g, b)
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
r, g, b	Los valores de rojo, verde y azul. Un valor de 0 corresponde a ninguna aportación y un valor de 255 corresponde a una aportación total.

Todos los argumentos deben ser expresiones que devuelven enteros en el rango comprendido entre 0 y 255.

Si se va a interpretar el componente numérico y se le va a asignar el formato en notación hexadecimal, los valores de los componentes de color serán más fáciles de ver. Por ejemplo, verde claro tiene el número 4 278 255 360, que en notación hexadecimal es FF00FF00. Las dos primeras posiciones "FF" (255) denotan el canal **alpha**. En las funciones **RGB** y **HSL**, esto es siempre 'FF' (opaco). Las dos siguientes posiciones '00' denotan la cantidad de **rojo**, las dos siguientes posiciones 'FF' denotan la cantidad de **verde** y las dos posiciones finales '00' denotan la cantidad de **azul**.

Ejemplo: Expresión de gráfico
Este ejemplo aplica un color personalizado a un gráfico.

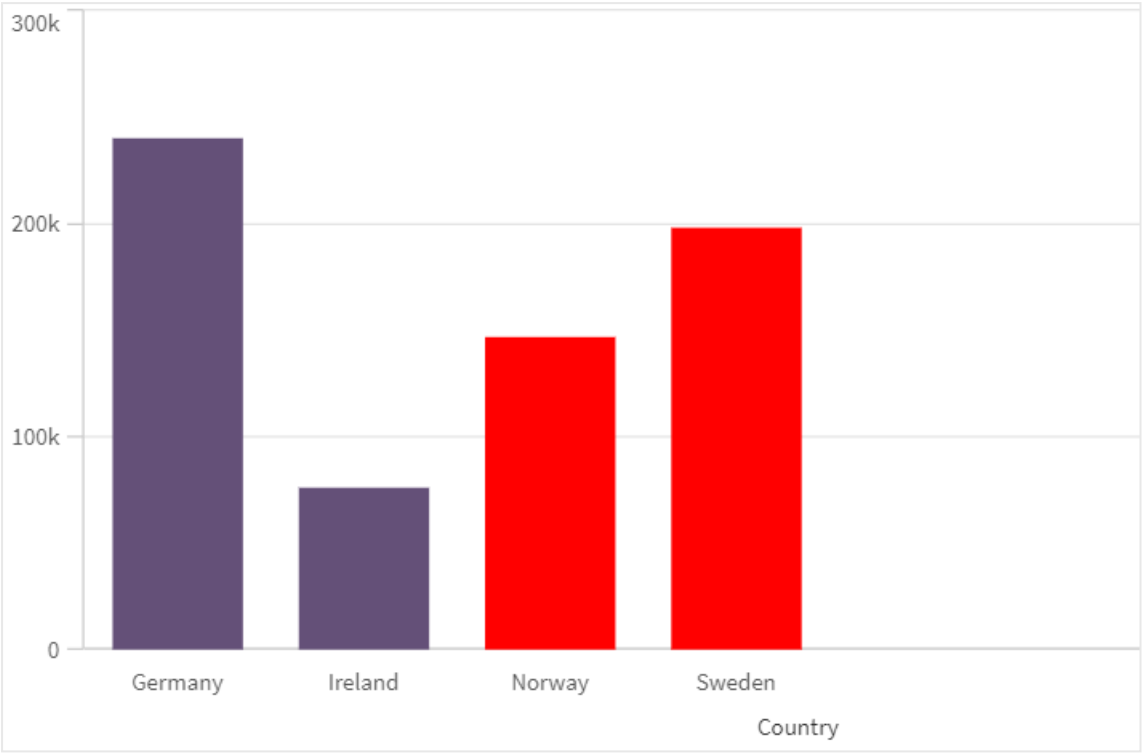
Datos utilizados en este ejemplo:

```
ProductSales:
Load * Inline
[Country,Sales,Budget
Sweden,100000,50000
Germany, 125000, 175000
Norway, 74850, 68500
Ireland, 45000, 48000
Sweden,98000,50000
Germany, 115000, 175000
Norway, 71850, 68500
Ireland, 31000, 48000
] (delimiter is ',');
```

Inserte la siguiente expresión en el panel de propiedades **Colores y leyenda**:

```
If (Sum(Sales)>Sum(Budget),RGB(255,0,0),RGB(100,80,120))
```

Resultado:



Ejemplo: Script de carga

El ejemplo a continuación muestra los valores RGB equivalentes a valores en formato hexadecimal.

```
Load
Text(R & G & B) as Text,
RGB(R,G,B)      as Color;
Load
Num#(R,'(HEX)') as R,
Num#(G,'(HEX)') as G,
Num#(B,'(HEX)') as B
Inline
[R,G,B
01,02,03
AA,BB,CC];
```

Resultado:

Texto	Color
010203	RGB(1,2,3)
AABBCC	RGB(170,187,204)

HSL

HSL() se utiliza en las expresiones para fijar o evaluar las propiedades de color de un objeto gráfico, donde el color se define mediante los valores de **hue**, **saturation** y **luminosity** entre 0 y 1.

Sintaxis:

```
HSL (hue, saturation, luminosity)
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
hue, saturation, luminosity	Valores de los componentes hue, saturation y luminosity que oscilan entre 0 y 1.

Todos los argumentos deben ser expresiones que devuelvan enteros en el rango comprendido entre 0 y 1.

Si se va a interpretar el componente numérico y asignarle formato en notación hexadecimal, los valores RGB de los componentes de color serán más fáciles de ver. Por ejemplo, el verde claro tiene el número 4 278 255 360, que en notación hexadecimal es FF00FF00 y RGB (0,255,0). Esto es equivalente a HSL (80/240, 240/240, 120/240) - un valor HSL de (0.33, 1, 0.5).

8.5 Funciones condicionales

Las funciones condicionales evalúan todas una condición y a continuación devuelven distintas respuestas dependiendo del valor de la condición. Las funciones pueden utilizarse en el script de carga de datos y en expresiones de gráficos.

Descripción general de las funciones condicionales

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

alt

La función **alt** devuelve el primero de los parámetros que tiene una representación numérica válida. Si no se encuentra tal condición, devolverá el último parámetro. Se puede utilizar cualquier número de parámetros.

```
alt (expr1[ , expr2 , expr3 , ...] , else)
```

class

La función **class** asigna el primer parámetro a un intervalo de clases. El resultado es un valor dual con a<=x<b como el valor textual, donde a y b son los límites superior e inferior del contenedor, y el límite inferior como valor numérico.

```
class (expression, interval [ , label [ , offset ]])
```

coalesce

La función **coalesce** devuelve el primero de los parámetros que tiene una representación válida non-NULL. Se puede utilizar cualquier número de parámetros.

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

if

La función **if** devuelve un valor dependiendo de si la condición proporcionada con la función se evalúa como True o False.

```
if (condition , then , else)
```

match

La función **match** compara el primer parámetro con todos los siguientes y devuelve la ubicación numérica del número de expresiones que coinciden. La comparación es sensible a mayúsculas.

```
match ( str, expr1 [ , expr2,...exprN ])
```

mixmatch

La función **mixmatch** compara el primer parámetro con todos los siguientes y devuelve la ubicación numérica de las expresiones que coinciden. La comparación no es sensible a mayúsculas.

```
mixmatch ( str, expr1 [ , expr2,...exprN ])
```

pick

La función de selección devuelve la enésima expresión *n* en la lista.

```
pick (n, expr1[ , expr2,...exprN])
```

wildmatch

La función **wildmatch** compara el primer parámetro con todos los siguientes y devuelve el número de la expresión que coincida. Permite el uso de los caracteres comodín (***** y **?**) en las cadenas de comparación. ***** es cualquier secuencia de caracteres. **?** es un único carácter. La comparación no es sensible a mayúsculas.

```
wildmatch ( str, expr1 [ , expr2,...exprN ])
```

alt

La función **alt** devuelve el primero de los parámetros que tiene una representación numérica válida. Si no se encuentra tal condición, devolverá el último parámetro. Se puede utilizar cualquier número de parámetros.

Sintaxis:

```
alt(expr1[ , expr2 , expr3 , ...] , else)
```

Argumentos:

Argumentos

Argumento	Descripción
expr1	La primera expresión para comprobar si hay una representación numérica válida.
expr2	La segunda expresión para comprobar si hay una representación numérica válida.
expr3	La tercera expresión para comprobar si hay una representación numérica válida.

Argumento	Descripción
else	Valor que devolver si ninguno de los parámetros anteriores tiene una representación numérica válida.

La función alt se utiliza a menudo con funciones de interpretación de números o fechas. De esta manera, Qlik Sense puede probar distintos formatos de fecha en un orden priorizado. También se puede utilizar para manejar valores NULL en expresiones numéricas.

Ejemplos:

Ejemplos

Ejemplo	Resultado
`alt( date#( dat , 'YYYY/MM/DD' ), date#( dat , 'MM/DD/YYYY' ), date#( dat , 'MM/DD/YY' ), 'No valid date' )`	Esta expresión comprobará si el campo date contiene una fecha conforme a cualquiera de los tres formatos de fecha especificados. Si es así, devolverá un valor dual con la cadena original y una representación numérica válida de una fecha. Si no se encuentra correspondencia alguna, devolverá el texto 'No valid date' (sin ninguna representación numérica válida).
`alt(Sales,0) + alt(Margin,0)`	Esta expresión añade los campos Sales y Margin, reemplazando cualquier valor perdido (NULL) por un 0.

class

La función **class** asigna el primer parámetro a un intervalo de clases. El resultado es un valor dual con a<=x<b como el valor textual, donde a y b son los límites superior e inferior del contenedor, y el límite inferior como valor numérico.

Sintaxis:

```
class(expression, interval [ , label [ , offset ]])
```

Argumentos:

Argumentos

Argumento	Descripción
interval	Un número que especifica el ancho del contenedor.
label	Una cadena arbitraria que puede reemplazar a la "x" en el texto del resultado.
offset	Un número que puede utilizarse como desplazamiento desde el punto de partida por defecto de la clasificación. El punto de inicio predeterminado normalmente es 0.

Ejemplos:

Ejemplos

Ejemplo	Resultado
`class( var,10 )` con `var = 23`	devuelve '20<=x<30'
`class( var,5,'value' )` con `var = 23`	devuelve '20<= value <25'
`class( var,10,'x',5 )` con `var = 23`	devuelve '15<=x<25'

Ejemplo: cargar script usando class

Ejemplo: script de carga

Script de carga

En este ejemplo cargamos una tabla que contiene el nombre y la edad de las personas. Queremos añadir un campo que clasifique las personas según un grupo de edad con un intervalo de diez años. La tabla fuente original presenta el siguiente aspecto.

Resultados

Name	Age
John	25
Karen	42
Yoshi	53

Para agregar el campo de clasificación por grupo de edad, puede agregar una sentencia load precedente usando la función **class**.

Cree una nueva pestaña en el editor de carga de datos y luego cargue los siguientes datos como una carga inline. Cree la tabla siguiente en Qlik Sense para ver los resultados.

```
LOAD *,
class(Age, 10, 'age') As Agegroup;

LOAD * INLINE
[ Age, Name
25, John
42, Karen
53, Yoshi];
```

Resultados

Resultados

Name	Age	Agegroup
John	25	20 <= age < 30
Karen	42	40 <= age < 50
Yoshi	53	50 <= age < 60

coalesce

La función **coalesce** devuelve el primero de los parámetros que tiene una representación válida non-NULL. Se puede utilizar cualquier número de parámetros.

Sintaxis:

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

Argumentos:

Argumentos

Argumento	Descripción
expr1	La primera expresión para comprobar si hay una representación no nula válida.
expr2	La segunda expresión para comprobar si hay una representación no nula válida.
expr3	La tercera expresión para comprobar si hay una representación no nula válida.

Ejemplos:

Ejemplos

Ejemplo	Resultado
	Esta expresión cambia todos los valores NULL de un campo a 'N/A'.
`Coalesce(ProductDescription, ProductName, ProductCode, 'no description available')`	Esta expresión seleccionará entre tres campos de descripción de producto diferentes, para cuando algunos campos pueden no tener valores para el producto. Se devolverá el primero de los campos, en el orden indicado, con un valor no nulo. Si ninguno de los campos contiene un valor, el resultado será "no hay descripción disponible".

Ejemplo	Resultado
`Coalesce(TextBetween(FileName, '"', '"'), FileName)`	Esta expresión recortará las posibles citas adjuntas del campo *FileName*. Si se cita el *FileName* proporcionado, estas se eliminan y se devuelve el *FileName* adjunto sin comillas. Si la función *TextBetween* no encuentra los delimitadores, devuelve un valor nulo, que **Coalesce** rechaza, devolviendo en su lugar el *FileName* sin formato.

if

La función **if** devuelve un valor dependiendo de si la condición proporcionada con la función se evalúa como True o False.

Sintaxis:

```
if(condition , then [, else])
```

Argumentos

Argumento	Descripción
condition	La expresión que se interpreta de una manera lógica.
then	La expresión que puede ser de cualquier tipo. Si la *condition* es True, entonces la función if devuelve el valor de la expresión *then*.
else	La expresión que puede ser de cualquier tipo. Si la *condition* es False, entonces la función if devuelve el valor de la expresión *else*. Este parámetro es opcional. Si la *condition* es False, devuelve NULL si no ha especificado else.

Ejemplo

Ejemplo	Resultado
`if( Amount>= 0, 'OK', 'Alarm' )`	Esta expresión comprueba si la cantidad es un número positivo (0 o mayor) y devuelve 'OK' si lo es. Si la cantidad es inferior a 0, devuelve 'Alarm'.

Ejemplo: cargar script usando if

Ejemplo: Script de carga

Script de carga

If se puede utilizar en el script de carga con otros métodos y objetos, incluidas las variables. Por ejemplo, si define una variable umbral *threshold* y desea incluir un campo en el modelo de datos basado en ese umbral, puede hacer lo siguiente:

Cree una nueva pestaña en el editor de carga de datos y luego cargue los siguientes datos como una carga inline. Cree la tabla siguiente en Qlik Sense para ver los resultados.

```
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;
```

Resultados

Tabla de Qlik Sense que muestra el resultado de usar la función *if* en el script de carga.

id_transacción	Comparado con umbral
3750	Menor que 100
3751	Mayor que 100
3752	Menor que 100
3753	Mayor que 100
3754	Mayor que 100
3756	Menor que 100
3757	Mayor que 100

Ejemplos: expresiones de gráfico usando if

Ejemplos: Expresiones de gráfico

Expresión de gráfico 1

Script de carga

Cree una nueva pestaña en el editor de carga de datos y luego cargue los siguientes datos como una carga inline. Después de cargar los datos, cree los ejemplos de la expresión de gráfico a continuación en una tabla de Qlik Sense.

```
MyTable:
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];
```

Tabla de Qlik Sense con ejemplos de la función *if* en una expresión de gráfico.

Fecha	**Localización**	**Incidentes**	**if(Incidentes>=10, 'Critical', 'Ok')**	**if(Incidentes>=10, 'Critical', If(Incidentes>=1 and Incidentes<10, 'Advertencia', 'Ok'))**
1/3/2016	Pekín	0	Ok	Ok
1/3/2016	Boston	12	Critical	Critical
1/3/2016	Estocolmo	3	Ok	Advertencia
1/3/2016	Toronto	0	Ok	Ok
1/4/2016	Pekín	0	Ok	Ok
1/4/2016	Boston	8	Ok	Advertencia

Expresión de gráfico 2

En una nueva app, agregue el siguiente script en una nueva pestaña en el editor de carga de datos y luego cargue los datos. Después, puede crear la tabla con las expresiones de gráfico a continuación.

```
SET FirstWeekDay=0;
Load
Date(MakeDate(2022)+RecNo()-1) as Date
Autogenerate 14;
```

Tabla Qlik Sense que muestra un ejemplo de la función *if* en una expresión de gráfico.

Fecha	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/1/2022	Sáb	WeekEnd
1/2/2022	Dom	WeekEnd
1/3/2022	Lun	Normal Day
1/4/2022	Mar	Normal Day
1/5/2022	Mié	Normal Day
1/6/2022	Jue	Normal Day
1/7/2022	Vie	Normal Day
1/8/2022	Sáb	WeekEnd
1/9/2022	Dom	WeekEnd
1/10/2022	Lun	Normal Day
1/11/2022	Mar	Normal Day
1/12/2022	Mié	Normal Day
1/13/2022	Jue	Normal Day
1/14/2022	Vie	Normal Day

match

La función **match** compara el primer parámetro con todos los siguientes y devuelve la ubicación numérica del número de expresiones que coinciden. La comparación es sensible a mayúsculas.

Sintaxis:

```
match( str, expr1 [ , expr2,...exprN ])
```

*Si desea utilizar una comparación que no distinga entre mayúsculas y minúsculas, utilice la función **mixmatch**. Si desea utilizar una comparación que no distinga entre mayúsculas y minúsculas además de comodines, utilice la función **wildmatch**.*

Ejemplo: Ejemplo: cargar script usando match

Ejemplo: Script de carga

Script de carga

Puede usar match para cargar un subconjunto de datos. Por ejemplo, puede devolver un valor numérico para una expresión en la función. A continuación puede limitar los datos cargados en función del valor numérico. Match devuelve 0 si no hay coincidencia. Todas las expresiones que no coinciden en este ejemplo devolverán 0 y serán excluidas de la carga de datos por la sentencia WHERE.

Cree una nueva pestaña en el editor de carga de datos y luego cargue los siguientes datos como una carga inline. Cree la tabla siguiente en Qlik Sense para ver los resultados.

```
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 statment 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;
```

Resultados

Tabla de Qlik Sense que muestra el resultado de usar la función match en el script de carga

Código de color Blue and Black	Cliente
Black	203521
Black	3036491
Blue	2038593

Ejemplos: expresiones de gráfico usando match

Ejemplos: Expresiones de gráfico

Expresión de gráfico 1

Script de carga

Cree una nueva pestaña en el editor de carga de datos y luego cargue los siguientes datos como una carga inline. Después de cargar los datos, cree los ejemplos de la expresión de gráfico a continuación en una tabla de Qlik Sense.

```
MyTable:
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];
```

La primera expresión en la tabla a continuación devuelve 0 para Estocolmo porque 'Estocolmo' no está incluido en la lista de expresiones de la función **match**. También devuelve 0 para 'Zurich' porque la comparación **match** es sensible a mayúsculas.

Tabla de Qlik Sense con ejemplos de la función *match* en una expresión de gráfico

Cities	**match(Cities,'Toronto','Boston','Beijing','Zurich')**	**match(Ciudades,'Toronto','Boston','Pekín','Estocolmo', 'zurich')**
Beijing	3	3
Boston	2	2
Stockholm	0	4
Toronto	1	1
zurich	0	5

Expresión de gráfico 2

Puede usar match para realizar una ordenación personalizada para una expresión.

De forma predeterminada, las columnas se ordenan numéricamente o alfabéticamente, dependiendo de los datos.

Tabla de Qlik Sense que muestra un ejemplo del orden predeterminado

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Para cambiar el orden, haga lo siguiente:

1. Abra la sección **Ordenar** de su gráfico en el panel de **Propiedades**.
2. Desactive la ordenación automática para la columna en la que desea realizar una ordenación personalizada.
3. Desmarque **Ordenar numéricamente** y **Ordenar alfabéticamente**.
4. Seleccione **Ordenar por expresión** y después introduzca una expresión similar a la siguiente:
   `=match( Cities, 'Toronto','Boston','Beijing','Stockholm','zurich')`
   El criterio de ordenación de la columna Cities cambia.

Tabla de Qlik Sense que muestra un ejemplo de un cambio de ordenación usando la función *match*

Cities
Toronto

Cities
Boston
Beijing
Stockholm
zurich

También puede ver el valor numérico que devuelve.

Tabla de Qlik Sense que muestra un ejemplo de los valores numéricos que devuelve la función *match*

Ciudades	Cities & ' - ' & match (Cities, 'Toronto','Boston', 'Beijing','Stockholm','zurich')
Toronto	Toronto - 1
Boston	Boston - 2
Pekín	Beijing - 3
Estocolmo	Stockholm - 4
zurich	zurich - 5

mixmatch

La función **mixmatch** compara el primer parámetro con todos los siguientes y devuelve la ubicación numérica de las expresiones que coinciden. La comparación no es sensible a mayúsculas.

Sintaxis:

```
mixmatch( str, expr1 [ , expr2,...exprN ])
```

Si en cambio desea utilizar una comparación que distinga entre mayúsculas y minúsculas, utilice la función **match**. Si desea utilizar una comparación que no distinga entre mayúsculas y minúsculas además de comodines, utilice la función **wildmatch**.

Ejemplo: cargar script usando mixmatch

Ejemplo: Script de carga

Script de carga

Puede usar mixmatch para cargar un subconjunto de datos. Por ejemplo, puede devolver un valor numérico para una expresión en la función. A continuación puede limitar los datos cargados en función del valor numérico. Mixmatch devuelve 0 si no hay coincidencia. Todas las expresiones que no coinciden en este ejemplo devolverán 0 y serán excluidas de la carga de datos por la instrucción WHERE.

Cree una nueva pestaña en el editor de carga de datos y luego cargue los siguientes datos como una carga inline. Cree la tabla siguiente en Qlik Sense para ver los resultados.

```
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;
```

Resultados

Tabla de Qlik Sense que muestra el resultado de usar la función mixmatch en el script de carga.

Código de color Black, Blue, blue	Cliente
Black	203521
Black	3036491
Blue	2038593
blue	5646471

Ejemplos: expresiones de gráfico usando mixmatch

Ejemplos: Expresiones de gráfico

Cree una nueva pestaña en el editor de carga de datos y luego cargue los siguientes datos como una carga inline. Después de cargar los datos, cree los ejemplos de la expresión de gráfico a continuación en una tabla de Qlik Sense.

Expresión de gráfico 1

```
MyTable: 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];
```

La primera expresión en la tabla a continuación devuelve 0 para Estocolmo porque 'Estocolmo' no está incluido en la lista de expresiones de la función **mixmatch**. Devuelve 4 para 'Zurich' porque la comparación **mixmatch** no es sensible a mayúsculas.

Tabla de Qlik Sense que muestra ejemplos de la función *mixmatch* en una expresión de gráfico

Cities	mixmatch(Cities,'Toronto','Boston','Beijing','Zurich')	mixmatch(Ciudades,'Toronto','Boston','Pekín','Estocolmo', 'Zurich')
Beijing	3	3
Boston	2	2

Cities	mixmatch(Cities,'Toronto','Boston','Beijing','Zurich')	mixmatch(Ciudades,'Toronto','Boston','Pekín','Estocolmo', 'Zurich')
Stockholm	0	4
Toronto	1	1
zurich	4	5

Expresión de gráfico 2

Puede usar mixmatch para realizar una ordenación personalizada para una expresión.

De forma predeterminada, las columnas se ordenan alfabética o numéricamente, dependiendo de los datos.

Tabla de Qlik Sense que muestra un ejemplo del orden predeterminado

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Para cambiar el orden, haga lo siguiente:

1. Abra la sección **Ordenar** de su gráfico en el panel de **Propiedades**.
2. Desactive la ordenación automática para la columna en la que desea realizar una ordenación personalizada.
3. Desmarque **Ordenar numéricamente** y **Ordenar alfabéticamente**.
4. Seleccione **Ordenar por expresión** y después introduzca la siguiente expresión:
   `=mixmatch( Cities, 'Toronto','Boston','Beijing','Stockholm','Zurich')`
   El criterio de ordenación de la columna Cities cambia.

Tabla de Qlik Sense que muestra un ejemplo de un cambio de ordenación usando la función *mixmatch*.

Cities
Toronto
Boston
Beijing
Stockholm
zurich

También puede ver el valor numérico que devuelve.

Tabla de Qlik Sense que muestra un ejemplo de los valores numéricos que devuelve la función *mixmatch*.

Ciudades	**Cities & ' - ' & mixmatch (Cities, 'Toronto','Boston', 'Beijing','Stockholm','Zurich')**
Toronto	Toronto - 1
Boston	Boston - 2
Pekín	Beijing - 3
Estocolmo	Stockholm - 4
zurich	zurich - 5

pick

La función de selección devuelve la enésima expresión *n* en la lista.

Sintaxis:

```
pick(n, expr1[ , expr2,...exprN])
```

Argumentos:

Argumentos

Argumento	**Descripción**
n	*n* es un entero entre 1 y N.

Ejemplo:

Ejemplo

Ejemplo	**Resultado**
`pick( N, 'A','B',4, 6 )`	devuelve 'B' si N = 2 devuelve 4 si N = 3

wildmatch

La función **wildmatch** compara el primer parámetro con todos los siguientes y devuelve el número de la expresión que coincida. Permite el uso de los caracteres comodín (***** y **?**) en las cadenas de comparación. ***** es cualquier secuencia de caracteres. **?** es un único carácter. La comparación no es sensible a mayúsculas.

Sintaxis:

```
wildmatch( str, expr1 [ , expr2,...exprN ])
```

Si desea utilizar la comparación sin comodines, utilice las funciones **match** o **mixmatch**.

Ejemplo: Ejemplo: cargar script usando wildmatch

Ejemplo: Script de carga

Script de carga

Puede usar wildmatch para cargar un subconjunto de datos. Por ejemplo, puede devolver un valor numérico para una expresión en la función. A continuación puede limitar los datos cargados en función del valor numérico. Wildmatch devuelve 0 si no hay coincidencia. Todas las expresiones que no coinciden en este ejemplo devolverán 0 y serán excluidas de la carga de datos por la instrucción WHERE.

Cree una nueva pestaña en el editor de carga de datos y luego cargue los siguientes datos como una carga inline. Cree la tabla siguiente en Qlik Sense para ver los resultados.

```
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;
```

Resultados

Tabla de Qlik Sense que muestra el resultado de usar la función *wildmatch* en el script de carga

Código de color Black, Blue, blue, Red	Cliente
Black	203521
Black	3036491
Blue	2038593
blue	5646471
Red	049681
Red	2038593

Ejemplos: Expresiones de gráfico que usan wildmatch

Ejemplo: Expresión de gráfico

Expresión de gráfico 1

Cree una nueva pestaña en el editor de carga de datos y luego cargue los siguientes datos como una carga inline. Después de cargar los datos, cree los ejemplos de la expresión de gráfico a continuación en una tabla de Qlik Sense.

```
MyTable: 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];
```

La primera expresión en la tabla a continuación devuelve 0 para Estocolmo porque 'Estocolmo' no está incluido en la lista de expresiones de la función **wildmatch**. También devuelve 0 para 'Boston' porque ? solo coincide con un único carácter.

Tabla de Qlik Sense que muestra ejemplos de la función *wildmatch* en una expresión de gráfico

Cities	**wildmatch(Cities,'Tor*','?ton','Beijing','*urich')**	**wildmatch(Ciudades,'Tor*','???ton','Pekín','Estocolmo','*urich')**
Beijing	3	3
Boston	0	2
Stockholm	0	4
Toronto	1	1
zurich	4	5

Expresión de gráfico 2

Puede usar wildmatch para realizar una ordenación personalizada para una expresión.

De forma predeterminada, las columnas se ordenan numéricamente o alfabéticamente, dependiendo de los datos.

Tabla de Qlik Sense que muestra un ejemplo del orden predeterminado

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Para cambiar el orden, haga lo siguiente:

1. Abra la sección **Ordenar** de su gráfico en el panel de **Propiedades**.
2. Desactive la ordenación automática para la columna en la que desea realizar una ordenación personalizada.
3. Desmarque **Ordenar numéricamente** y **Ordenar alfabéticamente**.
4. Seleccione **Ordenar por expresión** y después introduzca una expresión similar a la siguiente:
   `=wildmatch( Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')`
   El criterio de ordenación de la columna Cities cambia.

Tabla de Qlik Sense que muestra un ejemplo de un cambio de ordenación usando la función *wildmatch*.

Cities
Toronto
Boston
Beijing
Stockholm
zurich

También puede ver el valor numérico que devuelve.

Tabla de Qlik Sense que muestra un ejemplo de los valores numéricos que devuelve la función *wildmatch*

Ciudades	**Cities & ' - ' & wildmatch (Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')**
Toronto	Toronto - 1
Boston	Boston - 2
Pekín	Beijing - 3
Estocolmo	Stockholm - 4
zurich	zurich - 5

8.6 Funciones de contador

Esta sección describe funciones relacionadas con los contadores de registros durante la evaluación de la sentencia **LOAD** en el script de carga de datos. La única función que puede utilizarse en las expresiones de gráficos es **RowNo()**.

Algunas funciones de contador no tienen ningún parámetro, pero los paréntesis finales siempre son necesarios no obstante.

Descripción general de las funciones de contador

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

autonumber
Esta función de script devuelve un valor entero único por cada valor distinto evaluado de *expression* que encuentre durante la ejecución del script. Esta función se puede utilizar por ej. para crear un representación compacta de memoria de una clave compleja.

```
autonumber (expression[ , AutoID])
```

autonumberhash128
Esta función de script calcula un hash de 128 bits de los valores de entrada combinados de la expresión y devuelve un único entero por cada valor hash distinto encontrado durante la ejecución de script. Esta función se puede utilizar por ejemplo para crear un representación compacta de memoria de una clave compleja.

```
autonumberhash128 (expression {, expression})
```

autonumberhash256
Esta función de script calcula un hash de 256 bits de los valores de entrada combinados de la expresión y devuelve un único entero por cada valor de resumen diferente encontrado durante la ejecución de script. Esta función se puede utilizar por ej. para crear un representación compacta de memoria de una clave compleja.

```
autonumberhash256 (expression {, expression})
```

IterNo
Esta función de script devuelve un número entero que indica la iteración o el número de vez en que se está evaluando un único registro en una sentencia **LOAD** con una cláusula **while**. La primera iteración tiene el número 1. La función **IterNo** solo tiene sentido si se utiliza con una cláusula **while**.

```
IterNo ( )
```

RecNo
Esta función de script devuelve un entero con el número de la fila actual de un tabla interna. El primer registro es el número 1.

```
RecNo ( )
```

RowNo - script function
Esta función devuelve un entero para indicar la posición de la fila actual en la tabla interna de Qlik Sense resultante. La primera fila es la número 1.

```
RowNo ( )
```

RowNo - chart function
RowNo() devuelve el número de la fila actual dentro del segmento de columna actual de una tabla. Para los gráficos de mapa de bits, **RowNo()** devuelve el número de la fila actual dentro del equivalente de la tabla simple del gráfico.

```
RowNo - función de gráfico([TOTAL])
```

autonumber

Esta función de script devuelve un valor entero único por cada valor distinto evaluado de *expression* que encuentre durante la ejecución del script. Esta función se puede utilizar por ej. para crear un representación compacta de memoria de una clave compleja.

*Solo puede conectar las claves de **autonumber** que se hayan generado en la misma carga de datos, ya que el entero se genera de acuerdo con el orden en que se lee la tabla. Si necesita usar claves que sean persistentes entre las cargas de datos, independientemente de la clasificación de datos de origen, debe usar el **hash128**, **hash160** o las funciones **hash256**.*

Sintaxis:

```
autonumber(expression[ , AutoID])
```

Argumentos:

Argumento	Descripción
AutoID	Para crear múltiples instancias de contador si la función **autonumber** se usa en diferentes claves dentro del script, el parámetro opcional *AutoID* se puede usar para nombrar cada contador.

Ejemplo: Crear una clave compuesta

En este ejemplo, creamos una clave compuesta usando la función **autonumber** para conservar la memoria. El ejemplo es breve y con fines de demostración, pero tendría todo el sentido en una tabla con un número elevado de filas.

Datos de ejemplo

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

Los datos fuente se cargan usando datos inline. Luego agregamos un load precedente que crea una clave compuesta desde los campos Region, Year y 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
];
```

La tabla resultante tiene el siguiente aspecto:

Tabla de resultados

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

En este ejemplo, puede hacer referencia a RYMkey, por ejemplo 1, en lugar de hacer referencia a la cadena “North2014May” si desea enlazar a otra tabla.

Ahora cargaremos una tabla de costes fuente de un modo similar. Los campos Region, Year y Month se excluyen de la carga precedente para evitar crear una clave sintética, ya estamos creando una clave compuesta con la función **autonumber**, vinculando las tablas.

```
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
];
```

Ahora podemos agregar una visualización de tabla a una hoja y agregar los campos Region, Year y Month, así como las medidas de suma para las ventas y los costes. La tabla presentará el siguiente aspecto:

Tabla de resultados

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56

Region	Year	Month	Sum([Sales])	Sum([Costs])
South	2014	June	645	64
South	2013	May	588	465

autonumberhash128

Esta función de script calcula un hash de 128 bits de los valores de entrada combinados de la expresión y devuelve un único entero por cada valor hash distinto encontrado durante la ejecución de script. Esta función se puede utilizar por ejemplo para crear un representación compacta de memoria de una clave compleja.

*Solo puede conectar las claves de **autonumberhash128** que se hayan generado en la misma carga de datos, ya que el entero se genera de acuerdo con el orden en que se lee la tabla. Si necesita usar claves que sean persistentes entre las cargas de datos, independientemente de la clasificación de datos de origen, debe usar el **hash128**, **hash160** o las funciones **hash256**.*

Sintaxis:

```
autonumberhash128(expression {, expression})
```

Ejemplo: Crear una clave compuesta

En este ejemplo, creamos una clave compuesta usando la función **autonumberhash128** para conservar la memoria. El ejemplo es breve y con fines de demostración, pero tendría todo el sentido en una tabla con un número elevado de filas.

Datos de ejemplo

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

Los datos fuente se cargan usando datos inline. Luego agregamos un load precedente que crea una clave compuesta desde los campos Region, Year y 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
];
```

La tabla resultante tiene el siguiente aspecto:

Tabla de resultados

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

En este ejemplo, puede hacer referencia a RYMkey, por ejemplo 1, en lugar de hacer referencia a la cadena “North2014May” si desea enlazar a otra tabla.

Ahora cargaremos una tabla de costes fuente de un modo similar. Los campos Region, Year y Month se excluyen de la carga precedente para evitar crear una clave sintética, ya estamos creando una clave compuesta con la función **autonumberhash128**, vinculando las tablas.

```
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
];
```

Ahora podemos agregar una visualización de tabla a una hoja y agregar los campos Region, Year y Month, así como las medidas de suma para las ventas y los costes. La tabla presentará el siguiente aspecto:

Tabla de resultados

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

Esta función de script calcula un hash de 256 bits de los valores de entrada combinados de la expresión y devuelve un único entero por cada valor de resumen diferente encontrado durante la ejecución de script. Esta función se puede utilizar por ej. para crear un representación compacta de memoria de una clave compleja.

*Solo puede conectar las claves de **autonumberhash256** que se hayan generado en la misma carga de datos, ya que el entero se genera de acuerdo con el orden en que se lee la tabla. Si necesita usar claves que sean persistentes entre las cargas de datos, independientemente de la clasificación de datos de origen, debe usar el **hash128**, **hash160** o las funciones **hash256**.*

Sintaxis:

```
autonumberhash256(expression {, expression})
```

Ejemplo: Crear una clave compuesta

En este ejemplo, creamos una clave compuesta usando la función **autonumberhash256** para conservar la memoria. El ejemplo es breve con fines de demostración, pero sería significativo en una tabla con un número elevado de filas.

Tabla de ejemplo

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

Los datos fuente se cargan usando datos inline. Luego agregamos un load precedente que crea una clave compuesta desde los campos Region, Year y 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
];
```

La tabla resultante tiene el siguiente aspecto:

Tabla de resultados

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

En este ejemplo, puede hacer referencia a RYMkey, por ejemplo 1, en lugar de hacer referencia a la cadena “North2014May” si desea enlazar a otra tabla.

Ahora cargaremos una tabla de costes fuente de un modo similar. Los campos Region, Year y Month se excluyen de la carga precedente para evitar crear una clave sintética, ya estamos creando una clave compuesta con la función **autonumberhash256**, vinculando las tablas.

```
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
];
```

Ahora podemos agregar una visualización de tabla a una hoja y agregar los campos Region, Year y Month, así como las medidas de suma para las ventas y los costes. La tabla presentará el siguiente aspecto:

Tabla de resultados

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

Esta función de script devuelve un número entero que indica la iteración o el número de vez en que se está evaluando un único registro en una sentencia **LOAD** con una cláusula **while**. La primera iteración tiene el número 1. La función **IterNo** solo tiene sentido si se utiliza con una cláusula **while**.

Sintaxis:

```
IterNo( )
```

Ejemplos y resultados:

Ejemplo:

```
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
];
```

Esta sentencia **LOAD** generará un registro por fecha dentro del rango definido por **StartDate** y **EndDate**.

En este ejemplo, la tabla resultante tendrá el siguiente aspecto:

Tabla de resultados

Day	Date
1	2014-01-22

Day	Date
2	2014-01-23
3	2014-01-24
4	2014-01-25
5	2014-01-26

RecNo

Esta función de script devuelve un entero con el número de la fila actual de un tabla interna. El primer registro es el número 1.

Sintaxis:

```
RecNo( )
```

A diferencia de **RowNo( )**, que cuenta las filas en la tabla Qlik Sense resultante, **RecNo( )** cuenta los registros en la tabla de datos sin procesar y se restablece cuando una tabla de datos sin procesar se concatena a otra.

Ejemplo: Script de carga de datos

Carga de tabla de datos sin procesar:

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Registro de carga y números de fila de las filas seleccionadas:

```
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;
```

La tabla Qlik Sense interna resultante:

Tabla de resultados

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo

Esta función devuelve un entero para indicar la posición de la fila actual en la tabla interna de Qlik Sense resultante. La primera fila es la número 1.

Sintaxis:

```
RowNo( [TOTAL])
```

A diferencia de **RecNo( )**, que cuenta los registros en la tabla de datos sin procesar, la función **RowNo( )** no cuenta los registros que están excluidos por las cláusulas **where** y no se restablece cuando una tabla de datos sin procesar se concatena a otra.

> *Si usa un load precedente, es decir, una serie de sentencias **LOAD** apiladas que se leen en la misma tabla, solo podrá usar **RowNo( )** en la sentencia **LOAD** superior. Si utiliza **RowNo( )** en sentencias **LOAD** subsiguientes, devuelve 0.*

Ejemplo: Script de carga de datos

Carga de tabla de datos sin procesar:

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Registro de carga y números de fila de las filas seleccionadas:

```
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;
```

La tabla Qlik Sense interna resultante:

Tabla de resultados

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo - función de gráfico

RowNo() devuelve el número de la fila actual dentro del segmento de columna actual de una tabla. Para los gráficos de mapa de bits, **RowNo()** devuelve el número de la fila actual dentro del equivalente de la tabla simple del gráfico.

Si la tabla o el equivalente de tabla tiene múltiples dimensiones verticales, el segmento de columna actual incluirá solo filas con los mismos valores que la fila actual en todas las columnas de dimensión, excepto para la columna que muestra la última dimensión en el orden de campos interno.

Segmentos de columna

	Region	Country	Population	Rank(Population)
Column segment #1	Americas	Mexico	128,932,753	2
	Americas	Canada	37,742,154	3
	Americas	United States of America	331,002,651	1
Column segment #2	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

No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función.

Sintaxis:

```
RowNo([TOTAL])
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumento	Descripción
TOTAL	Si la tabla es unidimensional o si el cualificador **TOTAL** se utiliza como argumento, el segmento de columna actual es siempre igual a la columna completa.

Ejemplo: Expresiones de gráfico que usan RowNo

Ejemplo: expresión de gráfico

Script de carga

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear los ejemplos de expresión del gráfico a continuación.

```
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 '|');
```

Expresión de gráfico

Cree una visualización de tabla en una hojade Qlik Sense con **Customer** y **UnitSales** como dimensiones. Añada `RowNo( )` y `RowNo(TOTAL)` como medidas y denomínelas **Row in segment** y **Row Number**, respectivamente. Añada la siguiente expresión a la tabla como medida.

```
If( RowNo( )=1, 0, UnitSales / Above( UnitSales ))
```

Resultado

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

Explicación

La columna **Row in Segment** muestra los resultados 1,2,3 para el segmento de columna que contiene los valores de UnitSales para el cliente Astrida. La numeración de filas comienza de nuevo en 1 para el siguiente segmento de columna, que es Betacab.

La columna **Row Number** ignora las dimensiones debido al argumento `TOTAL` de `RowNo()` y cuenta las filas de la tabla.

Esta expresión devuelve 0 para la primera fila de cada segmento de columna, por lo que la columna muestra:

0, 2.25, 1.1111111, 0, 2.5, 5, 0, 2, 0 y 4.

Vea también:

Above - función de gráfico (page 1293)

8.7 Funciones de fecha y hora

Las funciones de fecha y hora de Qlik Sense se usan para transformar y convertir valores de fecha y hora. Todas las funciones pueden utilizarse tanto en el script de carga de datos como en las expresiones de gráficos.

Las funciones están basadas en un número de serie de fecha-hora que es igual al número de días transcurridos desde el 30 de diciembre de 1899. El valor entero representa el día, y el valor fraccional representa la hora del día.

Qlik Sense utiliza el valor numérico del parámetro, por tanto, un número también es válido como parámetro cuando no tiene formato de fecha u hora. Si el parámetro no se corresponde con el valor numérico, p. ej. si fuera una cadena, entonces Qlik Sense trata de interpretar la cadena conforme a las variables de fecha y hora del sistema operativo.

Si el formato de hora utilizado en el parámetro no se corresponde con el establecido en el sistema operativo, Qlik Sense no podrá realizar una interpretación correcta. Para resolver este problema, modifique la configuración o bien utilice una función de interpretación.

En los ejemplos para cada función, se asumen los formatos de fecha y hora hh:mm:ss y YYYY-MM-DD (ISO 8601) predeterminados.

Al procesar una marca de fecha-hora con una función de fecha u hora, Qlik Sense ignora todo parámetro de cambio horario, a menos que la función de fecha u hora incluya una posición geográfica.

Por ejemplo, `ConvertToLocalTime( filetime('Time.qvd'), 'Paris')` utilizaría los parámetros del horario de verano, mientras que `ConvertToLocalTime(filetime('Time.qvd'), 'GMT-01:00')` no utilizaría los parámetros del horario de verano.

Descripción general de las funciones de fecha y hora

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Expresiones de tiempo con enteros

second

Esta función devuelve un entero que representa el segundo en que la fracción de **expression** se interpreta como una hora de acuerdo con la interpretación numérica estándar.

```
second (expression)
```

minute

Esta función devuelve un entero que representa el minuto en que la fracción de la **expression** se interpreta como una hora de acuerdo con la interpretación numérica estándar.

```
minute (expression)
```

hour
Esta función devuelve un entero que representa la hora en que la fracción de **expression** se interpreta como una hora de acuerdo con la interpretación numérica estándar.

```
hour (expression)
```

day
Esta función devuelve un entero que representa el día en que la fracción de **expression** se interpreta como una fecha de acuerdo con la interpretación numérica estándar.

```
day (expression)
```

week
Esta función devuelve un entero que representa el número de semana conforme a la ISO 8601. El número de semana se calcula a partir de la interpretación de la fecha de la expresión, conforme a la interpretación numérica estándar.

```
week (expression)
```

month
Esta función devuelve un valor dual: un nombre de mes tal como se define en la variable de entorno **MonthNames** y un entero entre 1-12. El mes se calcula a partir de la interpretación de la fecha de la expresión, conforme a la interpretación numérica estándar.

```
month (expression)
```

year
Esta función devuelve un entero que representa el año en que **expression** se interpreta como una fecha de acuerdo con la interpretación numérica estándar.

```
year (expression)
```

weekyear
Esta función devuelve el año al que pertenece el número de semana según las variables de entorno. El número de semana varía entre 1 y 52 aproximadamente.

```
weekyear (expression)
```

weekday
Esta función devuelve un valor dual con lo siguiente:

- Un nombre de día tal como se define en la variable de entorno **DayNames**.
- Un número entero entre 0 y 6 correspondiente al día nominal de la semana (0-6).

```
weekday (date)
```

Funciones de indicación de tiempo

now

Esta función devuelve una marca de tiempo con la hora actual. La función devuelve valores en el formato de la variable del sistema **TimeStamp**. El valor predeterminado es 1 **timer_mode**.

```
now ([ timer_mode])
```

today

Esta función devuelve la fecha actual. La función devuelve valores en el formato de la variable del sistema `DateFormat`.

```
today ([timer_mode])
```

LocalTime

Esta función devuelve una marca de tiempo con la hora actual para una zona horaria especificada.

```
localtime ([timezone [, ignoreDST ]])
```

Funciones make

makedate

Esta función devuelve una fecha calculada a partir del año **YYYY**, el mes **MM** y el día **DD**.

```
makedate (YYYY [ , MM [ , DD ] ])
```

makeweekdate

Esta función devuelve una fecha calculada a partir del año, el número de semana y el día de la semana.

```
makeweekdate (YYYY [ , WW [ , D ] ])
```

maketime

Esta función devuelve una hora calculada a partir de la hora **hh**, el minuto **mm** y el segundo **ss**.

```
maketime (hh [ , mm [ , ss [ .fff ] ] ])
```

Otras funciones de tiempo

AddMonths

Esta función devuelve la fecha que figura **n** meses después de **startdate** o, si **n** es negativa, la fecha que figura **n** meses antes de **startdate**.

```
addmonths (startdate, n , [ , mode])
```

AddYears

Esta función devuelve la fecha que aparece **n** años después de **startdate** o, si **n** es negativa, la fecha que aparece **n** años antes de **startdate**.

```
addyears (startdate, n)
```

yeartodate
Esta función encuentra si la marca de tiempo (una fecha-hora) de entrada se encuentra dentro del año de la fecha en que se cargó el script por última vez, y devuelve True si lo hace o False si no lo hace.

```
yeartodate (date [ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Funciones de zona horaria

timezone
Esta función devuelve la zona horaria, tal como se define en el equipo informático donde está funcionando el motor de Qlik.

```
timezone ( )
```

GMT
Esta función devuelve la Greenwich Mean Time actual, tal como se deriva de la configuración regional.

```
GMT ( )
```

UTC
Devuelve la hora Coordinated Universal Time actual.

```
UTC ( )
```

daylightsaving
Devuelve el ajuste actual para el horario de verano, tal como se define en Windows.

```
daylightsaving ( )
```

converttolocaltime
Convierte una fecha-hora UTC o GMT en la fecha y hora local como un valor dual. El lugar puede ser cualquier ciudad, población o zona horaria del mundo.

```
converttolocaltime (timestamp [, place [, ignore_dst=false]])
```

Funciones de establecimiento de hora

setdateyear
Esta función toma como datos de entrada una marca de tiempo **timestamp** y un año **year** y actualiza la marca de tiempo **timestamp** con el año **year** especificado en los datos de entrada.

```
setdateyear  (timestamp, year)
```

setdateyearmonth
Esta función toma como datos de entrada una marca de tiempo **timestamp**, un mes **month** y un año **year** y actualiza la marca de tiempo **timestamp** con el año **year** y el mes **month** especificados en los datos de entrada.

```
setdateyearmonth  (timestamp, year, month)
```

Funciones in...

inyear

Esta función devuelve True si **timestamp** se encuentra dentro del año que contiene a **base_date**.

```
inyear (date, basedate , shift [, first_month_of_year = 1])
```

inyeartodate

Esta función devuelve True si **timestamp** se encuentra dentro de la parte del año que contiene a **base_date** hasta e incluido el último milisegundo de **base_date**.

```
inyeartodate  (date, basedate , shift [, first_month_of_year = 1])
```

inquarter

Esta función devuelve True si **timestamp** se encuentra dentro del trimestre que contiene a **base_date**.

```
inquarter  (date, basedate , shift [, first_month_of_year = 1])
```

inquartertodate

Esta función devuelve True si **timestamp** se encuentra dentro de la parte del trimestre que contiene a **base_date** hasta e incluido el último milisegundo de **base_date**.

```
inquartertodate  (date, basedate , shift [, first_month_of_year = 1])
```

inmonth

Esta función devuelve True si **timestamp** se encuentra dentro del mes que contiene a **base_date**.

```
inmonth (date, basedate , shift)
```

inmonthtodate

Devuelve True si **date** se encuentra dentro de la parte del mes que contiene a **basedate** hasta e incluido el último milisegundo de **basedate**.

```
inmonthtodate  (date, basedate , shift)
```

inmonths

Esta función determina si una fecha-hora se encuentra dentro del mismo periodo mensual, bimensual, trimestral, cuatrimestre o semestral que fecha base. También es posible hallar si la fecha-hora cae dentro de un periodo anterior o posterior.

```
inmonths (n, date, basedate , shift [, first_month_of_year = 1])
```

inmonthstodate

Esta función determina si una fecha-hora se encuentra dentro de la parte de un período mensual, bimensual, trimestral, cuatrimestral o semestral hasta e incluyendo el último milisegundo de `base_date`. También es posible hallar si la fecha-hora cae dentro de un periodo anterior o posterior.

```
inmonthstodate  (n, date, basedate , shift [, first_month_of_year = 1])
```

inweek
La función devuelve True si **timestamp** se encuentra dentro de la semana que contiene a **base_date**.

```
inweek (date, basedate , shift [, weekstart])
```

inweektodate
Esta función devuelve True si **timestamp** se encuentra dentro de la parte de la semana que contiene a **base_date** hasta e incluido el último milisegundo de **base_date**.

```
inweektodate  (date, basedate , shift [, weekstart])
```

inlunarweek
Esta función determina si **timestamp** se encuentra dentro de la semana lunar que contiene a **base_date**. Las semanas lunares en Qlik Sense se definen contando el 1 de enero como el primer día de la semana. Aparte de la última semana del año, cada semana contendrá exactamente siete días.

```
inlunarweek  (date, basedate , shift [, weekstart])
```

inlunarweektodate
Esta función halla si **timestamp** se encuentra dentro de la parte de la semana lunar hasta e incluido el último milisegundo de **base_date**. Las semanas lunares en Qlik Sense se definen contando el 1 de enero como el primer día de la semana y, aparte de la última semana del año, contendrán exactamente siete días.

```
inlunarweektodate  (date, basedate , shift [, weekstart])
```

inday
Esta función devuelve True si **timestamp** se encuentra dentro del día que contiene a **base_timestamp**.

```
inday (timestamp, basetimestamp , shift [, daystart])
```

indaytotime
Esta función devuelve True si **timestamp** se encuentra dentro de la parte del día que contiene a **base_timestamp** hasta e incluido el milisegundo exacto de **base_timestamp**.

```
indaytotime (timestamp, basetimestamp , shift [, daystart])
```

Funciones start ... end

yearstart
Esta función devuelve una marca de tiempo correspondiente al inicio del primer día del año que contiene a **date**. El formato de salida predefinido será el **DateFormat** definido en el script.

```
yearstart ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearend
Esta función devuelve un valor correspondiente a una marca de tiempo del último milisegundo del último día del año que contiene a **date**. El formato de salida predeterminado será el **DateFormat** definido en el script.

```
yearend ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearname
Esta función devuelve un año de cuatro dígitos como valor de visualización con un valor numérico subyacente correspondiente a una marca de tiempo (fecha-hora) del primer milisegundo del primer día del año que contiene **date**.

```
yearname (date [, shift = 0 [, first_month_of_year = 1]] )
```

quarterstart
Esta función devuelve un valor correspondiente a una marca de tiempo con el primer milisegundo del trimestre que contiene a **date**. El formato de salida predeterminado será el **DateFormat** establecido en el script.

```
quarterstart (date [, shift = 0 [, first_month_of_year = 1]])
```

quarterend
Esta función devuelve un valor correspondiente a una marca de tiempo del último milisegundo del trimestre que contiene a **date**. El formato de salida predeterminado será el **DateFormat** establecido en el script.

```
quarterend (date [, shift = 0 [, first_month_of_year = 1]])
```

quartername
Esta función devuelve un valor de visualización que muestra los meses del trimestre (con formato conforme a la variable de script **MonthNames**) y el año con un valor numérico subyacente correspondiente a una marca de tiempo (una fecha-hora) del primer milisegundo del primer día del trimestre.

```
quartername (date [, shift = 0 [, first_month_of_year = 1]])
```

monthstart
Esta función devuelve un valor correspondiente a una marca de tiempo (fecha-hora) del primer milisegundo del primer día del mes que contiene a **date**. El formato de salida predeterminado será el **DateFormat** establecido en el script.

```
monthstart (date [, shift = 0])
```

monthend
Esta función devuelve un valor correspondiente a una marca de tiempo (fecha-hora) del último milisegundo del último día del mes que contiene a `date`. El formato de salida predeterminado será el `DateFormat` establecido en el script.

```
monthend (date [, shift = 0])
```

monthname
Esta función devuelve un valor de visualización que muestra el mes (con formato de acuerdo con la variable de script **MonthNames**) y el año con un valor numérico subyacente correspondiente a una marca de tiempo (fecha-hora) del primer milisegundo del primer día del mes.

```
monthname (date [, shift = 0])
```

monthsstart
Esta función devuelve un valor correspondiente a la marca de tiempo del primer milisegundo del periodo mensual, bimensual, trimestral, cuatrimestral o semestral que contiene una fecha base. También es posible hallar la marca de tiempo de un periodo anterior o posterior.El formato de salida predefinido es el **DateFormat** definido en el script.

```
monthsstart (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsend
Esta función devuelve un valor correspondiente a una marca de tiempo del último milisegundo del periodo mensual, bimensual, trimestral, cuatrimestral o semestral que contiene una fecha base. También es posible hallar la marca de tiempo de un periodo anterior o posterior.

```
monthsend (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsname
Esta función devuelve un valor de visualización que representa el rango de los meses del período (con formato de acuerdo con la variable de script **MonthNames**), así como el año. El valor numérico subyacente corresponde a una marca de tiempo del primer milisegundo del periodo mensual, bimensual, trimestral, cuatrimestral o semestral que contiene una fecha base.

```
monthsname (n, date [, shift = 0 [, first_month_of_year = 1]])
```

weekstart
Esta función devuelve un valor correspondiente a una marca de tiempo del primer milisegundo del primer día de la semana natural que contiene a **date**. El formato de salida predefinido es el **DateFormat** definido en el script.

```
weekstart (date [, shift = 0 [,weekoffset = 0]])
```

weekend
Esta función devuelve un valor correspondiente a una marca de tiempo del último milisegundo del último día de la semana natural que contiene a **date**. El formato de salida predefinido será el **DateFormat** definido en el script.

```
weekend (date [, shift = 0 [,weekoffset = 0]])
```

weekname
Esta función devuelve un valor que muestra el número de año y de semana con un valor numérico subyacente correspondiente a una marca de tiempo del primer milisegundo del primer día de la semana que contiene a **date**.

```
weekname (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekstart
Esta función devuelve un valor correspondiente a una marca de tiempo del primer milisegundo del primer día de la semana lunar que contiene a **date**. Las semanas lunares en Qlik Sense se definen contando el 1 de enero como el primer día de la semana y, aparte de la última semana del año, contendrán exactamente siete días.

```
lunarweekstart (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekend
Esta función devuelve un valor correspondiente a una marca de tiempo del último milisegundo del último día de la semana lunar que contiene a **date**. Las semanas lunares en Qlik Sense se definen contando el 1 de enero como el primer día de la semana y, aparte de la última semana del año, contendrán exactamente siete días.

```
lunarweekend (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekname
Esta función devuelve un valor de visualización que muestra el año y el número de la semana lunar correspondiente a una marca de tiempo (fecha-hora) del primer milisegundo del primer día de la semana lunar que contiene a **date**. Las semanas lunares en Qlik Sense se definen contando el 1 de enero como el primer día de la semana y, aparte de la última semana del año, contendrán exactamente siete días.

```
lunarweekname (date [, shift = 0 [,weekoffset = 0]])
```

daystart
Esta función devuelve un valor correspondiente a una marca de tiempo (fecha-hora) con el primer milisegundo del día contenido en el argumento **time**. El formato de salida predeterminado será el **TimestampFormat** establecido en el script.

```
daystart (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayend
Esta función devuelve un valor correspondiente a una marca de tiempo (una fecha-hora) del milisegundo final del día contenido en **time**. El formato de salida predeterminado será el **TimestampFormat** establecido en el script.

```
dayend (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayname
Esta función devuelve un valor que muestra la fecha con un valor numérico subyacente correspondiente a una marca de tiempo (fecha-hora) del primer milisegundo del día que contiene a **time**.

```
dayname (timestamp [, shift = 0 [, dayoffset = 0]])
```

Funciones de número de día

age
La función **age** devuelve la edad en el momento **timestamp** (en años completos) de una persona nacida el día **date_of_birth**.

```
age (timestamp, date_of_birth)
```

networkdays
La función **networkdays** devuelve el número de días laborables (de lunes a viernes) entre e incluidos los días **start_date** y **end_date** teniendo en cuenta cualquier listado opcional de vacaciones: **holiday**.

```
networkdays  (start:date, end_date {, holiday})
```

firstworkdate

La función **firstworkdate** devuelve la última fecha de inicio para obtener un **no_of_workdays** (de lunes a viernes) sin sobrepasar **end_date** teniendo en cuenta la lista opcional de vacaciones. **end_date** y **holiday** deben ser fechas y horas válidas.

```
firstworkdate (end_date, no_of_workdays {, holiday} )
```

lastworkdate

La función **lastworkdate** devuelve la fecha más temprana de finalización para obtener el **no_of_workdays** (lunes-viernes) si comienza en **start_date** y teniendo en cuenta cualquier periodo vacacional opcionalmente indicado **holiday**. **start_date** y **holiday** deben ser fechas o fecha-hora válidas.

```
lastworkdate (start_date, no_of_workdays {, holiday})
```

daynumberofyear

Esta función calcula el número de día del año en el que cae una fecha-hora. El cálculo se hace a partir del primer milisegundo del primer día del año, pero el primer mes puede desplazarse.

```
daynumberofyear (date[,firstmonth])
```

daynumberofquarter

Esta función calcula el número de día del trimestre en el que cae una fecha-hora. Esta función se utiliza al crear un calendario maestro.

```
daynumberofquarter (date[,firstmonth])
```

addmonths

Esta función devuelve la fecha que figura **n** meses después de **startdate** o, si **n** es negativa, la fecha que figura **n** meses antes de **startdate**.

Sintaxis:

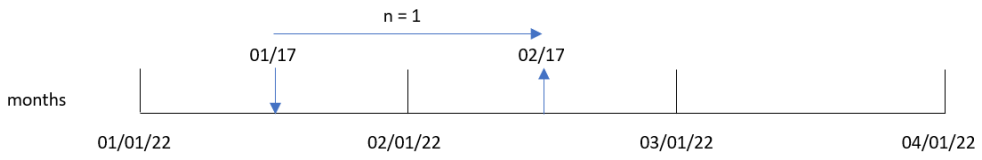
```
AddMonths(startdate, n , [ , mode])
```

Tipo de datos que devuelve: dual

La función `addmonths()` suma o resta un número definido de meses, `n`, desde una fecha de inicio `startdate` y devuelve la fecha resultante.

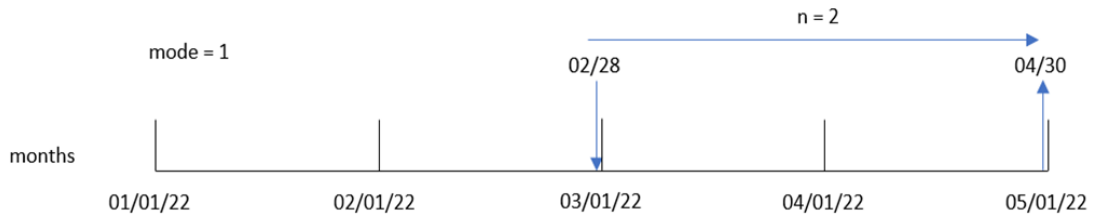
El argumento de `mode` impactará en los valores de `startdate` en o a partir del día 28 de cada mes. Al establecer el argumento de `mode` en 1, la función `addmonths()` devuelve una fecha que es igual en distancia relativa al final del mes que la fecha `startdate`.

Ejemplo de diagrama de la función `addmonths()`



Por ejemplo, el 28 de febrero es el último día del mes. Si la función `addmonths()`, con un `mode` de 1, se utiliza para devolver la fecha dos meses después, la función devolverá la última fecha de abril, el 30 de abril.

Diagrama de ejemplo de la función `addmonths()`, con `mode=1`



Argumentos

Argumento	Descripción
startdate	La fecha de inicio como una indicación de fecha, por ejemplo '2012-10-12'.
n	El número de meses como un entero positivo o negativo.
mode	Especifica si el mes se agrega en relación con el comienzo o el final del mes. El modo predeterminado es 0 para las adiciones relativas al comienzo del mes. Establezca el modo en 1 para las adiciones relativas al final del mes. Cuando el modo se establece en 1 y la fecha de entrada es 28 o superior, la función verifica cuántos días faltan para llegar al final del mes en la fecha de inicio. El mismo número de días para llegar al final del mes se establece en la fecha devuelta.

Cuándo se utiliza

La función `addmonths()` se utiliza normalmente en una expresión para encontrar una fecha un número determinado de meses antes o después de un período de tiempo.

Por ejemplo, la función `addmonths()` puede servir para identificar la fecha de finalización de los contratos de telefonía móvil.

Ejemplos de funciones

Ejemplo	Resultado
`addmonths ('01/29/2003' ,3)`	Devuelve "04/29/2003".
`addmonths ('01/29/2003',3,0)`	Devuelve "04/29/2003".
`addmonths ('01/29/2003',3,1)`	Devuelve "04/28/2003".
`addmonths ('01/29/2003',1,0)`	Devuelve "02/28/2003".
`addmonths ('01/29/2003',1,1)`	Devuelve "02/26/2003".
`addmonths ('02/28/2003',1,0)`	Devuelve "03/28/2003".
`addmonths ('02/28/2003',1,1)`	Devuelve "03/31/2003".
`addmonths ('01/29/2003',-3)`	Devuelve "10/29/2002".

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones entre 2020 y 2022, que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo, `two_months_later`, que devuelve la fecha correspondiente a dos meses después de que se haya realizado la transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `two_months_later`

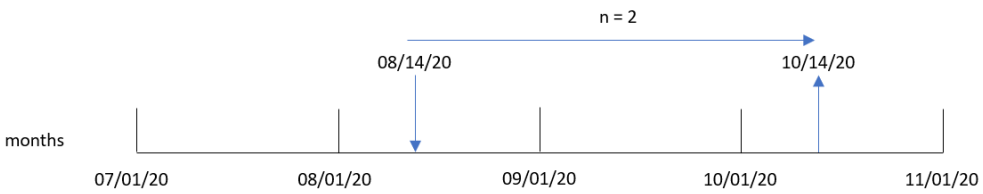
Tabla de resultados

date	two_months_later
01/10/2020	03/10/2020
02/28/2020	04/28/2020
04/09/2020	06/09/2020
04/16/2020	06/16/2020
05/21/2020	07/21/2020
08/14/2020	10/14/2020
10/07/2020	12/07/2020
12/05/2020	02/05/2021
01/22/2021	03/22/2021
02/03/2021	04/03/2021
03/17/2021	05/17/2021
04/23/2021	06/23/2021
05/04/2021	07/04/2021
06/30/2021	08/30/2021

date	two_months_later
07/26/2021	09/26/2021
12/27/2021	02/27/2022
02/02/2022	04/02/2022
02/26/2022	04/26/2022
03/07/2022	05/07/2022
03/11/2022	05/11/2022

El campo "two_months_later" se crea en la instrucción load anterior, mediante el uso de la función addmonths(). El primer argumento proporcionado identifica qué fecha se está evaluando. El segundo argumento es el número de meses que hay que sumar o restar a startdate. En este caso, se proporciona el valor de 2.

Diagrama de la función addmonths(), ejemplo sin argumentos adicionales



La transacción 8193 tuvo lugar el 14 de agosto. Por lo tanto, la función addmonths() devuelve el 14 de octubre de 2020 para el campo two_months_later.

Ejemplo 2: Final de mes relativo

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de fin de mes de 2022, que se carga en una tabla llamada Transactions.
- El campo de fecha proporcionado en el formato de la variable del sistema DateFormat: (MM/DD/AAAA).
- La creación de un campo, relative_two_months_prior, que devuelve la fecha relativa de fin de mes de dos meses antes de que se realizara la transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

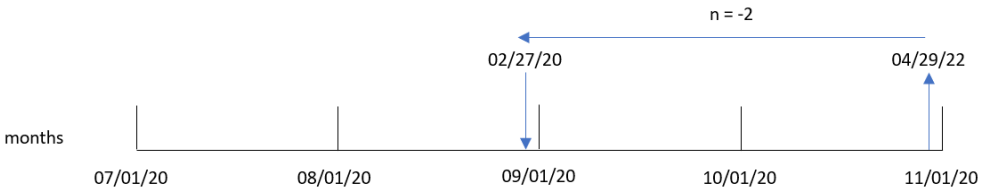
- `date`
- `relative_two_months_prior`

Tabla de resultados

date	relative_two_months_prior
01/28/2022	11/27/2021
01/31/2022	11/30/2021
02/28/2022	12/31/2021
04/29/2022	02/27/2022
04/30/2022	02/28/2022
05/31/2022	03/31/2022
08/14/2022	06/14/2022
10/07/2022	08/07/2022

El campo `relative_two_months_prior` se crea en la instrucción de carga anterior mediante el uso de la función `addmonths()`. El primer argumento proporcionado identifica qué fecha se está evaluando. El segundo argumento es el número de meses que hay que sumar o restar a `startdate`. En este caso, se proporciona el valor de -2. El último argumento es la moda, con un valor de 1, que obliga a la función a calcular la fecha relativa de fin de mes para todas las fechas mayores o iguales que 28.

Diagrama de la función `addmonths()`*, ejemplo con* `n=-2`



La transacción 8191 tiene lugar el 29 de abril de 2022. Inicialmente, dos meses antes fijaría el mes en febrero. Entonces, debido a que el tercer argumento de la función establece el modo en 1 y el valor del día es posterior al 27, la función calcula el valor relativo de fin de mes. La función identifica que el 29 es el penúltimo día de abril y por tanto devuelve el penúltimo día de febrero, el 27.

Ejemplo 3: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que devuelve la fecha de dos meses posteriores a la realización de la transacción se crea como una medida en un objeto gráfico.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Cree la siguiente medida:

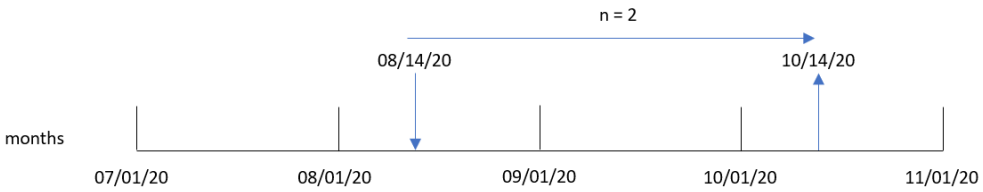
```
=addmonths(date,2)
```

Tabla de resultados

date	=addmonths(date,2)
01/10/2020	03/10/2020
02/28/2020	04/28/2020
04/09/2020	06/09/2020
04/16/2020	06/16/2020
05/21/2020	07/21/2020
08/14/2020	10/14/2020
10/07/2020	12/07/2020
12/05/2020	02/05/2021
01/22/2021	03/22/2021
02/03/2021	04/03/2021
03/17/2021	05/17/2021
04/23/2021	06/23/2021
05/04/2021	07/04/2021
06/30/2021	08/30/2021
07/26/2021	09/26/2021
12/27/2021	02/27/2022
02/02/2022	04/02/2022
02/26/2022	04/26/2022
03/07/2022	05/07/2022
03/11/2022	05/11/2022

La medida `two_months_later` se crea en el objeto gráfico usando la función `addmonths()`. El primer argumento proporcionado identifica qué fecha se está evaluando. El segundo argumento es el número de meses que hay que sumar o restar a `startdate`. En este caso, se proporciona el valor de 2.

Diagrama de la función `addmonths()`, ejemplo de objeto gráfico



La transacción 8193 tuvo lugar el 14 de agosto. Por lo tanto, la función `addmonths()` devuelve el 14 de octubre de 2020 para el campo `two_months_later`.

Ejemplo 4: Escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Mobile_Plans`.
- Información con el ID del contrato, la fecha de inicio, la duración del contrato y la cuota mensual.

El usuario final desea un objeto gráfico que muestre, por ID de contrato, la fecha de finalización de cada contrato telefónico.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `contract_id`
- `start_date`
- `contract_length`

Cree la siguiente medida para calcular la fecha de finalización de cada contrato:

```
=addmonths(start_date,contract_length, 0)
```

Tabla de resultados

contract_id	start_date	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	01/22/2022
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

contract_id	start_date	contract_length	=addmonths(start_date,contract_length,0)
8204	06/06/2022	24	06/06/2024
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

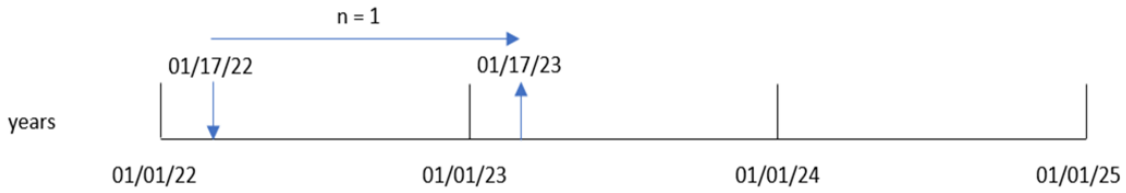
Esta función devuelve la fecha que aparece **n** años después de **startdate** o, si **n** es negativa, la fecha que aparece **n** años antes de **startdate**.

Sintaxis:

```
AddYears(startdate, n)
```

Tipo de datos que devuelve: dual

Ejemplo de diagrama de la función addyears()



La función addyears() suma o resta un número determinado de años, n, a partir de una startdate. Y después devuelve la fecha resultante.

Argumentos

Argumento	Descripción
startdate	La fecha de inicio como una indicación de fecha, por ejemplo '2012-10-12'.
n	El número de años como un entero positivo o negativo.

Ejemplos de funciones

Ejemplo	Resultado
addyears ('01/29/2010',3)	Devuelve "01/29/2013".
addyears ('01/29/2010',-1)	Devuelve "01/29/2009".

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: ejemplo sencillo

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones entre 2020 y 2022, que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo, `two_years_later`, que devuelve la fecha dos años después de que se haya realizado la transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `two_years_later`

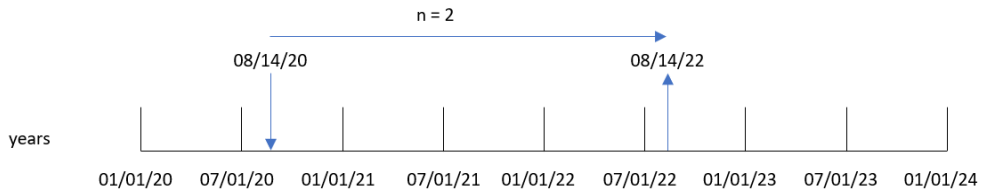
Tabla de resultados

date	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	05/21/2022
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

date	two_years_later
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

El campo "two_years_later" se crea en la instrucción load anterior, mediante el uso de la función addyears(). El primer argumento proporcionado identifica qué fecha se está evaluando. El segundo argumento es el número de años que hay que sumar o restar de la fecha de inicio. En este caso, se proporciona el valor de 2.

Diagrama de la función addyears(), ejemplo básico



La transacción 8193 tuvo lugar el 14 de agosto de 2020. Por lo tanto, la función addyears() devuelve el 14 de agosto de 2022 para el campo two_years_later.

Ejemplo 2: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones entre 2020 y 2022, que se carga en una tabla llamada Transactions.
- El campo de fecha proporcionado en el formato de la variable del sistema DateFormat: (MM/DD/AAAA).

En un objeto gráfico, cree una medida, prior_year_date, que devuelva la fecha de un año antes a que tuviera lugar la transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Cree la siguiente medida para calcular la fecha un año antes de cada transacción:

```
=addyears(date,-1)
```

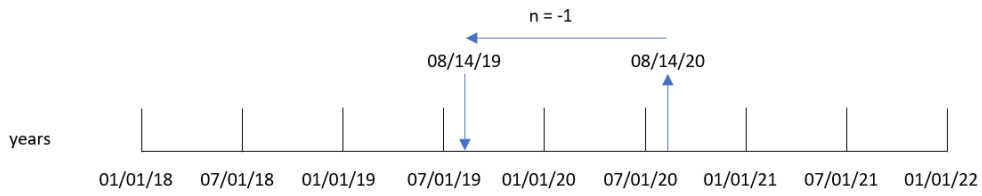
Tabla de resultados

date	=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

date	=addyears(date,-1)
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
02/26/2022	02/26/2021
03/07/2022	03/07/2021
03/11/2022	03/11/2021

La medida `one_year_prior` se crea en el objeto gráfico usando la función `addyears()`. El primer argumento proporcionado identifica qué fecha se está evaluando. El segundo argumento es el número de años que hay que sumar o restar de la fecha de inicio `startdate`. En este caso, se proporciona el valor de -1.

Diagrama de la función `addyears()`, ejemplo de objeto gráfico



La transacción 8193 tuvo lugar el 14 de agosto. Por lo tanto, la función `addyears()` devuelve el 14 de agosto de 2019 para el campo `one_year_prior`.

Ejemplo 3: escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Warranties`.
- Información con el ID del producto, la fecha de compra, la duración de la garantía y el precio de compra.

El usuario final desearía tener un objeto gráfico que muestre, por ID de producto, la fecha de finalización de la garantía de cada producto.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- product_id
- purchase_date
- warranty_length

Cree la siguiente medida para calcular la fecha de finalización de la garantía de cada producto:

```
=addyears(purchase_date,warranty_length)
```

Tabla de resultados

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

product_id	purchase_date	warranty_length	=addyears(purchase_date,warranty_length)
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
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

La función **age** devuelve la edad en el momento **timestamp** (en años completos) de una persona nacida el día **date_of_birth**.

Sintaxis:

```
age(timestamp, date_of_birth)
```

Puede ser una expresión.

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
timestamp	La fecha hora, o expresión que viene a dar una fecha hora, hasta la que calcular el número completo de años.
date_of_birth	Fecha de nacimiento de la persona cuya edad se calcula. Puede ser una expresión.

Ejemplos y resultados:
Estos ejemplos utilizan el formato de fecha **DD/MM/YYYY**. El formato de fecha se especifica en la sentencia **SET DateFormat** en la parte superior de su script de carga de datos. Cambie el formato en los ejemplos según se ajuste a sus necesidades.

Ejemplos de script

Ejemplo	Resultado
`age('25/01/2014', '29/10/2012')`	Devuelve 1.
`age('29/10/2014', '29/10/2012')`	Devuelve 2.

Ejemplo:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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;
```

La tabla resultante muestra los valores que devuelve age por cada uno de los registros de la tabla.

Tabla resultante

Member	DateOfBirth	Age
John	28/03/1989	26
Linda	10/12/1990	24
Steve	5/2/1992	23
Birg	31/3/1993	22

Member	DateOfBirth	Age
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

Convierte una fecha-hora UTC o GMT en la fecha y hora local como un valor dual. El lugar puede ser cualquier ciudad, población o zona horaria del mundo.

Sintaxis:

```
ConvertToLocalTime(timestamp [, place [, ignore_dst=false]])
```

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
timestamp	La indicación de fecha-hora, o expresión que devuelve una fecha-hora, que se ha de convertir.

Argumento	Descripción
place	Un lugar o zona horaria de la tabla de lugares y zonas horarias válidos a continuación. También puede usar GMT o UTC para definir la hora local. Son válidos los siguientes valores y rangos de desplazamiento de tiempo. • 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 *Si utiliza un desplazamiento de DST (es decir, especifica un valor de argumento **ignore_dst** que se evalúa como False), debe especificar un lugar, en vez de un desplazamiento de GMT, en el argumento de **place**. Esto se debe a que ajustar el horario de verano requiere latitudinal, además de la información longitudinal proporcionada por un desplazamiento GMT. Si desea información, vea Uso de compensaciones GMT en combinación con DST (page 636).* *Solo podrá utilizar desplazamientos de tiempo estándar. No es posible utilizar un desplazamiento de tiempo arbitrario, por ejemplo, GMT-04:27.*
ignore_dst	Si este argumento devuelve True, se ignora el horario de verano (DST). Los valores de argumento válidos que devuelven True incluyen `-1` y `True()`. Si este argumento se evalúa como False, la fecha-hora se ajusta al horario de verano. Los valores de argumento válidos que devuelven False incluyen `0` y `False()`. Si el valor del argumento **ignore_dst** no es válido, la función evalúa la expresión como si el valor de **ignore_dst** fuera True. Si no se especifica el valor del argumento **ignore_dst**, la función evalúa la expresión como si el valor **ignore_dst** fuera False.

Lugares y zonas horarias válidos

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	-	-	-

Ejemplos y resultados:

Ejemplos de script

Ejemplo	Resultado
`ConvertToLocalTime('2023-08-14 08:39:47','Paris')`	Devuelve "2023-08-14 10:39:47" y la correspondiente indicación de fecha-hora interna.
`ConvertToLocalTime(UTC(), 'Stockholm')`	Devuelve la hora de Estocolmo, ajustándose al horario de verano.
`ConvertToLocalTime(UTC(), 'Stockholm', -1)`	Devuelve la hora de Estocolmo, sin ajustarse al horario de verano.
`ConvertToLocalTime(UTC(), 'GMT-05:00')`	Devuelve la hora en la costa este norteamericana, por ejemplo en Nueva York. No se realiza ningún ajuste para el horario de verano porque se especifica un desplazamiento GMT, en lugar de una ubicación.
`ConvertToLocalTime(UTC(), 'New York', -1)`	Devuelve la hora de la costa este de América del Norte (Nueva York), sin el ajuste del horario de verano.
`ConvertToLocalTime(UTC(), 'New York', True())`	Devuelve la hora de la costa este de América del Norte (Nueva York), sin el ajuste del horario de verano.
`ConvertToLocalTime(UTC(), 'New York', 0)`	Devuelve la hora de la costa este de América del Norte (Nueva York), ajustándose al horario de verano.
`ConvertToLocalTime(UTC(), 'New York', False())`	Devuelve la hora de la costa este de América del Norte (Nueva York), ajustándose al horario de verano.

Uso de compensaciones GMT en combinación con DST

Tras la implementación de las bibliotecas de componentes internacionales para Unicode (ICU) en Qlik Sense, el uso de compensaciones GMT (hora media de Greenwich) en combinación con DST (horario de verano) requiere información adicional de latitud.

GMT supone un desplazamiento longitudinal (este-oeste), mientras que DST representa un desplazamiento latitudinal (norte-sur). Por ejemplo, Helsinki (Finlandia) y Johannesburgo (Sudáfrica) comparten la misma diferencia GMT+02:00, pero no comparten la misma diferencia DST. Esto significa que, además del desfase de GMT, cualquier desfase de DST requiere información sobre la posición latitudinal de la zona horaria local (datos de zona horaria geográfica) para tener información completa sobre las condiciones locales de DST.

day

Esta función devuelve un entero que representa el día en que la fracción de **expression** se interpreta como una fecha de acuerdo con la interpretación numérica estándar.

La función devuelve el día del mes de una fecha en particular. Se suele utilizar para encontrar un campo de un día como parte de una dimensión de calendario.

Sintaxis:

```
day(expression)
```

Tipo de datos que devuelve: Entero

Ejemplos de funciones

Ejemplo	Resultado
`day( 1971-10-12 )`	devuelve 12
`day( 35648 )`	devuelve 6, porque 35648 = 1997-08-06

Ejemplo 1: conjunto de datos DateFormat (script)

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de fechas denominado `Master_Calendar`. La variable de sistema `DateFormat`, que está configurada como DD/MM/AAAA.
- Un load precedente que crea un campo adicional, llamado `day_of_month`, usando la función `day()`.
- Un campo adicional, denominado `long_date`, usando la función `date()` para expresar el nombre completo del mes.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `long_date`
- `day_of_month`

Tabla de resultados

date	long_date	day_of_month
03/11/2022	11 marzo 2022	11
03/12/2022	12 marzo 2022	12
03/13/2022	13 marzo 2022	13
03/14/2022	14 marzo 2022	14
03/15/2022	15 marzo 2022	15
03/16/2022	16 marzo 2022	16
03/17/2022	17 marzo 2022	17
03/18/2022	18 marzo 2022	18
03/19/2022	19 marzo 2022	19

date	long_date	day_of_month
03/20/2022	20 marzo 2022	20
03/21/2022	21 marzo 2022	21

La función de script `day()` evalúa correctamente el día del mes.

Ejemplo 2: fechas ANSI (script)

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de fechas denominado `Master_Calendar`. Se utiliza la variable de sistema DateFormat DD/MM/AAAA. Sin embargo, las fechas que se incluyen en el conjunto de datos están en formato de fecha estándar ANSI.
- Un load precedente que crea un campo adicional, llamado `day_of_month`, usando la función `date()`.
- Un campo adicional, denominado `long_date`, usando la función `date()` para expresar la fecha con el nombre completo del mes.

Script de carga

```
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
 ];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `long_date`
- `day_of_month`

Tabla de resultados

date	long_date	day_of_month
03/11/2022	11 marzo 2022	11
03/12/2022	12 marzo 2022	12
03/13/2022	13 marzo 2022	13
03/14/2022	14 marzo 2022	14
03/15/2022	15 marzo 2022	15
03/16/2022	16 marzo 2022	16
03/17/2022	17 marzo 2022	17
03/18/2022	18 marzo 2022	18
03/19/2022	19 marzo 2022	19
03/20/2022	20 marzo 2022	20
03/21/2022	21 marzo 2022	21

La función de script `day()` evalúa correctamente el día del mes.

Ejemplo 3: fechas sin formato (script)

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de fechas denominado `Master_Calendar`. Se utiliza la variable de sistema `DateFormat` DD/MM/AAAA.
- Un load precedente que crea un campo adicional, llamado `day_of_month`, usando la función `day()`.
- La fecha original sin formato, denominada `unformatted_date`.
- Un campo adicional, denominado `long_date`, usando la función `date()` que sirve para convertir la fecha numérica en un campo de fecha con formato.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `unformatted_date`
- `long_date`
- `day_of_month`

Tabla de resultados

unformatted_date	long_date	day_of_month
44868	03 noviembre 2022	3
44898	03 diciembre 2022	3
44928	02 enero 2023	2
44958	01 febrero 2023	1
44988	03 marzo 2023	3
45008	03 marzo 2023	23
45018	02 abril 2023	2
45038	22 abril 2023	22

unformatted_date	long_date	day_of_month
45048	02 mayo 2023	2
45068	22 mayo 2023	22
45078	01 junio 2023	1

La función de script `day()` evalúa correctamente el día del mes.

Ejemplo 4: Calcular el mes de vencimiento (gráfico)

Script de carga y expresión de gráfico

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de pedidos realizados en marzo denominado `Orders`. La tabla contiene tres campos:
  - id
  - order_date
  - amount

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:`order_date`.

Para calcular la fecha de entrega, cree esta medida `=day(order_date+5)`.

Tabla de resultados

order_date	=day(order_date+5)
03/11/2022	16
03/12/2022	17
03/13/2022	18
03/14/2022	19
03/15/2022	20
03/16/2022	21
03/17/2022	22
03/18/2022	23
03/19/2022	24
03/20/2022	25
03/21/2022	26

La función `day()` determina correctamente que un pedido realizado el 11 de marzo se entregará el 16 conforme a un período de entrega de 5 días.

dayend

Esta función devuelve un valor correspondiente a una marca de tiempo (una fecha-hora) del milisegundo final del día contenido en **time**. El formato de salida predeterminado será el **TimestampFormat** establecido en el script.

Sintaxis:

```
DayEnd(time[, [period_no[, day_start]])
```

Cuándo se utiliza

La función `dayend()` se suele utilizar como parte de una expresión cuando el usuario desea que el cálculo utilice la fracción del día que aún no ha ocurrido. Por ejemplo, para calcular el total de gastos que faltan por hacer durante el día.

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
time	La fecha/hora que se ha de evaluar.
period_no	**period_no** es un entero, o una expresión que devuelve un entero, donde el valor 0 indica el día que contiene a **time**. Los valores negativos en **period_no** indican días precedentes y los valores positivos indican días subsiguientes.
day_start	Para especificar que los días no comienzan a la medianoche, indique un desplazamiento como una fracción de un día en **day_start**. Por ejemplo, 0,125 para indicar las 3.00 am. En otras palabras, para crear el desplazamiento, divida la hora de inicio por 24 horas. Por ejemplo, para que un día comience a las 7:00 am, utilice la fracción 7/24.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
dayend('01/25/2013 16:45:00')	Devuelve 01/25/2013 23:59:59. PM
dayend('01/25/2013 16:45:00', -1)	Devuelve 01/24/2013 23:59:59. PM
dayend('01/25/2013 16:45:00', 0, 0.5)	Devuelve 01/26/2013 11:59:59. PM

Ejemplo 1: script básico

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene una lista de fechas que se carga en una tabla denominada "Calendar".
- La variable de sistema `DateFormat` predefinida `(MM/DD/YYYY)`.
- Un load precedente que crea un campo adicional, llamado "`EOD_timestamp`", usando la función `dayend ()`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `EOD_timestamp`

Tabla de resultados

date	EOD_timestamp
03/11/2022 1:47:15 AM	3/11/2022 11:59:59 PM
03/12/2022 4:34:58 AM	3/12/2022 11:59:59 PM
03/13/2022 5:15:55 AM	3/13/2022 11:59:59 PM
03/14/2022 9:25:14 AM	3/14/2022 11:59:59 PM
03/15/2022 10:06:54 AM	3/15/2022 11:59:59 PM

date	EOD_timestamp
03/16/2022 10:44:42 AM	3/16/2022 11:59:59 PM
03/17/2022 11:33:30 AM	3/17/2022 11:59:59 PM
03/18/2022 12:58:14 PM	3/18/2022 11:59:59 PM
03/19/2022 4:23:12 PM	3/19/2022 11:59:59 PM
03/20/2022 6:42:15 PM	3/20/2022 11:59:59 PM
03/21/2022 7:41:16 PM	3/21/2022 11:59:59 PM

Como puede ver en la tabla anterior, la marca de tiempo (fecha-hora) del final del día se genera para cada fecha de nuestro conjunto de datos. La marca de tiempo está en el formato de la variable del sistema `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

Cargará un conjunto de datos que contiene reservas de servicios en una tabla denominada "Servicios".

El conjunto de datos incluye los siguientes campos:

- `service_id`
- `service_date`
- `amount`

Creará dos nuevos campos en la tabla:

- `deposit_due_date`: La fecha en que debe recibirse el depósito. Este es el final del día tres días antes de la fecha `service_date`.
- `final_payment_due_date`: La fecha en que debe recibirse el pago final. Este es el final del día, siete días antes de la fecha `service_date`.

Los dos campos anteriores se crean en un load anterior mediante la función `dayend()` y proporcionan los dos primeros parámetros: `time` y `period_no`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- service_date
- deposit_due_date
- final_payment_due_date

Tabla de resultados

service_date	deposit_due_date	final_payment_due_date
03/11/2022 9:25:14 AM	3/8/2022 11:59:59 PM	3/18/2022 11:59:59 PM
03/12/2022 10:06:54 AM	3/9/2022 11:59:59 PM	3/19/2022 11:59:59 PM
03/13/2022 10:44:42 AM	3/10/2022 11:59:59 PM	3/20/2022 11:59:59 PM
03/14/2022 11:33:30 AM	3/11/2022 11:59:59 PM	3/21/2022 11:59:59 PM
03/15/2022 12:58:14 PM	3/12/2022 11:59:59 PM	3/22/2022 11:59:59 PM
03/16/2022 4:23:12 PM	3/13/2022 11:59:59 PM	3/23/2022 11:59:59 PM
03/17/2022 6:42:15 PM	3/14/2022 11:59:59 PM	3/24/2022 11:59:59 PM
03/18/2022 7:41:16 PM	3/15/2022 11:59:59 PM	3/25/2022 11:59:59 PM
03/19/2022 8:14:15 PM	3/16/2022 11:59:59 PM	3/26/2022 11:59:59 PM
03/20/2022 9:23:51 PM	3/17/2022 11:59:59 PM	3/27/2022 11:59:59 PM
03/21/2022 10:04:41 PM	3/18/2022 11:59:59 PM	3/28/2022 11:59:59 PM

Los valores de los nuevos campos están en `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Porque se utilizó la función `dayend()`, los valores de la marca de tiempo son todos el último milisegundo del día.

Los valores de la fecha de vencimiento del depósito son tres días antes de la fecha de servicio porque el segundo argumento que se indica en la función `dayend()` es negativo.

Los valores de la fecha de vencimiento del pago final son siete días después de la fecha de servicio porque el segundo argumento que se indica en la función `dayend()` es positivo.

Ejemplo 3: script de day_start

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El conjunto de datos y el escenario utilizados en este ejemplo son los mismos que en el ejemplo anterior.

Como en el ejemplo anterior, creará dos nuevos campos:

- `deposit_due_date`: La fecha en que debe recibirse el depósito. Este es el final del día tres días antes de la fecha `service_date`.
- `final_payment_due_date`: La fecha en que debe recibirse el pago final. Este es el final del día, siete días antes de la fecha `service_date`.

Sin embargo, su empresa desearía operar bajo una política en la que la jornada laboral comienza a las 5 PM y termina a las 5 PM del día siguiente. Su empresa podrá controlar luego las transacciones que se produzcan en esas horas de trabajo.

Para lograr estos requisitos, los dos campos anteriores se crean en un load precedente utilizando la función `dayend()` y utilizan los tres argumentos: `time`, `period_no` y `day_start`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `service_date`
- `deposit_due_date`
- `final_payment_due_date`

Tabla de resultados

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

Aunque las fechas siguen siendo las mismas que en el ejemplo 2, las fechas tienen ahora una fecha-hora del último milisegundo antes de las 5:00 p.m., porque el valor del tercer argumento, `day_start` indicado en la función `dayend()` es 17/24.

Ejemplo 4: ejemplo de gráfico

Script de carga y expresión de gráfico

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El conjunto de datos y el escenario utilizados en este ejemplo son los mismos que en los dos ejemplos anteriores. A su empresa le gustaría operar bajo una política en la que la jornada laboral comienza a las 17:00 p.m. y termina a las 17:00 p.m. del día siguiente.

Como en el ejemplo anterior, creará dos nuevos campos:

- `deposit_due_date`: La fecha en que debe recibirse el depósito. Este es el final del día tres días antes de la fecha `service_date`.
- `final_payment_due_date`: La fecha en que debe recibirse el pago final. Este es el final del día, siete días antes de la fecha `service_date`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

`service_date`.

Para crear el campo `deposit_due_date`, cree esta medida.

```
=dayend(service_date,-3,17/24).
```

Después, para crear el campo `final_payment_due_date`, cree esta medida:

```
=dayend(service_date,7,17/24).
```

Tabla de resultados

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	3/19/2022 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
03/18/2022	3/15/2022 16:59:59 PM	3/25/2022 16:59:59 PM
03/19/2022	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	3/28/2022 16:59:59 PM

Los valores de los nuevos campos están en `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Porque se utilizó la función `dayend()`, los valores de la marca de tiempo son todos el último milisegundo del día.

Los valores de la fecha de vencimiento del pago son tres días antes de la fecha de servicio porque el segundo argumento que se indica en la función `dayend()` es negativo.

Los valores de la fecha de vencimiento del pago final son siete días después de la fecha de servicio porque el segundo argumento que se indica en la función `dayend()` es positivo.

Las fechas tienen una marca de tiempo del último milisegundo antes de las 5:00 PM porque el valor del tercer argumento, `day_start`, indicado en la función `dayend()` es 17/24.

Argumentos

Argumento	Descripción
time	La fecha/hora que se ha de evaluar.
period_no	**period_no** es un entero, o una expresión que devuelve un entero, donde el valor 0 indica el día que contiene a **time**. Los valores negativos en **period_no** indican días precedentes y los valores positivos indican días subsiguientes.
day_start	Para especificar que los días no comienzan a la medianoche, indique un desplazamiento como una fracción de un día en **day_start**. Por ejemplo, 0,125 para indicar las 3.00 am.

daylightsaving

Devuelve el ajuste actual para el horario de verano, tal como se define en Windows.

Sintaxis:

```
DaylightSaving( )
```

Tipo de datos que devuelve: dual

Ejemplo:

```
daylightsaving( )
```

dayname

Esta función devuelve un valor que muestra la fecha con un valor numérico subyacente correspondiente a una marca de tiempo (fecha-hora) del primer milisegundo del día que contiene a **time**.

Sintaxis:

```
DayName(time[, period_no [, day_start]])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
time	La fecha/hora que se ha de evaluar.
period_no	**period_no** es un entero, o una expresión que devuelve un entero, donde el valor 0 indica el día que contiene a **time**. Los valores negativos en **period_no** indican días precedentes y los valores positivos indican días subsiguientes.
day_start	Para especificar que los días no comienzan a la medianoche, indique un desplazamiento como una fracción de un día en **day_start**. Por ejemplo, 0,125 para indicar las 3.00 am.

Ejemplos y resultados:
Estos ejemplos utilizan el formato de fecha **DD/MM/YYYY**. El formato de fecha se especifica en la sentencia **SET DateFormat** en la parte superior de su script de carga de datos. Cambie el formato en los ejemplos según se ajuste a sus necesidades.

Ejemplos de script

Ejemplo	Resultado
`dayname('25/01/2013 16:45:00')`	Devuelve 25/01/2013.
`dayname('25/01/2013 16:45:00', -1)`	Devuelve 24/01/2013.
`dayname('25/01/2013 16:45:00', 0, 0.5 )`	Devuelve 25/01/2013. Mostrar la fecha-hora completa muestra el valor numérico subyacente correspondiente a '25/01/2013 12:00:00.000.

Ejemplo:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

En este ejemplo, el nombre del día se crea a partir de la fecha hora que marca el comienzo del día posterior a cada fecha de facturación en la tabla.

```
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;
```

La tabla resultante contiene las fechas originales y una columna con el valor de retorno de la función dayname(). Podemos mostrar la fecha hora completa especificando el formato en el panel de propiedades.

Tabla de resultados

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
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

Esta función calcula el número de día del trimestre en el que cae una fecha-hora. Esta función se utiliza al crear un calendario maestro.

Sintaxis:

```
DayNumberOfQuarter(timestamp[,start_month])
```

Tipo de datos que devuelve: Entero

Argumentos

Argumento	Descripción
timestamp	La fecha o marca de tiempo para evaluar.
start_month	Especificando un **start_month** entre 2 y 12 (1, si se omite), el comienzo del año se puede mover hacia delante, al primer día de cualquier mes. Por ejemplo, si desea trabajar con un año fiscal que comience el 1 de marzo, especifique **start_month** = 3.

Estos ejemplos utilizan el formato de fecha **DD/MM/YYYY**. El formato de fecha se especifica en la sentencia **SET DateFormat** en la parte superior de su script de carga de datos. Cambie el formato en los ejemplos según se ajuste a sus necesidades.

Ejemplos de funciones

Ejemplo	Resultado
`DayNumberOfQuarter('12/09/2014')`	Devuelve 74, el número del día del trimestre actual.
`DayNumberOfQuarter ('12/09/2014',3)`	Devuelve 12, el número del día del trimestre actual. En este caso, el primer trimestre comienza en marzo (porque start_month se especifica como 3). Esto significa que el trimestre actual es el tercer trimestre, que empieza el 1 de septiembre.

Ejemplo 1 : Inicio del año en enero (script)

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos simple que contiene una lista de fechas, que se carga en una tabla denominada `Calendar`. Se utiliza la variable predefinida del sistema `DateFormat`: DD/MM/AAAA.
- Un load precedente que crea un campo adicional, llamado `DayNrQtr`, usando la función `DayNumberOfQuarter()`.

Aparte de la fecha, no se proporcionan parámetros adicionales a la función.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- daynrqtr

Tabla de resultados

date	daynrqtr
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
02/28/2022	59
03/01/2022	61
03/31/2022	91
04/01/2022	1

El primer día del año es el 1 de enero porque no se indicó ningún segundo argumento a la función DayNumberOfQuarter().

El 1 de enero es el primer día del trimestre, mientras que el 1 de febrero es el día 32º del trimestre. El 31 de marzo es el 91º y último día del trimestre, mientras que el 1 de abril es el primer día del 2º trimestre.

Ejemplo 2 : Inicio del año en febrero (script)

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- El mismo conjunto de datos del primer ejemplo.
- Se utiliza la variable predefinida del sistema DateFormat DD/MM/AAAA.
- Un argumento start_month que comienza el 1 de febrero. Esto fija el año financiero en el 1 de febrero.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `daynrqtr`

Tabla de resultados

date	daynrqtr
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	1
02/10/2022	10
02/28/2022	28
03/01/2022	30
03/31/2022	60
04/01/2022	61

El primer día del año es el 1 de febrero porque el segundo argumento introducido en la función `DayNumberOfQuarter()` era 2.

El primer trimestre del año opera entre febrero y abril, mientras que el cuarto trimestre lo hace entre noviembre y enero. Esto se muestra en la tabla de resultados, donde el 1 de febrero es el primer día del trimestre, mientras que el 31 de enero es el 92º y último día del trimestre.

Ejemplo 3 : Inicio del año en enero (gráfico)

Script de carga y expresión de gráfico

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- El mismo conjunto de datos del primer ejemplo.
- Se utiliza la variable predefinida del sistema `DateFormat` DD/MM/AAAA.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El valor del día del trimestre se calcula a través de una medida en un objeto gráfico.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Cree la siguiente medida:

```
=daynumberofquarter(date)
```

Tabla de resultados

date	=daynumberofquarter(date)
01/01/2022	1
01/10/2022	10

date	=daynumberofquarter(date)
01/31/2022	31
02/01/2022	32
02/10/2022	41
02/28/2022	59
03/01/2022	61
03/31/2022	91
04/01/2022	1

El primer día del año es el 1 de enero porque no se indicó ningún segundo argumento en la función `DayNumberOfQuarter()`.

El 1 de enero es el primer día del trimestre, mientras que el 1 de febrero es el día 32º del trimestre. El 31 de marzo es el 91º y último día del trimestre, mientras que el 1 de abril es el primer día del 2º trimestre.

Ejemplo 4: Inicio del año en febrero (gráfico)

Script de carga y expresión de gráfico

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- El mismo conjunto de datos del primer ejemplo.
- Se utiliza la variable predefinida del sistema `DateFormat` DD/MM/AAAA.
- El ejercicio financiero va del 1 de febrero al 31 de enero.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El valor del día del trimestre se calcula a través de una medida en un objeto gráfico.

Script de carga

```
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
];
```

Objeto gráfico

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Cree la siguiente medida:

```
=daynumberofquarter(date,2)
```

Resultados

Tabla de resultados

date	=daynumberofquarter(date,2)
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	1
02/10/2022	10
02/28/2022	28
03/01/2022	30
03/31/2022	60
04/01/2022	61

El primer día del año es el 1 de enero porque el segundo argumento introducido en la función `DayNumberOfQuarter()` era 2.

El primer trimestre del año opera entre febrero y abril, mientras que el cuarto trimestre lo hace entre noviembre y enero. Esto se evidencia en la tabla de resultados, donde el 1 de febrero es el primer día del trimestre, mientras que el 31 de enero es el 92º y último día del trimestre.

daynumberofyear

Esta función calcula el número de día del año en el que cae una fecha-hora. El cálculo se hace a partir del primer milisegundo del primer día del año, pero el primer mes puede desplazarse.

Sintaxis:

```
DayNumberOfYear(timestamp[,start_month])
```

Tipo de datos que devuelve: Entero

Argumentos

Argumento	Descripción
timestamp	La fecha o marca de tiempo para evaluar.
start_month	Especificando un **start_month** entre 2 y 12 (1, si se omite), el comienzo del año se puede mover hacia delante, al primer día de cualquier mes. Por ejemplo, si desea trabajar con un año fiscal que comience el 1 de marzo, especifique **start_month** = 3.

Estos ejemplos utilizan el formato de fecha **DD/MM/YYYY**. El formato de fecha se especifica en la sentencia **SET DateFormat** en la parte superior de su script de carga de datos. Cambie el formato en los ejemplos según se ajuste a sus necesidades.

Ejemplos de funciones

Ejemplo	Resultado
`DayNumberOfYear( '12/09/2014' )`	Devuelve 256, el número de día contado desde el primer día del año.
`DayNumberOfYear( '12/09/2014',3 )`	Devuelve 196, el número de día contado desde el día 1 de marzo.

Ejemplo 1 : Inicio del año en enero (script)

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos simple que contiene una lista de fechas, que se carga en una tabla denominada `Calendar`. Se utiliza la variable predefinida del sistema `DateFormat`: DD/MM/AAAA.
- Un load precedente que crea un campo adicional, llamado `daynryear`, usando la función `DayNumberOfYear()`.

Aparte de la fecha, no se proporcionan parámetros adicionales a la función.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `daynryear`

Tabla de resultados

date	daynryear
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
06/30/2022	182
07/26/2022	208
10/31/2022	305
11/01/2022	306
12/31/2022	366

El primer día del año es el 1 de enero porque no se indicó ningún segundo argumento en la función `DayNumberOfYear()`.

El 1 de enero es el primer día del trimestre, mientras que el 1 de febrero es el día 32º del trimestre. El 30 de junio es el 182º, mientras que el 31 de diciembre es el 366º y último día del año.

Ejemplo 2: Inicio del año en noviembre (script)

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- El mismo conjunto de datos del primer ejemplo.
- Se utiliza la variable predefinida del sistema `DateFormat` DD/MM/AAAA.
- Un argumento `start_month` que comienza el 1 de noviembre. Esto fija el año financiero en el 1 de noviembre.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `daynryear`

Tabla de resultados

date	daynryear
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	93
02/10/2022	102
06/30/2022	243
07/26/2022	269
10/31/2022	366
11/01/2022	1
12/31/2022	61

El primer día del año es el 1 de noviembre porque el segundo argumento introducido en la función `DayNumberOfYear()` era 11.

El 1 de enero es el primer día del trimestre, mientras que el 1 de febrero es el día 32º del trimestre. El 30 de junio es el 182º, mientras que el 31 de diciembre es el 366º y último día del año.

Ejemplo 3 : Inicio del año en enero (gráfico)

Script de carga y expresión de gráfico

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- El mismo conjunto de datos del primer ejemplo.
- Se utiliza la variable predefinida del sistema `DateFormat` DD/MM/AAAA.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El valor del día del trimestre se calcula a través de una medida en un objeto gráfico.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Cree la siguiente medida:

```
=daynumberofyear(date)
```

Tabla de resultados

date	=daynumberofyear(date)
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
06/30/2022	182
07/26/2022	208
10/31/2022	305
11/01/2022	306
12/31/2022	366

El primer día del año es el 1 de enero porque no se indicó ningún segundo argumento en la función `DayNumberOfYear()`.

El 1 de enero es el primer día del año, mientras que el 1 de febrero es el día 32º del año. El 30 de junio es el 182º, mientras que el 31 de diciembre es el 366º y último día del año.

Ejemplo 4: Inicio del año en noviembre (gráfico)

Script de carga y expresión de gráfico

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- El mismo conjunto de datos del primer ejemplo.
- Se utiliza la variable predefinida del sistema `DateFormat` DD/MM/AAAA.
- El ejercicio financiero va del 1 de noviembre al 31 de octubre.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El valor del día del año se calcula mediante una medida en un objeto gráfico.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Cree la siguiente medida:

```
=daynumberofyear(date)
```

Tabla de resultados

date	**=daynumberofyear(date,11)**
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	93
02/10/2022	102
06/30/2022	243
07/26/2022	269

date	=daynumberofyear(date,11)
10/31/2022	366
11/01/2022	1
12/31/2022	61

El primer día del año es el 1 de noviembre porque el segundo argumento introducido en la función `DayNumberOfYear()` era 11.

El año fiscal va de noviembre a octubre. Esto se muestra en la tabla de resultados, donde el 1 de noviembre es el primer día del trimestre, mientras que el 31 de octubre es el 366º y último día del año.

daystart

Esta función devuelve un valor correspondiente a una marca de tiempo (fecha-hora) con el primer milisegundo del día contenido en el argumento **time**. El formato de salida predeterminado será el **TimestampFormat** establecido en el script.

Sintaxis:

```
DayStart(time[, [period_no[, day_start]])
```

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
time	La fecha/hora que se ha de evaluar.
period_no	**period_no** es un entero, o una expresión que devuelve un entero, donde el valor 0 indica el día que contiene a **time**. Los valores negativos en **period_no** indican días precedentes y los valores positivos indican días subsiguientes.
day_start	Para especificar que los días no comienzan a la medianoche, indique un desplazamiento como una fracción de un día en **day_start**. Por ejemplo, 0,125 para indicar las 3.00 am. En otras palabras, para crear el desplazamiento, divida la hora de inicio por 24 horas. Por ejemplo, para que un día comience a las 7:00 am, utilice la fracción 7/24.

Cuándo se utiliza

La función `daystart()` se suele utilizar como parte de una expresión cuando el usuario desea que el cálculo utilice la fracción del día que ya ha transcurrido. Por ejemplo, se podría usar para calcular los salarios totales obtenidos por los empleados en el día hasta el momento.

Estos ejemplos utilizan el formato de fecha y hora `'M/D/YYYY h:mm:ss[.fff] TT'`. El formato de fecha y hora se especifica en la sentencia `SET TimeStamp` en la parte superior de su script de carga de datos. Cambie el formato en los ejemplos según se ajuste a sus necesidades.

Ejemplos de funciones

Ejemplo	Resultado
daystart('01/25/2013 4:45:00 PM')	Devuelve 1/25/2013 12:00:00 AM.
daystart('1/25/2013 4:45:00 PM', -1)	Devuelve 1/24/2013 12:00:00 AM.
daystart('1/25/2013 16:45:00',0,0.5)	Devuelve 1/25/2013 12:00:00 PM.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: ejemplo sencillo

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos simple que contiene una lista de fechas, que se carga en una tabla denominada `Calendar`.
- Se utiliza la variable predefinida del sistema `TimeStampFormat` (`M/D/YYYY h:mm:ss[.fff] TT`).
- Una instrucción load precedente que crea un campo adicional, llamado `SOD_timestamp`, mediante la función `daystart()`.

Aparte de la fecha, no se proporcionan parámetros adicionales a la función.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `SOD_timestamp`

Tabla de resultados

date	SOD_timestamp
03/11/2022 1:47:15 AM	3/11/2022 12:00:00 AM
03/12/2022 4:34:58 AM	3/12/2022 12:00:00 AM
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	3/16/2022 12:00:00 AM
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	3/19/2022 12:00:00 AM
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

Como puede ver en la tabla anterior, la marca de tiempo (fecha-hora) del final del día se genera para cada fecha de nuestro conjunto de datos. La marca de tiempo está en el formato de la variable del sistema `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene multas de aparcamiento, se carga en una tabla llamada `Fines`. El conjunto de datos incluye los siguientes campos:
    - `id`
    - `due_date`
    - `number_plate`
    - `amount`
- Un load precedente usando la función `daystart()` y suministrando los tres parámetros: `time`, `period_no` y `day_start`. Este load precedente crea los siguientes dos nuevos campos de fecha:
    - Un campo de fecha `early_repayment_period`, que comienza siete días antes de la fecha de vencimiento del pago.
    - Un campo de fecha `late_penalty_period`, que comienza 14 días después de la fecha de vencimiento del pago.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- due_date
- early_repayment_period
- late_penalty_period

Tabla de resultados

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

Los valores de los nuevos campos están en `TimestampFormat M/DD/YYYY tt`. Como se utilizó la función `daystart()`, los valores de la marca de tiempo corresponden todos al primer milisegundo del día.

Los valores del período de amortización anticipada son siete días antes de la fecha de vencimiento, dado que el segundo argumento proporcionado en la función `daystart()` es negativo.

Los valores del período de pago atrasado son 14 días posteriores a la fecha de vencimiento, dado que el segundo argumento proporcionado en la función `daystart()` es positivo.

Ejemplo 3: day_start

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el ejemplo anterior.
- El mismo load precedente que en el ejemplo anterior.

En este ejemplo, configuramos la jornada laboral para que comience y finalice a las 7:00 a. m. todos los días.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- due_date
- early_repayment_period
- late_penalty_period

Tabla de resultados

due_date	early_repayment_period	late_penalty_period
02/11/2022	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

Las fechas ahora tienen una marca de las 7:00 a.m. porque el valor del argumento day_start indicado en la función daystart() era 7/24. Esto establece el comienzo del día a las 7:00 a. m.

Debido a que el campo due_date no tiene una marca de tiempo, se trata como las 12:00 a. m., que por lo tanto sigue siendo parte del día anterior, ya que los días comienzan y terminan a las 7:00 a. m. Por tanto, el plazo de amortización anticipada de una multa con vencimiento el 11 de febrero comienza el 3 de febrero a las 7:00 horas.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

Este ejemplo utiliza el mismo conjunto de datos y escenario que el ejemplo anterior.

Sin embargo, solo la tabla Fines original se carga en la aplicación, y los dos valores de fechas de vencimiento adicionales se calculan en un objeto gráfico.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: due_date.
2. Para crear el campo early_repayment_period, cree la siguiente medida.
   =daystart(due_date,-7,7/24)
3. Para crear el campo late_penalty_period, cree la siguiente medida:
   =daystart(due_date,14,7/24)

Tabla de resultados

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

Los valores de los nuevos campos están en `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Como se utilizó la función `daystart()`, los valores de la marca de tiempo corresponden al primer milisegundo del día.

Los valores del período de amortización anticipada son siete días antes de la fecha de vencimiento, dado que el segundo argumento proporcionado en la función `daystart()` era negativo.

Los valores del período de pago atrasado son 14 días posteriores a la fecha de vencimiento, dado que el segundo argumento proporcionado en la función `daystart()` era positivo.

Las fechas marcan las 7:00 a. m. porque el valor del tercer argumento indicado en la función `daystart()`, `day_start`, era 7/24.

firstworkdate

La función **firstworkdate** devuelve la última fecha de inicio para obtener un **no_of_workdays** (de lunes a viernes) sin sobrepasar **end_date** teniendo en cuenta la lista opcional de vacaciones. **end_date** y **holiday** deben ser fechas y horas válidas.

Sintaxis:

```
firstworkdate(end_date, no_of_workdays {, holiday} )
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
end_date	La fecha/hora de la fecha final que se ha de evaluar.
no_of_workdays	El número de días laborables que se ha de alcanzar.

Argumento	Descripción
holiday	Los períodos de vacaciones que deben excluirse de los días laborables. Las vacaciones se enuncian como cadena de fecha constante. Puede especificar más fechas de vacaciones separadas por comas. **Ejemplo:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Ejemplos y resultados:
Estos ejemplos utilizan el formato de fecha **DD/MM/YYYY**. El formato de fecha se especifica en la sentencia **SET DateFormat** en la parte superior de su script de carga de datos. Cambie el formato en los ejemplos según se ajuste a sus necesidades.

Ejemplos de script

Ejemplo	Resultado
`firstworkdate ('29/12/2014', 9)`	Devuelve '17/12/2014.
`firstworkdate ('29/12/2014', 9, '25/12/2014', '26/12/2014')`	Devuelve '15/12/2014 porque se tiene en cuenta un período de vacaciones de dos días.

Ejemplo:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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;
```

La tabla resultante muestra los valores que devuelve FirstWorkDate por cada uno de los registros de la tabla.

Tabla de resultados

InvID	EndDate	StartDate
1	28/03/2015	13/10/2014
2	10/12/2015	26/06/2015

InvID	EndDate	StartDate
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

Esta función devuelve la Greenwich Mean Time actual, tal como se deriva de la configuración regional. La función devuelve valores en el formato de la variable del sistema `TimestampFormat`.

Cada vez que se recargue la app, cualquier tabla del script de carga, variable u objeto gráfico que utilice la función `GMT` se ajustará a la última hora actual del meridiano de Greenwich según se deriva del reloj del sistema.

Sintaxis:

```
GMT( )
```

Tipo de datos que devuelve: dual

Estos ejemplos utilizan el formato de fecha y hora `M/D/YYYY h:mm:ss[.fff] TT`. El formato de fecha se especifica en la sentencia `SET TimestampFormat` en la parte superior de su script de carga de datos. Cambie el formato en los ejemplos según se ajuste a sus necesidades.

Ejemplos de funciones

Ejemplo	Resultado
`GMT()`	3/28/2022 2:47:36 PM

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: Variable (script)

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña. Este ejemplo establecerá la hora del meridiano de Greenwich actual como una variable en el script de carga usando la función GMT.

Script de carga

```
LET vGMT = GMT();
```

Resultados

Cargue los datos y cree una hoja. Cree un cuadro de texto usando el objeto gráfico **Texto e imagen**.

Agregue esta medida al cuadro de texto:

```
=vGMT
```

El cuadro de texto debe contener una línea de texto con una fecha y hora, similar a la que se muestra a continuación:

```
3/28/2022 2:47:36 PM
```

Ejemplo 2: Inicio del año en noviembre (script)

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene libros de biblioteca vencidos, se carga en una tabla llamada Overdue. Se utiliza la variable predefinida del sistema DateFormat: DD/MM/AAAA.
- La creación de un nuevo campo llamado days_overdue, que calcula cuántos días de atraso tiene cada libro.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- due_date
- book_id
- days_overdue

Tabla de resultados

due_date	book_id	days_overdue
01/01/2021	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

Los valores del campo days_overdue se calculan encontrando la diferencia entre la hora actual del meridiano de Greenwich, utilizando la función GMT(), y la fecha de vencimiento original. Para calcular solo los días, los resultados se redondean al número entero más cercano usando la función Floor().

Ejemplo 3: Objeto gráfico (gráfico)

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación, en una nueva pestaña. El script de carga contiene el mismo conjunto de datos que el ejemplo anterior. Se utiliza la variable predefinida del sistema `DateFormat`: DD/MM/AAAA.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El valor del número de días vencidos se calcula mediante una medida en un objeto gráfico.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `due_date`
- `book_id`

Cree la siguiente medida:

```
=Floor(GMT() - due_date)
```

Tabla de resultados

due_date	book_id	=Floor(GMT()-due_date)
01/01/2021	4	455

due_date	book_id	=Floor(GMT()-due_date)
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

Los valores del campo `days_overdue` se calculan encontrando la diferencia entre la hora actual del meridiano de Greenwich, utilizando la función `GMT()`, y la fecha de vencimiento original. Para calcular solo los días, los resultados se redondean al número entero más cercano usando la función `Floor()`.

hour

Esta función devuelve un entero que representa la hora en que la fracción de **expression** se interpreta como una hora de acuerdo con la interpretación numérica estándar.

Sintaxis:

```
hour(expression)
```

Tipo de datos que devuelve: Entero

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
`hour( '09:14:36' )`	La cadena de texto proporcionada se convierte implícitamente en una marca de tiempo, ya que coincide con el formato de marca de tiempo definido en la variable TimestampFormat. La expresión devuelve 9.
`hour( '0.5555' )`	La expresión devuelve 13 (porque 0,5555 = 13:19:55).

Ejemplo 1 : Variable (script)

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene transacciones por fecha y hora
- La variable de sistema `TimeStamp` predefinida `(M/D/YYYY h:mm:ss[.fff] TT)`

Cree un campo, "`hour`", que calcule cuándo se realizaron las compras.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- hour

Tabla de resultados

date	hour
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

Los valores del campo de hora se crean utilizando la función `hour()` e indicando la fecha como expresión en la sentencia de load anterior.

Ejemplo 2 : Objeto gráfico (gráfico)

Script de carga y expresión de gráfico

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- El mismo conjunto de datos del primer ejemplo.
- La variable de sistema `TimeStamp` predefinida `(M/D/YYYY h:mm:ss[.fff] TT)`.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. Los valores de `hour` se calculan por medio de una medida en un objeto gráfico.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Para calcular "`hour`", cree la siguiente medida:

```
=hour(date)
```

Tabla de resultados

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
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

Los valores de "`hour`" se crean usando la función `hour()` e introduciendo la fecha como expresión en una medida del objeto gráfico.

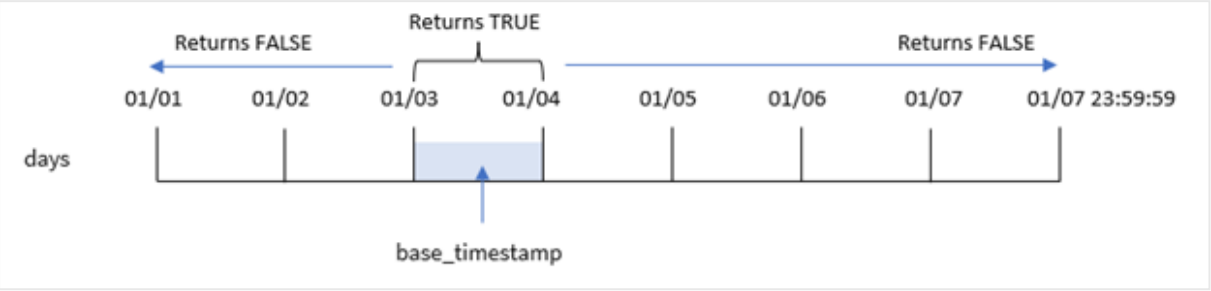
inday

Esta función devuelve True si **timestamp** se encuentra dentro del día que contiene a **base_timestamp**.

Sintaxis:

```
InDay (timestamp, base_timestamp, period_no[, day_start])
```

Diagrama de la función inday



La función `inday()` utiliza el argumento `base_timestamp` para identificar en qué día cae la marca de tiempo. La hora de inicio del día es, por defecto, la medianoche; pero puede cambiar la hora de inicio del día utilizando el argumento `day_start` de la función `inday()`. Una vez definido este día, la función devolverá resultados booleanos al comparar los valores de las marcas de tiempo prescritas con ese día.

Cuándo se utiliza

La función `inday()` devuelve un resultado booleano. Normalmente, este tipo de función se utilizará como condición en una expresión if (`if expression`). Esto devuelve una agregación o cálculo dependiente de si una fecha evaluada ocurrió en el día de la marca de tiempo en cuestión.

Por ejemplo, la función `inday()` puede utilizarse para identificar todos los equipos fabricados en un día determinado.

Tipo de datos que devuelve: Booleano

En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.

Argumentos

Argumento	Descripción
`timestamp`	La fecha y la hora que desea comparar con `base_timestamp`.
`base_timestamp`	La fecha y hora que se utiliza para evaluar la fecha-hora.
`period_no`	El día puede desplazarse mediante `period_no`. `period_no` es un entero, en el que el valor 0 indica el día que contiene a `base_timestamp`. Los valores negativos en `period_no` indican días precedentes y los valores positivos indican días subsiguientes.
`day_start`	Si desea trabajar con días que no comiencen a medianoche, indique un desplazamiento como una fracción de un día en `day_start`, por ejemplo, 0,125 para indicar las 3 AM.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	**Resultado**
`inday` ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	Devuelve Verdadero
`inday` ('01/12/2006 12:23:00 PM', '01/13/2006 12:00:00 AM', 0)	Devuelve Falso
`inday` ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	Devuelve Falso
`inday` ('01/11/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	Devuelve Verdadero
`inday` ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0, 0.5)	Devuelve Falso
`inday` ('01/12/2006 11:23:00 AM', '01/12/2006 12:00:00 AM', 0, 0.5)	Devuelve Verdadero

Ejemplo 1: Sentencia Load (script)

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene transacciones por marca de tiempo, que se carga en una tabla llamada `Transactions`.
- Un campo de fecha que se proporciona en el formato `(M/D/YYYY h:mm:ss[.fff] TT)` de la variable del sistema `TimeStamp`.
- Un load precedente que contiene la función `inday()` que se configura como el campo `in_day`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `in_day`

Tabla de resultados

date	in_day
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
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

El campo `in_day` se crea en la sentencia load anterior utilizando la función `inday()` e introduciendo el campo de fecha, una marca de tiempo codificada para el 5 de enero y un `period_no` de `0` como argumentos de la función.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

El script de carga utiliza el mismo conjunto de datos y el mismo escenario que se utilizó en el primer ejemplo.

Sin embargo, en este ejemplo, la tarea consiste en calcular si la fecha de la transacción ocurrió dos días antes del 5 de enero.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- in_day

Tabla de resultados

date	in_day
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0

date	in_day
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

En este caso, como se utiliza un `period_no` de `-2` como argumento de desplazamiento en la función `inday()`, esta determina si cada fecha de transacción tuvo lugar el 3 de enero. Esto se puede verificar en la tabla de salida, donde una transacción devuelve un resultado booleano TRUE.

Ejemplo 3: day_start

Script de carga y resultados

Vista general

El script de carga utiliza el mismo conjunto de datos y el mismo escenario que se utilizó en los ejemplos anteriores.

Sin embargo, en este ejemplo, la política de la empresa es que la jornada laboral comienza y termina a las 7 de la mañana.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `in_day`

Tabla de resultados

date	in_day
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

Como el argumento de `start_day` de 7/24, que es 7 AM, se utiliza en la función `inday()`, esta determina si cada fecha de transacción tuvo lugar el 4 de enero a partir de las 7 AM y el 5 de enero antes de las 7 AM.

Esto se puede verificar en la tabla de salida, donde las transacciones que tienen lugar después de las 7 de la mañana del 4 de enero devuelven un resultado booleano de mientras que las transacciones que tienen lugar después de las 7 de la mañana del 5 de enero devuelven un resultado booleano FALSE.

Ejemplo 4 : Objeto gráfico

Script de carga y expresión de gráfico

Vista general

El script de carga utiliza el mismo conjunto de datos y el mismo escenario que se utilizó en los ejemplos anteriores.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. Haremos un cálculo para ver si una transacción tiene lugar el 5 de enero creando una medida en un objeto gráfico.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

- date

Para calcular si una transacción tiene lugar el 5 de enero, cree la siguiente medida:

```
=inday(date,'01/05/2022 12:00:00 AM',0)
```

Tabla de resultados

date	inday(date,'01/05/2022 12:00:00 AM',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
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

Ejemplo 5: Escenario

Script de carga y resultados

Vista general

En este ejemplo, se ha identificado que, debido a un error del equipo, los productos que se fabricaron el 5 de enero eran defectuosos. El usuario final desea un objeto gráfico que muestre, por fecha, el estado de los productos fabricados "defectuosos" o "sin defectos" y el coste de los productos fabricados el 5 de enero.

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla llamada "Productos".
- La tabla contiene los siguientes campos:
    - ID de producto
    - hora de fabricación
    - precio de coste

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

```
=dayname(manufacture_date)
```

Cree las siguientes medidas:

- `=if(only(InDay(manufacture_date,makedate(2022,01,05),0)),'Defective','Faultless')`
- `=sum(cost_price)`

Establezca el **Formato numérico** de la medida en **Moneda**.

En **Aspecto**, deshabilite los **Totales**.

Tabla de resultados

dayname (manufacture_date)	=if(only(InDay(manufacture_date,makedate (2022,01,05),0)),'Defective','Faultless')	=sum(cost_ price)
01/01/2022	Sin defectos	44.67
01/02/2022	Sin defectos	36.34
01/03/2022	Sin defectos	51.75
01/04/2022	Sin defectos	89.69
01/05/2022	Defectuoso	170.78
01/06/2022	Sin defectos	74.23

La función `inday()` devuelve un valor booleano al evaluar las fechas de fabricación de cada uno de los productos. Para cualquier producto fabricado el 5 de enero, la función `inday()` devuelve un valor booleano de TRUE y marca los productos como "defectuosos". Para cualquier producto que devuelva un valor de FALSO, y que por lo tanto no se haya fabricado ese día, marca los productos como "Sin defectos".

indaytotime

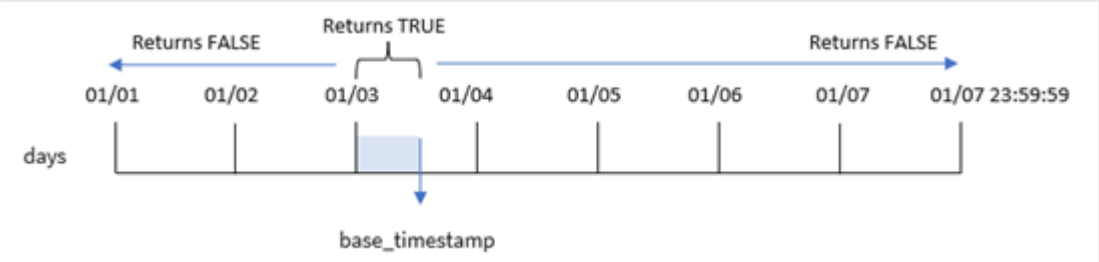
Esta función devuelve True si **timestamp** se encuentra dentro de la parte del día que contiene a **base_timestamp** hasta e incluido el milisegundo exacto de **base_timestamp**.

Sintaxis:

```
InDayToTime (timestamp, base_timestamp, period_no[, day_start])
```

La función `indaytotime()` devuelve un resultado booleano determinado por si un valor de marca de tiempo ocurre durante el segmento del día. El límite de inicio de este segmento es el comienzo del día, el cual se ha fijado por defecto en la medianoche, pero esto se puede modificar mediante el argumento `day_start` de la función `indaytotime()`. El límite final del segmento del día se determina mediante un argumento `base_timestamp` de la función.

Diagrama de la función `indaytotime`.



Cuándo se utiliza

La función `indaytotime()` devuelve un resultado booleano. Normalmente, este tipo de función se utilizará como condición en una expresión if (`if expression`). La función `indaytotime()` devuelve una agregación o cálculo dependiendo de si una marca de tiempo ocurrió en el segmento del día hasta la hora de la marca de tiempo base inclusive.

Por ejemplo, la función `indaytotime()` puede utilizarse para mostrar la suma de las ventas de entradas de los espectáculos que han tenido lugar hoy.

Tipo de datos que devuelve: Booleano

En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.

Argumentos

Argumento	Descripción
`timestamp`	La fecha y la hora que desea comparar con `base_timestamp`.
`base_timestamp`	La fecha y hora que se utiliza para evaluar la fecha-hora.
`period_no`	El día se puede desplazar mediante `period_no`. `period_no` es un entero, en el que el valor 0 indica el día que contiene a `base_timestamp`. Los valores negativos en `period_no` indican días precedentes y los valores positivos indican días subsiguientes.
`day_start`	(Opcional) Si desea trabajar con días que no comiencen a medianoche, indique un desplazamiento como una fracción de un día en `day_start`. Por ejemplo, use 0,125 para indicar las 3 AM.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
`indaytotime` ('01/12/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', 0)	Devuelve Verdadero

Ejemplo	Resultado
indaytotime ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	Devuelve Falso
indaytotime '01/11/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', -1)	Devuelve Verdadero

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones para el período comprendido entre el 4 y el 5 de enero se carga en una tabla denominada "Transacciones".
- Un campo de fecha que se proporciona en el formato (M/D/YYYY h:mm:ss[.fff] TT) de la variable del sistema TimeStamp.
- Una carga anterior que contiene la función indaytotime() que está configurada como el campo 'in_day_to_time' y que determina si cada una de las transacciones tiene lugar antes de las 9:00 a. m.

Script de carga

```
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
];
```

Resultados

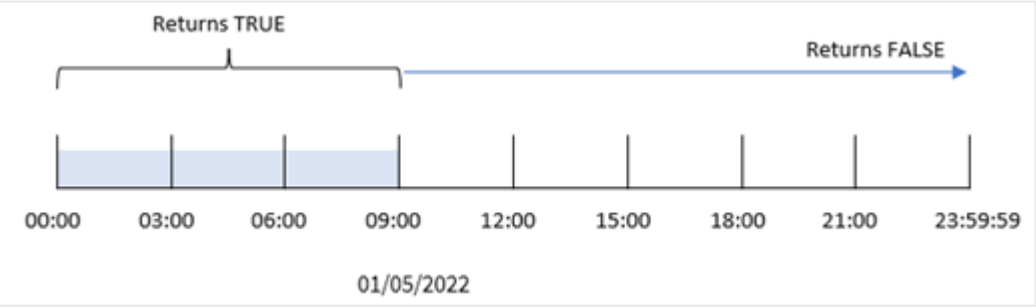
Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `in_day_to_time`

Tabla de resultados

fecha	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

Ejemplo 1: diagrama de la función `indaytotime` con límite de 9:00 AM.



El campo `in_day_to_time field` se crea en la sentencia de carga anterior utilizando la función `indaytotime ()` e introduciendo el campo de fecha, una marca de tiempo codificada para las 9:00 AM del 5 de enero y un desplazamiento de 0 como argumentos de la función. Todas las transacciones que se produzcan entre la medianoche y las 9:00 AM del 5 de enero devuelven VERDADERO.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

El script de carga utiliza el mismo conjunto de datos y el mismo escenario que se utilizó en el primer ejemplo.

Sin embargo, en este ejemplo, calcularemos si la fecha de la transacción ocurrió un día antes de las 9:00 AM del 5 de enero.

Script de carga

```
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
];
```

Resultados

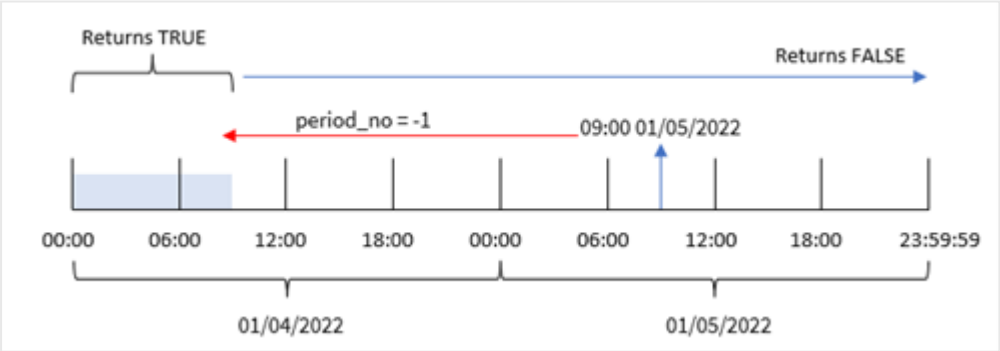
Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- in_day_to_time

Tabla de resultados

fecha	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

Ejemplo 2: diagrama de la función `indaytotime` con las transacciones del 4 de enero.



En este ejemplo, como se utilizó un desplazamiento de -1 como argumento de desplazamiento en la función `indaytotime()`, ésta determina si cada fecha de transacción tuvo lugar antes de las 9:00 AM del 4 de enero. Esto puede verificarse en la tabla de salida donde una transacción devuelve un resultado booleano de TRUE.

Ejemplo 3: day_start

Script de carga y resultados

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, la política de la empresa es que la jornada laboral comienza y termina a las 8 a.m.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

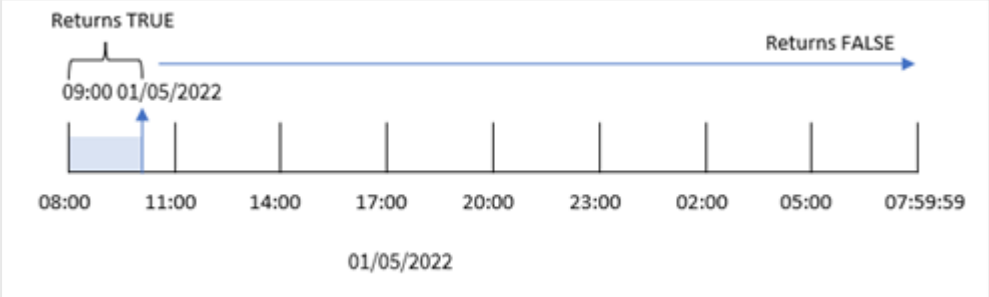
- `date`
- `in_day_to_time`

Tabla de resultados

fecha	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

fecha	in_day_to_time
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

Ejemplo 3: diagrama de la función `indaytotime` con transacciones de 8:00 AM a 9:00 AM.



Debido a que el argumento 8/24 de `start_day`, que equivale a las 8:00 a. m., se utiliza en la función `indaytotime()`, cada día comienza y termina a las 8:00 a. m. Por lo tanto, la función `indaytotime()` devolverá un resultado booleano TRUE para cualquier transacción que haya tenido lugar entre las 8:00 a. m. y las 9:00 a. m. del 5 de enero.

Ejemplo 4 : Objeto gráfico

Script de carga y expresión de gráfico

General

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. Haremos un cálculo para ver si una transacción tiene lugar el 5 de enero antes de las 9:00 AM creando una medida en un objeto gráfico.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

`date`.

Para determinar si una transacción tiene lugar el 5 de enero antes de las 9:00 AM, cree la siguiente medida:

```
=indaytotime(date,'01/05/2022 9:00:00 AM',0)
```

Tabla de resultados

fecha	**=indaytotime(date,'01/05/2022 9:00:00 AM',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

fecha	=indaytotime(date,'01/05/2022 9:00:00 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

La medida `in_day_to_time` se crea en el objeto gráfico utilizando la función `indaytotime()` e introduciendo el campo de fecha, una marca de tiempo codificada para las 9:00 a. m. del 5 de enero y un desplazamiento de `0` como argumentos de la función. Todas las transacciones que se produzcan entre la medianoche y las 9:00 AM del 5 de enero devuelven VERDADERO. Esto se valida en la tabla de resultados.

Ejemplo 5: Escenario

Script de carga y resultados

Vista general

En este ejemplo, un conjunto de datos que contiene las ventas de entradas de un cine local se carga en una tabla llamada Ticket_Sales. Hoy es 3 de mayo de 2022 y son las 11:00 AM.

El usuario desea que un objeto gráfico de KPI muestre los ingresos obtenidos de todos los espectáculos que han tenido lugar hoy.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cree un objeto KPI.
2. Cree una medida que muestre la suma de todas las ventas de entradas para los espectáculos que han tenido lugar hoy hasta el momento utilizando la función `indaytotime()`:

```
=sum(if(indaytotime([show time],'05/03/2022 11:00:00 AM',0),[ticket price],0))
```

3. Cree una etiqueta para el objeto KPI denominada "Ingresos actuales".
4. Establezca el **Formato numérico** de la medida en **Moneda**.

La suma total de las ventas de entradas hasta las 11:00 horas del 3 de mayo de 2022 es de 52,50 dólares.

La función `indaytotime ()` devuelve un valor booleano al comparar las horas de espectáculo de cada una de las ventas de entradas con la hora actual ('05/03/2022 11:00:00 AM'). Para cualquier espectáculo del 3 de mayo antes de las 11:00 horas, la función `indaytotime ()` devuelve un valor booleano de TRUE y su precio de entrada se incluirá en la suma total.

inlunarweek

Esta función determina si **timestamp** se encuentra dentro de la semana lunar que contiene a **base_date**. Las semanas lunares en Qlik Sense se definen contando el 1 de enero como el primer día de la semana. Aparte de la última semana del año, cada semana contendrá exactamente siete días.

Sintaxis:

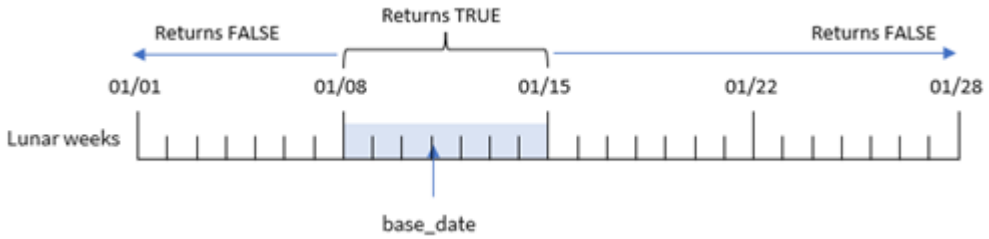
```
InLunarWeek (timestamp, base_date, period_no[, first_week_day])
```

Tipo de datos que devuelve: Booleano

En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.

La función `inlunarweek()` determina en qué semana lunar cae `base_date`. Luego devuelve un resultado booleano una vez que ha determinado si cada valor de marca de tiempo ocurre durante la misma semana lunar que el `base_date`.

Diagrama de la función `inlunarweek()`



Cuándo se utiliza

La función `inlunarweek()` devuelve un resultado booleano. Normalmente, este tipo de función se utiliza como condición en una expresión condicional. Esto devolvería una agregación o cálculo dependiendo de si la fecha evaluada ocurrió durante la semana lunar en cuestión.

Por ejemplo, la función `inlunarweek()` se puede utilizar para identificar todos los equipos fabricados en una determinada semana lunar.

Argumentos

Argumento	Descripción
timestamp	La fecha que desea comparar con **base_date**.
base_date	La fecha que se utiliza para evaluar la semana lunar.
period_no	La semana lunar puede desplazarse mediante **period_no**; period_no es un entero, en el que el valor 0 indica la semana lunar que contiene a **base_date**. Los valores negativos en **period_no** indican semanas lunares precedentes y los valores positivos indican semanas lunares subsiguientes.
first_week_day	Un desplazamiento que puede ser mayor que o menor que cero. Esto cambia el comienzo del año por el número especificado de días y/o fracciones de un día.

Ejemplos de funciones

Ejemplo	Resultado
`inlunarweek ('01/12/2013', '01/14/2013', 0)`	Devuelve TRUE, porque el valor de `timestamp`, 01/12/2013, cae en la semana del 01/08/2013 al 01/14/2013.
`inlunarweek ('01/12/2013', '01/07/2013', 0)`	Devuelve FALSE, porque la `base_date` 01/07/2013 está en la semana lunar definida como de 01/01/2013 a 01/07/2013.
`inlunarweek ('01/12/2013', '01/14/2013', -1)`	Devuelve FALSE. Especificar un valor de period_no como -1 cambia la semana a la semana anterior, del 01/01/2013 al 01/07/2013.
`inlunarweek ('01/07/2013', 01/14/2013', -1)`	Devuelve TRUE. Comparado con el ejemplo anterior, la fecha-hora `timestamp` está en la semana posterior a tenerse en cuenta el desplazamiento hacia atrás.

Ejemplo	Resultado
`inlunarweek ('01/11/2006', '01/08/2006', 0, 3)`	Devuelve `FALSE`. Especificar un valor de 3 para `first_week_day` significa que el inicio del año se calcula a partir del 01/04/2013. Por lo tanto, el valor de `base_date` cae en la primera semana y el valor de `timestamp` cae en la semana del 01/11/2013 al 01/17/2013.

La función `inlunarweek()` se utiliza a menudo en combinación con las siguientes funciones:

Funciones relacionadas

Función	Interacción
lunarweekname (page 876)	Esta función sirve para indicar el número de semana lunar del año en el que ocurre una fecha de entrada.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de transacciones en el mes de enero, que se carga en una tabla denominada `Transactions`.
- El campo de fecha se ha proporcionado en el formato (MM/DD/AAAA) de la variable del sistema `DateFormat`.

Cree un campo, `in_lunar_week`, que determine si las transacciones tuvieron lugar en la misma semana lunar que el 10 de enero.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

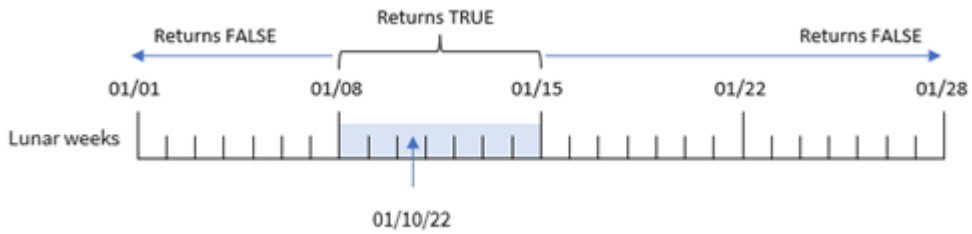
- `date`
- `in_lunar_week`

Tabla de resultados

date	in_lunar_week
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	-1

date	in_lunar_week
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

La función `inlunarweek()`, ejemplo básico



El campo `in_lunar_week` se crea en la instrucción load anterior utilizando la función `inlunarweek()` e insertando lo siguiente como argumentos de la función:

- El campo `date`
- Una fecha codificada para el 10 de enero como `base_date`
- Un `period_no` de 0

Como las semanas lunares comienzan el 1 de enero, el 10 de enero caería en la semana lunar que comienza el 8 de enero y finaliza el 14 de enero. Por lo tanto, cualquier transacción que ocurra entre esas dos fechas en enero devolvería un valor booleano de TRUE. Esto se valida en la tabla de resultados.

Ejemplo 2: period_no

Ejemplos y resultados:

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- El campo de fecha se ha proporcionado en el formato de la variable del sistema `DateFormat` (MM/DD/AAAA).

Sin embargo, en este ejemplo, la tarea es crear un campo, `2_lunar_weeks_later`, que determine si las transacciones se realizaron o no dos semanas lunares después del 10 de enero.

Script de carga

```
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
];
```

Resultados

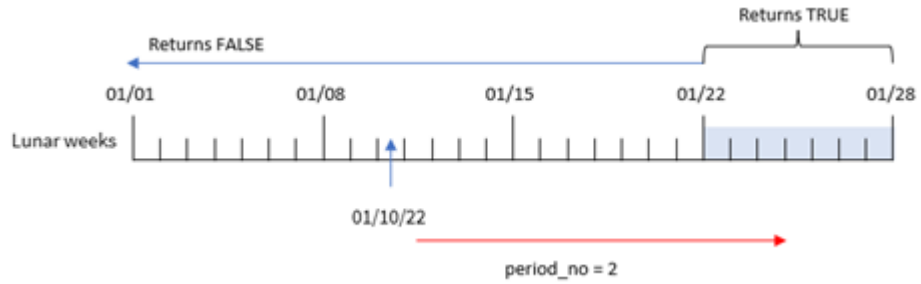
Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `2_lunar_weeks_later`

Tabla de resultados

date	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

La función `inlunarweek()`, ejemplo de period_no



En este caso, debido a que se utilizó un `period_no` de 2 como argumento del desplazamiento en la función `inlunarweek()`, la función define la semana que comienza el 22 de enero como la semana lunar con la que contrastar para validar las transacciones. Por lo tanto, cualquier transacción que ocurra entre el 22 de enero y el 28 de enero, devolverá un resultado booleano de `TRUE`.

Ejemplo 3: first_week_day

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga utiliza el mismo conjunto de datos y escenario que el primer ejemplo. Sin embargo, en el ejemplo, establecimos que las semanas lunares comiencen el 6 de enero.

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- Se utiliza la variable predefinida del sistema `DateFormat`: DD/MM/AAAA.
- Un argumento `first_week_day` de 5. Esto establece que las semanas lunares comiencen el 5 de enero.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

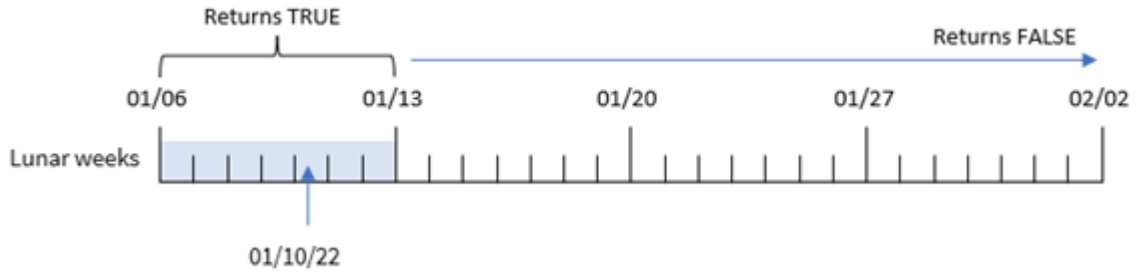
- `date`
- `in_lunar_week`

Tabla de resultados

date	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

date	in_lunar_week
1/22/2022	0
1/23/2022	0

La función `inlunarweek()`, ejemplo de first_week_day



En este caso, debido a que el argumento `first_week_date` de 5 se usa en la función `inlunarweek()`, se desplaza el inicio del calendario de la semana lunar al 6 de enero. Por lo tanto, el 10 de enero cae dentro de la semana lunar que comienza el 6 de enero y finaliza el 12 de enero. Cualquier transacción que se encuentre entre estas dos fechas devolverá un valor booleano de `TRUE`.

Ejemplo 4: Objeto gráfico

Script de carga y expresión de gráfico:

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- El campo de fecha se ha proporcionado en el formato de la variable del sistema `DateFormat` (MM/DD/AAAA).

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que determina si las transacciones tuvieron lugar en la misma semana lunar del 10 de enero se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Para calcular si una transacción tiene lugar en la semana lunar que contiene el 10 de enero, cree la siguiente medida:

```
= inlunarweek(date,'01/10/2022', 0)
```

Tabla de resultados

date	**=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

date	=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

Ejemplo 5: Escenario

Script de carga y expresión de gráfico:

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Products`.
- Información que consiste en la identificación del producto, la fecha de fabricación y el precio de fabricación.

Se ha identificado que, debido a un error en el equipo, los productos que se fabricaron en la semana lunar que englobaba al 12 de enero eran defectuosos. Al usuario final le gustaría tener un objeto gráfico que muestre, por nombre de semana lunar, el estado de los productos fabricados, si eran "defectuosos" o "sin defectos" y el coste de los productos fabricados en ese mes.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla.
2. Cree una dimensión para mostrar los nombres de los meses:
   `=lunarweekname(manufacture_date)`
3. Cree una medida para identificar cuáles de los productos son defectuosos y cuáles no tienen defectos utilizando la función `inlunarweek()`:
   `=if(only(inlunarweek(manufacture_date,makedate(2022,01,12),0)), 'Defective','Faultless')`
4. Cree una medida para sumar el `cost_price` de los productos:
   `=sum(cost_price)`
5. Establezca el **Formato numérico** de la medida en **Moneda**.
6. En **Aspecto**, deshabilite los **Totales**.

Tabla de resultados

lunarweekname (manufacture_date)	**=if(only(inlunarweek(manufacture_date,makedate (2022,01,12),0)), 'Defectuoso','Sin defectos')**	**=sum(cost_ price)**
2022/01	Sin defectos	$125.79
2022/02	Defectuoso	$316.38
2022/03	Sin defectos	$455.75
2022/04	Sin defectos	$146.09

La función `inlunarweek()` devuelve un valor booleano al evaluar las fechas de fabricación de cada uno de los productos. Para cualquier producto fabricado en la semana lunar que contiene el 10 de enero, la función `inlunarweek()` devuelve un valor booleano de `TRUE` y marca los productos como "defectuosos". Para cualquier producto que devuelva un valor de `FALSE`, y que por lo tanto no se haya fabricado en esa semana, marca los productos como "Sin defectos".

inlunarweektodate

Esta función halla si **timestamp** se encuentra dentro de la parte de la semana lunar hasta e incluido el último milisegundo de **base_date**. Las semanas lunares en Qlik Sense se definen contando el 1 de enero como el primer día de la semana y, aparte de la última semana del año,

contendrán exactamente siete días.

Sintaxis:

```
InLunarWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Tipo de datos que devuelve: Booleano

> *En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.*

Ejemplo de diagrama de la función `inlunarweektodate()`

La función `inlunarweektodate()` actúa como el punto final de la semana lunar. A diferencia de esta, la función `inlunarweek()` determina en qué semana lunar cae `base_date`. Por ejemplo, si la fecha `base_date` fuera el 5 de enero, cualquier fecha-hora entre el 1 y el 5 de enero arrojaría un resultado booleano de TRUE, mientras que las fechas del 6 y 7 de enero y posteriores arrojarían un resultado booleano de FALSE.

Argumentos

Argumento	Descripción
timestamp	La fecha que desea comparar con **base_date**.
base_date	La fecha que se utiliza para evaluar la semana lunar.
period_no	La semana lunar puede desplazarse mediante **period_no**; period_no es un entero, en el que el valor 0 indica la semana lunar que contiene a **base_date**. Los valores negativos en **period_no** indican semanas lunares precedentes y los valores positivos indican semanas lunares subsiguientes.
first_week_day	Un desplazamiento que puede ser mayor que o menor que cero. Esto cambia el comienzo del año por el número especificado de días y/o fracciones de un día.

Cuándo se utiliza

La función `inlunarweektodate()` devuelve un resultado booleano. Normalmente, este tipo de función se utiliza como condición en una expresión condicional. La función `inlunarweektodate()` se utilizará cuando el usuario quiera que el cálculo devuelva una agregación o un cálculo, dependiendo de si la fecha evaluada ocurrió durante un segmento particular de la semana en cuestión.

Por ejemplo, la función `inlunarweektodate()` se puede utilizar para identificar todos los equipos fabricados en una determinada semana hasta e incluida una fecha en particular.

Ejemplos de funciones

Ejemplo	Resultado
`inlunarweektodate ('01/12/2013', '01/13/2013', 0)`	Devuelve `TRUE`, porque el valor de `timestamp`, 01/12/2013, cae en la parte de la semana del 01/08/2013 al 01/13/2013.
`inlunarweektodate ('01/12/2013', '01/11/2013', 0)`	Devuelve `FALSE`, porque el valor de `timestamp`es posterior al valor de `base_date`, aunque las dos fechas están en la misma semana lunar anterior a 01/12/2012.
`inlunarweektodate ('01/12/2006', '01/05/2006', 1)`	Devuelve `TRUE`. Especificar un valor de 1 para `period_no` desplaza a `base_date` una semana hacia delante, de modo que el valor de `timestamp` cae en la parte de la semana lunar.

La función `inlunarweektodate()` se utiliza a menudo en combinación con las siguientes funciones:

Funciones relacionadas

Función	Interacción
lunarweekname (page 876)	Esta función sirve para indicar el número de semana lunar del año en el que ocurre una fecha de entrada.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones del mes de enero, el cual se carga en una tabla denominada `Transactions`. Se utiliza la variable predefinida del sistema `DateFormat`: MM/DD/AAAA.
- Cree un campo, in_lunar_week_to_date, que determine qué transacciones se realizaron en la semana lunar hasta la fecha del 10 de enero.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

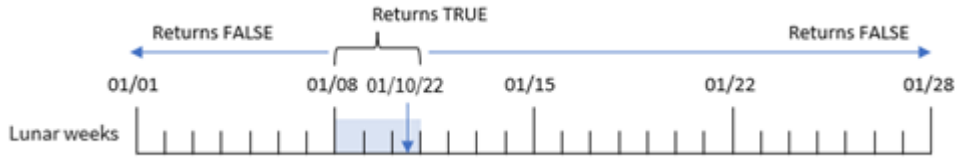
- `date`
- `in_lunar_week_to_date`

Tabla de resultados

date	in_lunar_week_to_date
1/1/2022	0
1/4/2022	0

date	in_lunar_week_to_date
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

La función `inlunarweektodate()`, sin argumentos adicionales



El campo `in_lunar_week_to_date` se crea en la sentencia load anterior utilizando la función `inlunarweektodate()` e introduciendo el campo `date`, una marca de tiempo codificada para el 10 de enero como nuestra `base_date` y un desplazamiento de 0 como argumentos de la función.

Debido a que las semanas lunares comienzan el 1 de enero, el 10 de enero caería en la semana lunar que comienza el 8 de enero; y debido a que estamos usando la función `inlunarweektodate()`, esa semana lunar terminaría el día 10. Por lo tanto, cualquier transacción que ocurra entre esas dos fechas en enero devolvería un valor booleano de `TRUE`. Esto se valida en la tabla de resultados.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo. Sin embargo, en este ejemplo, la tarea es crear un campo, `2_lunar_weeks_later`, que determine si las transacciones se realizaron o no dos semanas después de la semana lunar hasta la fecha del 1 de enero.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

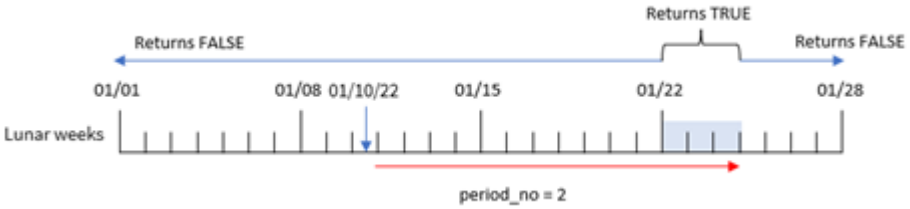
- `date`
- `2_lunar_weeks_later`

Tabla de resultados

date	2_lunar_weeks_later
1/1/2022	0
1/4/2022	0
1/10/2022	0
1/11/2022	0

date	2_lunar_weeks_later
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	-1
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

La función `inlunarweektodate()`, ejemplo de period_no



En este caso, la función `inlunarweektodate()` determina que la semana lunar hasta el 10 de enero equivale a tres días (8, 9 y 10 de enero). Dado que se utilizó un `period_no` de 2 como argumento del desplazamiento, esta semana lunar se desplaza 14 días. Por lo tanto, esto define que la semana lunar de tres días incluye el 22, 23 y 24 de enero. Cualquier transacción que tenga lugar entre el 22 y el 24 de enero devolverá un resultado booleano de `TRUE`.

Ejemplo 3: first_week_day

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- Se utiliza la variable predefinida del sistema `DateFormat`: DD/MM/AAAA.

- Un argumento `first_week_date` de 3. Esto establece que las semanas lunares comiencen el 3 de enero.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

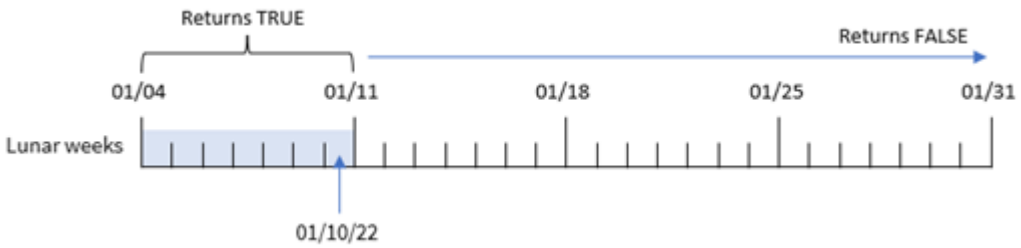
- `date`
- `in_lunar_week_to_date`

Tabla de resultados

date	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

date	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

La función `inlunarweektodate()`, ejemplo de first_week_day



En este caso, como se usa un argumento `the first_week_date` de 3 en la función `inlunarweek()`, la primera semana lunar será del 3 al 10 de enero. Debido a que el 10 de enero también es la `base_date`, cualquier transacción que se encuentre entre estas dos fechas devolverá un valor booleano de `TRUE`.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación, en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que determina si las transacciones tuvieron lugar en la semana lunar hasta el 10 de enero se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Cree la siguiente medida:

```
=inlunarweektodate(date,'01/10/2022', 0)
```

Tabla de resultados

date	=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

date	=inlunarweektodate(date,'01/10/2022', 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

La función `inlunarweektodate()`, ejemplo de objeto gráfico

La medida `in_lunar_week_to_date` se crea en el objeto gráfico utilizando la función `inlunarweektodate()` e introduciendo el campo de fecha, una marca de tiempo codificada para el 10 de enero como `base_date` y un desplazamiento de 0 como argumentos de la función.

Como las semanas lunares comienzan el 1 de enero, el 10 de enero caería en la semana lunar que comienza el 8 de enero. Además, como estamos usando la función `inlunarweektodate()`, esa semana lunar terminaría el día 10. Por lo tanto, cualquier transacción que ocurra entre esas dos fechas en enero devolvería un valor booleano de `TRUE`. Esto se valida en la tabla de resultados.

Ejemplo 5: Escenario

Script de carga y expresiones de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Products`.
- Información que consiste en la identificación del producto, la fecha de fabricación y el precio de fabricación.

Se ha identificado que, debido a un error en el equipo, los productos que se fabricaron en la semana lunar del 12 de enero eran defectuosos. El problema se resolvió el 13 de enero. Al usuario final le gustaría tener un objeto gráfico que muestre, por semana, el estado de los productos fabricados, si son "defectuosos" o "sin defectos", y el costo de los productos fabricados en esa semana.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla.
2. Cree una dimensión para mostrar los nombres de las semanas:
   `=weekname(manufacture_date)`
3. A continuación, cree una dimensión que utilice la función `inlunarweektodate()` para identificar cuáles de los productos son defectuosos y cuáles son sin defectos:
   `=if(inlunarweektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
4. Cree una medida para sumar el `cost_price` de los productos:
   `=sum(cost_price)`
5. Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

=lunarweekname (manufacture_date)	**=if(InLunarWeekToDate(manufacture_date,makedate (2022,01,12),0),'Defectuoso','Sin defectos')**	**=Sum(cost_ price)**
2022/01	Sin defectos	$142.67

=lunarweekname (manufacture_date)	=if(InLunarWeekToDate(manufacture_date,makedate (2022,01,12),0),'Defectuoso','Sin defectos')	=Sum(cost_ price)
2022/02	Defectuoso	$320.88
2022/02	Sin defectos	$141.82
2022/03	Sin defectos	$214.64
2022/04	Sin defectos	$147.46
2022/05	Sin defectos	$248.12

La función `inlunarweektodate()` devuelve un valor booleano al evaluar las fechas de fabricación de cada uno de los productos. Para aquellos que devuelven un valor booleano de `TRUE`, marca los productos como `'Defective'`. Para cualquier producto que devuelva un valor de `FALSE`, y por tanto no realizados en la semana lunar hasta el 12 de enero inclusive, marca los productos como Sin defectos (`'Faultless'`).

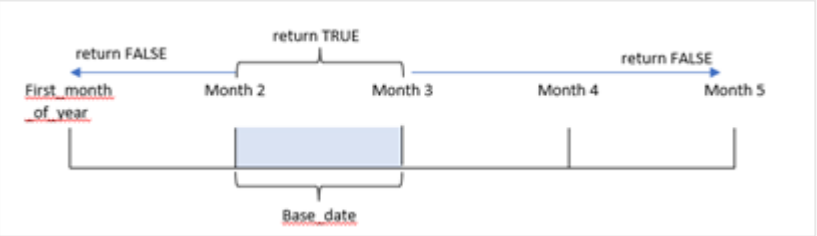
inmonth

Esta función devuelve True si **timestamp** se encuentra dentro del mes que contiene a **base_ date**.

Sintaxis:

```
InMonth (timestamp, base_date, period_no)
```

Diagrama de la función `indaytotime`.



En otras palabras, la función `inmonth()` determina si un conjunto de fechas cae dentro de este mes y devuelve un valor booleano basado en `base_date` que identifica el mes.

Cuándo se utiliza

La función `inmonth()` devuelve un resultado booleano. Normalmente, este tipo de función se utilizará como condición en una `if expression`. Esto devuelve una agregación o cálculo dependiendo de si una fecha ocurrió en el mes, incluida la fecha en cuestión.

Por ejemplo, la función `inmonth()` se puede utilizar para identificar todos los equipos fabricados en un mes determinado.

Tipo de datos que devuelve: Booleano

En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.

Argumentos

Argumento	Descripción
fecha-hora	La fecha que desea comparar con `base_date`.
base_date	La fecha que se utiliza para evaluar el mes. Es importante tener en cuenta que la `base_date` puede ser cualquier día de un mes.
period_no	El mes puede desplazarse mediante `period_no`. `period_no` es un entero, en el que el valor 0 indica el mes que contiene a `base_date`. Los valores negativos en `period_no` indican meses precedentes y los valores positivos indican meses subsiguientes.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
`inmonth ('25/01/2013', '01/01/2013', 0)`	Devuelve Verdadero
`inmonth('25/01/2013', '23/04/2013', 0)`	Devuelve Falso
`inmonth ('25/01/2013', '01/01/2013', -1)`	Devuelve Falso
`inmonth ('25/12/2012', '17/01/2013', -1)`	Devuelve Verdadero

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones durante la primera mitad de 2022.
- Un load precedente con una variable adicional, "in_month", que determina si las transacciones se realizaron en abril.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- in_month

Ejemplos de funciones

date	in_month
1/10/2022	0
1/14/2022	0
1/20/2022	0
1/22/2022	0

date	in_month
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
5/22/2022	0
6/19/2022	0
6/22/2022	0

El campo "in_month" se crea en la instrucción load anterior utilizando la función `inmonth()` e introduciendo el campo de fecha, una marca de tiempo codificada para el 1 de abril como nuestra `base_date` y un `period_no` de 0 como argumentos de la función.

La `base_date` identifica el mes que devolverá un resultado booleano de TRUE Por lo tanto, todas las transacciones que ocurrieron en abril devuelven TRUE, que se valida en la tabla de resultados.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, creará un campo, ''2_months_prior", que determina si las transacciones se realizaron dos meses antes de abril.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `2_months_prior`

Ejemplos de funciones

date	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
5/22/2022	0
6/19/2022	0
6/22/2022	0

Usar -2 como el argumento de `period_no` en la función `inmonth()` cambia el mes definido por el argumento `base_date` a dos meses antes. En este ejemplo cambia el mes definido de abril a febrero.

Por lo tanto, cualquier transacción que tenga lugar en febrero devolverá un resultado booleano de TRUE.

Ejemplo 3 : Objeto gráfico

Script de carga y expresión de gráfico

General

Se utilizan el mismo conjunto de datos y el mismo escenario que en los ejemplos anteriores.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que determina si las transacciones se realizaron en abril se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Objeto gráfico

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

```
date
```

Para calcular si una transacción tiene lugar en abril, cree la siguiente medida:

```
=inmonth(date,'04/01/2022', 0)
```

Resultados

	Ejemplos de funciones
date	**=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
5/20/2022	0
5/22/2022	0
6/19/2022	0
6/22/2022	0

Ejemplo 4: Escenario

Script de carga y resultados

Vista general

En este ejemplo, un conjunto de datos se carga en una tabla denominada "`Products`". La tabla contiene los siguientes campos:

- ID de producto
- Fecha de fabricación
- Precio de coste

Debido a un error del equipo, los productos que se fabricaron en el mes de julio de 2022 resultaron defectuosos. El problema se resolvió el 27 de julio de 2022.

Al usuario final le gustaría tener un gráfico que muestre, por mes, el estado de los productos que se fabricaron como "defectuosos" (booleano TRUE) o "sin defectos" (booleano FALSE) y el costo de los productos fabricados en ese mes.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

```
=monthname(manufacture_date)
```

Cree las siguientes medidas:

- `=sum(cost_price)`
- `=if(only(inmonth(manufacture_date,makedate(2022,07,01),0)),'Defective','Faultless')`

1. Establezca el **Formato numérico** de la medida en **Moneda**.
2. En **Aspecto**, deshabilite los **Totales**.

Tabla de resultados

monthname (manufacture_date)	=if(only(inmonth(manufacture_date,makedate (2022,07,01),0)),'Defectuosos','Sin defectos')	=sum(cost_ price)
Ene 2022	Sin defectos	$54.40
Feb 2022	Sin defectos	$145.69
Mar 2022	Sin defectos	$53.80

monthname (manufacture_date)	=if(only(inmonth(manufacture_date,makedate (2022,07,01),0)),'Defectuosos','Sin defectos')	=sum(cost_ price)
Abr 2022	Sin defectos	$82.06
May 2022	Sin defectos	$127.60
Jun 2022	Sin defectos	$141.82
Jul 2022	Defectuoso	$214.64
Ago 2022	Sin defectos	$147.46
Sep 2022	Sin defectos	$84.21
Oct 2022	Sin defectos	$163.91

La función `inmonth()` devuelve un valor booleano al evaluar las fechas de fabricación de cada uno de los productos. Para cualquier producto fabricado en julio de 2022, la función `inmonth()` devuelve un valor booleano de TRUE y marca los productos como "defectuosos". Para cualquier producto que devuelva un valor de FALSE, y que por lo tanto no se haya fabricado en julio, marca los productos como "Sin defectos".

inmonths

Esta función determina si una fecha-hora se encuentra dentro del mismo periodo mensual, bimensual, trimestral, cuatrimestre o semestral que fecha base. También es posible hallar si la fecha-hora cae dentro de un periodo anterior o posterior.

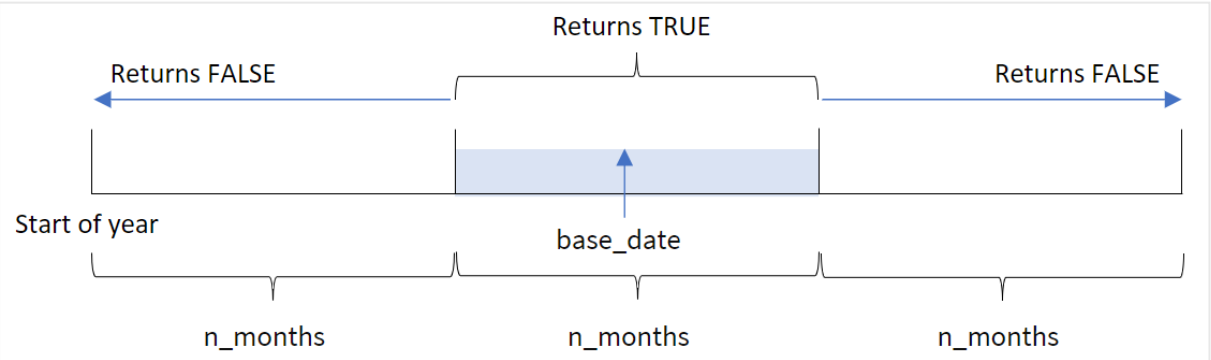
Sintaxis:

```
InMonths(n_months, timestamp, base_date, period_no [, first_month_of_year])
```

Tipo de datos que devuelve: Booleano

En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.

Diagrama de la función `inmonths()`



La función `inmonths()` divide el año en segmentos basándose en el argumento `n_months` proporcionado. Luego determina si cada marca de tiempo evaluada cae en el mismo segmento que el argumento de `base_date`. Sin embargo, si se proporciona un argumento `period_no`, la función determina si las marcas de tiempo caen en un período anterior o posterior a `base_date`.

Los siguientes segmentos del año están disponibles en la función `n_month` como argumentos.

Argumentos de `n_month`

Periodo	Número de meses
mes	1
bimestre	2
trimestre	3
cuatrimestre	4
semestre	6

Cuándo se utiliza

La función `inmonths()` devuelve un resultado booleano. Normalmente, este tipo de función se utilizará como condición en una `if expression`. Mediante el uso de la función `inmonths()`, puede seleccionar el período que desea evaluar. Por ejemplo, permitir que el usuario identifique los productos fabricados en el mes, trimestre o semestre de un período determinado.

Tipo de datos que devuelve: Booleano

En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.

Argumentos

Argumento	Descripción
n_months	El número de meses que define el periodo. Un entero o expresión que devuelve un entero que debe ser uno de los siguientes: 1 (equivalente a la función inmonth()), 2 (bimestral), 3 (equivalente a la función inquarter()), 4 (cuatrimestral) o 6 (medio año).
timestamp	La fecha que desea comparar con **base_date**.
base_date	La fecha que se utiliza para evaluar el periodo.
period_no	El período se puede desplazar mediante **period_no**, un entero o una expresión que devuelve un entero, donde el valor 0 indica el período que contiene a **base_date**. Los valores negativos en **period_no** indican períodos precedentes y los valores positivos indican períodos subsiguientes.
first_month_of_year	Si desea trabajar con años (fiscales) que no comiencen en enero, indique un valor entre 2 y 12 en **first_month_of_year**.

Puede utilizar los siguientes valores para establecer el primer mes del año en el argumento `first_month_of_year`:

first_month_of_year values

Month	Valor
Febrero	2
Marzo	3
Abril	4
May	5
Junio	6
Julio	7
Agosto	8
Septiembre	9
Octubre	10
Noviembre	11
Diciembre	12

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
`inmonths(4, '01/25/2013', '04/25/2013', 0)`	Devuelve TRUE. Porque el valor de la marca de tiempo 25/01/2013 se encuentra dentro del período de cuatro meses: 01/01/2013 a 30/04/2013, en el que se encuentra el valor de `base_date`, 25/04/2013.
`inmonths(4, '05/25/2013', '04/25/2013', 0)`	Devuelve FALSE. Porque el 25/05/2013 está fuera del mismo periodo que el ejemplo anterior.

Ejemplo	Resultado
`inmonths(4, '11/25/2012', '02/01/2013', -1)`	Devuelve TRUE. Porque el valor de `period_no`, `-1`, desplaza el período de búsqueda hacia atrás un período de cuatro meses (el valor de n-meses), lo que hace que el período de búsqueda sea del 01/09/2012 al 31/12/2012.
`inmonths(4, '05/25/2006', '03/01/2006', 0, 3)`	Devuelve TRUE. Porque el valor de `first_month_of_year` está establecido en `3`, lo que hace que el período de búsqueda sea del 01/03/2006 al 30/07/2006 en lugar de 01/01/2006 al 30/04/2006.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022 se carga en una tabla llamada "`Transactions`".
- Una instrucción load anterior con una variable adicional, "`in_months`'", que determina qué transacciones tuvieron lugar en el mismo trimestre que el 15 de mayo de 2022.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

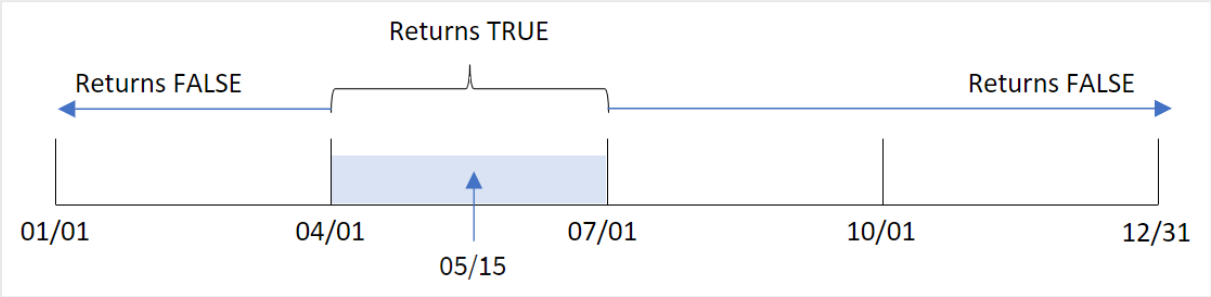
- `date`
- `in_months`

Tabla de resultados

date	in_months
2/19/2022	0
3/7/2022	0
3/30/2022	0
4/5/2022	-1
4/16/2022	-1
5/1/2022	-1
5/7/2022	-1
5/22/2022	-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

El campo "in_months" se crea en la instrucción load anterior, mediante el uso de la función inmonths(). El primer argumento proporcionado es 3, que divide el año en segmentos trimestrales. El segundo argumento identifica qué campo se está evaluando, el campo de fecha en este ejemplo. El tercer argumento es una fecha codificada para el 15 de mayo, que constituye la base_date y un period_no de 0 es el argumento final.

Diagrama de la función inmonths() con segmentos trimestrales



Mayo cae en el segundo trimestre del año. Por lo tanto, cualquier transacción que ocurra entre el 1 de abril y el 30 de junio devolverá un resultado booleano de TRUE. Esto se valida en la tabla de resultados.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022 se carga en una tabla llamada "Transactions".
- Un load precedente con una variable adicional, "previous_quarter", que determina si las transacciones se realizaron en el trimestre anterior al 15 de mayo de 2022.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `previous_quarter`

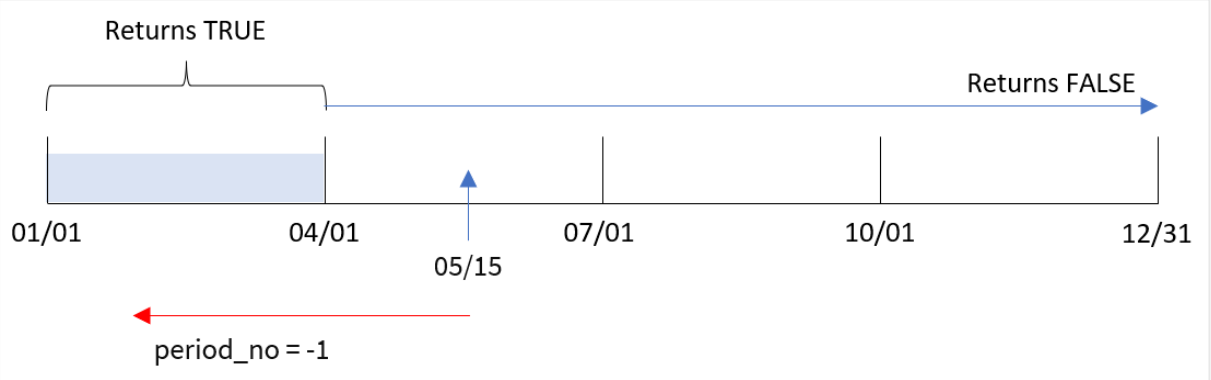
Tabla de resultados

date	previous quarter
2/19/2022	-1
3/7/2022	-1
3/30/2022	-1
4/5/2022	0
4/16/2022	0
5/1/2022	0
5/7/2022	0
5/22/2022	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

date	previous quarter
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

La función evalúa si las transacciones ocurrieron en el primer trimestre del año utilizando `-1` como el argumento de `period_no` en la función `inmonths()`. El 15 de mayo es la `base_date` y cae en el segundo trimestre del año (abril-junio).

Diagrama de la función `inmonths()` con segmentos trimestrales y el `period_no` establecido en `-1`



Por lo tanto, cualquier transacción que ocurra entre enero y marzo devolverá un resultado booleano de TRUE.

Ejemplo 3: first_month_of_year

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022 se carga en una tabla llamada "`Transactions`".
- Una instrucción load anterior con una variable adicional, "`in_months`'", que determina qué transacciones tuvieron lugar en el mismo trimestre que el 15 de mayo de 2022.

En este ejemplo, la política de la organización es que marzo sea el primer mes del año fiscal.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `in_months`

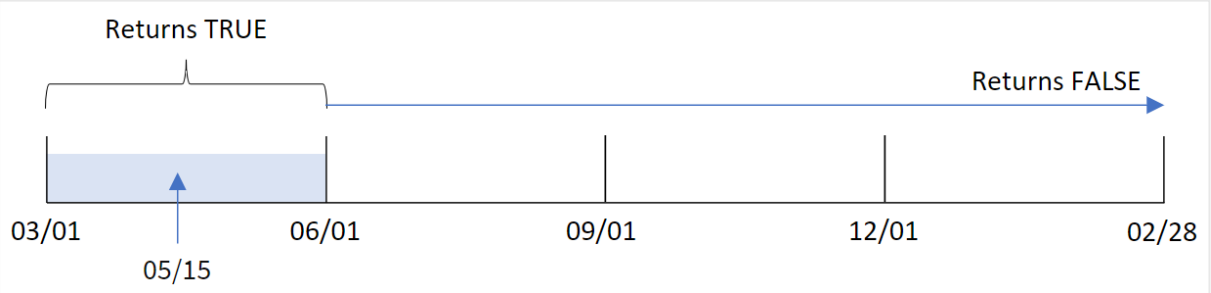
Tabla de resultados

date	in_months
2/19/2022	0
3/7/2022	-1
3/30/2022	-1
4/5/2022	-1

date	in_months
4/16/2022	-1
5/1/2022	-1
5/7/2022	-1
5/22/2022	-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
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Usando `3` como el argumento de `first_month_of_year` en la función `inmonths()`, la función comienza el año el 1 de marzo. Después la función `inmonths()` divide el año en trimestres. Mar-May, Jun-Ago, Sep-Nov, Dic-Feb. Por lo tanto, el 15 de mayo cae en el primer trimestre del año (marzo-mayo).

Diagrama de la función `inmonths()` con marzo fijado como primer mes del año.



Cualquier transacción que ocurra en estos meses devolverá un resultado booleano de TRUE.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que determina si las transacciones se realizaron en el mismo trimestre que el 15 de mayo de 2022 se crea como una medida en un objeto gráfico de la app.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

- `date`

Para calcular si las transacciones se realizaron en el mismo trimestre que el 15 de mayo, cree la siguiente medida:

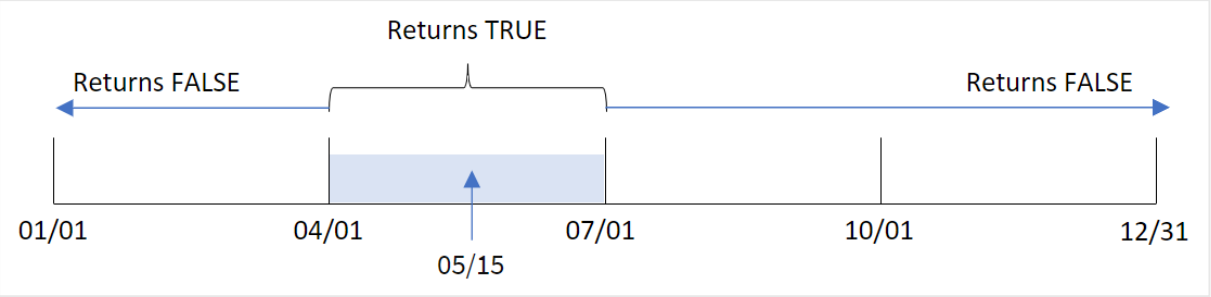
```
=inmonths(3,date,'05/15/2022', 0)
```

Tabla de resultados

date	=inmonths(3,date,'05/15/2022', 0)
2/19/2022	0
3/7/2022	0
3/30/2022	0
4/5/2022	-1
4/16/2022	-1
5/1/2022	-1
5/7/2022	-1
5/22/2022	-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

El campo "in_months" se crea en el gráfico mediante la función `inmonths()`. El primer argumento proporcionado es `3`, que divide el año en segmentos trimestrales. El segundo argumento identifica qué campo se está evaluando, el campo de fecha en este ejemplo. El tercer argumento es una fecha codificada para el 15 de mayo, que constituye la `base_date` y un `period_no` de `0` es el argumento final.

Diagrama de la función `inmonths()` con segmentos trimestrales



Mayo cae en el segundo trimestre del año. Por lo tanto, cualquier transacción que ocurra entre el 1 de abril y el 30 de junio devolverá un resultado booleano de TRUE. Esto se valida en la tabla de resultados.

Ejemplo 5: Escenario

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada "`Products`".
- La tabla contiene los siguientes campos:
    - product ID
    - product type
    - manufacture date
    - cost price

Al usuario final le gustaría tener un objeto gráfico que muestre, por tipo de producto, el coste de producción de los productos fabricados en el primer segmento de 2021. Al usuario le gustaría poder definir la longitud de este segmento.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja.

Al comienzo del script de carga se crea una variable, `vPeriod`, que se vinculará al control de entrada de la variable.

Haga lo siguiente:

1. En el panel de activos, haga clic en **Objetos personalizados**.
2. Seleccione **Qlik Dashboard bundle**, cree un objeto **Entrada de variables**.
3. Escriba un título para el objeto gráfico.
4. En **Variable**, seleccione **vPeriod** como nombre y configure el objeto para que se muestre como un **Menú desplegable**.
5. En **Valores**, haga clic en **Valores dinámicos**. Escriba lo siguiente:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. Agregue una nueva tabla a la hoja.
7. En el panel de propiedades, en **Datos**, añada `product_type` como dimensión.
8. Añada la siguiente expresión como medida:
   `=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))`
9. Configure el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

product_type	**=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))**
producto A	$88.27
producto B	$37.23
producto C	$17.17
producto D	$0.00

La función `inmonths()` utiliza la entrada del usuario como argumento para definir el tamaño del segmento inicial del año. La función introduce la fecha de fabricación de cada uno de los productos como segundo argumento de la función `inmonths()`. Al usar el 1 de enero como tercer argumento de la función `inmonths()`, los productos con fechas de fabricación que caen en el segmento de apertura del año devolverán un valor booleano de TRUE y, por lo tanto, la función de suma agregará los costes de fabricación de esos productos.

inmonthstodate

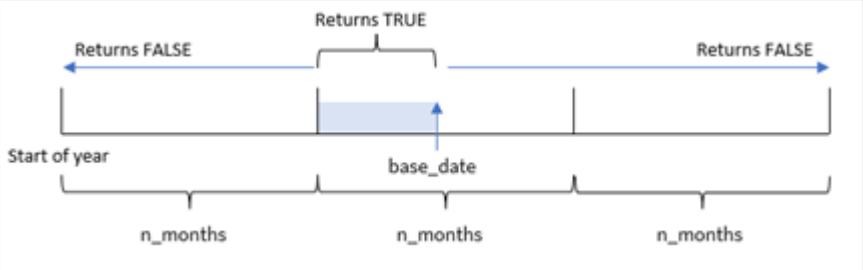
Esta función determina si una fecha-hora se encuentra dentro de la parte de un período mensual, bimensual, trimestral, cuatrimestral o semestral hasta e incluyendo el último milisegundo de `base_date`. También es posible hallar si la fecha-hora cae dentro de un periodo anterior o posterior.

Sintaxis:

```
InMonths (n_months, timestamp, base_date, period_no[, first_month_of_year ])
```

Tipo de datos que devuelve: Booleano

Diagrama de la función `inmonthstodate`.



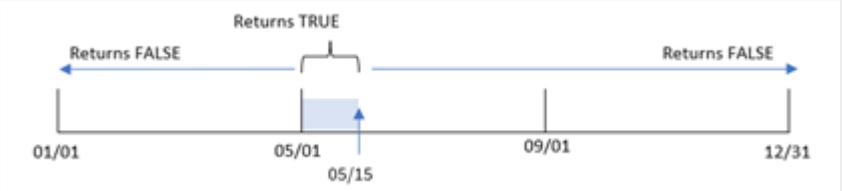
Argumentos

Argumento	Descripción
n_months	El número de meses que define el periodo. Un entero o expresión que devuelve un entero que debe ser uno de los siguientes: 1 (equivalente a la función inmonth()), 2 (bimestral), 3 (equivalente a la función inquarter()), 4 (cuatrimestral) o 6 (medio año).
timestamp	La fecha que desea comparar con **base_date**.
base_date	La fecha que se utiliza para evaluar el periodo.
period_no	El período se puede desplazar mediante **period_no**, un entero o una expresión que devuelve un entero, donde el valor 0 indica el período que contiene a **base_date**. Los valores negativos en **period_no** indican períodos precedentes y los valores positivos indican períodos subsiguientes.
first_month_of_year	Si desea trabajar con años (fiscales) que no comiencen en enero, indique un valor entre 2 y 12 en **first_month_of_year**.

En la función `inmonthstodate()`, la `base_date` actúa como el punto final del segmento de año en particular del que forma parte.

Por ejemplo, si el año se dividió en segmentos cuatrimestrales y la `base_date` era el 15 de mayo, cualquier marca de tiempo entre el comienzo de enero y el final de abril devolvería un resultado booleano de FALSE. Las fechas entre el 1 y el 15 de mayo devolverían TRUE. El resto del año devolvería FALSE.

Diagrama del rango de resultados booleanos de la función `inmonthstodate`.



Los siguientes segmentos del año están disponibles en la función `n_month` como argumentos.

Argumentos de `n_month`

Periodo	Número de meses
mes	1
bimestre	2
trimestre	3
cuatrimestre	4
semestre	6

Cuándo se utiliza

La función `inmonthstodate()` devuelve un resultado booleano. Normalmente, este tipo de función se utiliza como condición en una expresión condicional (`if expression`). Mediante el uso de la función `inmonthstodate()`, puede seleccionar el período que desea evaluar. Por ejemplo, proporcionar una variable de entrada que permita al usuario identificar los productos fabricados en el mes, trimestre o semestre de un período, hasta una fecha determinada.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
`inmonthstodate(4, '01/25/2013', '04/25/2013', 0)`	Devuelve True, porque el valor de timestamp, 01/25/2013, cae dentro del período de cuatro meses, del 01/01/2013 hasta el final de 04/25/2013, en el que se encuentra el valor de base_date, 04/25/2013.
`inmonthstodate(4, '04/26/2013', '04/25/2006', 0)`	Devuelve False, porque 04/26/2013 está fuera del mismo período que el ejemplo anterior.
`inmonthstodate(4, '09/25/2005', '02/01/2006', -1)`	Devuelve True, porque el valor de period_no, -1, desplaza el período de búsqueda un período de cuatro meses atrás (el valor de n-months), lo que hace que el período de búsqueda sea del 01/09/2005 al 02/01/2006.
`inmonthstodate(4, '04/25/2006', '06/01/2006', 0, 3)`	Devuelve True, porque el valor de first_month_of_year está configurado en 3, lo que hace que el período de búsqueda sea del 03/01/2006 al 06/01/2006, en lugar del 05/01/2006 al 06/01/2006.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene una serie de transacciones de 2022 que se carga en una tabla llamada "`Transactions`".
- Un campo de fecha en el formato (`MM/DD/YYYY`) de la variable del sistema (`DateFormat`).
- Una instrucción load anterior que contiene lo siguiente:
    - La función `inmonthstodate()` que está configurada como el campo "`in_months_to_date`". Esto determina qué transacciones se realizaron en el trimestre hasta el 15 de mayo de 2022.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `in_months_to_date`

Tabla de resultados

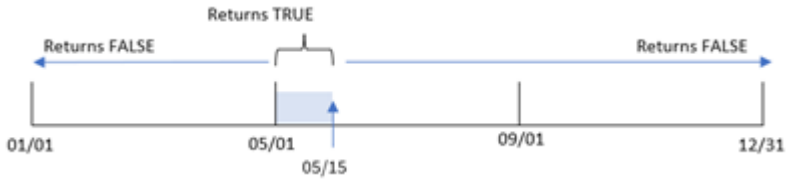
date	in_months_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0

date	in_months_to_date
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

El campo "`in_months_to_date`" se crea en la instrucción load anterior mediante el uso de la función `inmonthstodate()`.

El primer argumento proporcionado es `3`, que divide el año en segmentos trimestrales. El segundo argumento identifica qué campo se está evaluando. El tercer argumento es una fecha codificada para el 15 de mayo, que es la `base_date` que define el límite final del segmento. Un `period_no` de `0` es el argumento final.

Diagrama de la función `inmonthstodate`, sin argumentos adicionales.



Cualquier transacción que ocurra entre el 1 de abril y el 15 de mayo devuelve un resultado booleano de TRUE. Las fechas de transacciones fuera de ese período devuelven FALSE.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, la tarea es crear un campo, "`previous_qtr_to_date`", que determine si las transacciones se realizaron un trimestre antes del 15 de mayo.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `previous_qtr_to_date`

Tabla de resultados

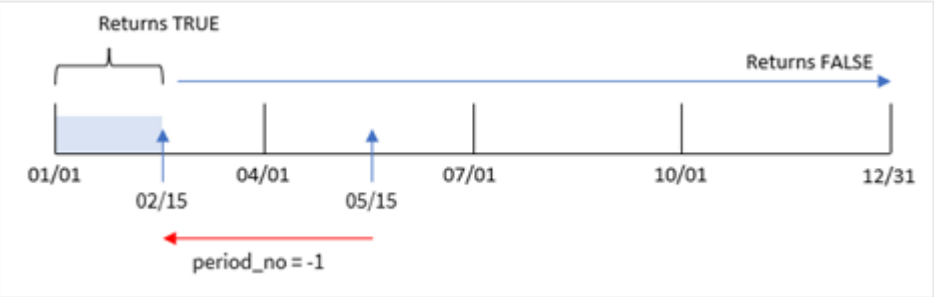
date	previous_qtr_to_date
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0

date	previous_qtr_to_date
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

Utilizando `-1` como el argumento de `period_no` en la función `inmonthstodate()`, la función desplaza los límites del segmento del año de comparación en un trimestre.

El 15 de mayo cae en el segundo trimestre del año y por lo tanto el segmento inicialmente equivale al período entre el 1 de abril y el 15 de mayo. El argumento de `period_no` desplaza este segmento en tres meses negativos. Los límites de fecha pasan a ser del 1 de enero al 15 de febrero.

Diagrama de la función `inmonthstodate` con un period_no de -1.



Por lo tanto, cualquier transacción que ocurra entre el 1 de enero y el 15 de febrero devolverá un resultado booleano de TRUE.

Ejemplo 3: first_month_of_year

Script de carga y resultados

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

En este ejemplo, la política de la organización es que marzo sea el primer mes del año fiscal.

Cree un campo, "in_months_to_date", que determine qué transacciones se realizaron en el mismo trimestre hasta el 15 de mayo de 2022.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- in_months_to_date

Tabla de resultados

date	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
9/26/2022	0
10/14/2022	0
10/29/2022	0

Usando `3` como el argumento de `first_month_of_year` en la función `inmonthstodate()`, la función comienza el año el 1 de marzo y después divide el año en trimestres según lo proporcionado en el primer argumento. Por lo tanto, los segmentos del cuatrimestre son:

- Mar-May
- Jun-Ago
- Sep-Nov
- Dic-Feb

La `base_date` del 15 de mayo segmenta a continuación el trimestre de marzo a mayo estableciendo su límite final en el 15 de mayo.

Diagrama de la función `inmonthstodate` con marzo fijado como primer mes del año.

Por lo tanto, cualquier transacción que ocurra entre el 1 de marzo y el 15 de mayo arrojará un resultado booleano de TRUE, y las transacciones que tengan fechas fuera de estos límites arrojarán un valor de FALSE.

Ejemplo 4: ejemplo de gráfico

Script de carga y expresión de gráfico

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

En este ejemplo el conjunto de datos está sin modificar y cargado en la app. La tarea es crear un cálculo que determine si las transacciones se realizaron en el mismo trimestre que el 15 de mayo como medida en un gráfico de la app.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

```
date
```

Para calcular si las transacciones se realizaron en el mismo trimestre que el 15 de mayo, cree la siguiente medida:

```
=inmonthstodate(3,date,'05/15/2022', 0)
```

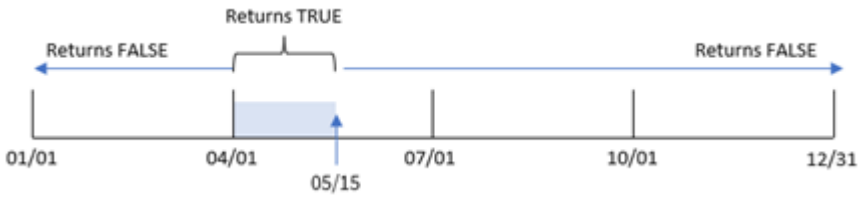
Tabla de resultados

date	=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
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

La medida "`‘in_months_to_date`" se crea en el gráfico usando la función `inmonthstodate()`.

El primer argumento proporcionado es 3, que divide el año en segmentos trimestrales. El segundo argumento identifica qué campo se está evaluando. El tercer argumento es una fecha codificada del 15 de mayo, que es la base_date que define el límite final del segmento. Un period_no de 0 es el argumento final.

Diagrama de la función inmonthstodate con segmentos trimestrales.



Cualquier transacción que ocurra entre el 1 de abril y el 15 de mayo devolverá un resultado booleano de TRUE. Las fechas de transacción fuera de ese segmento devolverán FALSE.

Ejemplo 5: Escenario

Script de carga y resultados

Vista general

En este ejemplo, un conjunto de datos se carga en una tabla denominada "Sales". La tabla contiene los siguientes campos:

- ID de producto
- Tipo de producto
- Fecha de venta
- Precio de venta

Al usuario final le gustaría tener un gráfico que muestre, por tipo de producto, las ventas de productos que se produjeron en el período anterior al 24 de diciembre de 2022. Al usuario le gustaría poder definir la duración de ese período.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja.

Al comienzo del script de carga se crea una variable, `vPeriod`, que se vinculará al control de entrada de la variable.

Haga lo siguiente:

1. En el panel de activos, haga clic en **Objetos personalizados**.
2. Seleccione **Qlik Dashboard bundle** y agregue una **Entrada de variable** a su hoja.
3. Escriba un título para el gráfico.
4. En **Variable**, seleccione **vPeriod** como nombre y configure el objeto para que se muestre como un **Menú desplegable**.
5. En **Valores**, haga clic en **Valores dinámicos**. Escriba lo siguiente:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. Agregue una nueva tabla a la hoja.
7. En el panel de propiedades, en **Datos**, añada `product_type` como dimensión.
8. Añada la siguiente expresión como medida:
   `=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))`
9. Configure el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

product_type	**=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))**
producto A	$186.48
producto B	$190.52
producto C	$220.43
producto D	$261.46

La función `inmonthstodate()` utiliza la entrada del usuario como argumento para definir el tamaño del segmento inicial del año.

La función introduce la fecha de venta de cada uno de los productos como segundo argumento de la función `inmonthstodate()`. Al usar el 24 de diciembre como tercer argumento de la función `inmonthstodate()`, los productos con fechas de venta que ocurren en el período definido hasta el 24 de diciembre inclusive, devuelven un valor booleano de TRUE. La función sum suma las ventas de estos productos.

inmonthtodate

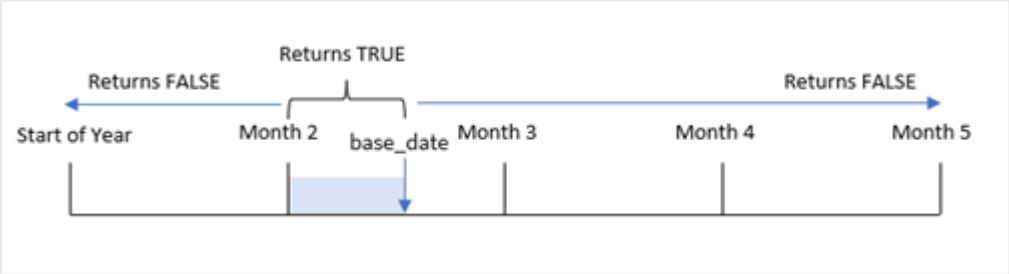
Devuelve True si **date** se encuentra dentro de la parte del mes que contiene a **basedate** hasta e incluido el último milisegundo de **basedate**.

Sintaxis:

```
InMonthToDate (timestamp, base_date, period_no)
```

Tipo de datos que devuelve: Booleano

Diagrama de la función `inmonthtodate`.



La función `inmonthtodate()` identifica un mes seleccionado como segmento. El límite de inicio es el comienzo del mes. El límite final se puede establecer como una fecha posterior en ese mes. Luego determina si un conjunto de fechas cae en ese segmento o no, devolviendo un valor booleano de TRUE o FALSE.

Argumentos

Argumento	Descripción
timestamp	La fecha que desea comparar con **base_date**.
base_date	La fecha que se utiliza para evaluar el mes.
period_no	El mes puede desplazarse mediante **period_no**. **period_no** es un entero, en el que el valor 0 indica el mes que contiene a **base_date**. Los valores negativos en **period_no** indican meses precedentes y los valores positivos indican meses subsiguientes.

Cuándo se utiliza

La función `inmonthtodate()` devuelve un resultado booleano. Normalmente, este tipo de función se utiliza como condición en una expresión condicional (`if expression`). La función `inmonthtodate()` devuelve una agregación o cálculo dependiendo de si una fecha ocurrió en el mes hasta e incluida la fecha en cuestión.

Por ejemplo, la función `inmonthtodate()` se puede utilizar para identificar todos los equipos fabricados en un determinado mes hasta una fecha específica.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	**Resultado**
`inmonthtodate ('01/25/2013', '25/01/2013', 0)`	Devuelve True
`inmonthtodate ('01/25/2013', '24/01/2013', 0)`	Devuelve False
`inmonthtodate ('01/25/2013', '28/02/2013', -1)`	Devuelve True

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022 se carga en una tabla llamada "`Transactions`".
- Un campo de fecha se proporciona en el formato `MM/DD/YYYY` de la variable del sistema `DateFormat`.
- Una instrucción load anterior que contiene lo siguiente:
    - La función `inmonthtodate()` que está establecida como el campo "`in_month_to_date`". Esto determina qué transacciones tuvieron lugar entre el 1 y el 26 de julio de 2022.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `in_month_to_date`

Tabla de resultados

date	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

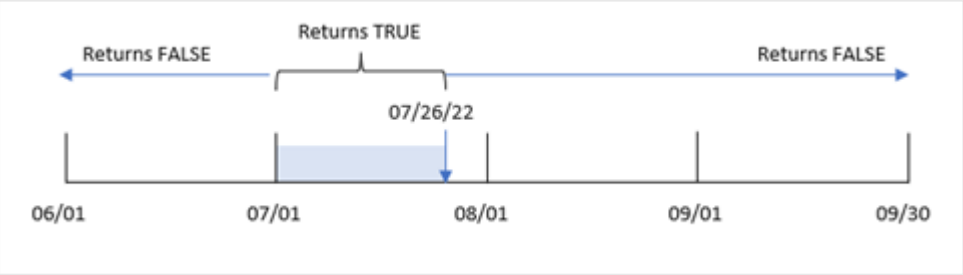
date	in_month_to_date
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	-1
7/22/2022	-1
7/23/2022	-1
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

El campo "`in_month_to_date`" se crea en la instrucción load anterior mediante el uso de la función `inmonthtodate()`.

El primer argumento identifica qué campo se está evaluando. El segundo argumento es una fecha codificada, 26 de julio, que es la `base_date`. Este argumento de `base_date`identifica qué mes está segmentado y el límite final de ese segmento.

Un `period_no` de `0` es el argumento final, lo que significa que la función no compara los meses que preceden o siguen al mes segmentado.

Diagrama de la función `inmonthtodate`, sin argumentos adicionales.



Como resultado, cualquier transacción que ocurra entre el 1 y el 26 de julio devuelve un resultado booleano de TRUE. Cualquier transacción que ocurra en julio después del 26 de julio devolverá un resultado booleano de FALSE, al igual que lo hará cualquier transacción que ocurra en cualquier otro mes del año.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

En este ejemplo, la tarea es crear un campo, "six_months_prior", que determine qué transacciones se realizaron 6 meses completos antes del 1 de julio y el 26 de julio.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

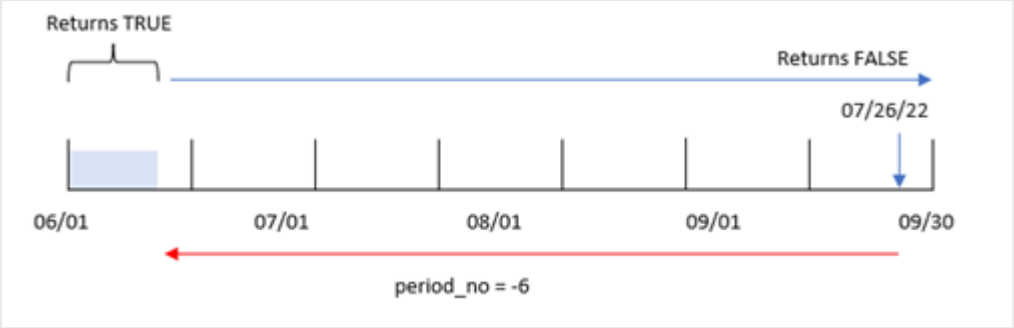
- date
- six_months_prior

Tabla de resultados

date	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

Utilizando `-6` como el argumento de `period_no` en la función `inmonthtodate()`, la función desplaza los límites del segmento del mes de comparación en un semestre. Inicialmente, el segmento de mes equivale a entre el 1 y el 26 de julio. El `period_no` luego desplaza este segmento seis meses y los límites de fecha se desplazan y caen entre el 1 de enero y el 26 de enero.

Diagrama de la función `inmonthtodate` con un period_no de-6.



Como resultado, cualquier transacción que ocurra entre el 1 y el 26 de enero devolverá un resultado booleano de TRUE.

Ejemplo 3: ejemplo gráfico

Script de carga y expresión de gráfico

General

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

En este ejemplo el conjunto de datos está sin modificar y cargado en la app. La tarea es crear un cálculo que determine si las transacciones se realizaron entre el 1 y el 26 de julio como una medida en un gráfico de la app.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

```
date
```

Para calcular si las transacciones se realizaron entre el 1 y el 26 de julio, cree la siguiente medida:

```
=inmonthtodate(date,'07/26/2022', 0)
```

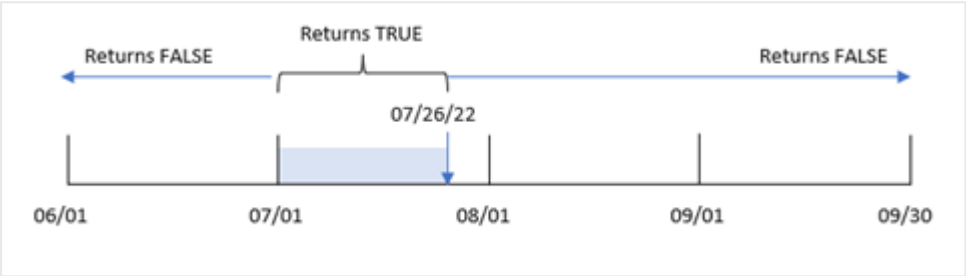
Tabla de resultados

date	=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
9/26/2022	0
10/14/2022	0
10/29/2022	0

La medida del campo "in_month_to_date" se crea en el gráfico mediante la función inmonthtodate().

El primer argumento identifica qué campo se está evaluando. El segundo argumento es una fecha codificada, 26 de julio, que es la base_date. Este argumento de la fecha base identifica qué mes está segmentado y el límite final de ese segmento. Un period_no de 0 es el argumento final. Esto significa que la función no compara los meses anteriores o posteriores al mes segmentado.

Diagrama de la función inmonthtodate, sin argumentos adicionales.



Como resultado, cualquier transacción que ocurra entre el 1 y el 26 de julio devuelve un resultado booleano de TRUE. Cualquier transacción que ocurra en julio después del 26 de julio devolverá un resultado booleano de FALSE, al igual que lo hará cualquier transacción que ocurra en cualquier otro mes del año.

Ejemplo 4: Escenario

Script de carga y resultados

Vista general

En este ejemplo, un conjunto de datos se carga en una tabla denominada "Products". La tabla contiene los siguientes campos:

- ID de producto
- Fecha de fabricación
- Precio de coste

Debido a un error del equipo, los productos que se fabricaron en el mes de julio de 2022 resultaron defectuosos. El problema se resolvió el 27 de julio de 2022.

Al usuario final le gustaría tener un gráfico que muestre, por mes, el estado de los productos que se fabricaron como "defectuosos" (booleano TRUE) o "sin defectos" (booleano FALSE) y el costo de los productos fabricados en ese mes.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `=monthname(manufacture_date)`
- `=if(Inmonthtodate(manufacture_date,makedate(2022,07,26),0),'Defective','Faultless')`

Para calcular el coste total de los productos, cree esta medida:

```
=sum(cost_price)
```

Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

monthname (manufacture_date)	if(Inmonthtodate(manufacture_date,makedate (2022,07,26),0),'Defectuoso','Sin defectos')	Sum(cost_ price)
Ene 2022	Sin defectos	$54.40
Feb 2022	Sin defectos	$145.69
Mar 2022	Sin defectos	$53.80
Abr 2022	Sin defectos	$82.06
May 2022	Sin defectos	$127.60
Jun 2022	Sin defectos	$141.82
Jul 2022	Defectuoso	$144.66
Jul 2022	Sin defectos	$69.98
Ago 2022	Sin defectos	$147.46

monthname (manufacture_date)	if(Inmonthtodate(manufacture_date,makedate (2022,07,26),0),'Defectuoso','Sin defectos')	Sum(cost_ price)
Sep 2022	Sin defectos	$84.21
Oct 2022	Sin defectos	$163.91

La función `inmonthtodate()` devuelve un valor booleano al evaluar las fechas de fabricación de cada uno de los productos.

Para las fechas que arrojan un valor booleano de TRUE, el producto se marca como "Defectuoso". Para cualquier producto que devuelva un valor de FALSE, y que por lo tanto no se haya fabricado en el mes hasta e incluido el 26 de julio, los productos se marcan como "Sin defectos".

inquarter

Esta función devuelve True si **timestamp** se encuentra dentro del trimestre que contiene a **base_date**.

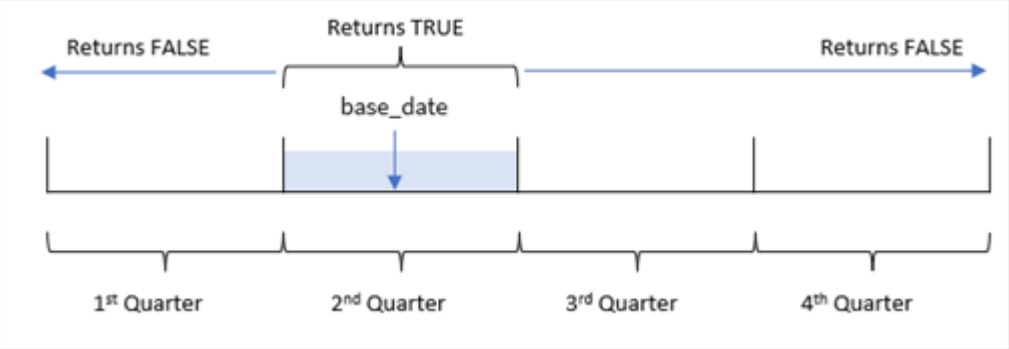
Sintaxis:

```
InQuarter (timestamp, base_date, period_no[, first_month_of_year])
```

Tipo de datos que devuelve: Booleano

En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.

Diagrama de rango de la función `inquarter()`



En otras palabras, la función `inquarter()` divide el año en cuatro trimestres iguales entre el 1 de enero y el 31 de diciembre. Puede utilizar el argumento de `first_month_of_year` para cambiar qué mes debe considerarse como el primero en su app y los trimestres cambiarán según ese argumento proporcionado. La `base_date` de la función identifica qué trimestre debe usarse como elemento comparador para la función. Por último, la función devuelve un resultado booleano al comparar valores de fecha con ese segmento trimestral.

Cuándo se utiliza

La función `inquarter()` devuelve un resultado booleano. Normalmente, este tipo de función se utilizará como condición en una `if expression`. Esto devuelve una agregación o cálculo que depende de si una fecha ocurrió o no en el trimestre seleccionado.

Por ejemplo, la función `inquarter()` se puede utilizar para identificar todos los equipos fabricados en un segmento trimestral en función de las fechas en que se fabricó el equipo.

Argumentos

Argumento	Descripción
timestamp	La fecha que desea comparar con **base_date**.
base_date	La fecha que se utiliza para evaluar el trimestre.
period_no	El trimestre puede desplazarse mediante **period_no**. **period_no** es un entero, en el que el valor 0 indica el trimestre que contiene a **base_date**. Los valores negativos en **period_no** indican trimestres precedentes y los valores positivos indican trimestres subsiguientes.
first_month_of_year	Si desea trabajar con años (fiscales) que no comiencen en enero, indique un valor entre 2 y 12 en **first_month_of_year**.

Puede utilizar los siguientes valores para establecer el primer mes del año en el argumento `first_month_of_year`:

first_month_of_year values

Month	Valor
Febrero	2
Marzo	3
Abril	4
May	5
Junio	6
Julio	7
Agosto	8
Septiembre	9
Octubre	10
Noviembre	11
Diciembre	12

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	**Resultado**
inquarter ('01/25/2013', '01/01/2013', 0)	Devuelve TRUE
inquarter ('01/25/2013', '04/01/2013', 0)	Devuelve FALSE
inquarter ('01/25/2013', '01/01/2013', -1)	Devuelve FALSE
inquarter ('12/25/2012', '01/01/2013', -1)	Devuelve TRUE
inquarter ('01/25/2013', '03/01/2013', 0, 3)	Devuelve FALSE
inquarter ('03/25/2013', '03/01/2013', 0, 3)	Devuelve TRUE

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- Una instrucción load precedente, que contiene la función "`inquarter()`" configurada como el campo "`in_quarter`" y que determina qué transacciones tuvieron lugar en el mismo trimestre que el 15 de mayo de 2022.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `in_quarter`

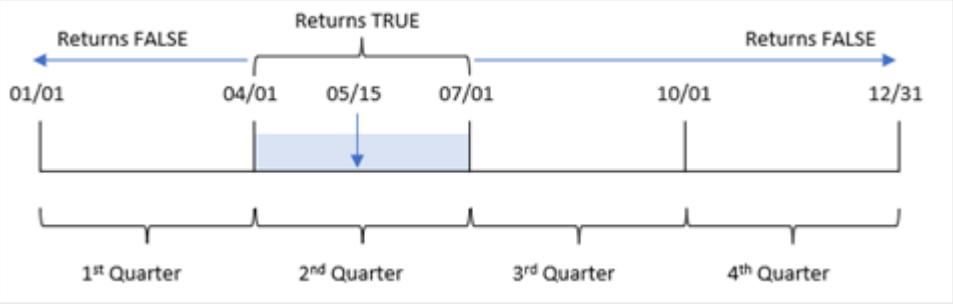
Tabla de resultados

date	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
6/15/2022	-1
6/26/2022	-1
7/9/2022	0

date	in_quarter
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

El campo "`in_quarter`" se crea en la instrucción load anterior mediante el uso de la función `inquarter()`. El primer argumento identifica qué campo se está evaluando. El segundo argumento es una fecha codificada del 15 de mayo que identifica qué trimestre definir como elemento comparador. Un `period_no` de 0 es el argumento final y garantiza que la función `inquarter()` no compare los trimestres que preceden o siguen al trimestre segmentado.

Diagrama de la función `inquarter()` con el 15 de mayo como fecha base



Cualquier transacción que ocurra entre el 1 de abril y el final del 30 de junio devuelve un resultado booleano de TRUE.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".

- Una instrucción load precedente, que contiene la función `inquarter()` configurada como el campo "`previous_quarter`" y que determina qué transacciones tuvieron lugar en el trimestre que precede al trimestre del 15 de mayo de 2022.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `previous_qtr`

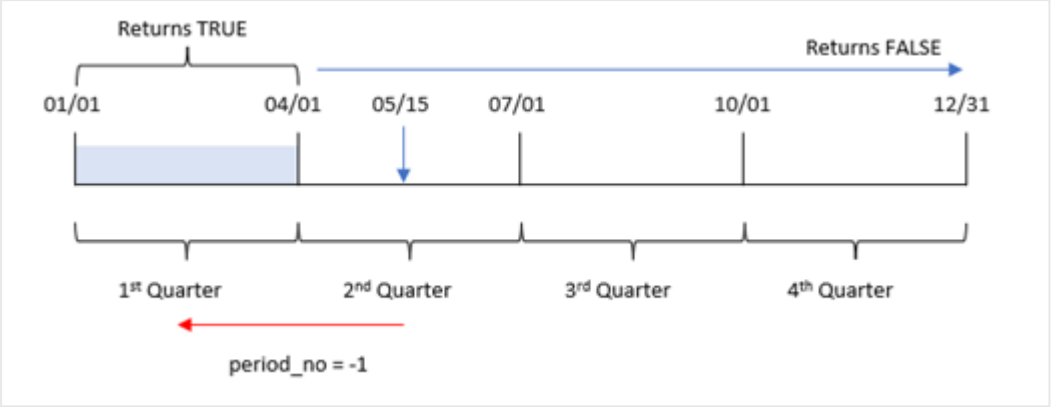
Tabla de resultados

date	previous_qtr
1/7/2022	-1

date	previous_qtr
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

Utilizando `-1` como el argumento de `period_no` en la función `inquarter()`, la función desplaza los límites del segmento de comparación hacia atrás en un trimestre completo. El 15 de mayo cae en el segundo trimestre del año y por lo tanto el segmento inicialmente equivale al trimestre comprendido entre el 1 de abril y el 30 de junio. El `period_no` desplaza este segmento por tres meses negativos y hace que los límites de fecha sean del 1 de enero al 30 de marzo.

Diagrama de la función `inquarter()` con el 15 de mayo como fecha base



Por lo tanto, cualquier transacción que ocurra entre el 1 de enero y el 30 de marzo devolverá un resultado booleano de TRUE.

Ejemplo 3: first_month_of_year

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- Una instrucción load precedente, que contiene la función "`inquarter()`" configurada como el campo "`in_quarter`" y que determina qué transacciones tuvieron lugar en el mismo trimestre que el 15 de mayo de 2022.

Sin embargo, en este ejemplo, la política de la organización es que marzo sea el primer mes del año fiscal.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `previous_qtr`

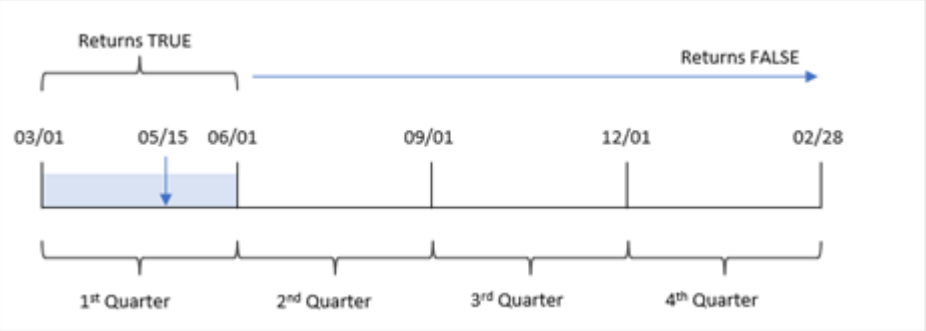
Tabla de resultados

date	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

date	previous_qtr
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

Utilizando `3` como el argumento de `first_month_of_year` en la función `inquarter()`, la función establece el comienzo del año en el 1 de marzo y luego divide el año en trimestres. Por lo tanto, los segmentos trimestrales son marzo-mayo, junio-agosto, septiembre-noviembre, diciembre-febrero. La `base_date` del 15 de mayo establece el trimestre marzo-mayo como trimestre de comparación para la función.

Diagrama de la función `inquarter()` con marzo fijado como el primer mes del año.



Por lo tanto, cualquier transacción que ocurra entre el 1 de marzo y el 31 de mayo devolverá un resultado booleano de TRUE.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- Una instrucción load precedente, que contiene la función "`inquarter()`" configurada como el campo "`in_quarter`" y que determina qué transacciones tuvieron lugar en el mismo trimestre que el 15 de mayo de 2022.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

- `date`

Cree la siguiente medida para calcular si las transacciones se realizaron en el mismo trimestre que el 15 de mayo:

```
=inquarter(date,'05/15/2022', 0)
```

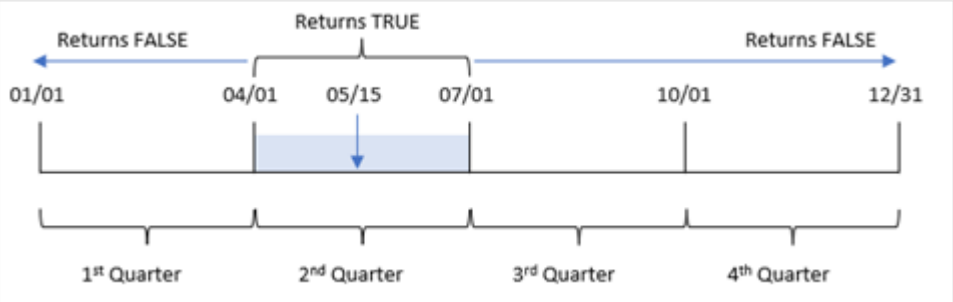
Tabla de resultados

date	in_quarter
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0

date	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

La medida "in_quarter" se crea en el gráfico usando la función `inquarter()`. El primer argumento identifica qué campo se está evaluando. El segundo argumento es una fecha codificada del 15 de mayo que identifica qué trimestre definir como elemento comparador. Un `period_no` de 0 es el argumento final y garantiza que la función `inquarter()` no compare los trimestres que preceden o siguen al trimestre segmentado.

Diagrama de la función `inquarter()` con el 15 de mayo como fecha base



Cualquier transacción que ocurra entre el 1 de abril y el final del 30 de junio devuelve un resultado booleano de TRUE.

Ejemplo 5: Escenario

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada "`Products`".
- La tabla contiene los siguientes campos:
    - product ID
    - product type
    - manufacture date
    - cost price

Se ha identificado que, debido a un error del equipo, los productos que se fabricaron en el trimestre del 15 de mayo de 2022 eran defectuosos. Al usuario final le gustaría tener un gráfico que muestre, por nombre de trimestre, el estado de los productos fabricados, si eran "defectuosos" o "sin defectos" y el coste de los productos fabricados en ese trimestre.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

`=quartername(manufacture_date)`

Cree las siguientes medidas:

- `=if(only(InQuarter(manufacture_date,makedate(2022,05,15),0)),'Defective','Faultless')` para identificar cuáles de los productos son defectuosos y cuáles no tienen defectos utilizando la función `inquarter()`:
- `=sum(cost_price)`, para mostrar la suma del coste de fabricación de cada producto.

Haga lo siguiente:

1. Establezca el **Formato numérico** de la medida en **Moneda**.
2. En **Aspecto**, deshabilite los **Totales**.

Tabla de resultados

quartername (manufacture_date)	=if(only(InQuarter(manufacture_date,makedate (2022,05,15),0)),'Defective','Faultless')	Sum(cost_ price)
Ene-Mar 2022	Sin defectos	253.89
Abr-Jun 2022	Defectuoso	351.48
Jul-Sep 2022	Sin defectos	446.31
Oct-Dic 2022	Sin defectos	163.91

La función `inquarter()` devuelve un valor booleano al evaluar las fechas de fabricación de cada uno de los productos. Para cualquier producto fabricado en el trimestre que contiene el 15 de mayo, la función `inquarter()` devuelve un valor booleano de TRUE y marca los productos como "defectuosos". Para cualquier producto que devuelva un valor de FALSE, y que por lo tanto no se haya fabricado en ese trimestre, marca los productos como "Sin defectos".

inquartertodate

Esta función devuelve True si **timestamp** se encuentra dentro de la parte del trimestre que contiene a **base_date** hasta e incluido el último milisegundo de **base_date**.

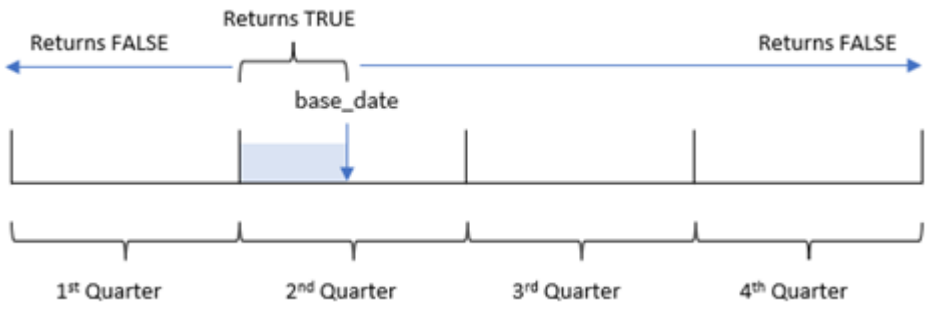
Sintaxis:

```
InQuarterToDate (timestamp, base_date, period_no [, first_month_of_year])
```

Tipo de datos que devuelve: Booleano

> *En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.*

Diagrama de la función inquartertodate



La función `inquartertodate()` divide el año en cuatro trimestres iguales entre el 1 de enero y el 31 de diciembre (o el inicio del año definido por el usuario y su fecha de finalización correspondiente). Usando la `base_date`, la función luego segmentará un trimestre en particular, con la `base_date` identificando tanto qué trimestre como la fecha máxima permitida para ese segmento de trimestre. Por último, la función devuelve un resultado booleano al comparar los valores de fecha prescritos con ese segmento.

Argumentos

Argumento	Descripción
timestamp	La fecha que desea comparar con **base_date**.
base_date	La fecha que se utiliza para evaluar el trimestre.
period_no	El trimestre puede desplazarse mediante **period_no**. **period_no** es un entero, en el que el valor 0 indica el trimestre que contiene a **base_date**. Los valores negativos en **period_no** indican trimestres precedentes y los valores positivos indican trimestres subsiguientes.
first_month_of_year	Si desea trabajar con años (fiscales) que no comiencen en enero, indique un valor entre 2 y 12 en **first_month_of_year**.

Cuándo se utiliza

La función `inquartertodate()` devuelve un resultado booleano. Normalmente, este tipo de función se utilizará como condición en una expresión `if`. La función `inquartertodate()` se usaría para devolver una agregación o cálculo dependiendo de si una fecha evaluada ocurrió en el trimestre hasta la fecha en cuestión inclusive.

Por ejemplo, la función `inquartertodate()` se puede utilizar para identificar todos los equipos fabricados en un trimestre hasta una fecha específica.

Ejemplos de funciones

Ejemplo	Resultado
`inquartertodate ('01/25/2013', '03/25/2013', 0)`	Devuelve `TRUE`, ya que el valor del `timestamp`, 01/25/2013, se encuentra dentro del período de tres meses del 01/01/2013 al 25/03/2013, en el cual se encuentra el valor de `base_date`, 25/03/2013.

Ejemplo	Resultado
`inquartertodate ('04/26/2013', '03/25/2013', 0)`	Devuelve `FALSE`, ya que el 26/04/2013 está fuera del mismo periodo que el ejemplo anterior.
`inquartertodate ('02/25/2013', '06/09/2013', -1)`	Devuelve `TRUE`, ya que el valor de `period_no`, -1, desplaza el período de búsqueda hacia atrás un período de tres meses (un trimestre del año). Esto hace que el período de búsqueda sea del 01/01/2013 al 09/03/2013.
`inquartertodate ('03/25/2006', '04/15/2006', 0, 2)`	Devuelve `TRUE`, ya que el valor de `first_month_of_year` está establecido en 2, lo que hace que el período de búsqueda sea del 01/02/2006 al 15/04/2006 en lugar del 01/04/2006 al 15/04/2006.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo, `in_quarter_to_date`, que determina qué transacciones se realizaron en el trimestre hasta el 15 de mayo de 2022.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `in_quarter_to_date`

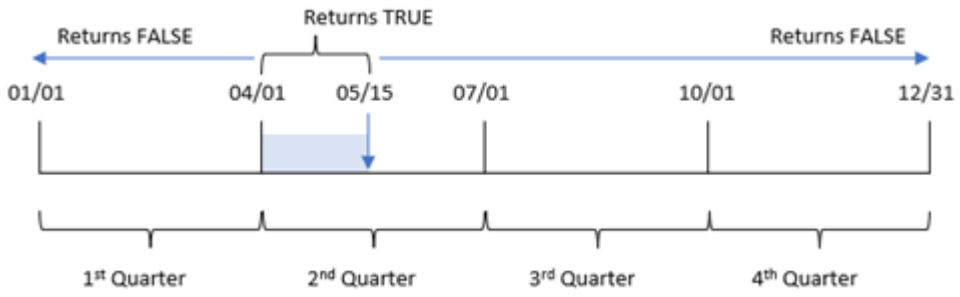
Tabla de resultados

date	in_quarter_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

date	in_quarter_to_date
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

El campo "`in_quarter_to_date`" se crea en la instrucción load anterior, mediante el uso de la función `inquartertodate()`. El primer argumento proporcionado identifica qué campo se está evaluando. El segundo argumento es una fecha codificada para el 15 de mayo, que es la `base_date` que identifica qué trimestre segmentar y define el límite final de ese segmento. Un `period_no` de 0 es el argumento final, lo que significa que la función no compara los trimestres que preceden o siguen al trimestre segmentado.

Diagrama de la función inquartertodate, sin argumentos adicionales



Cualquier transacción que ocurra entre el 1 de abril y el 15 de mayo devuelve un resultado booleano de TRUE. Las transacciones del 16 de mayo en adelante devolverán FALSE, al igual que cualquier transacción anterior al 1 de abril.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `previous_qtr_to_date`, que determina qué transacciones se realizaron un trimestre completo antes del segmento del trimestre que finaliza el 15 de mayo de 2022.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

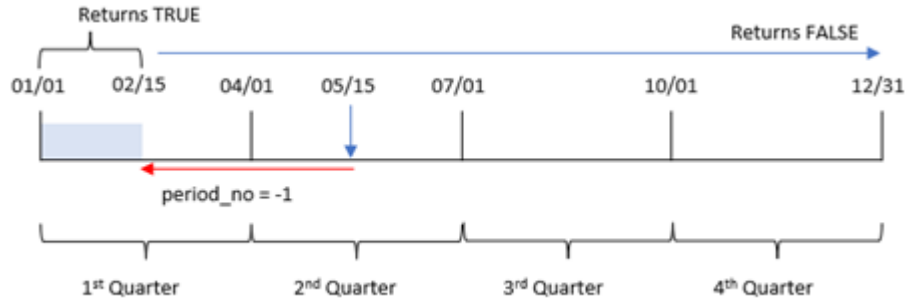
- `date`
- `previous_qtr_to_date`

Tabla de resultados

date	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

Un valor `period_no` de `-1` indica que la función `inquartertodate ()` compara el segmento del trimestre introducido con el trimestre anterior. El 15 de mayo cae en el segundo trimestre del año, por lo que el segmento inicialmente equivale al 1 de abril y al 15 de mayo. El `period_no` luego desplaza este segmento a tres meses antes, lo que hace que los límites de la fecha sean del 1 de enero al 15 de febrero.

Diagrama de la función inquartertodate, ejemplo de period_no



Por lo tanto, cualquier transacción que ocurra entre el 1 de enero y el 15 de febrero devolverá un resultado booleano de TRUE.

Ejemplo 3: first_month_of_year

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `in_quarter_to_date`, que determina qué transacciones se realizaron en el mismo trimestre hasta el 15 de mayo de 2022.

En este ejemplo, establecemos marzo como el primer mes del año fiscal.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `in_quarter_to_date`

Tabla de resultados

date	in_quarter_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

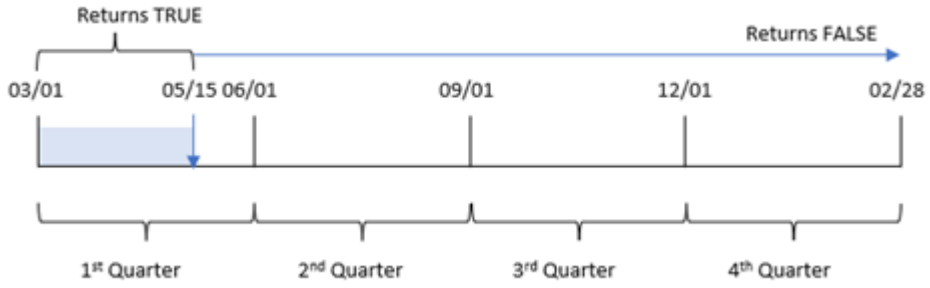
date	in_quarter_to_date
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Utilizando 3 como el argumento de `first_month_of_year` en la función `inquartertodate()`, la función comienza el año el 1 de marzo y luego divide el año en trimestres. Por lo tanto, los segmentos del cuatrimestre son:

- De marzo a mayo
- De junio a agosto
- De septiembre a noviembre
- De diciembre a febrero

La `base_date` del 15 de mayo luego segmenta el trimestre de marzo a mayo estableciendo su límite final como el 15 de mayo.

Diagrama de la función inquartertodate, ejemplo de first_month_of_year



En consecuencia, cualquier transacción que ocurra entre el 1 de marzo y el 15 de mayo arrojará un resultado booleano de TRUE, mientras que las transacciones con fechas fuera de estos límites arrojarán un valor de FALSE.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo. Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que determina qué transacciones se realizaron en el mismo trimestre que el 15 de mayo se crea como una medida en el objeto gráfico.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:`date`.

Cree la siguiente medida:

```
=inquartertodate(date,'05/15/2022', 0)
```

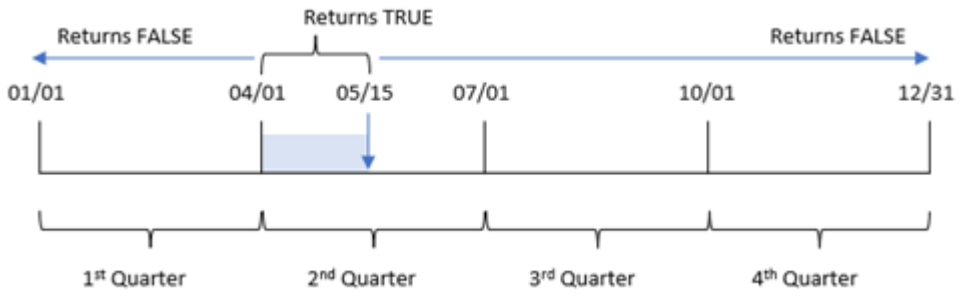
Tabla de resultados

date	=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

date	=inquartertodate(date,'05/15/2022', 0)
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

La medida `in_quarter_to_date` se crea en un objeto gráfico usando la función `inquartertodate()`. El primer argumento es el campo de fecha que se está evaluando. El segundo argumento es una fecha codificada para el 15 de mayo, que es la `base_date` que identifica qué trimestre segmentar y define el límite final de ese segmento. Un `period_no` de 0 es el argumento final, lo que significa que la función no compara los trimestres que preceden o siguen al trimestre segmentado.

Diagrama de la función inquartertodate, ejemplo de objeto gráfico



Cualquier transacción que ocurra entre el 1 de abril y el 15 de mayo devuelve un resultado booleano de TRUE. Las transacciones del 16 de mayo y posteriores devolverán FALSE, al igual que cualquier transacción anterior al 1 de abril.

Ejemplo 5: Escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Products`.
- Información sobre la identificación del producto, la fecha de fabricación y el precio de coste.

El 15 de mayo de 2022, se identificó y resolvió un error de equipo en el proceso de fabricación. Los productos que se fabricaron en ese trimestre hasta la fecha serán defectuosos. El usuario final desearía tener un objeto gráfico que muestre, por nombre de trimestre, el estado del producto fabricado, si es "defectuoso" o "sin defectos" y el costo de los productos fabricados en ese trimestre hasta la fecha.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla. Cree una dimensión para mostrar los nombres de los trimestres:
   `=quartername(manufacture_date)`
2. A continuación, cree una dimensión para identificar cuáles de los productos son defectuosos y cuáles son sin defectos:
   `=if(inquartertodate(manufacture_date,makedate(2022,05,15),0),'Defective','Faultless')`
3. Cree una medida para sumar el `cost_price` de los productos:
   `=sum(cost_price)`
4. Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

quartername (manufacture_date)	**if(inquartertodate(manufacture_date,makedate (2022,05,15),0),'Defective','Faultless')**	**Sum(cost_ price)**
Ene-Mar 2022	Sin defectos	$253.89
Abr-Jun 2022	Sin defectos	$229.03
Abr-Jun 2022	Defectuoso	$122.45
Jul-Sep 2022	Sin defectos	$446.31
Oct-Dic 2022	Sin defectos	$163.91

La función `inquartertodate()` devuelve un valor booleano al evaluar las fechas de fabricación de cada uno de los productos. Para aquellos que devuelven un valor booleano de `TRUE`, marca los productos como `'Defective'`. Para cualquier producto que devuelva un valor de `FALSE`, y por tanto no realizados en el trimestre hasta el 15 de mayo inclusive, marca los productos como `'Faultless'`.

inweek

La función devuelve True si **timestamp** se encuentra dentro de la semana que contiene a **base_ date**.

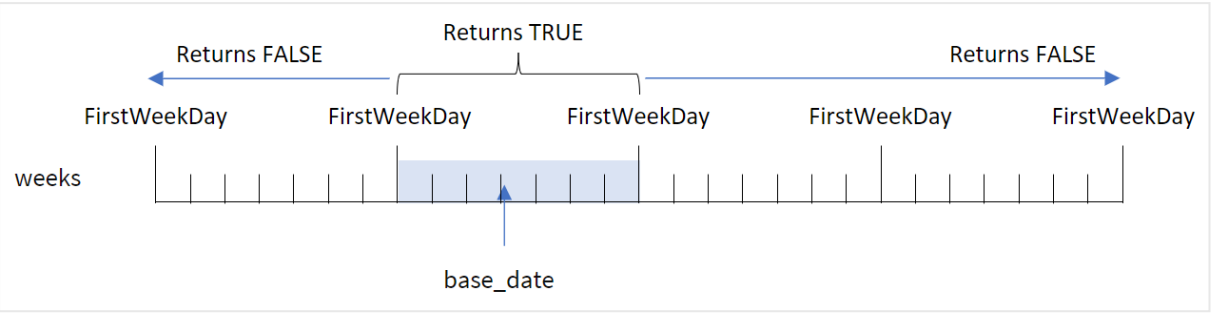
Sintaxis:

```
InWeek (timestamp, base_date, period_no[, first_week_day])
```

Tipo de datos que devuelve: Booleano

En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.

Diagrama del rango de la función `inweek()`



La función `inweek()` utiliza el argumento de `base_date` para identificar en qué periodo de siete días cae la fecha. El día de inicio de la semana se basa en la variable del sistema `FirstWeekDay`. No obstante, también puede cambiar el primer día de la semana usando el argumento `first_week_day` de la función `inweek()`.

Una vez que se haya definido la semana seleccionada, la función devolverá resultados booleanos al comparar los valores de fecha prescritos con ese segmento de semana.

Cuándo se utiliza

La función `inweek()` devuelve un resultado booleano. Normalmente, este tipo de función se utilizará como condición en una `if expression`. La función `inweek()` devuelve una agregación o cálculo que depende de si una fecha evaluada ocurrió en la semana con la fecha seleccionada del argumento `base_date`.

Por ejemplo, la función `inweek()` se puede utilizar para identificar todos los equipos fabricados en una semana determinada.

Argumentos

Argumento	Descripción
timestamp	La fecha que desea comparar con **base_date**.
base_date	La fecha que se utiliza para evaluar la semana.
period_no	La semana puede desplazarse mediante **period_no**. **period_no** es un entero, en el que el valor 0 indica la semana que contiene a **base_date**. Los valores negativos en **period_no** indican semanas precedentes y los valores positivos indican semanas subsiguientes.
first_week_day	De forma predeterminada, el primer día de la semana es el domingo (según lo determina la variable de sistema FirstWeekDay), comenzando a la medianoche entre el sábado y el domingo. El parámetro **first_week_day** reemplaza a la variable **FirstWeekDay**. Para indicar una semana que comienza en otro día, especifique un desplazamiento entre 0 y 6.

first_week_day values

Día	Valor
Lunes	0
Martes	1

Día	Valor
Miércoles	2
Jueves	3
Viernes	4
Sábado	5
Domingo	6

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
`inweek ('01/12/2006', '01/14/2006', 0)`	Devuelve TRUE
`inweek ('01/12/2006', '01/20/2006', 0 )`	Devuelve FALSE
`inweek ('01/12/2006', '01/14/2006', -1 )`	Devuelve FALSE
`inweek ('01/07/2006', '01/14/2006', -1)`	Devuelve TRUE
`inweek ('01/12/2006', '01/09/2006', 0, 3)`	Devuelve FALSE porque `first_week_day` está especificado como 3 (Jueves), lo que hace que 12/01/2006 sea el primer día de la semana que sigue a la semana que contiene a 09/01/2006.

Estos temas le ayudarán a trabajar con esta función:

Temas relacionados

Tema	Valor/indicador predeterminado	Descripción
FirstWeekDay (page 229)	6 / Domingo	Define el día de inicio de cada semana.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones del mes de enero de 2022, el cual se carga en una tabla denominada "`Transactions`".
- La variable del sistema `FirstWeekDay` configurada en `6` (domingo).
- Un load precedente que contiene lo siguiente:
  - La función `inweek()`, configurada como el campo "`in_week`" que determina qué transacciones se realizaron en la semana del 14 de enero de 2022.
  - La función `weekday()`, configurada como el campo "`week_day`" que muestra qué día de la semana corresponde a cada fecha.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `week_day`
- `in_week`

Tabla de resultados

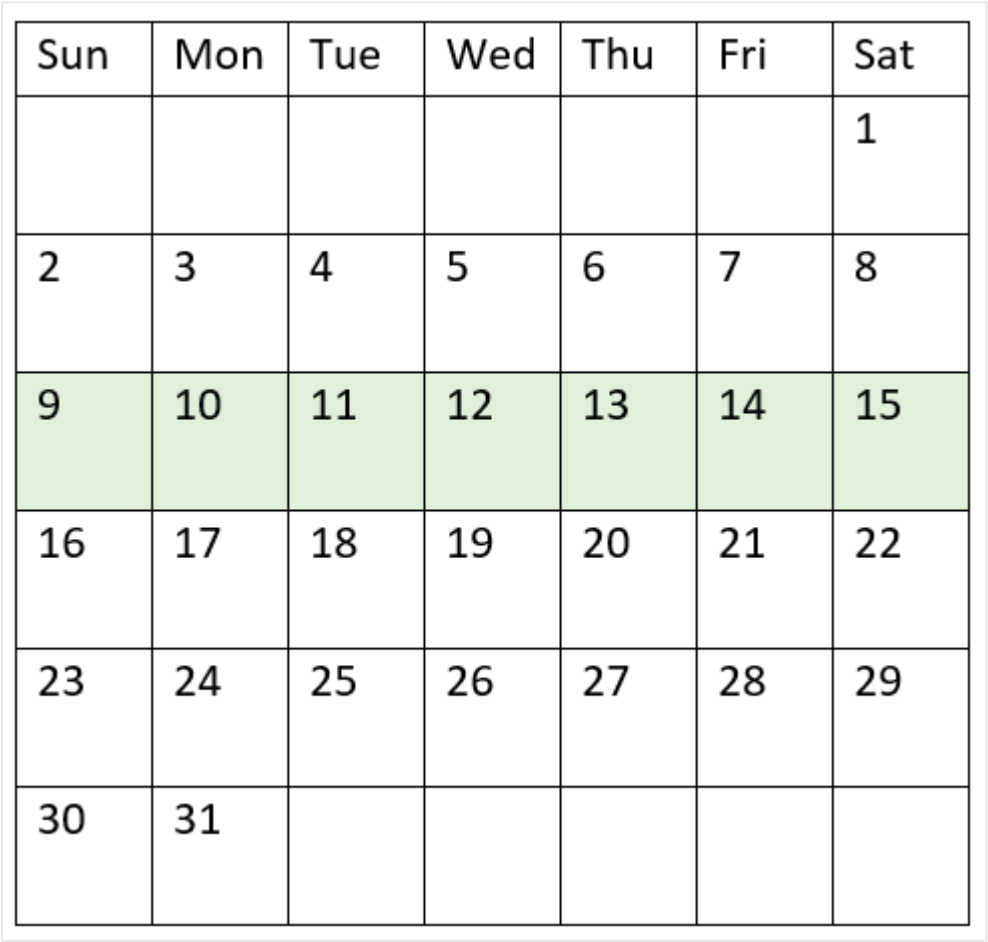
date	week_day	in_week
01/02/2022	Dom	0
01/05/2022	Mié	0
01/06/2022	Jue	0
01/08/2022	Sáb	0
01/09/2022	Dom	-1
01/10/2022	Lun	-1
01/11/2022	Mar	-1
01/12/2022	Mié	-1
01/13/2022	Jue	-1
01/14/2022	Vie	-1
01/15/2022	Sáb	-1
01/16/2022	Dom	0
01/17/2022	Lun	0
01/18/2022	Mar	0
01/26/2022	Mié	0
01/27/2022	Jue	0
01/28/2022	Vie	0

date	week_day	in_week
01/29/2022	Sáb	0
01/30/2022	Dom	0
01/31/2022	Lun	0

El campo "in_week" se crea en la instrucción load anterior mediante el uso de la función `inweek()`. El primer argumento identifica qué campo se está evaluando. El segundo argumento es una fecha codificada, 14 de enero, que es la `base_date`. El argumento de `base_date` funciona con la variable del sistema `FirstWeekDay` para identificar la semana de comparación. Un `period_no` de `0` — lo que significa que la función no compara las semanas anteriores o posteriores a la semana segmentada — es el argumento final.

La variable de sistema `FirstWeekDay` determina que las semanas comienzan en domingo y terminan en sábado. Por lo tanto, enero se dividiría en semanas de acuerdo con el diagrama inferior, siendo las fechas entre el 9 y el 15 de enero el período válido para el cálculo de `inweek()`:

Diagrama del calendario con el rango de la función `inweek()` resaltado



Cualquier transacción que ocurra entre el 9 y el 15 de enero devolverá un resultado booleano de `TRUE`.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- El mismo conjunto de datos que contiene un conjunto de transacciones para 2022 se carga en una tabla llamada "`Transactions`".
- La variable del sistema `FirstWeekDay` configurada en 6 (domingo).
- Un load precedente que contiene lo siguiente:
    - La función `inweek ()`, configurada como el campo "`prev_week`" que determina qué transacciones se realizaron una semana completa antes de la semana del 14 de enero de 2022.
    - La función `weekday()`, configurada como el campo "`week_day`" que muestra qué día de la semana corresponde a cada fecha.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

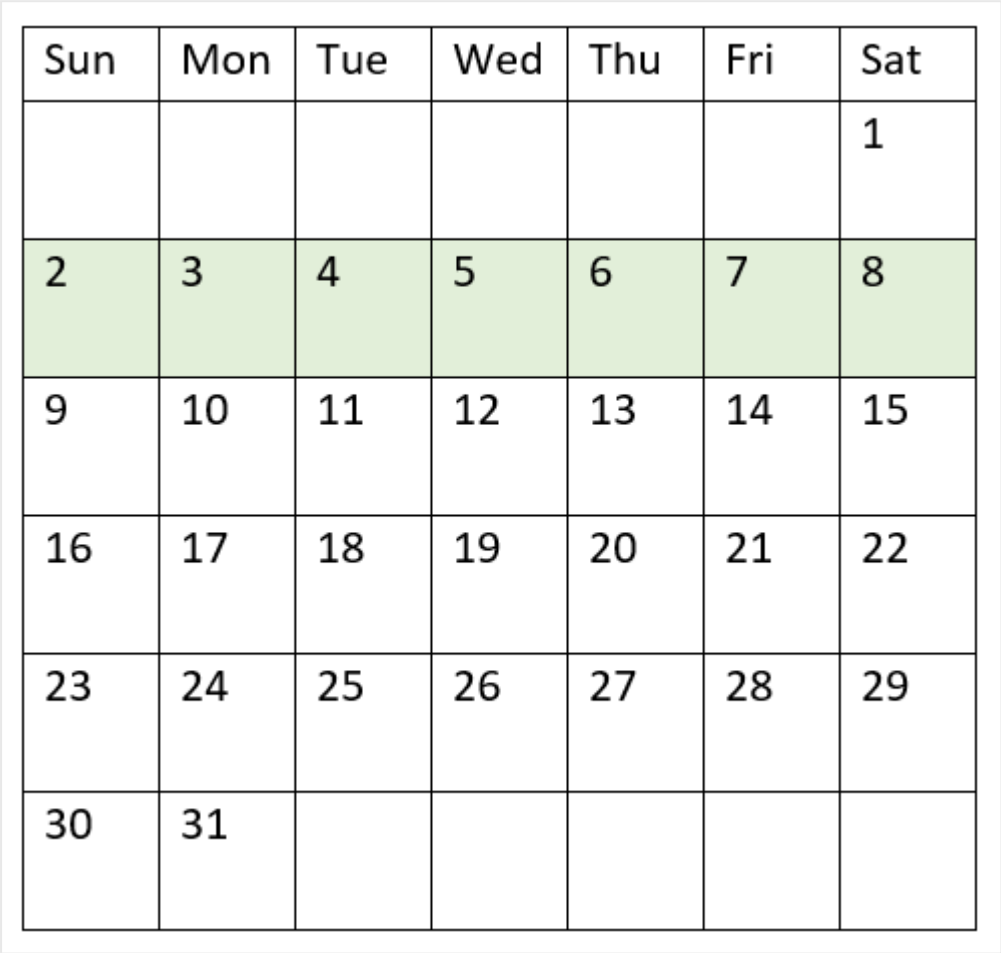
- `date`
- `week_day`
- `prev_week`

Tabla de resultados

date	week_day	prev_week
01/02/2022	Dom	-1
01/05/2022	Mié	-1
01/06/2022	Jue	-1
01/08/2022	Sáb	-1
01/09/2022	Dom	0
01/10/2022	Lun	0
01/11/2022	Mar	0
01/12/2022	Mié	0
01/13/2022	Jue	0
01/14/2022	Vie	0
01/15/2022	Sáb	0
01/16/2022	Dom	0
01/17/2022	Lun	0
01/18/2022	Mar	0
01/26/2022	Mié	0
01/27/2022	Jue	0
01/28/2022	Vie	0
01/29/2022	Sáb	0
01/30/2022	Dom	0
01/31/2022	Lun	0

Utilizar -1 como el argumento de period_no en la función inweek(), desplaza los límites de la semana de comparación hacia atrás en siete días completos. Con un period_no de 0 la semana sería entre el 9 y el 15 de enero. Pero en este ejemplo, el period_no de -1 desplaza el límite inicial y final de este segmento hacia atrás una semana. Los límites de fecha pasan a ser del 2 de enero al 8 de enero.

Diagrama del calendario con el rango de la función inweek() resaltado



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					

Por lo tanto, cualquier transacción que ocurra entre el 2 y el 8 de enero devolverá un resultado booleano de TRUE.

Ejemplo: first_week_day

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- El mismo conjunto de datos que contiene un conjunto de transacciones para 2022 se carga en una tabla llamada "Transactions".
- La variable del sistema FirstWeekDay configurada en 6 (domingo).
- Un load precedente que contiene lo siguiente:
    - La función inweek(), configurada como el campo "in_week" que determina qué transacciones se realizaron en la semana del 14 de enero de 2022.
    - La función weekday(), configurada como el campo "week_day" que muestra qué día de la semana corresponde a cada fecha.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `week_day`
- `in_week`

Tabla de resultados

date	week_day	in_week
01/02/2022	Dom	0
01/05/2022	Mié	0
01/06/2022	Jue	0
01/08/2022	Sáb	0
01/09/2022	Dom	0
01/10/2022	Lun	-1
01/11/2022	Mar	-1
01/12/2022	Mié	-1
01/13/2022	Jue	-1
01/14/2022	Vie	-1
01/15/2022	Sáb	-1
01/16/2022	Dom	-1
01/17/2022	Lun	0
01/18/2022	Mar	0
01/26/2022	Mié	0
01/27/2022	Jue	0
01/28/2022	Vie	0
01/29/2022	Sáb	0
01/30/2022	Dom	0
01/31/2022	Lun	0

Utilizando `0` como el argumento de `first_week_day` en la función `inweek()`, el argumento de la función reemplaza a la variable de sistema `FirstWeekDay` y establece el lunes como el primer día de la semana.

Diagrama del calendario con el rango de la función `inweek()` resaltado

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					

Por lo tanto, cualquier transacción que ocurra entre el 10 y el 16 de enero devolverá un resultado booleano de TRUE.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. Cree una medida en la tabla de resultados para determinar qué transacciones se realizaron en la semana del 14 de enero de 2022.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

- `date`

Cree las siguientes medidas:

- `=inweek (date,'01/14/2022',0)`, para calcular si las transacciones se realizaron en la misma semana del 14 de enero.
- `=weekday(date)`, para mostrar qué día de la semana corresponde a cada fecha.

Tabla de resultados

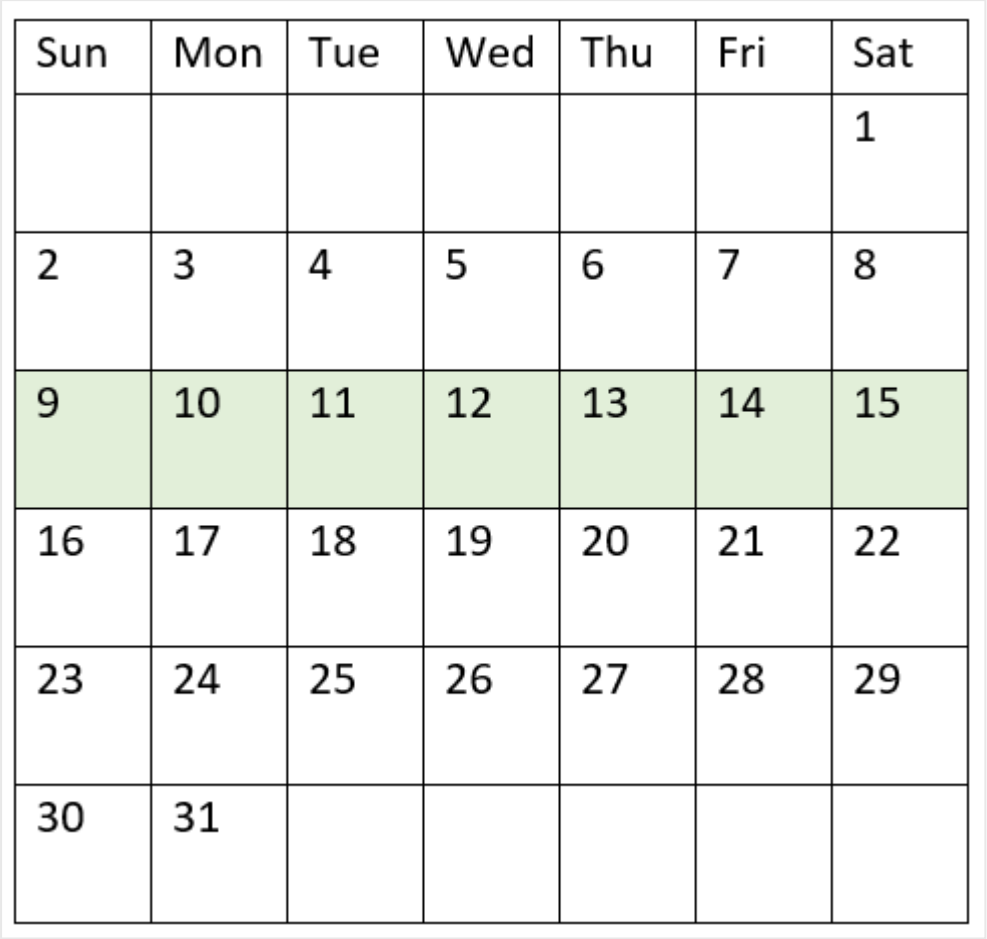
date	week_day	=inweek (date,'01/14/2022',0)
01/02/2022	Dom	0
01/05/2022	Mié	0
01/06/2022	Jue	0
01/08/2022	Sáb	0
01/09/2022	Dom	-1
01/10/2022	Lun	-1

date	week_day	=inweek (date,'01/14/2022',0)
01/11/2022	Mar	-1
01/12/2022	Mié	-1
01/13/2022	Jue	-1
01/14/2022	Vie	-1
01/15/2022	Sáb	-1
01/16/2022	Dom	0
01/17/2022	Lun	0
01/18/2022	Mar	0
01/26/2022	Mié	0
01/27/2022	Jue	0
01/28/2022	Vie	0
01/29/2022	Sáb	0
01/30/2022	Dom	0
01/31/2022	Lun	0

La medida "in_week" se crea en el gráfico mediante la función `inweek()`. El primer argumento identifica qué campo se está evaluando. El segundo argumento es una fecha codificada, 14 de enero, que es la `base_date`. El argumento de `base_date` funciona con la variable del sistema `FirstWeekDay` para identificar la semana de comparación. Un `period_no` de `0` es el argumento final.

La variable de sistema `FirstWeekDay` determina que las semanas comienzan en domingo y terminan en sábado. Por lo tanto, enero se dividiría en semanas de acuerdo con el diagrama inferior, siendo las fechas entre el 9 y el 15 de enero el período válido para el cálculo de `inweek()`:

Diagrama del calendario con el rango de la función `inweek()` resaltado



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					

Cualquier transacción que ocurra entre el 9 y el 15 de enero devolverá un resultado booleano de TRUE.

Ejemplo 5: escenario

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada "`Products`".
- La tabla contiene los siguientes campos:
  - product ID
  - product type
  - manufacture date
  - cost price

Se ha identificado que, debido a un error del equipo, los productos que se fabricaron en la semana del 12 de enero eran defectuosos. Al usuario final le gustaría tener un gráfico que muestre, por semana, el estado de los productos fabricados, si son "defectuosos" o "sin defectos", y el coste de producción de los productos fabricados en esa semana.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

- `=weekname(manufacture_date)`

Cree las siguientes medidas:

- `=if(only(inweek(manufacture_date,makedate(2022,01,12),0)),'Defective','Faultless')` para identificar cuáles de los productos son defectuosos y cuáles no tienen defectos utilizando la función `inweek()`:
- `=sum(cost_price)`, para mostrar la suma del coste de fabricación de cada producto.

Haga lo siguiente:

1. Establezca el **Formato numérico** de la medida en **Moneda**.
2. En **Aspecto**, deshabilite los **Totales**.

Tabla de resultados

weekname (manufacture_date)	=if(only(inweek(manufacture_date,makedate(2022,01,12),0)), 'Defectuoso','Sin defectos')	Sum(cost_ price)
2022/02	Sin defectos	200.09
2022/03	Defectuoso	441.51
2022/04	Sin defectos	178.41
2022/05	Sin defectos	231.67
2022/06	Sin defectos	163.91

La función `inweek()` devuelve un valor booleano al evaluar las fechas de fabricación de cada uno de los productos. Para cualquier producto fabricado en la semana del 12 de enero, la función `inweek()` devuelve un valor booleano de TRUE y marca los productos como "Defectuoso". Para cualquier producto que devuelva un valor de FALSE, y que por lo tanto no se haya fabricado en esa semana, marca los productos como "Sin defectos".

inweektodate

Esta función devuelve True si **timestamp** se encuentra dentro de la parte de la semana que contiene a **base_date** hasta e incluido el último milisegundo de **base_date**.

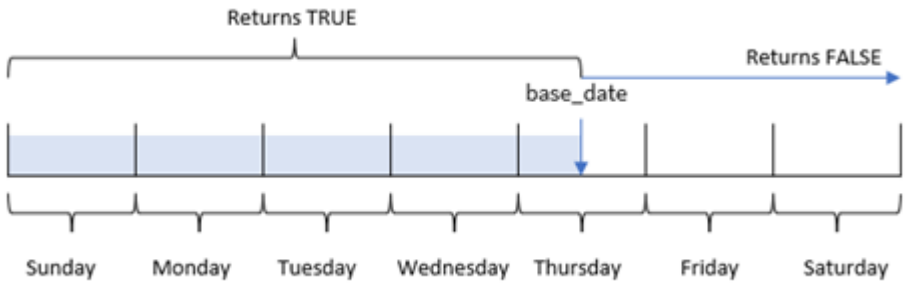
Sintaxis:

```
InWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Tipo de datos que devuelve: Booleano

> *En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.*

Diagrama de la función inweektodate



La función `inweektodate()` utiliza el parámetro `base_date` para identificar una fecha límite máxima de un segmento de semana, así como su fecha correspondiente para el inicio de la semana, que se basa en la variable de sistema `FirstWeekDay` (o parámetro `first_week_day` definido por el usuario). Una vez que se haya definido este segmento de semana, la función devolverá resultados booleanos al comparar los valores de fecha prescritos con ese segmento.

Cuándo se utiliza

La función `inweektodate()` devuelve un resultado booleano. Normalmente, este tipo de función se utilizará como condición en una expresión `if`. Esto devolverá una agregación o cálculo dependiendo de si una fecha evaluada ocurrió durante la semana en cuestión hasta una fecha en particular inclusive.

Por ejemplo, la función `inweektodate()` se puede utilizar para calcular todas las ventas realizadas durante una semana específica hasta una fecha determinada.

Argumentos

Argumento	Descripción
timestamp	La fecha que desea comparar con **base_date**.
base_date	La fecha que se utiliza para evaluar la semana.
period_no	La semana puede desplazarse mediante **period_no**. **period_no** es un entero, en el que el valor 0 indica la semana que contiene a **base_date**. Los valores negativos en **period_no** indican semanas precedentes y los valores positivos indican semanas subsiguientes.
first_week_day	De forma predeterminada, el primer día de la semana es el domingo (según lo determina la variable de sistema FirstWeekDay), comenzando a la medianoche entre el sábado y el domingo. El parámetro **first_week_day** reemplaza a la variable **FirstWeekDay**. Para indicar una semana que comienza en otro día, especifique un desplazamiento entre 0 y 6. Para una semana que comience el lunes y finalice el domingo, utilice un indicador de 0 para el lunes, 1 para el martes, 2 para el miércoles, 3 para el jueves, 4 para el viernes, 5 para el sábado y 6 para el domingo.

Ejemplos de funciones

Ejemplo	Interacción
`inweektodate ('01/12/2006', '01/12/2006', 0)`	Devuelve `TRUE`.
`inweektodate ('01/12/2006', '01/11/2006', 0)`	Devuelve `FALSE`.
`inweektodate ('01/12/2006', '01/18/2006', -1)`	Devuelve `FALSE`. Porque `period_no` está especificado como -1, los datos efectivos con los que se mide `timestamp` son 01/11/2006.
`inweektodate ('01/11/2006', '01/12/2006', 0, 3 )`	Devuelve `FALSE`, porque `first_week_day` está especificado como `3` (jueves), lo que hace que 01/12/2006 sea el primer día de la semana que sigue a la semana que contiene a 01/12/2006.

Estos temas le ayudarán a trabajar con esta función:

Temas relacionados

Tema	Valor/indicador predeterminado	Descripción
FirstWeekDay (page 229)	6 / Domingo	Define el día de inicio de cada semana.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones del mes de enero de 2022, el cual se carga en una tabla denominada "Transactions".
- El campo de datos proporcionado en el formato TimestampFormat='M/D/YYYY h:mm:ss[.fff]'.
- La creación de un campo, in_week_to_date, que determina qué transacciones se realizaron en la semana hasta el 14 de enero de 2022.
- La creación de un campo adicional, llamado weekday, usando la función weekday(). Este nuevo campo se crea para mostrar qué día de la semana corresponde a cada fecha.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- week_day

- in_week_to_date

Tabla de resultados

date	week_day	in_week_to_date
2022-01-02 12:22:06	Dom	0
2022-01-05 01:02:30	Mié	0
2022-01-06 15:36:20	Jue	0
2022-01-08 10:58:35	Sáb	0
2022-01-09 08:53:32	Dom	-1
2022-01-10 21:13:01	Lun	-1
2022-01-11 00:57:13	Mar	-1
2022-01-12 09:26:02	Mié	-1
2022-01-13 15:05:09	Jue	-1
2022-01-14 18:44:57	Vie	-1
2022-01-15 06:10:46	Sáb	0
2022-01-16 06:39:27	Dom	0
2022-01-17 10:44:16	Lun	0
2022-01-18 18:48:17	Mar	0
2022-01-26 04:36:03	Mié	0
2022-01-27 08:07:49	Jue	0
2022-01-28 12:24:29	Vie	0
2022-01-30 11:56:56	Dom	0
2022-01-30 14:40:19	Dom	0
2022-01-31 05:28:21	Lun	0

El campo "in_week_to_date" se crea en la instrucción load anterior, mediante el uso de la función inweektodate(). El primer argumento proporcionado identifica qué campo se está evaluando. El segundo argumento es una fecha codificada para el 14 de enero, que es la base_date que identifica qué semana segmentar y define el límite final de ese segmento. Un period_no de 0 es el argumento final, lo que significa que la función no compara las semanas que preceden o siguen a la semana segmentada.

La variable de sistema FirstWeekDay determina que las semanas comienzan en domingo y terminan en sábado. Por lo tanto, enero se dividiría en semanas de acuerdo con el siguiente diagrama, siendo las fechas entre el 9 y el 14 de enero el período válido para el cálculo de inweekdodate():

Diagrama de calendario que muestra las fechas de transacciones que devolverían un resultado booleano de 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					

Cualquier transacción que ocurra entre el 9 y el 14 de enero devuelve un resultado booleano de TRUE. Las transacciones anteriores y posteriores a las fechas devuelven un resultado booleano de FALSE.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `prev_week_to_date`, que determina qué transacciones se realizaron en una semana completa antes del segmento de semana que finaliza el 14 de enero de 2022.
- La creación de un campo adicional, llamado `weekday`, usando la función `weekday()`. Este se crea para mostrar qué día de la semana corresponde a cada fecha.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `week_day`
- `prev_week_to_date`

Tabla de resultados

date	week_day	prev_week_to_date
2022-01-02 12:22:06	Dom	-1
2022-01-05 01:02:30	Mié	-1
2022-01-06 15:36:20	Jue	-1
2022-01-08 10:58:35	Sáb	0
2022-01-09 08:53:32	Dom	0
2022-01-10 21:13:01	Lun	0
2022-01-11 00:57:13	Mar	0
2022-01-12 09:26:02	Mié	0
2022-01-13 15:05:09	Jue	0
2022-01-14 18:44:57	Vie	0
2022-01-15 06:10:46	Sáb	0

date	week_day	prev_week_to_date
2022-01-16 06:39:27	Dom	0
2022-01-17 10:44:16	Lun	0
2022-01-18 18:48:17	Mar	0
2022-01-26 04:36:03	Mié	0
2022-01-27 08:07:49	Jue	0
2022-01-28 12:24:29	Vie	0
2022-01-30 11:56:56	Dom	0
2022-01-30 14:40:19	Dom	0
2022-01-31 05:28:21	Lun	0

Un valor `period_no` de -1 indica que la función `inweektodate ()` compara el segmento del trimestre introducido con la semana anterior. El segmento de la semana equivale inicialmente al período entre el 9 y el 14 de enero. El `period_no` desplaza después el límite inicial y final de este segmento a una semana antes, lo que hace que los límites de fecha se conviertan del 2 de enero al 7 de enero.

Diagrama de calendario que muestra las fechas de transacciones que devolverían un resultado booleano de 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					

Por lo tanto, cualquier transacción que ocurra entre el 2 y el 8 de enero (sin incluir el mismo 8 de enero) devolverá un resultado booleano de TRUE.

Ejemplo: first_week_day

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `in_week_to_date`, que determina qué transacciones se realizaron en la semana hasta el 14 de enero de 2022.
- La creación de un campo adicional, llamado `weekday`, usando la función `weekday()`. Este se crea para mostrar qué día de la semana corresponde a cada fecha.

En este ejemplo, consideramos el lunes como primer día de la semana.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `week_day`

- in_week_to_date

Tabla de resultados

date	week_day	in_week_to_date
2022-01-02 12:22:06	Dom	0
2022-01-05 01:02:30	Mié	0
2022-01-06 15:36:20	Jue	0
2022-01-08 10:58:35	Sáb	0
2022-01-09 08:53:32	Dom	0
2022-01-10 21:13:01	Lun	-1
2022-01-11 00:57:13	Mar	-1
2022-01-12 09:26:02	Mié	-1
2022-01-13 15:05:09	Jue	-1
2022-01-14 18:44:57	Vie	-1
2022-01-15 06:10:46	Sáb	0
2022-01-16 06:39:27	Dom	0
2022-01-17 10:44:16	Lun	0
2022-01-18 18:48:17	Mar	0
2022-01-26 04:36:03	Mié	0
2022-01-27 08:07:49	Jue	0
2022-01-28 12:24:29	Vie	0
2022-01-30 11:56:56	Dom	0
2022-01-30 14:40:19	Dom	0
2022-01-31 05:28:21	Lun	0

Utilizando 0 como el argumento de `first_week_day` en la función `inweektodate()`, el argumento de la función reemplaza a la variable de sistema `FirstWeekDay` y establece el lunes como el primer día de la semana.

Diagrama de calendario que muestra las fechas de transacciones que devolverían un resultado booleano de 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						

Por lo tanto, cualquier transacción que ocurra entre el 10 y el 14 de enero arrojará un resultado booleano de TRUE, mientras que las transacciones con fechas fuera de estos límites arrojarán un valor de FALSE.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo. Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que determina qué transacciones se realizaron en la semana hasta el 14 de enero de 2022 se crea como una medida en el objeto gráfico.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.
2. Para calcular si las transacciones se realizaron en la misma semana hasta el 14 de enero, cree la siguiente medida:
   `=inweektodate(date,'01/14/2022',0)`
3. Para mostrar qué día de la semana corresponde a cada fecha, cree una medida adicional:
   `=weekday(date)`

Tabla de resultados

date	week_day	in_week_to_date
2022-01-02 12:22:06	Dom	0
2022-01-05 01:02:30	Mié	0
2022-01-06 15:36:20	Jue	0
2022-01-08 10:58:35	Sáb	0
2022-01-09 08:53:32	Dom	-1
2022-01-10 21:13:01	Lun	-1
2022-01-11 00:57:13	Mar	-1
2022-01-12 09:26:02	Mié	-1
2022-01-13 15:05:09	Jue	-1
2022-01-14 18:44:57	Vie	-1
2022-01-15 06:10:46	Sáb	0
2022-01-16 06:39:27	Dom	0
2022-01-17 10:44:16	Lun	0

date	week_day	in_week_to_date
2022-01-18 18:48:17	Mar	0
2022-01-26 04:36:03	Mié	0
2022-01-27 08:07:49	Jue	0
2022-01-28 12:24:29	Vie	0
2022-01-30 11:56:56	Dom	0
2022-01-30 14:40:19	Dom	0
2022-01-31 05:28:21	Lun	0

El campo `in_week_to_date` se crea como una medida usando la función `inweektodate()`. El primer argumento proporcionado identifica qué campo se está evaluando. El segundo argumento es una fecha codificada para el 14 de enero, que es la `base_date` que identifica qué semana segmentar y define el límite final de ese segmento. Un `period_no` de 0 es el argumento final, lo que significa que la función no compara las semanas que preceden o siguen a la semana segmentada.

La variable de sistema `FirstWeekDay` determina que las semanas comienzan en domingo y terminan en sábado. Por lo tanto, enero se dividiría en semanas de acuerdo con el siguiente diagrama, siendo las fechas entre el 9 y el 14 de enero el período válido para el cálculo de `inweekdodate()`:

Diagrama de calendario que muestra las fechas de transacciones que devolverían un resultado booleano de 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					

Cualquier transacción que ocurra entre el 9 y el 14 de enero devuelve un resultado booleano de TRUE. Las transacciones anteriores y posteriores a las fechas devuelven un resultado booleano de FALSE.

Ejemplo 5: Escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Products`.
- Información sobre la identificación del producto, la fecha de fabricación y el precio de coste.

Se ha identificado que, debido a un error del equipo, los productos que se fabricaron en la semana del 12 de enero eran defectuosos. El problema se resolvió el 13 de enero. Al usuario final le gustaría tener un objeto gráfico que muestre, por semana, si los productos fabricados son "defectuosos" o "sin defectos", y el costo de los productos fabricados en esa semana.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla. Cree una dimensión para mostrar los nombres de las semanas:
   `=weekname(manufacture_date)`
2. A continuación, cree una dimensión para identificar cuáles de los productos son defectuosos y cuáles son sin defectos:
   `=if(inWeektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
3. Cree una medida para sumar el `cost_price` de los productos:
   `=sum(cost_price)`
4. Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

weekname(manufacture_date)	if(inweektodate(manufacture_date,makedate (2022,01,12),0),'Defectuoso','Sin defectos')	Sum(cost_price)
2022/02	Sin defectos	$200.09
2022/03	Defectuoso	$263.46
2022/03	Sin defectos	$178.05
2022/04	Sin defectos	$178.41
2022/05	Sin defectos	$147.46
2022/06	Sin defectos	$248.12

La función `inweektodate()` devuelve un valor booleano al evaluar las fechas de fabricación de cada uno de los productos. Para aquellos que devuelven un valor booleano de `TRUE`, marca los productos como `'Defective'`. Para cualquier producto que devuelva un valor de `FALSE`, y por tanto no realizados en el trimestre hasta el 12 de mayo inclusive, marca los productos como `'Faultless'`.

inyear

Esta función devuelve True si **timestamp** se encuentra dentro del año que contiene a **base_date**.

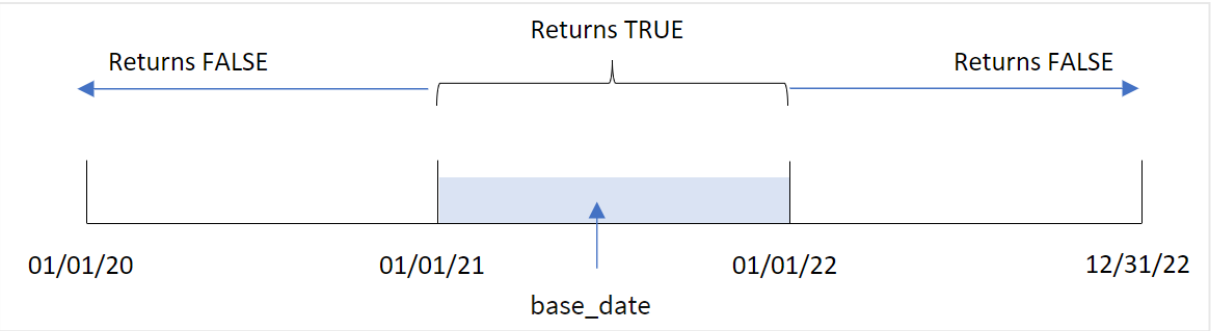
Sintaxis:

```
InYear (timestamp, base_date, period_no [, first_month_of_year])
```

Tipo de datos que devuelve: Booleano

En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.

Diagrama del rango de la función `inyear()`



La función `inyear()` devuelve un resultado booleano al comparar los valores de fecha seleccionados con un año definido por `base_date`.

Cuándo se utiliza

La función `inyear()` devuelve un resultado booleano. Normalmente, este tipo de función se utilizará como condición en una `if expression`. Esto devuelve una agregación o cálculo dependiente de si una fecha evaluada ocurrió en el año en cuestión. Por ejemplo, la función `inyear()` se puede usar para identificar todas las ventas que ocurrieron en un determinado año.

Argumentos

Argumento	Descripción
timestamp	La fecha que desea comparar con **base_date**.
base_date	La fecha que se utiliza para evaluar el año.
period_no	El año puede desplazarse mediante **period_no**. **period_no** es un entero, en el que el valor 0 indica el año que contiene a **base_date**. Los valores negativos en **period_no** indican años precedentes y los valores positivos indican años subsiguientes.
first_month_of_year	Si desea trabajar con años (fiscales) que no comiencen en enero, indique un valor entre 2 y 12 en **first_month_of_year**.

Puede utilizar los siguientes valores para establecer el primer mes del año en el argumento `first_month_of_year`:

first_month_of_year values

Month	Valor
Febrero	2
Marzo	3
Abril	4
May	5
Junio	6

Month	Valor
Julio	7
Agosto	8
Septiembre	9
Octubre	10
Noviembre	11
Diciembre	12

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
`inyear ('01/25/2013', '01/01/2013', 0 )`	Devuelve TRUE
`inyear ('01/25/2012', '01/01/2013', 0)`	Devuelve FALSE
`inyear ('01/25/2013', '01/01/2013', -1)`	Devuelve FALSE
`inyear ('01/25/2012', '01/01/2013', -1 )`	Devuelve TRUE
`inyear ('01/25/2013', '01/01/2013', 0, 3)`	Devuelve TRUE El valor de `base_date` y `first_month_of_year` especifican que la marca de tiempo debe estar entre el 03/01/2012 y el 28/02/2013
`inyear ('03/25/2013', '07/01/2013', 0, 3 )`	Devuelve TRUE

Ejemplo 1: ejemplo básico

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones entre 2020 y 2022, que se carga en una tabla llamada "`Transactions`".
- Una instrucción load precedente, que contiene la función `inyear()` configurada como el campo "`in_year`" y determina qué transacciones tuvieron lugar en el mismo año que el 26 de julio de 2021.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

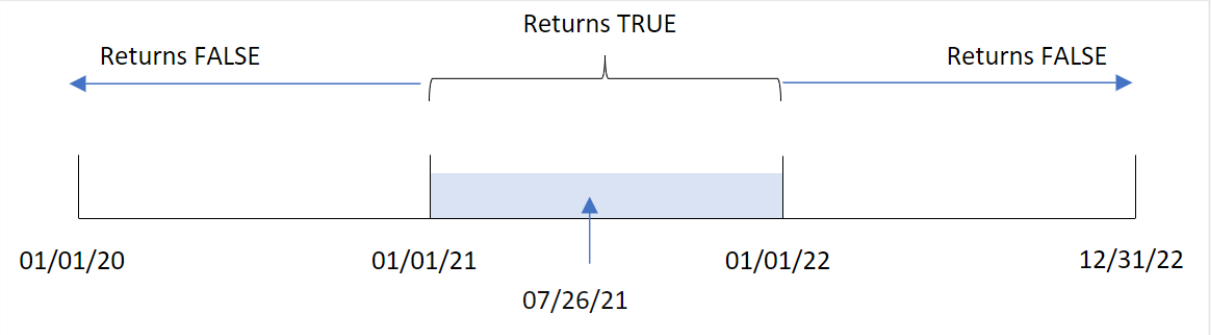
- `date`
- `in_year`

Tabla de resultados

date	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
07/26/2021	-1
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

El campo "`in_year`" se crea en la instrucción load anterior mediante el uso de la función `inyear()`. El primer argumento identifica qué campo se está evaluando. El segundo argumento es una fecha codificada del 26 de julio de 2021, que es la `base_date` que identifica el año de comparación. Un `period_no` de `0` es el argumento final, lo que significa que la función `inyear()` no compara los años que preceden o siguen al año.

Diagrama del rango de la función `inyear()` con el 26 de julio como fecha base



Cualquier transacción que ocurra en 2021 devolverá un resultado booleano de TRUE.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones entre 2020 y 2022, que se carga en una tabla llamada "`Transactions`".
- Una instrucción load precedente, que contiene la función `inyear()` configurada como el campo "`previous_year`" y que determina qué transacciones tuvieron lugar antes del año que contiene el 26 de julio de 2021.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `previous_year`

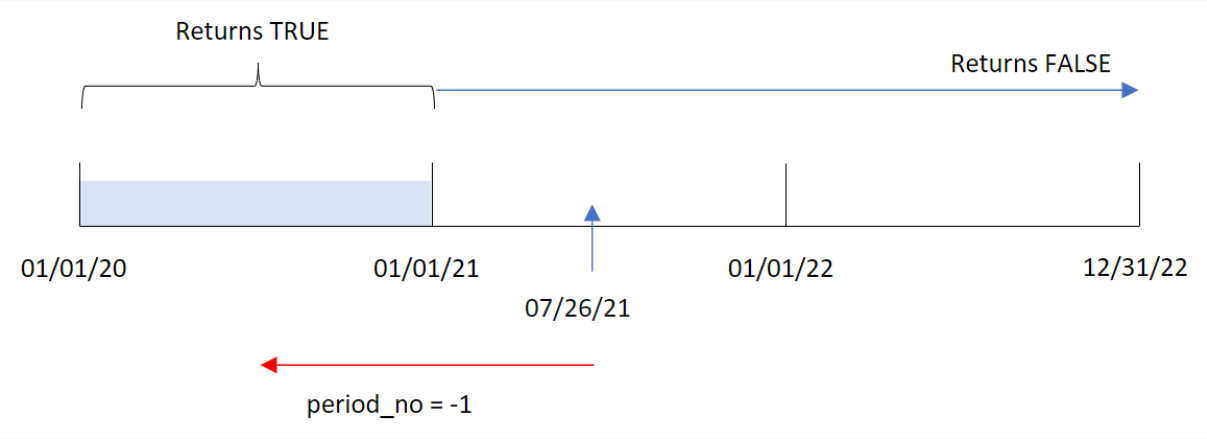
Tabla de resultados

date	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
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

date	previous_year
07/18/2022	0
11/14/2022	0
12/12/2022	0

Usar -1 como el argumento de period_no en la función inyear() desplaza los límites del año de comparación hacia atrás un año completo. 2021 se identifica inicialmente como el año de comparación. period_no desplaza el año de comparación en un año, lo que convierte a 2020 en el año de comparación.

Diagrama de rango de la función inyear() con el argumento de period_no establecido en -1



Por lo tanto, cualquier transacción que ocurra en 2020 devolverá un resultado booleano de TRUE.

Ejemplo 3: first_month_of_year

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones entre 2020 y 2022, que se carga en una tabla llamada "Transactions".
- Una instrucción load precedente, que contiene la función inyear() configurada como el campo "in_year" y determina qué transacciones tuvieron lugar en el mismo año que el 26 de julio de 2021.

Sin embargo, en este ejemplo, la política de la organización es que marzo sea el primer mes del año fiscal.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `in_year`

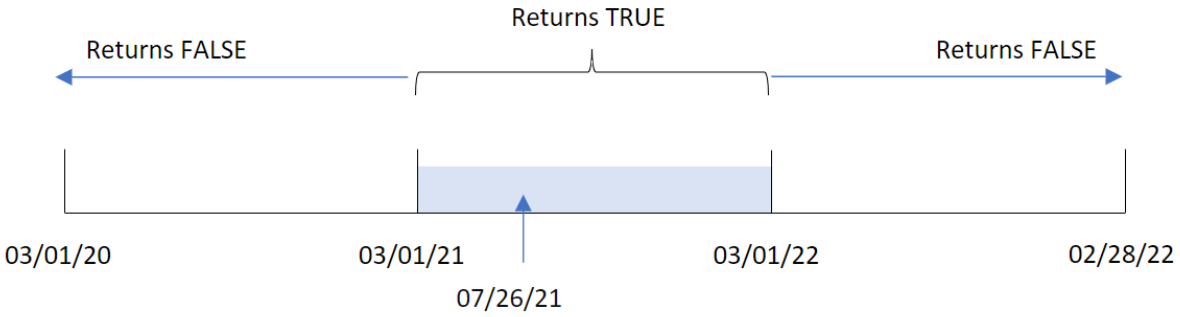
Tabla de resultados

date	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

date	in_year
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

Usando `3` como argumento de `first_month_of_year` en la función `inyear()`, inicia el año el 1 de marzo y termina el año a finales de febrero.

Diagrama del rango de la función `inyear()` con marzo fijado como el primer mes del año.



Por lo tanto, cualquier transacción que ocurra entre el 1 de marzo de 2021 y el 1 de marzo de 2022 devolverá un resultado booleano de TRUE.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que determina si las transacciones tuvieron lugar en el mismo año que el 26 de julio de 2021 se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

- `date`

Para calcular si las transacciones tuvieron lugar en el mismo año que el 26 de julio de 2021, cree la siguiente medida:

- `=inyear(date,'07/26/2021', 0)`

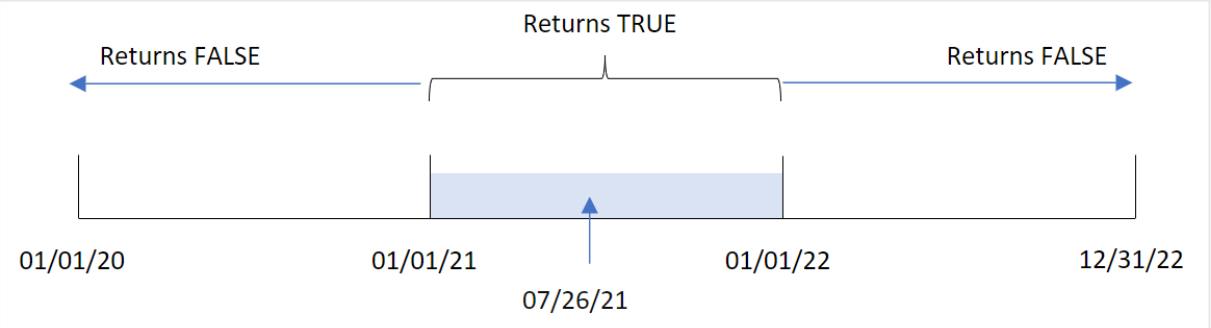
Tabla de resultados

date	=inyear(date,'07/26/2021',0)
01/13/2020	0
02/26/2020	0

date	=inyear(date,'07/26/2021',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

El campo "in_year" se crea en el gráfico mediante la función `inyear()`. El primer argumento identifica qué campo se está evaluando. El segundo argumento es una fecha codificada del 26 de julio de 2021, que es la base_date que identifica el año de comparación. Un period_no de 0 es el argumento final, lo que significa que la función `inyear()` no compara los años que preceden o siguen al año.

Diagrama del rango de la función `inyear()` con el 27 de julio como fecha base



Cualquier transacción que ocurra en 2021 devolverá un resultado booleano de TRUE.

Ejemplo 5: escenario

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada "Products".
- La tabla contiene los siguientes campos:
    - product ID
    - product type
    - manufacture date
    - cost price

Al usuario final le gustaría tener un objeto gráfico que muestre, por tipo de producto, el coste de producción de los productos fabricados en 2021.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

- `product_type`

Para calcular la suma de cada producto que se fabricó en 2021, cree la siguiente medida:

- `=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))`

Haga lo siguiente:

1. Establezca el **Formato numérico** de la medida en **Moneda**.
2. En **Aspecto**, deshabilite los **Totales**.

Tabla de resultados

product_type	=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))
producto A	$95.93
producto B	$128.66
producto C	$61.89
producto D	$171.21

La función `inyear()` devuelve un valor booleano al evaluar las fechas de fabricación de cada uno de los productos. Para cualquier producto fabricado en 2021, la función `inyear()` devuelve un valor booleano de TRUE y muestra la suma de `cost_price`.

inyeartodate

Esta función devuelve True si **timestamp** se encuentra dentro de la parte del año que contiene a **base_date** hasta e incluido el último milisegundo de **base_date**.

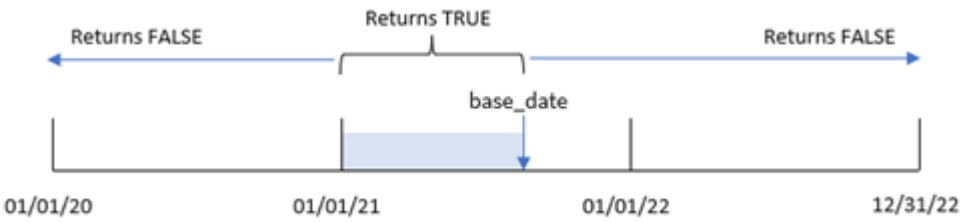
Sintaxis:

```
InYearToDate (timestamp, base_date, period_no[, first_month_of_year])
```

Tipo de datos que devuelve: Booleano

En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.

Diagrama de la función inyeartodate



La función `inyeartodate()` segmentará una porción particular del año con la `base_date`, identificando la fecha máxima permitida para ese segmento de año. La función evalúa a continuación si un campo de fecha o valor cae dentro de este segmento y devuelve un resultado booleano.

Argumentos

Argumento	Descripción
timestamp	La fecha que desea comparar con **base_date**.
base_date	La fecha que se utiliza para evaluar el año.
period_no	El año puede desplazarse mediante **period_no**. **period_no** es un entero, en el que el valor 0 indica el año que contiene a **base_date**. Los valores negativos en **period_no** indican años precedentes y los valores positivos indican años subsiguientes.
first_month_of_year	Si desea trabajar con años (fiscales) que no comiencen en enero, indique un valor entre 2 y 12 en **first_month_of_year**.

Cuándo se utiliza

La función `inyeartodate()` devuelve un resultado booleano. Normalmente, este tipo de función se utilizará como condición en una expresión `if`. Esto devolvería una agregación o cálculo dependiendo de si una fecha evaluada ocurrió en el año hasta e incluida la fecha en cuestión.

Por ejemplo, la función `inyeartodate()` se puede utilizar para identificar todos los equipos fabricados en un año hasta una fecha específica.

Estos ejemplos utilizan el formato de fecha MM/DD/AAAA. El formato de fecha se especifica en la sentencia `SET DateFormat` en la parte superior de su script de carga de datos. Cambie el formato en los ejemplos según se ajuste a sus necesidades.

Ejemplos de funciones

Ejemplo	Resultado
`inyeartodate ('01/25/2013', '02/01/2013', 0)`	Devuelve `TRUE`.
`inyeartodate ('01/25/2012', '01/01/2013', 0)`	Devuelve `FALSE`.

Ejemplo	Resultado
`inyeartodate ('01/25/2012', '02/01/2013', -1)`	Devuelve `TRUE`.
`inyeartodate ('11/25/2012', '01/31/2013', 0, 4)`	Devuelve `TRUE`. El valor de `timestamp` cae dentro del año fiscal que comienza en el cuarto mes y antes del valor de `base_date`.
`inyeartodate ('3/31/2013', '01/31/2013', 0, 4 )`	Devuelve `FALSE`. Comparado con el ejemplo anterior, el valor de `timestamp` todavía está dentro del año fiscal, pero es posterior al valor de `base_date`, por lo que cae fuera de la parte del año.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones entre 2020 y 2022, que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo, `in_year_to_date`, que determina qué transacciones se realizaron en el año hasta el 26 de julio de 2021.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `in_year_to_date`

Tabla de resultados

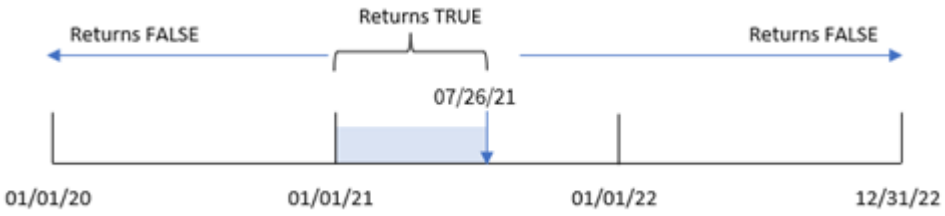
date	in_year_to_date
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0

date	in_year_to_date
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

El campo "`in_year_to_date`" se crea en la instrucción load anterior, mediante el uso de la función `inyeartodate()`. El primer argumento proporcionado identifica qué campo se está evaluando.

El segundo argumento es una fecha codificada para el 26 de julio de 2021, que es la `base_date` que identifica el límite final del segmento del año. Un `period_no` de 0 es el argumento final, lo que significa que la función no compara los años que preceden o siguen al año segmentado.

Diagrama de la función inyeartodate, sin argumentos adicionales



Cualquier transacción que ocurra entre el 1 de enero y el 26 de julio devuelve un resultado booleano de TRUE. Las fechas de transacciones anteriores a 2021 y posteriores al 26 de julio de 2021 devuelven FALSE.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `previous_year_to_date`, que determina qué transacciones se realizaron en un año completo antes del segmento de año que finaliza el 26 de julio de 2021.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

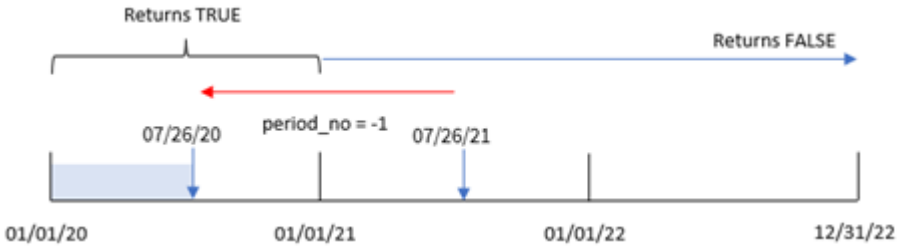
- `date`
- `previous_year_to_date`

Tabla de resultados

date	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

Un valor `period_no` de -1 indica que la función `inyeartodate ()` compara el segmento del trimestre introducido con el año anterior. Con una fecha de entrada del 26 de julio de 2021, el segmento del 1 de enero de 2021 al 26 de julio de 2021 se identificó inicialmente como el año hasta la fecha. El `period_no` luego desplaza este segmento a un año completo antes, lo que hace que los límites de fecha sean del 1 de enero al 26 de juli0 de 2020.

Diagrama de la función inyeartodate, ejemplo de period_no



Por lo tanto, cualquier transacción que ocurra entre el 1 de enero y el 26 de julio de 2020, devolverá un resultado booleano de TRUE.

Ejemplo 3: first_month_of_year

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `in_year_to_date`, que determina qué transacciones se realizaron en el mismo año hasta el 26 de julio de 2021.

En este ejemplo, establecemos marzo como el primer mes del año fiscal.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `in_year_to_date`

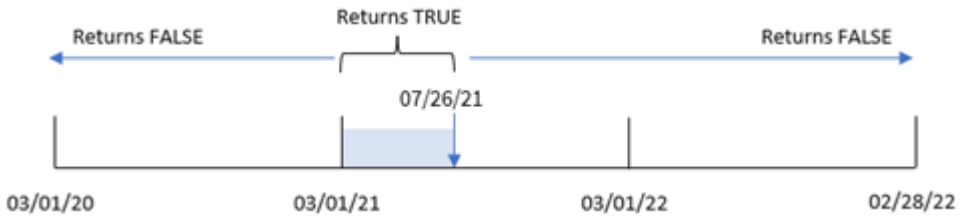
Tabla de resultados

date	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

date	in_year_to_date
07/18/2022	0
11/14/2022	0
12/12/2022	0

Usando 3 como el argumento de `first_month_of_year` en la función `inyeartodate()`, la función comienza el año el 1 de marzo. La `base_date` del 26 de julio de 2021 luego establece la fecha de finalización de ese segmento de año.

Diagrama de la función inyeartodate, ejemplo de first_month_of_year



En consecuencia, cualquier transacción que ocurra entre el 1 de marzo y el 26 de julio arrojará un resultado booleano de TRUE, mientras que las transacciones con fechas fuera de estos límites devolverán un valor de FALSE.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo. Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que determina qué transacciones tuvieron lugar en el mismo año hasta el 26 de julio de 2021 se crea como una medida en un objeto gráfico en la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:`date`.

Cree la siguiente medida:

```
=inyeartodate(date,'07/26/2021', 0)
```

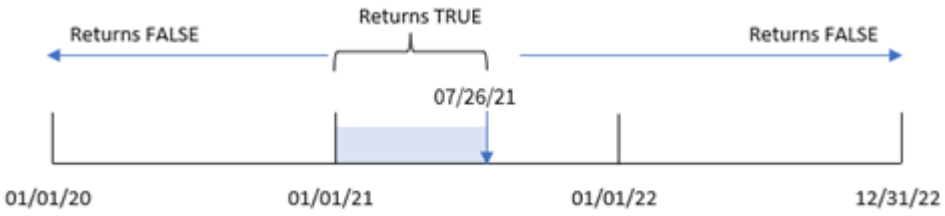
Tabla de resultados

date	=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

date	=inyeartodate(date,'07/26/2021', 0)
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

La medida `in_year_to_date` se crea en el objeto gráfico usando la función `inyeartodate()`. El primer argumento proporcionado identifica qué campo se está evaluando. El segundo argumento es una fecha codificada para el 26 de julio de 2021, que es la `base_date` que identifica el límite final del segmento del año de comparación. Un `period_no` de 0 es el argumento final, lo que significa que la función no compara los años que preceden o siguen al año segmentado.

Diagrama de la función inyeartodate, ejemplo de objeto gráfico



Cualquier transacción que ocurra entre el 1 de enero y el 26 de julio de 2021 devuelve un resultado booleano de TRUE. Las fechas de transacciones anteriores a 2021 y posteriores al 26 de julio de 2021 devuelven FALSE.

Ejemplo 5: Escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Products`.
- Información relativa a la identificación del producto, tipo de producto, fecha de fabricación y precio de coste.

Al usuario final le gustaría un objeto gráfico que muestre, por tipo de producto, el coste de producción de los productos fabricados en 2021 hasta el 26 de julio.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:`product_type`.

Cree una medida que calcule la suma de cada producto que se fabricó en 2021 antes del 27 de julio:

```
=sum(if(inyeartodate(manufacture_date,makedate(2021,07,26),0),cost_price,0))
```

Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

product_type	**=sum(if(inyeartodate(manufacture_date,makedate (2021,07,26),0),cost_price,0))**
producto A	$95.93
producto B	$128.66
producto C	$61.89
producto D	$146.09

La función `inyeartodate()` devuelve un valor booleano al evaluar las fechas de fabricación de cada uno de los productos. Para cualquier producto fabricado en 2021 antes del 27 de julio, la función `inyeartodate()` devuelve un valor booleano de `TRUE` y suma el `cost_price`.

El Producto D es el único producto que también se fabricó después del 26 de julio de 2021. La entrada con `product_ID` 8203 se fabricó el 27 de diciembre y costó 25,12 $. Por lo tanto, este costo no se incluyó en el total del Producto D en el objeto gráfico.

lastworkdate

La función **lastworkdate** devuelve la fecha más temprana de finalización para obtener el **no_of_workdays** (lunes-viernes) si comienza en **start_date** y teniendo en cuenta cualquier periodo vacacional opcionalmente indicado **holiday**. **start_date** y **holiday** deben ser fechas o fecha-hora válidas.

Sintaxis:

```
lastworkdate(start_date, no_of_workdays {, holiday})
```

Tipo de datos que devuelve: Entero

Un calendario que muestra cómo se utiliza la función `lastworkdate()`

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			

Limitaciones

No existe ningún método para modificar la función `lastworkdate()` para regiones o escenarios que impliquen otra cosa que no sea una semana laboral que comienza el lunes y finaliza el viernes.

El parámetro de vacaciones debe ser una constante de cadena de texto. No acepta una expresión.

Cuándo se utiliza

La función `lastworkdate()` se utiliza normalmente como parte de una expresión cuando el usuario desea calcular la fecha de finalización propuesta de un proyecto o asignación, en función de cuándo comienza el proyecto y las vacaciones que ocurrirán en dicho período.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Argumentos

Argumento	Descripción
start_date	La fecha inicial que se ha de evaluar.
no_of_ workdays	El número de días laborables que se ha de alcanzar.
holiday	Los períodos de vacaciones que deben excluirse de los días laborables. Las vacaciones se enuncian como cadena de fecha constante. Puede especificar más fechas de vacaciones separadas por comas. **Ejemplo:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Ejemplo 1: ejemplo básico

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene ID de proyectos, fechas de inicio de proyectos y el esfuerzo estimado en días requerido para los proyectos. El conjunto de datos se carga en una tabla denominada "`Projects`".

- Un load precedente que contiene la función `lastworkdate()`, configurada como el campo "`end_date`" e identifica cuándo está programado que finalice cada proyecto.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `start_date`
- `effort`
- `end_date`

Tabla de resultados

id	fecha_inicio	esfuerzo	fecha_fin
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/23/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

Debido a que no hay días de vacaciones programados, la función agrega el número definido de días laborables, de lunes a viernes, a la fecha de inicio para encontrar la fecha de finalización más temprana posible.

El calendario siguiente muestra la fecha de inicio y fin del proyecto 3, con los días laborables resaltados en verde.

Un calendario que muestra la fecha de inicio y fin del proyecto 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				

Ejemplo 2: Un día de vacaciones

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene ID de proyectos, fechas de inicio de proyectos y el esfuerzo estimado en días requerido para los proyectos. El conjunto de datos se carga en una tabla denominada "`Projects`".
- Un load precedente que contiene la función `lastworkdate()`, configurada como el campo "`end_date`" e identifica cuándo está programado que finalice cada proyecto.

Sin embargo, hay un día libre programado para el 18 de mayo de 2022. La función `lastworkdate()` en el load precedente incluye el día libre en su tercer argumento para identificar cuándo está programado que finalice cada proyecto.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `start_date`
- `effort`
- `end_date`

Tabla de resultados

id	fecha_inicio	esfuerzo	fecha_fin
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/24/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

El único día libre se indica como tercer argumento en la función `lastworkdate()`. Como resultado, la fecha de finalización del proyecto 3 se desplaza a un día después porque el día libre se produce en uno de los días laborables anteriores a la fecha de finalización.

El calendario siguiente muestra la fecha de inicio y finalización del proyecto 3 y muestra que el día libre cambia la fecha de finalización del proyecto en un día.

Un calendario que muestra la fecha de inicio y finalización del proyecto 3 con un día libre el 18 de mayo.

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				

Ejemplo 3: Vacaciones de varios días

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene ID de proyectos, fechas de inicio de proyectos y el esfuerzo estimado en días requerido para los proyectos. El conjunto de datos se carga en una tabla denominada "`Projects`".
- Un load precedente que contiene la función `lastworkdate()`, configurada como el campo "`end_date`" e identifica cuándo está programado que finalice cada proyecto.

No obstante, hay cuatro días de vacaciones programados para el 19, 20, 21 y 22 de mayo. La función `lastworkdate()` en la instrucción load precedente incluye cada uno de los días de vacaciones en su tercer argumento para identificar cuándo está programado que finalice cada proyecto.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `start_date`
- `effort`
- `end_date`

Tabla de resultados

id	fecha_inicio	esfuerzo	fecha_fin
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/25/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

Los cuatro días de vacaciones se indican como una lista de argumentos en la función `lastworkdate()` después de la fecha de inicio y el número de días hábiles laborables.

El siguiente calendario muestra la fecha de inicio y finalización del proyecto 3 y muestra que las vacaciones alteran la fecha de finalización del proyecto en tres días.

Un calendario que muestra la fecha de inicio y de finalización del proyecto 3 con vacaciones del 19 al 22 de mayo

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				

Ejemplo 4: Un día festivo o feriado (gráfico)

Script de carga y expresión de gráfico

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo el conjunto de datos está sin modificar y cargado en la app. El campo `end_date` se calcula como una medida en un gráfico.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `start_date`
- `effort`

Para calcular la fecha de finalización, cree la siguiente medida:

- `=LastWorkDate(start_date,effort,'05/18/2022')`

Tabla de resultados

id	fecha_inicio	esfuerzo	=LastWorkDate(start_date,effort,'05/18/2022')
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/23/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

El único día festivo o feriado programado se inserta como una medida en el gráfico. Como resultado, la fecha de finalización del proyecto 3 se desplaza a un día después porque el día libre se produce en uno de los días laborables anteriores a la fecha de finalización.

El calendario siguiente muestra la fecha de inicio y finalización del proyecto 3 y muestra que el día libre cambia la fecha de finalización del proyecto en un día.

Un calendario que muestra la fecha de inicio y finalización del proyecto 3 con un día libre el 18 de mayo.

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

Esta función devuelve una marca de tiempo con la hora actual para una zona horaria especificada.

Sintaxis:

```
LocalTime([timezone [, ignoreDST ]])
```

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
timezone	La zona **timezone** se especifica como una cadena que contiene cualquiera de los lugares geográficos enumerados en **Zona horaria** en el **Panel de control de Windows** para **Fecha y hora** o como una cadena en el formato "GMT+hh:mm". En la tabla siguiente también se presenta una lista de lugares y zonas horarias aceptadas. Si no se especifica ninguna zona horaria, devuelve la hora local. *Si utiliza un desplazamiento de DST (es decir, especifica un valor de argumento **ignoreDST** que devuelve False), debe especificar un lugar, en vez de un desplazamiento de GMT, en el argumento de **place**. Esto se debe a que ajustar el horario de verano requiere latitudinal, además de la información longitudinal proporcionada por un desplazamiento GMT. Para más información, vea Uso de compensaciones GMT en combinación con DST (page 864).*
ignoreDST	Si este argumento devuelve True, se ignora el horario de verano (DST). Los valores de argumento válidos que devuelven True incluyen `-1` y `True()`. Si este argumento se evalúa como False, la fecha-hora se ajusta al horario de verano. Los valores de argumento válidos que devuelven False incluyen `0` y `False()`. Si el valor del argumento **ignoreDST** no es válido, la función evalúa la expresión como si el valor de **ignore_dst** devolviera True. Si no se especifica el valor del argumento **ignoreDST**, la función evalúa la expresión como si el valor de **ignore_dst** devolviera False.

Lugares y zonas horarias válidos

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	-	-	-

Ejemplos y resultados:

Los ejemplos a continuación se basan en la función que se invoca el 2023-08-14 08:39:47 hora local, con la zona horaria local del servidor o entorno de escritorio siendo GMT-05:00, y en una región que ha implementado la luz del día ahorrando tiempo a partir de esta fecha indicada.

Ejemplos de script

Ejemplo	Resultado
`localtime ()`	Devuelve la hora local 2023-08-14 08:39:47.
`localtime ('London')`	Devuelve la hora local en Londres, 2023-08-14 13:39:47.
`localtime ('GMT+02:00')`	Devuelve la hora local en la zona horaria de GMT+02:00, 2023-08-14 14:39:47. No se realiza ningún ajuste para el horario de verano porque se especifica un desplazamiento GMT, en lugar de una ubicación.
`localtime ('Paris',-1)`	Devuelve la hora local de París ignorando el horario de verano, 2023-08-14 13:39:47.
`localtime ('Paris',True())`	Devuelve la hora local de París ignorando el horario de verano, 2023-08-14 13:39:47.
`localtime ('Paris',0)`	Devuelve la hora local de París, teniendo en cuenta el horario de verano, 2023-08-14 14:39:47.
`localtime ('Paris',False ())`	Devuelve la hora local de París, teniendo en cuenta el horario de verano, 2023-08-14 14:39:47.

Uso de compensaciones GMT en combinación con DST

Tras la implementación de las bibliotecas de componentes internacionales para Unicode (ICU) en Qlik Sense, el uso de compensaciones GMT (hora media de Greenwich) en combinación con DST (horario de verano) requiere información adicional de latitud.

GMT supone un desplazamiento longitudinal (este-oeste), mientras que DST representa un desplazamiento latitudinal (norte-sur). Por ejemplo, Helsinki (Finlandia) y Johannesburgo (Sudáfrica) comparten la misma diferencia GMT+02:00, pero no comparten la misma diferencia DST. Esto significa que, además del desfase de GMT, cualquier desfase de DST requiere información sobre la posición latitudinal de la zona horaria local (datos de zona horaria geográfica) para tener información completa sobre las condiciones locales de DST.

lunarweekend

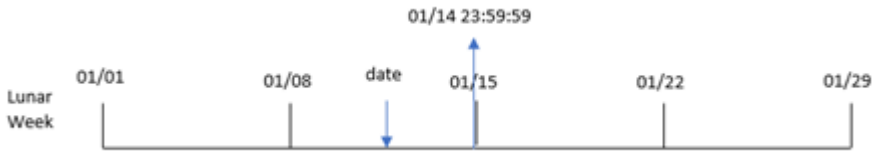
Esta función devuelve un valor correspondiente a una marca de tiempo del último milisegundo del último día de la semana lunar que contiene a **date**. Las semanas lunares en Qlik Sense se definen contando el 1 de enero como el primer día de la semana y, aparte de la última semana del año, contendrán exactamente siete días.

Sintaxis:

```
LunarweekEnd(date[, period_no[, first_week_day]])
```

Tipo de datos que devuelve: dual

Ejemplo de diagrama de la función `lunarweekend()`



La función `lunarweekend()` determina en qué semana lunar cae `date`. Luego devuelve una marca de tiempo, en formato de fecha, del último milisegundo de esa semana.

Argumentos

Argumento	Descripción
date	La fecha o marca de tiempo para evaluar.
period_no	**period_no** es un entero o expresión que devuelve un entero, donde el valor 0 indica la semana lunar que contiene a **date**. Los valores negativos en **period_no** indican semanas lunares precedentes y los valores positivos indican semanas lunares subsiguientes.
first_week_day	Un desplazamiento que puede ser mayor que o menor que cero. Esto cambia el comienzo del año por el número especificado de días y/o fracciones de un día.

Cuándo se utiliza

La función `lunarweekend()` se suele utilizar como parte de una expresión cuando el usuario desea que el cálculo utilice la fracción del día que aún no ha transcurrido. A diferencia de la función `weekend()`, la última semana lunar de cada año natural finalizará el 31 de diciembre. Por ejemplo, la función `lunarweekend()` se puede utilizar para calcular los intereses que aún no se han devengado durante la semana.

Ejemplos de funciones

Ejemplo	Resultado
`lunarweekend('01/12/2013')`	Devuelve 01/14/2013 23:59:59.
`lunarweekend('01/12/2013', -1)`	Devuelve 01/07/2013 23:59:59.
`lunarweekend('01/12/2013', 0, 1)`	Devuelve 01/15/2013 23:59:59.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo, `end_of_week`, que devuelve una marca de tiempo del final de la semana lunar en que las transacciones tuvieron lugar.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

Tabla de resultados

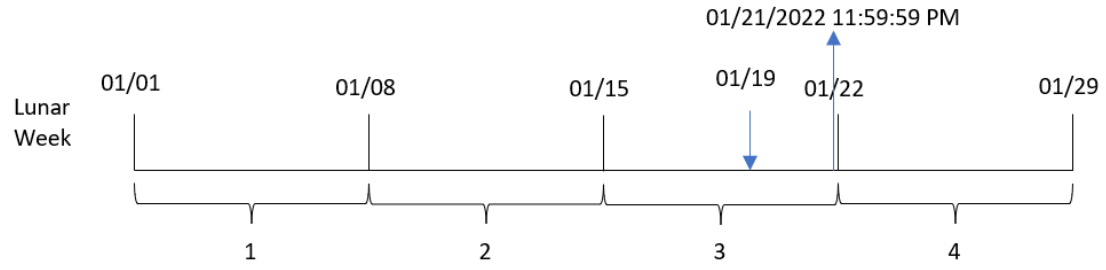
date	end_of_week	end_of_week_timestamp
1/7/2022	01/07/2022	1/7/2022 11:59:59 PM
1/19/2022	01/21/2022	1/21/2022 11:59:59 PM
2/5/2022	02/11/2022	2/11/2022 11:59:59 PM
2/28/2022	03/04/2022	3/4/2022 11:59:59 PM
3/16/2022	03/18/2022	3/18/2022 11:59:59 PM
4/1/2022	04/01/2022	4/1/2022 11:59:59 PM
5/7/2022	05/13/2022	5/13/2022 11:59:59 PM
5/16/2022	05/20/2022	5/20/2022 11:59:59 PM
6/15/2022	06/17/2022	6/17/2022 11:59:59 PM
6/26/2022	07/01/2022	7/1/2022 11:59:59 PM
7/9/2022	07/15/2022	7/15/2022 11:59:59 PM
7/22/2022	07/22/2022	7/22/2022 11:59:59 PM
7/23/2022	07/29/2022	7/29/2022 11:59:59 PM

date	end_of_week	end_of_week_timestamp
7/27/2022	07/29/2022	7/29/2022 11:59:59 PM
8/2/2022	08/05/2022	8/5/2022 11:59:59 PM
8/8/2022	08/12/2022	8/12/2022 11:59:59 PM
8/19/2022	08/19/2022	8/19/2022 11:59:59 PM
9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
10/14/2022	10/14/2022	10/14/2022 11:59:59 PM
10/29/2022	11/04/2022	11/4/2022 11:59:59 PM

El campo `end_of_week` se crea en la instrucción load anterior utilizando la función `lunarweekend()` e introduciendo el campo `date` como argumento de la función.

La función `lunarweekend()` identifica en qué semana lunar cae el valor de la fecha y devuelve una marca de tiempo del último milisegundo de esa semana.

Diagrama de la función `lunarweekend()`, ejemplo sin argumentos adicionales



La transacción 8189 tuvo lugar el 19 de enero. La función `lunarweekend()` identifica que la semana lunar comienza el 15 de enero. Por lo tanto, el valor `end_of_week` de esa transacción devuelve el último milisegundo de la semana lunar, que es el 21 de enero a las 23:59:59.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `previous_lunar_week_end`, que devuelve la marca de tiempo del final de la semana lunar anterior a la transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- previous_lunar_week_end
- previous_lunar_week_end_timestamp

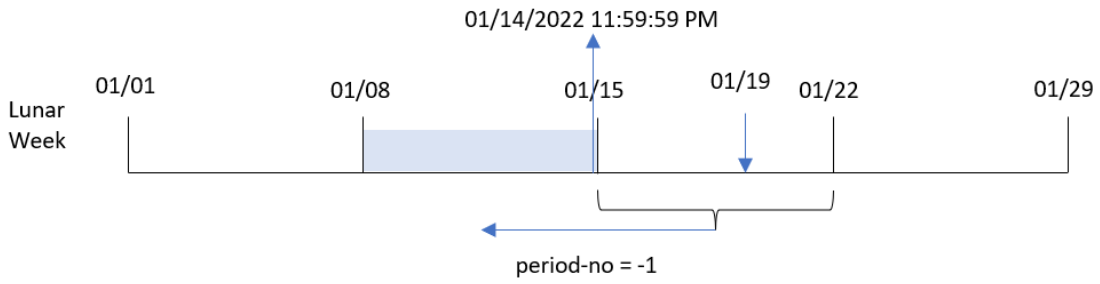
Tabla de resultados

date	previous_lunar_week_end	previous_lunar_week_end_timestamp
1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
1/19/2022	01/14/2022	1/14/2022 11:59:59 PM

date	previous_lunar_week_end	previous_lunar_week_end_timestamp
2/5/2022	02/04/2022	2/4/2022 11:59:59 PM
2/28/2022	02/25/2022	2/25/2022 11:59:59 PM
3/16/2022	03/11/2022	3/18/2022 11:59:59 PM
4/1/2022	03/25/2022	3/25/2022 11:59:59 PM
5/7/2022	05/06/2022	5/6/2022 11:59:59 PM
5/16/2022	05/13/2022	5/13/2022 11:59:59 PM
6/15/2022	06/10/2022	6/10/2022 11:59:59 PM
6/26/2022	06/24/2022	6/24/2022 11:59:59 PM
7/9/2022	07/08/2022	7/8/2022 11:59:59 PM
7/22/2022	07/15/2022	7/15/2022 11:59:59 PM
7/23/2022	07/22/2022	7/22/2022 11:59:59 PM
7/27/2022	07/22/2022	7/22/2022 11:59:59 PM
8/2/2022	07/29/2022	7/29/2022 11:59:59 PM
8/8/2022	08/05/2022	8/5/2022 11:59:59 PM
8/19/2022	08/12/2022	8/12/2022 11:59:59 PM
9/26/2022	09/23/2022	9/23/2022 11:59:59 PM
10/14/2022	10/07/2022	10/7/2022 11:59:59 PM
10/29/2022	10/28/2022	10/28/2022 11:59:59 PM

En este caso, debido a que se usó un `period_no` de -1 como argumento del desplazamiento en la función `lunarweekend()`, la función primero identifica la semana lunar en la que se realizaron las transacciones. Luego se desplaza a una semana antes e identifica el último milisegundo de esa semana.

Diagrama de la función `lunarweekend()`, ejemplo de period_no



La transacción 8189 tuvo lugar el 19 de enero. La función `lunarweekend()` identifica que la semana lunar comienza el 15 de enero. Por lo tanto, la semana lunar anterior comenzó el 8 de enero y finalizó el 14 de enero a las 23:59:59; este es el valor que devuelve para el campo `previous_lunar_week_end`.

Ejemplo: first_week_day

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo. En este ejemplo, establecemos que las semanas lunares comiencen el 5 de enero.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

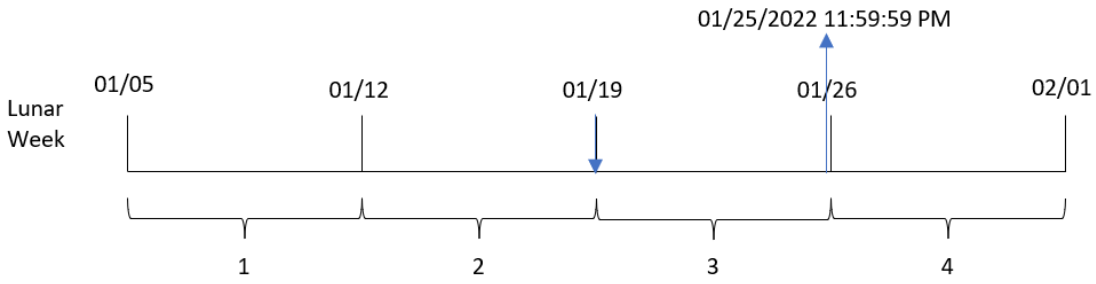
- date
- end_of_week
- end_of_week_timestamp

Tabla de resultados

date	end_of_week	end_of_week_timestamp
1/7/2022	01/11/2022	1/11/2022 11:59:59 PM
1/19/2022	01/25/2022	1/25/2022 11:59:59 PM
2/5/2022	02/08/2022	2/8/2022 11:59:59 PM
2/28/2022	03/01/2022	3/1/2022 11:59:59 PM
3/16/2022	03/22/2022	3/22/2022 11:59:59 PM
4/1/2022	04/05/2022	4/5/2022 11:59:59 PM
5/7/2022	05/10/2022	5/10/2022 11:59:59 PM
5/16/2022	05/17/2022	5/17/2022 11:59:59 PM
6/15/2022	06/21/2022	6/21/2022 11:59:59 PM
6/26/2022	06/28/2022	6/28/2022 11:59:59 PM
7/9/2022	07/12/2022	7/12/2022 11:59:59 PM
7/22/2022	07/26/2022	7/26/2022 11:59:59 PM
7/23/2022	07/26/2022	7/26/2022 11:59:59 PM
7/27/2022	08/02/2022	8/2/2022 11:59:59 PM
8/2/2022	08/02/2022	8/2/2022 11:59:59 PM
8/8/2022	08/09/2022	8/9/2022 11:59:59 PM
8/19/2022	08/23/2022	8/23/2022 11:59:59 PM
9/26/2022	09/27/2022	9/27/2022 11:59:59 PM
10/14/2022	10/18/2022	10/18/2022 11:59:59 PM
10/29/2022	11/01/2022	11/1/2022 11:59:59 PM

En este caso, como el argumento `first_week_date` de 4 se usa en la función `lunarweekend()`, desplaza el inicio del año del 1 de enero al 5 de enero.

Diagrama de la función `lunarweekend()`, ejemplo de first_week_day



La transacción 8189 tuvo lugar el 19 de enero. Debido a que las semanas lunares comienzan el 5 de enero, la función `lunarweekend()` identifica que la semana lunar que contiene el 19 de enero también comienza el 19 de enero. Por lo tanto, el final de esa semana lunar se produce el 25 de enero a las 23:59:59; este es el valor que devuelve para el campo `end_of_week`.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo, que devuelve una marca de tiempo con el final de la semana lunar en que se realizaron las transacciones, se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Agregue las siguientes medidas:

```
=lunarweekend(date)
```

```
=timestamp(lunarweekend(date))
```

Tabla de resultados

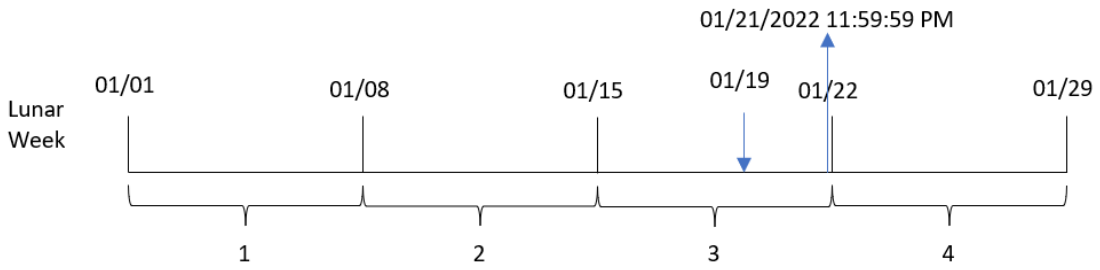
date	=lunarweekend(date)	=timestamp(lunarweekend(date))
1/7/2022	01/07/2022	1/7/2022 11:59:59 PM
1/19/2022	01/21/2022	1/21/2022 11:59:59 PM
2/5/2022	02/11/2022	2/11/2022 11:59:59 PM
2/28/2022	03/04/2022	3/4/2022 11:59:59 PM
3/16/2022	03/18/2022	3/18/2022 11:59:59 PM
4/1/2022	04/01/2022	4/1/2022 11:59:59 PM
5/7/2022	05/13/2022	5/13/2022 11:59:59 PM
5/16/2022	05/20/2022	5/20/2022 11:59:59 PM
6/15/2022	06/17/2022	6/17/2022 11:59:59 PM
6/26/2022	07/01/2022	7/1/2022 11:59:59 PM
7/9/2022	07/15/2022	7/15/2022 11:59:59 PM
7/22/2022	07/22/2022	7/22/2022 11:59:59 PM
7/23/2022	07/29/2022	7/29/2022 11:59:59 PM
7/27/2022	07/29/2022	7/29/2022 11:59:59 PM
8/2/2022	08/05/2022	8/5/2022 11:59:59 PM
8/8/2022	08/12/2022	8/12/2022 11:59:59 PM
8/19/2022	08/19/2022	8/19/2022 11:59:59 PM
9/26/2022	09/30/2022	9/30/2022 11:59:59 PM

date	=lunarweekend(date)	=timestamp(lunarweekend(date))
10/14/2022	10/14/2022	10/14/2022 11:59:59 PM
10/29/2022	11/04/2022	11/4/2022 11:59:59 PM

La medida `end_of_week` se crea en el objeto gráfico utilizando la función `lunarweekend()` e introduciendo el campo `date` como argumento de la función.

La función `lunarweekend()` identifica en qué semana lunar cae el valor de la fecha y devuelve una marca de tiempo del último milisegundo de esa semana.

Diagrama de la función `lunarweekend()`, ejemplo de objeto gráfico



La transacción 8189 tuvo lugar el 19 de enero. La función `lunarweekend()` identifica que la semana lunar comienza el 15 de enero. Por lo tanto, el valor `end_of_week` de esa transacción devuelve el último milisegundo de la semana lunar, que es el 21 de enero a las 23:59:59.

Ejemplo 5: Escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Employee_Expenses`.
- Las ID de los empleados, los nombres de empleado y las reclamaciones o declaraciones de gastos diarios promedio de cada empleado.

Al usuario final le gustaría tener un objeto gráfico que muestre, por ID de empleado y nombre de empleado, las reclamaciones de gastos estimados que faltan por presentar durante el resto de la semana lunar.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla.
2. Agregue los siguientes campos como dimensiones.
    - `employee_id`
    - `employee_name`
3. A continuación, cree la siguiente medida para calcular el interés acumulado:
   `=(lunarweekend(today(1))-today(1))*avg_daily_claim`
4. Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

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

Al utilizar la fecha de hoy como único argumento, la función `lunarkweekend()` devuelve la fecha de finalización de la semana lunar actual. Luego, al restar la fecha de hoy de la fecha de finalización de la semana lunar, la expresión devuelve el número de días que quedan de esta semana.

A continuación, este valor se multiplica por la reclamación de gastos diarios promedio de cada empleado para calcular el valor estimado de las reclamaciones que se espera que haga cada empleado en la semana lunar restante.

lunarweekname

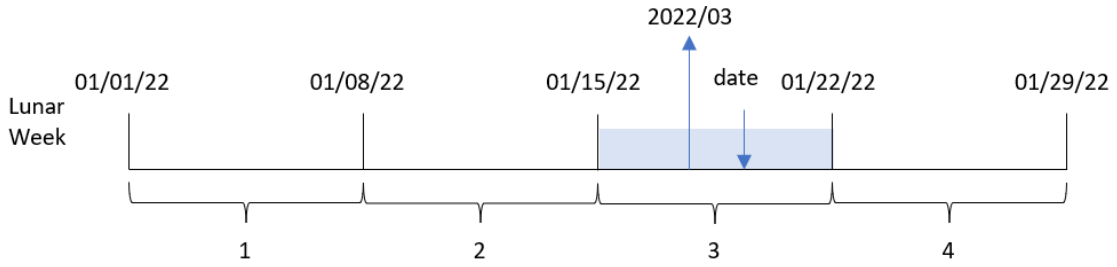
Esta función devuelve un valor de visualización que muestra el año y el número de la semana lunar correspondiente a una marca de tiempo (fecha-hora) del primer milisegundo del primer día de la semana lunar que contiene a **date**. Las semanas lunares en Qlik Sense se definen contando el 1 de enero como el primer día de la semana y, aparte de la última semana del año, contendrán exactamente siete días.

Sintaxis:

```
LunarWeekName(date [, period_no[, first_week_day]])
```

Tipo de datos que devuelve: dual

Ejemplo de diagrama de la función `lunarweekname()`



La función `lunarweekname()` determina en qué semana lunar cae la fecha, comenzando un recuento de semanas desde el 1 de enero. Luego devuelve un valor compuesto por `year/weekcount`.

Argumentos

Argumento	Descripción
date	La fecha o marca de tiempo para evaluar.
period_no	**period_no** es un entero o expresión que devuelve un entero, donde el valor 0 indica la semana lunar que contiene a **date**. Los valores negativos en **period_no** indican semanas lunares precedentes y los valores positivos indican semanas lunares subsiguientes.
first_week_day	Un desplazamiento que puede ser mayor que o menor que cero. Esto cambia el comienzo del año por el número especificado de días y/o fracciones de un día.

Cuándo se utiliza

La función `lunarweekname()` es útil cuando se desea comparar agregaciones por semanas lunares. Por ejemplo, la función podría usarse para determinar el total de ventas de productos por semana lunar. Las semanas lunares son útiles cuando se desea garantizar que todos los valores contenidos en la primera semana del año contengan solo valores a partir del 1 de enero como mínimo.

Estas dimensiones se pueden crear en el script de carga utilizando la función para crear un campo en una tabla de calendario maestro. La función también se puede utilizar directamente en un gráfico como una dimensión calculada.

Ejemplos de funciones

Ejemplo	Resultado
`lunarweekname('01/12/2013')`	Devuelve 2006/02.
`lunarweekname('01/12/2013', -1)`	Devuelve 2006/01.
`lunarweekname('01/12/2013', 0, 1)`	Devuelve 2006/02.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: la fecha sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo, `lunar_week_name`, que devuelve el año y el número de semana de la semana lunar en la que se realizaron las transacciones.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `lunar_week_name`

Tabla de resultados

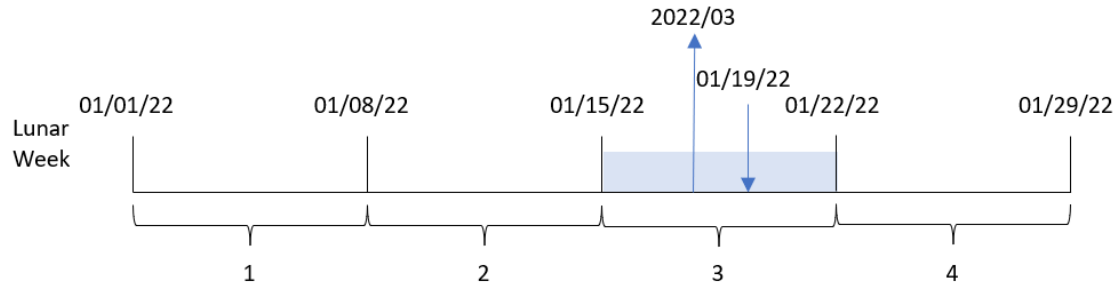
date	lunar_week_name
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/06
2/28/2022	2022/09
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/19
5/16/2022	2022/20
6/15/2022	2022/24
6/26/2022	2022/26
7/9/2022	2022/28
7/22/2022	2022/29
7/23/2022	2022/30
7/27/2022	2022/30

date	lunar_week_name
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

El campo `lunar_week_name` se crea en la instrucción load anterior utilizando la función `lunarweekname()` e introduciendo el campo `date` como argumento de la función.

La función `lunarweekname()` identifica en qué semana lunar cae el valor de la fecha, devolviendo el año y el número de semana de esa fecha.

Diagrama de la función `lunarweekname()`, ejemplo sin argumentos adicionales



La transacción 8189 tuvo lugar el 19 de enero. La función `lunarweekname()` identifica que esta fecha cae en la semana lunar que comienza el 15 de enero; esta es la tercera semana lunar del año. Por lo tanto, el valor de `lunar_week_name` que devuelve para esa transacción es 2022/03.

Ejemplo 2: fecha con argumento period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `previous_lunar_week_name`, que devuelve el año y el número de semana de la semana lunar anterior a la fecha en que se realizaron las transacciones.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `previous_lunar_week_name`

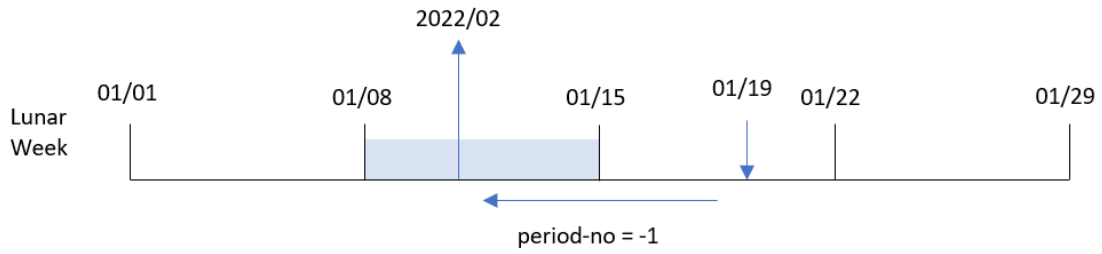
Tabla de resultados

date	previous_lunar_week_name
1/7/2022	2021/52
1/19/2022	2022/02
2/5/2022	2022/05
2/28/2022	2022/08

date	previous_lunar_week_name
3/16/2022	2022/10
4/1/2022	2022/12
5/7/2022	2022/18
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

En este caso, debido a que se usó un `period_no` de -1 como argumento del desplazamiento en la función `lunarweekname()`, la función primero identifica la semana lunar en la que se realizaron las transacciones. Luego devuelve el año y el número de una semana anterior.

Diagrama de la función `lunarweekname()`, ejemplo de period_no



La transacción 8189 tuvo lugar el 19 de enero. La función `lunarweekname()` identifica que esta transacción tuvo lugar en la tercera semana lunar del año, por lo que luego devuelve el año y el valor de una semana anterior, 2022/02, para el campo `previous_lunar_week_name`.

Ejemplo 3: fecha con el argumento de first_week_day

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo. En este ejemplo, establecemos que las semanas lunares comiencen el 5 de enero.

Script de carga

```
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
];
```

Resultados

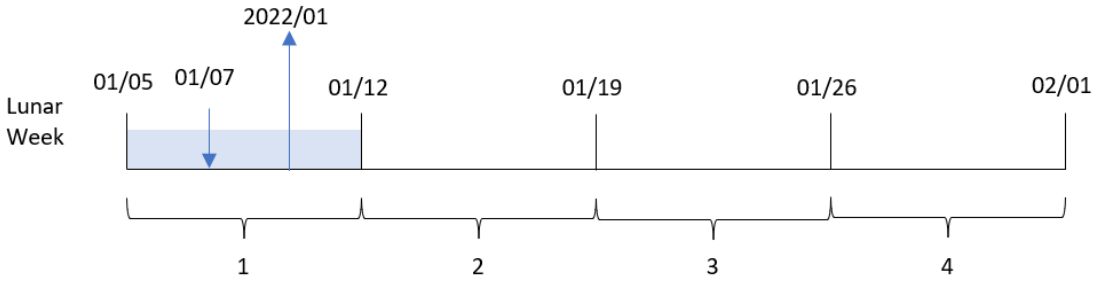
Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- lunar_week_name

Tabla de resultados

date	lunar_week_name
1/7/2022	2022/01
1/19/2022	2022/03
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

Diagrama de la función `lunarweekname()`, ejemplo de first_week_day



En este caso, como se utiliza el argumento 4 de `first_week_date` en la función `lunarweekname()`, esta desplaza el inicio de las semanas lunares del 1 de enero al 5 de enero.

La transacción 8188 tuvo lugar el 7 de enero. Debido a que las semanas lunares comienzan el 5 de enero, la función `lunarweekname()` identifica que la semana lunar que contiene el 7 de enero es la primera semana lunar del año. Por lo tanto, el valor de `lunar_week_name` que devuelve para esa transacción es 2022/01.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que devuelve el número de la semana lunar y el año en que se realizaron las transacciones se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Para calcular la fecha de inicio de la semana lunar en la que se realizó una transacción, cree la siguiente medida:

```
=lunarweekname(date)
```

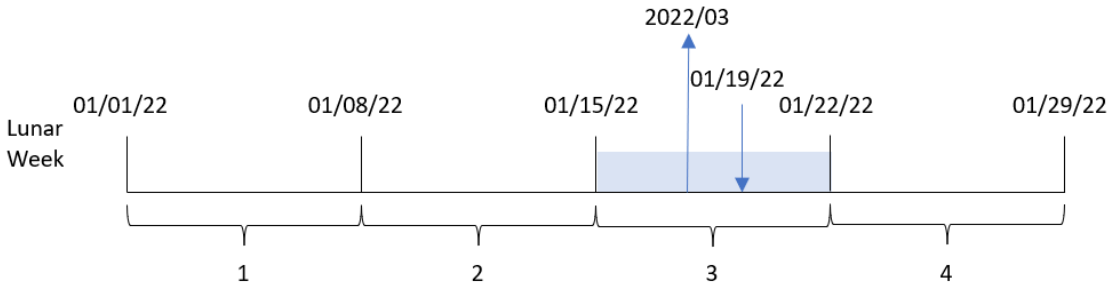
Tabla de resultados

date	=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

La medida `lunar_week_name` se crea en el objeto gráfico utilizando la función `lunarweekname()` e introduciendo el campo `date` como argumento de la función.

La función `lunarweekname()` identifica en qué semana lunar cae el valor de la fecha, devolviendo el año y el número de semana de esa fecha.

Diagrama de la función `lunarweekname()`, ejemplo de objeto gráfico



La transacción 8189 tuvo lugar el 19 de enero. La función `lunarweekname()` identifica que esta fecha cae en la semana lunar que comienza el 15 de enero; esta es la tercera semana lunar del año. Por lo tanto, el valor de `lunar_week_name` para dicha transacción es 2022/03.

Ejemplo 5: Escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada `Transactions`.
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat` (MM/DD/AAAA).

Al usuario final le gustaría tener un objeto gráfico que presente el total de ventas por semana del año actual. La semana 1, con una duración de siete días, debe comenzar el 1 de enero. Esto podría lograrse incluso cuando esta dimensión no esté disponible en el modelo de datos utilizando la función `lunarweekname()` como una dimensión calculada en el gráfico.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla.
2. Cree una dimensión calculada usando la siguiente expresión:
   `=lunarweekname(date)`
3. Calcule el total de ventas con la siguiente medida de agregación:
   `=sum(amount)`
4. Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

=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

Esta función devuelve un valor correspondiente a una marca de tiempo del primer milisegundo del primer día de la semana lunar que contiene a **date**. Las semanas lunares en Qlik Sense se definen contando el 1 de enero como el primer día de la semana y, aparte de la última semana del año, contendrán exactamente siete días.

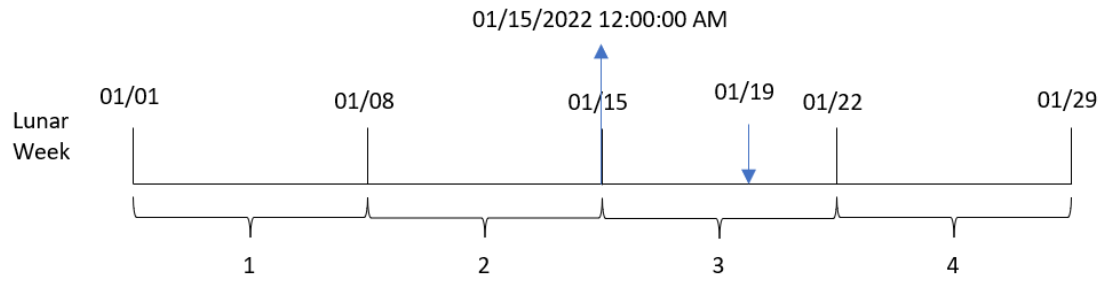
Sintaxis:

```
LunarweekStart(date[, period_no[, first_week_day]])
```

Tipo de datos que devuelve: dual

La función `lunarweekstart()` determina en qué semana lunar cae `date`. Luego devuelve una marca de tiempo, en formato de fecha, con el primer milisegundo de esa semana.

Ejemplo de diagrama de la función `lunarweekstart()`



Argumentos

Argumento	Descripción
date	La fecha o marca de tiempo para evaluar.
period_no	**period_no** es un entero o expresión que devuelve un entero, donde el valor 0 indica la semana lunar que contiene a **date**. Los valores negativos en **period_no** indican semanas lunares precedentes y los valores positivos indican semanas lunares subsiguientes.
first_week_day	Un desplazamiento que puede ser mayor que o menor que cero. Esto cambia el comienzo del año por el número especificado de días y/o fracciones de un día.

Cuándo se utiliza

La función `lunarweekstart()` se suele utilizar como parte de una expresión cuando el usuario desea que el cálculo utilice la fracción de la semana que ya ha transcurrido. A diferencia de la función `weekstart()`, al comienzo de cada nuevo año natural, las semanas comienzan el 1 de enero y cada semana subsiguiente comienza siete días después. La función `lunarweekstart()` no se ve afectada por la variable del sistema `FirstWeekDay`.

Por ejemplo, `lunarweekstart()` se puede utilizar para calcular el interés acumulado en una semana hasta la fecha.

Ejemplos de funciones

Ejemplo	Resultado
`lunarweekstart ('01/12/2013')`	Devuelve 01/08/2013.
`lunarweekstart ('01/12/2013', -1)`	Devuelve 01/01/2013.
`lunarweekstart ('01/12/2013', 0, 1 )`	Devuelve 01/09/2013, porque configurar first_week_day en 1 quiere decir que el inicio del año se pasa al 01/02/2013.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "Transactions".
- El campo de fecha proporcionado en el formato de la variable del sistema DateFormat: (MM/DD/AAAA).
- La creación de un campo, start_of_week, que devuelve una marca de tiempo con el inicio de la semana lunar en la que se realizaron las transacciones.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- start_of_week
- start_of_week_timestamp

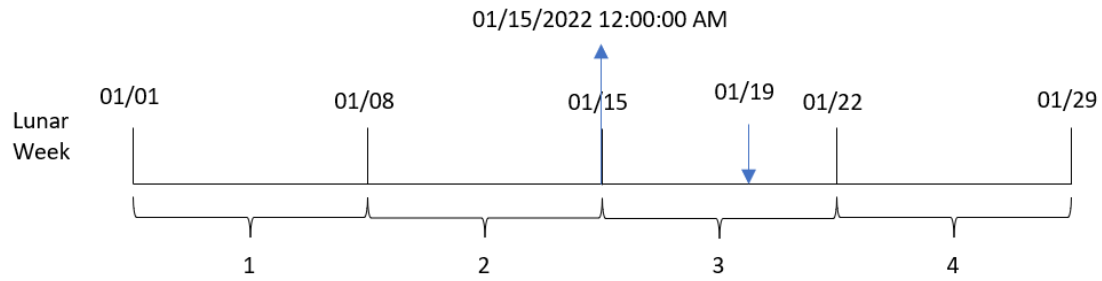
Tabla de resultados

date	start_of_week	start_of_week_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/15/2022	1/15/2022 12:00:00 AM
2/5/2022	02/05/2022	2/5/2022 12:00:00 AM
2/28/2022	02/26/2022	2/26/2022 12:00:00 AM
3/16/2022	03/12/2022	3/12/2022 12:00:00 AM
4/1/2022	03/26/2022	3/26/2022 12:00:00 AM
5/7/2022	05/07/2022	5/7/2022 12:00:00 AM
5/16/2022	05/14/2022	5/14/2022 12:00:00 AM
6/15/2022	06/11/2022	6/11/2022 12:00:00 AM
6/26/2022	06/25/2022	6/25/2022 12:00:00 AM
7/9/2022	07/09/2022	7/9/2022 12:00:00 AM
7/22/2022	07/16/2022	7/16/2022 12:00:00 AM
7/23/2022	07/23/2022	7/23/2022 12:00:00 AM
7/27/2022	07/23/2022	7/23/2022 12:00:00 AM
8/2/2022	07/30/2022	7/30/2022 12:00:00 AM
8/8/2022	08/06/2022	8/6/2022 12:00:00 AM
8/19/2022	08/13/2022	8/13/2022 12:00:00 AM
9/26/2022	09/24/2022	9/24/2022 12:00:00 AM
10/14/2022	10/08/2022	10/8/2022 12:00:00 AM
10/29/2022	10/29/2022	10/29/2022 12:00:00 AM

El campo `start_of_week` se crea en la instrucción load anterior utilizando la función `lunarweekstart()` e introduciendo el campo `date` como argumento de la función.

La función `lunarweekstart()` identifica en qué semana lunar cae el valor de la fecha y devuelve una marca de tiempo del primer milisegundo de esa semana.

Diagrama de la función `lunarweekstart()`, ejemplo sin argumentos adicionales



La transacción 8189 tuvo lugar el 19 de enero. La función `lunarweekstart()` identifica que la semana lunar comienza el 15 de enero. Por lo tanto, el valor `start_of_week` de esa transacción devuelve el primer milisegundo de ese día, que es el 15 de enero a las 12:00:00 a. m.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `previous_lunar_week_start`, que devuelve la marca de tiempo del inicio de la semana lunar anterior a la transacción.

Script de carga

```
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
];
```

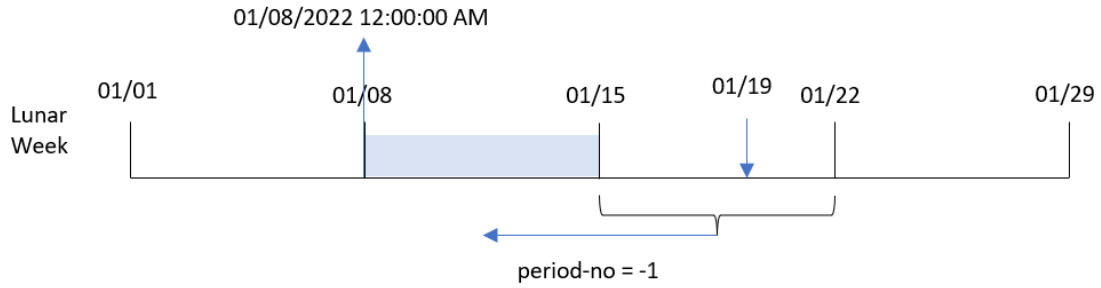
Resultados

Tabla de resultados

date	previous_lunar_week_start	previous_lunar_week_start_timestamp
1/7/2022	12/24/2021	12/24/2021 12:00:00 AM
1/19/2022	01/08/2022	1/8/2022 12:00:00 AM
2/5/2022	01/29/2022	1/29/2022 12:00:00 AM
2/28/2022	02/19/2022	2/19/2022 12:00:00 AM
3/16/2022	03/05/2022	3/5/2022 12:00:00 AM
4/1/2022	03/19/2022	3/19/2022 12:00:00 AM
5/7/2022	04/30/2022	4/30/2022 12:00:00 AM
5/16/2022	05/07/2022	5/7/2022 12:00:00 AM
6/15/2022	06/04/2022	6/4/2022 12:00:00 AM
6/26/2022	06/18/2022	6/18/2022 12:00:00 AM
7/9/2022	07/02/2022	7/2/2022 12:00:00 AM
7/22/2022	07/09/2022	7/9/2022 12:00:00 AM
7/23/2022	07/16/2022	7/16/2022 12:00:00 AM
7/27/2022	07/16/2022	7/16/2022 12:00:00 AM
8/2/2022	07/23/2022	7/23/2022 12:00:00 AM
8/8/2022	07/30/2022	7/30/2022 12:00:00 AM
8/19/2022	08/06/2022	8/6/2022 12:00:00 AM
9/26/2022	09/17/2022	9/17/2022 12:00:00 AM
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

En este caso, como se usó un `period_no` de -1 como el argumento del desplazamiento en la función `lunarweekstart()`, la función primero identifica la semana lunar en la que se realizan las transacciones. Luego cambia a una semana antes e identifica el primer milisegundo de esa semana lunar.

Diagrama de la función `lunarweekstart()`, ejemplo de period_no



La transacción 8189 tuvo lugar el 19 de enero. La función `lunarweekstart()` identifica que la semana lunar comienza el 15 de enero. Por lo tanto, la semana lunar anterior comenzó el 8 de enero a las 12:00:00; este es el valor que devuelve para el campo `previous_lunar_week_start`.

Ejemplo: first_week_day

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo. En este ejemplo, establecemos que las semanas lunares comiencen el 5 de enero.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

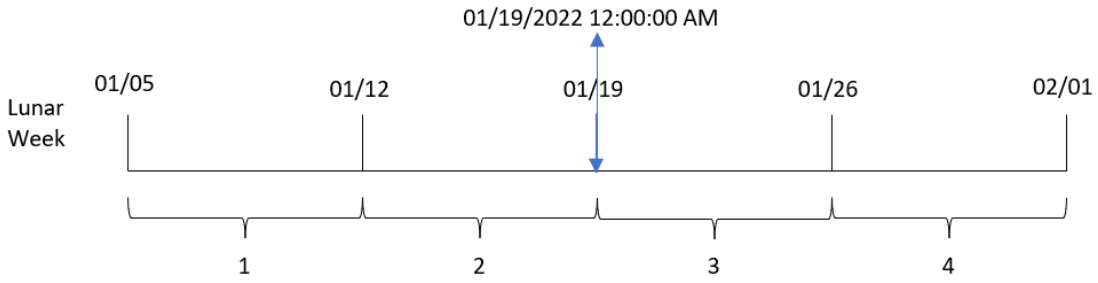
Tabla de resultados

date	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	2/2/2022 12:00:00 AM
2/28/2022	02/23/2022	2/23/2022 12:00:00 AM
3/16/2022	03/16/2022	3/16/2022 12:00:00 AM
4/1/2022	03/30/2022	3/30/2022 12:00:00 AM
5/7/2022	05/04/2022	5/4/2022 12:00:00 AM
5/16/2022	05/11/2022	5/11/2022 12:00:00 AM
6/15/2022	06/15/2022	6/15/2022 12:00:00 AM
6/26/2022	06/22/2022	6/22/2022 12:00:00 AM
7/9/2022	07/06/2022	7/6/2022 12:00:00 AM
7/22/2022	07/20/2022	7/20/2022 12:00:00 AM
7/23/2022	07/20/2022	7/20/2022 12:00:00 AM
7/27/2022	07/27/2022	7/27/2022 12:00:00 AM
8/2/2022	07/27/2022	7/27/2022 12:00:00 AM

date	start_of_week	start_of_week_timestamp
8/8/2022	08/03/2022	8/3/2022 12:00:00 AM
8/19/2022	08/17/2022	8/17/2022 12:00:00 AM
9/26/2022	09/21/2022	9/21/2022 12:00:00 AM
10/14/2022	10/12/2022	10/12/2022 12:00:00 AM
10/29/2022	10/26/2022	10/26/2022 12:00:00 AM

En este caso, como el argumento `first_week_date` de 4 se usa en la función `lunarweekstart()`, desplaza el inicio del año del 1 de enero al 5 de enero.

Diagrama de la función `lunarweekstart()`, ejemplo de first_week_day



La transacción 8189 tuvo lugar el 19 de enero. Debido a que las semanas lunares comienzan el 5 de enero, la función `lunarweekstart()` identifica que la semana lunar que contiene el 19 de enero comienza el 19 de enero a las 12:00:00 a. m. Por lo tanto, ese es el valor que devuelve para el campo `start_of_week`.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que devuelve una marca de tiempo del inicio de la semana lunar en que se realizaron las transacciones se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Agregue las siguientes medidas:

```
=lunarweekstart(date)
```

```
=timestamp(lunarweekstart(date))
```

Tabla de resultados

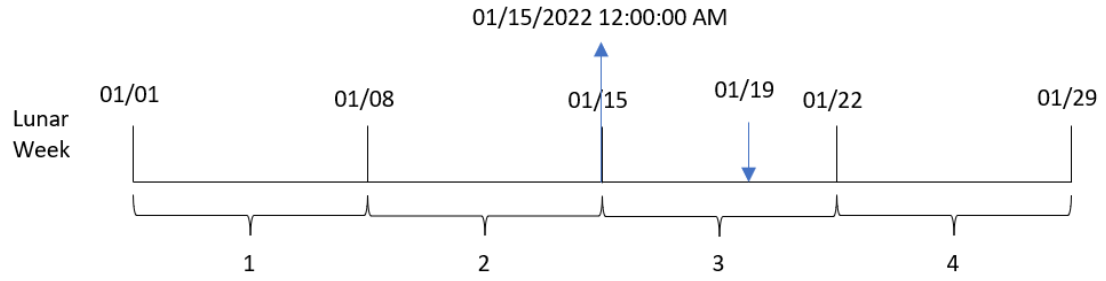
date	=lunarweekstart(date)	=timestamp(lunarweekstart(date))
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/15/2022	1/15/2022 12:00:00 AM
2/5/2022	02/05/2022	2/5/2022 12:00:00 AM
2/28/2022	02/26/2022	2/26/2022 12:00:00 AM
3/16/2022	03/12/2022	3/12/2022 12:00:00 AM
4/1/2022	03/26/2022	3/26/2022 12:00:00 AM
5/7/2022	05/07/2022	5/7/2022 12:00:00 AM
5/16/2022	05/14/2022	5/14/2022 12:00:00 AM
6/15/2022	06/11/2022	6/11/2022 12:00:00 AM
6/26/2022	06/25/2022	6/25/2022 12:00:00 AM
7/9/2022	07/09/2022	7/9/2022 12:00:00 AM

date	=lunarweekstart(date)	=timestamp(lunarweekstart(date))
7/22/2022	07/16/2022	7/16/2022 12:00:00 AM
7/23/2022	07/23/2022	7/23/2022 12:00:00 AM
7/27/2022	07/23/2022	7/23/2022 12:00:00 AM
8/2/2022	07/30/2022	7/30/2022 12:00:00 AM
8/8/2022	08/06/2022	8/6/2022 12:00:00 AM
8/19/2022	08/13/2022	8/13/2022 12:00:00 AM
9/26/2022	09/24/2022	9/24/2022 12:00:00 AM
10/14/2022	10/08/2022	10/8/2022 12:00:00 AM
10/29/2022	10/29/2022	10/29/2022 12:00:00 AM

La medida `start_of_week` se crea en el objeto gráfico utilizando la función `lunarweekstart()` e indicando el campo de fecha como argumento de la función.

La función `lunarweekstart()` identifica en qué semana lunar cae el valor de la fecha y devuelve una marca de tiempo del último milisegundo de esa semana.

Diagrama de la función `lunarweekstart()`, ejemplo de objeto gráfico



La transacción 8189 tuvo lugar el 19 de enero. La función `lunarweekstart()` identifica que la semana lunar comienza el 15 de enero. Por lo tanto, el valor `start_of_week` de esa transacción es el primer milisegundo de ese día, que es el 15 de enero a las 12:00:00 a. m.

Ejemplo 5: Escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de saldos de préstamos, que se carga en una tabla llamada `Loans`.
- Datos que consisten en ID de préstamos, el saldo al comienzo de la semana y la tasa de interés simple cobrada en cada préstamo por año.

Al usuario final le gustaría tener un objeto gráfico que muestre, por ID de préstamo, el interés actual que se ha acumulado en cada préstamo en la semana hasta la fecha.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla.
2. Agregue los siguientes campos como dimensiones.
    - `loan_id`
    - `start_balance`
3. A continuación, cree la siguiente medida para calcular el interés acumulado:
   `=start_balance*(rate*(today(1)-lunarweekstart(today(1)))/365)`
4. Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

loan_id	start_balance	=start_balance*(rate*(today(1)- lunarweekstart (today(1)))/365)
8188	$10000.00	$15.07
8189	$15000.00	$128.84
8190	$17500.00	$63.29
8191	$21000.00	$107.59
8192	$90000.00	$1139.18

La función `lunarweekstart()`, utilizando la fecha de hoy como único argumento, devuelve la fecha de inicio del año actual. Al restar ese resultado de la fecha actual, la expresión devuelve el número de días que han transcurrido en lo que va de semana.

Luego, este valor se multiplica por la tasa de interés y se divide por 365 para obtener la tasa de interés efectiva en que se ha incurrido durante este período. A continuación el resultado se multiplica por el saldo inicial del préstamo para devolver el interés que se ha acumulado en lo que va de semana.

makedate

Esta función devuelve una fecha calculada a partir del año **YYYY**, el mes **MM** y el día **DD**.

Sintaxis:

```
MakeDate(YYYY [ , MM [ , DD ] ])
```

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
YYYY	El año como un entero.
MM	El mes como un entero. Si no se especifica un mes, se presupone 1 (Enero).
DD	El día como un entero. Si no se define día alguno, se presupone 1 (el primero de mes).

Cuándo se utiliza

La función `makedate()` se utiliza habitualmente en el script para generación de datos, para generar un calendario. También puede utilizarse cuando el campo de fecha no está disponible directamente como fecha, pero necesita algunas transformaciones para extraer los componentes de año, mes y día.

Estos ejemplos utilizan el formato de fecha MM/DD/AAAA. El formato de fecha se especifica en la sentencia `SET DateFormat` en la parte superior de su script de carga de datos. Cambie el formato en los ejemplos según se ajuste a sus necesidades.

Ejemplos de funciones

Ejemplo	Resultado
`makedate(2012)`	Devuelve 01/01/2012.
`makedate(12)`	Devuelve 01/01/2012.
`makedate(2012,12)`	Devuelve 12/01/2012.
`makedate(2012,2,14)`	Devuelve 02/14/2012.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y

otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: ejemplo básico

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2018, que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo, `transaction_date`, que devuelve una fecha en formato MM/DD/AAAA.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- transaction_year
- transaction_month
- transaction_day
- transaction_date

Tabla de resultados

transaction_year	transaction_month	transaction_day	transaction_date
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	09/23/2018

El campo transaction_date se crea en la instrucción load anterior utilizando la función makedate() e introduciendo los campos de año, mes y día como argumentos de la función.

Luego la función combina y convierte estos valores en un campo de fecha, devolviendo los resultados en el formato de la variable del sistema DateFormat.

Ejemplo 2: formato de fecha modificado

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, transaction_date, en el formato DD/MM/AAAA, sin modificar la variable del sistema DateFormat.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `transaction_year`
- `transaction_month`
- `transaction_day`
- `transaction_date`

Tabla de resultados

transaction_year	transaction_month	transaction_day	transaction_date
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

En este caso, la función `makedate()` está anidada dentro de la función `date()`. El segundo argumento de la función `date()` establece el formato de los resultados de la función `makedate()` como el DD/MM/AAAA requerido.

Ejemplo 3: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2018, que se carga en una tabla llamada `Transactions`.
- Las fechas de transacción proporcionadas en dos campos: `year` y `month`.

Cree una medida de objeto gráfico, `transaction_date`, que devuelva una fecha en el formato MM/DD/AAAA.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- year
- month

Para averiguar la transaction_date, cree esta medida:

```
=makedate(transaction_year,transaction_month)
```

Tabla de resultados

transaction_year	transaction_month	transaction_date
2018	08	08/01/2018
2018	09	09/01/2018

La medida transaction_date se crea en el objeto gráfico utilizando la función makedate() e indicando los campos de año y mes como argumentos de la función.

La función luego combina estos valores, así como el valor del día asumido de 01. Estos valores luego se convierten en un campo de fecha, devolviendo los resultados en el formato de la variable del sistema DateFormat.

Ejemplo 4: Escenario

Script de carga y expresión de gráfico

Vista general

Cree un conjunto de datos de calendario para el año natural de 2022.

Script de carga

```
SET DateFormat='MM/DD/YYYY';

Calendar:
       load
                *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 400;
```

Resultados

Tabla de resultados

date
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
01/11/2022
01/12/2022
01/13/2022
01/14/2022
01/15/2022
01/16/2022
01/17/2022
01/18/2022
01/19/2022
01/20/2022
01/21/2022

date
01/22/2022
01/23/2022
01/24/2022
01/25/2022
+ 340 filas más

La función `makedate()` crea un valor de fecha para el 31 de diciembre de 2021. La función `recno()` proporciona el número de registro del registro actual que se está cargando en la tabla, comenzando desde 1. Por lo tanto, el primer registro tiene la fecha del 1 de enero de 2022. Cada fecha sucesiva `recno()` incrementará esta fecha en 1. Esta expresión va incluida en una función `date()` para convertir el valor en una fecha. La función `autogenerate` repite este proceso 400 veces. Por último, usando un load anterior, se puede aplicar una condición `where` para que cargue solo fechas del año 2022. Este script genera un calendario que contiene todas las fechas en 2022.

maketime

Esta función devuelve una hora calculada a partir de la hora **hh**, el minuto **mm** y el segundo **ss**.

Sintaxis:

```
MakeTime(hh [ , mm [ , ss ] ])
```

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
hh	La hora como un entero.
mm	El minuto como un entero. Si no se especifica minuto, se presupone 00.
ss	El segundo como un entero. Si no se especifica segundo, se presupone 00.

Cuándo se utiliza

La función `maketime()` se utiliza habitualmente en el script para generación de datos, para generar un campo de hora. A veces, cuando el campo de hora se deriva del texto de entrada, esta función puede servir para crear la hora usando sus componentes.

Estos ejemplos utilizan el formato de hora `h:mm:ss`. El formato de hora se especifica en la sentencia `SET TimeFormat` en la parte superior de su script de carga de datos. Cambie el formato en los ejemplos según se ajuste a sus necesidades.

Ejemplos de funciones

Ejemplo	Resultado
`maketime(22)`	Devuelve 22:00:00.
`maketime(22, 17)`	Devuelve 22:17:00.
`maketime(22,17,52 )`	Devuelve 22:17:52.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: maketime()

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones, que se carga en una tabla llamada `Transactions`.
- Horas de las transacciones, proporcionado en tres campos: `hours`, `minutes` y `seconds`.
- La creación de un campo, `transaction_time`, que devuelve la hora en el formato de la variable del sistema `TimeFormat`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `transaction_hour`
- `transaction_minute`
- `transaction_second`
- `transaction_time`

Tabla de resultados

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

El campo `transaction_time` se crea en la instrucción load anterior utilizando la función `maketime()` e indicando los campos de hora, minuto y segundo como argumentos de la función.

A continuación la función combina y convierte estos valores en un campo de hora, devolviendo los resultados en el formato de la variable del sistema `TimeFormat`.

Ejemplo 2: función time()

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `transaction_time`, que nos permitirá mostrar los resultados en el formato horario de 24 horas, sin modificar la variable del sistema `TimeFormat`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `transaction_hour`
- `transaction_minute`
- `transaction_second`
- `transaction_time`

Tabla de resultados

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

En este caso, la función maketime() está anidada dentro de la función time(). El segundo argumento de la función time() establece el formato de los resultados de la función maketime() como el DD/MM/AAAA requerido h:mm:ss.

Ejemplo 3: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones, que se carga en una tabla llamada Transactions.
- Horas de las transacciones, proporcionado en dos campos: hours y minutes.
- La creación de un campo, transaction_time, que devuelve la hora en el formato de la variable del sistema TimeFormat.

Cree una medida de objeto gráfico, transaction_time, que devuelva una fecha en el formato h:mm:ss TT.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- transaction_hour
- transaction_minute

Para calcular transaction_time, cree esta medida:

```
=maketime(transaction_hour,transaction_minute)
```

Tabla de resultados

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

La medida `transaction_time` se crea en el objeto gráfico utilizando la función `maketime()` e insertando los campos de hora y minuto como argumentos de la función.

Luego, la función combina estos valores y se supone que los segundos son 00. Estos valores se convierten luego en un campo de hora, devolviendo los resultados en el formato de la variable del sistema `TimeFormat`.

Ejemplo 4: Escenario

Script de carga y expresión de gráfico

Vista general

Cree un conjunto de datos de calendario para el mes de enero de 2022, desglosado en incrementos de ocho horas.

Script de carga

```
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;
```

Resultados

Tabla de resultados

timestamp
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
1/7/2022 4:00:00 PM
1/8/2022 12:00:00 AM
1/8/2022 8:00:00 AM
1/8/2022 4:00:00 PM
1/9/2022 12:00:00 AM
+ 68 filas más

La función autogenerate inicial crea un calendario que contiene todas las fechas de enero en una tabla llamada tmpCalendar.

Se crea una segunda tabla, que contiene tres registros. Para cada registro, se toma recno() - 1 (valores 0, 1, 2) y el resultado se multiplica por 8. Como resultado, esto genera los valores 0, 8 16. Estos valores se utilizan como parámetro de hora en una función maketime(), con valores de minutos y segundos de 0. Como resultado, la tabla contiene tres campos de hora: 12:00:00 a. m., 8:00:00 a. m. y 4:00: 00 p. m.

Esta tabla se une a la tabla tmpCalendar. Dado que no hay campos coincidentes entre las dos tablas para la combinación, las filas de hora se agregan a cada fila de fecha. Como resultado, cada fila de fecha ahora se repite tres veces con cada valor de tiempo.

Por último, la tabla Calendar se crea a partir de una carga residente de la tabla tmpCalendar. Los campos de fecha y hora se concatenan e indican en la función timestamp() para crear el campo de marca de tiempo.

Después, se elimina la tabla tmpCalendar.

makeweekdate

Esta función devuelve una fecha calculada a partir del año, el número de semana y el día de la semana.

Sintaxis:

```
MakeWeekDate(weekyear [, week [, weekday [, first_week_day [, broken_weeks [, reference_day]]]]])
```

Tipo de datos que devuelve: dual

La función makeweekdate() está disponible tanto como función de script como de gráfico. La función calculará la fecha en función de los parámetros indicados en la función.

Argumentos

Argumento	Descripción
weekyear	El año definido por la función weekYear() para la fecha específica, es decir, el año al que pertenece el número de semana. *En algunos casos, el año de la semana puede ser diferente del año natural, por ejemplo, si la semana 1 ya comienza en diciembre del año anterior.*
week	El número de semana definido por la función week() para la fecha específica. Si no se indica el número de semana, se presupone 1.

Argumento	Descripción
weekday	El día de la semana definido por la función `WeekDay()` para la fecha en cuestión. 0 es el primer día de la semana y 6 es el último día de la semana. Si no se indica el día de la semana, se presupone 0. *Aunque 0 siempre significa el primer día de la semana y 6 siempre es el último, a qué día de la semana corresponde el parámetro **first_week_day** . Si se omite, se utiliza el valor de la variable **FirstWeekDay**.* Si se utilizan semanas interrumpidas, junto con una combinación imposible de parámetros, esto puede llevar a un resultado que no corresponde al año elegido. **Ejemplo:** `MakeWeekDate(2021,1,0,6,1)` Devuelve "27 de diciembre de 2020" ya que este día es el primer día (el domingo) de la semana especificada. El 1 de enero de 2021 fue viernes.
first_week_day	Especifica el día en el que se inicia la semana. Si se omite, se utiliza el valor de la variable **FirstWeekDay**. Los valores posibles de **first_week_day** son 0 para el lunes, 1 para el martes, 2 para el miércoles, 3 para el jueves, 4 para el viernes, 5 para el sábado y 6 para el domingo. Para más información sobre la variable del sistema, vea *FirstWeekDay (page 229)*.
broken_weeks	Si no se especifica **broken_weeks**, el valor de la variable **BrokenWeeks** se usará para definir si las semanas se rompen o no.
reference_day	Si no especifica **reference_day**, el valor de la variable **ReferenceDay** se utilizará para definir qué día de enero establecer como día de referencia para definir la semana 1.

Cuándo se utiliza

La función `makeweekdate()` se utiliza normalmente en el script para generación de datos, para generar una lista de fechas o para crear fechas cuando el año, la semana y el día de la semana se proporcionan en los datos de entrada.

Los siguientes ejemplos suponen:

```
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;
```

Ejemplos de funciones

Ejemplo	Resultado
`makeweekdate(2014,6,6)`	devuelve 02/09/2014
`makeweekdate(2014,6,1)`	devuelve 02/04/2014
`makeweekdate(2014,6)`	devuelve 02/03/2014 (se presupone el día 0 de la semana)

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: día incluido

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene el total de ventas semanales para 2022 en una tabla llamada `Sales`.
- Fechas de transacción proporcionadas en tres campos: `year`, `week` y `sales`.
- Una instrucción load anterior, que se utiliza para crear una medida `end_of_week`, utilizando la función `makeweekdate()` para devolver la fecha del viernes de esa semana en el formato MM/DD/AAAA.

Para probar que la fecha devuelta es un viernes, la expresión `end_of_week` también se incluye en la función `weekday()` para mostrar el día de la semana.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- transaction_year
- transaction_week
- end_of_week
- week_day

Tabla de resultados

transaction_year	transaction_week	end_of_week	week_day
2022	01	01/07/2022	Vie
2022	02	01/14/2022	Vie
2022	03	01/21/2022	Vie
2022	04	01/28/2022	Vie
2022	05	02/04/2022	Vie
2022	06	02/11/2022	Vie
2022	07	02/18/2022	Vie

El campo "end_of_week" se crea en la instrucción load anterior, mediante el uso de la función makeweekdate(). Los campos transaction_year, transaction_week se indican mediante la función como argumentos de año y semana. Se utiliza un valor de 4 para el argumento del día.

Luego la función combina y convierte estos valores en un campo de fecha, devolviendo los resultados en el formato de la variable del sistema DateFormat.

La función makeweekdate() y sus argumentos también van dentro de una función weekday() para devolver el campo week_day; y como se puede ver en la tabla anterior, el campo week_day muestra que estas fechas ocurren un viernes.

Ejemplo 2: día excluido

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene los totales de ventas semanales para 2022 en una tabla llamada `Sales`.
- Fechas de transacción proporcionadas en tres campos: `year`, `week` y `sales`.
- Un load anterior, que se utiliza para crear una medida, `first_day_of_week`, utilizando la función `makeweekdate()`. Esto devolverá la fecha del lunes de esa semana en el formato MM/DD/AAAA.

Para probar que la fecha devuelta es un lunes, la expresión `first_day_of_week` también se incluye en la función `weekday()` para mostrar el día de la semana.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `transaction_year`
- `transaction_week`

- first_day_of_week
- week_day

Tabla de resultados

transaction_year	transaction_week	first_day_of_week	week_day
2022	01	01/03/2022	Lun
2022	02	01/10/2022	Lun
2022	03	01/17/2022	Lun
2022	04	01/24/2022	Lun
2022	05	01/31/2022	Lun
2022	06	02/07/2022	Lun
2022	07	02/14/2022	Lun

El campo first_day_of_week se crea en la instrucción de carga anterior mediante el uso de la función makeweekdate(). Los parámetros transaction_year y transaction_week se indican como argumentos de la función y el parámetro del día se deja en blanco.

Luego la función combina y convierte estos valores en un campo de fecha, devolviendo los resultados en el formato de la variable del sistema DateFormat.

La función makeweekdate() y sus argumentos también van incluidos en una función weekday() que devuelve el campo week_day. Como se puede ver en la tabla anterior, el campo week_day devuelve el lunes en todos los casos, ya que ese parámetro se dejó en blanco en la función makeweekdate(), que por defecto es 0 (primer día de la semana), y el sistema establece el lunes como primer día de la semana en la variable de sistema FirstWeekDay.

Ejemplo 3: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene los totales de ventas semanales para 2022 en una tabla llamada Sales.
- Fechas de transacción proporcionadas en tres campos: year, week y sales.

En este ejemplo, se usará un objeto gráfico para crear una medida equivalente al cálculo de end_of_week del primer ejemplo. Esta medida utilizará la función makeweekdate() para devolver la fecha del viernes de esa semana en el formato MM/DD/AAAA.

Para demostrar que la fecha devuelta es un viernes, se crea una segunda medida que devuelve el día de la semana.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:
   - `transaction_year`
   - `transaction_week`
2. Para realizar el cálculo equivalente al del campo `end_of_week` del primer ejemplo, cree la siguiente medida:
   `=makeweekdate(transaction_year,transaction_week,4)`
3. Para calcular el día de la semana de cada transacción, cree la siguiente medida:
   `=weekday(makeweekdate(transaction_year,transaction_week,4))`

Tabla de resultados

transaction_ year	transaction_ week	=makeweekdate (transaction_ year,transaction_week,4)	=weekday(makeweekdate (transaction_year,transaction_ week,4))
2022	01	01/07/2022	Vie
2022	02	01/14/2022	Vie
2022	03	01/21/2022	Vie
2022	04	01/28/2022	Vie
2022	05	02/04/2022	Vie
2022	06	02/11/2022	Vie
2022	07	02/18/2022	Vie

Se crea un campo equivalente a `end_of_week` en el objeto gráfico como medida usando la función `makeweekdate()`. Los campos `transaction_year` y `transaction_week` se indican como argumentos de año y semana. Se utiliza un valor de 4 para el argumento del día.

Luego la función combina y convierte estos valores en un campo de fecha, devolviendo los resultados en el formato de la variable del sistema `DateFormat`.

La función `makeweekdate()` y sus argumentos también van incluidos en una función `weekday()` que devuelve un cálculo equivalente al del campo `week_day` del primer ejemplo. Como se puede ver en la tabla anterior, la última columna de la derecha muestra que estas fechas ocurren en viernes.

Ejemplo 4: Escenario

Script de carga y expresión de gráfico

Vista general

En este ejemplo, cree una lista de fechas que contenga todos los viernes del año 2022.

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

Script de carga

```
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;
```

Resultados

Tabla de resultados

date	weekday
01/07/2022	Vie
01/14/2022	Vie
01/21/2022	Vie
01/28/2022	Vie
02/04/2022	Vie

date	weekday
02/11/2022	Vie
02/18/2022	Vie
02/25/2022	Vie
03/04/2022	Vie
03/11/2022	Vie
03/18/2022	Vie
03/25/2022	Vie
04/01/2022	Vie
04/08/2022	Vie
04/15/2022	Vie
04/22/2022	Vie
04/29/2022	Vie
05/06/2022	Vie
05/13/2022	Vie
05/20/2022	Vie
05/27/2022	Vie
06/03/2022	Vie
06/10/2022	Vie
06/17/2022	Vie
+ 27 filas más	

La función `makeweekdate()` encuentra cada viernes de 2022. Usar un parámetro de semana de -2 garantiza que no se pierda ninguna fecha. Por último, una instrucción load anterior crea un campo `weekday` adicional para mayor claridad, para mostrar que cada valor de `date` es un viernes.

minute

Esta función devuelve un entero que representa el minuto en que la fracción de la **expression** se interpreta como una hora de acuerdo con la interpretación numérica estándar.

Sintaxis:

```
minute(expression)
```

Tipo de datos que devuelve: Entero

Cuándo se utiliza

La función `minute()` es útil cuando se desea comparar agregaciones por minuto. Por ejemplo, podría usar la función si desea ver la distribución del recuento de actividades por minuto.

Estas dimensiones se pueden crear o bien en el script de carga, utilizando la función para crear un campo en una tabla de calendario maestro. O bien, se pueden usar directamente en un gráfico como una dimensión calculada.

Ejemplos de funciones

Ejemplo	Resultado
`minute ( '09:14:36' )`	Devuelve 14.
`minute ( '0.5555' )`	Devuelve 19 (porque 0,5555 = 13:19:55).

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1 : Variable (script)

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene transacciones por marca de tiempo, que se carga en una tabla llamada `Transactions`.
- Se utiliza la variable predefinida del sistema `TimeStamp` (`M/D/YYYY h:mm:ss[.fff] TT`).
- La creación de un campo, `minute`, para calcular cuándo se realizaron las transacciones.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `timestamp`
- `minute`

Tabla de resultados

timestamp	minute
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

Los valores del campo `minute` se crean utilizando la función `minute()` e insertando `timestamp` como expresión en la instrucción load anterior.

Ejemplo 2 : Objeto gráfico (gráfico)

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- Se utiliza la variable predefinida del sistema `TimeStamp` (`M/D/YYYY h:mm:ss[.fff] TT`).

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. Los valores de `minute` se calculan por medio de una medida en un objeto gráfico.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `timestamp`.

Cree la siguiente medida:

```
=minute(timestamp)
```

Tabla de resultados

timestamp	minute
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

Los valores de `minute` se crean usando la función `minute()` e introduciendo `timestamp` como expresión en una medida del objeto gráfico.

Ejemplo 3: escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de fechas y horas, que se genera para representar entradas en una barrera de tickets.
- Información con cada `timestamp` y su `id` correspondiente, que se carga en una tabla denominada `Ticket_Barrier_Tracker`.
- Se utiliza la variable predefinida del sistema `TimeStamp` (`M/D/YYYY h:mm:ss[.fff] TT`).

Al usuario le gustaría tener un objeto gráfico que muestre, por minuto, el recuento de entradas de barrera.

Script de carga

```
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;
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla.
2. Cree una dimensión calculada usando la siguiente expresión:
   `=minute(timestamp)`
3. Agregue la siguiente medida de agregación para calcular el recuento total de entradas:
   `=count(id)`
4. Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

minute(timestamp)	=count(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

minute(timestamp)	=count(id)
14	159
15	161
16	152
17	160
18	176
19	164
20	170
21	170
22	142
23	145
24	155
+ 35 filas más	

month

Esta función devuelve un valor dual: un nombre de mes tal como se define en la variable de entorno **MonthNames** y un entero entre 1-12. El mes se calcula a partir de la interpretación de la fecha de la expresión, conforme a la interpretación numérica estándar.

La función devuelve el nombre del mes en el formato de la variable del sistema `MonthName` de una fecha en particular. Normalmente se utiliza para crear un campo de día como dimensión en un calendario maestro.

Sintaxis:

```
month(expression)
```

Tipo de datos que devuelve: Entero

Ejemplos de funciones

Ejemplo	Resultado
`month( 2012-10-12 )`	devuelve Oct
`month( 35648 )`	devuelve Aug, porque 35648 = 1997-08-06

Ejemplo 1: conjunto de datos DateFormat (script)

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de fechas denominado `Master_Calendar`. La variable de sistema `DateFormat`, que está configurada como DD/MM/AAAA.
- Un load precedente que crea un campo adicional, llamado `month_name`, usando la función `month()`.
- Un campo adicional, denominado `long_date`, usando la función `date()` para expresar la fecha completa.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `long_date`
- `month_name`

Tabla de resultados

date	long_date	month_name
03/01/2022	03 Enero 2022	Ene
03/02/2022	03 Febrero 2022	Feb
03/03/2022	03 Marzo 2022	Mar
03/04/2022	03 Abril 2022	Abr

date	long_date	month_name
03/05/2022	03 Mayo 2022	May
03/06/2022	03 Junio 2022	Jun
03/07/2022	03 Julio 2022	Jul
03/08/2022	03 Agosto 2022	Ago
03/09/2022	03 Diciembre 2022	Sep
03/10/2022	03 Octubre 2022	Oct
03/11/2022	03 noviembre 2022	Nov

La función de script `month()` evalúa correctamente el nombre del mes.

Ejemplo 2: fechas ANSI (script)

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de fechas denominado `Master_Calendar`. Se utiliza la variable de sistema DateFormat DD/MM/AAAA. Sin embargo, las fechas que se incluyen en el conjunto de datos están en formato de fecha estándar ANSI.
- Un load precedente que crea un campo adicional, llamado `month_name`, usando la función month().
- Un campo adicional, denominado `long_date`, usando la función date() para expresar la fecha completa.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `long_date`
- `month_name`

Tabla de resultados

date	long_date	month_name
03/11/2022	11 marzo 2022	11
03/12/2022	12 marzo 2022	12
03/13/2022	13 marzo 2022	13
03/14/2022	14 marzo 2022	14
03/15/2022	15 marzo 2022	15
03/16/2022	16 marzo 2022	16
03/17/2022	17 marzo 2022	17
03/18/2022	18 marzo 2022	18
03/19/2022	19 marzo 2022	19
03/20/2022	20 marzo 2022	20
03/21/2022	21 marzo 2022	21

La función de script `month()` evalúa correctamente el nombre del mes.

Ejemplo 3: fechas sin formato (script)

Script de carga y resultados

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de fechas denominado `Master_Calendar`. Se utiliza la variable de sistema `DateFormat` DD/MM/AAAA.

- Un load precedente que crea un campo adicional, llamado `month_name`, usando la función `month()` .
- La fecha original sin formato, denominada `unformatted_date`.
- Un campo adicional, denominado `long_date`, usando la función `date()` para expresar la fecha completa.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `unformatted_date`
- `long_date`
- `month_name`

Tabla de resultados

unformatted_date	long_date	month_name
44868	03 Enero 2022	Ene
44898	03 Febrero 2022	Feb
44928	03 Marzo 2022	Mar
44958	03 Abril 2022	Abr
44988	03 Mayo 2022	May

unformatted_date	long_date	month_name
45018	03 Junio 2022	Jun
45048	03 Julio 2022	Jul
45078	03 Agosto 2022	Ago
45008	03 Diciembre 2022	Sep
45038	03 Octubre 2022	Oct
45068	03 noviembre 2022	Nov

La función de script `month()` evalúa correctamente el nombre del mes.

Ejemplo 4: cálculo del mes de vencimiento

Script de carga y expresión de gráfico

Vista general

Abra el Editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de pedidos realizados en marzo denominado `Subscriptions`. La tabla contiene tres campos:
  - id
  - order_date
  - amount

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `order_date`.

Para calcular el mes de vencimiento de un pedido, cree esta medida: `=month(order_date+180)`.

Tabla de resultados

order_date	=month(order_date+180)
03/01/2022	Jul
03/02/2022	Ago
03/03/2022	Ago
03/04/2022	Sep
03/05/2022	Oct
03/06/2022	Nov
03/07/2022	Dec
03/08/2022	Ene
03/09/2022	Mar
03/10/2022	Abr
03/11/2022	May

La función `month()` determina correctamente que un pedido realizado el 11 de marzo vencería en julio.

monthend

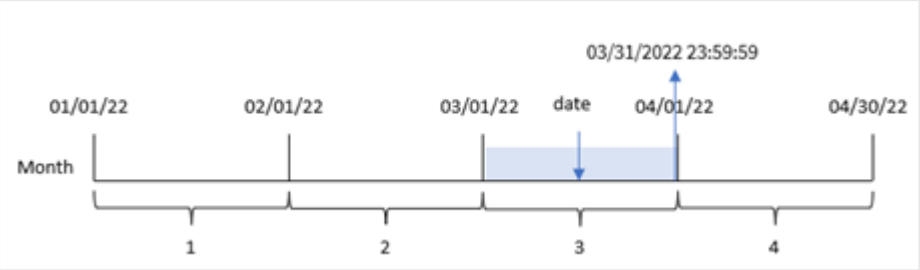
Esta función devuelve un valor correspondiente a una marca de tiempo (fecha-hora) del último milisegundo del último día del mes que contiene a `date`. El formato de salida predeterminado será el `DateFormat` establecido en el script.

Sintaxis:

```
MonthEnd(date[, period_no])
```

En otras palabras, la función `monthend()` determina en qué mes cae la fecha. Luego devuelve una marca de tiempo, en formato de fecha, con el último milisegundo de ese mes.

Diagrama de la función monthend.



Cuándo se utiliza

La función `monthend()` se usa como parte de una expresión cuando deseamos que el cálculo utilice la fracción del mes que aún no ha ocurrido. Por ejemplo, si desea calcular el interés total aún no devengado durante el mes.

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
date	La fecha o marca de tiempo para evaluar.
period_no	**period_no** es un número entero que, si es 0 o se omite, indica el mes que contiene **date**. Los valores negativos en **period_no** indican meses precedentes y los valores positivos indican meses subsiguientes.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
`monthend('02/19/2012')`	Devuelve 02/29/2012 23:59:59.
`monthend('02/19/2001', -1)`	Devuelve 01/31/2001 23:59:59.

Ejemplo 1: ejemplo básico

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene una serie de transacciones de 2022 que se carga en una tabla llamada "`Transactions`".
- Un campo de fecha con el formato de la variable del sistema `DateFormat` (`MM/DD/YYYY`).
- Una instrucción load anterior que contiene lo siguiente:
    - La función `monthend()` que está definida como el campo "`end_of_month`".
    - La función `timestamp` que está definida como el campo "`end_of_month_timestamp`".

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`
- `end_of_month`
- `end_of_month_timestamp`

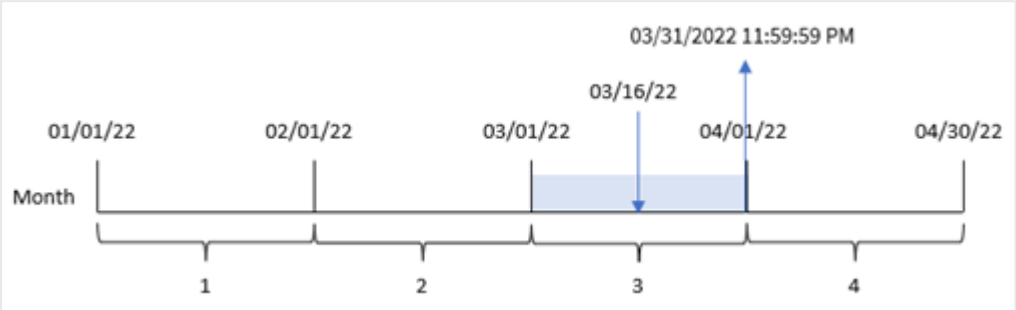
Tabla de resultados

id	date	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

El campo `end_of_month` se crea en la instrucción load anterior utilizando la función `monthend()` e introduciendo el campo de fecha como argumento de la función.

La función `monthend()` identifica en qué mes del año cae el valor de la fecha y devuelve una marca de tiempo del último milisegundo de ese mes.

Diagrama de la función `monthend` con marzo como el mes seleccionado.



La transacción 8192 tuvo lugar el 16 de marzo. La función `monthend()` devuelve el último milisegundo de ese mes, que es el 31 de marzo a las 23:59:59.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

En este ejemplo, la tarea es crear un campo, "`previous_month_end`'', que devuelva la marca de tiempo del final del mes anterior a la transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`
- `previous_month_end`
- `previous_month_end_timestamp`

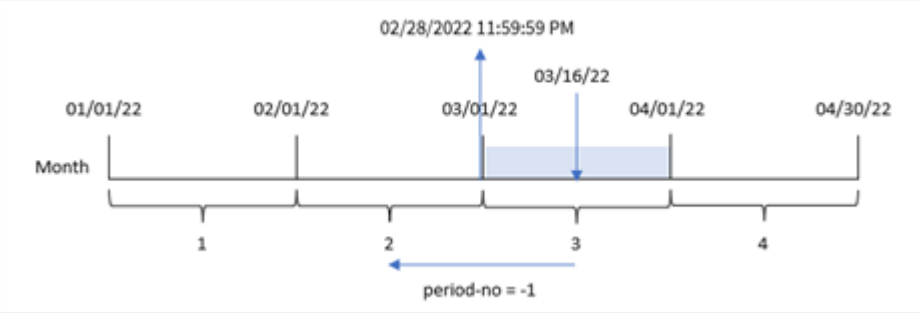
Tabla de resultados

id	date	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

id	date	previous_month_end	previous_month_end_timestamp
8204	8/19/2022	07/31/2022	7/31/2022 11:59:59 PM
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

La función `monthend()` identifica primero el mes en que se realizan las transacciones, ya que se ha utilizado un `period_no` de `-1` como argumento del desplazamiento. Luego cambia a un mes antes e identifica el último milisegundo de ese mes.

Diagrama de la función `monthend` con la variable period_no.



La transacción 8192 tuvo lugar el 16 de marzo. La función `monthend()` identifica que el mes anterior a la transacción fue febrero. Luego devuelve el último milisegundo de ese mes, el 28 de febrero a las 23:59:59.

Ejemplo 3: ejemplo gráfico

Script de carga y expresión de gráfico

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

En este ejemplo el conjunto de datos está sin modificar y cargado en la app. La tarea es crear un cálculo que devuelve una marca de tiempo del final de mes en que se realizaron las transacciones como una medida en un gráfico de la app.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `id`

Para calcular la fecha de fin del mes en el que tiene lugar una transacción, cree las siguientes medidas:

- `=monthend(date)`
- `=timestamp(monthend(date))`

Tabla de resultados

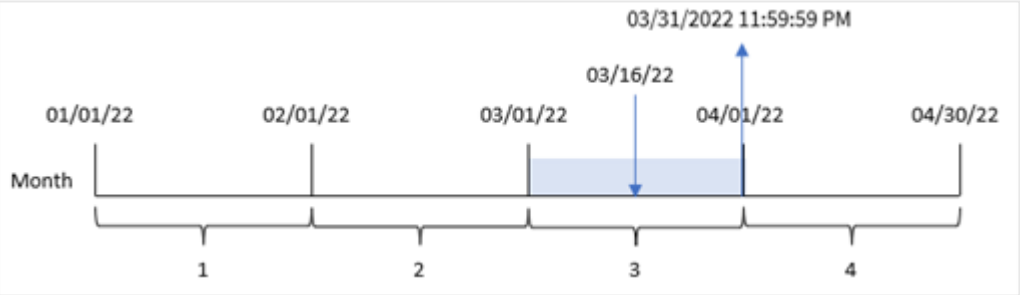
id	date	=monthend(date)	=timestamp(monthend(date))
8188	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8189	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM
8190	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8191	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8192	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8193	8/19/2022	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	date	=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

La medida "`end_of_month`" se crea en el gráfico utilizando la función `monthend()` e indicando el campo de fecha como argumento de la función.

La función `monthend()` identifica en qué mes del año cae el valor de la fecha y devuelve una marca de tiempo del último milisegundo de ese mes.

Diagrama de la función `monthend` con la variable period_no.



La transacción 8192 tuvo lugar el 16 de marzo. La función `monthend()` devuelve el último milisegundo de ese mes, que es el 31 de marzo a las 23:59:59.

Ejemplo 4: Escenario

Script de carga y resultados

Vista general

En este ejemplo, un conjunto de datos se carga en una tabla denominada "`Employee_Expenses`". La tabla contiene los siguientes campos:

- Employee IDs
- Employee names
- Las reclamaciones de gastos diarios promedio de cada empleado.

Al usuario final le gustaría tener un gráfico que muestre, por ID y nombre de empleado, la declaración de gastos estimada para el resto del mes.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `employee_id`
- `employee_name`

Para calcular el interés acumulado, cree esta medida:

```
=floor(monthend(today(1),0)-today(1))*avg_daily_claim
```

Esta medida es dinámica y producirá diferentes resultados en la tabla dependiendo de la fecha en que cargue los datos.

Defina el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

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

La función `monthend()` devuelve la fecha de finalización del mes actual utilizando la fecha de hoy como único argumento. La expresión devuelve el número de días que quedan de este mes al restar la fecha de hoy de la fecha de finalización del mes.

Luego, este valor se multiplica por la reclamación de gastos diarios promedio de cada empleado para calcular el valor estimado de las reclamaciones de gastos que se espera que haga cada empleado en el mes restante.

monthname

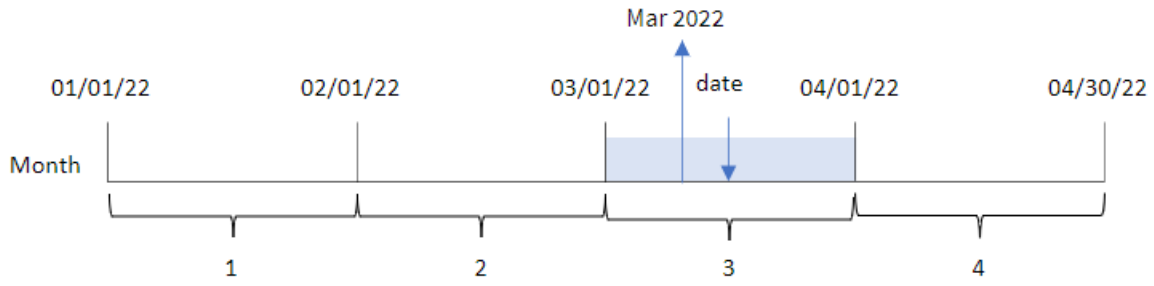
Esta función devuelve un valor de visualización que muestra el mes (con formato de acuerdo con la variable de script **MonthNames**) y el año con un valor numérico subyacente correspondiente a una marca de tiempo (fecha-hora) del primer milisegundo del primer día del mes.

Sintaxis:

```
MonthName(date[, period_no])
```

Tipo de datos que devuelve: dual

Diagrama de la función monthname



Argumentos

Argumento	Descripción
date	La fecha o marca de tiempo para evaluar.
period_no	**period_no** es un número entero que, si es 0 o se omite, indica el mes que contiene **date**. Los valores negativos en **period_no** indican meses precedentes y los valores positivos indican meses subsiguientes.

Ejemplos de funciones

Ejemplo	Resultado
`monthname('10/19/2013')`	Devuelve Oct 2013
`monthname('10/19/2013', -1)`	Devuelve Sep 2013

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: ejemplo básico

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo, `transaction_month`, que devuelve el mes en el que se realizaron las transacciones.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

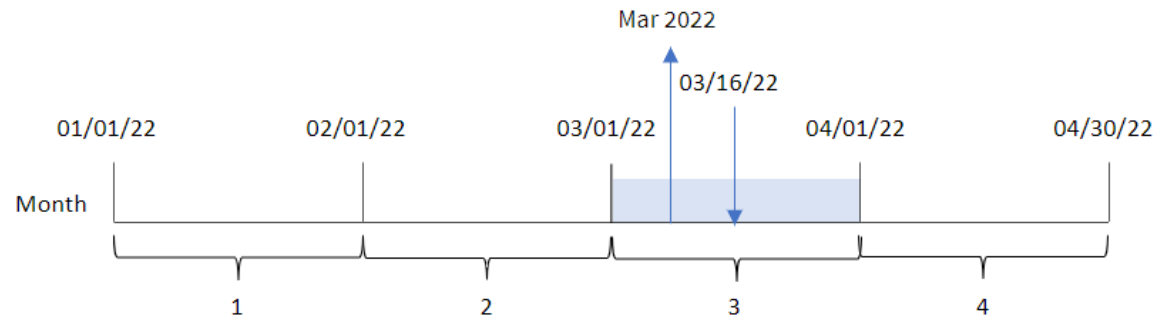
- `date`
- `transaction_month`

Tabla de resultados

date	transaction_month
1/7/2022	Ene 2022
1/19/2022	Ene 2022
2/5/2022	Feb 2022
2/28/2022	Feb 2022
3/16/2022	Mar 2022
4/1/2022	Abr 2022
5/7/2022	May 2022
5/16/2022	May 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	Ago 2022
8/8/2022	Ago 2022
8/19/2022	Ago 2022
9/26/2022	Sep 2022
10/14/2022	Oct 2022
10/29/2022	Oct 2022

El campo `transaction_month` se crea en la instrucción load anterior utilizando la función `monthname()` e introduciendo el campo `date` como argumento de la función.

Diagrama de la función monthname, ejemplo básico



La función `monthname()` identifica que la transacción 8192 tuvo lugar en marzo de 2022 y devuelve este valor mediante la variable de sistema `MonthNames`.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos inline y el mismo escenario que en el primer ejemplo.
- La creación de un campo, `transaction_previous_month`, que devuelve la marca de tiempo del final del mes anterior a la transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `transaction_previous_month`

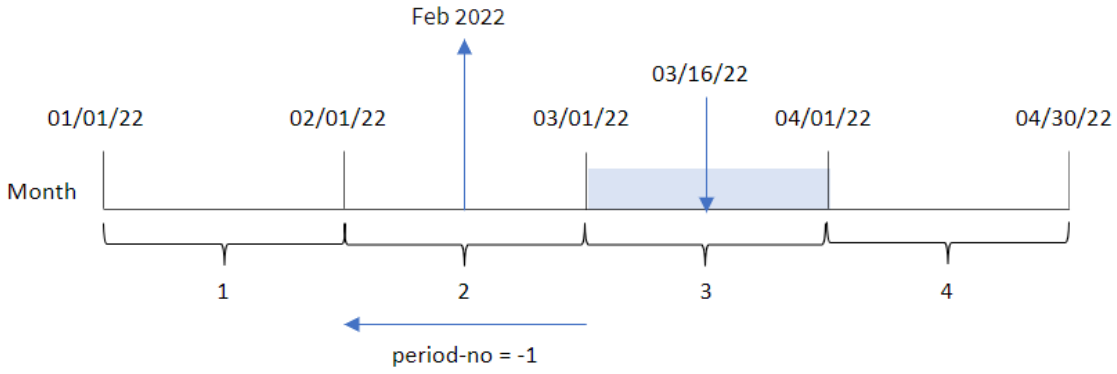
Tabla de resultados

date	transaction_previous_month
1/7/2022	Dic 2021
1/19/2022	Dic 2021
2/5/2022	Ene 2022
2/28/2022	Ene 2022
3/16/2022	Feb 2022
4/1/2022	Mar 2022
5/7/2022	Abr 2022
5/16/2022	Abr 2022
6/15/2022	May 2022
6/26/2022	May 2022
7/9/2022	Jun 2022
7/22/2022	Jun 2022
7/23/2022	Jun 2022
7/27/2022	Jun 2022

date	transaction_previous_month
8/2/2022	Jul 2022
8/8/2022	Jul 2022
8/19/2022	Jul 2022
9/26/2022	Ago 2022
10/14/2022	Sep 2022
10/29/2022	Sep 2022

En este caso, debido a que se usó un `period_no` de `-1` como argumento de desplazamiento en la función `monthname()`, la función primero identifica el mes en el que se realizan las transacciones. Luego cambia a un mes anterior y devuelve el nombre del mes y el año.

Diagrama de la función monthname, ejemplo de period_no



La transacción 8192 tuvo lugar el 16 de marzo. La función `monthname()` identifica que el mes anterior a la transacción fue febrero y devuelve el mes, en el formato de variable del sistema `MonthNames`, junto con el año 2022.

Ejemplo 3: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos inline y escenario que el primer ejemplo. Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que devuelve una marca de tiempo del final del mes en que se realizaron las transacciones se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:`date`.

Cree la siguiente medida:

```
=monthname(date)
```

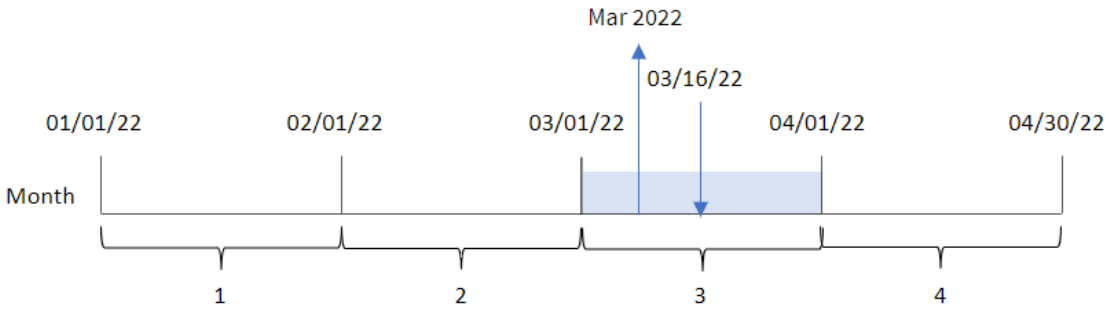
Tabla de resultados

date	=monthname(date)
1/7/2022	Ene 2022
1/19/2022	Ene 2022
2/5/2022	Feb 2022
2/28/2022	Feb 2022
3/16/2022	Mar 2022

date	=monthname(date)
4/1/2022	Abr 2022
5/7/2022	May 2022
5/16/2022	May 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	Ago 2022
8/8/2022	Ago 2022
8/19/2022	Ago 2022
9/26/2022	Sep 2022
10/14/2022	Oct 2022
10/29/2022	Oct 2022

La medida `month_name` se crea en el objeto gráfico utilizando la función `monthname()` e introduciendo el campo `date` como argumento de la función.

Diagrama de la función monthname, ejemplo de objeto gráfico



La función `monthname()` identifica que la transacción 8192 tuvo lugar en marzo de 2022 y devuelve este valor mediante la variable de sistema `MonthNames`.

monthsend

Esta función devuelve un valor correspondiente a la marca de tiempo del último milisegundo del periodo mensual, bimensual, trimestral, cuatrimestral o semestral que contiene una fecha base. También es posible hallar la marca de tiempo del final de un periodo anterior o posterior.

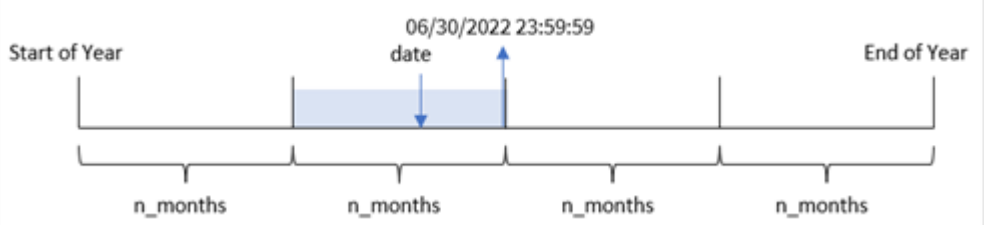
El formato de salida predefinido es el `DateFormat` definido en el script.

Sintaxis:

```
MonthsEnd(n_months, date[, period_no [, first_month_of_year]])
```

Tipo de datos que devuelve: dual

Diagrama de la función monthsend.



Argumentos

Argumento	Descripción
n_months	El número de meses que define el periodo. Un entero o expresión que devuelve un entero que debe ser uno de los siguientes: 1 (equivalente a la función inmonth()), 2 (bimestral), 3 (equivalente a la función inquarter()), 4 (cuatrimestral) o 6 (medio año).
date	La fecha o marca de tiempo para evaluar.
period_no	El período se puede desplazar mediante **period_no**, un entero o una expresión que devuelve un entero, donde el valor 0 indica el período que contiene a **base_date**. Los valores negativos en **period_no** indican períodos precedentes y los valores positivos indican períodos subsiguientes.
first_month_of_year	Si desea trabajar con años (fiscales) que no comiencen en enero, indique un valor entre 2 y 12 en **first_month_of_year**.

La función `monthsend()` divide el año en segmentos basándose en el argumento `n_months` proporcionado. Después evalúa en qué segmento cae cada fecha proporcionada y devuelve el último milisegundo, en formato de fecha, de ese segmento. La función puede devolver la marca de tiempo de finalización de los segmentos anteriores o posteriores, así como redefinir el primer mes del año.

Los siguientes segmentos del año están disponibles en la función `n_month` como argumentos.

Argumentos de `n_month`

Periodo	Número de meses
mes	1
bimestre	2
trimestre	3
cuatrimestre	4
semestre	6

Cuándo se utiliza

La función `monthsend()` se utiliza como parte de una expresión cuando el usuario desea que el cálculo utilice la fracción del mes que ha transcurrido hasta el momento. El usuario tiene la oportunidad, mediante el uso de una variable, de seleccionar el período de su elección. Por ejemplo, `monthsend()` puede proporcionar una variable de entrada para permitir que el usuario calcule el interés total aún no devengado durante el mes, trimestre o semestre.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
`monthsend(4, '07/19/2013')`	Devuelve 08/31/2013.
`monthsend(4, '10/19/2013', -1)`	Devuelve 08/31/2013.
`monthsend(4, '10/19/2013', 0, 2)`	Devuelve 01/31/2014. Porque el inicio del año se convierte en el mes 2.

Ejemplo 1: ejemplo básico

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022 se carga en una tabla llamada "`Transactions`".
- Un campo de fecha proporcionado en el formato `(MM/DD/YYYY)` de la variable del sistema `DateFormat`.
- Una instrucción load anterior que contiene lo siguiente:

- La función monthsend que está definida como el campo "bi_monthly_end". Esto agrupa las transacciones en segmentos bimensuales.
- La función timestamp que devuelve la marca de tiempo inicial del segmento de cada transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

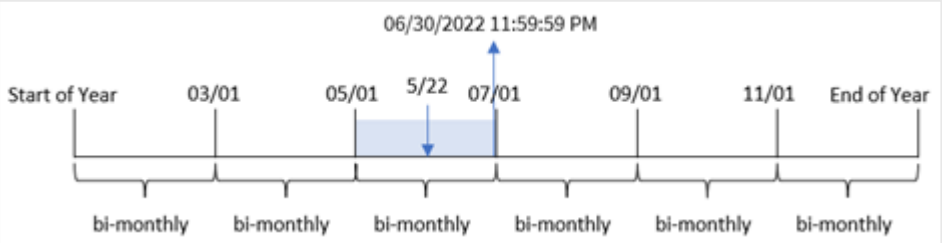
- id
- date
- bi_monthly_end
- bi_monthly_end_timestamp

Tabla de resultados

id	date	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	5/22/2022	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

El campo "`bi_monthly_end`" se crea en la instrucción load anterior, mediante el uso de la función `monthsend ()`. El primer argumento proporcionado es `2`, dividiendo el año en segmentos bimensuales. El segundo argumento identifica qué campo se está evaluando.

Diagrama de la función `monthsend` con segmentos bimensuales.



La transacción 8195 tuvo lugar el 22 de mayo. La función `monthsend()` inicialmente divide el año en segmentos bimensuales. La transacción 8195 cae en el segmento entre el 1 de mayo y el 30 de junio. Como resultado, la función devuelve el último milisegundo de este segmento, 30/06/2022 a las 23:59:59.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

En este ejemplo, la tarea es crear un campo, "`prev_bi_monthly_end`'', que devuelva el primer milisegundo del segmento bimensual anterior a la transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

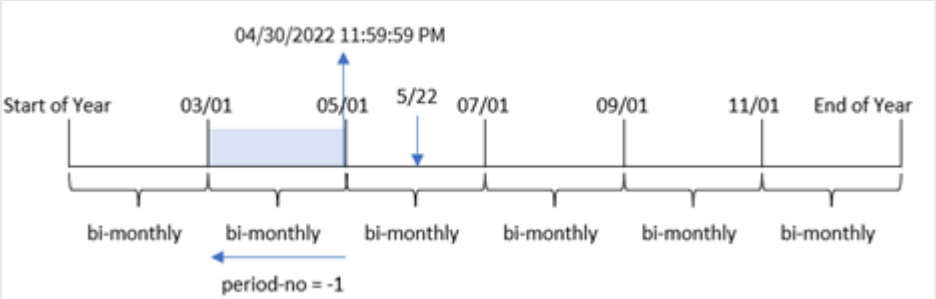
- `id`
- `date`
- `prev_bi_monthly_end`
- `prev_bi_monthly_end_timestamp`

Tabla de resultados

id	date	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	5/22/2022	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

Usando `-1` como argumento de `period_no` en la función `monthsend()`, después de dividir inicialmente un año en segmentos bimensuales, la función devuelve el último milisegundo del segmento bimensual anterior al momento en que se realiza una transacción.

Diagrama de la función `monthsend` que devuelve el segmento bimensual anterior.



La transacción 8195 ocurre en el segmento entre el mayo y junio. Como resultado, el segmento bimensual anterior ocurrió entre el 1 de marzo y el 30 de abril, por lo que la función devuelve el último milisegundo de este segmento, 30/04/2022, 23:59:59.

Ejemplo 3: first_month_of_year

Script de carga y resultados

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

En este ejemplo, la política de la organización es que abril sea el primer mes del año fiscal.

Cree un campo, "bi_monthly_end", que agrupe las transacciones en segmentos bimensuales y devuelva la marca de tiempo del último milisegundo del segmento de cada transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`
- `bi_monthly_end`
- `bi_monthly_end_timestamp`

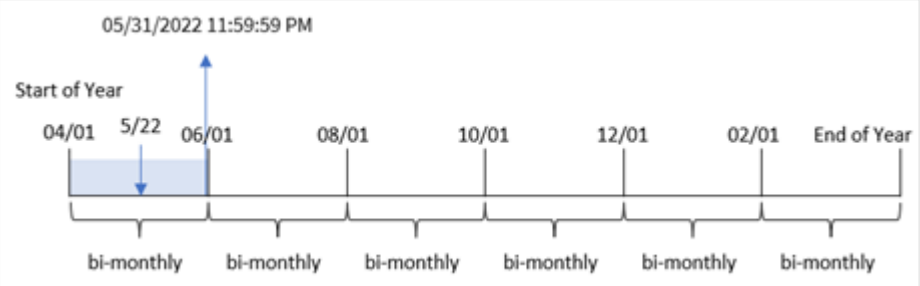
Tabla de resultados

id	date	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	5/22/2022	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	date	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

Al usar `4` como argumento de `first_month_of_year` en la función `monthsend()`, la función comienza el año el 1 de abril. Luego divide el año en segmentos bimensuales: Abr-May, Jun-Jul, Ago-Sep, Oct-Nov, Dic-Ene, Feb-Mar.

Diagrama de la función `monthsend` con abril establecido como primer mes del año



La transacción 8195 tuvo lugar el 22 de mayo y cae en el segmento entre el 1 de abril y el 31 de mayo. Como resultado, la función devuelve el último milisegundo de este segmento, 31/05/2022 23:59:59.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo. Sin embargo, en este ejemplo el conjunto de datos está sin modificar y cargado en la app.

En este ejemplo, la tarea es crear un cálculo que agrupe las transacciones en segmentos bimensuales y devuelva la marca de tiempo del último milisegundo del segmento de cada transacción como una medida en un objeto gráfico de una aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

`date`

Para obtener la marca de tiempo del último milisegundo del segmento bimensual en que se realizó la transacción, cree las siguientes medidas:

- `=monthsEnd(2,date)`
- `=timestamp(monthsend(2,date))`

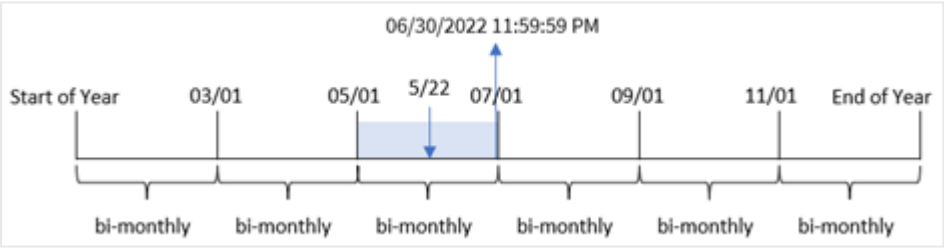
Tabla de resultados

id	date	=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	5/22/2022	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	date	=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

El campo "bi_monthly_end" se crea como una medida en el objeto gráfico usando la función monthsend(). El primer argumento proporcionado es 2, que divide el año en segmentos bimensuales. El segundo argumento identifica qué campo se está evaluando.

Diagrama de la función monthsend con segmentos bimensuales.



La transacción 8195 tuvo lugar el 22 de mayo. La función monthsend() inicialmente divide el año en segmentos bimensuales. La transacción 8195 cae en el segmento entre el 1 de mayo y el 30 de junio. Como resultado, la función devuelve el primer milisegundo de este segmento, 30/06/2022 23:59:59.

Ejemplo 5: escenario

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

En este ejemplo, un conjunto de datos se carga en una tabla denominada "Employee_Expenses". La tabla contiene los siguientes campos:

- Employee IDs
- Employee names

- Las reclamaciones de gastos diarios promedio de cada empleado.

Al usuario final le gustaría tener un gráfico que muestre, por ID y nombre de empleado, la declaración de gastos estimada para el resto de un período de su elección. El año fiscal comienza en enero.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una nueva hoja.

Al comienzo del script de carga se crea una variable, `vPeriod`, que se vinculará al control de entrada de la variable.

Haga lo siguiente:

1. En el panel de activos, haga clic en **Objetos personalizados**.
2. Seleccione **Qlik Dashboard bundle**, cree un objeto **Entrada de variables**.
3. Escriba un título para el objeto gráfico.
4. En **Variable**, seleccione **vPeriod** como nombre y configure el objeto para que se muestre como un **Menú desplegable**.
5. En **Valores**, haga clic en **Valores dinámicos**. Escriba lo siguiente:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`

Cree una nueva tabla y estos campos como dimensiones:

- `employee_id`
- `employee_name`

Para calcular el interés acumulado, cree esta medida:

```
=floor(monthsend($(vPeriod),today(1))-today(1))*avg_daily_claim
```

Esta medida es dinámica y producirá diferentes resultados en la tabla dependiendo de la fecha en que cargue los datos.

Defina el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

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

La función `monthsend()` utiliza lo que ha introducido el usuario como primer argumento y la fecha de hoy como segundo argumento. Esto devuelve la fecha de finalización del período de tiempo seleccionado por el usuario. Luego, la expresión devuelve el número de días que quedan del período de tiempo seleccionado al restar la fecha de hoy de esta fecha de finalización.

Luego, este valor se multiplica por la reclamación de gastos diaria promedio de cada empleado, para calcular el valor estimado de las reclamaciones que se espera que haga cada empleado en los días restantes de ese período.

monthsname

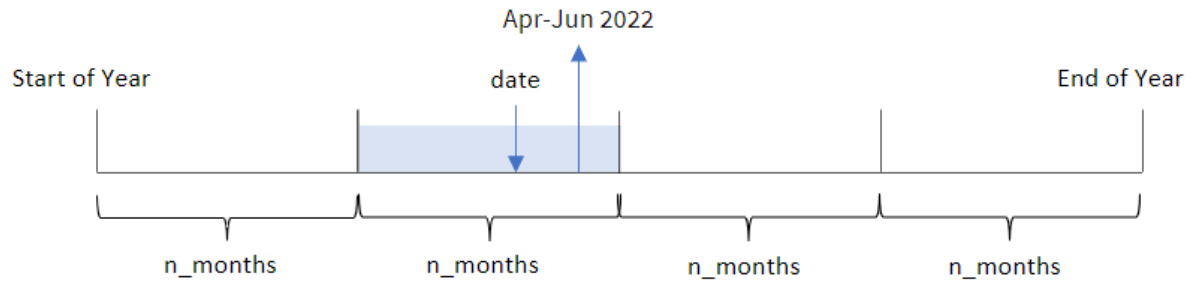
Esta función devuelve un valor de visualización que representa el rango de los meses del período (con formato de acuerdo con la variable de script **MonthNames**), así como el año. El valor numérico subyacente corresponde a una marca de tiempo del primer milisegundo del periodo mensual, bimensual, trimestral, cuatrimestral o semestral que contiene una fecha base.

Sintaxis:

```
MonthsName(n_months, date[, period_no[, first_month_of_year]])
```

Tipo de datos que devuelve: dual

Diagrama de la función monthsname



La función monthsname() divide el año en segmentos basándose en el argumento n_months proporcionado. A continuación, evalúa el segmento al que pertenece cada uno de los date proporcionados y devuelve los nombres de los meses de inicio y finalización de ese segmento, así como el año. La función también brinda la capacidad de devolver estos límites de segmentos anteriores o posteriores, así como redefinir cuál será el primer mes del año.

Los siguientes segmentos del año están disponibles en la función n_month como argumentos:

Posibles argumentos de n_month

Períodos	**Número de meses**
mes	1
bimestre	2
trimestre	3
cuatrimestre	4
semestre	6

Argumentos

Argumento	**Descripción**
n_months	El número de meses que define el periodo. Un entero o expresión que devuelve un entero que debe ser uno de los siguientes: 1 (equivalente a la función inmonth()), 2 (bimestral), 3 (equivalente a la función inquarter()), 4 (cuatrimestral) o 6 (medio año).
date	La fecha o marca de tiempo para evaluar.
period_no	El período se puede desplazar mediante **period_no**, un entero o una expresión que devuelve un entero, donde el valor 0 indica el período que contiene a **base_date**. Los valores negativos en **period_no** indican períodos precedentes y los valores positivos indican períodos subsiguientes.
first_month_of_year	Si desea trabajar con años (fiscales) que no comiencen en enero, indique un valor entre 2 y 12 en **first_month_of_year**.

Cuándo se utiliza

La función monthsname() es útil cuando desea proporcionar al usuario la funcionalidad de comparar agregaciones por un período de su elección. Por ejemplo, podría proporcionar una variable de entrada para que el usuario pueda ver las ventas totales de productos por mes, trimestre o semestre.

Estas dimensiones se pueden crear en el script de carga agregando la función como un campo en una tabla de Calendario maestro o, alternativamente, creando la dimensión directamente en un gráfico como una dimensión calculada.

Ejemplos de funciones

Ejemplo	Resultado
`monthsname(4, '10/19/2013')`	Devuelve "Sep-Dec 2013". En este y en los demás ejemplos, la sentencia **SET Monthnames** está configurada en Jan;Feb;Mar, y así sucesivamente.
`monthsname(4, '10/19/2013', -1)`	Devuelve "May-Aug 2013".
`monthsname(4, '10/19/2013', 0, 2)`	Devuelve "Oct-Jan 2014", ya que el año se especifica para comenzar en el mes 2. Por lo tanto, el período de cuatro meses termina el primer mes del año siguiente.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: ejemplo básico

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo, `bi_monthly_range`, que agrupa las transacciones en segmentos bimensuales y devuelve los nombres de límite de ese segmento para cada transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `bi_monthly_range`

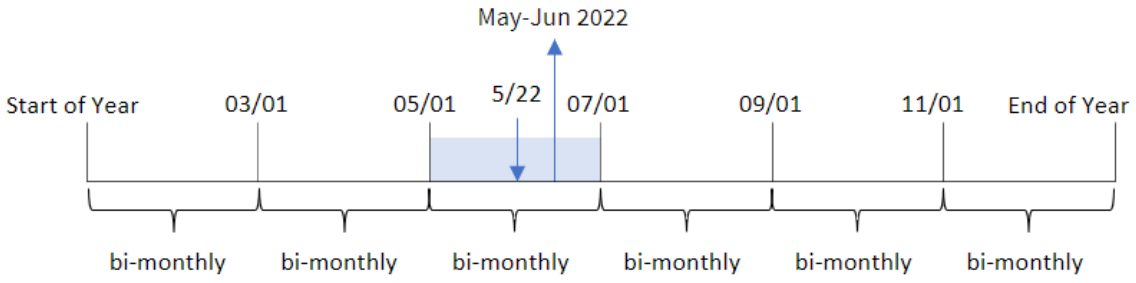
Tabla de resultados

date	bi_monthly_range
2/19/2022	Ene-Feb 2022
3/7/2022	Mar-Abr 2022
3/30/2022	Mar-Abr 2022
4/5/2022	Mar-Abr 2022
4/16/2022	Mar-Abr 2022
5/1/2022	May-Jun 2022
5/7/2022	May-Jun 2022

date	bi_monthly_range
5/22/2022	May-Jun 2022
6/15/2022	May-Jun 2022
6/26/2022	May-Jun 2022
7/9/2022	Jul-Ago 2022
7/22/2022	Jul-Ago 2022
7/23/2022	Jul-Ago 2022
7/27/2022	Jul-Ago 2022
8/2/2022	Jul-Ago 2022
8/8/2022	Jul-Ago 2022
8/19/2022	Jul-Ago 2022
9/26/2022	Sep-Oct 2022
10/14/2022	Sep-Oct 2022
10/29/2022	Sep-Oct 2022

El campo "`bi_monthly_range`" se crea en la instrucción load anterior, mediante el uso de la función monthsname(). El primer argumento proporcionado es 2, dividiendo el año en segmentos bimensuales. El segundo argumento identifica qué campo se está evaluando.

Diagrama de la función monthsname, ejemplo básico



La transacción 8195 tuvo lugar el 22 de mayo. La función `monthsname()` inicialmente divide el año en segmentos bimensuales. La transacción 8195 cae en el segmento entre el 1 de mayo y el 30 de junio. Por lo tanto, la función devuelve estos meses en el formato de la variable de sistema `MonthNames`, así como el año, mayo-junio de 2022.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos inline y el mismo escenario que en el primer ejemplo.
- La creación de un campo, `prev_bi_monthly_range`, que agrupa las transacciones en segmentos bimensuales y devuelve los nombres de los límites del segmento anterior para cada transacción.

Agregue su otro texto aquí, según sea necesario, con listas, etc.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

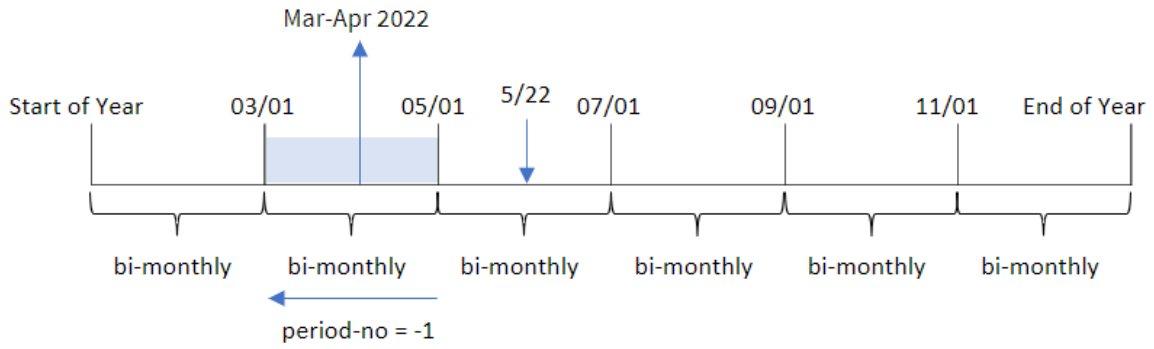
- `date`
- `prev_bi_monthly_range`

Tabla de resultados

date	prev_bi_monthly_range
2/19/2022	Nov-Dic 2021
3/7/2022	Ene-Feb 2022
3/30/2022	Ene-Feb 2022
4/5/2022	Ene-Feb 2022
4/16/2022	Ene-Feb 2022
5/1/2022	Mar-Abr 2022
5/7/2022	Mar-Abr 2022
5/22/2022	Mar-Abr 2022
6/15/2022	Mar-Abr 2022
6/26/2022	Mar-Abr 2022
7/9/2022	May-Jun 2022
7/22/2022	May-Jun 2022
7/23/2022	May-Jun 2022
7/27/2022	May-Jun 2022
8/2/2022	May-Jun 2022
8/8/2022	May-Jun 2022
8/19/2022	May-Jun 2022
9/26/2022	Jul-Ago 2022
10/14/2022	Jul-Ago 2022
10/29/2022	Jul-Ago 2022

En este ejemplo, `-1` se usa como el argumento `period_no` de la función `monthsname()`. Tras dividir inicialmente un año en segmentos bimensuales, la función devuelve a continuación los límites del segmento anterior a cuando se realiza una transacción.

Diagrama de la función monthsname, ejemplo de period_no



La transacción 8195 ocurre en el segmento entre el mayo y junio. Por lo tanto, el segmento bimensual anterior fue entre el 1 de marzo y el 30 de abril, por lo que la función devuelve Mar-Abr 2022.

Ejemplo 3: first_month_of_year

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos inline y el mismo escenario que en el primer ejemplo.
- La creación de un campo diferente, `bi_monthly_range`, que agrupa las transacciones en segmentos bimensuales y devuelve los límites de segmento de cada transacción.

Sin embargo, en este ejemplo, también debemos establecer abril como el primer mes del año fiscal.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `bi_monthly_range`

Tabla de resultados

date	bi_monthly_range
2/19/2022	Feb-Mar 2021
3/7/2022	Feb-Mar 2021
3/30/2022	Feb-Mar 2021
4/5/2022	Abr-May 2022
4/16/2022	Abr-May 2022
5/1/2022	Abr-May 2022
5/7/2022	Abr-May 2022
5/22/2022	Abr-May 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

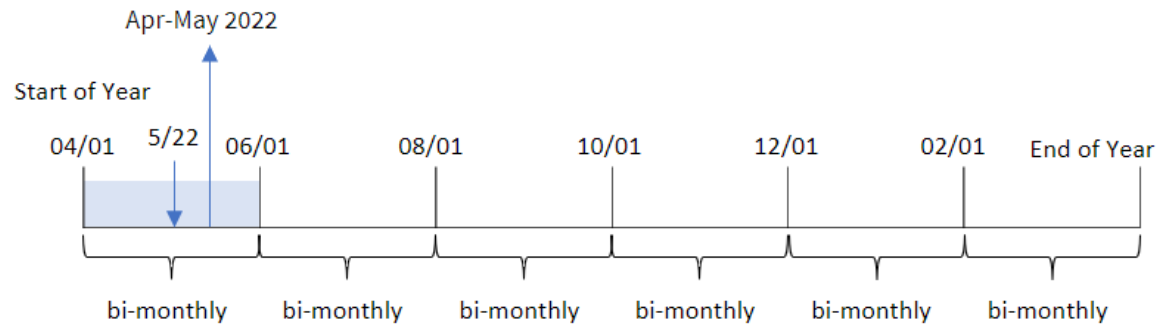
date	bi_monthly_range
8/2/2022	Ago-Sep 2022
8/8/2022	Ago-Sep 2022
8/19/2022	Ago-Sep 2022
9/26/2022	Ago-Sep 2022
10/14/2022	Oct-Nov 2022
10/29/2022	Oct-Nov 2022

Utilizando 4 como el argumento de `first_month_of_year` en la función `monthsname()`, la función comienza el año el 1 de abril y luego divide el año en segmentos bimensuales. Abr-May,Jun-Jul,Ago-Sep,Oct-Nov,Dic-Ene,Feb-Mar.

Texto de párrafo para Resultados.

La transacción 8195 tuvo lugar el 22 de mayo y cae en el segmento entre el 1 de abril y el 31 de mayo. Por lo tanto, la función devuelve abril-mayo de 2022.

Diagrama de la función monthsname, ejemplo de first_month_of_year



Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos inline y escenario que el primer ejemplo. Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que agrupa las transacciones en segmentos bimensuales y devuelve los límites del segmento para cada transacción se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:`date`.

Cree la siguiente medida:

```
=monthsname(2,date)
```

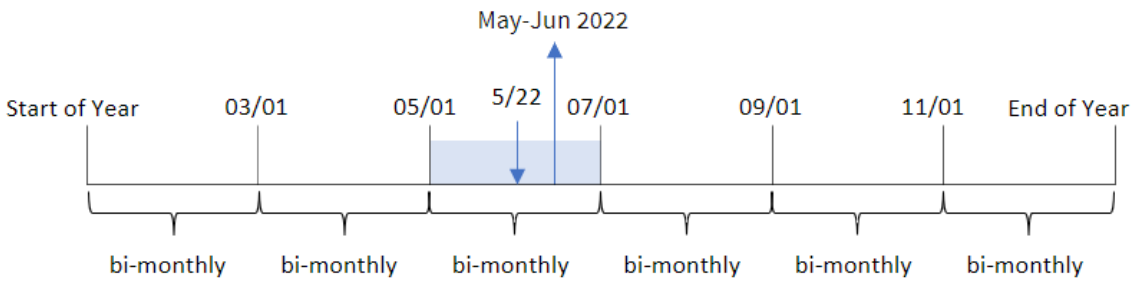
Tabla de resultados

date	=monthsname(2,date)
2/19/2022	Ene-Feb 2022
3/7/2022	Mar-Abr 2022
3/30/2022	Mar-Abr 2022
4/5/2022	Mar-Abr 2022
4/16/2022	Mar-Abr 2022
5/1/2022	May-Jun 2022

date	=monthsname(2,date)
5/7/2022	May-Jun 2022
5/22/2022	May-Jun 2022
6/15/2022	May-Jun 2022
6/26/2022	May-Jun 2022
7/9/2022	Jul-Ago 2022
7/22/2022	Jul-Ago 2022
7/23/2022	Jul-Ago 2022
7/27/2022	Jul-Ago 2022
8/2/2022	Jul-Ago 2022
8/8/2022	Jul-Ago 2022
8/19/2022	Jul-Ago 2022
9/26/2022	Sep-Oct 2022
10/14/2022	Sep-Oct 2022
10/29/2022	Sep-Oct 2022

El campo `bi_monthly_range` se crea como una medida en el objeto gráfico usando la función `monthsname()`. El primer argumento proporcionado es 2, dividiendo el año en segmentos bimensuales. El segundo argumento identifica qué campo se está evaluando.

Diagrama de la función monthsname, ejemplo de objeto gráfico



La transacción 8195 tuvo lugar el 22 de mayo. La función `monthsname()` inicialmente divide el año en segmentos bimensuales. La transacción 8195 cae en el segmento entre el 1 de mayo y el 30 de junio. Por lo tanto, la función devuelve estos meses en el formato de la variable de sistema `MonthNames`, así como el año, mayo-junio de 2022.

Ejemplo 5: Escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada `Transactions`.
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat` (MM/DD/AAAA).

Al usuario final le gustaría tener un objeto gráfico que muestre las ventas totales por un período de su elección. Esto podría lograrse incluso cuando esta dimensión no esté disponible en el modelo de datos, utilizando la función `monthsname()` como una dimensión calculada que se modifica dinámicamente mediante un control de entrada variable.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja.

Al comienzo del script de carga, se ha creado una variable (`vPeriod`) que se vinculará al control de entrada de la variable. A continuación, configure la variable como un objeto personalizado en la hoja.

Haga lo siguiente:

1. En el panel de activos, haga clic en **Objetos personalizados**.
2. Seleccione **Qlik Dashboard bundle** y cree un objeto **Entrada de variables**.
3. Escriba un título para el objeto gráfico.
4. En **Variable**, seleccione **vPeriod** como Nombre y configure el objeto para que se muestre como un **Menú desplegable**.
5. En **Valores**, configure el objeto para que utilice valores dinámicos. Escriba lo siguiente:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

A continuación, cree la tabla de resultados.

Haga lo siguiente:

1. Cree una nueva tabla y agregue la siguiente dimensión calculada:
   `=monthsname($(vPeriod),date)`
2. Agregue esta medida para calcular el total de ventas:
   `=sum(amount)`
3. Establezca el **Formato numérico** de la medida en **Moneda**. Haga clic en ✔ **Edición finalizada**. Ahora puede modificar los datos que se muestran en la tabla ajustando el segmento de tiempo en el objeto variable.

Así es como se verá la tabla de resultados con la opción `tertial` seleccionada:

Tabla de resultados

monthsname($(vPeriod),date)	=sum(amount)
Ene-Abr 2022	253.89
May-ago 2022	713.58
Sep-Dic 2022	248.12

monthsstart

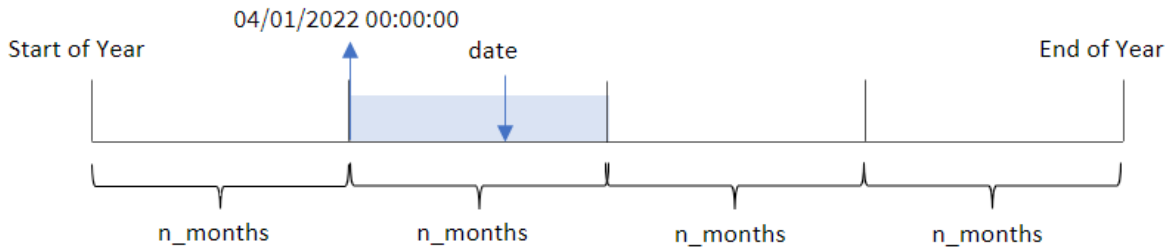
Esta función devuelve un valor correspondiente a la marca de tiempo del primer milisegundo del periodo mensual, bimensual, trimestral, cuatrimestral o semestral que contiene una fecha base. También es posible hallar la marca de tiempo de un periodo anterior o posterior.El formato de salida predefinido es el **DateFormat** definido en el script.

Sintaxis:

```
MonthsStart(n_months, date[, period_no [, first_month_of_year]])
```

Tipo de datos que devuelve: dual

Diagrama de la función monthsstart()



La función monthsstart() divide el año en segmentos basándose en el argumento n_months proporcionado. Luego evalúa en qué segmento cae cada fecha proporcionada y devuelve el primer milisegundo de ese segmento, en formato de fecha. La función también brinda la capacidad de devolver la marca de tiempo de inicio de los segmentos anteriores o posteriores, así como redefinir cuál habrá de ser el primer mes del año.

Los siguientes segmentos del año están disponibles en la función n_month como argumentos:

Posibles argumentos de n_month

Períodos	Número de meses
mes	1
bimestre	2
trimestre	3
cuatrimestre	4
semestre	6

Argumentos

Argumento	Descripción
n_months	El número de meses que define el periodo. Un entero o expresión que devuelve un entero que debe ser uno de los siguientes: 1 (equivalente a la función inmonth()), 2 (bimestral), 3 (equivalente a la función inquarter()), 4 (cuatrimestral) o 6 (medio año).
date	La fecha o marca de tiempo para evaluar.
period_no	El período se puede desplazar mediante **period_no**, un entero o una expresión que devuelve un entero, donde el valor 0 indica el período que contiene a **base_date**. Los valores negativos en **period_no** indican períodos precedentes y los valores positivos indican períodos subsiguientes.
first_month_of_year	Si desea trabajar con años (fiscales) que no comiencen en enero, indique un valor entre 2 y 12 en **first_month_of_year**.

Cuándo se utiliza

La función `monthsstart()` se suele utilizar como parte de una expresión cuando el usuario desea que el cálculo utilice la fracción de un período que aún no ha transcurrido. Esto podría usarse, por ejemplo, para tener una variable de entrada que permita al usuario calcular el interés total que se ha acumulado hasta el momento en el mes, trimestre o semestre.

Ejemplos de funciones

Ejemplo	**Resultado**
`monthsstart(4, '10/19/2013')`	Devuelve 09/01/2013.
`monthsstart(4, '10/19/2013, -1)`	Devuelve 05/01/2013.
`monthsstart(4, '10/19/2013', 0, 2 )`	Devuelve 10/01/2013, porque el inicio del año se convierte en el mes 2.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo, `bi_monthly_start`, que agrupa las transacciones en segmentos bimensuales y devuelve la marca de tiempo inicial del segmento para cada transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `bi_monthly_start`
- `bi_monthly_start_timestamp`

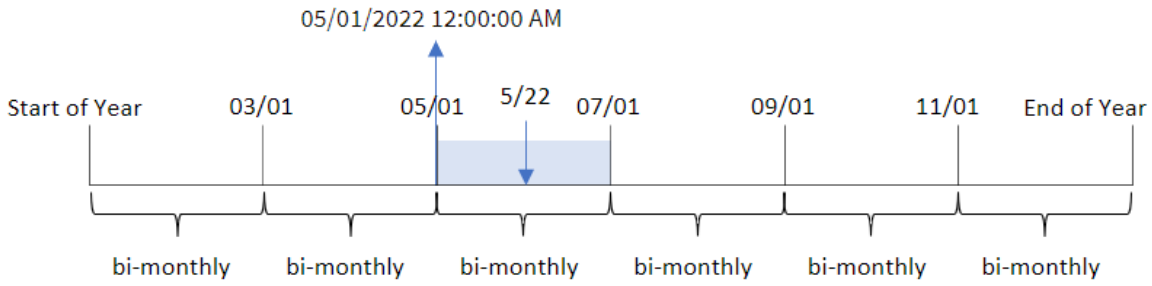
Tabla de resultados

date	bi_monthly_start	bi_monthly_start_timestamp
2/19/2022	01/01/2022	1/1/2022 12:00:00 AM
3/7/2022	03/01/2022	3/1/2022 12:00:00 AM

date	bi_monthly_start	bi_monthly_start_timestamp
3/30/2022	03/01/2022	3/1/2022 12:00:00 AM
4/5/2022	03/01/2022	3/1/2022 12:00:00 AM
4/16/2022	03/01/2022	3/1/2022 12:00:00 AM
5/1/2022	05/01/2022	5/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/22/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
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

El campo "`bi_monthly_start`" se crea en la instrucción load anterior, mediante el uso de la función monthsstart(). El primer argumento proporcionado es 2, dividiendo el año en segmentos bimensuales. El segundo argumento identifica qué campo se está evaluando.

Diagrama de la función `monthsstart()`, ejemplo sin argumentos adicionales



La transacción 8195 tuvo lugar el 22 de mayo. La función `monthsstart()` inicialmente divide el año en segmentos bimensuales. La transacción 8195 cae en el segmento entre el 1 de mayo y el 30 de junio. Por lo tanto, la función devuelve el primer milisegundo de este segmento, 1 de mayo de 2022 a las 00:00.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- Creación de un campo, `prev_bi_monthly_start`, que devuelve el primer milisegundo del segmento bimensual anterior a la transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

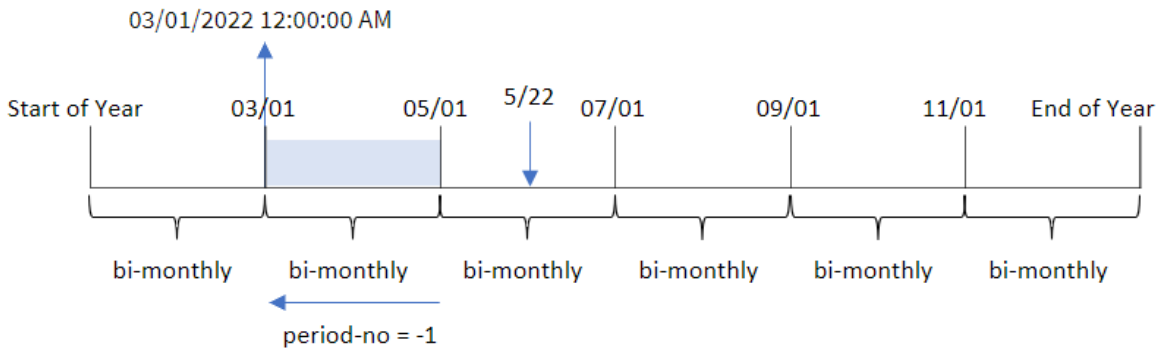
- `date`
- `prev_bi_monthly_start`
- `prev_bi_monthly_start_timestamp`

Tabla de resultados

date	prev_bi_monthly_start	prev_bi_monthly_start_timestamp
2/19/2022	11/01/2021	11/1/2021 12:00:00 AM
3/7/2022	01/01/2022	1/1/2022 12:00:00 AM
3/30/2022	01/01/2022	1/1/2022 12:00:00 AM
4/5/2022	01/01/2022	1/1/2022 12:00:00 AM
4/16/2022	01/01/2022	1/1/2022 12:00:00 AM
5/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	03/01/2022	3/1/2022 12:00:00 AM
5/22/2022	03/01/2022	3/1/2022 12:00:00 AM
6/15/2022	03/01/2022	3/1/2022 12:00:00 AM
6/26/2022	03/01/2022	3/1/2022 12:00:00 AM
7/9/2022	05/01/2022	5/1/2022 12:00:00 AM
7/22/2022	05/01/2022	5/1/2022 12:00:00 AM
7/23/2022	05/01/2022	5/1/2022 12:00:00 AM
7/27/2022	05/01/2022	5/1/2022 12:00:00 AM
8/2/2022	05/01/2022	5/1/2022 12:00:00 AM
8/8/2022	05/01/2022	5/1/2022 12:00:00 AM
8/19/2022	05/01/2022	5/1/2022 12:00:00 AM
9/26/2022	07/01/2022	7/1/2022 12:00:00 AM
10/14/2022	07/01/2022	7/1/2022 12:00:00 AM
10/29/2022	07/01/2022	7/1/2022 12:00:00 AM

Usando `-1` como argumento de `period_no` en la función `monthsstart()`, después de dividir inicialmente un año en segmentos bimensuales, la función devuelve el primer milisegundo del segmento bimensual anterior al momento en que se realizó una transacción.

Diagrama de la función `monthsstart()`, ejemplo de period_no



La transacción 8195 ocurre en el segmento entre el mayo y junio. Por lo tanto, el segmento bimensual anterior fue entre el 1 de marzo y el 30 de abril, por lo que la función devuelve el primer milisegundo de este segmento, el 1 de marzo de 2022 a las 00:00.

Ejemplo 3: first_month_of_year

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `bi_monthly_start`, que agrupa las transacciones en segmentos bimensuales y devuelve la marca de tiempo inicial del conjunto de cada transacción.

Sin embargo, en este ejemplo, también debemos establecer abril como el primer mes del año fiscal.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `bi_monthly_start`
- `bi_monthly_start_timestamp`

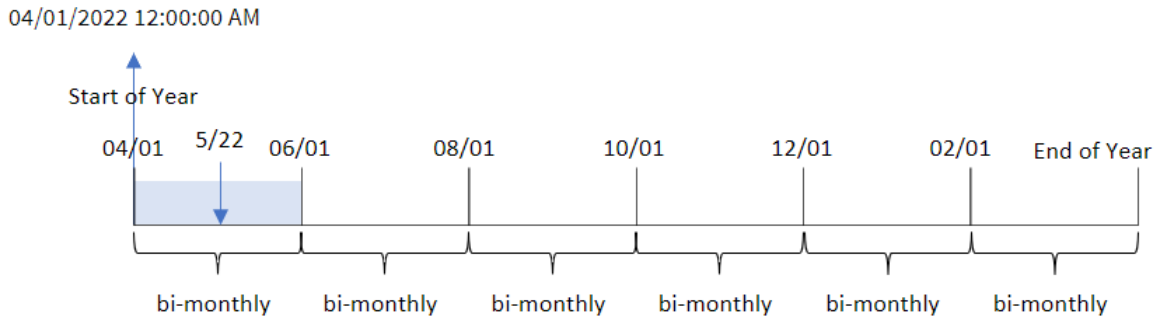
Tabla de resultados

date	bi_monthly_start	bi_monthly_start_timestamp
2/19/2022	02/01/2022	2/1/2022 12:00:00 AM
3/7/2022	02/01/2022	2/1/2022 12:00:00 AM
3/30/2022	02/01/2022	2/1/2022 12:00:00 AM
4/5/2022	04/01/2022	4/1/2022 12:00:00 AM
4/16/2022	04/01/2022	4/1/2022 12:00:00 AM
5/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/22/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM

date	bi_monthly_start	bi_monthly_start_timestamp
7/23/2022	06/01/2022	6/1/2022 12:00:00 AM
7/27/2022	06/01/2022	6/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
9/26/2022	08/01/2022	8/1/2022 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

Utilizando 4 como el argumento de `first_month_of_year` en la función `monthsstart()`, la función comienza el año el 1 de abril y luego divide el año en segmentos bimensuales. Abr-May,Jun-Jul,Ago-Sep,Oct-Nov,Dic-Ene,Feb-Mar.

Diagrama de la función `monthsstart()`, ejemplo de first_month_of_year



La transacción 8195 tuvo lugar el 22 de mayo y cae en el segmento entre el 1 de abril y el 31 de mayo. Por lo tanto, la función devuelve el primer milisegundo de este segmento, 1 de abril de 2022 a las 00:00.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que agrupa las transacciones en segmentos bimensuales y devuelve la marca de tiempo de inicio del conjunto para cada transacción se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Cree las siguientes medidas:

```
=monthsstart(2,date)
```

```
=timestamp(monthsstart(2,date))
```

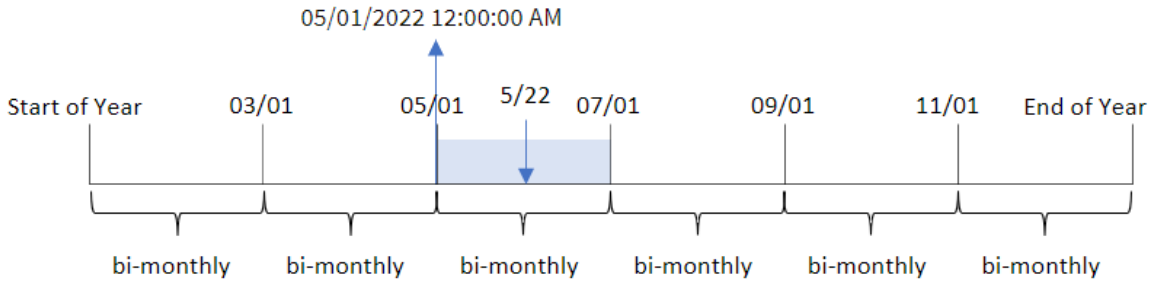
Estos cálculos recuperarán el momento de inicio del segmento bimensual en el que tuvo lugar cada transacción.

Tabla de resultados

date	=monthsstart(2,date)	=timestamp(monthsstart(2,date))
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

date	=monthsstart(2,date)	=timestamp(monthsstart(2,date))
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
5/1/2022	05/01/2022	5/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/22/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM
3/7/2022	03/01/2022	3/1/2022 12:00:00 AM
3/30/2022	03/01/2022	3/1/2022 12:00:00 AM
4/5/2022	03/01/2022	3/1/2022 12:00:00 AM
4/16/2022	03/01/2022	3/1/2022 12:00:00 AM
2/19/2022	01/01/2022	1/1/2021 12:00:00 AM

Diagrama de la función `monthsstart()`, ejemplo de objeto gráfico



La transacción 8195 tuvo lugar el 22 de mayo. La función `monthsstart()` inicialmente divide el año en segmentos bimensuales. La transacción 8195 cae en el segmento entre el 1 de mayo y el 30 de junio. Por lo tanto, la función devuelve el primer milisegundo de este segmento, 1 de mayo de 2022 a las 12:00 a. m.

Ejemplo 5: escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de saldos de préstamos, que se carga en una tabla llamada `Loans`.
- Los datos consisten en varios ID de préstamos, el saldo al comienzo del mes y la tasa de interés simple cobrada en cada préstamo por año.

Al usuario final le gustaría tener un objeto gráfico que muestre, por ID de préstamo, el interés actual que se ha acumulado en cada préstamo durante el período de su elección. El año fiscal comienza en enero.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja.

Al comienzo del script de carga, se ha creado una variable (`vPeriod`) que se vinculará al control de entrada de la variable. A continuación, configure la variable como un objeto personalizado en la hoja.

Haga lo siguiente:

1. En el panel de activos, haga clic en **Objetos personalizados**.
2. Seleccione **Qlik Dashboard bundle** y cree un objeto **Entrada de variables**.
3. Escriba un título para el objeto gráfico.
4. En **Variable**, seleccione **vPeriod** como Nombre y configure el objeto para que se muestre como un **Menú desplegable**.

5. En **Valores**, configure el objeto para que utilice valores dinámicos. Escriba lo siguiente:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

A continuación, cree la tabla de resultados.

Haga lo siguiente:

1. Cree una nueva tabla. Agregue los siguientes campos como dimensiones.
   - `employee_id`
   - `employee_name`
2. Cree una medida para calcular el interés acumulado:
   `=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)`
3. Establezca el **Formato numérico** de la medida en **Moneda**. Haga clic en ✔ **Edición finalizada**. Ahora puede modificar los datos que se muestran en la tabla ajustando el segmento de tiempo en el objeto variable.

Así es como se verá la tabla de resultados con la opción del período `month` seleccionada:

Tabla de resultados

loan_id	start_balance	=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)
8188	$10000.00	$7.95
8189	$15000.00	$67.93
8190	$17500.00	$33.37
8191	$21000.00	$56.73
8192	$90000.00	$600.66

La función `monthsstart()` utiliza lo que ha introducido el usuario como primer argumento y la fecha de hoy como segundo argumento y devuelve la fecha de inicio del período elegido por el usuario. Al restar ese resultado de la fecha actual, la expresión devuelve el número de días que han transcurrido en lo que va de este período.

Luego, este valor se multiplica por la tasa de interés y se divide por 365 para obtener la tasa de interés efectiva en que se ha incurrido durante este período. A continuación el resultado se multiplica por el saldo inicial del préstamo para devolver el interés que se ha acumulado en lo que va de este período.

monthstart

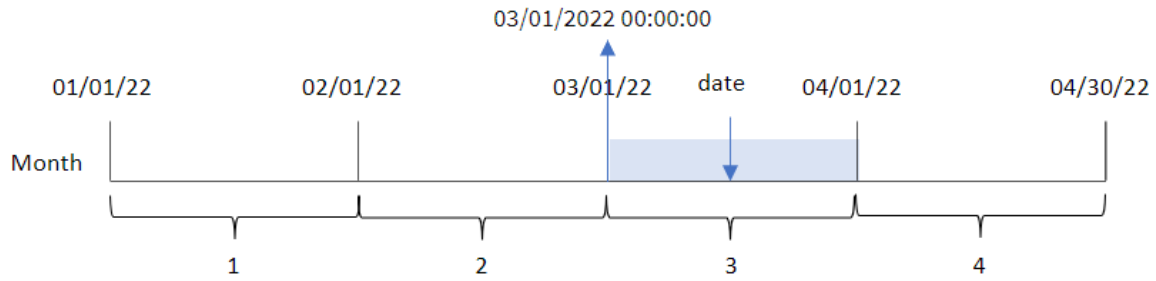
Esta función devuelve un valor correspondiente a una marca de tiempo (fecha-hora) del primer milisegundo del primer día del mes que contiene a **date**. El formato de salida predeterminado será el **DateFormat** establecido en el script.

Sintaxis:

```
MonthStart(date[, period_no])
```

Tipo de datos que devuelve: dual

Diagrama de la función `monthstart()`



La función `monthstart()` determina en qué mes cae la fecha. Luego devuelve una marca de tiempo, en formato de fecha, con el primer milisegundo de ese mes.

Argumentos

Argumento	Descripción
date	La fecha o marca de tiempo para evaluar.
period_no	**period_no** es un número entero que, si es 0 o se omite, indica el mes que contiene **date**. Los valores negativos en **period_no** indican meses precedentes y los valores positivos indican meses subsiguientes.

Cuándo se utiliza

La función `monthstart()` se suele utilizar como parte de una expresión cuando el usuario desea que el cálculo utilice la fracción del mes que ya ha transcurrido. Por ejemplo, se puede utilizar para calcular el interés que se ha acumulado en un mes hasta una fecha determinada.

Ejemplos de funciones

Ejemplo	Resultado
`monthstart('10/19/2001')`	Devuelve 10/01/2001.
`monthstart('10/19/2001', -1)`	Devuelve 09/01/2001.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las

fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo, `start_of_month`, que devuelve una marca de tiempo con el inicio del mes en que se realizaron las transacciones.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `start_of_month`
- `start_of_month_timestamp`

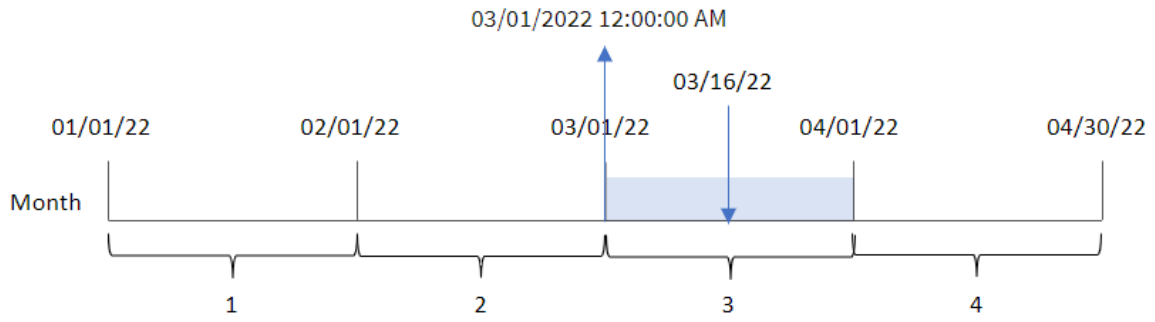
Tabla de resultados

date	start_of_month	start_of_month_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 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
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	07/01/2022	6/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
9/26/2022	09/01/2022	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	10/1/2022 12:00:00 AM

El campo `start_of_month` se crea en la instrucción load anterior utilizando la función `monthstart()` e introduciendo el campo de fecha como argumento de la función.

La función `monthstart()` identifica en qué mes cae el valor de la fecha y devuelve una marca de tiempo con el primer milisegundo de ese mes.

Diagrama de la función `monthstart()`, ejemplo sin argumentos adicionales



La transacción 8192 tuvo lugar el 16 de marzo. La función `monthstart()` devuelve el primer milisegundo de ese mes, que es el 1 de marzo a las 12:00:00.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `previous_month_start`, que devuelve una marca de tiempo con el inicio del mes anterior a que se realizara la transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `previous_month_start`
- `previous_month_start_timestamp`

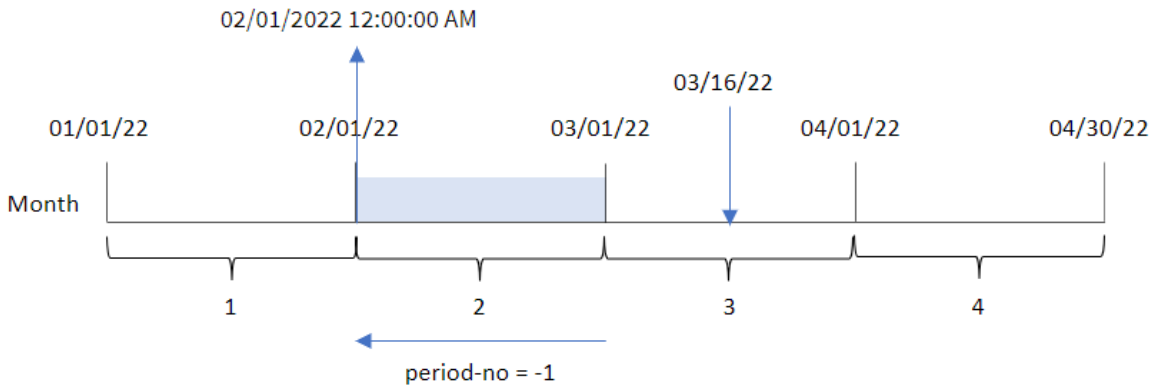
Tabla de resultados

date	previous_month_start	previous_month_start_timestamp
1/7/2022	12/01/2021	12/1/2021 12:00:00 AM
1/19/2022	12/01/2021	12/1/2021 12:00:00 AM
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	1/1/2022 12:00:00 AM
3/16/2022	02/01/2022	2/1/2022 12:00:00 AM
4/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/16/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM

date	previous_month_start	previous_month_start_timestamp
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM
7/23/2022	06/01/2022	6/1/2022 12:00:00 AM
7/27/2022	06/01/2022	6/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
9/26/2022	08/01/2022	8/1/2022 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

En este caso, debido a que se usó un `period_no` de -1 como argumento de desplazamiento en la función `monthstart()`, la función primero identifica el mes en el que se realizan las transacciones. Luego cambia a un mes antes e identifica el primer milisegundo de ese mes.

Diagrama de la función `monthstart()`, ejemplo de period_no



La transacción 8192 tuvo lugar el 16 de marzo. La función `monthstart()` identifica que el mes anterior a la transacción fue febrero. Luego devuelve el primer milisegundo de ese mes, el 1 de febrero a las 12:00:00.

Ejemplo 3: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que devuelve una marca de tiempo del inicio del mes en que se realizaron las transacciones se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Para calcular la fecha de inicio del mes en que tiene lugar una transacción, cree las siguientes medidas:

- `=monthstart(date)`
- `=timestamp(monthstart(date))`

Tabla de resultados

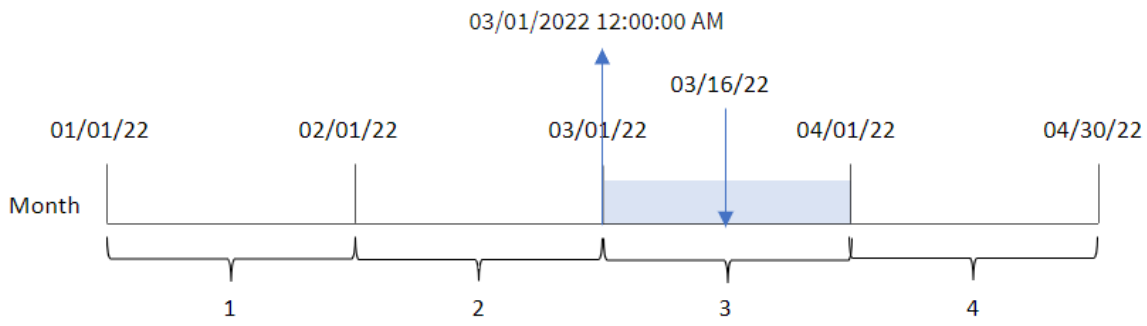
date	=monthstart(date)	=timestamp(monthstart(date))
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

date	=monthstart(date)	=timestamp(monthstart(date))
9/26/2022	09/01/2022	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	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
5/7/2022	05/01/2022	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	4/1/2022 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	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM

La medida `start_of_month` se crea en el objeto gráfico utilizando la función `monthstart()` e indicando el campo de fecha como argumento de la función.

La función `monthstart()` identifica en qué mes cae el valor de la fecha y devuelve una marca de tiempo del primer milisegundo de ese mes.

Diagrama de la función `monthstart()`, ejemplo de objeto gráfico



La transacción 8192 tuvo lugar el 16 de marzo. La función `monthstart()` identifica que la transacción tuvo lugar en marzo y devuelve el primer milisegundo de ese mes, que es el 1 de marzo a las 12:00 del mediodía.

Ejemplo 4: Escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de saldos de préstamos, que se carga en una tabla llamada `Loans`.
- Los datos consisten en varios ID de préstamos, el saldo al comienzo del mes y la tasa de interés simple cobrada en cada préstamo por año.

Al usuario final le gustaría tener un objeto gráfico que muestre, por ID de préstamo, el interés actual que se ha acumulado en cada préstamo en el mes hasta la fecha.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:
   - `loan_id`
   - `start_balance`
2. Ahora cree una medida para calcular el interés acumulado:
   `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
3. Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

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

La función `monthstart()`, utilizando la fecha de hoy como único argumento, devuelve la fecha de inicio del mes actual. Restando ese resultado de la fecha actual, la expresión devuelve el número de días que han transcurrido en lo que va de este mes.

Luego, este valor se multiplica por la tasa de interés y se divide por 365 para obtener la tasa de interés efectiva en que se ha incurrido durante este período. Luego, el resultado se multiplica por el saldo inicial del préstamo para devolver los intereses que se han acumulado en lo que va del mes.

networkdays

La función **networkdays** devuelve el número de días laborables (de lunes a viernes) entre e incluidos los días **start_date** y **end_date** teniendo en cuenta cualquier listado opcional de vacaciones: **holiday**.

Sintaxis:

```
networkdays (start_date, end_date [, holiday])
```

Tipo de datos que devuelve: Entero

Diagrama de calendario que muestra el intervalo de fechas devuelto por la función networkdays

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			

La función `networkdays` tiene las siguientes limitaciones:

- No existe ningún método para modificar los días laborables. En otras palabras, no hay forma de modificar la función para regiones o situaciones que impliquen otra cosa que no sea trabajar de lunes a viernes.
- El parámetro `holiday` debe ser una constante de cadena de texto. No se aceptan expresiones.

Argumentos

Argumento	Descripción
start_date	La fecha inicial que se ha de evaluar.
end_date	La fecha final que se ha de evaluar.
holiday	Los períodos de vacaciones que deben excluirse de los días laborables. Las vacaciones se enuncian como cadena de fecha constante. Puede especificar más fechas de vacaciones separadas por comas. **Ejemplo:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Cuándo se utiliza

La función `networkdays()` se suele utilizar como parte de una expresión cuando el usuario desea que el cálculo utilice el número de días laborables de la semana que ocurren entre dos fechas. Por ejemplo, si un usuario quisiera calcular los salarios totales que ganará un empleado en un contrato PAYE (pago según los

ingresos).

Ejemplos de funciones

Ejemplo	Resultado
`networkdays ('12/19/2013', '01/07/2014')`	Devuelve 14. Este ejemplo no tiene las vacaciones en cuenta.
`networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013')`	Devuelve 12. Este ejemplo tiene en cuenta el período de vacaciones de 12/25/2013 a 12/26/2013.
`networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014')`	Devuelve 10. Este ejemplo tiene en cuenta dos períodos de vacaciones.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: ejemplo básico

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene los ID de proyecto, sus fechas de inicio y sus fechas de finalización. Esta información se carga en una tabla llamada `Projects`.
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo adicional, `net_work_days`, para calcular el número de días laborables de cada proyecto.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `start_date`
- `end_date`
- `net_work_days`

Tabla de resultados

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	13

Debido a que no hay días de vacaciones programados (esto habría estado presente en el tercer argumento de la función `networkdays()`), la función resta la `start_date` de la `end_date`, así como todos los fines de semana, para calcular el número de días laborables entre las dos fechas.

Diagrama de calendario que destaca los días laborables del proyecto 5 (sin vacaciones)

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			

El calendario anterior describe visualmente el proyecto con un `id` de 5. El proyecto 5 comienza el miércoles 10 de agosto de 2022 y finaliza el 26 de agosto de 2022. Con todos los sábados y domingos ignorados, hay 13 días hábiles entre estas dos fechas, inclusive.

Ejemplo 2: un día de vacaciones

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- El mismo conjunto de datos y escenario del ejemplo anterior.
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat` (MM/DD/AAAA).
- La creación de un campo adicional, `net_work_days`, para calcular el número de días laborables de cada proyecto.

En este ejemplo, hay un día libre programado para el 19 de agosto de 2022.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- id
- start_date
- end_date
- net_work_days

Tabla de resultados

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	12

El único día libre programado se indica como tercer argumento en la función `networkdays()`.

Diagrama de calendario que destaca los días laborables del proyecto 5 (un único día de vacaciones)

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			

El calendario anterior describe visualmente el proyecto 5, lo que demuestra este ajuste para incluir el día que no se trabaja. Este día libre ocurre en el proyecto 5, el viernes 19 de agosto de 2022. Como resultado, el valor total `net_work_days` del proyecto 5 disminuye en un día, de 13 a 12 días.

Ejemplo 3: varios días de vacaciones

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat` (MM/DD/AAAA).
- La creación de un campo adicional, `net_work_days`, para calcular el número de días laborables de cada proyecto.

Sin embargo, en este ejemplo, hay cuatro días de vacaciones programados, del 18 al 21 de agosto de 2022.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- id
- start_date
- end_date
- net_work_days

Tabla de resultados

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	11

Los cuatro días de vacaciones programados se indican como una lista separada por comas, desde el tercer argumento en adelante, en la función networkdays().

Diagrama de calendario que destaca los días laborables del proyecto 5 (varios días de vacaciones)

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			

El calendario anterior describe visualmente el proyecto 5, demostrando este ajuste para incluir los días de vacaciones. Este período programado de vacaciones se produce durante el proyecto 5, y dos de los días son jueves y viernes. Como resultado, el valor total `net_work_days` del proyecto 5 disminuye de 13 a 11 días.

Ejemplo 4: Un día de vacaciones

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat` (MM/DD/AAAA).

En este ejemplo, hay un día libre programado para el 19 de agosto de 2022.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El campo `net_work_days` se calcula como una medida en un objeto gráfico.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `start_date`
- `end_date`

Cree la siguiente medida:

```
= networkdays(start_date,end_date,'08/19/2022')
```

Tabla de resultados

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	12

El único día libre programado se indica como tercer argumento en la función `networkdays()`.

Diagrama de calendario que muestra los días de trabajo con un solo día de vacaciones (objeto gráfico)

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			

El calendario anterior describe visualmente el proyecto 5, lo que demuestra este ajuste para incluir el día que no se trabaja. Este día libre ocurre en el proyecto 5, el viernes 19 de agosto de 2022. Como resultado, el valor total `net_work_days` del proyecto 5 disminuye en un día, de 13 a 12 días.

now

Esta función devuelve una marca de tiempo con la hora actual. La función devuelve valores en el formato de la variable del sistema **TimeStamp**. El valor predeterminado es 1 **timer_mode**.

Sintaxis:

```
now([ timer_mode])
```

Tipo de datos que devuelve: dual

La función `now()` se puede utilizar en el script de carga o en los objetos del gráfico.

Argumentos

Argumento	Descripción
timer_mode	Puede tener los siguientes valores: 0 (hora de la última carga de datos terminada) 1 (hora de la llamada ala función) 2 (hora en que se abrió la app) *Si utiliza la función en un script de carga de datos, **timer_mode=0** dará como resultado la hora de la última carga de datos finalizada, mientras que **timer_mode=1** dará la hora de la llamada a la función en la carga de datos actual.*

La función `now()` tiene un alto impacto en el rendimiento, lo que podría dar lugar a problemas de desplazamiento si la función se utiliza dentro de las expresiones de las tablas. Siempre que su uso no sea estrictamente necesario, se recomienda utilizar la función `today()` en su lugar. Si se requiere el uso de `now()` en un diseño, se recomienda utilizar la configuración no predeterminada `now(0)` o `now(2)` cuando sea posible, ya que no requieren recálculos constantes

Cuándo se utiliza

La función `now()` se utiliza habitualmente como un componente dentro de una expresión. Por ejemplo, se puede utilizar para calcular el tiempo restante en el ciclo de vida de un producto. Se utilizaría la función `now()` en lugar de la función `today()` cuando la expresión requiera el uso de una fracción de día.

La tabla siguiente ofrece una explicación del resultado que devuelve la función `now()`, dados diferentes valores para el argumento de `timer_mode`:

Ejemplos de funciones

valor de timer_mode	Resultado si se usa en el script de carga	Resultado si se usa en el objeto gráfico
0	Devuelve una marca de tiempo, en el formato de la variable del sistema `TimeStamp`, de la última recarga de datos correcta anterior a la última recarga de datos.	Devuelve una marca de tiempo, en el formato de la variable del sistema `TimeStamp`, de la última recarga de datos.
1	Devuelve una marca de tiempo, en el formato de la variable del sistema `TimeStamp`, de la última recarga de datos.	Devuelve una marca de tiempo, en el formato de la variable del sistema `TimeStamp`, de la llamada a la función.

valor de timer_mode	Resultado si se usa en el script de carga	Resultado si se usa en el objeto gráfico
2	Devuelve una marca de tiempo, en el formato de la variable del sistema `TimeStamp`, de cuándo comenzó la sesión del usuario en la aplicación. Esto no se actualizará a menos que el usuario vuelva a cargar el script.	Devuelve la marca de tiempo, en el formato de la variable del sistema `TimeStamp`, de cuándo comenzó la sesión del usuario en la aplicación. Esto se actualizará una vez que comience una nueva sesión o se vuelvan a cargar los datos en la aplicación.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: generación de objetos utilizando script de carga

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

Este ejemplo crea tres variables usando la función `now()`. Cada variable utiliza una de las opciones `timer_mode` para demostrar su efecto.

Para que las variables cumplan su finalidad, vuelva a cargar la secuencia de script y luego, tras un breve período de tiempo, vuelva a cargar la secuencia de script por segunda vez. Esto dará como resultado que las variables de `now(0)` y `now(1)` muestren valores diferentes, cumpliendo así correctamente su propósito.

Script de carga

```
LET vPreviousDataLoad = now(0);
LET vCurrentDataLoad = now(1);
LET vApplicationOpened = now(2);
```

Resultados

Una vez cargados los datos por segunda vez, siga las instrucciones que se indican a continuación para crear tres cuadros de texto.

En primer lugar, cree un cuadro de texto para los datos que se cargaron anteriormente.

Haga lo siguiente:

1. Cree un cuadro de texto con el objeto de gráfico **Texto e imagen**.
2. Agregue al objeto la medida siguiente:
   `=vPreviousDataLoad`
3. En **Aspecto**, seleccione **Show titles** y agregue al objeto el título "Hora de recarga anterior".

A continuación, cree un cuadro de texto para los datos que se están cargando actualmente.

Haga lo siguiente:

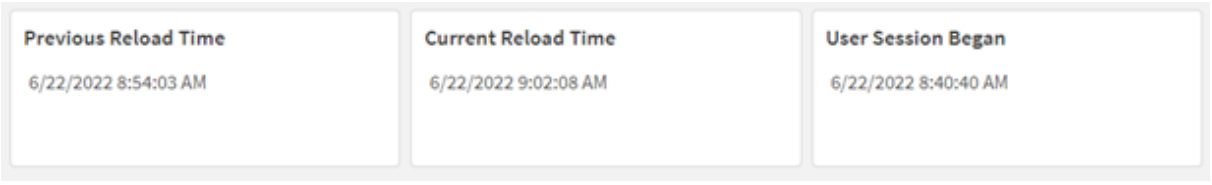
1. Cree un cuadro de texto con el objeto de gráfico **Texto e imagen**.
2. Agregue al objeto la medida siguiente:
   `=vCurrentDataLoad`
3. En **Aspecto**, seleccione **Show titles** y agregue al objeto el título "Hora de recarga actual".

Cree un cuadro de texto final para mostrar cuándo se inició la sesión del usuario en la aplicación.

Haga lo siguiente:

1. Cree un cuadro de texto con el objeto de gráfico **Texto e imagen**.
2. Agregue al objeto la medida siguiente:
   `=vApplicationOpened`
3. En **Aspecto**, seleccione **Show titles** y añada al objeto el título "Inicio de la sesión del usuario".

Variables del script de carga de now()



La imagen superior muestra valores a modo de ejemplo de cada una de las variables creadas. Por ejemplo, los valores podrían ser los siguientes:

- Tiempo de recarga anterior: 22/6/2022 8:54:03 a. m.
- Tiempo de recarga actual: 22/6/2022 9:02:08 a. m.
- Comienzo de la sesión de usuario: 22/6/2022 8:40:40 a. m.

Ejemplo 2: generación de objetos sin script de carga

Script de carga y expresión de gráfico

Vista general

En este ejemplo, creará tres objetos gráficos usando la función `now()`, sin cargar ninguna variable o dato en la aplicación. Cada objeto gráfico utiliza una de las opciones `timer_mode` para demostrar su efecto.

No hay script de carga para este ejemplo.

Haga lo siguiente:

1. Abra el Editor de carga de datos.
2. Sin cambiar el script de carga existente, haga clic en **Cargar datos**.
3. Tras un breve período de tiempo, cargue el script por segunda vez.

Resultados

Una vez que los datos se hayan cargado por segunda vez, cree tres cuadros de texto.

Primero, cree un cuadro de texto para la última recarga de datos.

Haga lo siguiente:

1. Cree un cuadro de texto con el objeto gráfico **Texto e imagen**.
2. Agregue la siguiente medida.
   `=now(0)`
3. En **Aspecto**, seleccione **Mostrar títulos** y agregue al objeto el título "Última recarga de datos".

A continuación, cree un cuadro de texto para mostrar la hora actual.

Haga lo siguiente:

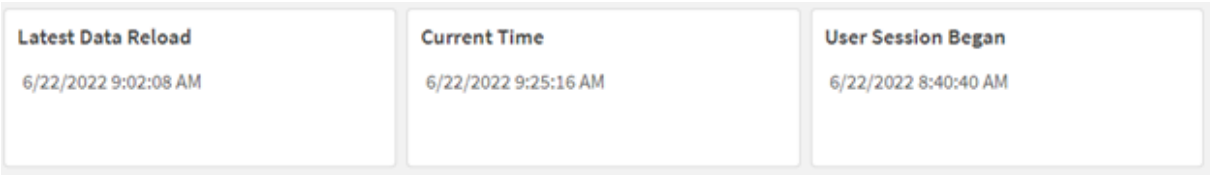
1. Cree un cuadro de texto con el objeto gráfico **Texto e imagen**.
2. Agregue la siguiente medida:
   `=now(1)`
3. En **Aspecto**, seleccione **Mostrar títulos** y agregue al objeto el título "Hora actual".

Cree un cuadro de texto final para mostrar cuándo se inició la sesión del usuario en la aplicación.

Haga lo siguiente:

1. Cree un cuadro de texto con el objeto gráfico **Texto e imagen**.
2. Agregue la siguiente medida:
   `=now(2)`
3. En **Aspecto**, seleccione **Mostrar títulos** y añada al objeto el título "Comienzo de la sesión de usuario".

Ejemplos de objeto gráfico now()



La imagen superior muestra valores de ejemplo para cada uno de los objetos creados. Por ejemplo, los valores podrían ser los siguientes:

- Última recarga de datos: 6/22/2022 9:02:08 AM
- Hora actual: 6/22/2022 9:25:16 AM
- Comienzo de la sesión de usuario: 22/6/2022 8:40:40 a. m.

El objeto gráfico "Última recarga de datos" utiliza un valor `timer_mode` de 0. Esto devuelve la marca de tiempo de la última vez que los datos se recargaron correctamente.

El objeto gráfico "Hora actual" utiliza un valor `timer_mode` de 1. Esto devuelve la hora actual según el reloj del sistema. Si se actualiza la hoja o el objeto, se actualizará este valor.

El objeto gráfico "Comienzo de la sesión de usuario" utiliza un valor `timer_mode` de 2. Esto devuelve la fecha y hora de cuándo se abrió la aplicación y comenzó la sesión del usuario.

Ejemplo 3: escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que consiste en el inventario de una operación de minería de criptomonedas, que se carga en una tabla llamada `Inventory`.
- Datos con los siguientes campos: `id`, `purchase_date` y `wph` (vatios por hora).

Al usuario le gustaría tener una tabla que muestre, por `id`, el coste total en el que ha incurrido cada equipo de minería en el mes hasta el momento, en términos de consumo de energía.

Este valor debería actualizarse cada vez que se actualice el objeto del gráfico. El coste actual de la electricidad es de 0,0678 dólares por kWH.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `id`.

Cree la siguiente medida:

```
=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
```

Si el objeto gráfico se actualizó el 22/06/2022 a las 10:39:05 a. m., arrojaría los siguientes resultados:

Tabla de resultados

id	=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
8188	$39.18
8189	$49.97
8190	$42.81
8191	$46.13
8192	$44.42
8193	$39.18
8194	$46.83
8195	$81.72
8196	$42.95

id	=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
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
8206	$39.53

Al usuario le gustaría que los resultados del objeto se actualicen cada vez que se actualice el objeto. Por lo tanto, el argumento de `timer_mode` es lo proporcionado para instancias de la función `now()` en la expresión. La marca de tiempo del inicio del mes, identificada mediante la función `now()` como argumento de marca de tiempo en la función `monthstart()`, se resta de la hora actual identificada por la función `now()`. Esto proporciona la cantidad total de tiempo que ha transcurrido hasta ahora este mes, en días.

Este valor se multiplica por 24 (el número de horas en un día) y luego por el valor del campo `wph`.

Para convertir de vatios por hora a kilovatios por hora, el resultado se divide por 1000 antes de multiplicarse finalmente por la tarifa de kWH proporcionada.

quarterend

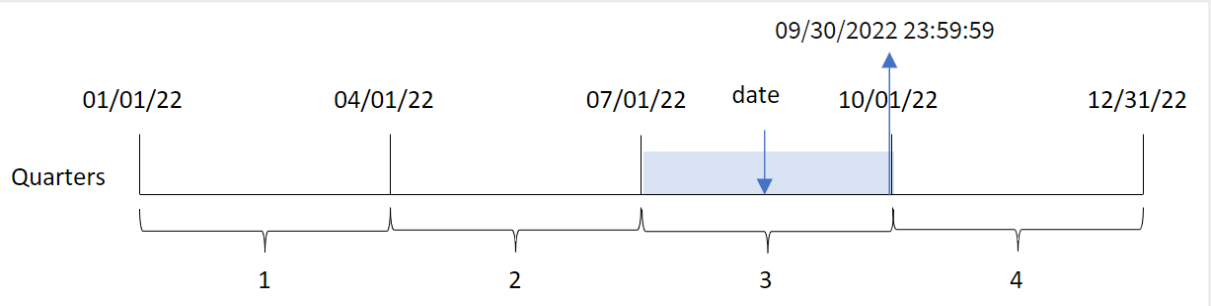
Esta función devuelve un valor correspondiente a una marca de tiempo del último milisegundo del trimestre que contiene a **date**. El formato de salida predeterminado será el **DateFormat** establecido en el script.

Sintaxis:

```
QuarterEnd(date[, period_no[, first_month_of_year]])
```

Tipo de datos que devuelve: dual

Diagrama de la función quarterend()



La función `quarterend()` determina en qué trimestre cae la fecha. Luego devuelve una marca de tiempo, en formato de fecha, con el último milisegundo del último mes de ese trimestre. El primer mes del año es, por defecto, enero. No obstante, también puede cambiar cuál será el primer mes del año usando el argumento `first_month_of_year` en la función `quarterend()`.

> *La función `quarterend()` no considera la variable del sistema `FirstMonthOfYear`. El año comienza el 1 de enero a menos que se use el argumento `first_month_of_year` para cambiarlo.*

Cuándo se utiliza

La función `quarterend()` se suele utilizar como parte de una expresión cuando el usuario desea que el cálculo utilice la fracción del trimestre que aún no ha ocurrido. Por ejemplo, si desea calcular el interés total aún no devengado durante el trimestre.

Argumentos

Argumento	Descripción
date	La fecha o marca de tiempo para evaluar.
period_no	**period_no** es un entero, donde el valor 0 indica el trimestre que contiene a **date**. Los valores negativos en **period_no** indican trimestres precedentes y los valores positivos indican trimestres subsiguientes.
first_month_of_year	Si desea trabajar con años (fiscales) que no comiencen en enero, indique un valor entre 2 y 12 en **first_month_of_year**.

Puede utilizar los siguientes valores para establecer el primer mes del año en el argumento `first_month_of_year`:

first_month_of_year values

Month	Valor
Febrero	2

Month	Valor
Marzo	3
Abril	4
May	5
Junio	6
Julio	7
Agosto	8
Septiembre	9
Octubre	10
Noviembre	11
Diciembre	12

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
`quarterend('10/29/2005')`	Devuelve 12/31/2005 23:59:59.
`quarterend('10/29/2005', -1)`	Devuelve 09/30/2005 23:59:59.
`quarterend('10/29/2005', 0, 3)`	Devuelve 11/30/2005 23:59:59.

Ejemplo 1: ejemplo básico

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "Transactions".
- Un load precedente que contiene lo siguiente:
    - La función quarterend() que se define como el campo "end_of_quarter" y devuelve una marca de tiempo del final del trimestre en que se realizaron las transacciones.
    - La función timestamp() que se define como el campo "end_of_quarter_timestamp" y devuelve la marca de tiempo exacta del final del trimestre seleccionado.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- id
- date
- end_of_quarter
- end_of_quarter_timestamp

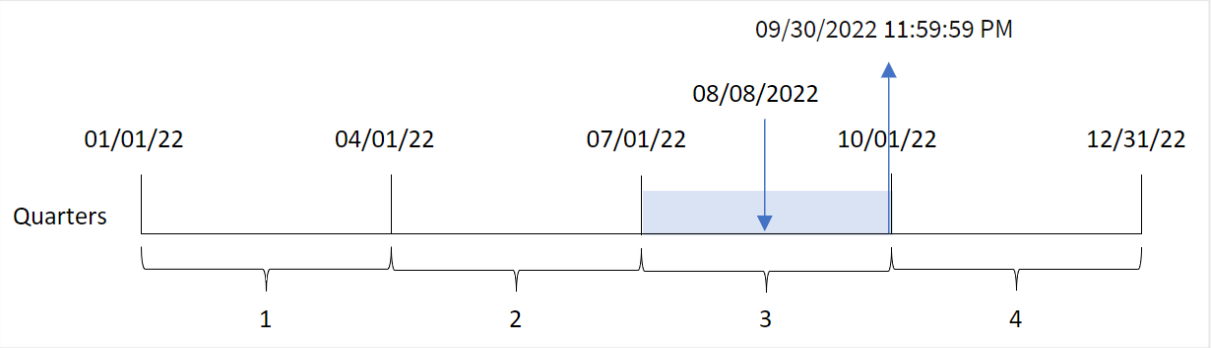
Tabla de resultados

id	date	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

El campo `end_of_quarter` se crea en la instrucción load anterior utilizando la función `quarterend()` e introduciendo el campo de fecha como argumento de la función.

La función `quarterend()` identifica inicialmente en qué trimestre cae el valor de la fecha y devuelve una marca de tiempo del último milisegundo de ese trimestre.

Diagrama de la función `quarterend()` con el final del trimestre de la transacción 8203 identificado



La transacción 8203 tuvo lugar el 8 de agosto. La función `quarterend()` identifica que la transacción tuvo lugar en el tercer trimestre y devuelve el último milisegundo de ese trimestre, que es el 30 de septiembre a las 23:59:59.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- Un load precedente que contiene lo siguiente:
  - La función `quarterend()` que está definida como el campo "`previous_quarter_end`" y devuelve una marca de tiempo del final del trimestre anterior a que se realizara la transacción.
  - La función `timestamp()` que está definida como el campo "`previous_end_of_quarter_timestamp`" y devuelve la marca de tiempo exacta del final del trimestre anterior a que se realizara la transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`
- `previous_quarter_end`
- `previous_quarter_end_timestamp`

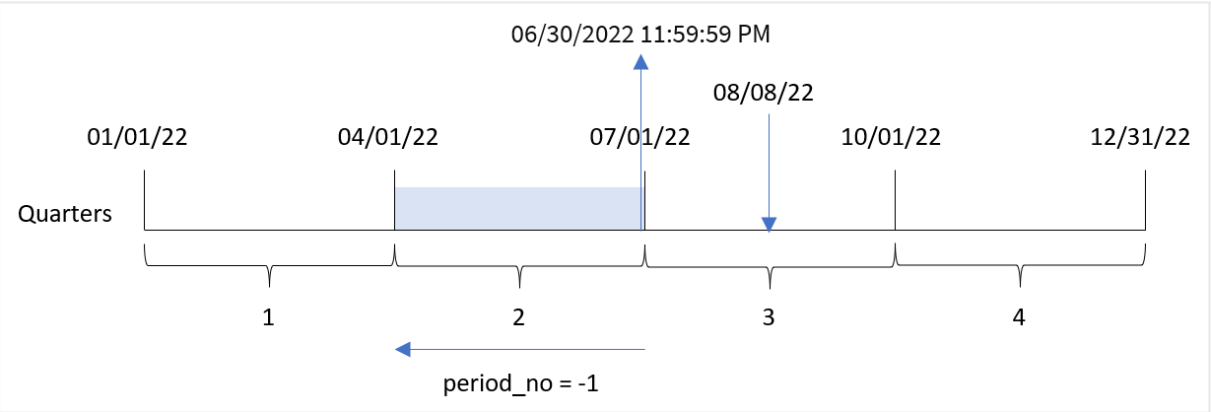
Tabla de resultados

id	date	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
8197	6/26/2022	03/31/2022	3/31/2022 11:59:59 PM

id	date	previous_quarter_end	previous_quarter_end_timestamp
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

Como se usó un valor `period_no` de `-1` como argumento del desplazamiento en la función `quarterend()`, la función identifica primero el trimestre en que se realizan las transacciones. Luego cambia a un trimestre antes e identifica el milisegundo final de ese trimestre.

Diagrama de la función `quarterend()` con un period_no de -1.



La transacción 8203 tuvo lugar el 8 de agosto. La función `quarterend()` identifica que el trimestre anterior a la transacción fue entre el 1 de abril y el 30 de junio. A continuación, la función devuelve el milisegundo final de ese trimestre, el 30 de junio a las 23:59:59.

Ejemplo 3: first_month_of_year

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "Transactions".
- Un load precedente que contiene lo siguiente:
    - La función quarterend() que se define como el campo "end_of_quarter" y devuelve una marca de tiempo del final del trimestre en que se realizaron las transacciones.
    - La función timestamp() que se define como el campo "end_of_quarter_timestamp" y devuelve la marca de tiempo exacta del final del trimestre seleccionado.

Sin embargo, en este ejemplo, la política de la empresa es que el año fiscal comience el 1 de marzo.

Script de carga

```
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
];
```

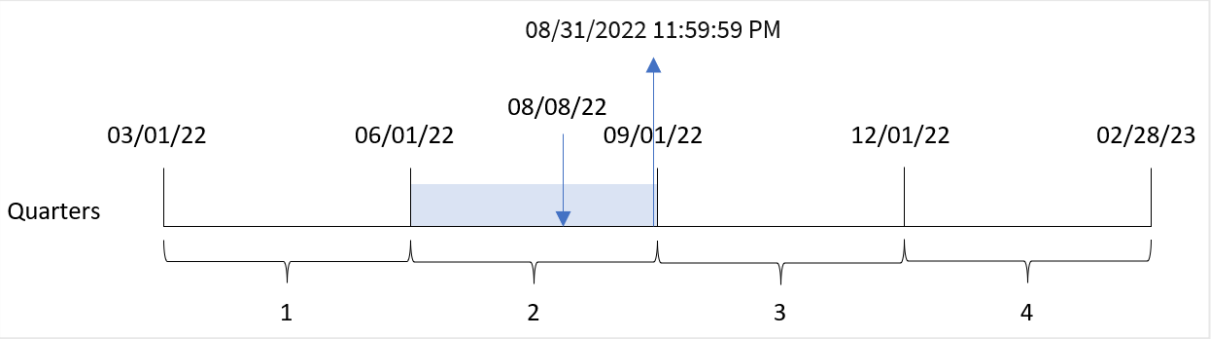
Resultados

Tabla de resultados

id	date	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

Como el argumento `first_month_of_year` de `3` se utiliza en la función `quarterend()`, el inicio del año se desplaza del 1 de enero al 1 de marzo.

Diagrama de la función `quarterend()` con marzo como primer mes del año



La transacción 8203 tuvo lugar el 8 de agosto. Debido a que el comienzo del año es el 1 de marzo, los trimestres del año ocurren entre marzo y mayo, junio y agosto, septiembre y noviembre, y diciembre y febrero.

La función `quarterend()` identifica que la transacción tuvo lugar en el trimestre entre el comienzo de junio y agosto y devuelve el último milisegundo de ese trimestre, que es el 31 de agosto a las 23:59:59.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que devuelve una marca de tiempo del final del trimestre en que se realizaron las transacciones se crea como una medida en un objeto gráfico de la app.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`

Para calcular la fecha de final del trimestre en el que tiene lugar una transacción, cree las siguientes medidas:

- `=quarterend(date)`
- `=timestamp(quarterend(date))`

Tabla de resultados

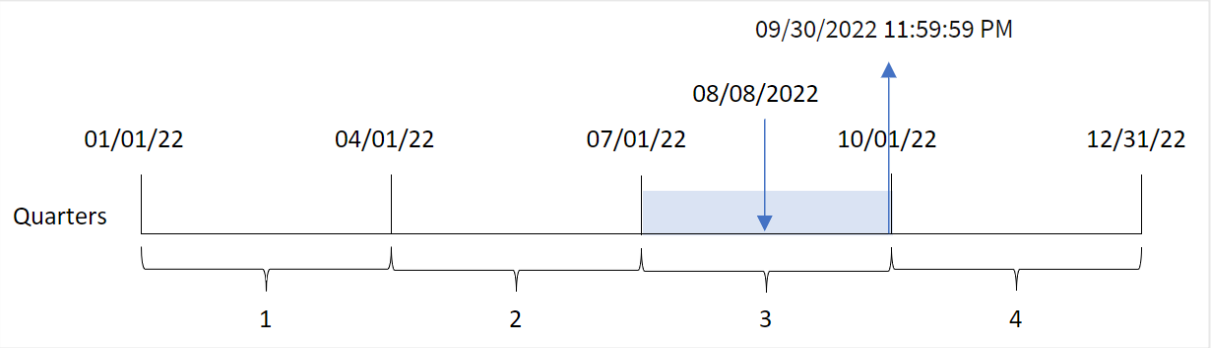
id	date	=quarterend(date)	=timestamp(quarterend(date))
8188	1/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8189	1/19/2022	03/31/2022	3/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	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

id	date	=quarterend(date)	=timestamp(quarterend(date))
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

El campo `end_of_quarter` se crea en la instrucción load anterior utilizando la función `quarterend()` e introduciendo el campo de fecha como argumento de la función.

La función `quarterend()` identifica inicialmente en qué trimestre cae el valor de la fecha y devuelve una marca de tiempo del último milisegundo de ese trimestre.

Diagrama de la función `quarterend()` con el final del trimestre de la transacción 8203 identificado



La transacción 8203 tuvo lugar el 8 de agosto. La función `quarterend()` identifica que la transacción tuvo lugar en el tercer trimestre y devuelve el último milisegundo de ese trimestre, que es el 30 de septiembre a las 23:59:59.

Ejemplo 5: escenario

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos se carga en una tabla denominada "`Employee_Expenses`". La tabla contiene los siguientes campos:
  - Employee IDs
  - Employee names
  - Las reclamaciones de gastos diarios promedio de cada empleado.

Al usuario final le gustaría tener un objeto gráfico que muestre, por ID y nombre de empleado, las reclamaciones de gastos estimados que aún deben realizarse para el resto del trimestre. El año fiscal comienza en enero.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `employee_id`
- `employee_name`

Para calcular el interés acumulado, cree la siguiente medida:

- `=(quarterend(today(1))-today(1))*avg_daily_claim`

Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

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

La función `quarterend()` utiliza la fecha de hoy como único argumento y devuelve la fecha final del mes actual. Luego, resta la fecha de hoy de la fecha de fin del año y la expresión devuelve la cantidad de días que quedan de este mes.

Luego, este valor se multiplica por la reclamación de gastos diarios promedio de cada empleado para calcular el valor estimado de las reclamaciones que se espera que haga cada empleado en el trimestre restante.

quartername

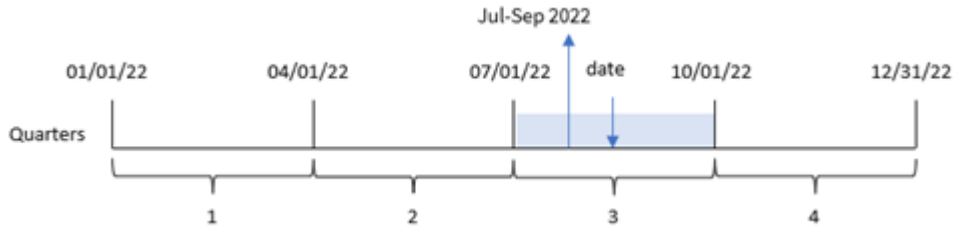
Esta función devuelve un valor de visualización que muestra los meses del trimestre (con formato conforme a la variable de script **MonthNames**) y el año con un valor numérico subyacente correspondiente a una marca de tiempo (una fecha-hora) del primer milisegundo del primer día del trimestre.

Sintaxis:

```
QuarterName(date[, period_no[, first_month_of_year]])
```

Tipo de datos que devuelve: dual

Diagrama de la función quartername()



La función quartername() determina en qué trimestre cae la fecha. A continuación, devuelve un valor que muestra los meses de inicio y fin de este trimestre, así como el año. El valor numérico subyacente de este resultado es el primer milisegundo del trimestre.

Argumentos

Argumento	Descripción
date	La fecha o marca de tiempo para evaluar.
period_no	**period_no** es un entero, donde el valor 0 indica el trimestre que contiene a **date**. Los valores negativos en **period_no** indican trimestres precedentes y los valores positivos indican trimestres subsiguientes.
first_month_of_year	Si desea trabajar con años (fiscales) que no comiencen en enero, indique un valor entre 2 y 12 en **first_month_of_year**.

Cuándo se utiliza

La función quartername() es útil cuando se desea comparar agregaciones por trimestre. Por ejemplo, si desea ver el total de ventas de productos por trimestre.

Esta función podría usarse en el script de carga para crear un campo en una tabla de calendario maestro. Alternativamente, podría usarse directamente en un gráfico como una dimensión calculada.

Estos ejemplos utilizan el formato de fecha MM/DD/AAAA. El formato de fecha se especifica en la sentencia SET DateFormat en la parte superior de su script de carga de datos. Cambie el formato en los ejemplos según se ajuste a sus necesidades.

Ejemplos de funciones

Ejemplo	Resultado
`quartername('10/29/2013')`	Devuelve Oct-Dec 2013.
`quartername('10/29/2013', -1)`	Devuelve Jul-Sep 2013.
`quartername('10/29/2013', 0, 3)`	Devuelve Sep-Nov 2013.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: la fecha sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo, `transaction_quarter`, que devuelve el trimestre en el que se realizaron las transacciones.

Agregue su otro texto aquí, según sea necesario, con listas, etc.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `transaction_quarter`

Tabla de resultados

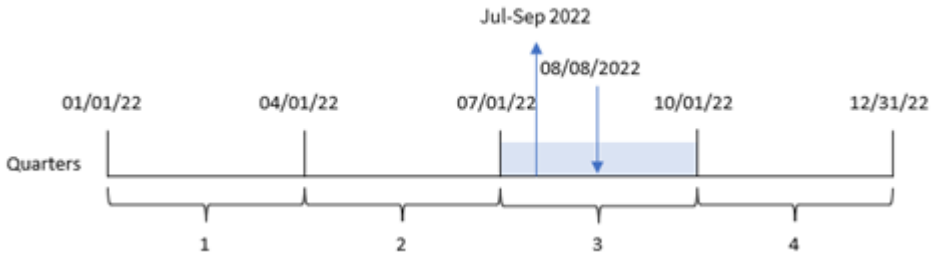
date	transaction_quarter
1/7/2022	Ene-Mar 2022
1/19/2022	Ene-Mar 2022
2/5/2022	Ene-Mar 2022
2/28/2022	Ene-Mar 2022
3/16/2022	Ene-Mar 2022
4/1/2022	Abr-Jun 2022
5/7/2022	Abr-Jun 2022
5/16/2022	Abr-Jun 2022

date	transaction_quarter
6/15/2022	Abr-Jun 2022
6/26/2022	Abr-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	Oct-Dic 2022
10/29/2022	Oct-Dic 2022

El campo `transaction_quarter` se crea en la instrucción load anterior utilizando la función `quartername()` e introduciendo el campo de fecha como argumento de la función.

La función `quartername()` identifica inicialmente el trimestre en el que cae el valor de la fecha. Después devuelve un valor que muestra los meses de inicio y finalización de este trimestre, así como el año.

Diagrama de la función `quartername()`, ejemplo sin argumentos adicionales



La transacción 8203 tuvo lugar el 8 de agosto de 2022. La función `quartername()` identifica que la transacción tuvo lugar en el tercer trimestre y, por lo tanto, devuelve julio-septiembre de 2022. Los meses se muestran en el mismo formato que la variable del sistema `MonthNames`.

Ejemplo 2: fecha con el argumento de period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `previous_quarter`, que devuelve el trimestre anterior a que se realizaran las transacciones.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

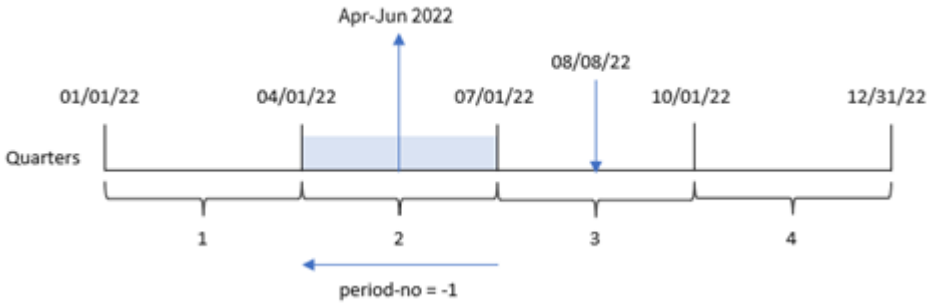
- `date`
- `previous_quarter`

Tabla de resultados

date	previous_quarter
1/7/2022	Oct-Dic 2021
1/19/2022	Oct-Dic 2021
2/5/2022	Oct-Dic 2021
2/28/2022	Oct-Dic 2021
3/16/2022	Oct-Dic 2021
4/1/2022	Ene-Mar 2022
5/7/2022	Ene-Mar 2022
5/16/2022	Ene-Mar 2022
6/15/2022	Ene-Mar 2022
6/26/2022	Ene-Mar 2022
7/9/2022	Abr-Jun 2022
7/22/2022	Abr-Jun 2022
7/23/2022	Abr-Jun 2022
7/27/2022	Abr-Jun 2022
8/2/2022	Abr-Jun 2022
8/8/2022	Abr-Jun 2022
8/19/2022	Abr-Jun 2022
9/26/2022	Abr-Jun 2022
10/14/2022	Jul-Sep 2022
10/29/2022	Jul-Sep 2022

En este caso, debido a que se usó un `period_no` de `-1` como argumento del desplazamiento en la función `quartername()`, la función primero identifica que las transacciones se realizaron en el tercer trimestre. Luego se desplaza un trimestre anterior y devuelve un valor que muestra los meses de inicio y finalización de este trimestre, así como el año.

Diagrama de la función `quartername()`, ejemplo de period_no



La transacción 8203 tuvo lugar el 8 de agosto. La función `quartername()` identifica que el trimestre anterior a la transacción fue entre el 1 de abril y el 30 de junio. Por lo tanto, devuelve de abril a junio de 2022.

Ejemplo 3: fecha con el argumento de first_week_day

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo. Sin embargo, en este ejemplo, necesitamos establecer el 1 de marzo como el comienzo del año fiscal.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `transaction_quarter`

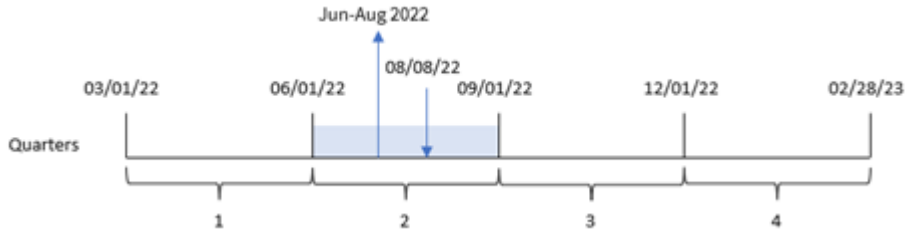
Tabla de resultados

date	transaction_quarter
1/7/2022	Dic-Feb 2021
1/19/2022	Dic-Feb 2021
2/5/2022	Dic-Feb 2021
2/28/2022	Dic-Feb 2021
3/16/2022	Mar-May 2022
4/1/2022	Mar-May 2022
5/7/2022	Mar-May 2022
5/16/2022	Mar-May 2022
6/15/2022	Jun-Ago 2022
6/26/2022	Jun-Ago 2022
7/9/2022	Jun-Ago 2022
7/22/2022	Jun-Ago 2022
7/23/2022	Jun-Ago 2022
7/27/2022	Jun-Ago 2022
8/2/2022	Jun-Ago 2022
8/8/2022	Jun-Ago 2022
8/19/2022	Jun-Ago 2022
9/26/2022	Sep-Nov 2022

date	transaction_quarter
10/14/2022	Sep-Nov 2022
10/29/2022	Sep-Nov 2022

En este caso, como se usa el argumento `first_month_of_year` de 3 en la función `quartername()`, el comienzo del año se mueve del 1 de enero al 1 de marzo. Por lo tanto, los trimestres del año se separan en marzo-mayo, junio-agosto, septiembre-noviembre y diciembre-febrero.

Diagrama de la función `quartername()`, ejemplo de first_week_day



La transacción 8203 tuvo lugar el 8 de agosto. La función `quartername()` identifica que la transacción se realizó en el segundo trimestre, entre principios de junio y finales de agosto. Por lo tanto, devuelve de junio a agosto de 2022.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que devuelve una marca de tiempo del final del trimestre en que se realizaron las transacciones se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Cree la siguiente medida:

```
=quartername(date)
```

Tabla de resultados

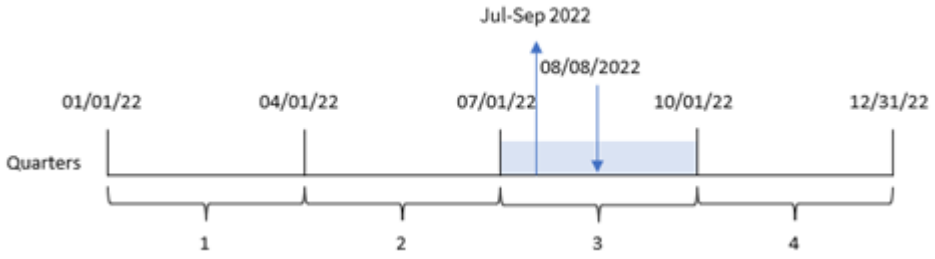
date	=quartername(date)
1/7/2022	Ene-Mar 2022
1/19/2022	Ene-Mar 2022
2/5/2022	Ene-Mar 2022
2/28/2022	Ene-Mar 2022
3/16/2022	Ene-Mar 2022
4/1/2022	Abr-Jun 2022
5/7/2022	Abr-Jun 2022
5/16/2022	Abr-Jun 2022
6/15/2022	Abr-Jun 2022
6/26/2022	Abr-Jun 2022
7/9/2022	Jul-Sep 2022
7/22/2022	Jul-Sep 2022
7/23/2022	Jul-Sep 2022
7/27/2022	Jul-Sep 2022
8/2/2022	Jul-Sep 2022

date	=quartername(date)
8/8/2022	Jul-Sep 2022
8/19/2022	Jul-Sep 2022
9/26/2022	Jul-Sep 2022
10/14/2022	Oct-Dic 2022
10/29/2022	Oct-Dic 2022

La medida `transaction_quarter` se crea en el objeto gráfico utilizando la función `quartername()` e introduciendo el campo `date` como argumento de la función.

La función `quartername()` identifica inicialmente el trimestre en el que cae el valor de la fecha. Después devuelve un valor que muestra los meses de inicio y finalización de este trimestre, así como el año.

Diagrama de la función `quartername()`, ejemplo de objeto gráfico



La transacción 8203 tuvo lugar el 8 de agosto de 2022. La función `quartername()` identifica que la transacción tuvo lugar en el tercer trimestre y, por lo tanto, devuelve julio-septiembre de 2022. Los meses se muestran en el mismo formato que la variable del sistema `MonthNames`.

Ejemplo 5: escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada `Transactions`.
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat` (MM/DD/AAAA).

Al usuario final le gustaría tener un objeto gráfico que presente el total de ventas por trimestre de las transacciones. Esto podría lograrse incluso cuando esta dimensión no esté disponible en el modelo de datos, utilizando la función `quartername()` como una dimensión calculada en el gráfico.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja.Cree una nueva tabla.
2. Cree una dimensión calculada usando la siguiente expresión:
   `=quartername(date)`
3. A continuación calcule el total de ventas con la siguiente medida de agregación:
   `=sum(amount)`
4. Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

=quartername(date)	=sum(amount)
Jul-Sep 2022	$446.31
Abr-Jun 2022	$351.48
Ene-Mar 2022	$253.89
Oct-Dic 2022	$163.91

quarterstart

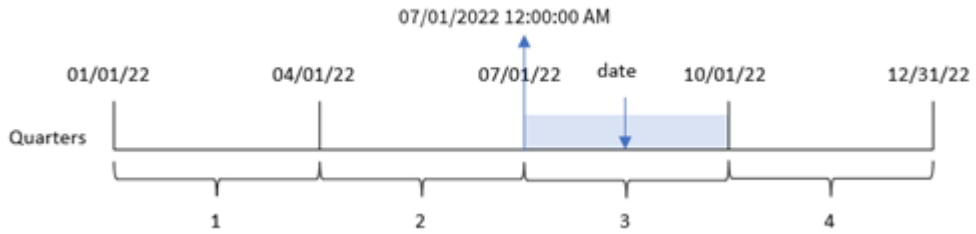
Esta función devuelve un valor correspondiente a una marca de tiempo con el primer milisegundo del trimestre que contiene a **date**. El formato de salida predeterminado será el **DateFormat** establecido en el script.

Sintaxis:

```
QuarterStart(date[, period_no[, first_month_of_year]])
```

Tipo de datos que devuelve: dual

Diagrama de la función quarterstart()



La función `quarterstart()` determina en qué trimestre cae la fecha `date`. Luego devuelve una marca de tiempo, en formato de fecha, con el primer milisegundo del primer mes de ese trimestre.

Argumentos

Argumento	Descripción
date	La fecha o marca de tiempo para evaluar.
period_no	**period_no** es un entero, donde el valor 0 indica el trimestre que contiene a **date**. Los valores negativos en **period_no** indican trimestres precedentes y los valores positivos indican trimestres subsiguientes.
first_month_of_year	Si desea trabajar con años (fiscales) que no comiencen en enero, indique un valor entre 2 y 12 en **first_month_of_year**.

Cuándo se utiliza

La función `quarterstart()` se suele utilizar como parte de una expresión cuando el usuario desea que el cálculo utilice la fracción del trimestre que ya ha transcurrido. Por ejemplo, se puede utilizar para calcular el interés que se ha acumulado en un trimestre hasta la fecha.

Ejemplos de funciones

Ejemplo	Resultado
`quarterstart('10/29/2005')`	Devuelve 10/01/2005.
`quarterstart('10/29/2005', -1 )`	Devuelve 07/01/2005.
`quarterstart('10/29/2005', 0, 3)`	Devuelve 09/01/2005.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo, `start_of_quarter`, que devuelve una marca de tiempo con el inicio del trimestre en el que se realizaron las transacciones.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `start_of_quarter`
- `start_of_quarter_timestamp`

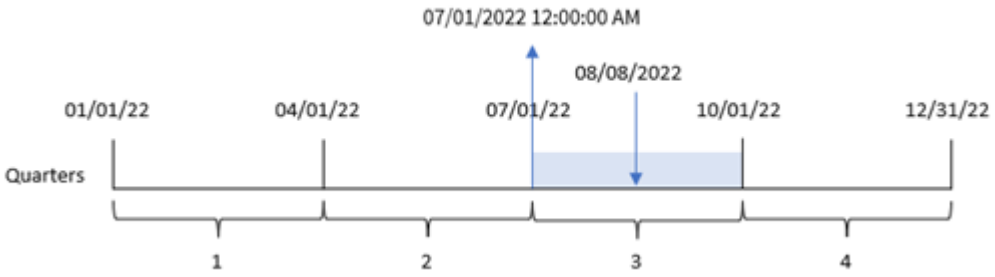
Tabla de resultados

date	start_of_quarter	start_of_quarter_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	1/1/2022 12:00:00 AM
3/16/2022	01/01/2022	1/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2021 12:00:00 AM
5/7/2022	04/01/2022	4/1/2021 12:00:00 AM
5/16/2022	04/01/2022	4/1/2021 12:00:00 AM
6/15/2022	04/01/2022	4/1/2021 12:00:00 AM
6/26/2022	04/01/2022	4/1/2021 12:00:00 AM
7/9/2022	07/01/2022	7/1/2021 12:00:00 AM
7/22/2022	07/01/2022	7/1/2021 12:00:00 AM
7/23/2022	07/01/2022	7/1/2021 12:00:00 AM

date	start_of_quarter	start_of_quarter_timestamp
7/27/2022	07/01/2022	7/1/2021 12:00:00 AM
8/2/2022	07/01/2022	7/1/2021 12:00:00 AM
8/8/2022	07/01/2022	7/1/2021 12:00:00 AM
8/19/2022	07/01/2022	7/1/2021 12:00:00 AM
9/26/2022	07/01/2022	7/1/2021 12:00:00 AM
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

El campo `start_of_quarter` se crea en la instrucción load anterior utilizando la función `quarterstart()` e introduciendo el campo de fecha como argumento de la función. La función `uarterstart()` identifica inicialmente en qué trimestre cae el valor de la fecha. Luego devuelve una marca de tiempo con el primer milisegundo de ese trimestre.

Diagrama de la función `quarterstart()`, ejemplo sin argumentos adicionales



La transacción 8203 tuvo lugar el 8 de agosto. La función `quarterstart()` identifica que la transacción tuvo lugar en el tercer trimestre y devuelve el primer milisegundo de ese trimestre, que es el 1 de julio a las 12:00:00 a. m.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `previous_quarter_start`, que devuelve la marca de tiempo de inicio del trimestre anterior a la transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `previous_quarter_start`
- `previous_quarter_start_timestamp`

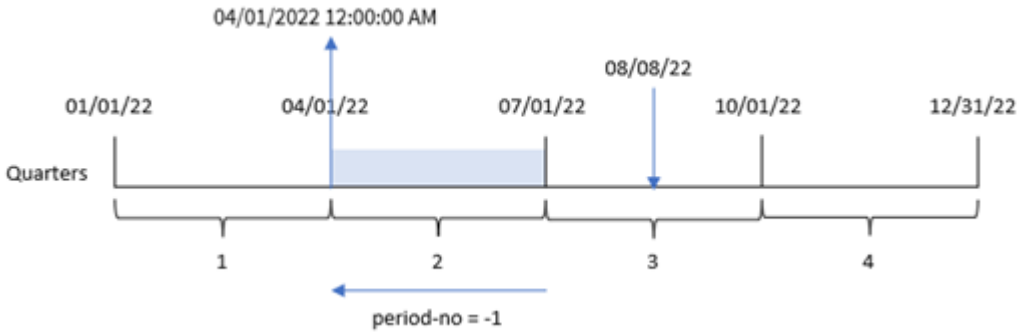
Tabla de resultados

date	previous_quarter_start	previous_quarter_start_timestamp
1/7/2022	10/01/2021	10/1/2021 12:00:00 AM
1/19/2022	10/01/2021	10/1/2021 12:00:00 AM

date	previous_quarter_start	previous_quarter_start_timestamp
2/5/2022	10/01/2021	10/1/2021 12:00:00 AM
2/28/2022	10/01/2021	10/1/2021 12:00:00 AM
3/16/2022	10/01/2021	10/1/2021 12:00:00 AM
4/1/2022	01/01/2022	1/1/2022 12:00:00 AM
5/7/2022	01/01/2022	1/1/2022 12:00:00 AM
5/16/2022	01/01/2022	1/1/2022 12:00:00 AM
6/15/2022	01/01/2022	1/1/2022 12:00:00 AM
6/26/2022	01/01/2022	1/1/2022 12:00:00 AM
7/9/2022	04/01/2022	4/1/2021 12:00:00 AM
7/22/2022	04/01/2022	4/1/2021 12:00:00 AM
7/23/2022	04/01/2022	4/1/2021 12:00:00 AM
7/27/2022	04/01/2022	4/1/2021 12:00:00 AM
8/2/2022	04/01/2022	4/1/2021 12:00:00 AM
8/8/2022	04/01/2022	4/1/2021 12:00:00 AM
8/19/2022	04/01/2022	4/1/2021 12:00:00 AM
9/26/2022	04/01/2022	4/1/2021 12:00:00 AM
10/14/2022	07/01/2022	7/1/2022 12:00:00 AM
10/29/2022	07/01/2022	7/1/2022 12:00:00 AM

En este caso, debido a que se usó un `period_no` de -1 como argumento de desplazamiento en la función `quarterstart()`, la función primero identifica el trimestre en el que se realizan las transacciones. Luego cambia a un trimestre antes e identifica el primer milisegundo de ese trimestre.

Diagrama de la función `quarterstart()`, ejemplo de period_no



La transacción 8203 tuvo lugar el 8 de agosto. La función `quarterstart()` identifica que el trimestre anterior a la transacción fue entre el 1 de abril y el 30 de junio. Luego devuelve el primer milisegundo de ese trimestre, el 1 de abril a las 12:00:00 a. m.

Ejemplo 3: first_month_of_year

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo. Sin embargo, en este ejemplo, necesitamos establecer el 1 de marzo como el comienzo del año fiscal.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

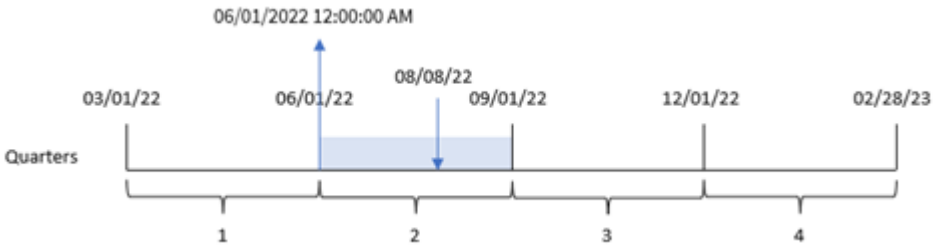
- date
- start_of_quarter
- start_of_quarter_timestamp

Tabla de resultados

date	start_of_quarter	start_of_quarter_timestamp
1/7/2022	12/01/2021	12/1/2021 12:00:00 AM
1/19/2022	12/01/2021	12/1/2021 12:00:00 AM
2/5/2022	12/01/2021	12/1/2021 12:00:00 AM
2/28/2022	12/01/2021	12/1/2021 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	03/01/2022	3/1/2022 12:00:00 AM
5/16/2022	03/01/2022	3/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM
7/23/2022	06/01/2022	6/1/2022 12:00:00 AM
7/27/2022	06/01/2022	6/1/2022 12:00:00 AM
8/2/2022	06/01/2022	6/1/2022 12:00:00 AM
8/8/2022	06/01/2022	6/1/2022 12:00:00 AM
8/19/2022	06/01/2022	6/1/2022 12:00:00 AM
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

En este caso, como el argumento `first_month_of_year` de 3 se utiliza en la función `quarterstart()`, el inicio del año se desplaza del 1 de enero al 1 de marzo.

Diagrama de la función quarterstart(), ejemplo de first_month_of_year



La transacción 8203 tuvo lugar el 8 de agosto. Debido a que el comienzo del año es el 1 de marzo, los trimestres del año ocurren entre marzo y mayo, junio y agosto, septiembre y noviembre, y diciembre y febrero. La función `quarterstart()` identifica que la transacción tuvo lugar en el trimestre entre el comienzo de junio y de agosto y devuelve el primer milisegundo de ese trimestre, que es el 1 de junio a las 12:00:00 a. m.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que devuelve una marca de tiempo del final del trimestre en que se realizaron las transacciones se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Agregue las siguientes medidas:

- `=quarterstart(date)`
- `=timestamp(quarterstart(date))`

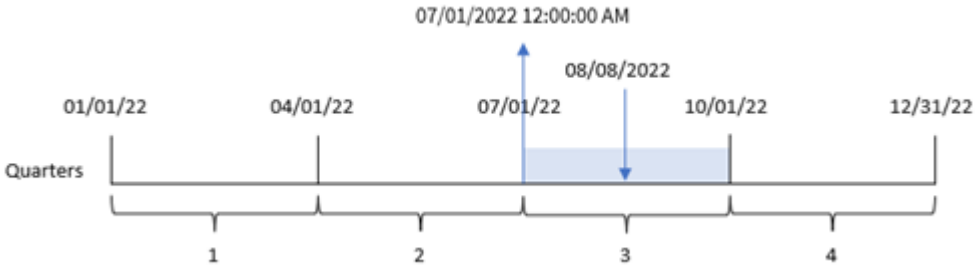
Tabla de resultados

date	=quarterstart(date)	=timestamp(quarterstart(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
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
9/26/2022	07/01/2022	7/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/16/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	04/01/2022	4/1/2022 12:00:00 AM
6/26/2022	04/01/2022	4/1/2022 12:00:00 AM
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	1/1/2022 12:00:00 AM
3/16/2022	01/01/2022	1/1/2022 12:00:00 AM

La medida `start_of_quarter` se crea en el objeto gráfico utilizando la función `quarterstart()` e introduciendo el campo `date` como argumento de la función.

La función `quarterstart()` identifica en qué trimestre cae el valor de la fecha y devuelve una marca de tiempo con el primer milisegundo de ese trimestre.

Diagrama de la función `quarterstart()`, ejemplo de objeto gráfico



La transacción 8203 tuvo lugar el 8 de agosto. La función `quarterstart()` identifica que la transacción tuvo lugar en el tercer trimestre y devuelve el primer milisegundo de ese trimestre. El valor que devuelve es el 1 de julio a las 12:00:00

Ejemplo 5: escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de saldos de préstamos, que se carga en una tabla llamada `Loans`.
- Datos que consisten en ID de préstamos, el saldo al comienzo del trimestre y la tasa de interés simple cobrada en cada préstamo por año.

Al usuario final le gustaría tener un objeto gráfico que muestre, por ID de préstamo, el interés actual que se ha acumulado en cada préstamo en el trimestre hasta la fecha.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:
   - `loan_id`
   - `start_balance`
2. A continuación, cree una medida para calcular el interés acumulado:
   `=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)`
3. Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

loan_id	start_balance	=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)
8188	$10000.00	$15.07
8189	$15000.00	$128.84
8190	$17500.00	$63.29
8191	$21000.00	$107.59
8192	$90000.00	$1139.18

La función `quarterstart()`, utilizando la fecha de hoy como único argumento, devuelve la fecha de inicio del año actual. Al restar ese resultado de la fecha actual, la expresión devuelve el número de días que han transcurrido en lo que va de trimestre.

Luego, este valor se multiplica por la tasa de interés y se divide por 365 para obtener la tasa de interés efectiva en que se ha incurrido durante este período. Luego, el resultado se multiplica por el saldo inicial del préstamo para devolver el interés que se ha acumulado en lo que va del trimestre.

second

Esta función devuelve un entero que representa el segundo en que la fracción de **expression** se interpreta como una hora de acuerdo con la interpretación numérica estándar.

Sintaxis:

```
second (expression)
```

Tipo de datos que devuelve: Entero

Cuándo se utiliza

La función `second()` es útil cuando se desea comparar agregaciones por segundo. Por ejemplo, podría usar la función si desea ver la distribución del recuento de actividad por segundo.

Estas dimensiones se pueden crear en el script de carga utilizando la función para crear un campo en una tabla de calendario maestro o se pueden utilizar directamente en un gráfico como una dimensión calculada.

Ejemplos de funciones

Ejemplo	Resultado
`second( '09:14:36' )`	devuelve 36
`second( '0.5555' )`	devuelve 55 (porque 0,5555 = 13:19:55)

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: variable

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene transacciones por marca de tiempo, que se carga en una tabla llamada `Transactions`.
- Se utiliza la variable predefinida del sistema `TimeStamp` (`M/D/YYYY h:mm:ss[.fff] TT`).
- La creación de un campo, `second`, para calcular cuándo se realizaron las compras.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `second`

Tabla de resultados

date	second
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
01/06/2022 11:29:38 AM	38
01/06/2022 10:58:34 PM	34

Los valores del campo `second` se crean utilizando la función `second()` e indicando la fecha como expresión en la sentencia load anterior.

Ejemplo 2: objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo. Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. Los valores de `second` se calculan por medio de una medida en un objeto gráfico.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:`date`.

Cree la siguiente medida:

```
=second(date)
```

Tabla de resultados

date	=second(date)
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

date	=second(date)
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
01/06/2022 11:29:38 AM	38
01/06/2022 10:58:34 PM	34

Los valores de second se crean usando la función `second()` e introduciendo la fecha como la expresión en una medida del objeto gráfico.

Ejemplo 3: escenario

Script de carga y expresiones de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de tiempo, que se genera para representar el tráfico a un sitio web de venta de entradas de un festival en particular. Estos datos de tiempo y una `id` correspondiente se cargan en una tabla llamada `Web_Traffic`.
- Se utiliza la variable del sistema `TimeStamp`: `M/D/YYYY h:mm:ss[.fff] TT`.

En este escenario, había 10000 entradas, que salieron a la venta a las 9:00 a.m. del 20 de mayo de 2021. Un minuto después, las entradas se agotaron.

Al usuario le gustaría tener un objeto gráfico que muestre, por segundo, el recuento de visitas al sitio web.

Script de carga

```
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;
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla.
2. Después cree una dimensión calculada usando la siguiente expresión:
   `=second(timestamp)`
3. Cree una medida de agregación para calcular el recuento total de entradas:
   `=count(id)`

La tabla de resultados se verá de aspecto similar a la siguiente tabla, pero con diferentes valores para la medida de agregación:

Tabla de resultados

second(timestamp)	=count(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

second(timestamp)	=count(id)
18	158
19	159
20	163
+ 39 filas más	

setdateyear

Esta función toma como datos de entrada una marca de tiempo **timestamp** y un año **year** y actualiza la marca de tiempo **timestamp** con el año **year** especificado en los datos de entrada.

Sintaxis:

```
setdateyear (timestamp, year)
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
timestamp	Es una indicación de fecha-hora estándar de Qlik Sense (a menudo solo una fecha).
year	Es un año de cuatro dígitos.

Ejemplos y resultados:
Estos ejemplos utilizan el formato de fecha **DD/MM/YYYY**. El formato de fecha se especifica en la sentencia **SET DateFormat** en la parte superior de su script de carga de datos. Cambie el formato en los ejemplos según se ajuste a sus necesidades.

Ejemplos de script

Ejemplo	Resultado
`setdateyear ('29/10/2005', 2013)`	Devuelve '29/10/2013
`setdateyear ('29/10/2005 04:26:14', 2013)`	Devuelve '29/10/2013 04:26:14' Para ver la parte de la hora en una visualización, debe definir el formato numérico en Fecha y escoger un valor de formato que muestre valores de hora.

Ejemplo:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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

];
```

La tabla resultante contiene las fechas originales y una columna en la que el año se debe fijar en 2013.

Tabla de resultados

testdates	**NewYear**
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

testdates	NewYear
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

Esta función toma como datos de entrada una marca de tiempo **timestamp**, un mes **month** y un año **year** y actualiza la marca de tiempo **timestamp** con el año **year** y el mes **month** especificados en los datos de entrada. .

Sintaxis:

```
SetDateYearMonth (timestamp, year, month)
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
timestamp	Es una indicación de fecha-hora estándar de Qlik Sense (a menudo solo una fecha).
year	Es un año de cuatro dígitos.
month	Es un mes de uno o dos dígitos.

Ejemplos y resultados:
Estos ejemplos utilizan el formato de fecha **DD/MM/YYYY**. El formato de fecha se especifica en la sentencia **SET DateFormat** en la parte superior de su script de carga de datos. Cambie el formato en los ejemplos según se ajuste a sus necesidades.

Ejemplos de script

Ejemplo	Resultado
`setdateyearmonth ('29/10/2005', 2013, 3)`	Devuelve '29/03/2013
`setdateyearmonth ('29/10/2005 04:26:14', 2013, 3)`	Devuelve '29/03/2013 04:26:14' Para ver la parte de la hora en una visualización, debe definir el formato numérico en Fecha y escoger un valor de formato que muestre valores de hora.

Ejemplo:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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

];
```

La tabla resultante contiene las fechas originales y una columna en la que el año se debe fijar en 2013.

Tabla de resultados

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

testdates	NewYearMonth
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

Esta función devuelve la zona horaria, tal como se define en el equipo informático donde está funcionando el motor de Qlik.

Sintaxis:

```
TimeZone( )
```

Tipo de datos que devuelve: dual

Ejemplo:

`timezone( )`
Si desea ver una zona horaria diferente en una medida en su aplicación, puede usar la función `localtime()` en una medida.

today

Esta función devuelve la fecha actual. La función devuelve valores en el formato de la variable del sistema `DateFormat`.

Sintaxis:

```
today([ timer_mode])
```

Tipo de datos que devuelve: dual

La función `today()` se puede utilizar en el script de carga o en los objetos del gráfico.

El valor predeterminado es 1 `timer_mode`.

Argumentos

Argumento	Descripción
timer_mode	Puede tener los siguientes valores: 0 (día de la última carga de datos finalizada) 1 (día de la llamada a la función) 2 (día en que se abrió la app) *Si utiliza la función en un script de carga, **timer_mode=0** dará como resultado el día de la última carga de datos finalizada, mientras que **timer_mode=1** nos dará el día de la carga de datos actual.*

Ejemplos de funciones

valor de timer_mode	Resultado si se usa en el script de carga	Resultado si se usa en el objeto gráfico
0	Devuelve una fecha, en el formato de la variable del sistema `DateFormat`, de la última recarga de datos correcta anterior a la última recarga de datos.	Devuelve una fecha, en el formato de la variable del sistema `DateFormat`, de la última recarga de datos.
1	Devuelve una fecha, en el formato de la variable del sistema `DateFormat`, de la última recarga de datos.	Devuelve una fecha, en el formato de la variable del sistema `DateFormat`, de la llamada a la función.
2	Devuelve una fecha, en el formato de la variable del sistema `DateFormat`, de cuándo comenzó la sesión del usuario en la aplicación. Esto no se actualizará a menos que el usuario recargue el script.	Devuelve la fecha, en el formato de la variable del sistema `DateFormat`, de cuándo comenzó la sesión del usuario en la aplicación. Esto se actualizará una vez que comience una nueva sesión o se vuelvan a cargar los datos en la aplicación.

Cuándo se utiliza

La función `today()` se utiliza habitualmente como un componente dentro de una expresión. Por ejemplo, se puede utilizar para calcular el interés que se ha acumulado en un mes hasta la fecha actual.

La tabla siguiente ofrece una explicación del resultado que devuelve la función `today()`, dados diferentes valores para el argumento de `timer_mode`:

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: generación de objetos utilizando script de carga

Script de carga y resultados

Vista general

El ejemplo siguiente crea tres variables usando la función `today()`. Cada variable utiliza una de las opciones `timer_mode` para demostrar su efecto.

Para que las variables cumplan su finalidad, recargue el script y después de 24 horas, vuelva a cargar el script por segunda vez. Esto dará como resultado que las variables de `today(0)` y `today(1)` muestren valores diferentes, cumpliendo así correctamente su propósito.

Script de carga

```
LET vPreviousDataLoad = today(0);
LET vCurrentDataLoad = today(1);
LET vApplicationOpened = today(2);
```

Resultados

Una vez cargados los datos por segunda vez, siga las instrucciones que se indican a continuación para crear tres cuadros de texto.

En primer lugar, cree un cuadro de texto para los datos que se cargaron anteriormente.

Haga lo siguiente:

1. Cree un cuadro de texto con el objeto de gráfico **Texto e imagen**.
2. Agregue al objeto la medida siguiente:
   `=vPreviousDataLoad`
3. En **Aspecto**, seleccione **Show titles** y agregue al objeto el título "Hora de recarga anterior".

A continuación, cree un cuadro de texto para los datos que se están cargando actualmente.

Haga lo siguiente:

1. Cree un cuadro de texto con el objeto de gráfico **Texto e imagen**.
2. Agregue al objeto la medida siguiente:

```
=vCurrentDataLoad
```

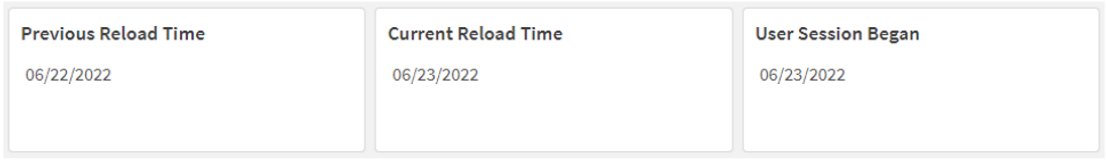
3. En **Aspecto**, seleccione **Show titles** y agregue al objeto el título "Hora de recarga actual".

Cree un cuadro de texto final para mostrar cuándo se inició la sesión del usuario en la aplicación.

Haga lo siguiente:

1. Cree un cuadro de texto con el objeto de gráfico **Texto e imagen**.
2. Agregue al objeto la medida siguiente:
   `=vApplicationOpened`
3. En **Aspecto**, seleccione **Show titles** y añada al objeto el título "Inicio de la sesión del usuario".

Diagrama de variables creadas utilizando la función `today()` en el script de carga



La imagen superior muestra valores a modo de ejemplo de cada una de las variables creadas. Por ejemplo, los valores podrían ser los siguientes:

- Fecha de recarga anterior: 22/06/2022
- Fecha de recarga actual: 23/06/2022
- Comienzo de la sesión de usuario: 23/06/2022

Ejemplo 2: generación de objetos sin script de carga

Script de carga y expresión de gráfico

Vista general

El ejemplo siguiente crea objetos gráficos usando la función `today()`. Cada objeto gráfico utiliza una de las opciones de `timer_mode` para demostrar su efecto.

No hay script de carga para este ejemplo.

Resultados

Una vez que los datos se hayan cargado por segunda vez, cree tres cuadros de texto.

Primero, cree un cuadro de texto para la última recarga de datos.

Haga lo siguiente:

1. Cree un cuadro de texto con el objeto gráfico **Texto e imagen**.
2. Agregue la siguiente medida.
   `=today(0)`
3. En **Aspecto**, seleccione **Mostrar títulos** y agregue al objeto el título "Última recarga de datos".

A continuación, cree un cuadro de texto para mostrar la hora actual.

Haga lo siguiente:

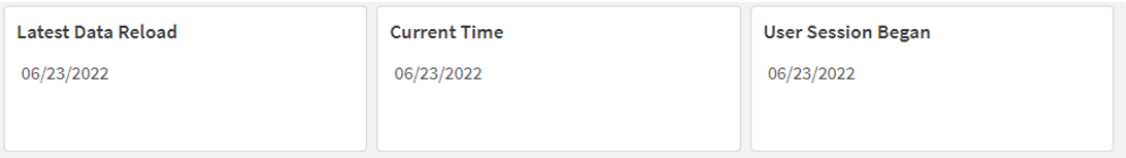
1. Cree un cuadro de texto con el objeto gráfico **Texto e imagen**.
2. Agregue la siguiente medida:
   `=today(1)`
3. En **Aspecto**, seleccione **Mostrar títulos** y agregue al objeto el título "Hora actual".

Cree un cuadro de texto final para mostrar cuándo se inició la sesión del usuario en la aplicación.

Haga lo siguiente:

1. Cree un cuadro de texto con el objeto gráfico **Texto e imagen**.
2. Agregue la siguiente medida:
   `=today(2)`
3. En **Aspecto**, seleccione **Mostrar títulos** y añada al objeto el título "Comienzo de la sesión de usuario".

Diagrama de los objetos creados utilizando la función `today()` sin script de carga



La imagen superior muestra valores de ejemplo para cada uno de los objetos creados. Por ejemplo, los valores podrían ser los siguientes:

- Última recarga de datos: 23/06/2022
- Fecha actual: 23/06/2022
- Comienzo de la sesión de usuario: 23/06/2022

El objeto gráfico "Última recarga de datos" utiliza un valor `timer_mode` de 0. Esto devuelve la marca de tiempo de la última vez que los datos se recargaron correctamente.

El objeto gráfico "Hora actual" utiliza un valor `timer_mode` de 1. Esto devuelve la hora actual según el reloj del sistema. Si se actualiza la hoja o el objeto, se actualizará este valor.

El objeto gráfico "Comienzo de la sesión de usuario" utiliza un valor `timer_mode` de 2. Esto devuelve la fecha y hora de cuándo se abrió la aplicación y comenzó la sesión del usuario.

Ejemplo 3: escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de saldos de préstamos, que se carga en una tabla llamada `Loans`.
- Datos de tabla con los campos de ID de préstamo, saldo al inicio del mes y la tasa de interés simple cobrada en cada préstamo por año.

Al usuario final le gustaría tener un objeto gráfico que muestre, por ID de préstamo, el interés actual que se ha acumulado en cada préstamo en el mes hasta la fecha. Aunque la aplicación solo se recarga una vez por semana, al usuario le gustaría que los resultados se actualicen cada vez que se actualice el objeto o la aplicación.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja.Cree una nueva tabla.
2. Agregue los siguientes campos como dimensiones.
    - `loan_id`
    - `start_balance`
3. Ahora cree una medida para calcular el interés acumulado:
   `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
4. Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

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

La función `monthstart()`, utilizando la función `today()` para devolver la fecha de hoy como su único argumento, devuelve la fecha de inicio del mes actual. Restando ese resultado de la fecha actual, de nuevo utilizando la función `today()`, la expresión devuelve el número de días que han transcurrido en lo que llevamos de mes.

Luego, este valor se multiplica por la tasa de interés y se divide por 365 para obtener la tasa de interés efectiva en que se ha incurrido durante este período. Luego, el resultado se multiplica por el saldo inicial del préstamo para devolver los intereses que se han acumulado en lo que va del mes.

Debido a que el valor de 1 se usa como argumento de `timer_mode` en las funciones `today()` dentro de la expresión, cada vez que se actualiza el objeto gráfico (al abrir la aplicación, actualizar la página, moverse entre hojas, etc.), la fecha que devuelve será conforme a la fecha actual, y los resultados se actualizarán de manera acorde.

UTC

Devuelve la hora Coordinated Universal Time actual.

Sintaxis:

```
UTC( )
```

Tipo de datos que devuelve: dual

Ejemplo:

```
utc( )
```

week

Esta función devuelve un entero que representa el número de semana correspondiente a la fecha introducida.

Sintaxis:

```
week(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

Tipo de datos que devuelve: Entero

Argumentos

Argumento	Descripción
timestamp	La fecha o marca de tiempo para evaluar.
first_week_day	Especifica el día en el que se inicia la semana. Si se omite, se utiliza el valor de la variable **FirstWeekDay**. Los valores posibles de **first_week_day** son 0 para el lunes, 1 para el martes, 2 para el miércoles, 3 para el jueves, 4 para el viernes, 5 para el sábado y 6 para el domingo. Para más información sobre la variable del sistema, vea *FirstWeekDay (page 229)*.
broken_weeks	Si no especifica **broken_weeks**, el valor de la variable **BrokenWeeks** se usará para definir si las semanas se rompen o no.
reference_day	Si no especifica **reference_day**, el valor de la variable **ReferenceDay** se usará para definir qué día de enero debe configurarse como día de referencia para definir la semana 1. De forma predeterminada, las funciones Qlik Sense usan 4 como día de referencia. Esto significa que la semana 1 debe incluir el 4 de enero o, dicho de otro modo, que la semana 1 siempre debe incluir 4 días de enero como mínimo.

La función `week()` determina en qué semana cae la fecha y devuelve el número de semana.

En Qlik Sense, la configuración regional se obtiene cuando se crea la app y la configuración correspondiente se almacena en el script como variables de entorno. Estos se utilizan para determinar el número de semana.

Esto significa que la mayoría de los desarrolladores de aplicaciones europeos obtienen las siguientes variables de entorno, correspondientes a la definición ISO 8601:

```
Set FirstWeekDay =0;    // Monday as first week day
Set BrokenWeeks  =0;    // Use unbroken weeks
Set ReferenceDay =4;    // Jan 4th is always in week 1
```

Un desarrollador de aplicaciones norteamericano a menudo obtiene las siguientes variables de entorno:

```
Set FirstWeekDay =6;    // Sunday as first week day
Set BrokenWeeks  =1;    // Use broken weeks
Set ReferenceDay =1;    // Jan 1st is always in week 1
```

El primer día de la semana viene determinado por la variable del sistema `FirstWeekDay`. También puede cambiar el primer día de la semana usando el argumento `first_week_day` en la función `week()`.

Si su aplicación utiliza semanas interrumpidas, el número de semanas comienza el 1 de enero y finaliza el día anterior a la variable del sistema `FirstWeekDay`, independientemente de cuántos días hayan transcurrido.

Si su aplicación utiliza semanas ininterrumpidas, la semana 1 puede comenzar en el año anterior o en los primeros días de enero. Esto depende de cómo use las variables de entorno `FirstWeekDay` y `ReferenceDay`.

Cuándo se utiliza

La función `The week()` es útil cuando se desea comparar agregaciones por semanas. Por ejemplo, si desea ver el total de ventas de productos por semana. La función `week()` se elige por encima de `weekname()` cuando el usuario desea que el cálculo no utilice necesariamente las variables del sistema, `BrokenWeeks`, `FirstWeekDay` o `ReferenceDay` de la aplicación.

Por ejemplo, si desea ver el total de ventas de productos por semana.

Si la aplicación utiliza semanas ininterrumpidas, la semana 1 puede contener fechas de diciembre del año anterior o excluir fechas de enero del año en curso. Si la aplicación utiliza semanas interrumpidas, la semana 1 puede contener menos de siete días.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Los siguientes ejemplos suponen

```
Set DateFormat= 'MM/DD/YYYY';
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

Ejemplos de funciones

Ejemplo	**Resultado**
`week('12/28/2021')`	Devuelve 52.
`week(44614)`	Devuelve 8, ya que este es el número de serie del 22/02/2022.
`week('01/03/2021')`	Devuelve 53.
`week('01/03/2021',6)`	Devuelve 1.

Ejemplo 1: variables predeterminadas del sistema

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de la última semana de 2021 y las primeras dos semanas de 2022 y que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat` (MM/DD/AAAA).
- La creación de un campo, `week_number`, que devuelve el año y el número de semana en que se realizaron las transacciones.
- La creación de un campo denominado `week_day`, que muestra el valor del día de la semana de cada fecha de transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`
- `week_day`
- `week_number`

Tabla de resultados

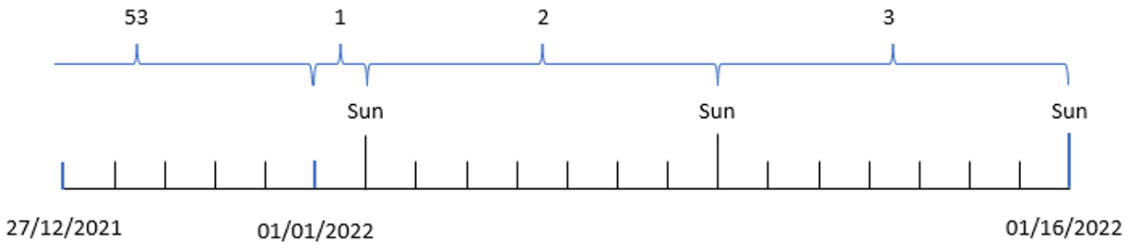
id	date	week_day	week_number
8183	12/27/2021	Lun	53
8184	12/28/2021	Mar	53
8185	12/29/2021	Mié	53
8186	12/30/2021	Jue	53
8187	12/31/2021	Vie	53
8188	01/01/2022	Sáb	1
8189	01/02/2022	Dom	2
8190	01/03/2022	Lun	2
8191	01/04/2022	Mar	2
8192	01/05/2022	Mié	2
8193	01/06/2022	Jue	2
8194	01/07/2022	Vie	2
8195	01/08/2022	Sáb	2
8196	01/09/2022	Dom	3
8197	01/10/2022	Lun	3
8198	01/11/2022	Mar	3
8199	01/12/2022	Mié	3
8200	01/13/2022	Jue	3
8201	01/14/2022	Vie	3

El campo `week_number` se crea en la instrucción load anterior utilizando la función `week()` e introduciendo el campo `date` como argumento de la función.

No se indica ningún otro parámetro en la función y, por lo tanto, las siguientes variables predeterminadas que afectan a la función `week()` siguen vigentes:

- `BrokenWeeks`: El recuento de semanas comienza el 1 de enero
- `FirstWeekDay`: El primer día de la semana es domingo

Diagrama de la función week(), utilizando variables predeterminadas del sistema



Debido a que la aplicación utiliza la variable del sistema predeterminada `BrokenWeeks`, la semana 1 comienza el 1 de enero, un sábado.

Debido a la variable del sistema predeterminada `FirstWeekDay`, las semanas comienzan en domingo. El primer domingo después del 1 de enero ocurre el 2 de enero, que es cuando comienza la semana 2.

Ejemplo 2: first_week_day

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- La creación de un campo, `week_number`, que devuelve el año y el número de semana en que se realizaron las transacciones.
- La creación de un campo denominado `week_day`, que muestra el valor del día de la semana de cada fecha de transacción.

En este ejemplo, nos gustaría configurar el martes como inicio de la semana laboral.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`
- `week_day`
- `week_number`

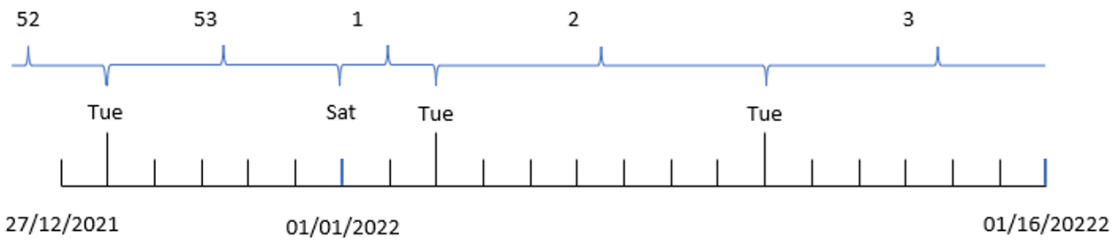
Tabla de resultados

id	date	week_day	week_number
8183	12/27/2021	Lun	52
8184	12/28/2021	Mar	53
8185	12/29/2021	Mié	53
8186	12/30/2021	Jue	53
8187	12/31/2021	Vie	53
8188	01/01/2022	Sáb	1

id	date	week_day	week_number
8189	01/02/2022	Dom	1
8190	01/03/2022	Lun	1
8191	01/04/2022	Mar	2
8192	01/05/2022	Mié	2
8193	01/06/2022	Jue	2
8194	01/07/2022	Vie	2
8195	01/08/2022	Sáb	2
8196	01/09/2022	Dom	2
8197	01/10/2022	Lun	2
8198	01/11/2022	Mar	3
8199	01/12/2022	Mié	3
8200	01/13/2022	Jue	3
8201	01/14/2022	Vie	3

La aplicación sigue utilizando semanas interrumpidas. Sin embargo, el argumento de `first_week_day` se ha definido en 1 en la función `week()`. Esto establece el primer día de la semana en martes.

Diagrama de la función week(), ejemplo de first_week_day



La aplicación utiliza la variable de sistema predeterminada `BrokenWeeks`, por lo que la semana 1 comienza el 1 de enero, un sábado.

El argumento de `first_week_day` de la función `week()` establece el primer día de la semana en martes. Por lo tanto, la semana 53 comienza el 28 de diciembre de 2021.

Sin embargo, debido a que la función todavía utiliza semanas interrumpidas, la semana 1 solo durará dos días, debido a que el primer martes después del 1 de enero se produce el 3 de enero.

Ejemplo 3: unbroken_weeks

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo.

En este ejemplo, usamos semanas no interrumpidas.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`
- `week_day`
- `week_number`

Diagrama de la función week(), ejemplo de objeto gráfico

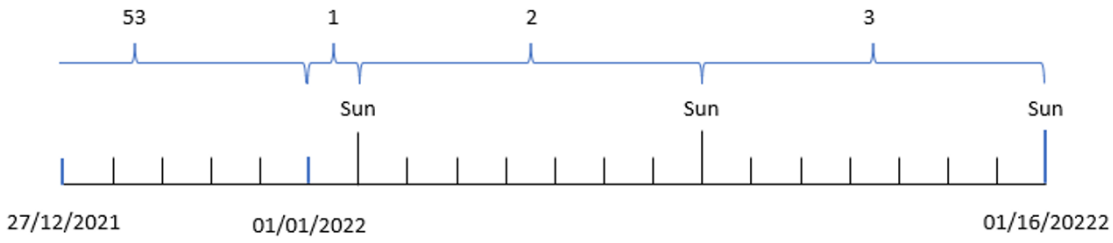


Tabla de resultados

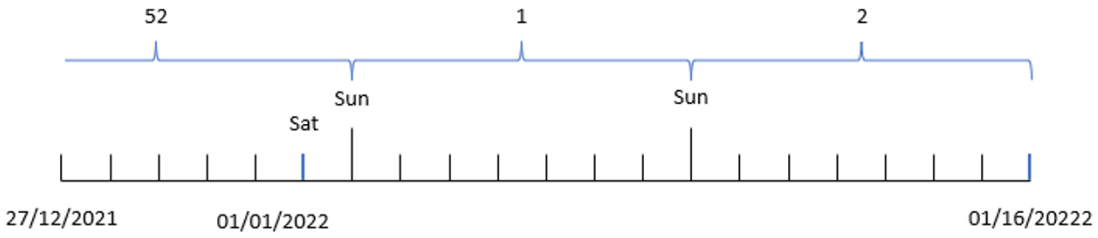
id	date	week_day	week_number
8183	12/27/2021	Lun	52
8184	12/28/2021	Mar	52
8185	12/29/2021	Mié	52
8186	12/30/2021	Jue	52
8187	12/31/2021	Vie	52
8188	01/01/2022	Sáb	52
8189	01/02/2022	Dom	1
8190	01/03/2022	Lun	1
8191	01/04/2022	Mar	1
8192	01/05/2022	Mié	1
8193	01/06/2022	Jue	1
8194	01/07/2022	Vie	1
8195	01/08/2022	Sáb	1
8196	01/09/2022	Dom	2
8197	01/10/2022	Lun	2
8198	01/11/2022	Mar	2

id	date	week_day	week_number
8199	01/12/2022	Mié	2
8200	01/13/2022	Jue	2
8201	01/14/2022	Vie	2

El parámetro de first_week_date está definido en 1, lo que hace que el martes sea el primer día de la semana. El parámetro de broken_weeks está establecido en 0, lo que obliga a la función a utilizar semanas ininterrumpidas. Por último, el tercer parámetro establece el día de referencia "reference_day" en 2.

El parámetro de `first_week_date` se establece en 6, lo que hace que el domingo sea el primer día de la semana. El parámetro de `broken_weeks` se establece en 0, lo que obliga a la función a utilizar semanas ininterrumpidas.

Diagrama de la función week(), ejemplo que utiliza semanas continuas



Al usar semanas continuas, ininterrumpidas, la semana 1 no necesariamente comienza el 1 de enero; en cambio, se requiere tener un mínimo de cuatro días. Por lo tanto, en el conjunto de datos, la semana 52 concluye el sábado 1 de enero de 2022. Luego, la semana 1 comienza en la variable del sistema `FirstWeekDay`, que es el domingo 2 de enero. Esta semana concluirá el siguiente sábado, 8 de enero.

Ejemplo 4: reference_day

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el tercer ejemplo.
- La creación de un campo, `week_number`, que devuelve el año y el número de semana en que se realizaron las transacciones.
- La creación de un campo denominado `week_day`, que muestra el valor del día de la semana de cada fecha de transacción.

Además, se deben cumplir las siguientes condiciones:

- La semana laboral comienza un martes.
- La empresa utiliza semanas no interrumpidas.
- El valor de reference_day es 2. En otras palabras, el número mínimo de días en enero en la semana 1 será de 2.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- id
- date

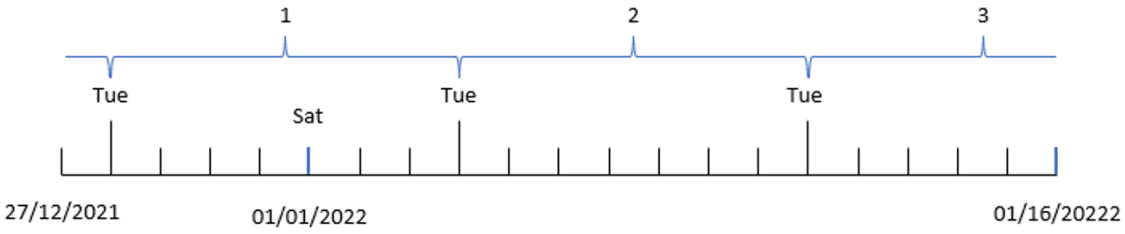
- week_day
- week_number

Tabla de resultados

id	date	week_day	week_number
8183	12/27/2021	Lun	52
8184	12/28/2021	Mar	1
8185	12/29/2021	Mié	1
8186	12/30/2021	Jue	1
8187	12/31/2021	Vie	1
8188	01/01/2022	Sáb	1
8189	01/02/2022	Dom	1
8190	01/03/2022	Lun	1
8191	01/04/2022	Mar	2
8192	01/05/2022	Mié	2
8193	01/06/2022	Jue	2
8194	01/07/2022	Vie	2
8195	01/08/2022	Sáb	2
8196	01/09/2022	Dom	2
8197	01/10/2022	Lun	2
8198	01/11/2022	Mar	3
8199	01/12/2022	Mié	3
8200	01/13/2022	Jue	3
8201	01/14/2022	Vie	3

El parámetro de `first_week_date` está establecido en 1, lo que hace que el martes sea el primer día de la semana. El parámetro de `broken_weeks` está establecido en 0, lo que obliga a la función a utilizar semanas no interrumpidas. Por último, el tercer parámetro establece el parámetro de `reference_day` en 2.

Diagrama de la función week(), ejemplo de reference_day



Con la función utilizando semanas no interrumpidas y un valor de `reference_day` de 2 como parámetro, la semana 1 solo necesita incluir dos días en enero. Debido a que el primer día de la semana es martes, la semana 1 comienza el 28 de diciembre de 2021 y concluye el lunes 3 de enero de 2022.

Ejemplo 5: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que devuelve el número de la semana se crea como una medida en un objeto gráfico.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja.Cree una nueva tabla.
2. Agregue los siguientes campos como dimensiones:
   - `id`
   - `date`
3. A continuación, cree la siguiente medida:
   `=week (date)`
4. Cree una medida , `week_day`, para mostrar el valor del día de la semana de cada fecha de transacción:
   `=weekday(date)`

Tabla de resultados

id	date	=week(date)	=weekday(date)
8183	12/27/2021	53	Lun
8184	12/28/2021	53	Mar
8185	12/29/2021	53	Mié
8186	12/30/2021	53	Jue
8187	12/31/2021	53	Vie
8188	01/01/2022	1	Sáb
8189	01/02/2022	2	Dom
8190	01/03/2022	2	Lun
8191	01/04/2022	2	Mar
8192	01/05/2022	2	Mié
8193	01/06/2022	2	Jue
8194	01/07/2022	2	Vie
8195	01/08/2022	2	Sáb
8196	01/09/2022	3	Dom
8197	01/10/2022	3	Lun
8198	01/11/2022	3	Mar
8199	01/12/2022	3	Mié

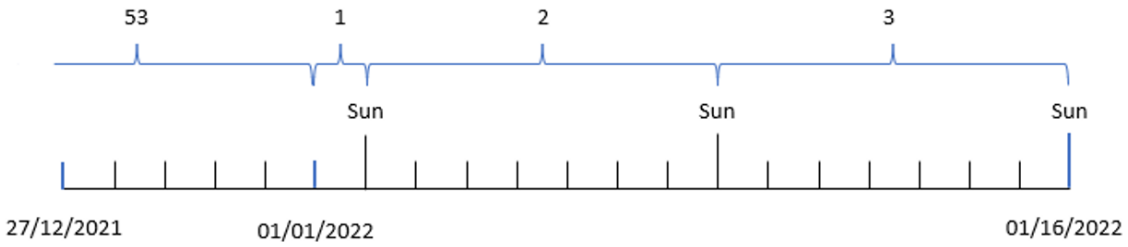
id	date	=week(date)	=weekday(date)
8200	01/13/2022	3	Jue
8201	01/14/2022	3	Vie

El campo `week_number` se crea en la instrucción load anterior utilizando la función `week()` e introduciendo el campo `date` como argumento de la función.

No se indica ningún otro parámetro en la función y, por lo tanto, las siguientes variables predeterminadas que afectan a la función `week()` siguen vigentes:

- `BrokenWeeks`: El recuento de semanas comienza el 1 de enero
- `FirstWeekDay`: El primer día de la semana es domingo

Diagrama de la función week(), ejemplo de objeto gráfico



Debido a que la aplicación utiliza la variable del sistema predeterminada `BrokenWeeks`, la semana 1 comienza el 1 de enero, un sábado.

Debido a la variable del sistema predeterminada `FirstWeekDay`, las semanas comienzan en domingo. El primer domingo después del 1 de enero ocurre el 2 de enero, que es cuando comienza la semana 2.

Ejemplo 6: Escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se carga un conjunto de datos que contiene una serie de transacciones de la última semana de 2019 y las primeras dos semanas de 2020 en una tabla llamada `Transactions`.
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat` (MM/DD/AAAA).

La aplicación utiliza principalmente semanas interrumpidas en su tablero. Sin embargo, al usuario final le gustaría tener un objeto gráfico que presente el total de ventas por semana utilizando semanas no interrumpidas. El día de referencia debe ser el 2 de enero y las semanas comienzan en martes. Esto podría lograrse incluso cuando esta dimensión no esté disponible en el modelo de datos, utilizando la función `week()` como una dimensión calculada en el gráfico.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja.Cree una nueva tabla.
2. Cree la siguiente dimensión calculada:
   `=week(date)`
3. A continuación, cree la siguiente medida de agregación:
   `=sum(amount)`
4. Establezca el **Formato numérico** de la medida en **Moneda**.

5. Seleccione el menú **Ordenar** y, para la dimensión calculada, elimine la ordenación personalizada.
6. Desmarque las opciones **Ordenar numéricamente** y **Ordenar alfabéticamente**.

Tabla de resultados

=week(date)	sum(amount)
52	$125.69
53	$146.42
1	$200.09
2	$347.57
3	$122.01

weekday

Esta función devuelve un valor dual con lo siguiente:

- Un nombre de día tal como se define en la variable de entorno **DayNames**.
- Un número entero entre 0 y 6 correspondiente al día nominal de la semana (0-6).

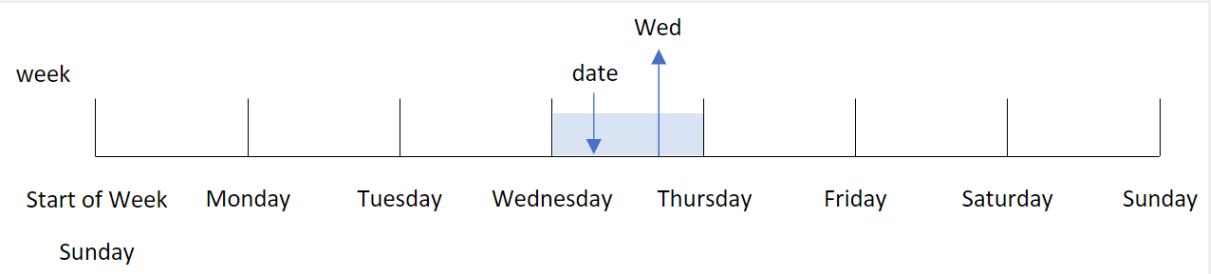
Sintaxis:

```
weekday(date [,first_week_day=0])
```

Tipo de datos que devuelve: dual

La función `weekday()` determina en qué día de la semana ocurre una fecha. Luego devuelve un valor de cadena de texto que representa ese día.

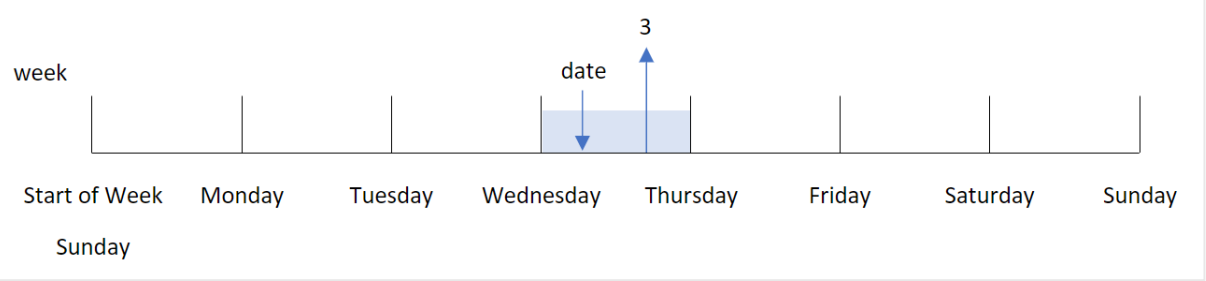
Diagrama de la función weekday() que devuelve el nombre del día en que cae una fecha



El resultado devuelve el valor numérico correspondiente a ese día de la semana (0-6), según el día de inicio de la semana. Por ejemplo, si el primer día de la semana se establece en domingo, un miércoles devolverá un valor numérico de 3. Este día de inicio lo determina la variable del sistema `FirstWeekDay` o el parámetro de la función `first_week_day`.

Puede utilizar este valor numérico como parte de una expresión aritmética. Por ejemplo, multiplíquelo por 1 para devolver el propio valor.

Diagrama de la función weekday() con el valor numérico del día, que se muestra en lugar del nombre del día



Cuándo se utiliza

La función weekday() es útil cuando desea comparar agregaciones por día de la semana. Por ejemplo, si desea comparar las ventas promedio de varios productos por día de la semana.

Estas dimensiones se pueden crear en el script de carga utilizando la función para crear un campo en una tabla de **calendario maestro** o se pueden utilizar directamente en un gráfico como una medida calculada.

Temas relacionados

Temas	Interacción
FirstWeekDay (page 229)	Define el día de inicio de cada semana.

Argumentos

Argumento	Descripción
date	La fecha o marca de tiempo para evaluar.
first_week_day	Especifica el día en el que se inicia la semana. Si se omite, se utiliza el valor de la variable **FirstWeekDay**. *FirstWeekDay (page 229)*

Puede utilizar los siguientes valores para establecer el día en que comienza la semana en el argumento first_week_day:

first_week_day values

Día	Valor
Lunes	0
Martes	1
Miércoles	2
Jueves	3
Viernes	4

Día	Valor
Sábado	5
Domingo	6

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

A menos que se indique lo contrario, `FirstWeekDay` se ha definido como `0` en estos ejemplos.

Ejemplos de funciones

Ejemplo	Resultado
`weekday('10/12/1971')`	Devuelve "Mar" y 1.
`weekday('10/12/1971' , 6)`	Devuelve "Mar" y 2. En este ejemplo, el domingo (6) es el primer día de la semana.

`SET FirstWeekDay=6;`
`...`
 	
`weekday('10/12/1971')`
	Devuelve "Mar" y 2.

Ejemplo 1: cadena de texto de día de la semana

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "Transactions".
- La variable del sistema FirstWeekDay configurada en 6 (domingo).
- La variable DayNames que está configurada para usar los nombres de día predeterminados.
- Una instrucción load anterior que contiene la función weekday(), que se establece como el campo "week_day" y devuelve el día de la semana en que se realizaron las transacciones.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- id
- date
- week_day

Tabla de resultados

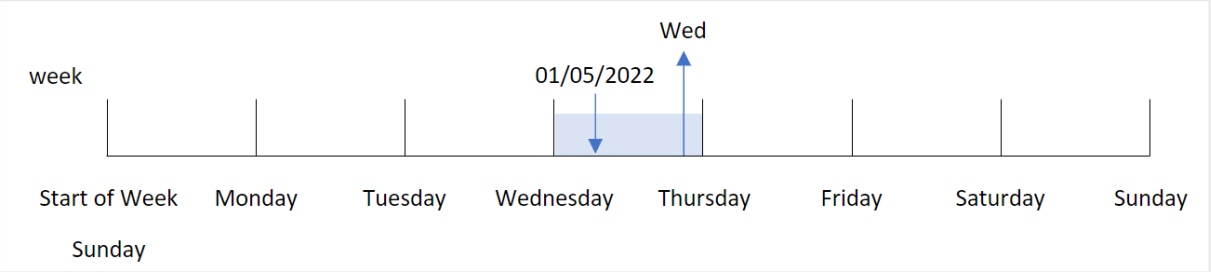
id	date	week_day
8188	01/01/2022	Sáb
8189	01/02/2022	Dom
8190	01/03/2022	Lun
8191	01/04/2022	Mar
8192	01/05/2022	Mié

id	date	week_day
8193	01/06/2022	Jue
8194	01/07/2022	Vie

El campo week_day se crea en la instrucción load anterior utilizando la función weekday() e introduciendo el campo de fecha como argumento de la función.

La función weekday() devuelve un valor de cadena de texto con el día de la semana; es decir, devuelve el nombre del día de la semana establecido en la variable del sistema DayNames.

Diagrama de la función weekday() que devuelve el miércoles como día de la semana de la transacción 8192



La transacción 8192 tuvo lugar el 5 de enero. La variable del sistema FirstWeekDay establece el primer día de la semana como domingo. La transacción de la función weekday() tuvo lugar un miércoles y devuelve este valor, en la forma abreviada de la variable del sistema DayNames, en el campo week_day.

Los valores del campo "week_day" están alineados a la derecha en la columna porque hay un número dual y un resultado de texto para el campo (miércoles, 3). Para convertir el valor del campo en su equivalente numérico, el campo se puede incluir en la función num(). Por ejemplo, en la transacción 8192, el valor del miércoles se convertiría en el número 3.

Ejemplo 2: first_week_day

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "Transactions".
- La variable del sistema FirstWeekDay configurada en 6 (domingo).
- La variable DayNames que está configurada para usar los nombres de día predeterminados.
- Una instrucción load anterior que contiene la función weekday(), que se establece como el campo "week_day" y devuelve el día de la semana en que se realizaron las transacciones.

Script de carga

```
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
];
```

Resultados

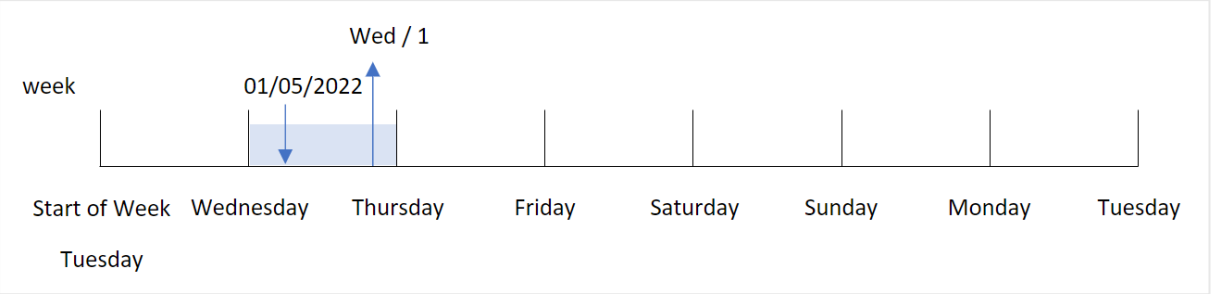
Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- id
- date
- week_day

Tabla de resultados

id	date	week_day
8188	01/01/2022	Sáb
8189	01/02/2022	Dom
8190	01/03/2022	Lun
8191	01/04/2022	Mar
8192	01/05/2022	Mié
8193	01/06/2022	Jue
8194	01/07/2022	Vie

Diagrama de la función weekday() que muestra que el miércoles tiene el valor numérico dual de 1



Como el argumento de `first_week_day` está establecido en `1` en la función `weekday()`, el primer día de la semana es martes. Por lo tanto, todas las transacciones que se realicen un martes tendrán un valor numérico dual de `0`.

La transacción 8192 tuvo lugar el 5 de enero. La función `weekday()` identifica que es miércoles, por lo que la expresión devolverá el valor numérico dual de `1`.

Ejemplo 3: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- La variable del sistema `FirstWeekDay` configurada en `6` (domingo).
- La variable `DayNames` que está configurada para usar los nombres de día predeterminados.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que identifica el valor del día de la semana se crea como una medida en un gráfico en la app.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`

Para calcular el valor del día de la semana, cree la siguiente medida:

- `=weekday(date)`

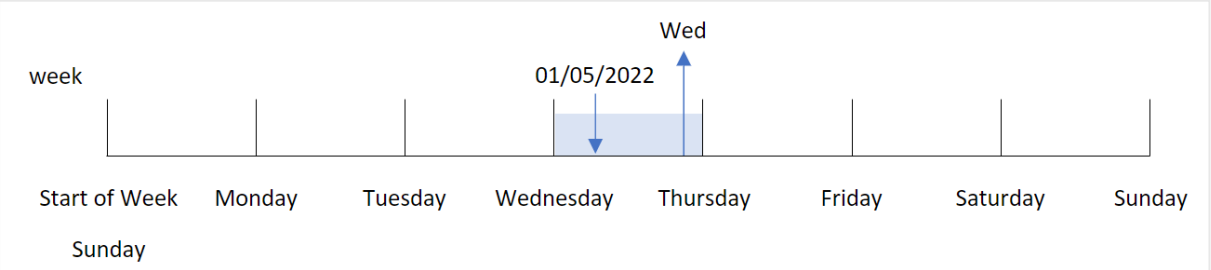
Tabla de resultados

id	date	=weekday(date)
8188	01/01/2022	Sáb
8189	01/02/2022	Dom
8190	01/03/2022	Lun
8191	01/04/2022	Mar
8192	01/05/2022	Mié
8193	01/06/2022	Jue
8194	01/07/2022	Vie

El campo "`=weekday(date)`" se crea en el gráfico utilizando la función `weekday()` e introduciendo el campo de fecha como argumento de la función.

La función `weekday()` devuelve un valor de cadena de texto con el día de la semana; es decir, devuelve el nombre del día de la semana establecido en la variable del sistema `DayNames`.

Diagrama de la función weekday() que devuelve el miércoles como día de la semana de la transacción 8192



La transacción 8192 tuvo lugar el 5 de enero. La variable del sistema `FirstWeekDay` establece el primer día de la semana como domingo. La transacción de la función `weekday()` tuvo lugar un miércoles y devuelve este valor, en la forma abreviada de la variable del sistema `DayNames`, en el campo `=weekday(date)`.

Ejemplo 4: escenario

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- La variable del sistema `FirstWeekDay` configurada en 6 (domingo).
- La variable `DayNames` que está configurada para usar los nombres de día predeterminados.

Al usuario final le gustaría tener un gráfico que presente la media de ventas por día de la semana de las transacciones.

Script de carga

```
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);
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `=weekday(date)`
- `=avg(amount)`

Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

weekday(date)	Avg(amount)
Dom	$536.96
Lun	$500.80
Mar	$515.63

weekday(date)	Avg(amount)
Mié	$509.21
Jue	$482.70
Vie	$441.33
Sáb	$505.22

weekend

Esta función devuelve un valor correspondiente a una marca de tiempo del último milisegundo del último día de la semana natural que contiene a **date**. El formato de salida predefinido será el **DateFormat** definido en el script.

Sintaxis:

```
WeekEnd(timestamp [, period_no [, first_week_day ]])
```

Tipo de datos que devuelve: dual

La función `weekend()` determina en qué semana cae la fecha. Luego devuelve una marca de tiempo, en formato de fecha, del último milisegundo de esa semana. El primer día de la semana viene determinado por la variable de entorno `FirstWeekDay`. No obstante, esto puede ser reemplazado por el argumento de `first_week_day` en la función `weekend()`.

Argumentos

Argumento	Descripción
timestamp	La fecha o marca de tiempo para evaluar.
period_no	**shift** es un entero, donde el valor 0 indica la semana que contiene a **date**. Los valores negativos en el desplazamiento indican semanas precedentes y los valores positivos indican semanas subsiguientes.
first_week_day	Especifica el día en el que se inicia la semana. Si se omite, se utiliza el valor de la variable **FirstWeekDay**. Los valores posibles para **first_week_day** son 0 para el lunes, 1 para el martes, 2 para el miércoles, 3 para el jueves, 4 para el viernes, 5 para el sábado y 6 para el domingo. Para más información sobre la variable del sistema, vea *FirstWeekDay (page 229)*

Cuándo se utiliza

La función `weekend()` se suele utilizar como parte de una expresión cuando el usuario desea que el cálculo utilice los días restantes de la semana para la fecha especificada. Por ejemplo, podría usarse si un usuario quisiera calcular el interés total aún no devengado durante la semana.

Los siguientes ejemplos suponen:

```
SET FirstWeekDay=0;
```

Ejemplo	Resultado
weekend('01/10/2013')	Devuelve 01/12/2013 23:59:59.
weekend('01/10/2013', -1)	Devuelve 01/05/2013 23:59:59..
weekend('01/10/2013', 0, 1)	Devuelve 01/14/2013 23:59:59.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos:

Si desea configuraciones ISO para semanas y números de semana, asegúrese de tener lo siguiente en el script:

```
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
```

Si desea la configuración de EE. UU., asegúrese de tener lo siguiente en el script:

```
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
```

Los ejemplos anteriores dan como resultado lo siguiente, de la función `weekend()`:

Ejemplo de la función Weekend

Fecha	Fin de semana ISO	Fin de semana EE. UU.
Sab 2020 Dic 26	2020-12-27	12/26/2020
Dom 2020 Dic 27	2020-12-27	1/2/2021
Lun 2020 Dic 28	2021-01-03	1/2/2021
Mar 2020 Dic 29	2021-01-03	1/2/2021
Mie 2020 Dic 30	2021-01-03	1/2/2021

Fecha	Fin de semana ISO	Fin de semana EE. UU.
Jue 2020 Dic 31	2021-01-03	1/2/2021
Vie 2021 Ene 1	2021-01-03	1/2/2021
Sab 2021 Ene 2	2021-01-03	1/2/2021
Dom 2021 Ene 3	2021-01-03	1/9/2021
Lun 2021 Ene 4	2021-01-10	1/9/2021
Mar 2021 Ene 5	2021-01-10	1/9/2021

Los fines de semana son los domingos en la columna ISO y los sábados en la columna de EE. UU.

Ejemplo 1: ejemplo básico

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo, `end_of_week`, que devuelve una marca de tiempo del final de la semana en que la transacción tuvo lugar.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

Tabla de resultados

date	end_of_week	end_of_week_timestamp
1/7/2022	01/08/2022	1/8/2022 11:59:59 PM
1/19/2022	01/22/2022	1/22/2022 11:59:59 PM
2/5/2022	02/05/2022	2/5/2022 11:59:59 PM
2/28/2022	03/05/2022	3/5/2022 11:59:59 PM
3/16/2022	03/19/2022	3/19/2022 11:59:59 PM
4/1/2022	04/02/2022	4/2/2022 11:59:59 PM
5/7/2022	05/07/2022	5/7/2022 11:59:59 PM
5/16/2022	05/21/2022	5/21/2022 11:59:59 PM
6/15/2022	06/18/2022	6/18/2022 11:59:59 PM
6/26/2022	07/02/2022	7/2/2022 11:59:59 PM
7/9/2022	07/09/2022	7/9/2022 11:59:59 PM
7/22/2022	07/23/2022	7/23/2022 11:59:59 PM
7/23/2022	07/23/2022	7/23/2022 11:59:59 PM
7/27/2022	07/30/2022	7/30/2022 11:59:59 PM
8/2/2022	08/06/2022	8/6/2022 11:59:59 PM

date	end_of_week	end_of_week_timestamp
8/8/2022	08/13/2022	8/13/2022 11:59:59 PM
8/19/2022	08/20/2022	8/20/2022 11:59:59 PM
9/26/2022	10/01/2022	10/1/2022 11:59:59 PM
10/14/2022	10/15/2022	10/15/2022 11:59:59 PM
10/29/2022	10/29/2022	10/29/2022 11:59:59 PM

El campo `end_of_week` se crea en la instrucción load anterior utilizando la función `weekend()` e introduciendo el campo de fecha como argumento de la función.

La función `weekend()` identifica en qué semana cae el valor de la fecha y devuelve una marca de tiempo del último milisegundo de esa semana.

Diagrama de la función `weekend()`, ejemplo básico

La transacción 8191 tuvo lugar el 5 de febrero. La variable del sistema `FirstWeekDay` establece el primer día de la semana en domingo. La función `weekend()` identifica que el primer sábado después del 5 de febrero, y por lo tanto el final de la semana, fue el 5 de febrero. Por lo tanto, el valor de `end_of_week` de esa transacción devuelve el último milisegundo de ese día, que es el 5 de febrero a las 23:59:59.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `previous_week_end`, que devuelve la marca de tiempo del inicio de la semana anterior a la transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `previous_week_end`
- `previous_week_end_timestamp`

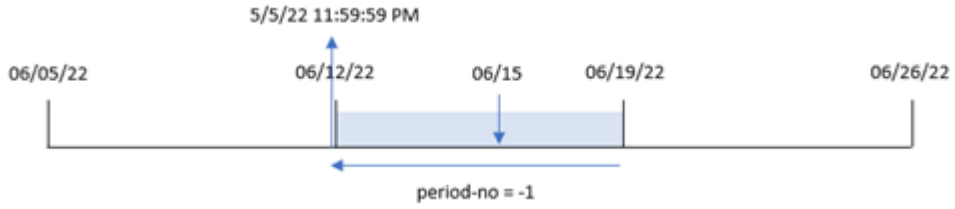
Tabla de resultados

date	end_of_week	end_of_week_timestamp
1/7/2022	01/01/2022	1/1/2022 11:59:59 PM
1/19/2022	01/15/2022	1/15/2022 11:59:59 PM
2/5/2022	01/29/2022	1/29/2022 11:59:59 PM
2/28/2022	02/26/2022	2/26/2022 11:59:59 PM
3/16/2022	03/12/2022	3/12/2022 11:59:59 PM
4/1/2022	03/26/2022	3/26/2022 11:59:59 PM

date	end_of_week	end_of_week_timestamp
5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
5/16/2022	05/14/2022	5/14/2022 11:59:59 PM
6/15/2022	06/11/2022	6/11/2022 11:59:59 PM
6/26/2022	06/25/2022	6/25/2022 11:59:59 PM
7/9/2022	07/02/2022	7/2/2022 11:59:59 PM
7/22/2022	07/16/2022	7/16/2022 11:59:59 PM
7/23/2022	07/16/2022	7/16/2022 11:59:59 PM
7/27/2022	07/23/2022	7/23/2022 11:59:59 PM
8/2/2022	07/30/2022	7/30/2022 11:59:59 PM
8/8/2022	08/06/2022	8/6/2022 11:59:59 PM
8/19/2022	08/13/2022	8/13/2022 11:59:59 PM
9/26/2022	09/24/2022	9/24/2022 11:59:59 PM
10/14/2022	10/08/2022	10/8/2022 11:59:59 PM
10/29/2022	10/22/2022	10/22/2022 11:59:59 PM

En este caso, debido a que se usó un `period_no` de -1 como argumento del desplazamiento en la función `weekend()`, la función primero identifica la semana en la que se realizan las transacciones. Luego busca una semana antes e identifica el último milisegundo de esa semana.

Diagrama de la función weekend(), ejemplo de period_no



La transacción 8196 tuvo lugar el 15 de junio. La función `weekend()` identifica que la semana comienza el 12 de junio. Por lo tanto, la semana anterior finaliza el 11 de junio a las 23:59:59; este es el valor que devuelve para el campo `previous_week_end`.

Ejemplo 3: first_week_day

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo. Sin embargo, en este ejemplo, debemos establecer el martes como el primer día de la semana laboral.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

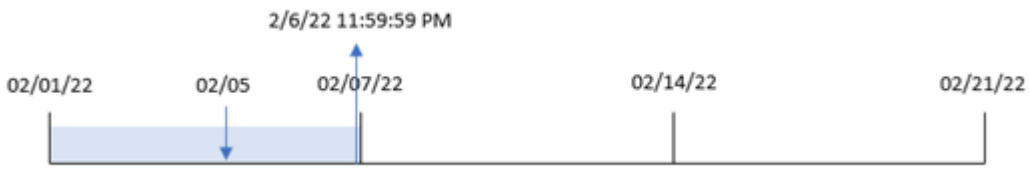
- date
- end_of_week
- end_of_week_timestamp

Tabla de resultados

date	end_of_week	end_of_week_timestamp
1/7/2022	01/10/2022	1/10/2022 11:59:59 PM
1/19/2022	01/24/2022	1/24/2022 11:59:59 PM
2/5/2022	02/07/2022	2/7/2022 11:59:59 PM
2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
3/16/2022	03/21/2022	3/21/2022 11:59:59 PM
4/1/2022	04/04/2022	4/4/2022 11:59:59 PM
5/7/2022	05/09/2022	5/9/2022 11:59:59 PM
5/16/2022	05/16/2022	5/16/2022 11:59:59 PM
6/15/2022	06/20/2022	6/20/2022 11:59:59 PM
6/26/2022	06/27/2022	6/27/2022 11:59:59 PM
7/9/2022	07/11/2022	7/11/2022 11:59:59 PM
7/22/2022	07/25/2022	7/25/2022 11:59:59 PM
7/23/2022	07/25/2022	7/25/2022 11:59:59 PM
7/27/2022	08/01/2022	8/1/2022 11:59:59 PM
8/2/2022	08/08/2022	8/8/2022 11:59:59 PM
8/8/2022	08/08/2022	8/8/2022 11:59:59 PM
8/19/2022	08/22/2022	8/22/2022 11:59:59 PM
9/26/2022	09/26/2022	9/26/2022 11:59:59 PM
10/14/2022	10/17/2022	10/17/2022 11:59:59 PM
10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

En este caso, como el argumento de `first_week_date` se utiliza en la función `weekend()`, establece el primer día de la semana en martes.

*Diagrama de la función **weekend()**, ejemplo de first_week_day*



La transacción 8191 tuvo lugar el 5 de febrero. La función `weekend()` identifica que el primer lunes después de esta fecha (y, por lo tanto, el final de la semana y el valor devuelto) fue el 6 de febrero a las 23:59:59.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo. Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que devuelve una marca de tiempo del final de la semana en que se realizaron las transacciones se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Para calcular el inicio de la semana en que se realiza una transacción, agregue las siguientes medidas:

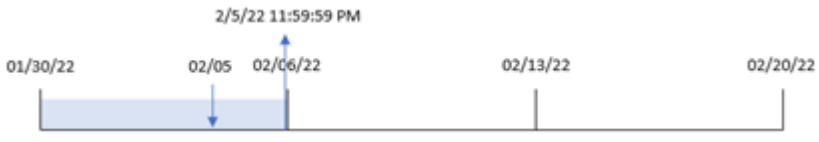
- `=weekend(date)`
- `=timestamp(weekend(date))`

Tabla de resultados

date	=weekend(date)	=timestamp(weekend(date))
1/7/2022	01/08/2022	1/8/2022 11:59:59 PM
1/19/2022	01/22/2022	1/22/2022 11:59:59 PM
2/5/2022	02/05/2022	2/5/2022 11:59:59 PM
2/28/2022	03/05/2022	3/5/2022 11:59:59 PM
3/16/2022	03/19/2022	3/19/2022 11:59:59 PM
4/1/2022	04/02/2022	4/2/2022 11:59:59 PM
5/7/2022	05/07/2022	5/7/2022 11:59:59 PM
5/16/2022	05/21/2022	5/21/2022 11:59:59 PM
6/15/2022	06/18/2022	6/18/2022 11:59:59 PM
6/26/2022	07/02/2022	7/2/2022 11:59:59 PM
7/9/2022	07/09/2022	7/9/2022 11:59:59 PM
7/22/2022	07/23/2022	7/23/2022 11:59:59 PM
7/23/2022	07/23/2022	7/23/2022 11:59:59 PM
7/27/2022	07/30/2022	7/30/2022 11:59:59 PM
8/2/2022	08/06/2022	8/6/2022 11:59:59 PM
8/8/2022	08/13/2022	8/13/2022 11:59:59 PM
8/19/2022	08/20/2022	8/20/2022 11:59:59 PM
9/26/2022	10/01/2022	10/1/2022 11:59:59 PM
10/14/2022	10/15/2022	10/15/2022 11:59:59 PM
10/29/2022	10/29/2022	10/29/2022 11:59:59 PM

La medida `end_of_week` se crea en el objeto gráfico utilizando la función `weekend()` e introduciendo el campo de fecha como argumento de la función. La función `weekend()` identifica en qué semana cae el valor de la fecha y devuelve una marca de tiempo del último milisegundo de esa semana.

*Diagrama de la función **weekend()**, ejemplo de objeto gráfico*



La transacción 8191 tuvo lugar el 5 de febrero. La variable del sistema `FirstWeekDay` establece el primer día de la semana en domingo. La función `weekend()` identifica que el primer sábado después del 5 de febrero, y por lo tanto el final de la semana, fue el 5 de febrero. Por lo tanto, el valor de `end_of_week` de esa transacción devuelve el último milisegundo de ese día, que es el 5 de febrero a las 23:59:59.

Ejemplo 5: Escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Employee_Expenses`.
- Datos que consisten en ID de empleados, nombres de empleados y declaraciones o reclamaciones de gastos diarios promedio de cada empleado.

Al usuario final le gustaría tener un objeto gráfico que muestre, por ID y nombre de empleado, las reclamaciones de gastos estimadas que aún deberán presentarse durante el resto de la semana.

Script de carga

```
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
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:
   - `employee_id`
   - `employee_name`
2. Ahora cree una medida para calcular el interés acumulado:
   `=(weekend(today(1))-today(1))*avg_daily_claim`
3. Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

employee_id	employee_name	=(weekend(today(1))-today(1))*avg_daily_claim
182	Mark	$90.00
183	Deryck	$75.00

employee_id	employee_name	=(weekend(today(1))-today(1))*avg_daily_claim
184	Dexter	$75.00
185	Sydney	$162.00
186	Agatha	$108.00

Al utilizar la fecha de hoy como único argumento en la función `weekend()`, esta devuelve la fecha de finalización de la semana actual. Luego, al restar la fecha de hoy de la fecha de finalización de la semana, la expresión devuelve el número de días que quedan de esta semana.

A continuación, este valor se multiplica por la reclamación de gastos diarios promedio de cada empleado para calcular el valor estimado de las reclamaciones que se espera que haga cada empleado en la semana restante.

weekname

Esta función devuelve un valor que muestra el número de año y de semana con un valor numérico subyacente correspondiente a una marca de tiempo del primer milisegundo del primer día de la semana que contiene a **date**.

Sintaxis:

```
WeekName(date[, period_no [, first_week_day [, broken_weeks [, reference_
day]]]])
```

La función `weekname()` determina en qué semana cae la fecha y devuelve el número de semana y el año de esa semana. El primer día de la semana viene determinado por la variable del sistema `FirstWeekDay`. No obstante, también puede cambiar el primer día de la semana usando el argumento `first_week_day` en la función `weekname()`.

En Qlik Sense, la configuración regional se obtiene cuando se crea la app y la configuración correspondiente se almacena en el script como variables de entorno.

Un desarrollador de aplicaciones norteamericano a menudo incluye `Set BrokenWeeks=1;` en el script, lo que corresponde a semanas interrumpidas. Un desarrollador de aplicaciones europeo a menudo incluye `Set BrokenWeeks=0;` en el script, lo que corresponde a semanas ininterrumpidas.

Si su aplicación utiliza semanas interrumpidas, el recuento de semanas comienza el 1 de enero y finaliza el día anterior a la variable del sistema `FirstWeekDay`, independientemente de cuántos días hayan transcurrido.

Sin embargo, si su aplicación usa semanas ininterrumpidas, la semana 1 puede comenzar en el año anterior o en los primeros días de enero. Esto depende de cómo use las variables del sistema `ReferenceDay` y `FirstWeekDay`.

Ejemplo de la función Weekname

Fecha	Nombre de semana ISO	Nombre de semana en EE. UU.
Sab 2020 Dic 26	2020/52	2020/52

Fecha	Nombre de semana ISO	Nombre de semana en EE. UU.
Dom 2020 Dic 27	2020/52	2020/53
Lun 2020 Dic 28	2020/53	2020/53
Mar 2020 Dic 29	2020/53	2020/53
Mie 2020 Dic 30	2020/53	2020/53
Jue 2020 Dic 31	2020/53	2020/53
Vie 2021 Ene 1	2020/53	2021/01
Sab 2021 Ene 2	2020/53	2021/01
Dom 2021 Ene 3	2020/53	2021/02
Lun 2021 Ene 4	2021/01	2021/02
Mar 2021 Ene 5	2021/01	2021/02

Cuándo se utiliza

La función `weekname()` es útil cuando desea comparar agregaciones por semanas.

Por ejemplo, si desea ver el total de ventas de productos por semana. Para mantener la coherencia con la variable de entorno `BrokenWeeks` en la aplicación, utilice `weekname()` en vez de `lunarweekname()`. Si la aplicación utiliza semanas ininterrumpidas, la semana 1 puede contener fechas de diciembre del año anterior o excluir fechas de enero del año en curso. Si la aplicación utiliza semanas segmentadas, la semana 1 puede contener menos de siete días.

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
timestamp	La fecha o marca de tiempo para evaluar.
period_no	**shift** es un entero, donde el valor 0 indica la semana que contiene a **date**. Los valores negativos en el desplazamiento indican semanas precedentes y los valores positivos indican semanas subsiguientes.
first_week_day	Especifica el día en el que se inicia la semana. Si se omite, se utiliza el valor de la variable **FirstWeekDay**. Los valores posibles de **first_week_day** son 0 para el lunes, 1 para el martes, 2 para el miércoles, 3 para el jueves, 4 para el viernes, 5 para el sábado y 6 para el domingo. Para más información sobre la variable del sistema, vea *FirstWeekDay (page 229)*.
broken_weeks	Si no especifica **broken_weeks**, el valor de la variable **BrokenWeeks** se usará para definir si las semanas se rompen o no.

Argumento	Descripción
reference_day	Si no especifica **reference_day**, el valor de la variable **ReferenceDay** se usará para definir qué día de enero debe configurarse como día de referencia para definir la semana 1. De forma predeterminada, las funciones Qlik Sense usan 4 como día de referencia. Esto significa que la semana 1 debe incluir el 4 de enero o, dicho de otro modo, que la semana 1 siempre debe incluir 4 días de enero como mínimo.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Los siguientes ejemplos suponen:

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

Ejemplos de funciones

Ejemplo	Resultado
weekname('01/12/2013')	Devuelve 2013/02.
weekname('01/12/2013', -1)	Devuelve 2013/01.
weekname('01/12/2013', 0, 1)	Devuelve 2013/02.

Ejemplo 1: Fecha sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de la última semana de 2021 y las primeras dos semanas de 2022 se carga en una tabla llamada "Transactions".
- La variable del sistema DateFormat configurada en el formato MM/DD/YYYY.
- La variable del sistema BrokenWeeks configurada en 1.
- La variable del sistema FirstWeekDay configurada en 6.
- Un load precedente que contiene lo siguiente:
    - La función weekday() que se establece como el campo, ''week_number", que devuelve el año y el número de semana en que se realizaron las transacciones.
    - La función weekname() que se establece como el campo denominado "week_day", para mostrar el valor del día de la semana de cada fecha de transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`
- `week_day`
- `week_number`

Tabla de resultados

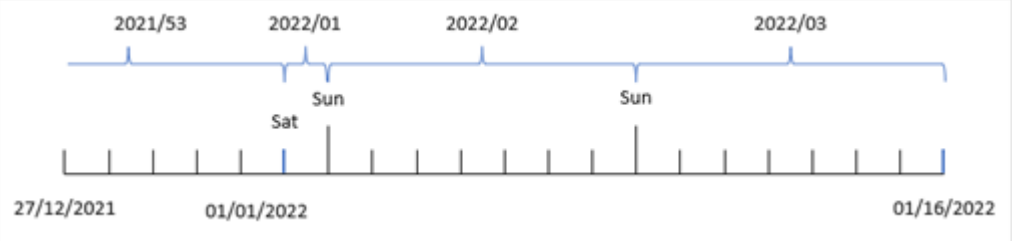
id	date	week_day	week_number
8183	12/27/2021	Lun	2021/53
8184	12/28/2021	Mar	2021/53
8185	12/29/2021	Mié	2021/53
8186	12/30/2021	Jue	2021/53
8187	12/31/2021	Vie	2021/53
8188	01/01/2022	Sáb	2022/01
8189	01/02/2022	Dom	2022/02
8190	01/03/2022	Lun	2022/02
8191	01/04/2022	Mar	2022/02
8192	01/05/2022	Mié	2022/02
8193	01/06/2022	Jue	2022/02
8194	01/07/2022	Vie	2022/02
8195	01/08/2022	Sáb	2022/02
8196	01/09/2022	Dom	2022/03
8197	01/10/2022	Lun	2022/03
8198	01/11/2022	Mar	2022/03
8199	01/12/2022	Mié	2022/03
8200	01/13/2022	Jue	2022/03
8201	01/14/2022	Vie	2022/03

El campo `week_number` se crea en la instrucción load anterior utilizando la función `weekname()` e introduciendo el campo de fecha como argumento de la función.

La función `weekname()` inicialmente identifica en qué semana cae el valor de la fecha y devuelve el recuento del número de semana y el año en que se lleva a cabo la transacción.

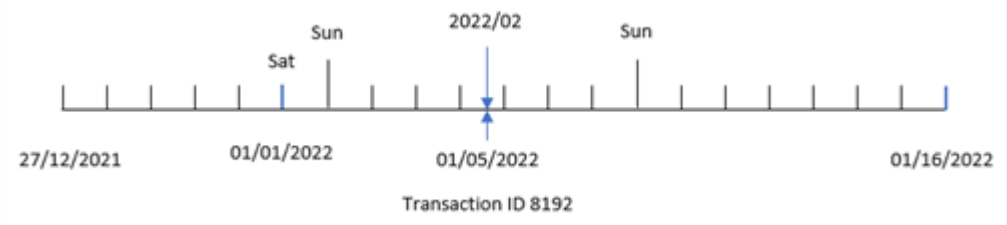
La variable de sistema `FirstWeekDay` establece el domingo como el primer día de la semana. La variable de sistema `BrokenWeeks` establece que la aplicación utilice semanas divididas, lo que significa que la semana 1 comenzará el 1 de enero.

*Diagrama de la función **weekname()** con las variables predeterminadas.*



La semana 1 comienza el 1 de enero, que es sábado y, por lo tanto, las transacciones que ocurren en esta fecha devuelven el valor `2022/01` (el año y el número de semana).

*Diagrama de la función **weekname()** que identifica el número de semana de la transacción 8192.*



Debido a que la aplicación usa semanas divididas y el primer día de la semana es el domingo, las transacciones que ocurren del 2 al 8 de enero devuelven el valor `2022/02` (semana número 2 de 2022). Un ejemplo de esto sería la transacción 8192 que tuvo lugar el 5 de enero y devuelve el valor `2022/02` para el campo "`week_number`".

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, la tarea es crear un campo, "`previous_week_number`", que devuelva el año y el número de semana anteriores a que se realizaran las transacciones.

Abra el Editor de carga de datos y agregue el siguiente script de carga en una nueva pestaña.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- id
- date
- week_day
- week_number

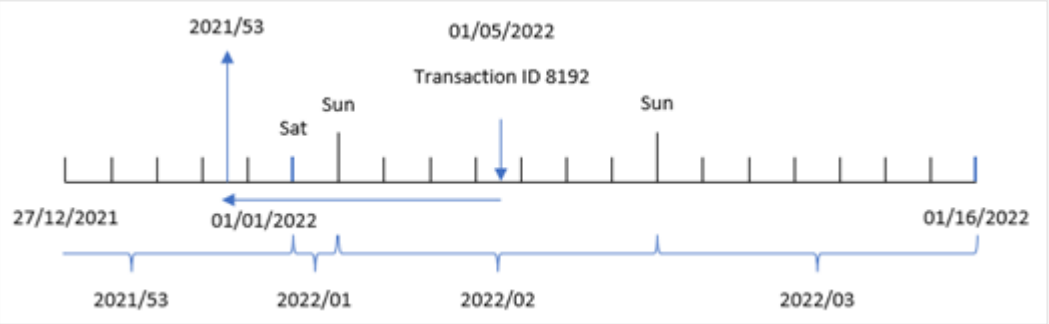
Tabla de resultados

id	date	week_day	week_number
8183	12/27/2021	Lun	2021/52
8184	12/28/2021	Mar	2021/52
8185	12/29/2021	Mié	2021/52
8186	12/30/2021	Jue	2021/52
8187	12/31/2021	Vie	2021/52
8188	01/01/2022	Sáb	2021/52

id	date	week_day	week_number
8189	01/02/2022	Dom	2021/53
8190	01/03/2022	Lun	2021/53
8191	01/04/2022	Mar	2021/53
8192	01/05/2022	Mié	2021/53
8193	01/06/2022	Jue	2021/53
8194	01/07/2022	Vie	2021/53
8195	01/08/2022	Sáb	2022/01
8196	01/09/2022	Dom	2022/02
8197	01/10/2022	Lun	2022/02
8198	01/11/2022	Mar	2022/02
8199	01/12/2022	Mié	2022/02
8200	01/13/2022	Jue	2022/02
8201	01/14/2022	Vie	2022/02

En este caso, debido a que se usó un valor `period_no` de `-1` como argumento de desplazamiento en la función `weekname()`, la función primero identifica la semana en la que se realizan las transacciones. Luego busca una semana antes e identifica el primer milisegundo de esa semana.

Diagrama de la función `weekname()` con un desplazamiento de `period_no` de -1.



La transacción 8192 tuvo lugar el 5 de enero de 2022. La función `weekname()` busca una semana antes, el 30 de diciembre de 2021, y devuelve el número de semana y el año de esa fecha: `2021/53`.

Ejemplo: first_week_day

Script de carga y resultados

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, la política de la empresa es que la semana laboral comience el martes.

Abra el Editor de carga de datos y agregue el siguiente script de carga en una nueva pestaña.

Script de carga

```
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
];
```

Resultados

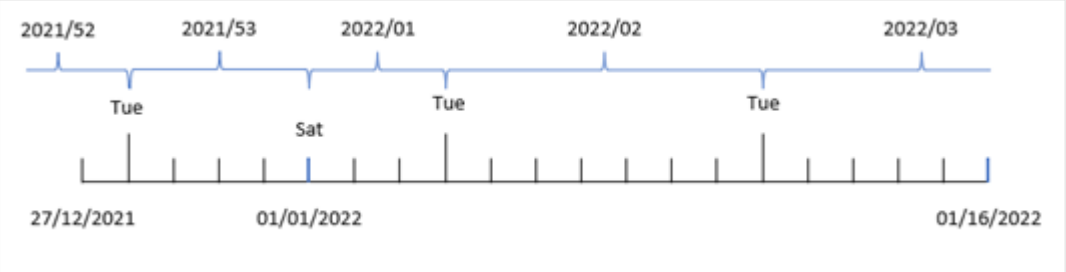
Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- id
- date
- week_day
- week_number

Tabla de resultados

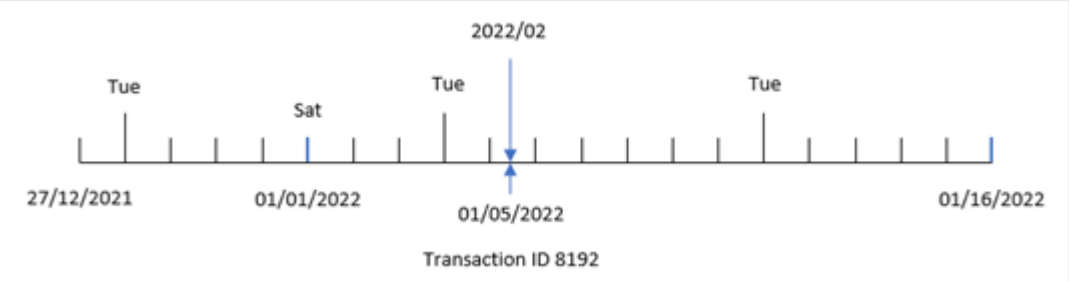
id	date	week_day	week_number
8183	12/27/2021	Lun	2021/52
8184	12/28/2021	Mar	2021/53
8185	12/29/2021	Mié	2021/53
8186	12/30/2021	Jue	2021/53
8187	12/31/2021	Vie	2021/53
8188	01/01/2022	Sáb	2022/01
8189	01/02/2022	Dom	2022/01
8190	01/03/2022	Lun	2022/01
8191	01/04/2022	Mar	2022/02
8192	01/05/2022	Mié	2022/02
8193	01/06/2022	Jue	2022/02
8194	01/07/2022	Vie	2022/02
8195	01/08/2022	Sáb	2022/02
8196	01/09/2022	Dom	2022/02
8197	01/10/2022	Lun	2022/02
8198	01/11/2022	Mar	2022/03
8199	01/12/2022	Mié	2022/03
8200	01/13/2022	Jue	2022/03
8201	01/14/2022	Vie	2022/03

Diagrama de la función weekname() con el martes como primer día de la semana.



Como el argumento `first_week_date` de `1` se usa en la función `weekname()`, utiliza el martes como primer día de la semana. La función por tanto determina que la semana 53 de 2021 comience el martes 28 de diciembre; y, debido a que la aplicación utiliza semanas divididas, la semana 1 comienza el 1 de enero de 2022 y finaliza el último milisegundo del lunes 3 de enero de 2022.

Diagrama que muestra el número de semana de la transacción 8192 con el martes como primer día de la semana.



La transacción 8192 tuvo lugar el 5 de enero de 2022. Por lo tanto, utilizando un parámetro `first_week_day` de martes, la función `weekname()` devuelve el valor `2022/02` del campo "`week_number`".

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que devuelve el número de año de la semana en que se realizaron las transacciones se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`
- `=week_day (date)`

Para calcular el inicio de la semana en que se realiza una transacción, cree la siguiente medida:

```
=weekname(date)
```

Tabla de resultados

id	date	=weekday(date)	=weekname(date)
8183	12/27/2021	Lun	2021/53
8184	12/28/2021	Mar	2021/53
8185	12/29/2021	Mié	2021/53
8186	12/30/2021	Jue	2021/53
8187	12/31/2021	Vie	2021/53
8188	01/01/2022	Sáb	2022/01
8189	01/02/2022	Dom	2022/02
8190	01/03/2022	Lun	2022/02
8191	01/04/2022	Mar	2022/02
8192	01/05/2022	Mié	2022/02
8193	01/06/2022	Jue	2022/02
8194	01/07/2022	Vie	2022/02
8195	01/08/2022	Sáb	2022/02
8196	01/09/2022	Dom	2022/03
8197	01/10/2022	Lun	2022/03
8198	01/11/2022	Mar	2022/03
8199	01/12/2022	Mié	2022/03
8200	01/13/2022	Jue	2022/03
8201	01/14/2022	Vie	2022/03

El campo "`week_number`" se crea como una medida en el objeto gráfico utilizando la función `weekname()` e introduciendo el campo de fecha como el argumento de la función.

La función `weekname()` identifica inicialmente en qué semana cae el valor de la fecha y devuelve el recuento del número de semana y el año en que se lleva a cabo la transacción.

La variable de sistema `FirstWeekDay` establece el domingo como el primer día de la semana. La variable de sistema `BrokenWeeks` configura la aplicación para usar semanas divididas, lo que significa que la semana 1 comienza el 1 de enero.

Diagrama que muestra el número de semana con el domingo como primer día de la semana.

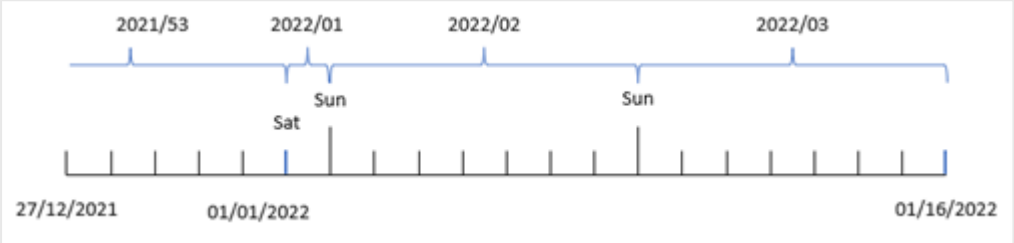
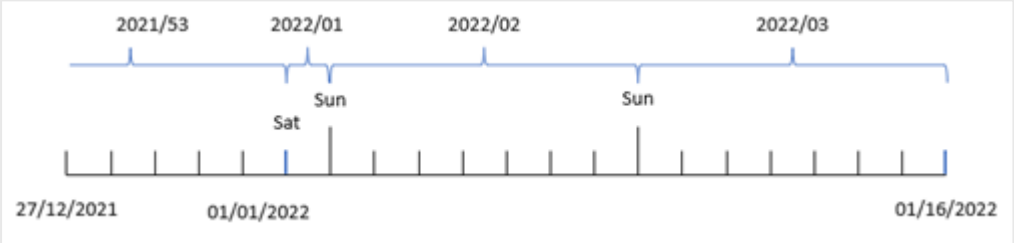


Diagrama que muestra que la transacción 8192 tuvo lugar en la semana número dos.



Debido a que la aplicación utiliza semanas divididas y el primer día de la semana es el domingo, las transacciones que ocurren del 2 al 8 de enero devuelven el valor 2022/02, la semana número 2 en 2022. Observe que la transacción 8192 tuvo lugar el 5 de enero y devuelve el valor 2022/02 para el campo "`week_number`".

Ejemplo 5: Escenario

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se carga un conjunto de datos que contiene un conjunto de transacciones de la última semana de 2019 y las primeras dos semanas de 2020 en una tabla llamada "`Transactions`".
- La variable del sistema `BrokenWeeks` configurada en `0`.
- La variable del sistema `ReferenceDay` configurada en `2`.
- La variable del sistema `DateFormat` configurada en el formato `MM/DD/YYYY`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla.

Cree una dimensión calculada usando la siguiente expresión:

```
=weekname(date)
```

Para calcular el total de ventas, cree la siguiente medida de agregación:

```
=sum(amount)
```

Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

weekname(date)	=sum(amount)
2019/52	$125.69
2020/01	$346.51

weekname(date)	=sum(amount)
2020/02	$347.57
2020/03	$122.01

Para demostrar los resultados de usar la función weekname() en este escenario, agregue el siguiente campo como una dimensión:

`date`

Tabla de resultados con el campo de fecha

weekname(date)	date	=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

Como la aplicación utiliza semanas ininterrumpidas y la semana 1 requiere un mínimo de dos días en enero debido a la variable de sistema `ReferenceDay`, la semana 1 de 2020 incluye transacciones desde el 29 de diciembre de 2019.

weekstart

Esta función devuelve un valor correspondiente a una marca de tiempo del primer milisegundo del primer día de la semana natural que contiene a **date**. El formato de salida predefinido es el **DateFormat** definido en el script.

Sintaxis:

```
WeekStart(timestamp [, period_no [, first_week_day ]])
```

Tipo de datos que devuelve: dual

La función `weekstart()` determina en qué semana cae la fecha. Luego devuelve una marca de tiempo, en formato de fecha, con el primer milisegundo de esa semana. El primer día de la semana viene determinado por la variable de entorno `FirstWeekDay`. No obstante, esto puede ser reemplazado por el argumento de `first_week_day` en la función `weekstart()`.

Argumentos

Argumento	Descripción
timestamp	La fecha o marca de tiempo para evaluar.
period_no	**shift** es un entero, donde el valor 0 indica la semana que contiene a **date**. Los valores negativos en el desplazamiento indican semanas precedentes y los valores positivos indican semanas subsiguientes.
first_week_day	Especifica el día en el que se inicia la semana. Si se omite, se utiliza el valor de la variable **FirstWeekDay**. Los valores posibles de **first_week_day** son 0 para el lunes, 1 para el martes, 2 para el miércoles, 3 para el jueves, 4 para el viernes, 5 para el sábado y 6 para el domingo. Para más información sobre la variable del sistema, vea *FirstWeekDay (page 229)*.

Cuándo se utiliza

La función `weekstart()` se suele utilizar como parte de una expresión cuando el usuario desea que el cálculo utilice la fracción de la semana que ya ha transcurrido. Por ejemplo, se podría usar para calcular los salarios totales obtenidos por los empleados en la semana transcurrida hasta ahora.

Los siguientes ejemplos suponen:

```
SET FirstWeekDay=0;
```

Ejemplos de funciones

Ejemplo	Resultado
`weekstart('01/12/2013')`	Devuelve 01/07/2013.
`weekstart('01/12/2013', -1 )`	Devuelve 11/31/2012.
`weekstart('01/12/2013', 0, 1)`	Devuelve 01/08/2013.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos:

Si desea configuraciones ISO para semanas y números de semana, asegúrese de tener lo siguiente en el script:

```
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
```

Si desea la configuración de EE. UU., asegúrese de tener lo siguiente en el script:

```
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
```

Los ejemplos anteriores dan como resultado lo siguiente, de la función `weekstart()`:

Ejemplo de la función Weekstart

Fecha	**Inicio de la semana ISO**	**Inicio de semana en EE. UU.**
Sab 2020 Dic 26	2020-12-21	12/20/2020
Dom 2020 Dic 27	2020-12-21	12/27/2020
Lun 2020 Dic 28	2020-12-28	12/27/2020
Mar 2020 Dic 29	2020-12-28	12/27/2020
Mie 2020 Dic 30	2020-12-28	12/27/2020
Jue 2020 Dic 31	2020-12-28	12/27/2020
Vie 2021 Ene 1	2020-12-28	12/27/2020
Sab 2021 Ene 2	2020-12-28	12/27/2020
Dom 2021 Ene 3	2020-12-28	1/3/2021

Fecha	Inicio de la semana ISO	Inicio de semana en EE. UU.
Lun 2021 Ene 4	2021-01-04	1/3/2021
Mar 2021 Ene 5	2021-01-04	1/3/2021

El inicio de la semana es los lunes en la columna ISO y los domingos en la columna de EE. UU.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de 2022, que se carga en una tabla llamada "`Transactions`".
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat`: (MM/DD/AAAA).
- La creación de un campo, `start_of_week`, que devuelve una marca de tiempo con el inicio de la semana en que las transacciones tuvieron lugar.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- start_of_week
- start_of_week_timestamp

Tabla de resultados

date	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	05/01/2022	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	8/7/2022 12:00:00 AM
8/19/2022	08/14/2022	8/14/2022 12:00:00 AM

date	start_of_week	start_of_week_timestamp
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

El campo `start_of_week` se crea en la instrucción load anterior utilizando la función `weekstart()` e introduciendo el campo de fecha como argumento de la función.

La función `weekstart()` identifica inicialmente en qué semana cae el valor de la fecha y devuelve una marca de tiempo del primer milisegundo de esa semana.

Diagrama de la función `weekstart()`, ejemplo sin argumentos adicionales

La transacción 8191 tuvo lugar el 5 de febrero. La variable del sistema `FirstWeekDay` establece el primer día de la semana en domingo. La función `weekstart()` identifica que el primer domingo antes del 5 de febrero, y por lo tanto el inicio de la semana, fue el 30 de enero. Por lo tanto, el valor de `start_of_week` de esa transacción devuelve el primer milisegundo de ese día, que es el 30 de enero a las 12:00:00 a. m.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `previous_week_start`, que devuelve la marca de tiempo de inicio del trimestre anterior a la transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `previous_week_start`
- `previous_week_start_timestamp`

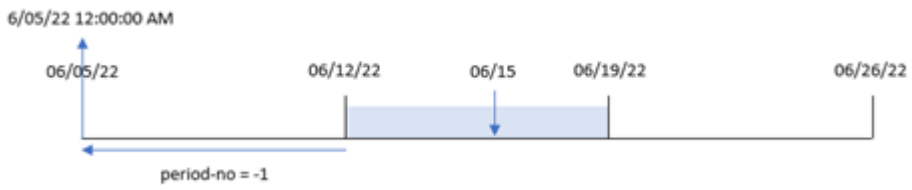
Tabla de resultados

date	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

date	previous_week_start	previous_week_start_timestamp
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
8/8/2022	07/31/2022	7/31/2022 12:00:00 AM
8/19/2022	08/07/2022	8/7/2022 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

En este caso, como se usó un `period_no` de -1 como argumento de desplazamiento en la función `weekstart()`, la función primero identifica la semana en la que se realizan las transacciones. Luego busca una semana antes e identifica el primer milisegundo de esa semana.

Diagrama de la función `weekstart()`, ejemplo de period_no



La transacción 8196 tuvo lugar el 15 de junio. La función `weekstart()` identifica que la semana comienza el 12 de junio. Por lo tanto, la semana anterior comenzó el 5 de junio a las 12:00:00 a. m.; este es el valor que devuelve para el campo `previous_week_start`.

Ejemplo 3: first_week_day

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo. Sin embargo, en este ejemplo, debemos establecer el martes como el primer día de la semana laboral.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- start_of_week
- start_of_week_timestamp

Tabla de resultados

date	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

date	start_of_week	start_of_week_timestamp
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
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

En este caso, como el argumento de `first_week_date` se utiliza en la función `weekstart()`, establece el primer día de la semana en martes.

Diagrama de la función `weekstart()`, ejemplo de first_week_day

La transacción 8191 tuvo lugar el 5 de febrero. La función `weekstart()` identifica que el primer martes anterior a esta fecha y, por lo tanto, el inicio de la semana y el valor devuelto, fue el 1 de febrero a las 12:00:00 a. m.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que devuelve una marca de tiempo del inicio de la semana en que se realizaron las transacciones se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Para calcular el inicio de la semana en que se realiza una transacción, agregue las siguientes medidas:

- `=weekstart(date)`
- `=timestamp(weekstart(date))`

Tabla de resultados

date	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	05/01/2022	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	8/7/2022 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

La medida `start_of_week` se crea en el objeto gráfico utilizando la función `weekstart()` e introduciendo el campo `date` como argumento de la función.

La función `weekstart()` identifica inicialmente en qué semana cae el valor de la fecha y devuelve una marca de tiempo del primer milisegundo de esa semana.

Diagrama de la función weekstart(), ejemplo de objeto gráfico

La transacción 8191 tuvo lugar el 5 de febrero. La variable del sistema `FirstWeekDay` establece el primer día de la semana en domingo. La función `weekstart()` identifica que el primer domingo anterior al 5 de febrero – y por lo tanto el inicio de la semana– fue el 30 de enero. Por lo tanto, el valor `start_of_week` de esa transacción devuelve el primer milisegundo de ese día, que es el 30 de enero a las 12:00:00 a. m.

Ejemplo 5: Escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que se carga en una tabla denominada `Payroll`.
- Datos que consisten en identificaciones de empleados, nombres de empleados y el salario diario que gana cada empleado.

Los empleados comienzan a trabajar el lunes y trabajan seis días a la semana. La variable del sistema `FirstWeekDay` no debe modificarse.

Al usuario final le gustaría tener un objeto gráfico que muestre, por ID de empleado y nombre de empleado, los salarios obtenidos en la semana hasta la fecha.

Script de carga

```
Payroll:
Load
*
Inline
[
employee_id,employee_name,day_rate
182,Mark, $150
183,Deryck, $125
184,Dexter, $125
185,Sydney,$270
186,Agatha,$128
];
```

Resultados

Haga lo siguiente:

1. Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:
   - `employee_id`
   - `employee_name`
2. A continuación, cree una medida para calcular los salarios obtenidos en la semana hasta la fecha: `=if(today(1)-weekstart(today(1),0,0)<7,(today(1)-weekstart(today(1),0,0))*day_rate,day_rate*6)`
3. Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

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

La función `weekstart()`, al utilizar la fecha de hoy como primer argumento y 0 como tercer argumento, establece el lunes como el primer día de la semana y devuelve la fecha de inicio de la semana actual. Al restar ese resultado de la fecha actual, la expresión devuelve a continuación el número de días que han transcurrido en lo que va de semana.

La condición luego evalúa si ha habido más de seis días esta semana. Si es así, la cifra `day_rate` del empleado se multiplica por 6 días. Sino es así, la cifra `day_rate` se multiplica por el número de días que han transcurrido en lo que va de semana.

weekyear

Esta función devuelve el año al que pertenece el número de semana según las variables de entorno. El número de semana varía entre 1 y 52 aproximadamente.

Sintaxis:

```
weekyear(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

Tipo de datos que devuelve: Entero

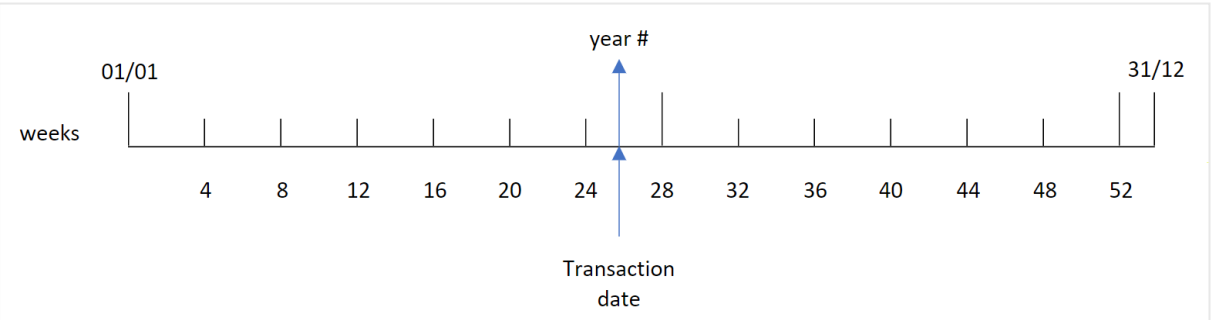
Argumentos

Argumento	Descripción
timestamp	La fecha o marca de tiempo para evaluar.
first_week_day	Especifica el día en el que se inicia la semana. Si se omite, se utiliza el valor de la variable **FirstWeekDay**. Los valores posibles de **first_week_day** son 0 para el lunes, 1 para el martes, 2 para el miércoles, 3 para el jueves, 4 para el viernes, 5 para el sábado y 6 para el domingo. Para más información sobre la variable del sistema, vea *FirstWeekDay (page 229)*.
broken_weeks	Si no especifica **broken_weeks**, el valor de la variable **BrokenWeeks** se usará para definir si las semanas se rompen o no.
reference_day	Si no especifica **reference_day**, el valor de la variable **ReferenceDay** se usará para definir qué día de enero debe configurarse como día de referencia para definir la semana 1. De forma predeterminada, las funciones Qlik Sense usan 4 como día de referencia. Esto significa que la semana 1 debe incluir el 4 de enero o, dicho de otro modo, que la semana 1 siempre debe incluir 4 días de enero como mínimo.

La función `weekyear()` determina en qué semana de un año cae una fecha. Luego devuelve el año correspondiente a ese número de semana.

Si `BrokenWeeks` está configurado como 0 (false), `weekyear()` devolverá lo mismo que `year()`.

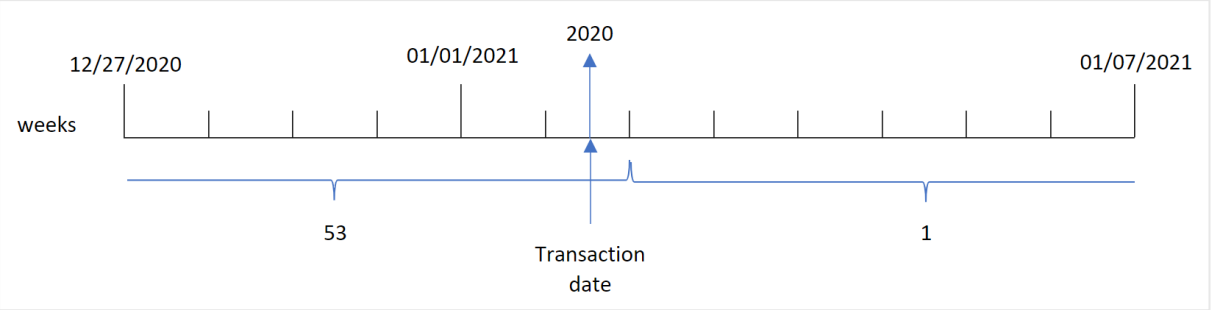
Diagrama del rango de la función `weekyear()`



Sin embargo, si la variable del sistema `BrokenWeeks` está configurada para usar semanas ininterrumpidas, la semana 1 solo debe contener una cierta cantidad de días en enero conforme al valor especificado en la variable del sistema `ReferenceDay`.

Por ejemplo, si se utiliza un valor `ReferenceDay` de 4, la semana 1 debe incluir al menos cuatro días en enero. Es posible que la semana 1 incluya fechas de diciembre del año anterior o que el número de semana final de un año incluya fechas de enero del año siguiente. En situaciones como esta, la función `weekyear()` devolverá un valor diferente al de la función `year()`.

Diagrama del rango de la función `weekyear()` cuando se usan semanas no interrumpidas



Cuándo se utiliza

La función `weekyear()` es útil cuando se desea comparar agregaciones por años. Por ejemplo, si desea ver el total de ventas de productos por año. La función `weekyear()` se elige por delante de `year()` cuando el usuario desea mantener la coherencia con la variable del sistema `BrokenWeeks` en la app.

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
`weekyear('12/30/1996',0,0,4)`	devuelve 1997, porque la semana 1 de 1997 empieza el 30/12/1996
`weekyear('01/02/1997',0,0,4)`	Devuelve 1997
`weekyear('12/28/1997',0,0,4)`	Devuelve 1997
`weekyear('12/30/1997',0,0,4)`	Devuelve 1998, porque la semana 1 de 1998 empieza el 29/12/1997
`weekyear('01/02/1999',0,0,4)`	Devuelve 1998, porque la semana 53 de 1998 finaliza el 03/01/1999

Temas relacionados

Tema	Interacción
week (page 1070)	Esta función devuelve un entero que representa el número de semana conforme a la ISO 8601
year (page 1145)	Devuelve un entero que representa el año en que la expresión se interpreta como una fecha según la interpretación numérica estándar.

Ejemplo 1: semanas interrumpidas

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de la última semana de 2020 y la primera semana de 2021, que se carga en una tabla llamada "`Transactions`".
- La variable `BrokenWeeks` que está establecida en 1.
- Un load precedente que contiene lo siguiente:
    - La función `weekyear()`, configurada como el campo ''`week_year`", que devuelve el año en que se realizaron las transacciones.
    - La función `week()`, configurada como el campo "`week`" que muestra el número de semana de cada fecha de transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`
- `week`
- `week_year`

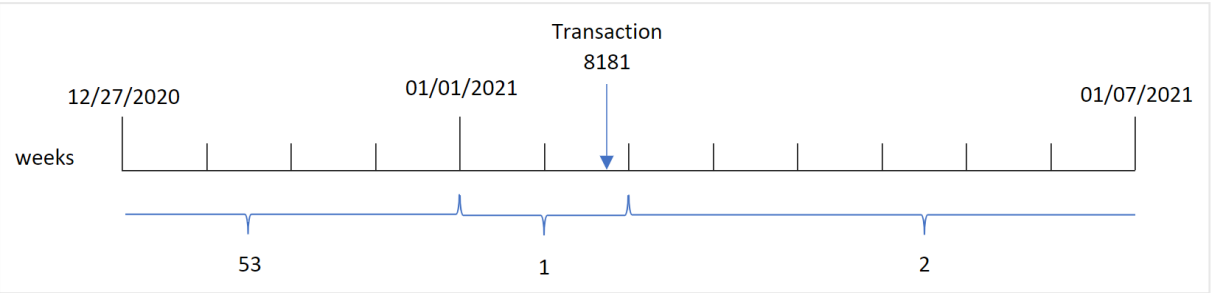
Tabla de resultados

id	date	semana	week_year
8176	12/28/2020	53	2020
8177	12/29/2020	53	2020
8178	12/30/2020	53	2020
8179	12/31/2020	53	2020
8180	01/01/2021	1	2021
8181	01/02/2021	1	2021
8182	01/03/2021	2	2021
8183	01/04/2021	2	2021
8184	01/05/2021	2	2021
8185	01/06/2021	2	2021
8186	01/07/2021	2	2021

El campo `week_year` se crea en la instrucción load anterior utilizando la función `weekyear()` e introduciendo el campo de fecha como argumento de la función.

La variable del sistema `BrokenWeeks` está configurada en `1` para indicar que la aplicación utiliza semanas interrumpidas. La semana `1` comienza el `1` de enero.

Diagrama del rango de la función weekyear() cuando se usan semanas interrumpidas



La transacción 8181 tiene lugar el 2 de enero, que forma parte de la semana 1. Por lo tanto, devuelve un valor de 2021 para el campo "week_year".

Ejemplo 2: semanas no interrumpidas

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de la última semana de 2020 y la primera semana de 2021, que se carga en una tabla llamada "Transactions".
- La variable BrokenWeeks que está establecida en "0".
- Un load precedente que contiene lo siguiente:
  - La función weekyear(), configurada como el campo ''week_year" que devuelve el año en que se realizaron las transacciones.
  - La función week(), configurada como el campo "week" que muestra el número de semana de cada fecha de transacción.

Sin embargo, en este ejemplo, la política de la empresa es utilizar semanas no interrumpidas.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- id
- date
- week
- week_year

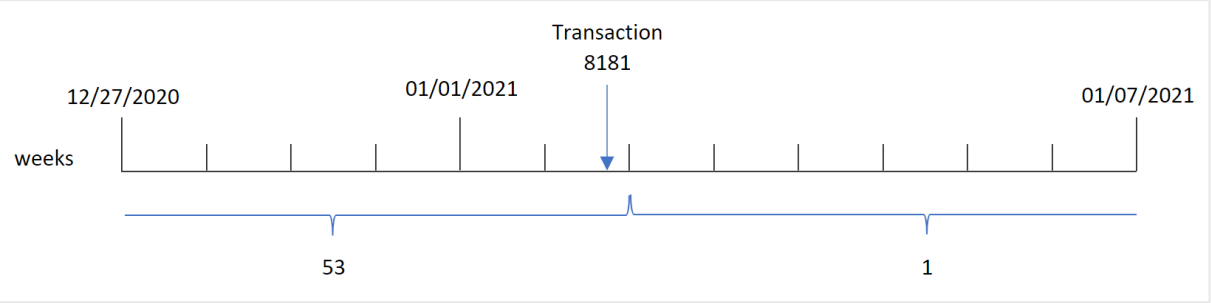
Tabla de resultados

id	date	semana	week_year
8176	12/28/2020	53	2020
8177	12/29/2020	53	2020
8178	12/30/2020	53	2020
8179	12/31/2020	53	2020
8180	01/01/2021	53	2020
8181	01/02/2021	53	2020
8182	01/03/2021	1	2021
8183	01/04/2021	1	2021
8184	01/05/2021	1	2021
8185	01/06/2021	1	2021
8186	01/07/2021	1	2021

La variable del sistema `BrokenWeeks` está configurada en `0` para indicar que la aplicación utiliza semanas no interrumpidas. Por lo tanto, no se requiere que la semana 1 comience el 1 de enero.

La semana 53 de 2020 continúa hasta el final del 2 de enero de 2021; y la semana 1 de 2020 comienza el domingo 3 de enero de 2021.

Diagrama del rango de la función weekyear() con semanas no interrumpidas



La transacción 8181 tiene lugar el 2 de enero, que forma parte de la semana 1. Por lo tanto, devuelve un valor de 2021 para el campo "week_year".

Ejemplo 3: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que devuelve el número de semana del año en que se realizaron las transacciones se crea como una medida en un gráfico de la app.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- id
- date

Para calcular la semana en que se realiza una transacción, cree la siguiente medida:

- =week(date)

Para calcular el año en que se realiza una transacción en función del número de semana, cree la siguiente medida:

- =weekyear(date)

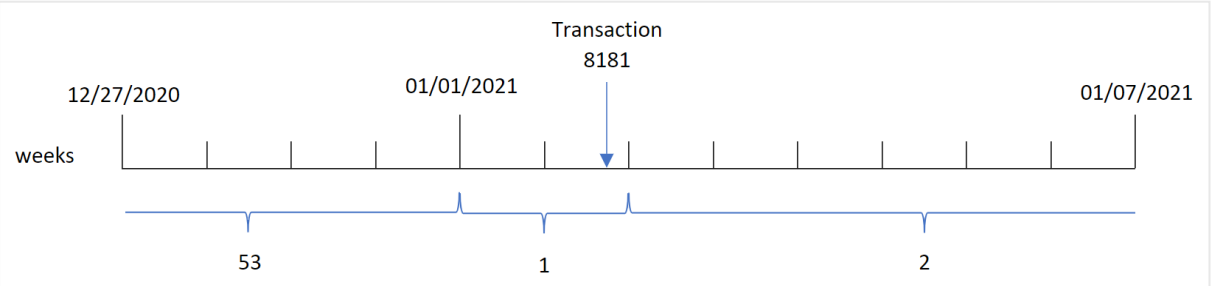
Tabla de resultados

id	date	semana	week_year
8176	12/28/2020	53	2020
8177	12/29/2020	53	2020
8178	12/30/2020	53	2020
8179	12/31/2020	53	2020
8180	01/01/2021	1	2021
8181	01/02/2021	1	2021
8182	01/03/2021	2	2021
8183	01/04/2021	2	2021
8184	01/05/2021	2	2021
8185	01/06/2021	2	2021
8186	01/07/2021	2	2021

El campo week_year se crea en la instrucción load anterior utilizando la función weekyear() e introduciendo el campo de fecha como argumento de la función.

La variable del sistema BrokenWeeks está configurada en 1, lo que significa que la app utiliza semanas interrumpidas. La semana 1 comienza el 1 de enero

Diagrama del rango de la función weekyear() cuando se usan semanas interrumpidas



La transacción 8181 tiene lugar el 2 de enero, que forma parte de la semana 1. Por lo tanto, devuelve un valor de 2021 para el campo "week_year".

Ejemplo 4: escenario

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones de la última semana de 2020 y la primera semana de 2021, que se carga en una tabla llamada "Transactions".
- La variable BrokenWeeks que está establecida en "0". Esto significa que la aplicación utilizará semanas ininterrumpidas.
- La variable ReferenceDay que está establecida en "2". Esto significa que el año comenzará el 2 de enero y contendrá un mínimo de dos días en enero.
- La variable FirstWeekDay que está establecida en "1". Esto significa que el primer día de la semana será el martes.

La política de la empresa es utilizar semanas interrumpidas. Al usuario final le gustaría tener un gráfico que presente el total de ventas por año. La app usa semanas ininterrumpidas y la semana 1 contiene un mínimo de dos días en enero.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla.

Para calcular el año en que se realiza una transacción en función del número de semana, cree la siguiente medida:

- `=weekyear(date)`

Para calcular el total de ventas, cree la siguiente medida:

- `sum(amount)`

Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

weekyear(date)	=sum(amount)
2020	19.42
2021	373.37

year

Esta función devuelve un entero que representa el año en que **expression** se interpreta como una fecha de acuerdo con la interpretación numérica estándar.

Sintaxis:

```
year(expression)
```

Tipo de datos que devuelve: Entero

La función `year()` está disponible como función de script y de gráfico. La función devuelve el año de una fecha en particular. Normalmente se utiliza para crear un campo de año como dimensión en un calendario maestro.

Cuándo se utiliza

La función `year()` es útil cuando se desea comparar agregaciones por año. Por ejemplo, si desea ver el total de ventas de productos por año.

Estas dimensiones se pueden crear o bien en el script de carga, utilizando la función para crear un campo en una tabla de calendario maestro. Alternativamente, podría usarse directamente en un gráfico como una dimensión calculada.

Ejemplos de funciones

Ejemplo	Resultado
`year( '2012-10-12' )`	devuelve 2012
`year( '35648' )`	devuelve 1997, porque 35648 = 1997-08-06

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: conjunto de datos DateFormat (script)

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos de fechas que se carga en una tabla denominada `Master Calendar`.
- Se utiliza la variable predefinida del sistema `DateFormat` (DD/MM/AAAA).
- Un load precedente, que se utiliza para crear un campo adicional, "`year`", usando la función `year()`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `year`

Tabla de resultados

date	**año**
12/28/2020	2020
12/29/2020	2020
12/30/2020	2020
12/31/2020	2020
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	2021

Ejemplo 2: fechas ANSI

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de fechas que se carga en una tabla denominada `Master Calendar`.
- Se utiliza la variable predefinida del sistema `DateFormat`: DD/MM/AAAA. Sin embargo, las fechas incluidas en el conjunto de datos están en el formato de fecha estándar ANSI.

- Un load precedente, que se utiliza para crear un campo adicional, llamado `year`, usando la función `year()`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `year`

Tabla de resultados

date	**año**
2020-12-28	2020
2020-12-29	2020
2020-12-30	2020
2020-12-31	2020
2021-01-01	2021
2021-01-02	2021
2021-01-03	2021

date	año
2021-01-04	2021
2021-01-05	2021
2021-01-06	2021
2021-01-07	2021

Ejemplo 3: fechas sin formato

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de fechas, en formato numérico, que se carga en una tabla denominada `Master Calendar`.
- Se utiliza la variable predefinida del sistema `DateFormat`: DD/MM/AAAA.
- Un load precedente, que se utiliza para crear un campo adicional, "`year`", usando la función `year()`.

Se carga la fecha original sin formato, llamada `unformatted_date` y, para proporcionar claridad, se usa un campo adicional, llamado `long_date`, para convertir la fecha numérica en un campo de fecha con formato usando la función `date()`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `unformatted_date`
- `long_date`
- `year`

Tabla de resultados

unformatted_date	long_date	año
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

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

En este ejemplo, se carga un conjunto de datos de pedidos en una tabla denominada "Sales". La tabla contiene tres campos:

- `id`
- `sales_date`
- `amount`

Las garantías por la venta de productos duran dos años a partir de la fecha de venta. La tarea es crear una medida en un gráfico para determinar el año en que expirará cada garantía.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `sales_date`.

Cree la siguiente medida:

```
=year(sales_date+365*2)
```

Tabla de resultados

sales_date	=year(sales_date+365*2)
12/28/2020	2022
12/29/2020	2022
12/30/2020	2022
12/31/2020	2022
01/01/2021	2023
01/02/2021	2023
01/03/2021	2023
01/04/2021	2023
01/05/2021	2023

sales_date	=year(sales_date+365*2)
01/06/2021	2023
01/07/2021	2023

Los resultados de esta medida se pueden ver en la tabla anterior. Para sumar dos años a una fecha, multiplique 365 por 2 y sume el resultado a la fecha de venta. Por lo tanto, las ventas que tuvieron lugar en 2020 tienen como año de vencimiento 2022.

yearend

Esta función devuelve un valor correspondiente a una marca de tiempo del último milisegundo del último día del año que contiene a **date**. El formato de salida predeterminado será el **DateFormat** definido en el script.

Sintaxis:

```
YearEnd( date[, period_no[, first_month_of_year = 1]])
```

En otras palabras, la función `yearend()` determina en qué año cae la fecha. Luego devuelve una marca de tiempo, en formato de fecha, del último milisegundo de ese año. El primer mes del año es, por defecto, enero. No obstante, también puede cambiar cuál será el primer mes del año usando el argumento `first_month_of_year` en la función `yearend()`.

> *La función `yearend()` no considera la variable del sistema `FirstMonthOfYear`. El año comienza el 1 de enero a menos que se use el argumento `first_month_of_year` para cambiarlo.*

Diagrama de la función `yearend()`.

Cuándo se utiliza

La función `yearend()` se utiliza como parte de una expresión cuando el usuario desea que el cálculo utilice la fracción del año que aún no ha ocurrido. Por ejemplo, si desea calcular el interés total aún no devengado durante el año.

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
date	La fecha o marca de tiempo para evaluar.
period_no	**period_no** es un entero, donde el valor 0 indica el año que contiene a **date**. Los valores negativos en **period_no** indican años precedentes y los valores positivos indican años subsiguientes.
first_month_of_year	Si desea trabajar con años (fiscales) que no comiencen en enero, indique un valor entre 2 y 12 en **first_month_of_year**.

Puede utilizar los siguientes valores para establecer el primer mes del año en el argumento `first_month_of_year`:

first_month_of_year values

Month	Valor
Febrero	2
Marzo	3
Abril	4
May	5
Junio	6
Julio	7
Agosto	8
Septiembre	9
Octubre	10
Noviembre	11
Diciembre	12

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las

fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
`yearend('10/19/2001')`	Devuelve 12/31/2001 23:59:59.
`yearend('10/19/2001', -1)`	Devuelve 12/31/2000 23:59:59.
`yearend('10/19/2001', 0, 4)`	Devuelve 03/31/2002 23:59:59.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones entre 2020 y 2022 se carga en una tabla llamada "`Transactions`".
- El campo de fecha se ha proporcionado en el formato `(MM/DD/YYYY)` de la variable del sistema `DateFormat`.
- Una instrucción load precedente que contiene lo siguiente:
    - La función `yearend()` que está establecida como el campo `year_end`.
    - La función `Timestamp()` que está establecida como el campo `year_end_timestamp`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`
- `year_end`
- `year_end_timestamp`

Tabla de resultados

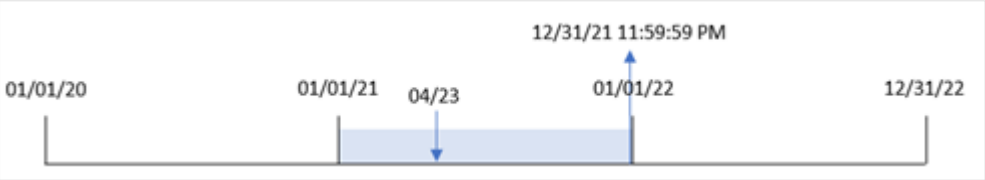
id	date	year_end	year_end_timestamp
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM
8192	05/21/2020	12/31/2020	12/31/2020 11:59:59 PM
8193	08/14/2020	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	date	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

El campo "year_end" se crea en la instrucción load anterior utilizando la función `yearend()` e introduciendo el campo de fecha como argumento de la función.

La función `yearend()` identifica inicialmente en qué año cae el valor de la fecha y devuelve una marca de tiempo del último milisegundo de ese año.

Diagrama de la función `yearend()` con la transacción 8199 seleccionada.



La transacción 8199 tuvo lugar el 23 de abril de 2021. La función `yearend()` devuelve el último milisegundo de ese año, que es el 31 de diciembre a las 23:59:59.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, la tarea es crear un campo, "previous_year_end", que devuelva la marca de tiempo de la fecha de finalización del año anterior al año en que tuvo lugar una transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- id
- date
- previous_year_end
- previous_ year_end_timestamp

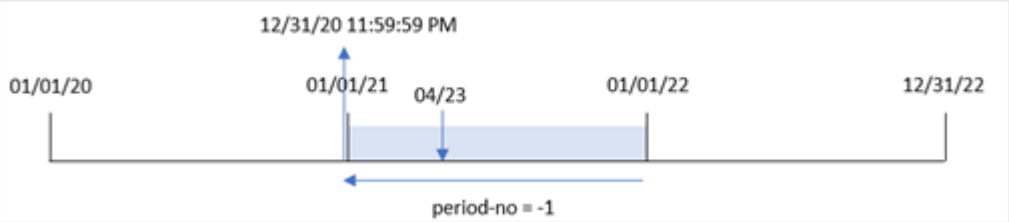
Tabla de resultados

id	date	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	date	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

Como se usó un `period_no` de `-1` como el argumento de desplazamiento en la función `yearend()`, la función identifica primero el año en que se realizan las transacciones. Luego busca un año antes e identifica el último milisegundo de ese año.

Diagrama de la función `yearend()` con un period_no de-1.



La transacción 8199 tuvo lugar el 23 de abril de 2021. La función `yearend()` devuelve el último milisegundo del año anterior, 31 de diciembre de 2020 a las 11:59:59 PM, del campo "`previous_year_end`".

Ejemplo 3: first_month_of_year

Script de carga y resultados

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, la política de la empresa es que el año comience a partir del 1 de abril.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- id
- date
- year_end
- year_end_timestamp

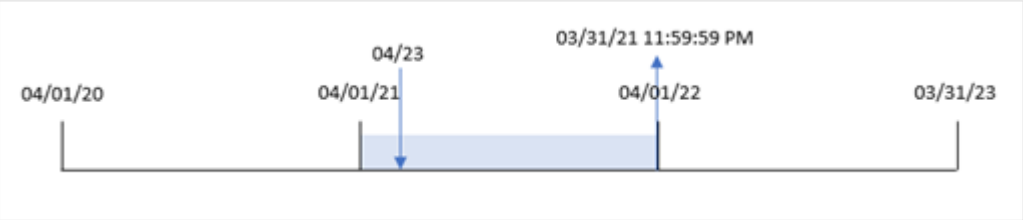
Tabla de resultados

id	date	year_end	year_end_timestamp
8188	01/13/2020	03/31/2020	3/31/2020 11:59:59 PM

id	date	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

Como el argumento `first_month_of_year` de `4` se utiliza en la función `yearend()`, establece el primer día del año en el 1 de abril y el último día del año en el 31 de marzo.

Diagrama de la función `yearend()` con abril como primer mes del año.



La transacción 8199 tuvo lugar el 23 de abril de 2021. Como la función `yearend()` establece el inicio del año en el 1 de abril, devuelve el 31 de marzo de 2022 como el valor "`year_end`" de la transacción.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que devuelve la marca de tiempo de la fecha de finalización del año en el que se realizó una transacción se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`

Para calcular en qué año tuvo lugar una transacción, cree las siguientes medidas:

- `=yearend(date)`
- `=timestamp(yearend(date))`

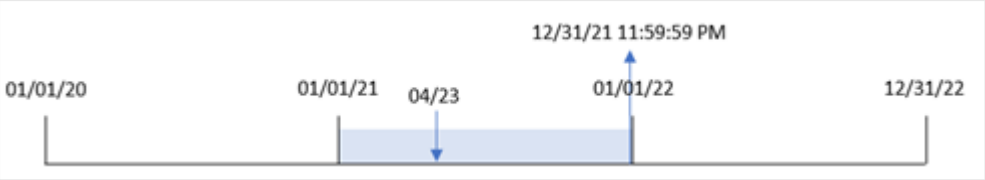
Tabla de resultados

id	date	=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

La medida "`end_of_year`" se crea en el objeto gráfico utilizando la función `yearend()` e introduciendo el campo de fecha como argumento de la función.

La función `yearend()` identifica inicialmente en qué año cae el valor de la fecha y devuelve una marca de tiempo del último milisegundo de ese año.

Diagrama de la función `yearend()` que muestra que la transacción 8199 tuvo lugar en abril.



La transacción 8199 tuvo lugar el 23 de abril de 2021. La función `yearend()` devuelve el último milisegundo de ese año, que es el 31 de diciembre a las 23:59:59.

Ejemplo 5: Escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos se carga en una tabla denominada "`Employee_Expenses`". La tabla contiene los siguientes campos:
    - employee IDs
    - employee name
    - reclamaciones de gastos diarios promedio de cada empleado

Al usuario final le gustaría tener un objeto gráfico que muestre, por ID y nombre de empleado, las reclamaciones de gastos estimados que aún deben realizarse para el resto del año. El año fiscal comienza en enero.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- employee_id
- employee_name

Para calcular las estimaciones de gastos proyectadas, cree la siguiente medida:

```
=(yearend(today(1))-today(1))*avg_daily_claim
```

Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

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

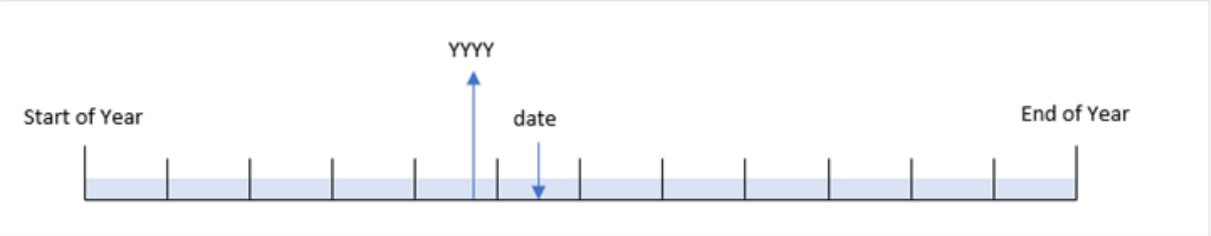
Al utilizar la fecha de hoy como único argumento, la función `yearend()` devuelve la fecha de finalización del año actual. Luego, al restar la fecha de hoy de la fecha de finalización del año, la expresión devuelve la cantidad de días que quedan en este año.

Luego, este valor se multiplica por la reclamación de gastos diarios promedio de cada empleado para calcular el valor estimado de las reclamaciones que se espera que haga cada empleado en el año restante.

yearname

Esta función devuelve un año de cuatro dígitos como valor de visualización con un valor numérico subyacente correspondiente a una marca de tiempo (fecha-hora) del primer milisegundo del primer día del año que contiene **date**.

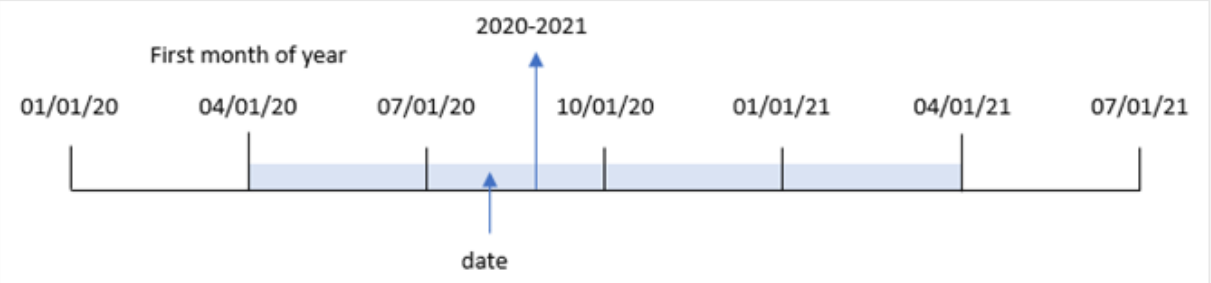
Diagrama del rango de tiempo de la función `yearname()`.



La función `yearname()` es diferente a la función `year()` ya que le permite desplazar la fecha que desea evaluar y le permite establecer el primer mes del año.

Si el primer mes del año no es enero, la función devolverá los dos años de cuatro dígitos del período de doce meses que contiene la fecha. Por ejemplo, si el inicio del año es abril y la fecha que se evalúa es el 30/06/2020, el resultado que devuelve sería 2020–2021.

Diagrama de la función `yearname()` con abril establecido como primer mes del año.



Sintaxis:

```
YearName(date[, period_no[, first_month_of_year]] )
```

Tipo de datos que devuelve: dual

Argumento	Descripción
date	La fecha o marca de tiempo para evaluar.
period_no	**period_no** es un entero, donde el valor 0 indica el año que contiene a **date**. Los valores negativos en **period_no** indican años precedentes y los valores positivos indican años subsiguientes.
first_month_of_year	Si desea trabajar con años (fiscales) que no comiencen en enero, indique un valor entre 2 y 12 en **first_month_of_year**. El valor resultante será pues una cadena que muestre dos años.

Puede utilizar los siguientes valores para establecer el primer mes del año en el argumento `first_month_of_year`:

first_month_of_year values

Month	Valor
Febrero	2
Marzo	3
Abril	4
May	5
Junio	6
Julio	7
Agosto	8
Septiembre	9
Octubre	10
Noviembre	11
Diciembre	12

Cuándo se utiliza

La función `yearname()` es útil para comparar agregaciones por año. Por ejemplo, si desea ver el total de ventas de productos por año.

Estas dimensiones se pueden crear en el script de carga utilizando la función para crear un campo en una tabla de calendario maestro. También se pueden crear en un gráfico como dimensiones calculadas

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
`yearname('10/19/2001')`	Devuelve "'2001".
`yearname('10/19/2001',-1)`	Devuelve "'2000".
`yearname('10/19/2001',0,4)`	Devuelve "'2001-2002".

Temas relacionados

Tema	Descripción
year (page 1145)	Esta función devuelve un entero que representa el año en que la expresión se interpreta como una fecha de acuerdo con la interpretación numérica estándar.

Ejemplo 1: sin argumentos adicionales

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones entre 2020 y 2022 se carga en una tabla llamada "Transactions".
- La variable de sistema DateFormat configurada como "MM/DD/YYYY".
- Un load precedente que utiliza yearname() y que está definida como el campo year_name.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- year_name

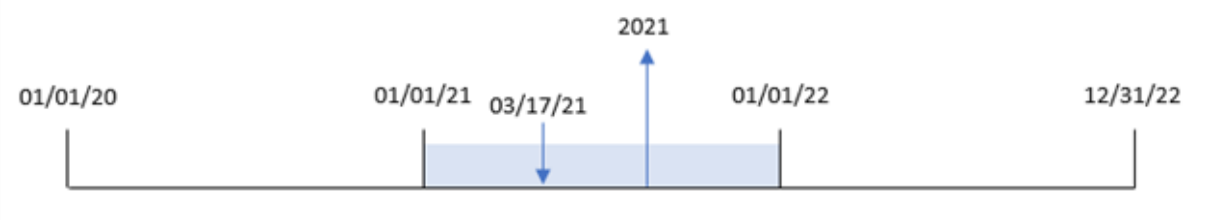
Tabla de resultados

date	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

El campo `year_name` se crea en la instrucción load anterior utilizando la función `yearname()` e introduciendo el campo de fecha como argumento de la función.

La función `yearname()` identifica en qué año cae el valor de la fecha y lo devuelve como un valor de año de cuatro dígitos.

Diagrama de la función `yearname()` que muestra 2021 como el valor del año.



Ejemplo 2: period_no

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones que se dan entre 2020 y 2022 se carga en una tabla denominada "Transactions".
- La variable de sistema DateFormat definida en el formato "`MM/DD/YYYY`".
- Un load precedente que utiliza `yearname()` y que está definida como el campo `year_name`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

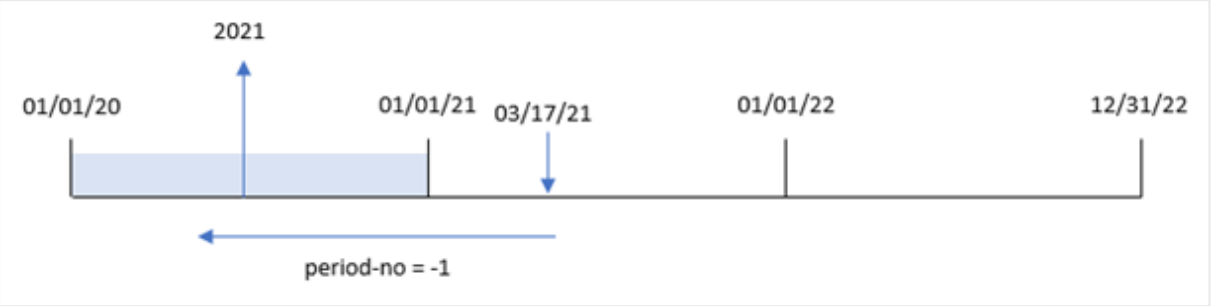
- `date`
- `prior_year_name`

Tabla de resultados

date	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

Como se usó un valor `period_no` de `-1` como argumento de desplazamiento en la función `yearname()`, la función identifica primero el año en que se realizan las transacciones. La función después se desplaza a un año antes y devuelve el año resultante.

Diagrama de la función `yearname()` con `period_no` definido en `-1`.



Ejemplo 3: first_month_of_year

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- El mismo conjunto de datos del primer ejemplo.
- La variable de sistema `DateFormat` configurada en "`MM/DD/YYYY`".
- Un load precedente que utiliza `yearname()` y que está definida como el campo `year_name`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`
- `year_name`

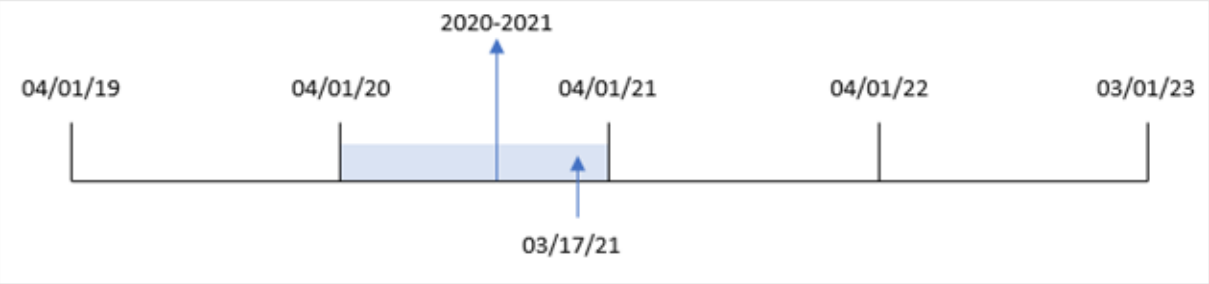
Tabla de resultados

date	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

Como el argumento `first_month_of_year` de 4 se utiliza en la función `yearname()`, el inicio del año se mueve del 1 de enero al 1 de abril. Por lo tanto, cada período de doce meses cruza dos años naturales y la función `yearname()` devuelve los dos años de cuatro dígitos para las fechas evaluadas.

La transacción 8198 tuvo lugar el 17 de marzo de 2021. La función `yearname()` establece el inicio del año el 1 de abril y el final el 30 de marzo. Por lo tanto, la transacción 8198 ocurrió en el período del año comprendido entre el 1 de abril de 2020 y el 30 de marzo de 2021. Como resultado, la función `yearname()` devuelve el valor 2020-2021.

Diagrama de la función `yearname()` con marzo fijado como el primer mes del año.



Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- El mismo conjunto de datos del primer ejemplo.
- La variable de sistema `DateFormat` configurada en "`MM/DD/YYYY`".

Sin embargo, el campo que devuelve el año en que tuvo lugar la transacción se crea como una medida en un objeto gráfico.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión:

`date`

Para calcular el campo "`year_name`", cree esta medida:

```
=yearname(date)
```

Tabla de resultados

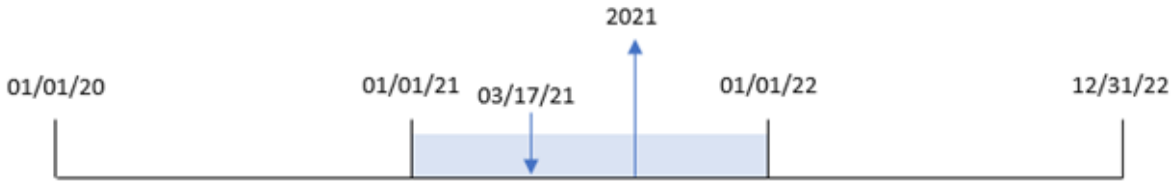
date	=yearname(date)
01/13/2020	2020
02/26/2020	2020
03/27/2020	2020
04/16/2020	2020
05/21/2020	2020
08/14/2020	2020
10/07/2020	2020
12/05/2020	2020
01/22/2021	2021
02/03/2021	2021
03/17/2021	2021
04/23/2021	2021
05/04/2021	2021
06/30/2021	2021
07/26/2021	2021

date	=yearname(date)
12/27/2021	2021
06/06/2022	2022
07/18/2022	2022
11/14/2022	2022
12/12/2022	2022

La medida "year_name" se crea en el objeto gráfico utilizando la función `yearname()` e introduciendo el campo de fecha como argumento de la función.

La función `yearname()` identifica en qué año cae el valor de la fecha y lo devuelve como un valor de año de cuatro dígitos.

Diagrama de la función `yearname()` que muestra 2021 como el valor del año.



Ejemplo 5: Escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- El mismo conjunto de datos del primer ejemplo.
- La variable de sistema `DateFormat` configurada en "MM/DD/YYYY".

Al usuario final le gustaría tener un gráfico que presente el total de ventas por trimestre de las transacciones. Utilice la función `yearname()` como una dimensión calculada para crear este gráfico cuando la dimensión `yearname()` no esté disponible en el modelo de datos.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla.

Para comparar agregaciones por año, cree esta dimensión calculada:

```
=yearname(date)
```

Cree esta medida:

```
=sum(amount)
```

Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

yearname(date)	=sum(amount)
2020	$463.55
2021	$457.69
2022	$294.35

yearstart

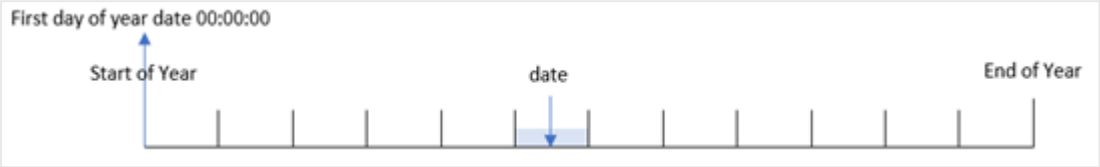
Esta función devuelve una marca de tiempo correspondiente al inicio del primer día del año que contiene a **date**. El formato de salida predefinido será el **DateFormat** definido en el script.

Sintaxis:

```
YearStart(date[, period_no[, first_month_of_year]])
```

En otras palabras, la función `yearstart()` determina en qué año cae la fecha. A continuación devuelve una marca de tiempo, en formato de fecha, del primer milisegundo de ese año. El primer mes del año es, por defecto, enero. No obstante, puede cambiar la definición del primer mes del año usando el argumento `first_month_of_year` en la función `yearstart()`.

Diagrama de la función `yearstart()` que muestra el rango de tiempo que puede abarcar la función.



Cuándo se utiliza

La función `yearstart()` se utiliza como parte de una expresión cuando deseamos que el cálculo utilice la fracción del año que ha transcurrido hasta el momento. Por ejemplo, si desea calcular el interés que se ha acumulado en un año hasta la fecha.

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
date	La fecha o marca de tiempo para evaluar.
period_no	**period_no** es un entero, donde el valor 0 indica el año que contiene a **date**. Los valores negativos en **period_no** indican años precedentes y los valores positivos indican años subsiguientes.
first_month_of_year	Si desea trabajar con años (fiscales) que no comiencen en enero, indique un valor entre 2 y 12 en **first_month_of_year**.

Se pueden utilizar los siguientes meses en el `first_month_of_year argument`:

first_month_of_year values

Month	Valor
Febrero	2
Marzo	3
Abril	4
May	5
Junio	6
Julio	7
Agosto	8
Septiembre	9

Month	Valor
Octubre	10
Noviembre	11
Diciembre	12

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplos de funciones

Ejemplo	Resultado
`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.

Ejemplo 1: ejemplo básico

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones entre 2020 y 2022 se carga en una tabla llamada "`Transactions`".
- El campo de fecha se ha proporcionado en el formato de la variable del sistema `DateFormat` (`MM/DD/YYYY`).
- Una instrucción load precedente que contiene lo siguiente:

- La función `yearstart()` que está establecida como el campo `year_start`.
- La función `Timestamp()` que está definida como el campo `year_start_timestamp`.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`
- `year_start`
- `year_start_timestamp`

Tabla de resultados

id	date	year_start	year_start_timestamp
8188	01/13/2020	01/01/2020	1/1/2020 12:00:00 AM
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	01/01/2021	1/1/2021 12:00:00 AM
8197	02/03/2021	01/01/2021	1/1/2021 12:00:00 AM
8198	03/17/2021	01/01/2021	1/1/2021 12:00:00 AM
8199	04/23/2021	01/01/2021	1/1/2021 12:00:00 AM
8200	05/04/2021	01/01/2021	1/1/2021 12:00:00 AM
8201	06/30/2021	01/01/2021	1/1/2021 12:00:00 AM
8202	07/26/2021	01/01/2021	1/1/2021 12:00:00 AM
8203	12/27/2021	01/01/2021	1/1/2021 12:00:00 AM
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

El campo `year_start` se crea en la instrucción load anterior utilizando la función `yearstart()` e introduciendo el campo de fecha como argumento de la función.

La función `yearstart()` identifica inicialmente en qué año cae el valor de la fecha y devuelve una marca de tiempo del primer milisegundo de ese año.

Diagrama de la función `yearstart()` y la transacción 8199.

La transacción 8199 tuvo lugar el 23 de abril de 2021. La función `yearstart()` devuelve el primer milisegundo de ese año, que es el 1 de enero a las 12:00:00 AM.

Ejemplo 2: period_no

Script de carga y resultados

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, la tarea es crear un campo, "`previous_year_start`", que devuelva la marca de tiempo de la fecha de inicio del año anterior al año en que tuvo lugar una transacción.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

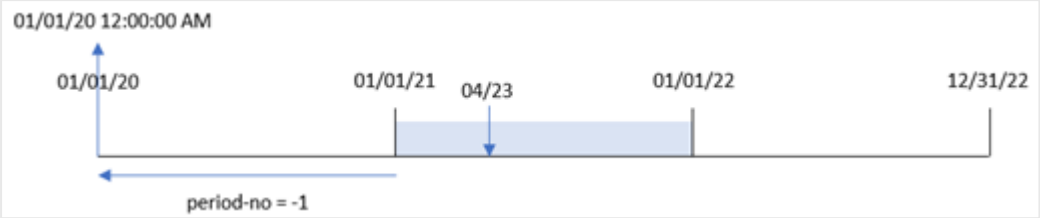
- `id`
- `date`
- `previous_year_start`
- `previous_ year_start_timestamp`

Tabla de resultados

id	date	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	01/01/2021	1/1/2021 12:00:00 AM
8205	07/18/2022	01/01/2021	1/1/2021 12:00:00 AM
8206	11/14/2022	01/01/2021	1/1/2021 12:00:00 AM
8207	12/12/2022	01/01/2021	1/1/2021 12:00:00 AM

En este caso, como se usó un `period_no` de `-1` como argumento de desplazamiento en la función `yearstart ()`, la función identifica primero el año en que se realizan las transacciones. Luego busca un año antes e identifica el primer milisegundo de ese año.

Diagrama de la función `yearstart()` con un desplazamiento de `period_no` de -1.



La transacción 8199 tuvo lugar el 23 de abril de 2021. La función `yearstart()` devuelve el primer milisegundo del año anterior, 1 de enero de 2020 a las 12:00:00 AM, para el campo "`previous_year_start`".

Ejemplo 3: first_month_of_year

Script de carga y resultados

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, la política de la empresa es que el año comience a partir del 1 de abril.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

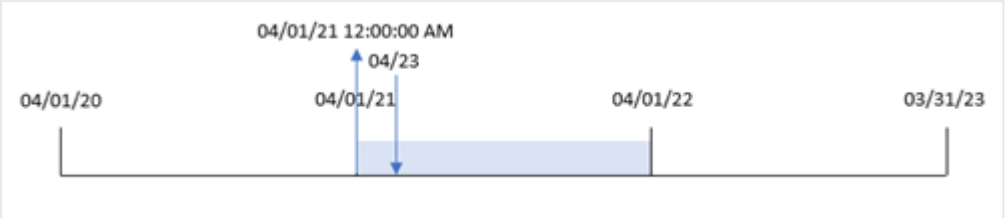
- `id`
- `date`
- `year_start`
- `year_start_timestamp`

Tabla de resultados

id	date	year_start	year_start_timestamp
8188	01/13/2020	04/01/2019	4/1/2019 12:00:00 AM
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	4/1/2021 12:00:00 AM
8200	05/04/2021	04/01/2021	4/1/2021 12:00:00 AM
8201	06/30/2021	04/01/2021	4/1/2021 12:00:00 AM
8202	07/26/2021	04/01/2021	4/1/2021 12:00:00 AM
8203	12/27/2021	04/01/2021	4/1/2021 12:00:00 AM
8204	06/06/2022	04/01/2022	4/1/2022 12:00:00 AM
8205	07/18/2022	04/01/2022	4/1/2022 12:00:00 AM
8206	11/14/2022	04/01/2022	4/1/2022 12:00:00 AM
8207	12/12/2022	04/01/2022	4/1/2022 12:00:00 AM

En este caso, como el argumento `first_month_of_year` de `4` se utiliza en la función `yearstart()`, establece el primer día del año en el 1 de abril y el último día del año en el 31 de marzo.

Diagrama de la función `yearstart()` con abril establecido como primer mes del año.



La transacción 8199 tuvo lugar el 23 de abril de 2021. Dado que la función `yearstart()` establece el inicio del año en el 1 de abril y lo devuelve como el valor "`year_start`" de la transacción.

Ejemplo 4: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Se utilizan el mismo conjunto de datos y el mismo escenario que en el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que devuelve la marca de tiempo de la fecha de inicio del año en el que se realizó una transacción se crea como una medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `id`
- `date`

Para calcular en qué año tuvo lugar una transacción, cree las siguientes medidas:

- `=yearstart(date)`
- `=timestamp(yearstart(date))`

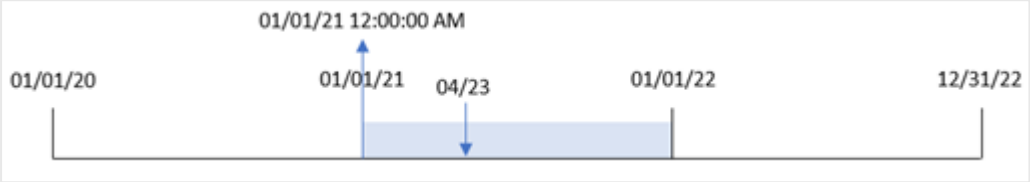
Tabla de resultados

id	date	=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	01/01/2021	1/1/2021 12:00:00 AM
8193	02/03/2021	01/01/2021	1/1/2021 12:00:00 AM
8194	03/17/2021	01/01/2021	1/1/2021 12:00:00 AM
8195	04/23/2021	01/01/2021	1/1/2021 12:00:00 AM
8196	05/04/2021	01/01/2021	1/1/2021 12:00:00 AM
8197	06/30/2021	01/01/2021	1/1/2021 12:00:00 AM
8198	07/26/2021	01/01/2021	1/1/2021 12:00:00 AM
8199	12/27/2021	01/01/2021	1/1/2021 12:00:00 AM
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

La medida "start_of_year" se crea en el objeto gráfico utilizando la función yearstart() e introduciendo el campo de fecha como argumento de la función.

La función yearstart() identifica inicialmente en qué año cae el valor de la fecha y devuelve una marca de tiempo del primer milisegundo de ese año.

Diagrama de la función yearstart() y la transacción 8199.



La transacción 8199 tuvo lugar el 23 de abril de 2021. La función yearstart() devuelve el primer milisegundo de ese año, que es el 1 de enero a las 12:00:00 AM.

Ejemplo 5: Escenario

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos se carga en una tabla denominada "Loans". La tabla contiene los siguientes campos:
    - Loan IDs.
    - El saldo al inicio del año.
    - La tasa de interés simple cobrada en cada préstamo por año.

Al usuario final le gustaría tener un objeto gráfico que muestre, por ID de préstamo, el interés actual que se ha acumulado en cada préstamo en el año hasta la fecha.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `loan_id`
- `start_balance`

Para calcular el interés acumulado, cree la siguiente medida:

`=start_balance*(rate*(today(1)-yearstart(today(1)))/365)`
Establezca el **Formato numérico** de la medida en **Moneda**.

Tabla de resultados

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

La función `yearstart()`, utilizando la fecha de hoy como único argumento, devuelve la fecha de inicio del año actual. Al restar ese resultado de la fecha actual, la expresión devuelve la cantidad de días que han transcurrido en lo que va del año.

Después este valor se multiplica por la tasa de interés y se divide por 365 para obtener la tasa de interés efectiva del período. Luego, la tasa de interés efectiva del período se multiplica por el saldo inicial del préstamo para obtener los intereses que se han acumulado en lo que va del año.

yeartodate

Esta función encuentra si la marca de tiempo (una fecha-hora) de entrada se encuentra dentro del año de la fecha en que se cargó el script por última vez, y devuelve True si lo hace o False si no lo hace.

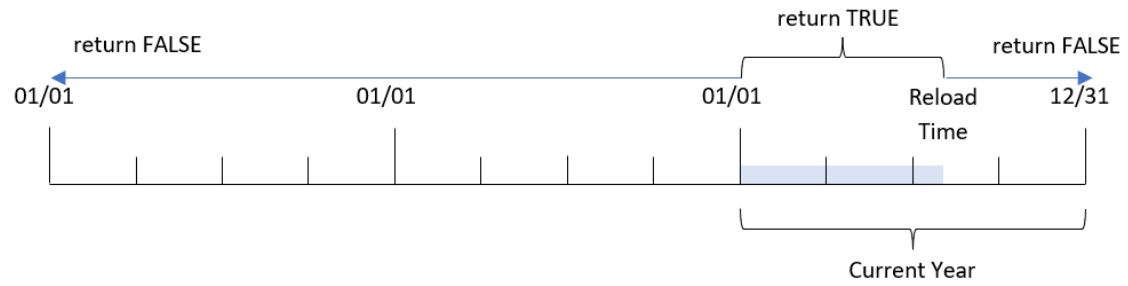
Sintaxis:

```
YearToDate(timestamp[ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Tipo de datos que devuelve: Booleano

> *En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.*

Ejemplo de diagrama de la función `yeartodate()`



Si no se utiliza ninguno de los parámetros opcionales, el año hasta la fecha implica cualquier fecha dentro de un año natural, desde el 1 de enero hasta e incluido el día de la última ejecución del script.

En otras palabras, la función `yeartodate()`, cuando se activa sin parámetros adicionales, se usa para evaluar una marca de tiempo y devolver un resultado booleano en función de si la fecha ocurrió dentro del año natural hasta la fecha en que se realizó la recarga, inclusive.

Sin embargo, también es posible reemplazar la fecha de inicio del año usando el argumento de `firstmonth`, así como hacer comparaciones con años anteriores o futuros usando el argumento de `yearoffset`.

Por último, en los casos de conjuntos de datos históricos, la función `yeartodate()` proporciona un parámetro para definir `todaydate`, que en su lugar comparará la marca de tiempo con el año natural hasta la fecha proporcionada en el argumento de `todaydate`, inclusive esta.

Argumentos

Argumento	Descripción
timestamp	La marca de tiempo que se ha de evaluar, por ejemplo "12/10/2012".
yearoffset	Especificando un **yearoffset**, **yeartodate** devuelve True para el mismo período de otro año. Un **yearoffset** negativo indica un año anterior; un desplazamiento positivo, indica un año futuro. El año más reciente hasta la fecha se logra especificando yearoffset = -1. Si se omite, se presupone 0.
firstmonth	Especificando un **firstmonth** entre 1 y 12 (1 si se omite), el comienzo del año se puede adelantar al primer día de cualquier mes. Por ejemplo, si desea trabajar con un año fiscal que comience el 1 de mayo, especifique **firstmonth** = 5. Un valor de 1 indicaría un año fiscal que comienza el 1 de enero y un valor de 12 indicaría un año fiscal que comienza el 1 de diciembre.
todaydate	Especificando una **todaydate** (o una marca de tiempo de la última ejecución de script, si se omite) es posible mover el día utilizado como límite superior del período.

Cuándo se utiliza

La función `yeartodate()` devuelve un resultado booleano. Normalmente, este tipo de función se utilizará como condición en una expresión condicional. Esto devolvería una agregación o un cálculo dependiendo de si la fecha evaluada ocurrió en el año hasta e incluida la última fecha de recarga de la aplicación.

Por ejemplo, la función YearToDate() se puede utilizar para identificar todos los equipos fabricados hasta el momento en el año actual.

Los siguientes ejemplos asumen la fecha de la última recarga = 18/11/2011.

Ejemplos de funciones

Ejemplo	Resultado
`yeartodate( '11/18/2010')`	devuelve False
`yeartodate( '02/01/2011')`	devuelve True
`yeartodate( '11/18/2011')`	devuelve True
`yeartodate( '11/19/2011')`	devuelve False
`yeartodate( '11/19/2011', 0, 1, '12/31/2011')`	devuelve True
`yeartodate( '11/18/2010', -1)`	devuelve True
`yeartodate( '11/18/2011', -1)`	devuelve False
`yeartodate( '04/30/2011', 0, 5)`	devuelve False
`yeartodate( '05/01/2011', 0, 5)`	devuelve True

Configuraciones regionales

A menos que se especifique algo distinto, los ejemplos de este tema utilizan el siguiente formato de fecha: MM/DD/YYYY. El formato de fecha se especifica en la sentencia `SET DateFormat` de su script de carga de datos. El formato de fecha predeterminado puede ser diferente en su sistema, debido a su configuración regional y otros factores. Puede cambiar el formato en los ejemplos a continuación para ajustarlo a sus necesidades. O puede cambiar los formatos en su script de carga para que coincidan con estos ejemplos.

La configuración regional predeterminada en las aps se basa en la configuración del sistema regional de la computadora o servidor donde esté instalado Qlik Sense. Si el servidor de Qlik Sense al que está accediendo está configurado en Suecia, el editor de carga de datos utilizará la configuración regional sueca para las fechas, la hora y la moneda. Estos ajustes de formato regional no están relacionados con el idioma mostrado en la interfaz de usuario de Qlik Sense. Qlik Sense se mostrará en el mismo idioma que esté utilizando en su navegador.

Ejemplo 1: ejemplo básico

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones entre 2020 y 2022, que se carga en una tabla llamada "Transactions".
- El campo de fecha proporcionado en el formato de la variable del sistema DateFormat: (MM/DD/AAAA).
- La creación de un campo year_to_date, que determina qué transacciones se realizaron en el año natural hasta la fecha de la última recarga.

Al momento de escribir, la fecha es el 26 de abril de 2022.

Script de carga

```
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
];
```

Resultados

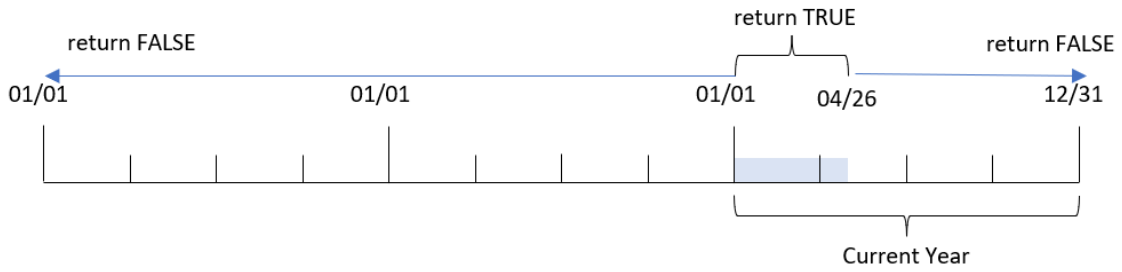
Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- date
- year_to_date

Tabla de resultados

date	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
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

Diagrama de la función `yeartodate()`, ejemplo básico



El campo `year_to_date` se crea en la instrucción load anterior utilizando la función `yeartodate()` e introduciendo el campo `date` como argumento de la función.

Debido a que no se indican más parámetros en la función, la función `yeartodate()` identifica inicialmente la fecha de recarga y, por lo tanto, los límites del año natural actual (comenzando el 1 de enero) que devolverán un resultado booleano de TRUE.

Por lo tanto, cualquier transacción que ocurra entre el 1 de enero y el 26 de abril, la fecha de recarga, devolverá un resultado booleano de TRUE. Cualquier transacción que ocurra antes del comienzo de 2022 devolverá un resultado booleano de FALSE.

Ejemplo 2: yearoffset

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `two_years_prior`, que determina qué transacciones se realizaron dos años completos antes del año natural hasta la fecha.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

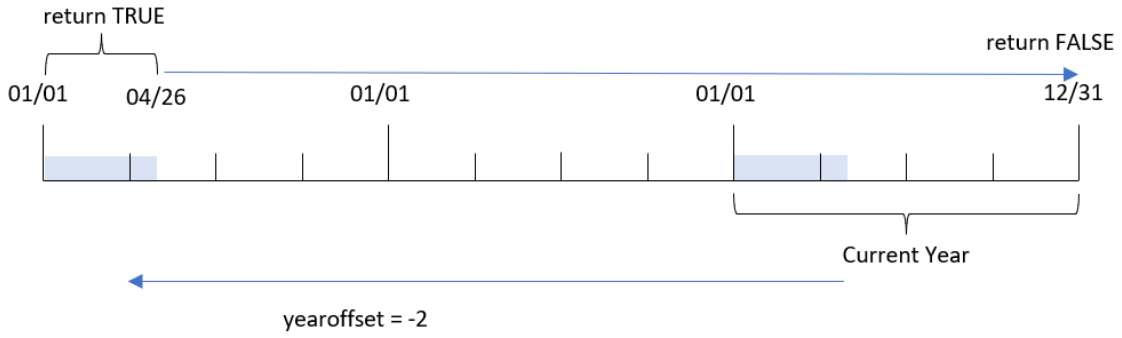
- `date`
- `two_years_prior`

Tabla de resultados

date	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

Utilizando -2 como el argumento de `yearoffset` en la función `yeartodate()`, la función desplaza los límites del segmento del año natural de comparación en dos años completos. Inicialmente, el segmento de año equivale a entre el 1 de enero y el 26 de abril de 2022. Luego, el argumento de `yearoffset` desplaza este segmento a dos años antes. Los límites de fecha caerán entre el 1 de enero y el 26 de abril de 2020.

Diagrama de la función `yeartodate()`, ejemplo de yearoffset



Por lo tanto, cualquier transacción que ocurra entre el 1 de enero y el 26 de abril de 2020, devolverá un resultado booleano de TRUE. Cualquier transacción que aparezca antes o después de este segmento devolverá FALSE.

Ejemplo 3: firstmonth

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `year_to_date`, que determina qué transacciones se realizaron en el año natural hasta la fecha de la última recarga.

En este ejemplo, establecemos el inicio del año fiscal en el 1 de julio.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `year_to_date`

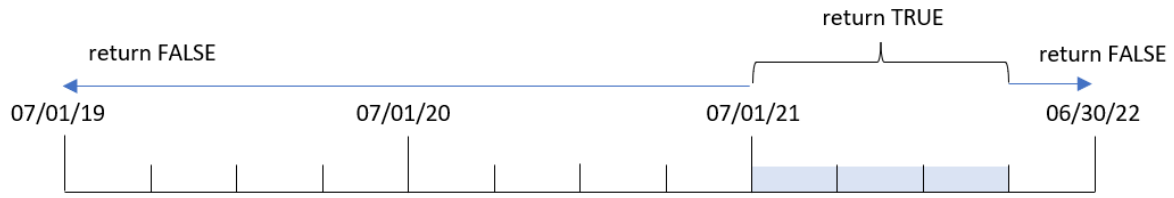
Tabla de resultados

date	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

date	year_to_date
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	-1
12/27/2021	-1
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

En este caso, como se utiliza el argumento de 7 como `firstmonth` en la función `yeartodate()`, se establece el primer día del año en el 1 de julio y el último día del año en el 30 de junio.

Diagrama de la función `yeartodate()`, ejemplo de firstmonth



Por lo tanto, cualquier transacción que ocurra entre el 1 de julio de 2021 y el 26 de abril de 2022, la fecha de recarga, devolverá un resultado booleano de TRUE. Cualquier transacción que ocurra antes del 1 de julio de 2021 devolverá un resultado booleano de FALSE.

Ejemplo 4: todaydate

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Se utilizan el mismo conjunto de datos y escenario que en el primer ejemplo.
- La creación de un campo, `year_to_date`, que determina qué transacciones se realizaron en el año natural hasta la fecha de la última recarga.

Sin embargo, en este ejemplo, necesitamos identificar todas las transacciones que tuvieron lugar en el año natural hasta e incluido el 1 de marzo de 2022.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `date`
- `year_to_date`

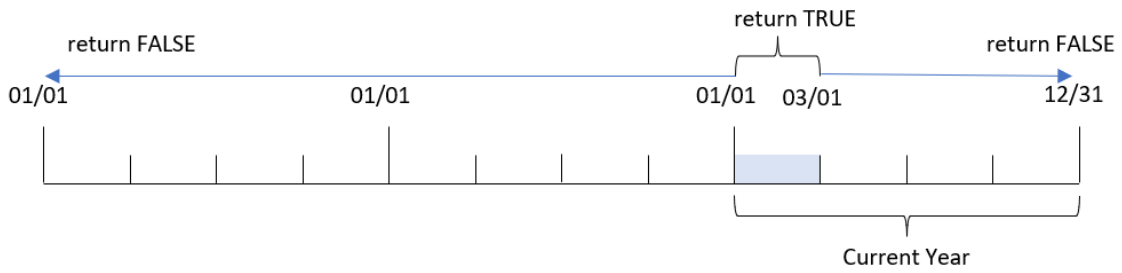
Tabla de resultados

date	year_to_date
01/10/2020	0
02/28/2020	0

date	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	0
03/11/2022	0

En este caso, como el argumento de `todaydate` 01/03/2022 se usa en la función `yeartodate()`, se establece el límite final del segmento del año natural de comparación en el 1 de marzo de 2022. Es fundamental proporcionar el parámetro de `firstmonth` (entre 1 y 12); de lo contrario, la función devolverá resultados nulos.

Diagrama de la función `yeartodate()`, ejemplo utilizando el argumento de todaydate.



Por lo tanto, cualquier transacción que ocurra entre el 1 de enero de 2022 y el 1 de marzo de 2022, el parámetro de `todaydate`, devolverá un resultado booleano de TRUE. Cualquier transacción que ocurra antes del 1 de enero de 2022 o después del 1 de marzo de 2022 devolverá un resultado booleano de FALSE.

Ejemplo 5: ejemplo de objeto gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación en una nueva pestaña.

El script de carga contiene el mismo conjunto de datos y escenario que el primer ejemplo.

Sin embargo, en este ejemplo, el conjunto de datos sin modificar se carga en la aplicación. El cálculo que determina qué transacciones tuvieron lugar en el año natural hasta la fecha de la última recarga se crea como medida en un objeto gráfico de la aplicación.

Script de carga

```
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
];
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue este campo como dimensión: `date`.

Agregue la siguiente medida:

```
=yeartodate(date)
```

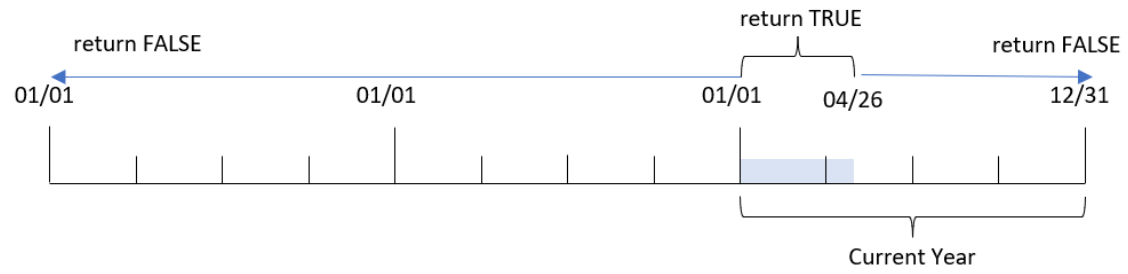
Tabla de resultados

date	=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

La medida `year_to_date` se crea en el objeto gráfico utilizando la función `yeartodate()` e indicando el campo `date` como argumento de la función.

Debido a que no se indican más parámetros en la función, la función `yeartodate()` identifica inicialmente la fecha de recarga y, por lo tanto, los límites del año natural actual (a partir del 1 de enero) que devolverán un resultado booleano de TRUE.

Diagrama de la función `yeartodate()`, ejemplo usando un objeto gráfico



Cualquier transacción que ocurra entre el 1 de enero y el 26 de abril, la fecha de recarga, devolverá un resultado booleano de TRUE. Cualquier transacción que ocurra antes del comienzo de 2022 devolverá un resultado booleano de FALSE.

Ejemplo 6: Escenario

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Un conjunto de datos que contiene un conjunto de transacciones entre 2020 y 2022, que se carga en una tabla llamada `Transactions`.
- El campo de fecha proporcionado en el formato de la variable del sistema `DateFormat` (MM/DD/AAAA).

El usuario final desea un objeto KPI que presente el total de ventas para el período equivalente en 2021 como el año actual a la fecha de la última recarga.

Al momento de escribir, la fecha es el 16 de junio de 2022.

Script de carga

```
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
```

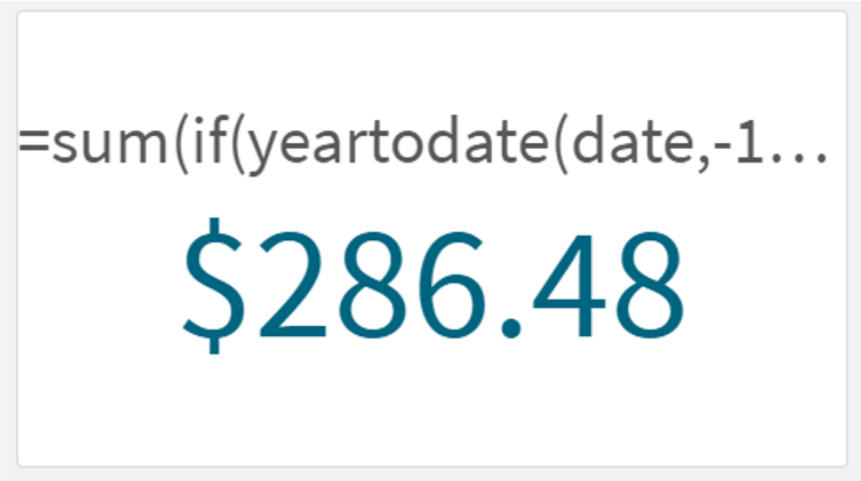
```
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Resultados

Haga lo siguiente:

1. Cree un objeto KPI.
2. Cree la siguiente medida de agregación para calcular el total de ventas:
   `=sum(if(yeartodate(date,-1),amount,0))`
3. Establezca el **Formato numérico** de la medida en **Moneda**.

Gráfico de KPI `yeartodate()` para 2021



La función `yeartodate()` devuelve un valor booleano al evaluar las fechas de cada ID de transacción. Como la recarga tuvo lugar el 16 de junio de 2022, la función `yeartodate` segmenta el período del año entre el 01/01/2022 y el 16/06/2022. Sin embargo, dado que se utilizó un valor `period_no` de -1 en la función, estos límites se desplazan al año anterior. Por lo tanto, para cualquier transacción que ocurra entre el 01/01/2021 y el 16/06/2021, la función `yeartodate()` devuelve un valor booleano de TRUE y suma la cantidad.

8.8 Funciones exponenciales y logarítmicas

En esta sección se describen funciones relacionadas con los cálculos exponenciales y logarítmicos. Todas las funciones pueden utilizarse tanto en el script de carga de datos como en las expresiones de gráficos.

En las funciones a continuación, los parámetros son expresiones donde **x** y **y** deben interpretarse como números con valores reales.

exp
La función exponencial natural, e^x, usando el logaritmo natural **e** como base. El resultado es un número positivo.

```
exp(x )
```

Ejemplos y resultados:

`exp(3)` devuelve 20.085.

log
El logaritmo natural de **x**. La función solo se define si **x**> 0. El resultado es un número.

```
log(x )
```

Ejemplos y resultados:

`log(3)` devuelve 1.0986

log10
El logaritmo común (base 10) de **x**. La función solo se define si **x**> 0. El resultado es un número.

```
log10(x )
```

Ejemplos y resultados:

`log10(3)` devuelve 0.4771

pow
Devuelve **x** a la potencia de **y**. El resultado es un número.

```
pow(x,y )
```

Ejemplos y resultados:

`pow(3, 3)` devuelve 27

sqr
x al cuadrado (**x** a la potencia de 2). El resultado es un número.

```
sqr (x )
```

Ejemplos y resultados:

`sqr(3)` devuelve 9

sqrt
Raíz cuadrada de **x**. La función solo se define si **x** >= 0. El resultado es un número positivo.

```
sqrt(x )
```

Ejemplos y resultados:

`sqrt(3)` devuelve 1.732

8.9 Funciones de campo

Estas funciones solo pueden emplearse en expresiones de gráficos.

Las funciones de campo devuelven enteros o cadenas que identifican diferentes aspectos de las selecciones de campo.

Funciones de contador

GetAlternativeCount
GetAlternativeCount()sirve para hallar el número de valores alternativos (de color gris claro) en el campo identificado.

```
GetAlternativeCount - función de gráfico (field_name)
```

GetExcludedCount
GetExcludedCount() halla el número de valores excluidos distintos en el campo identificado. Los valores excluidos incluyen campos alternativos (en gris claro), excluidos (en gris oscuro) y seleccionados excluidos (en gris oscuro con marca de verificación).

```
GetExcludedCount - función de gráfico (page 1209)(field_name)
```

GetNotSelectedCount
Esta función de gráfico devuelve el número de valores no seleccionados en el campo denominado **fieldname**. El campo deberá estar en modo And para que esta función surta efecto.

```
GetNotSelectedCount - función de gráfico(fieldname [, includeexcluded=false])
```

GetPossibleCount
GetPossibleCount() sirve para hallar el número de valores posibles en el campo identificado. Si el campo identificado incluye selecciones, los campos seleccionados (de color verde) se cuentan. De lo contrario, se cuentan los valores asociados (de color blanco).

```
GetPossibleCount - función de gráfico(field_name)
```

GetSelectedCount

GetSelectedCount() halla el número de valores seleccionados (de color verde) en un campo.

```
GetSelectedCount - función de gráfico (field_name [, include_excluded])
```

Funciones de campo y selección

GetCurrentSelections

GetCurrentSelections() devuelve una lista de las selecciones actuales en la app. Si en vez de ello las selecciones se hacen por medio de una cadena de búsqueda en un cuadro de búsqueda, **GetCurrentSelections()** devuelve la cadena de búsqueda.

```
GetCurrentSelections - función de gráfico([record_sep [,tag_sep [,value_sep
[,max_values]]]])
```

GetFieldSelections

GetFieldSelections() devuelve una **cadena** con las selecciones actuales en un campo.

```
GetFieldSelections - función de gráfico ( field_name [, value_sep [, max_
values]])
```

GetObjectDimension

GetObjectDimension() devuelve el nombre de la dimensión. **Index** es un entero opcional que denota la dimensión que debe devolverse.

```
GetObjectDimension - función de gráfico ([index])
```

GetObjectField

GetObjectField() devuelve el nombre de la dimensión. **Index** es un entero opcional que indica la dimensión que debe devolverse.

```
GetObjectField - función de gráfico ([index])
```

GetObjectMeasure

GetObjectMeasure() devuelve el nombre de la medida. **Index** es un entero opcional que indica la medida que debe devolverse.

```
GetObjectMeasure - función de gráfico ([index])
```

GetAlternativeCount - función de gráfico

GetAlternativeCount()sirve para hallar el número de valores alternativos (de color gris claro) en el campo identificado.

Sintaxis:

```
GetAlternativeCount (field_name)
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
field_name	El campo que contiene el rango de datos que se han de medir.

Ejemplos y resultados:

El ejemplo siguiente utiliza el campo **First name** cargado en un panel de filtrado.

Ejemplos y resultados

Ejemplos	Resultados
Dado que **John** está seleccionado en **First name**. `GetAlternativeCount ([First name])`	4 puesto que hay 4 valores únicos y excluidos (en gris) en **First name**.
Dado que **John** y **Peter** están seleccionados. `GetAlternativeCount ([First name])`	3 puesto que hay 3 valores únicos y excluidos (en gris) en **First name**.
Dado que no hay valores seleccionados en **First name**. `GetAlternativeCount ([First name])`	0 puesto que no hay selecciones.

Datos utilizados en el ejemplo:

```
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 '|');
```

GetCurrentSelections - función de gráfico

GetCurrentSelections() devuelve una lista de las selecciones actuales en la app. Si en vez de ello las selecciones se hacen por medio de una cadena de búsqueda en un cuadro de búsqueda, **GetCurrentSelections()** devuelve la cadena de búsqueda.

Si se utilizan opciones, deberá especificar record_sep. Para especificar una nueva línea, configure **record_sep** en **chr(13)&chr(10)**.

Si se seleccionan todos los valores menos dos, o todos menos uno, se utilizará el formato "NOT x,y" o "NOT y" respectivamente. Si selecciona todos los valores y el recuento de todos los valores es mayor que max_values, devolverá el texto ALL.

Sintaxis:

```
GetCurrentSelections ([record_sep [, tag_sep [, value_sep [, max_values [, state_name]]]]])
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumentos	Descripción
record_sep	Es el separador que se ha de colocar entre los registros de campo. Por defecto es <CR><LF>, que significa una nueva línea.
tag_sep	Es el separador que se ha de colocar entre la pestaña del nombre de campo y los valores de campo. El valor predeterminado es ': '.
value_sep	Es el separador que se ha de colocar entre valores de campo. Por defecto es ', '.
max_values	Es el número máximo de valores de campo que se han de listar de manera individual. Cuando se seleccione un número mayor de valores se utilizará el formato 'valores x de y' en su lugar. El valor por defecto es 6.
state_name	El nombre de un estado alterno que se ha elegido específicamente para la visualización. Si se utiliza el argumento **state_name**, solo se tienen en cuenta las selecciones asociadas con el nombre de estado especificado.

Ejemplos y resultados:

El ejemplo siguiente utiliza dos campos cargados en diferentes paneles de filtrado, uno para el nombre **First name** y otro para las iniciales **Initials**.

Ejemplos y resultados

Ejemplos	Resultados
Dado que **John** está seleccionado en **First name**. `GetCurrentSelections ()`	'First name: John'
Dado que **John** y **Peter** están seleccionados en **First name**. `GetCurrentSelections ()`	'First name: John, Peter'
Dado que **John** y **Peter** están seleccionados en **First name** y **JA** está seleccionado en **Initials**. `GetCurrentSelections ()`	'First name: John, Peter Initials: JA'

Ejemplos	Resultados
Dado que **John** está seleccionado en **First name** y **JA** está seleccionado en **Initials**. `GetCurrentSelections ( chr(13)&chr(10) , ' = ' )`	'First name = John Initials = JA'
Dado que ha seleccionado todos los nombres excepto Sue en **First name** y no ha hecho **selecciones** en Initials. `GetCurrentSelections (chr(13)&chr(10), '=', ',' ,3)`	'First name=NOT Sue'

Datos utilizados en el ejemplo:

```
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 '|');
```

GetExcludedCount - función de gráfico

GetExcludedCount() halla el número de valores excluidos distintos en el campo identificado. Los valores excluidos incluyen campos alternativos (en gris claro), excluidos (en gris oscuro) y seleccionados excluidos (en gris oscuro con marca de verificación).

Sintaxis:

```
GetExcludedCount (field_name)
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumentos	Descripción
field_name	El campo que contiene el rango de datos que se han de medir.

Ejemplos y resultados:

El ejemplo siguiente utiliza tres campos cargados en diferentes paneles de filtrado, uno para **First name**, otro para **Last name** y otro para **Initials**.

Ejemplos y resultados

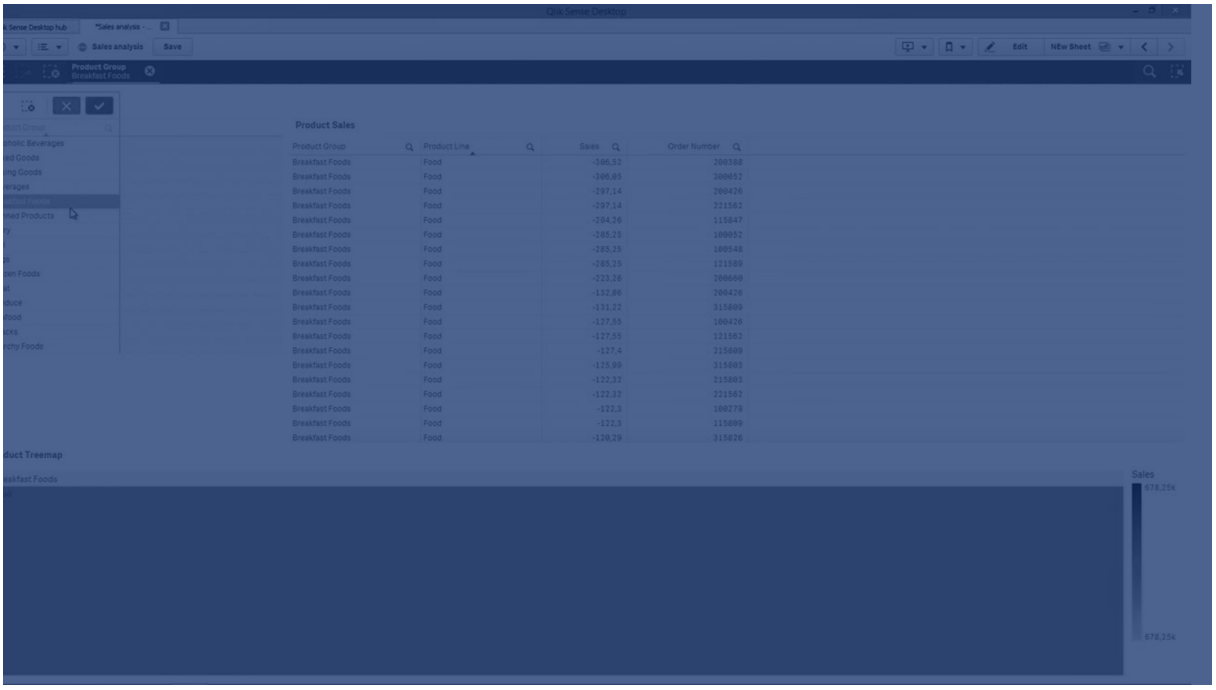
Ejemplos	Resultados
Si no hay valores seleccionados en **First name**.	`GetExcludedCount (Initials) = 0` No hay selecciones.
Si **John** está seleccionado en **First name**.	`GetExcludedCount (Initials) = 5` Hay 5 valores excluidos en **Iniciales** de un color gris oscuro. La sexta celda (JA) será blanca, ya que está asociada con la selección John en **First name**.
Si **John** y **Peter** están seleccionados.	`GetExcludedCount (Initials) = 3` John se asocia con el 1 valor y Peter está asociado con 2 valores, en **Initials**.
Si **John** y **Peter** están seleccionados en **First name**, y después **Franc** está seleccionado en **Last name**.	`GetExcludedCount ([First name]) = 4` Hay 4 valores excluidos en **First name** de un color gris oscuro. **GetExcludedCount()** evalúa campos con valores excluidos, incluyendo campos alternativos y seleccionados excluidos.
Si **John** y **Peter** están seleccionados en **First name**, y después **Franc** y **Anderson** están seleccionados en **Last name**.	`GetExcludedCount (Initials) = 4` Hay 4 valores excluidos en **Initials** de un color gris oscuro. Las otras dos celdas (JA y PF) serán de color blanco pues están asociadas con las selecciones John y Peter en **First name**.
Si **John** y **Peter** están seleccionados en **First name**, y después **Franc** y **Anderson** están seleccionados en **Last name**.	`GetExcludedCount ([Last name]) = 4` Hay 4 valores excluidos en **Initials**. Devonshire es de color gris claro, mientras que Brown, Carr y Elliot son de color gris oscuro.

Datos utilizados en el ejemplo:

```
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 '|');
```

GetFieldSelections - función de gráfico

GetFieldSelections() devuelve una **cadena** con las selecciones actuales en un campo.



Si se seleccionan todos los valores menos dos, o todos menos uno, se utilizará el formato "NOT x,y" o "NOT y" respectivamente. Si selecciona todos los valores y el recuento de todos los valores es mayor que max_values, devolverá el texto ALL.

Sintaxis:

```
GetFieldSelections ( field_name [, value_sep [, max_values [, state_name]]])
```

Tipo de datos que devuelve: cadena

Formatos de cadena de retorno

Formato	Descripción
'a, b, c'	Si el número de valores seleccionados es max_values o menos, la cadena que devuelve es una lista de los valores seleccionados. Los valores se separan con value_sep como delimitador.
'NOT a, b, c'	Si el número de valores no seleccionados es max_values o menos, la cadena que devuelve es una lista de los valores no seleccionados con el prefijo NOT. Los valores se separan con value_sep como delimitador.
'x of y'	x = el número de valores seleccionados y = el número total de valores Esto lo devuelve cuando max_values < x < (y - max_values).
'ALL'	Devuelto si se seleccionan todos los valores.
'-'	Devuelto si no se selecciona ningún valor.
<search string>	Si los ha seleccionado mediante la búsqueda, se devuelve la cadena de búsqueda.

Argumentos:

Argumentos

Argumentos	Descripción
field_name	El campo que contiene el rango de datos que se han de medir.
value_sep	Es el separador que se ha de colocar entre valores de campo. Por defecto es ', '.
max_values	Es el número máximo de valores de campo que se han de listar de manera individual. Cuando se seleccione un número mayor de valores se utilizará el formato 'valores x de y' en su lugar. El valor por defecto es 6.
state_name	El nombre de un estado alterno que se ha elegido específicamente para la visualización. Si se utiliza el argumento **state_name**, solo se tienen en cuenta las selecciones asociadas con el nombre de estado especificado.

Ejemplos y resultados:

El ejemplo siguiente utiliza el campo **First name** cargado en un panel de filtrado.

Ejemplos y resultados

Ejemplos	Resultados
Dado que **John** está seleccionado en **First name**. `GetFieldSelections ([First name])`	'John'
Dado que **John** y **Peter** están seleccionados. `GetFieldSelections ([First name])`	'John,Peter'
Dado que **John** y **Peter** están seleccionados. `GetFieldSelections ([First name],'; ')`	'John; Peter'
Dado que **John**, **Sue**, **Mark** están seleccionados en **First name**. `GetFieldSelections ([First name],';',2)`	'NOT Jane;Peter', porque el valor se afirma como el valor del argumento max_values. Si no, el resultado habría sido John; Sue; Mark.

Datos utilizados en el ejemplo:

```
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 '|');
```

GetNotSelectedCount - función de gráfico

Esta función de gráfico devuelve el número de valores no seleccionados en el campo denominado **fieldname**. El campo deberá estar en modo And para que esta función surta efecto.

Sintaxis:

```
GetNotSelectedCount(fieldname [, includeexcluded=false])
```

Argumentos:

Argumentos

Argumento	Descripción
fieldname	El nombre del campo que se ha de evaluar.
includeexcluded	Si **includeexcluded** se define como True, el recuento incluirá los valores seleccionados, que actualmente están excluidos de las selecciones en otros campos.

Ejemplos:

```
GetNotSelectedCount( Country )
```

```
GetNotSelectedCount( Country, true )
```

GetObjectDimension - función de gráfico

GetObjectDimension() devuelve el nombre de la dimensión. **Index** es un entero opcional que denota la dimensión que debe devolverse.

No puede utilizar esta función en un gráfico en las siguientes ubicaciones: título, subtítulo, pie de página, expresión de línea de referencia y expresión mínima/máxima.

No puede hacer referencia al nombre de una dimensión o medida en otro objeto utilizando el Object ID.

Sintaxis:

```
GetObjectDimension ([index])
```

Ejemplo:

```
GetObjectDimension(1)
```

Ejemplo: Expresión de gráfico

Tabla de Qlik Sense con ejemplos de la función GetObjectDimension en una expresión de gráfico

transaction_date	customer_id	transaction_quantity	=GetObjectDimension ()	=GetObjectDimension (0)	=GetObjectDimension (1)
2018/08/30	049681	13	transaction_date	transaction_date	customer_id
2018/08/30	203521	6	transaction_date	transaction_date	customer_id
2018/08/30	203521	21	transaction_date	transaction_date	customer_id

Si desea devolver el nombre de una medida, utilice la función **GetObjectMeasure** en su lugar.

GetObjectField - función de gráfico

GetObjectField() devuelve el nombre de la dimensión. **Index** es un entero opcional que indica la dimensión que debe devolverse.

No puede utilizar esta función en un gráfico en las siguientes ubicaciones: título, subtítulo, pie de página, expresión de línea de referencia y expresión mínima/máxima.

No puede hacer referencia al nombre de una dimensión o medida en otro objeto utilizando el Object ID.

Sintaxis:

```
GetObjectField ([index])
```

Ejemplo:

```
GetObjectField(1)
```

Ejemplo: Expresión de gráfico

Tabla de Qlik Sense con ejemplos de la función GetObjectField en una expresión de gráfico.

transaction_date	customer_id	transaction_quantity	=GetObjectField ()	=GetObjectField (0)	=GetObjectField (1)
2018/08/30	049681	13	transaction_date	transaction_date	customer_id
2018/08/30	203521	6	transaction_date	transaction_date	customer_id
2018/08/30	203521	21	transaction_date	transaction_date	customer_id

Si desea devolver el nombre de una medida, utilice la función **GetObjectMeasure** en su lugar.

GetObjectMeasure - función de gráfico

GetObjectMeasure() devuelve el nombre de la medida. **Index** es un entero opcional que indica la medida que debe devolverse.

No puede utilizar esta función en un gráfico en las siguientes ubicaciones: título, subtítulo, pie de página, expresión de línea de referencia y expresión mínima/máxima.

No puede hacer referencia al nombre de una dimensión o medida en otro objeto utilizando el Object ID.

Sintaxis:

```
GetObjectMeasure ([index])
```

Ejemplo:

```
GetObjectMeasure(1)
```

Ejemplo: Expresión de gráfico

Tabla de Qlik Sense con ejemplos de la función GetObjectMeasure en una expresión de gráfico

customer_id	sum (transaction_quantity)	Avg (transaction_quantity)	=GetObjectMeasure ()	=GetObjectMeasure(0)	=GetObjectMeasure(1)
49681	13	13	sum(transaction_quantity)	sum(transaction_quantity)	Avg(transaction_quantity)
203521	27	13.5	sum(transaction_quantity)	sum(transaction_quantity)	Avg(transaction_quantity)

Si desea devolver el nombre de una dimensión, utilice la función **GetObjectField** en su lugar.

GetPossibleCount - función de gráfico

GetPossibleCount() sirve para hallar el número de valores posibles en el campo identificado. Si el campo identificado incluye selecciones, los campos seleccionados (de color verde) se cuentan. De lo contrario, se cuentan los valores asociados (de color blanco).

Para campos con selecciones, **GetPossibleCount()** devuelve el número de campos seleccionados (en verde).

Tipo de datos que devuelve: Entero

Sintaxis:

```
GetPossibleCount (field_name)
```

Argumentos:

Argumentos

Argumentos	Descripción
field_name	El campo que contiene el rango de datos que se han de medir.

Ejemplos y resultados:

El ejemplo siguiente utiliza dos campos cargados en diferentes paneles de filtrado, uno para el nombre **First name** y otro para las iniciales **Initials**.

Ejemplos y resultados

Ejemplos	Resultados
Dado que **John** está seleccionado en **First name**. `GetPossibleCount ([Initials])`	1 puesto que hay 1 valor en Iniciales asociadas con la selección, **John**, en **First name**.
Dado que **John** está seleccionado en **First name**. `GetPossibleCount ([First name])`	1 puesto que hay 1 selección, **John**, en **First name**.
Dado que **Peter** está seleccionado en **First name**. `GetPossibleCount ([Initials])`	2 puesto que Peter está asociado con 2 valores en **Initials**.
Dado que no hay valores seleccionados en **First name**. `GetPossibleCount ([First name])`	5 puesto que no hay selecciones y hay 5 valores únicos en **First name**.
Dado que no hay valores seleccionados en **First name**. `GetPossibleCount ([Initials])`	6 puesto que no hay selecciones y hay 6 valores únicos en **Initials**.

Datos utilizados en el ejemplo:

```
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 '|');
```

GetSelectedCount - función de gráfico

GetSelectedCount() halla el número de valores seleccionados (de color verde) en un campo.

Sintaxis:

```
GetSelectedCount (field_name [, include_excluded [, state_name]])
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumentos	Descripción
field_name	El campo que contiene el rango de datos que se han de medir.
include_ excluded	Si se establece en **True()**, el recuento incluirá los valores seleccionados, que actualmente están excluidos por las selecciones en otros campos. Si es False o se omite, estos valores no se incluirán.
state_name	El nombre de un estado alterno que se ha elegido específicamente para la visualización. Si se utiliza el argumento **state_name**, solo se tienen en cuenta las selecciones asociadas con el nombre de estado especificado.

Ejemplos y resultados:

El siguiente ejemplo utiliza tres campos cargados en diferentes paneles de filtrado, uno para el nombre **First name**, otro para **Initials** y otro para **Has cellphone**.

Ejemplos y resultados

Ejemplos	Resultados
Dado que **John** está seleccionado en **First name**. `GetSelectedCount ([First name])`	1 puesto que hay un valor seleccionado en **First name**.
Dado que **John** está seleccionado en **First name**. `GetSelectedCount ([Initials])`	0 puesto que no hay valores seleccionados en **Initials**.
Sin selecciones en .**First name**, seleccione todos los valores de **Initials** y después seleccione el valor **Yes** en **Has cellphone**. `GetSelectedCount ([Initials], True())`	6. Aunque la selecciones con **Initials** MC y PD tienen **Has cellphone** configurado en **No**, el resultado sigue siendo 6, porque el argumento include_excluded está en True().

Datos utilizados en el ejemplo:

```
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 '|');
```

8.10 Funciones de archivo

Las funciones de archivo (solo disponibles en expresiones de script) devuelven información acerca del archivo de tabla que se está leyendo en ese momento. Estas funciones devolverán NULL para todas las fuentes de datos excepto los archivos de tabla (excepción: ConnectString

Visión global de las funciones de archivo

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Attribute
Esta función de script devuelve el valor de las meta etiquetas de diversos formatos de archivo como texto. Se admiten los siguientes formatos de archivo: MP3, WMA, WMV, PNG y JPG. Si el archivo **filename** no existe, no es un formato de archivo compatible o no contiene una metaetiqueta denominada **attributename**, devolverá NULL.

```
Attribute (filename, attributename)
```

ConnectString
La función **ConnectString()** devuelve el nombre de la conexión de datos activa para conexiones ODBC u OLE DB. La función devuelve una cadena vacía si no se ha ejecutado ninguna sentencia **connect**, o después de una sentencia **disconnect**.

```
ConnectString ()
```

FileBaseName
La función **FileBaseName** devuelve una cadena que contiene el nombre del archivo de tabla que se está leyendo en ese momento, sin ruta ni extensión.

```
FileBaseName ()
```

FileDir
La función **FileDir** devuelve una cadena que contiene la ruta al directorio del archivo de tabla que se está leyendo en ese momento.

```
FileDir ()
```

FileExtension
La función **FileExtension** devuelve una cadena que contiene la extensión del archivo de tabla que se esté leyendo en ese momento.

```
FileExtension ()
```

FileName
La función **FileName** devuelve una cadena que contiene el nombre del archivo de tabla que se esté leyendo, sin la ruta pero con la extensión.

```
FileName ()
```

FilePath
La función **FilePath** devuelve una cadena que contiene la ruta completa al archivo de tabla que se esté leyendo en ese momento.

```
FilePath ()
```

FileSize
La función **FileSize** devuelve un entero que contiene el tamaño en bytes del archivo filename o, si no se especifica ningún filename, del archivo de tabla que se esté leyendo en ese momento.

```
FileSize ()
```

FileTime
La función **FileTime** devuelve una marca de tiempo en formato UTC de la última modificación de un archivo especificado. Si no se especifica un archivo, la función devuelve una marca de tiempo en UTC de la última modificación del archivo de tabla leído actualmente.

```
FileTime ([ filename ])
```

GetFolderPath
La función **GetFolderPath** devuelve el valor de la función Microsoft Windows *SHGetFolderPath*. Esta función toma como entrada el nombre de una carpeta de Microsoft Windows y devuelve la ruta completa de la carpeta.

```
GetFolderPath ()
```

QvdCreateTime
Esta función de script devuelve la marca de tiempo del encabezado XML de un archivo QVD, si la hay, de lo contrario devuelve NULL. En la marca de tiempo, la hora se proporciona en UTC.

```
QvdCreateTime (filename)
```

QvdFieldName
Esta función de script devuelve el nombre del número de campo **fieldno** en un archivo QVD. Si el campo no existe, devuelve NULL.

```
QvdFieldName (filename , fieldno)
```

QvdNoOfFields
Esta función de script devuelve el número de campos de un archivo QVD.

```
QvdNoOfFields (filename)
```

QvdNoOfRecords
Esta función de script devuelve el número de registros que hay actualmente en un archivo QVD.

```
QvdNoOfRecords (filename)
```

QvdTableName
Esta función de script devuelve el nombre de la tabla almacenada en un archivo QVD.

```
QvdTableName (filename)
```

Attribute

Esta función de script devuelve el valor de las meta etiquetas de diversos formatos de archivo como texto. Se admiten los siguientes formatos de archivo: MP3, WMA, WMV, PNG y JPG. Si el archivo **filename** no existe, no es un formato de archivo compatible o no contiene una metaetiqueta denominada **attributename**, devolverá NULL.

Sintaxis:

```
Attribute(filename, attributename)
```

Se pueden leer un gran número de metaetiquetas. Los ejemplos de esta sección muestran qué etiquetas se pueden leer para los respectivos tipos de archivo admitidos.

*Solo puede leer metaetiquetas guardadas en el archivo de acuerdo con la especificación relevante, por ejemplo, ID2v3 para archivos MP3 o EXIF para archivos JPG, y no información meta guardada en el **Explorador de archivos de Windows**.*

Argumentos:

Argumentos

Argumento	Descripción
filename	El nombre de un archivo de medios con la ruta incluida, si fuera necesario, como una conexión a datos de carpetas. **Ejemplo: *'lib://Table Files/'*** En el modo de elaboración de scripts de legado, se admiten también los diguientes formatos de ruta: • Absoluta **Ejemplo: *c:\data*** • relativa al directorio de trabajo de la app Qlik Sense. **Ejemplo: *data***
attributename	Es el nombre de una meta etiqueta.

Los ejemplos utilizan la función **GetFolderPath** para hallar las rutas a archivos multimedia. Como **GetFolderPath** solo se admite en modo de legado, debe reemplazar las referencias a **GetFolderPath** por una ruta de conexión de datos lib:// cuando utilice esta función en modo estándar o en Qlik Sense SaaS.

Restricción de acceso al sistema de archivos (page 1527)

Example 1: Archivos MP3

Este script lee todas las posibles metaetiquetas MP3 en la carpeta *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

Este script lee todas las posibles metaetiquetas EXIF de archivos JPG en la carpeta *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: Archivos multimedia de Windows

Este script lee todas las posibles metaetiquetas WMA/WMV ASF en la carpeta *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

Este script lee todas las posibles metaetiquetas PNG en la carpeta *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

La función **ConnectString()** devuelve el nombre de la conexión de datos activa para conexiones ODBC u OLE DB. La función devuelve una cadena vacía si no se ha ejecutado ninguna sentencia **connect**, o después de una sentencia **disconnect**.

Sintaxis:

```
ConnectString()
```

Ejemplos y resultados:

Ejemplos de script

Ejemplo	Resultado

`LIB CONNECT TO 'Tutorial ODBC';`
 	
`ConnectString:`
 	
`Load ConnectString() as ConnectString AutoGenerate 1;`
	Devuelve "Tutorial ODBC" en el campo ConnectString.
	
	Este ejemplo presupone que tiene una conexión de datos disponible llamada Tutorial ODBC.

FileBaseName

La función **FileBaseName** devuelve una cadena que contiene el nombre del archivo de tabla que se está leyendo en ese momento, sin ruta ni extensión.

Sintaxis:

```
FileBaseName()
```

Ejemplos y resultados:

Ejemplos de script

Ejemplo	Resultado

`LOAD *, filebasename( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Devolverá 'abc' en el campo X en cada registro leído.

FileDir

La función **FileDir** devuelve una cadena que contiene la ruta al directorio del archivo de tabla que se está leyendo en ese momento.

Sintaxis:

```
FileDir()
```

Esta función admite solo conexiones de datos de carpetas en modo estándar.

Ejemplos y resultados:

Ejemplos de script

Ejemplo	Resultado

`Load *, filedir( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Devolverá 'C:\UserFiles' en el campo X en cada registro leído.

FileExtension

La función **FileExtension** devuelve una cadena que contiene la extensión del archivo de tabla que se esté leyendo en ese momento.

Sintaxis:

```
FileExtension()
```

Ejemplos y resultados:

Ejemplos de script

Ejemplo	Resultado

`LOAD *, FileExtension( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Devolverá 'txt' en el campo X en cada registro leído.

FileName

La función **FileName** devuelve una cadena que contiene el nombre del archivo de tabla que se esté leyendo, sin la ruta pero con la extensión.

Sintaxis:

```
FileName()
```

Ejemplos y resultados:

Ejemplos de script

Ejemplo	Resultado

`LOAD *, FileName( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Devolverá 'abc.txt' en el campo X en cada registro leído.

FilePath

La función **FilePath** devuelve una cadena que contiene la ruta completa al archivo de tabla que se esté leyendo en ese momento.

Sintaxis:

```
FilePath()
```

Esta función admite solo conexiones de datos de carpetas en modo estándar.

Ejemplos y resultados:

Ejemplos de script

Ejemplo	Resultado

`Load *, FilePath( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Devolverá 'C:\UserFiles\abc.txt' en el campo X en cada registro leído.

FileSize

La función **FileSize** devuelve un entero que contiene el tamaño en bytes del archivo filename o, si no se especifica ningún filename, del archivo de tabla que se esté leyendo en ese momento.

Sintaxis:

```
FileSize([filename])
```

Argumentos:

Argumentos

Argumento	Descripción
filename	El nombre de un archivo, si es necesario incluyendo la ruta, como una carpeta o conexión de datos a archivos web Si no especifica un nombre de archivo, se utiliza el archivo de tabla que se está leyendo en ese momento. **Ejemplo: *'lib://Table Files/'*** En el modo de elaboración de scripts de legado, se admiten también los diguientes formatos de ruta: • Absoluta **Ejemplo: *c:\data*** • relativa al directorio de trabajo de la app Qlik Sense. **Ejemplo: *data*** • Dirección URL (HTTP o FTP), que apunta a una ubicación en Internet o una intranet. **Ejemplo: *http://www.qlik.com***

Ejemplos y resultados:

Ejemplos de script

Ejemplo	Resultado
`LOAD *, FileSize( ) as X from abc.txt;`	Devolverá el tamaño del archivo especificado (abc.txt) como un número entero del campo X en cada registro leído.
`FileSize( 'lib://DataFiles/xyz.xls' )`	Devolverá el tamaño del archivo xyz.xls.

FileTime

La función **FileTime** devuelve una marca de tiempo en formato UTC de la última modificación de un archivo especificado. Si no se especifica un archivo, la función devuelve una marca de tiempo en UTC de la última modificación del archivo de tabla leído actualmente.

Sintaxis:

```
FileTime([ filename ])
```

Argumentos:

Argumentos

Argumento	Descripción
filename	El nombre de un archivo, si es necesario incluyendo la ruta, como una carpeta o conexión de datos a archivos web. **Ejemplo: *'lib://Table Files/'*** En el modo de elaboración de scripts de legado, se admiten también los diguientes formatos de ruta: • Absoluta **Ejemplo: *c:\data*** • relativa al directorio de trabajo de la app Qlik Sense. **Ejemplo: *data*** • Dirección URL (HTTP o FTP), que apunta a una ubicación en Internet o una intranet. **Ejemplo: *http://www.qlik.com***

Ejemplos y resultados:

Ejemplos de script

Ejemplo	Resultado
`LOAD *, FileTime( ) as X from abc.txt;`	Devolverá la marca de tiempo de la última modificación del archivo (abc.txt) en el campo X de cada registro leído.
`FileTime( 'xyz.xls' )`	Devolverá una marca de tiempo de la última modificación efectuada en el archivo xyz.xls.

GetFolderPath

La función **GetFolderPath** devuelve el valor de la función Microsoft Windows *SHGetFolderPath*. Esta función toma como entrada el nombre de una carpeta de Microsoft Windows y devuelve la ruta completa de la carpeta.

Esta función no es posible en modo estándar.

Sintaxis:

```
GetFolderPath(foldername)
```

Argumentos:

Argumentos

Argumento	Descripción
foldername	Nombre de la carpeta Microsoft Windows. El nombre de la carpeta no debe contener espacios. Cualquier espacio en el nombre de la carpeta que se detecte en el Windows Explorer debe eliminarse del nombre de la carpeta. Ejemplos: *MyMusic* *MyDocuments*

Ejemplos y resultados:

El objetivo de este ejemplo es obtener las rutas de las siguientes carpetas Microsoft Windows: *MyMusic*, *MyPictures* y *Windows*. Agregue el script de ejemplo a su app y recárguela.

```
LOAD
 GetFolderPath('MyMusic') as MyMusic,
 GetFolderPath('MyPictures') as MyPictures,
 GetFolderPath('Windows') as Windows
AutoGenerate 1;
```

Una vez que se ha recargado la app, los campos *MyMusic*, *MyPictures* y *Windows* se añaden al modelo de datos. Cada campo contiene la ruta a la carpeta definida en la entrada. Por ejemplo:

- *C:\Users\smu\Music* for the folder *MyMusic*
- *C:\Users\smu\Pictures* for the folder *MyPictures*
- *C:\Windows* for the folder *Windows*

QvdCreateTime

Esta función de script devuelve la marca de tiempo del encabezado XML de un archivo QVD, si la hay, de lo contrario devuelve NULL. En la marca de tiempo, la hora se proporciona en UTC.

Sintaxis:

```
QvdCreateTime(filename)
```

Argumentos:

Argumentos

Argumento	Descripción
filename	El nombre de un archivo QVD, si es necesario incluyendo la ruta, como una carpeta o una conexión de datos a archivos web. **Ejemplo: *'lib://Table Files/'*** En el modo de elaboración de scripts de legado, se admiten también los diguientes formatos de ruta: • Absoluta **Ejemplo: *c:\data*** • relativa al directorio de trabajo de la app Qlik Sense. **Ejemplo: *data*** • Dirección URL (HTTP o FTP), que apunta a una ubicación en Internet o una intranet. **Ejemplo: *http://www.qlik.com***

Ejemplo:

```
QvdCreateTime('MyFile.qvd')

QvdCreateTime('C:\MyDir\MyFile.qvd')

QvdCreateTime('lib://DataFiles/MyFile.qvd')
```

QvdFieldName

Esta función de script devuelve el nombre del número de campo **fieldno** en un archivo QVD. Si el campo no existe, devuelve NULL.

Sintaxis:

```
QvdFieldName(filename , fieldno)
```

Argumentos:

Argumentos

Argumento	Descripción
filename	El nombre de un archivo QVD, si es necesario incluyendo la ruta, como una carpeta o una conexión de datos a archivos web. **Ejemplo: *'lib://Table Files/'*** En el modo de elaboración de scripts de legado, se admiten también los diguientes formatos de ruta: • Absoluta **Ejemplo: *c:\data*** • relativa al directorio de trabajo de la app Qlik Sense. **Ejemplo: *data*** • Dirección URL (HTTP o FTP), que apunta a una ubicación en Internet o una intranet. **Ejemplo: *http://www.qlik.com***
fieldno	El número del campo dentro de la tabla contenida en el archivo QVD.

Ejemplos:

```
QvdFieldName ('MyFile.qvd', 5)

QvdFieldName ('C:\MyDir\MyFile.qvd', 5)

QvdFieldName ('lib://DataFiles/MyFile.qvd', 5)
```

Los tres ejemplos devuelven el nombre del quinto campo de la tabla contenida en el archivo QVD.

QvdNoOfFields

Esta función de script devuelve el número de campos de un archivo QVD.

Sintaxis:

```
QvdNoOfFields(filename)
```

Argumentos:

Argumentos

Argumento	Descripción
filename	El nombre de un archivo QVD, si es necesario incluyendo la ruta, como una carpeta o una conexión de datos a archivos web. **Ejemplo: *'lib://Table Files/'*** En el modo de elaboración de scripts de legado, se admiten también los diguientes formatos de ruta: • Absoluta **Ejemplo: *c:\data*** • relativa al directorio de trabajo de la app Qlik Sense. **Ejemplo: *data*** • Dirección URL (HTTP o FTP), que apunta a una ubicación en Internet o una intranet. **Ejemplo: *http://www.qlik.com***

Ejemplos:

```
QvdNoOfFields ('MyFile.qvd')

QvdNoOfFields ('C:\MyDir\MyFile.qvd')

QvdNoOfFields ('lib://DataFiles/MyFile.qvd')
```

QvdNoOfRecords

Ejemplo: Esta función de script devuelve el número de registros que hay actualmente en un archivo QVD.

Sintaxis:

```
QvdNoOfRecords(filename)
```

Argumentos:

Argumentos

Argumento	Descripción
filename	El nombre de un archivo QVD, si es necesario incluyendo la ruta, como una carpeta o una conexión de datos a archivos web. **Ejemplo: *'lib://Table Files/'*** En el modo de elaboración de scripts de legado, se admiten también los diguientes formatos de ruta: • Absoluta **Ejemplo: *c:\data*** • relativa al directorio de trabajo de la app Qlik Sense. **Ejemplo: *data*** • Dirección URL (HTTP o FTP), que apunta a una ubicación en Internet o una intranet. **Ejemplo: *http://www.qlik.com***

Ejemplos:

```
QvdNoOfRecords ('MyFile.qvd')

QvdNoOfRecords ('C:\MyDir\MyFile.qvd')

QvdNoOfRecords ('lib://DataFiles/MyFile.qvd')
```

QvdTableName

Esta función de script devuelve el nombre de la tabla almacenada en un archivo QVD.

Sintaxis:

```
QvdTableName(filename)
```

Argumentos:

Argumentos

Argumento	Descripción
filename	El nombre de un archivo QVD, si es necesario incluyendo la ruta, como una carpeta o una conexión de datos a archivos web. **Ejemplo: *'lib://Table Files/'*** En el modo de elaboración de scripts de legado, se admiten también los diguientes formatos de ruta: • Absoluta **Ejemplo: *c:\data*** • relativa al directorio de trabajo de la app Qlik Sense. **Ejemplo: *data*** • Dirección URL (HTTP o FTP), que apunta a una ubicación en Internet o una intranet. **Ejemplo: *http://www.qlik.com***

Ejemplos:

```
QvdTableName ('MyFile.qvd')

QvdTableName ('C:\MyDir\MyFile.qvd')

QvdTableName ('lib://data\MyFile.qvd')
```

8.11 Funciones financieras

Las funciones financieras pueden utilizarse en el script de carga de datos y en las expresiones de gráficos para calcular pagos y tipos de interés.
Para todos los argumentos, el dinero que se abona se representa mediante números negativos. El dinero que se recibe se representa en números positivos.
Aquí se enumeran los argumentos que se utilizan en las funciones financieras (excepto los que comienzan por **range**-).

*Para todas las funciones financieras es vital que sea coherente al especificar unidades para **rate** y **nper**. Si se realizan pagos mensuales en un préstamo a cinco años con un interés anual del 6%, utilice 0,005 (6%/12) para **rate** y 60 (5*12) para **nper**. Si los pagos anuales se realizan en el mismo préstamo, utilice 6% para **rate** y 5 para **nper**.*

Visión global de las funciones financieras

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

FV
Esta función devuelve el valor futuro de una inversión basada en pagos periódicos y constantes y a un interés anual simple.

```
FV (rate, nper, pmt [ ,pv [ , type ] ])
```

nPer
Esta función devuelve el número de periodos para una inversión basada en pagos periódicos y constantes y a un tipo de interés constante.

```
nPer (rate, pmt, pv [ ,fv [ , type ] ])
```

Pmt
Esta función devuelve el pago de un préstamo basado en pagos periódicos y constantes y a un tipo constante de interés. No puede cambiar durante la validez de un periodo anual. Un pago se indica como un número negativo, por ejemplo, -20.

```
Pmt (rate, nper, pv [ ,fv [ , type ] ] )
```

PV
Esta función devuelve el valor actual de una inversión.

```
PV (rate, nper, pmt [ ,fv [ , type ] ])
```

Rate
Esta función devuelve el tipo de interés por periodo en una anualidad. El resultado tiene un formato numérico predeterminado **Fix** de dos decimales y %.

```
Rate (nper, pmt , pv [ ,fv [ , type ] ])
```

BlackAndSchole

El modelo Black and Scholes es un modelo matemático para instrumentos derivados de los mercados financieros. La formula calcula el valor hipotético (teórico) de una opción. En Qlik Sense, la función **BlackAndSchole** devuelve el valor conforme a la fórmula no modificada de Black and Scholes (opciones del estilo Europeo).

```
BlackAndSchole(strike , time_left , underlying_price , vol , risk_free_rate , type)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
strike	El precio futuro de compra de las acciones.
time_left	El número de períodos de tiempo restantes.
underlying_price	El valor actual de las acciones.
vol	Volatilidad (del precio de las acciones) expresado como un porcentaje en forma decimal, por periodo de tiempo.
risk_free_rate	El tipo libre de riesgo expresado como un porcentaje en forma decimal, por periodo de tiempo.
call_or_put	El tipo de opción: 'c', 'call' o cualquier valor numérico distinto de cero para las opciones de llamada. 'p', 'put' o 0 para opciones de venta.

Limitaciones:

El valor de strike, time_left y underlying_price debe ser >0.

El valor de vol y risk_free_rate debe ser: <0 o >0.

Ejemplos y resultados:

Ejemplos de script

Ejemplo	Resultado
`BlackAndScholes(130, 4, 68.5, 0.4, 0.04, 'call')` Esto calcula el precio teórico de una opción de compra de una acción que vale 68,5 hoy, a un valor de 130 en 4 años. La fórmula presupone una volatilidad del 0,4 (40%) anual y un tipo libre de riesgo del 0,04 (4%).	Devuelve 11.245.

FV

Esta función devuelve el valor futuro de una inversión basada en pagos periódicos y constantes y a un interés anual simple.

Sintaxis:

```
FV(rate, nper, pmt [ ,pv [ , type ] ])
```

Tipo de datos que devuelve: numérico. De forma predeterminada, el resultado recibirá el formato de moneda..

Argumentos:

Argumentos

Argumento	Descripción
rate	Es el tipo de interés por periodo.
nper	Es el número total de pagos en un periodo anual.
pmt	Es el pago efectuado en cada periodo. No puede cambiar durante la validez de un periodo anual. Un pago se indica como un número negativo, por ejemplo, -20.
pv	Es el valor presente, o la cantidad total, que una serie de pagos futuros vale ahora mismo. Si se omite **pv**, se asume que es 0 (cero).
type	Debe ser 0 si los pagos están previstos para el final del periodo y 1 si los pagos se prevén para el comienzo del periodo. Si se omite **type**, se asume que es 0.

Ejemplos y resultados:

Ejemplo de script

Ejemplo	Resultado
Está pagando un nuevo grabador de vídeo a 36 cuotas mensuales de 20 $. El tipo de interés es del 6% anual. La factura de pago llega a finales de cada mes. ¿Cuál es el total invertido tras haber pagado la última factura? `FV(0.005,36,-20)`	Devuelve $786.72

nPer

Esta función devuelve el número de periodos para una inversión basada en pagos periódicos y constantes y a un tipo de interés constante.

Sintaxis:

```
nPer(rate, pmt, pv [ ,fv [ , type ] ])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
rate	Es el tipo de interés por periodo.
nper	Es el número total de pagos en un periodo anual.

Argumento	Descripción
pmt	Es el pago efectuado en cada periodo. No puede cambiar durante la validez de un periodo anual. Un pago se indica como un número negativo, por ejemplo, -20.
pv	Es el valor presente, o la cantidad total, que una serie de pagos futuros vale ahora mismo. Si se omite **pv**, se asume que es 0 (cero).
fv	Es el valor futuro o el balance de efectivo, que se desea lograr tras haberse realizado el último pago. Si se omite **fv**, se asume que es 0.
type	Debe ser 0 si los pagos están previstos para el final del periodo y 1 si los pagos se prevén para el comienzo del periodo. Si se omite **type**, se asume que es 0.

Ejemplos y resultados:

Ejemplo de script

Ejemplo	Resultado
Deseamos vender un electrodoméstico en cuotas mensuales de 20 $. El tipo de interés es del 6% anual. La factura de pago llega a finales de cada mes. ¿Cuántos periodos hacen falta si el valor del dinero recibido tras abonar la última factura debería ser igual a 800$? `nPer(0.005,-20,0,800)`	Devuelve 36,56

Pmt

Esta función devuelve el pago de un préstamo basado en pagos periódicos y constantes y a un tipo constante de interés. No puede cambiar durante la validez de un periodo anual. Un pago se indica como un número negativo, por ejemplo, -20.

```
Pmt(rate, nper, pv [ ,fv [ , type ] ] )
```

Tipo de datos que devuelve: numérico. De forma predeterminada, el resultado recibirá el formato de moneda..

Para encontrar la cantidad total abonada durante la duración del préstamo, multiplique el valor **pmt** devuelto por **nper**.

Argumentos:

Argumentos

Argumento	Descripción
rate	Es el tipo de interés por periodo.
nper	Es el número total de pagos en un periodo anual.
pv	Es el valor presente, o la cantidad total, que una serie de pagos futuros vale ahora mismo. Si se omite **pv**, se asume que es 0 (cero).

Argumento	Descripción
fv	Es el valor futuro o el balance de efectivo, que se desea lograr tras haberse realizado el último pago. Si se omite **fv**, se asume que es 0.
type	Debe ser 0 si los pagos están previstos para el final del periodo y 1 si los pagos se prevén para el comienzo del periodo. Si se omite **type**, se asume que es 0.

Ejemplos y resultados:

Ejemplos de script

Ejemplo	Resultado
La fórmula siguiente devuelve el pago mensual de un préstamo de 20.000 $ en un porcentaje anual del 10 por ciento, que debe liquidarse en 8 meses: `Pmt(0.1/12,8,20000)`	Devuelve -$2,594.66
Para el mismo préstamo, si el pago debe hacerse al inicio del periodo, el pago es: `Pmt(0.1/12,8,20000,0,1)`	Devuelve -$2,573.21

PV

Esta función devuelve el valor actual de una inversión.

```
PV(rate, nper, pmt [ ,fv [ , type ] ])
```

Tipo de datos que devuelve: numérico. De forma predeterminada, el resultado recibirá el formato de moneda..

El valor actual es la cantidad total que supone una serie de pagos futuros en el momento presente. Por ejemplo, cuando usted toma prestado un dinero, la cantidad total del préstamo es el valor actual que debe al prestamista.

Argumentos:

Argumentos

Argumento	Descripción
rate	Es el tipo de interés por periodo.
nper	Es el número total de pagos en un periodo anual.
pmt	Es el pago efectuado en cada periodo. No puede cambiar durante la validez de un periodo anual. Un pago se indica como un número negativo, por ejemplo, -20.
fv	Es el valor futuro o el balance de efectivo, que se desea lograr tras haberse realizado el último pago. Si se omite **fv**, se asume que es 0.
type	Debe ser 0 si los pagos están previstos para el final del periodo y 1 si los pagos se prevén para el comienzo del periodo. Si se omite **type**, se asume que es 0.

Ejemplos y resultados:

Ejemplo de script

Ejemplo	Resultado
¿Cuál es el valor actual de una deuda, si debe abonar 100 $ al final de cada mes durante un periodo de 5 años, dado un interés del 7%? `PV(0.07/12,12*5,-100,0,0)`	Devuelve $5,050.20

Rate

Esta función devuelve el tipo de interés por periodo en una anualidad. El resultado tiene un formato numérico predeterminado **Fix** de dos decimales y %.

Sintaxis:

```
Rate(nper, pmt , pv [ ,fv [ , type ] ])
```

Tipo de datos que devuelve: numérico.

La tasa **rate** se calcula por iteración y puede tener cero o más soluciones. Si los resultados sucesivos de **rate** no convergen, se devolverá un valor NULL.

Argumentos:

Argumentos

Argumento	Descripción
nper	Es el número total de pagos en un periodo anual.
pmt	Es el pago efectuado en cada periodo. No puede cambiar durante la validez de un periodo anual. Un pago se indica como un número negativo, por ejemplo, -20.
pv	Es el valor presente, o la cantidad total, que una serie de pagos futuros vale ahora mismo. Si se omite **pv**, se asume que es 0 (cero).
fv	Es el valor futuro o el balance de efectivo, que se desea lograr tras haberse realizado el último pago. Si se omite **fv**, se asume que es 0.
type	Debe ser 0 si los pagos están previstos para el final del periodo y 1 si los pagos se prevén para el comienzo del periodo. Si se omite **type**, se asume que es 0.

Ejemplos y resultados:

Ejemplo de script

Ejemplo	Resultado
¿Cuál es el tipo de interés de un préstamo anual de 10.000 $ a cinco años, con pagos mensuales de 300 $? `Rate(60,-300,10000)`	Devuelve 2.00%

8.12 Funciones de formato

Las funciones de formato imponen un formato de presentación a los campos o expresiones numéricos de entrada. Dependiendo del tipo de datos, podemos especificar los caracteres del separador decimal, el separador de miles, etc.

Las funciones devuelven todas ellas un valor dual con ambos valores, el de cadena y númerico, pero puede pensarse en ellas como que realizan una conversión de número a cadena. **Dual()** es un caso especial, pero las demás funciones de formato toman el valor numérico de la expresión de entrada y generan una cadena que representa el número.

Por el contrario, las funciones de interpretación toman expresiones de cadena y devuelven números, especificando el formato del número resultante.

Las funciones pueden utilizarse tanto en scripts de carga de datos como en expresiones de gráficos.

Todas las representaciones numéricas se dan con un punto decimal como separador decimal.

Descripción general de las funciones de formato

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

ApplyCodepage

ApplyCodepage() aplica un juego de caracteres de página de códigos diferente al campo o texto indicado en la expresión. El argumento **codepage** debe estar en formato numérico.

```
ApplyCodepage (text, codepage)
```

Date

Date() da formato a una expresión como una fecha utilizando el formato establecido en las variables de sistema del script de carga de datos, o en el sistema operativo, o en una cadena de formato, si se proporciona.

```
Date (number[, format])
```

Dual

Dual() combina un número y una cadena en un solo registro, de manera que la representación numérica del registro se puede utilizar con fines de ordenación y cálculo, mientras que el valor de la cadena se puede usar para fines de visualización.

```
Dual (text, number)
```

Interval

Interval() da formato a un número como un intervalo de tiempo utilizando el formato establecido en las variables de sistema del script de carga de datos, o el sistema operativo, o una cadena de formato, si se suministra.

```
Interval (number[, format])
```

Money

Money() da formato numérico a una expresión con el valor de moneda, en el formato numérico establecido en las variables de sistema del script de carga de datos o en el sistema operativo, a menos que se suministre una cadena de formato y, opcionalmente, unos separadores decimal y de miles.

```
Money (number[, format[, dec_sep [, thou_sep]]])
```

Num

Num() da formato a un número, es decir, convierte el valor numérico de la entrada para mostrar texto en vez, utilizando el formato especificado en el segundo parámetro. Si se omite el segundo parámetro, utiliza los separadores de decimal y de miles definidos en el script de carga de datos. Los símbolos de separador decimal o de miles personalizados son parámetros opcionales.

```
Num (number[, format[, dec_sep [, thou_sep]]])
```

Time

Time() da formato a una expresión como un valor de hora, en el formato de tiempo definido en las variables de sistema del script de carga de datos, o en el sistema operativo, a menos que se suministre una cadena de formato.

```
Time (number[, format])
```

Timestamp

TimeStamp() da formato a una expresión como un valor de fecha y hora, en el formato de tiempo definido en las variables de sistema del script de carga de datos, o el sistema operativo, a menos que se proporcione una cadena de formato.

```
Timestamp (number[, format])
```

Vea también:

Funciones de interpretación (page 1281)

ApplyCodepage

ApplyCodepage() aplica un juego de caracteres de página de códigos diferente al campo o texto indicado en la expresión. El argumento **codepage** debe estar en formato numérico.

Aunque se puede usar ApplyCodepage en expresiones de gráficos, se usa más habitualmente como una función de script en el editor de carga de datos. Por ejemplo, a medida que carga archivos que podrían haber sido guardados en diferentes juegos de caracteres fuera de su control, puede aplicar la página de código que representa el conjunto de caracteres que necesite.

Sintaxis:

```
ApplyCodepage(text, codepage)
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
text	Campo o texto al que se desea aplicar una página de códigos diferente, dado por el argumento **codepage**.
codepage	Número que representa la página de códigos que se aplicará al campo o expresión dado por **text**.

Ejemplos y resultados:

Ejemplos de script

Ejemplo	Resultado
`LOAD ApplyCodepage (ROWX,1253) as GreekProduct, ApplyCodepage (ROWY, 1255) as HebrewProduct, ApplyCodepage (ROWZ, 65001) as EnglishProduct; SQL SELECT ROWX, ROWY, ROWZ From Products;`	Al cargar desde SQL, la fuente puede tener una mezcla de diferentes conjuntos de caracteres: Cirílico, hebreo, etc., desde el formato UTF-8. Estos tendrían que cargarse fila por fila, aplicando una página de códigos diferente por cada fila. El valor **codepage** 1253 representa el conjunto de caracteres griegos de Windows, el valor 1255 representa el de hebreo y el valor 65001 representa caracteres estándar UTF-8 de latín.

Vea también: *Character set (page 169)*

Date

Date() da formato a una expresión como una fecha utilizando el formato establecido en las variables de sistema del script de carga de datos, o en el sistema operativo, o en una cadena de formato, si se proporciona.

Sintaxis:

```
Date(number[, format])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
number	El número al que se ha de dar formato.
format	La cadena que describe el formato de la cadena resultante. Si no se suministra cadena de formato alguna, se utiliza el formato de fecha definido en las variables del sistema del script de carga de datos, o se utiliza el sistema operativo.

Ejemplos y resultados:
Los ejemplos a continuación contienen estas dos configuraciones por defecto:

- Configuración de fecha 1: YY-MM-DD
- Configuración de fecha 2: M/D/YY

Ejemplo:

```
Date( A )
```
donde A=35648

Tabla resultante

Resultados	Configuración 1	Configuración 2
Cadena:	97-08-06	8/6/97
Número:	35648	35648

Ejemplo:

```
Date( A, 'YY.MM.DD' )
```
donde A=35648

Tabla resultante

Resultados	Configuración 1	Configuración 2
Cadena:	97.08.06	97.08.06
Número:	35648	35648

Ejemplo:

```
Date( A, 'DD.MM.YYYY' )
```
donde A=35648.375

Tabla resultante

Resultados	Configuración 1	Configuración 2
Cadena:	06.08.1997	06.08.1997
Número:	35648.375	35648.375

Ejemplo:

```
Date( A, 'YY.MM.DD' )
```
donde A=8/6/97

Tabla resultante

Resultados	Configuración 1	Configuración 2
Cadena:	NULL (nada)	97.08.06
Número:	NULL	35648

Dual

Dual() combina un número y una cadena en un solo registro, de manera que la representación numérica del registro se puede utilizar con fines de ordenación y cálculo, mientras que el valor de la cadena se puede usar para fines de visualización.

Sintaxis:

```
Dual(text, number)
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
text	El valor de la cadena que se ha de utilizar junto con el argumento de número.
number	El número que se ha de utilizar junto con la cadena en el argumento de número.

En Qlik Sense, todos los valores de campo son potencialmente valores duales. Esto significa que los valores de campo pueden tener un valor tanto numérico como de texto. Un ejemplo es una fecha que podría tener un valor numérico 40908 y la representación textual '2011-12-31'.

Cuando varios elementos de datos de un campo tienen diferentes representaciones de cadena pero una misma representación numérica válida, compartirán todos ellos la primera representación de cadena de caracteres que se encuentre.

*La función **dual** normalmente se usa al principio del script, antes de que se lean otros datos en el campo correspondiente, a fin de crear esa primera representación de cadena, que se mostrará en los paneles de filtrado.*

Ejemplos y resultados:

Ejemplos de script

Ejemplo	**Descripción**

Añada los ejemplos siguientes a su script y ejecútelo.
`Load dual ( NameDay,NumDay ) as DayOfWeek inline`
`[ NameDay,NumDay`
`Monday,0`
`Tuesday,1`
`Wednesday,2`
`Thursday,3`
`Friday,4`
`Saturday,5`
`Sunday,6 ];`
	El campo DayOfWeek se puede usar en una visualización, por ejemplo, como una dimensión. En una tabla con los días de la semana se clasifican automáticamente en su secuencia numérica correcta, en lugar de por orden alfabético.

`Load Dual('Q' & Ceil (Month(Now())/3), Ceil(Month(Now())/3)) as Quarter AutoGenerate 1;`	Este ejemplo halla el trimestre actual. Se muestra como Q1 cuando la función **Now()** se ejecuta en los tres primeros meses del año, Q2 durante los segundos tres meses, y así sucesivamente. Sin embargo, cuando se utiliza en la ordenación, el campo Quarter se comportará como su valor numérico: 1 a 4.
`Dual('Q' & Ceil(Month (Date)/3), Ceil(Month (Date)/3)) as Quarter`	Como en el ejemplo anterior, el campo Quarter se crea con los valores de texto 'Q1' a 'Q4' y se le asignan los valores numéricos 1 a 4. Para usar esto en el script deben cargarse los valores de Date.
`Dual(WeekYear(Date) & '-W' & Week(Date), WeekStart(Date)) as YearWeek`	Este ejemplo crea un campo YearWeek con valores de texto del formulario '2012-W22' y, al mismo tiempo, asigna un valor numérico correspondiente al número de fecha del primer día de la semana, por ejemplo: 41057. Para usar esto en el script deben cargarse los valores de Date.

Interval

Interval() da formato a un número como un intervalo de tiempo utilizando el formato establecido en las variables de sistema del script de carga de datos, o el sistema operativo, o una cadena de formato, si se suministra.

Podemos formatear los intervalos como una hora, días, o una combinación de días, horas, minutos, segundos y fracciones de segundo.

Sintaxis:

```
Interval(number[, format])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
number	El número al que se ha de dar formato.
format	Cadena que describe cómo ha de formatearse la cadena de intervalo resultante. Si se omite, se utilizan el formato breve de fecha, el formato de hora y el separador decimal establecidos en el sistema operativo.

Ejemplos y resultados:
Los ejemplos a continuación contienen estas dos configuraciones por defecto:

- Configuración de formato de fecha 1: YY-MM-DD
- Configuración de formato de fecha 2: hh:mm:ss
- Separador de números decimales: ,

Tabla de resultados

Ejemplo	Cadena	Número
`Interval( A )` donde A=0,375	09:00:00	0.375
`Interval( A )` donde A=1,375	33:00:00	1.375
`Interval( A, 'D hh:mm' )` donde A=1,375	1 09:00	1.375
`Interval( A-B, 'D hh:mm' )` donde A=97-08-06 09:00:00 and B=96-08-06 00:00:00	365 09:00	365.375

Money

Money() da formato numérico a una expresión con el valor de moneda, en el formato numérico establecido en las variables de sistema del script de carga de datos o en el sistema operativo, a menos que se suministre una cadena de formato y, opcionalmente, unos separadores decimal y de miles.

Sintaxis:

```
Money(number[, format[, dec_sep[, thou_sep]]])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
number	El número al que se ha de dar formato.
format	Cadena que describe cómo ha de formatearse la cadena de moneda resultante.
dec_sep	Cadena que especifica el separador numérico decimal.
thou_sep	Cadena que especifica el separador numérico de miles.

Si los argumentos 2-4 se omiten, se usa el formato de moneda establecido en el sistema operativo.

Ejemplos y resultados:
Los ejemplos a continuación contienen estas dos configuraciones por defecto:

- Configuración de formato de moneda 1: kr ##0,00MoneyThousandSep
- Configuración de formato de moneda 2: $ #,##0.00MoneyThousandSep

Ejemplo:

```
Money( A )
```
donde A=35648

Tabla de resultados

Resultados	Configuración 1	Configuración 2
Cadena:	kr 35 648,00	$ 35,648.00
Número:	35648.00	35648.00

Ejemplo:

```
Money( A, '#,##0 ¥', '.' , ',' )
```
donde A=3564800

Tabla de resultados

Resultados	Configuración 1	Configuración 2
Cadena:	3,564,800 ¥	3,564,800 ¥
Número:	3564800	3564800

Num

Num() da formato a un número, es decir, convierte el valor numérico de la entrada para mostrar texto en vez, utilizando el formato especificado en el segundo parámetro. Si se omite el segundo parámetro, utiliza los separadores de decimal y de miles definidos en el script de carga de datos. Los símbolos de separador decimal o de miles personalizados son parámetros opcionales.

Sintaxis:

```
Num(number[, format[, dec_sep [, thou_sep]]])
```

Tipo de datos que devuelve: dual

La función Num devuelve un valor dual con la cadena y el valor numérico. La función toma el valor numérico de la expresión introducida y genera una cadena que representa el número.

Argumentos:

Argumentos

Argumento	Descripción
number	El número al que se ha de dar formato.
format	Cadena que describe qué formato habrá de tener la cadena resultante. Si se omite el segundo parámetro, utiliza los separadores de decimal y de miles definidos en el script de carga de datos.
dec_sep	Cadena que especifica el separador numérico decimal. Si se omite, se utilizará el valor de la variable DecimalSep establecido en el script de carga de datos.
thou_sep	Cadena que especifica el separador numérico de miles. Si se omite, se utilizará el valor de la variable ThousandSep establecido en el script de carga de datos.

Ejemplo: Expresión de gráfico

Ejemplo:

La tabla siguiente muestra los resultados cuando el campo A es igual a 35648.312.

Resultados

A	Resultado
Num(A)	35648.312 (depende de las variables de entorno en el script)
Num(A, '0.0', '.')	35648.3
Num(A, '0,00', ',')	35648,31
Num(A, '#,##0.0', '.', ',')	35,648.3
Num(A, '# ##0', ',', ' ')	35 648

Ejemplo: Script de carga

Script de carga

Num se puede usar en el script de carga para dar formato a un número, incluso si los separadores de miles y decimales ya están configurados en el script. El script de carga siguiente incluye separadores de miles y decimales específicos, pero luego usa *Num* para dar formato a los datos de diferentes maneras.

En el **Editor de carga de datos**, cree una nueva sección y luego agregue el script de ejemplo y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

```
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
];
```

Tabla de Qlik Sense que muestra los resultados de la función *Num* en el script de carga. La cuarta columna de la tabla contiene un uso de formato incorrecto, a modo de ejemplo.

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

Ejemplo: Script de carga

Script de carga

Num se puede utilizar en un script de carga para dar formato a un número como un porcentaje.

En el **Editor de carga de datos**, cree una nueva sección y luego agregue el script de ejemplo y ejecútelo. A continuación, agregue como mínimo los campos de la columna de resultados en una hoja de su app para ver el resultado.

```
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
];
```

Tabla de Qlik Sense que muestra los resultados de la función *Num* utilizada en el script de carga para dar formato de porcentajes.

Discount	Discount #,##0%
0.333333333333333	33%
0.22	22%
0	0%
.14	14%
0.1	10%
0	0%
75	7,500%

Time

Time() da formato a una expresión como un valor de hora, en el formato de tiempo definido en las variables de sistema del script de carga de datos, o en el sistema operativo, a menos que se suministre una cadena de formato.

Sintaxis:

```
Time(number[, format])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
number	El número al que se ha de dar formato.
format	Cadena que describe qué formato ha de tener la cadena de hora resultante. Si se omite, se utilizan el formato breve de fecha, el formato de hora y el separador decimal establecidos en el sistema operativo.

Ejemplos y resultados:
Los ejemplos a continuación contienen estas dos configuraciones por defecto:

- Configuración de formato de hora 1: hh:mm:ss
- Configuración de formato de hora 2: hh.mm.ss

Ejemplo:

```
Time( A )
```
donde A=0,375

Tabla de resultados

Resultados	Configuración 1	Configuración 2
Cadena:	09:00:00	09.00.00
Número:	0.375	0.375

Ejemplo:

```
Time( A )
```
donde A=35648,375

Tabla de resultados

Resultados	Configuración 1	Configuración 2
Cadena:	09:00:00	09.00.00
Número:	35648.375	35648.375

Ejemplo:

```
Time( A, 'hh-mm' )
```
donde A=0,99999

Tabla de resultados

Resultados	Configuración 1	Configuración 2
Cadena:	23-59	23-59
Número:	0.99999	0.99999

Timestamp

TimeStamp() da formato a una expresión como un valor de fecha y hora, en el formato de tiempo definido en las variables de sistema del script de carga de datos, o el sistema operativo, a menos que se proporcione una cadena de formato.

Sintaxis:

```
Timestamp(number[, format])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
number	El número al que se ha de dar formato.
format	Cadena que describe qué formato ha de tener la cadena de fecha-hora resultante. Si se omite, se utilizan el formato breve de fecha, el formato de hora y el separador decimal establecidos en el sistema operativo.

Ejemplos y resultados:
Los ejemplos a continuación contienen estas dos configuraciones por defecto:

- Configuración de formato de fecha-hora 1: YY-MM-DD hh:mm:ss
- Configuración de formato de fecha-hora 2: M/D/YY hh:mm:ss

Ejemplo:

```
Timestamp( A )
```
donde A=35648,375

Tabla de resultados

Resultados	Configuración 1	Configuración 2
Cadena:	97-08-06 09:00:00	8/6/97 09:00:00
Número:	35648.375	35648.375

Ejemplo:

```
Timestamp( A,'YYYY-MM-DD hh.mm')
```
donde A=35648

Tabla de resultados

Resultados	Configuración 1	Configuración 2
Cadena:	1997-08-06 00.00	1997-08-06 00.00
Número:	35648	35648

8.13 Funciones numéricas generales

En estas funciones numéricas generales, los argumentos son expresiones donde **x** se debe interpretar como un número con valor real. Todas las funciones pueden utilizarse tanto en scripts de carga de datos como en expresiones de gráficos.

Descripción general de las funciones numéricas generales

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

bitcount

BitCount() devuelve cuántos bits en el equivalente binario de un número decimal están establecidos en 1. Es decir, la función devuelve el número de bits establecidos en **integer_number**, donde **integer_number** se interpreta como un entero de 32 bits con signo .

```
BitCount(integer_number)
```

div

Div() devuelve la parte entera de la división aritmética del primer argumento por el segundo argumento. Ambos parámetros se interpretan como números reales, esto es, no tienen que ser enteros.

```
Div (integer_number1, integer_number2)
```

fabs

Fabs() devuelve el valor absoluto de **x**. El resultado es un número positivo.

```
Fabs (x)
```

fact

Fact() devuelve el factorial de un entero positivo **x**.

```
Fact (x)
```

frac

Frac() devuelve la parte **x** de la fracción.

```
Frac (x)
```

sign

Sign() devuelve 1, 0 o -1 dependiendo de si **x** es un número positivo, 0 o un número negativo.

```
Sign (x)
```

Funciones de combinación y permutación

combin

Combin() devuelve el número de combinaciones de elementos **q** que se pueden seleccionar de un conjunto de elementos **p**. Según lo representado por la fórmula: `Combin(p,q) = p! / q!(p-q)!` El orden en que se seleccionan los elementos es insignificante.

```
Combin (p, q)
```

permut

Permut() devuelve el número de permutaciones de elementos **q** que pueden seleccionarse de un conjunto de elementos **p**. Según lo representado por la fórmula: `Permut(p,q) = (p)! / (p - q)!` El orden en el que se seleccionan los elementos es significativo.

```
Permut (p, q)
```

Funciones de módulo

fmod

fmod() es una función de módulo generalizada que devuelve la parte restante de la división entera del primer argumento (el dividendo) por el segundo argumento (el divisor). El resultado es un número real. Ambos argumentos se interpretan como números reales, esto es, no tienen que ser enteros.

```
Fmod (a, b)
```

mod

Mod() es una función matemática de módulo que devuelve el resto no negativo de una división de un entero. El primer argumento es el dividendo y el segundo argumento es el divisor. Ambos argumentos deben ser valores enteros.

```
Mod (integer_number1, integer_number2)
```

Funciones de paridad

even

Even() devuelve True (-1), si **integer_number** es un entero par o cero. Devuelve False (0), si **integer_number** es un entero impar y NULL si **integer_number** no es un entero.

```
Even (integer_number)
```

odd

Odd() devuelve True (-1), si **integer_number** es un entero impar o cero. Devuelve False (0), si **integer_number** es un entero par, y NULL si **integer_number** no es un entero.

```
Odd (integer_number)
```

Funciones de redondeo

ceil

Ceil() redondea un número hacia arriba, hacia el múltiplo más cercano de **step** desplazado por el número **offset** .

```
Ceil (x[, step[, offset]])
```

floor

Floor() redondea un número hacia abajo, hacia el múltiplo más cercano de **step** desplazado por el número **offset** .

```
Floor (x[, step[, offset]])
```

round

Round() devuelve el resultado de redondear un número hacia arriba o hacia abajo al múltiplo más cercano de **step** desplazado por el número **offset** .

```
Round ( x [ , step [ , offset ]])
```

BitCount

BitCount() devuelve cuántos bits en el equivalente binario de un número decimal están establecidos en 1. Es decir, la función devuelve el número de bits establecidos en **integer_number**, donde **integer_number** se interpreta como un entero de 32 bits con signo .

Sintaxis:

```
BitCount(integer_number)
```

Tipo de datos que devuelve: Entero

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
`BitCount ( 3 )`	3 equivale al binario 11, por lo que devuelve el valor 2
`BitCount ( -1 )`	-1 equivale a 64 unos en binario, por lo que devuelve el valor 64

Ceil

Ceil() redondea un número hacia arriba, hacia el múltiplo más cercano de **step** desplazado por el número **offset** .

Compare con la función **floor**, que redondea los números introducidos hacia abajo.

Sintaxis:

```
Ceil(x[, step[, offset]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
x	Número introducido.
step	Incremento de intervalo. El valor predeterminado es 1.
offset	Define la base del intervalo de step. El valor predeterminado es 0.

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
`Ceil(2.4 )`	Devuelve 3 En este ejemplo, el tamaño del intervalo step es 1 y la base del intervalo step es 0. Los intervalos son ...0 < x <=1, 1 < x <= 2, **2< x <=3**, 3< x <=4...
`Ceil(4.2 )`	Devuelve 5
`Ceil(3.88 ,0.1)`	Devuelve 3,9 En este ejemplo, el tamaño del intervalo es 0,1 y la base del intervalo es 0. Los intervalos son ... 3.7 < x <= 3.8, **3.8 < x <= 3.9**, 3.9 < x <= 4.0...
`Ceil(3.88 ,5)`	Devuelve 5
`Ceil(1.1 ,1)`	Devuelve 2
`Ceil(1.1 ,1,0.5)`	Devuelve 1,5 En este ejemplo, el tamaño del paso step es 1 y el desplazamiento offset es 0,5. Esto significa que la base del intervalo de step es 0,5 y no 0. Los intervalos son ...**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)`	Devuelve 1,99 Los intervalos son ...-0.01< x <= 0.99, **0.99< x <= 1.99**, 1.99 < x <=2.99...

Combin

Combin() devuelve el número de combinaciones de elementos **q** que se pueden seleccionar de un conjunto de elementos **p**. Según lo representado por la fórmula: `Combin(p,q) = p! / q!(p-q)!` El orden en que se seleccionan los elementos es insignificante.

Sintaxis:

```
Combin(p, q)
```

Tipo de datos que devuelve: Entero

Limitaciones:

Los elementos no enteros se truncarán.

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
¿Cuántas combinaciones de 7 números pueden obtenerse de un total de 35 números de lotería? `Combin( 35,7 )`	Devuelve 6.724.520

Div

Div() devuelve la parte entera de la división aritmética del primer argumento por el segundo argumento. Ambos parámetros se interpretan como números reales, esto es, no tienen que ser enteros.

Sintaxis:

```
Div(integer_number1, integer_number2)
```

Tipo de datos que devuelve: Entero

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
`Div( 7,2 )`	Devuelve 3
`Div( 7.1,2.3 )`	Devuelve 3
`Div( 9,3 )`	Devuelve 3
`Div( -4,3 )`	Devuelve -1
`Div( 4,-3 )`	Devuelve -1
`Div( -4,-3 )`	Devuelve 1

Even

Even() devuelve True (-1), si **integer_number** es un entero par o cero. Devuelve False (0), si **integer_number** es un entero impar y NULL si **integer_number** no es un entero.

Sintaxis:

```
Even(integer_number)
```

Tipo de datos que devuelve: Booleano

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
`Even( 3 )`	Devuelve 0, False
`Even( 2 * 10 )`	Devuelve -1, True
`Even( 3.14 )`	Devuelve NULL

Fabs

Fabs() devuelve el valor absoluto de **x**. El resultado es un número positivo.

Sintaxis:

```
fabs(x)
```

Tipo de datos que devuelve: numérico

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
`fabs( 2.4 )`	Devuelve 2,4
`fabs( -3.8 )`	Devuelve 3,8

Fact

Fact() devuelve el factorial de un entero positivo **x**.

Sintaxis:

```
Fact(x)
```

Tipo de datos que devuelve: Entero

Limitaciones:

Si el número **x** no es un entero, se truncará. Los números no positivos devolverán NULL.

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
`Fact( 1 )`	Devuelve 1
`Fact( 5 )`	Devuelve 120 (1 * 2 * 3 * 4 * 5 = 120)
`Fact( -5 )`	Devuelve NULL

Floor

Floor() redondea un número hacia abajo, hacia el múltiplo más cercano de **step** desplazado por el número **offset** .

Compare con la función **ceil**, que redondea los números introducidos hacia arriba.

Sintaxis:

```
Floor(x[, step[, offset]])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
x	Número introducido.
step	Incremento de intervalo. El valor predeterminado es 1.
offset	Define la base del intervalo de step. El valor predeterminado es 0.

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
`Floor(2.4)`	Devuelve 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)`	Devuelve 4

Ejemplos	Resultados
`Floor(3.88 ,0.1)`	Devuelve 3,8 En este ejemplo, el tamaño del intervalo es 0,1 y la base del intervalo es 0. Los intervalos son ... 3.7 <= x < 3.8, **3.8 <= x < 3.9**, 3.9 <= x < 4.0...
`Floor(3.88 ,5)`	Devuelve 0
`Floor(1.1 ,1)`	Devuelve 1
`Floor(1.1 ,1,0.5)`	Devuelve 0,5 En este ejemplo, el tamaño del paso step es 1 y el desplazamiento offset es 0,5. Esto significa que la base del intervalo de step es 0,5 y no 0. Los intervalos son ...**0.5 <= x <1.5**, 1.5 <= x < 2.5, 2.5<= x <3.5,...

Fmod

fmod() es una función de módulo generalizada que devuelve la parte restante de la división entera del primer argumento (el dividendo) por el segundo argumento (el divisor). El resultado es un número real. Ambos argumentos se interpretan como números reales, esto es, no tienen que ser enteros.

Sintaxis:

```
fmod(a, b)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
a	Dividendo
b	Divisor

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
`fmod( 7,2 )`	Devuelve 1
`fmod( 7.5,2 )`	Devuelve 1,5
`fmod( 9,3 )`	Devuelve 0
`fmod( -4,3 )`	Devuelve -1
`fmod( 4,-3 )`	Devuelve 1
`fmod( -4,-3 )`	Devuelve -1

Frac

Frac() devuelve la parte **x** de la fracción.

La fracción se define de tal manera que `Frac(x ) + Floor(x ) = x`. Dicho de una manera simple, esto significa que la parte fraccional de un número positivo es la diferencia entre el número (x) y el entero que precede a la parte fraccional.

Por ejemplo: La parte fraccional de 11,43 = 11,43 - 11 = 0,43

Para un número, por ejemplo, -1,4, Floor(-1.4) = -2, se produce el siguiente resultado:

La parte fraccional de -1,4 = -1,4 - (-2) = -1,4 + 2 = 0,6

Sintaxis:

```
Frac(x)
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
x	Número para el que se devuelve la fracción.

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
`Frac( 11.43 )`	Devuelve 0,43
`Frac( -1.4 )`	Devuelve 0,6
Extraiga el componente de tiempo de la representación numérica de una marca de tiempo, omitiendo así la fecha. `Time(Frac(44518.663888889))`	Devuelve 3:56:00 PM

Mod

Mod() es una función matemática de módulo que devuelve el resto no negativo de una división de un entero. El primer argumento es el dividendo y el segundo argumento es el divisor. Ambos argumentos deben ser valores enteros.

Sintaxis:

```
Mod(integer_number1, integer_number2)
```

Tipo de datos que devuelve: Entero

Limitaciones:

integer_number2 debe ser mayor que 0.

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
`Mod( 7,2 )`	Devuelve 1
`Mod( 7.5,2 )`	Devuelve NULL
`Mod( 9,3 )`	Devuelve 0
`Mod( -4,3 )`	Devuelve 2
`Mod( 4,-3 )`	Devuelve NULL
`Mod( -4,-3 )`	Devuelve NULL

Odd

Odd() devuelve True (-1), si **integer_number** es un entero impar o cero. Devuelve False (0), si **integer_number** es un entero par, y NULL si **integer_number** no es un entero.

Sintaxis:

```
Odd(integer_number)
```

Tipo de datos que devuelve: Booleano

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
`Odd( 3 )`	Devuelve -1, True
`Odd( 2 * 10 )`	Devuelve 0, False
`Odd( 3.14 )`	Devuelve NULL

Permut

Permut() devuelve el número de permutaciones de elementos **q** que pueden seleccionarse de un conjunto de elementos **p**. Según lo representado por la fórmula: `Permut(p,q) = (p)! / (p - q)!` El orden en el que se seleccionan los elementos es significativo.

Sintaxis:

```
Permut(p, q)
```

Tipo de datos que devuelve: Entero

Limitaciones:

Los argumentos no enteros serán truncados.

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
¿De cuántas formas pueden ser distribuidas las medallas de oro, plata y bronce después de una final de 100 m con 8 participantes? `Permut( 8,3 )`	Devuelve 336

Round

Round() devuelve el resultado de redondear un número hacia arriba o hacia abajo al múltiplo más cercano de **step** desplazado por el número **offset** .

Si el número que se ha de redondear está exactamente en el medio de un intervalo, se redondea hacia arriba.

Sintaxis:

```
Round(x[, step[, offset]])
```

Tipo de datos que devuelve: numérico

Si está redondeando un número de punto flotante puede que observe resultados erróneos. Estos errores de redondeo se deben a que los números de punto flotante están representados por un número finito de dígitos binarios. Por lo tanto, los resultados se calculan usando un número que ya está redondeado. Si estos errores de redondeo afectan a su trabajo, multiplique los números para convertirlos en enteros antes de redondear.

Argumentos:

Argumentos

Argumento	Descripción
x	Número introducido.
step	Incremento de intervalo. El valor predeterminado es 1.
offset	Define la base del intervalo de step. El valor predeterminado es 0.

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
`Round(3.8 )`	Devuelve 4 En este ejemplo, el tamaño de step es 1 y la base del intervalo de step es 0. Los intervalos son ...0 <= x <1, 1 <= x < 2, 2<= x <3, **3<= x <4**...
`Round(3.8,4 )`	Devuelve 4
`Round(2.5 )`	Devuelve 3. En este ejemplo, el tamaño de step es 1 y la base del intervalo de step es 0. Los intervalos son ...0 <= x <1, 1 <= x <2, **2<= x <3**...
`Round(2,4 )`	Devuelve 4. Redondeado hacia arriba porque 2 es exactamente la mitad del intervalo de step de 4. En este ejemplo, el tamaño de step es 4 y la base del intervalo de step es 0. Los intervalos son ...**0 <= x <4**, 4 <= x <8, 8<= x <12...
`Round(2,6 )`	Devuelve 0. Redondeado hacia abajo porque 2 es menos de la mitad del intervalo de step de 6. En este ejemplo, el tamaño de step es 6 y la base del intervalo de step es 0. Los intervalos son ...**0 <= x <6**, 6 <= x <12, 12<= x <18...
`Round(3.88 ,0.1)`	Devuelve 3,9 En este ejemplo, el tamaño de step es 0,1 y la base del intervalo de step es 0. Los intervalos son ... 3.7 <= x <3.8, **3.8 <= x <3.9**, 3.9 <= x < 4.0...
`Round (3.88875,1/1000)`	Devuelve 3,889 En este ejemplo, el tamaño del paso es 0,001, que redondea el número hacia arriba y lo limita a tres decimales.
`Round(3.88 ,5)`	Devuelve 5
`Round(1.1 ,1,0.5)`	Devuelve 1,5 En este ejemplo, el tamaño de step es 1 y la base del intervalo de step es 0,5. Los intervalos son ...**0.5 <= x <1.5**, 1.5 <= x <2.5, 2.5<= x <3.5...

Sign

Sign() devuelve 1, 0 o -1 dependiendo de si **x** es un número positivo, 0 o un número negativo.

Sintaxis:

```
Sign(x)
```

Tipo de datos que devuelve: numérico

Limitaciones:

Si no encuentra ningún valor numérico, devuelve NULL.

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
`Sign( 66 )`	Devuelve 1
`Sign( 0 )`	Devuelve 0
`Sign( - 234 )`	Devuelve -1

8.14 Funciones geoespaciales

Estas funciones se utilizan para manejar datos geoespaciales en visualizaciones de mapas. Qlik Sense sigue las especificaciones de GeoJSON para datos geoespaciales y admite lo siguiente:

- Punto
- Linestring
- Polígono
- Multipolígono

Si desea más información sobre las especificaciones de GeoJSON, vea:
GeoJSON.org

Descripción general de las funciones geoespaciales

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Hay dos categorías de funciones geoespaciales: las de agregación y de no agregación.

Las funciones de agregación toman un conjunto geométrico (puntos o áreas) como entrada y devuelven una geometría simple. Por ejemplo, se pueden fusionar múltiples áreas y trazarse en el mapa una única demarcación para la agregación.

Las funciones de no agregación toman una única geometría y devuelven una geometría. Por ejemplo, para la función GeoGetPolygonCenter(), si la geometría límite de un área se establece como entrada, se devuelve la geometría del punto (longitud y latitud) para el centro de esa área.

Las siguientes son funciones de agregación:

GeoAggrGeometry

GeoAggrGeometry() se utiliza para agregar una cantidad de áreas en un área más grande, por ejemplo, agregar una cantidad de subregiones a una región.

```
GeoAggrGeometry (field_name)
```

GeoBoundingBox

GeoBoundingBox() se utiliza para agregar una geometría a un área y calcular el recuadro de demarcación geoespacial más pequeño que contenga todas las coordenadas.

```
GeoBoundingBox (field_name)
```

GeoCountVertex

GeoCountVertex() se utiliza para hallar el número de vértices que contiene una geometría de un polígono.

```
GeoCountVertex(field_name)
```

GeoInvProjectGeometry

GeoInvProjectGeometry() se utiliza para agregar una geometría a un área y aplicar la inversa de una proyección.

```
GeoInvProjectGeometry(type, field_name)
```

GeoProjectGeometry

GeoProjectGeometry() se utiliza para agregar una geometría a un área y aplicar una proyección.

```
GeoProjectGeometry(type, field_name)
```

GeoReduceGeometry

GeoReduceGeometry() se utiliza para reducir el número de vértices de una geometría, y para agregar un número de áreas a un área, pero sin dejar de mostrar las líneas límite de las áreas individuales.

```
GeoReduceGeometry (geometry)
```

Las siguientes son funciones de no agregación:

GeoGetBoundingBox

GeoGetBoundingBox() se utiliza en scripts y expresiones de gráfico para calcular el recuadro de demarcación geoespacial más pequeño que contenga todas las coordenadas de una geometría.

```
GeoGetBoundingBox (geometry)
```

GeoGetPolygonCenter

GeoGetPolygonCenter() se utiliza en scripts y expresiones de gráfico para calcular y devolver el punto central de una geometría.

```
GeoGetPolygonCenter (geometry)
```

GeoMakePoint

GeoMakePoint() se utiliza en scripts y expresiones de gráfico para crear y etiquetar un punto con la latitud y la longitud.

```
GeoMakePoint (lat_field_name, lon_field_name)
```

GeoProject

GeoProject() se utiliza en scripts y expresiones de gráfico para aplicar una proyección a una geometría.

```
GeoProject (type, field_name)
```

GeoAggrGeometry

GeoAggrGeometry() se utiliza para agregar una cantidad de áreas en un área más grande, por ejemplo, agregar una cantidad de subregiones a una región.

Sintaxis:

```
GeoAggrGeometry(field_name)
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
field_name	Un campo o una expresión que se refieren a unc ampo que contiene la geometría que se ha de representar. Esto podría ser bien un punto (o conjunto de puntos) dándonos la longitud y la latitud, o un área.

Normalmente, **GeoAggrGeometry()** se puede usar para combinar datos de límites geoespaciales. Por ejemplo, podríamos tener códigos postales de áreas de una ciudad e ingresos por ventas en cada una de dichas áreas. Si el territorio de un vendedor cubre varias áreas de códigos postales, podría ser útil presentar el total de ventas por territorio de ventas, en vez de áreas individuales, y mostrar los resultados en una mapa codificado por colores.

GeoAggrGeometry() puede calcular la agregación de las geometrías de los barrios individualmente y generar la geometría del territorio fusionado en el modelo de datos. En ese caso, se ajustan después los límites del territorio de ventas, y cuando los datos se recargan las nuevas demarcaciones y cifras de ingresos se reflejan en el mapa.

Como **GeoAggrGeometry()** es una función de agregación, si la usa en el script se requiere una sentencia **LOAD** con una cláusula **Group by**.

*Las líneas limítrofes de mapas creados usando **GeoAggrGeometry()** son las de las áreas fusionadas. Si desea visualizar las líneas limítrofes e individuales de las áreas preagregadas, use **GeoReduceGeometry()**.*

Ejemplos:
Este ejemplo carga un archivo KML con datos de área, y después carga una tabla con los datos de área agregados.

```
[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() se utiliza para agregar una geometría a un área y calcular el recuadro de demarcación geoespacial más pequeño que contenga todas las coordenadas.

Un GeoBoundingBox se representa como una lista de cuatro valores: izquierda, derecha, arriba, abajo.

Sintaxis:

```
GeoBoundingBox(field_name)
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
field_name	Un campo o una expresión que se refieren a unc ampo que contiene la geometría que se ha de representar. Esto podría ser bien un punto (o conjunto de puntos) dándonos la longitud y la latitud, o un área.

GeoBoundingBox() agrega un conjunto de geometrías y devuelve cuatro coordenadas para el rectángulo más pequeño que contiene todas las coordenadas de esa geometría agregada.

Para visualizar el resultado en un mapa, transfiera la cadena resultante de cuatro coordenadas a un formato de polígono, etiquete el campo transferido con un formato geopoligonal, y arrastre y suelte ese campo en el objeto de mapa. Los cuadros rectangulares se mostrarán a continuación en la visualización de mapa.

GeoCountVertex

GeoCountVertex() se utiliza para hallar el número de vértices que contiene una geometría de un polígono.

Sintaxis:

```
GeoCountVertex(field_name)
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
field_name	Un campo o una expresión que se refieren a unc ampo que contiene la geometría que se ha de representar. Esto podría ser bien un punto (o conjunto de puntos) dándonos la longitud y la latitud, o un área.

GeoGetBoundingBox

GeoGetBoundingBox() se utiliza en scripts y expresiones de gráfico para calcular el recuadro de demarcación geoespacial más pequeño que contenga todas las coordenadas de una geometría.

Un cuadro delimitador geoespacial, creado por la función GeoBoundingBox() se representa como una lista de cuatro valores, izquierda, derecha, arriba, abajo.

Sintaxis:

```
GeoGetBoundingBox(field_name)
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
field_name	Un campo o una expresión que se refieren a unc ampo que contiene la geometría que se ha de representar. Esto podría ser bien un punto (o conjunto de puntos) dándonos la longitud y la latitud, o un área.

*No utilice la cláusula **Group by** en el editor de carga de datos con ésta y otras funciones geoespaciales de no agregación, porque esto ocasionará un error de carga.*

GeoGetPolygonCenter

GeoGetPolygonCenter() se utiliza en scripts y expresiones de gráfico para calcular y devolver el punto central de una geometría.

En algunos casos el requisito es trazar un punto en lugar de rellenar con color un área de un mapa. Si los datos geoespaciales existentes solo están disponibles en forma de geometría de área (por ejemplo, una frontera), use **GeoGetPolygonCenter()** para recuperar un par de longitud y latitud para el centro del área.

Sintaxis:

```
GeoGetPolygonCenter(field_name)
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
field_name	Un campo o una expresión que se refieren a unc ampo que contiene la geometría que se ha de representar. Esto podría ser bien un punto (o conjunto de puntos) dándonos la longitud y la latitud, o un área.

*No utilice la cláusula **Group by** en el editor de carga de datos con ésta y otras funciones geoespaciales de no agregación, porque esto ocasionará un error de carga.*

GeoInvProjectGeometry

GeoInvProjectGeometry() se utiliza para agregar una geometría a un área y aplicar la inversa de una proyección.

Sintaxis:

```
GeoInvProjectGeometry(type, field_name)
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
type	Un tipo de proyección utilizado para transformar la geometría del mapa. Esto puede tomar uno de dos valores: 'unidad', (opción por defecto), que deriva en una proyección de 1:1, o 'mercator', que utiliza la proyección estándar Mercator.
field_name	Un campo o una expresión que se refieren a unc ampo que contiene la geometría que se ha de representar. Esto podría ser bien un punto (o conjunto de puntos) dándonos la longitud y la latitud, o un área.

Ejemplo:

Ejemplo de script

Ejemplo	Resultado
En una sentencia `Load`: `GeoInvProjectGeometry ('mercator',AreaPolygon) as InvProjectGeometry`	La geometría cargada como **AreaPolygon** se transforma usando la transformación inversa de la proyección Mercator y se almacena como **InvProjectGeometry** para su uso en visualizaciones.

GeoMakePoint

GeoMakePoint() se utiliza en scripts y expresiones de gráfico para crear y etiquetar un punto con la latitud y la longitud. GeoMakePoint devuelve puntos en el orden de longitud y latitud.

Sintaxis:

```
GeoMakePoint(lat_field_name, lon_field_name)
```

Tipo de datos que devuelve: cadena, formateado [longitud, latitud]

Argumentos:

Argumentos

Argumento	Descripción
lat_field_name	Un campo o expresión que se refiere a un campo que representa la latitud del punto.
lon_field_name	Un campo o expresión que se refieren a un campo que representa la longitud del punto.

*No utilice la cláusula **Group by** en el editor de carga de datos con ésta y otras funciones geoespaciales de no agregación, porque esto ocasionará un error de carga.*

GeoProject

GeoProject() se utiliza en scripts y expresiones de gráfico para aplicar una proyección a una geometría.

Sintaxis:

```
GeoProject(type, field_name)
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
type	Un tipo de proyección utilizado para transformar la geometría del mapa. Esto puede tomar uno de dos valores: 'unidad', (opción por defecto), que deriva en una proyección de 1:1, o 'mercator', que utiliza la proyección web estándar Mercator.
field_name	Un campo o una expresión que se refieren a unc ampo que contiene la geometría que se ha de representar. Esto podría ser bien un punto (o conjunto de puntos) dándonos la longitud y la latitud, o un área.

*No utilice la cláusula **Group by** en el editor de carga de datos con ésta y otras funciones geoespaciales de no agregación, porque esto ocasionará un error de carga.*

Ejemplo:

Ejemplos de script

Ejemplo	Resultado
En una sentencia `Load`: `GeoProject ('mercator',Area) as GetProject`	La proyección Mercator se aplica a la geometría cargada como **Area**, y el resultado se almacena como **GetProject**.

GeoProjectGeometry

GeoProjectGeometry() se utiliza para agregar una geometría a un área y aplicar una proyección.

Sintaxis:

```
GeoProjectGeometry(type, field_name)
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
type	Un tipo de proyección utilizado para transformar la geometría del mapa. Esto puede tomar uno de dos valores: 'unidad', (opción por defecto), que deriva en una proyección de 1:1, o 'mercator', que utiliza la proyección web estándar Mercator.

Argumento	Descripción
field_name	Un campo o una expresión que se refieren a unc ampo que contiene la geometría que se ha de representar. Esto podría ser bien un punto (o conjunto de puntos) dándonos la longitud y la latitud, o un área.

Ejemplo:

Ejemplo	Resultado
En una sentencia `Load`: `GeoProjectGeometry ('mercator',AreaPolygon) as ProjectGeometry`	La geometría cargada como **AreaPolygon** se transforma usando la proyección Mercator y se almacena como **ProjectGeometry** para su uso en visualizaciones.

GeoReduceGeometry

GeoReduceGeometry() se utiliza para reducir el número de vértices de una geometría, y para agregar un número de áreas a un área, pero sin dejar de mostrar las líneas límite de las áreas individuales.

Sintaxis:

```
GeoReduceGeometry(field_name[, value])
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
field_name	Un campo o una expresión que se refieren a unc ampo que contiene la geometría que se ha de representar. Esto podría ser bien un punto (o conjunto de puntos) dándonos la longitud y la latitud, o un área.
value	La cantidad de reducción que se ha de aplicar a la geometría. El rango va de 0 a 1, donde 0 representa ninguna reducción y 1 representa la reducción máxima de vértices. *Usando un valor value de 0,9 o superior con un conjunto de datos complejo puede reducir el número de vértices a un nivel en el que la representación visual sea inexacta.*

GeoReduceGeometry() también realiza una función similar a la de **GeoAggrGeometry()** en cuanto que agrega una determinada cantidad de áreas a un área. La diferencia es que las líneas de límite individuales de los datos de agregación previa se muestran en el mapa si utiliza **GeoReduceGeometry()**.

Como **GeoReduceGeometry()** es una función de agregación, si la usa en el script se requiere una sentencia **LOAD** con una cláusula **Group by**.

Ejemplos:
Este ejemplo carga un archivo KML con datos de área, y después carga una tabla con los datos de área reducidos y agregados.

```
[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;
```

8.15 Funciones de interpretación

Las funciones de interpretación evalúan los contenidos de los distintos campos de texto o expresiones de entrada e imponen un formato específico de datos al valor numérico resultante. Con estas funciones podemos especificar el formato del número de acuerdo con el tipo de dato, incluyendo atributos tales como: separador decimal, separador de miles y formato de fecha.

Las funciones de interpretación devuelven todas ellas un valor dual con ambos valores, el de cadena y númerico, pero puede pensarse en ellas como que realizan una conversión de cadena a número. Las funciones toman el valor de texto de la expresión de entrada y generan un número que representa a la cadena.

Por el contrario, las funciones de formato toman expresiones numéricas y devuelven cadenas, especificando el formato del texto resultante.

Si no se utilizan funciones de interpretación, Qlik Sense interpreta los datos como una mezcla de números, fechas, horas, fecha-hora y cadenas de caracteres, empleando las configuraciones por defecto para el formato numérico, formato de fecha y formato de hora definidos por las variables de script y por el sistema operativo.

Todas las funciones de interpretación pueden utilizarse tanto en scripts de carga de datos como en expresiones de gráficos.

Todas las representaciones numéricas se dan con un punto decimal como separador decimal.

Descripción general de las funciones de interpretación

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Date#

Date# devuelve una expresión como una fecha en el formato especificado en el segundo argumento, si se proporciona este. Si el código de formato se omite, se usa la fecha por defecto del sistema operativo.

```
Date# (page 1283)(text[, format])
```

Interval#

Interval#() devuelve una expresión de texto como un intervalo de tiempo en el formato establecido por defecto en el sistema operativo, o en el formato especificado en el segundo argumento, si se proporciona este.

```
Interval# (page 1284)(text[, format])
```

Money#

Money#() convierte una cadena de texto en un valor de moneda, en el formato definido en el script de carga o el sistema operativo, a menos que se suministre una cadena de formato. Los símbolos de separador decimal o de miles personalizados son parámetros opcionales.

```
Money# (page 1284)(text[, format[, dec_sep[, thou_sep ] ] ])
```

Num#

Num#() interpreta una cadena de texto como un valor numérico, es decir, convierte la cadena de entrada en un número utilizando el formato especificado en el segundo parámetro. Si se omite el segundo parámetro, utiliza los separadores de decimal y de miles definidos en el script de carga de datos. Los símbolos de separador decimal o de miles personalizados son parámetros opcionales.

```
Num# (page 1286)(text[ , format[, dec_sep[ , thou_sep]]])
```

Text

Text() obliga a que la expresión se trate como texto, incluso si es posible una interpretación numérica.

```
Text(expr)
```

Time#

Time#() evalúa una expresión como un valor de tiempo, en el formato de tiempo establecido en el script de carga de datos o en el sistema operativo, a menos que se proporcione una cadena de formato..

```
Time# (page 1287)(text[, format])
```

Timestamp#

Timestamp#() evalúa una expresión como un valor de fecha y hora, en el formato de tiempo establecido en el script de carga de datos o en el sistema operativo, a menos que se proporcione una cadena de formato.

```
Timestamp# (page 1288)(text[, format])
```

Vea también:

- *Funciones de formato (page 1246)*

Date#

Date# devuelve una expresión como una fecha en el formato especificado en el segundo argumento, si se proporciona este.

Sintaxis:

```
Date#(text[, format])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
text	La cadena de texto que se ha de evaluar.
format	Cadena que describe el formato de la cadena de texto que se ha de evaluar. Si se omite, se utiliza el formato de fecha definido en las variables de sistema del script de carga de datos, o se utiliza el sistema operativo.

Ejemplos y resultados:
El ejemplo siguiente utiliza el formato de fecha **M/D/YYYY**. El formato de fecha se especifica en la sentencia **SET DateFormat** en la parte superior del script de carga de datos.

Agregue este script de ejemplo a su app y ejecútelo.

```
Load *,

Num(Date#(StringDate)) as Date;

LOAD * INLINE [

StringDate

8/7/97

8/6/1997

]
```

Si crea una tabla con **StringDate**y **Date** como dimensiones, los resultados son los siguientes:

Resultados

StringDate	Fecha
8/7/97	35649
8/6/1997	35648

Interval#

Interval#() devuelve una expresión de texto como un intervalo de tiempo en el formato establecido por defecto en el sistema operativo, o en el formato especificado en el segundo argumento, si se proporciona este.

Sintaxis:

```
Interval#(text[, format])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
text	La cadena de texto que se ha de evaluar.
format	Cadena que describe el formato de entrada esperado para utilizarlo en la conversión de la cadena a un intervalo numérico. Si se omite, se utilizan el formato breve de fecha, el formato de hora y el separador decimal establecidos en el sistema operativo.

La función **interval#** convierte un intervalo de tiempo de texto en un equivalente numérico.

Ejemplos y resultados:
Los ejemplos a continuación asumen las siguientes configuraciones del sistema:

- Formato de fecha reducido: YY-MM-DD
- Formato de hora: M/D/YY
- Separador de números decimales: ,

Resultados

Ejemplo	Resultado
`Interval#( A, 'D hh:mm' )` donde A='1 09:00'	1.375

Money#

Money#() convierte una cadena de texto en un valor de moneda, en el formato definido en el script de carga o el sistema operativo, a menos que se suministre una cadena de formato. Los símbolos de separador decimal o de miles personalizados son parámetros opcionales.

Sintaxis:

```
Money#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
text	La cadena de texto que se ha de evaluar.
format	Cadena que describe el formato de entrada esperado para utilizarlo en la conversión de la cadena a un intervalo numérico. Si se omite, se utilizará el formato de moneda establecido en el sistema operativo.
dec_sep	Cadena que especifica el separador numérico decimal. Si se omite, se utilizará el valor de separador decimal de moneda establecido en el script de carga de datos.
thou_sep	Cadena que especifica el separador numérico de miles. Si se omite, se utilizará el valor de separador de miles para moneda establecido en el script de carga de datos.

La función **money#** generalmente se comporta igual que la función **num#**, pero toma sus valores predeterminados de separador decimal y separador de miles de las variables de script para el formato de dinero o la configuración del sistema para moneda.

Ejemplos y resultados:
Los ejemplos a continuación asumen las dos configuraciones siguientes del sistema operativo:

- Configuración predeterminada del formato de dinero 1: kr # ##0,00
- Configuración predeterminada del formato de dinero 2: $ #,##0.00

```
Money#(A , '# ##0,00 kr' )
```
donde A=35 648,37 kr

Resultados

Resultados	Configuración 1	Configuración 2
Cadena	35 648.37 kr	35 648.37 kr
Número	35648.37	3564837

```
Money#( A, ' $#', '.', ',' )
```
donde A= $35,648.37

Resultados

Resultados	Configuración 1	Configuración 2
Cadena	$35,648.37	$35,648.37
Número	35648.37	35648.37

Num#

Num#() interpreta una cadena de texto como un valor numérico, es decir, convierte la cadena de entrada en un número utilizando el formato especificado en el segundo parámetro. Si se omite el segundo parámetro, utiliza los separadores de decimal y de miles definidos en el script de carga de datos. Los símbolos de separador decimal o de miles personalizados son parámetros opcionales.

Sintaxis:

```
Num#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Tipo de datos que devuelve: dual

La función **Num#()** devuelve un valor dual con la cadena y el valor numérico. La función toma la representación textual de la expresión introducida y genera un número. No cambia el formato del número: el resultado presenta el mismo formato que la cadena introducida.

Argumentos:

Argumentos

Argumento	Descripción
text	La cadena de texto que se ha de evaluar.
format	Cadena que especifica el formato de número utilizado en el primer parámetro. Si se omite el segundo parámetro, utiliza los separadores de decimal y de miles definidos en el script de carga de datos.
dec_sep	Cadena que especifica el separador numérico decimal. Si se omite, se utilizará el valor de la variable DecimalSep establecido en el script de carga de datos.
thou_sep	Cadena que especifica el separador numérico de miles. Si se omite, se utilizará el valor de la variable ThousandSep establecido en el script de carga de datos.

Ejemplos y resultados:
La tabla siguiente muestra el resultado de *Num#(A, '#', '.' , ',')* para diferentes valores de A.

Resultados

A	Representación de una cadena de texto	Valor numérico (aquí se muestra con un punto decimal)
35,648.31	35,648.31	35648.31
35 648.312	35 648.312	35648.312
35.648,3123	35.648,3123	-
35 648,31234	35 648,31234	-

Text

Text() obliga a que la expresión se trate como texto, incluso si es posible una interpretación numérica.

Sintaxis:

```
Text (expr)
```

Tipo de datos que devuelve: dual

Ejemplo:

```
Text( A )
```
donde A=1,234

Resultados

Cadena	Número
1234	-

Ejemplo:

```
Text( pi( ) )
```

Resultados

Cadena	Número
3.1415926535898	-

Time#

Time#() evalúa una expresión como un valor de tiempo, en el formato de tiempo establecido en el script de carga de datos o en el sistema operativo, a menos que se proporcione una cadena de formato..

Sintaxis:

```
time#(text[, format])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
text	La cadena de texto que se ha de evaluar.
format	Cadena que describe el formato de la cadena de texto que se ha de evaluar. Si se omite, se utilizan el formato breve de fecha, el formato de hora y el separador decimal establecidos en el sistema operativo.

Ejemplo:

- Configuración predeterminada de formato de hora 1: hh:mm:ss
- Configuración predeterminada de formato de hora 2: hh.mm.ss

```
time#( A )
```
donde A=09:00:00

Resultados

Resultados	Configuración 1	Configuración 2
Cadena:	09:00:00	09:00:00
Número:	0.375	-

Ejemplo:

- Configuración predeterminada de formato de hora 1: hh:mm:ss
- Configuración predeterminada de formato de hora 2: hh.mm.ss

```
time#( A, 'hh.mm' )
```
donde A=09.00

Resultados

Resultados	Configuración 1	Configuración 2
Cadena:	09.00	09.00
Número:	0.375	0.375

Timestamp#

Timestamp#() evalúa una expresión como un valor de fecha y hora, en el formato de tiempo establecido en el script de carga de datos o en el sistema operativo, a menos que se proporcione una cadena de formato.

Sintaxis:

```
timestamp#(text[, format])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
text	La cadena de texto que se ha de evaluar.

Argumento	Descripción
format	Cadena que describe el formato de la cadena de texto que se ha de evaluar. Si se omite, se utilizan el formato breve de fecha, el formato de hora y el separador decimal establecidos en el sistema operativo. ISO 8601 se admite para fechas-hora.

Ejemplo:

El ejemplo siguiente utiliza el formato de fecha **M/D/YYYY**. El formato de fecha se especifica en la sentencia **SET DateFormat** en la parte superior del script de carga de datos

Agregue este script de ejemplo a su app y ejecútelo.

```
Load *,
Timestamp(Timestamp#(String)) as TS;
LOAD * INLINE [
String
2015-09-15T12:13:14
1952-10-16T13:14:00+0200
1109-03-01T14:15
];
```

Si crea una tabla con **String** y **TS** como dimensiones, los resultados son los siguientes:

Resultados

Cadena	TS
2015-09-15T12:13:14	9/15/2015 12:13:14 PM
1952-10-16T13:14:00+0200	10/16/1952 11:14:00 AM
1109-03-01T14:15	3/1/1109 2:15:00 PM

8.16 Funciones inter-registro

Las funciones inter-registro se utilizan:

- En el script de carga de datos, cuando se necesite un valor de otros registros de datos cargados anteriormente para la evaluación del registro actual.
- En una expresión de gráfico, cuando se necesite otro valor del conjunto de datos de una visualización.

No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando se usa una función de gráfico interregistro en cualquiera de las expresiones de gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza una función de gráfico interregistro en una visualización o tabla, la clasificación de la visualización volverá a la entrada ordenada de la función interregistro. Esta limitación no se aplica a la función de script equivalente, si existe.

Las definiciones de expresiones de autorreferencia solo se pueden hacer de manera fiable en tablas con menos de 100 filas, pero esto puede variar dependiendo del hardware en el que se esté ejecutando el motor de Qlik.

Funciones de fila

Estas funciones solo pueden emplearse en expresiones de gráficos.

Above

Above() evalúa una expresión en una fila por encima de la fila actual dentro de un segmento de columna de una tabla. La fila para la que se calcula depende del valor de **offset**, si está presente, el valor predeterminado es la fila inmediatamente superior. Para los gráficos que no sean tablas, **Above()** evalúa la fila sobre la columna actual en el equivalente de tabla simple del gráfico.

```
Above - función de gráfico([TOTAL [<fld{,fld}>]] expr [ , offset [,count]])
```

Below

Below() evalúa una expresión en una fila debajo de la fila actual dentro de un segmento de columna de una tabla. La fila para la que se calcula depende del valor de **offset**, si está presente, el valor predeterminado es la fila inmediatamente inferior. Para los gráficos que no sean tablas, **Below()** evalúa la fila de debajo de la columna actual en el equivalente de tabla simple del gráfico.

```
Below - función de gráfico([TOTAL[<fld{,fld}>]] expression [ , offset [,count
]])
```

Bottom

Bottom() evalúa una expresión en la última fila (inferior) de un segmento de columna en una tabla. La fila para la que se calcula depende del valor de **offset**, si está presente, el valor predeterminado es la fila inferior. Para los gráficos que no sean tablas, la evaluación se realiza en la última fila de la columna actual en el equivalente de tabla simple del gráfico.

```
Bottom - función de gráfico([TOTAL[<fld{,fld}>]] expr [ , offset [,count ]])
```

Top

Top() evalúa una expresión en la primera fila (superior) de un segmento de columna en una tabla. La fila para la que se calcula depende del valor de **offset**, si está presente, el valor predeterminado es la fila superior. Para los gráficos que no sean tablas, la evaluación **Top()** se realiza en la primera fila de la columna actual en el equivalente de tabla simple del gráfico.

```
Top - función de gráfico([TOTAL [<fld{,fld}>]] expr [ , offset [,count ]])
```

NoOfRows

NoOfRows() devuelve el número de filas del segmento de columna actual de una tabla. Para los gráficos de mapa de bits, **NoOfRows()** devuelve el número de filas en el equivalente de tabla simple del gráfico.

```
NoOfRows - función de gráfico([TOTAL])
```

Funciones de columna

Estas funciones solo pueden emplearse en expresiones de gráficos.

Column

Column() devuelve el valor hallado en la columna correspondiente a **ColumnNo** en una tabla simple, sin tener en cuenta las dimensiones. Por ejemplo, **Column(2)** devuelve el valor de la segunda columna de medida.

```
Column - función de gráfico(ColumnNo)
```

Dimensionality

Dimensionality() devuelve el número de dimensiones de la fila actual. En el caso de las tablas pivotantes, la función devuelve el número total de columnas de dimensión que no tienen un contenido agregado, es decir, que no contienen sumas parciales o contenidos adicionales contraídos.

```
Dimensionality - función de gráfico ( )
```

Secondarydimensionality

SecondaryDimensionality() devuelve el número de filas de la tabla pivotante de dimensión que tienen contenido no agregado, es decir, que no contienen sumas parciales o agregados contraídos. Esta función es equivalente a la función **dimensionality()** para las dimensiones de tabla pivotante horizontales.

```
SecondaryDimensionality - función de gráfico ( )
```

Funciones de campo

FieldIndex

FieldIndex() devuelve la posición del valor de campo **value** en el campo **field_name** (por orden de carga).

```
FieldIndex (field_name , value)
```

FieldValue

FieldValue() devuelve el valor hallado en la posición **elem_no** del campo **field_name** (por orden de carga).

```
FieldValue (field_name , elem_no)
```

FieldValueCount

FieldValueCount() es una función de **entero** que devuelve el número de valores distintos de un campo.

```
FieldValueCount (field_name)
```

Funciones de la tabla pivotante

Estas funciones solo pueden emplearse en expresiones de gráficos.

After

After() devuelve el valor de una expresión evaluada con los valores de dimensión de una tabla pivotante tal y como aparecen en la columna tras la columna actual dentro de un segmento de fila en la tabla pivotante.

```
After - función de gráfico([TOTAL] expression [ , offset [,n]])
```

Before

Before() devuelve el valor de una expresión evaluada con los valores de dimensión de una tabla pivotante tal y como aparecen en la columna anterior a la columna actual dentro de un segmento de fila en la tabla pivotante.

```
Before - función de gráfico([TOTAL] expression [ , offset [,n]])
```

First

First() devuelve el valor de una expresión evaluada con los valores de dimensión de una tabla pivotante tal y como aparecen éstos en la primera columna del segmento de fila actual en la tabla pivotante. Esta función devuelve NULL en todos los tipos de gráfico excepto en las tablas pivotantes.

```
First - función de gráfico([TOTAL] expression [ , offset [,n]])
```

Last

Last() devuelve el valor de una expresión evaluada con los valores de dimensión de una tabla pivotante tal y como aparecen éstos en la última columna del segmento de fila actual en la tabla pivotante. Esta función devuelve NULL en todos los tipos de gráfico excepto en las tablas pivotantes.

```
Last - función de gráfico([TOTAL] expression [ , offset [,n]])
```

ColumnNo

ColumnNo() devuelve el número de la columna actual dentro del segmento de fila actual en una tabla pivotante. La primera columna es la número 1.

```
ColumnNo - función de gráfico([TOTAL])
```

NoOfColumns

NoOfColumns() devuelve el número de columnas que hay en el segmento de fila actual de una tabla pivotante.

```
NoOfColumns - función de gráfico([TOTAL])
```

Funciones inter-registro en el script de carga de datos

Exists

Exists() determina si un valor de campo específico ya se ha cargado en el campo en el script de carga de datos. La función devuelve TRUE o FALSE, así que se puede utilizar en la cláusula **where** de una sentencia **LOAD** o **IF**.

```
Exists (field_name [, expr])
```

LookUp

Lookup() busca en una tabla que ya está cargada y devuelve el valor de **field_name** correspondiente a la primera vez que aparece el valor **match_field_value** en el campo **match_field_name**. La tabla puede ser la actual u otra cargada anteriormente.

```
LookUp (field_name, match_field_name, match_field_value [, table_name])
```

Peek

Peek() devuelve el valor de un campo en una tabla para una fila que ya se ha cargado o que existe en la memoria interna. El número de fila se puede especificar, así como la tabla. Si no se especifica un número de fila, se utilizará el último registro cargado anteriormente.

```
Peek (field_name[, row_no[, table_name ] ])
```

Previous

Previous() halla el valor de la expresión **expr** utilizando datos del registro de entrada anterior que no se han descartado debido a una cláusula **where**. En el primer registro de una tabla interna, la función devolverá NULL.

```
Previous (page 1328)(expr)
```

Vea también:

- *Funciones de rango (page 1349)*

Above - función de gráfico

Above() evalúa una expresión en una fila por encima de la fila actual dentro de un segmento de columna de una tabla. La fila para la que se calcula depende del valor de **offset**, si está presente, el valor predeterminado es la fila inmediatamente superior. Para los gráficos que no sean tablas, **Above()** evalúa la fila sobre la columna actual en el equivalente de tabla simple del gráfico.

Sintaxis:

```
Above([TOTAL] expr [ , offset [,count]])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
offset	Especificar un offsetn mayor que 0 mueve la evaluación de la expresión n filas hacia arriba de la fila actual. Especificar un offset de 0 evaluará la expresión en la fila actual. Especificar un número offset negativo hace que la función Above opere como la función Below con el correspondiente número de offset positivo.
count	Especificando un tercer argumento **count** mayor que 1, la función devolverá un rango de valores **count**, uno por cada fila de la tabla **count** contando hacia arriba desde la celda original. De esta manera, la función puede utilizarse como argumento en cualquiera de las funciones de rango especiales. *Funciones de rango (page 1349)*
TOTAL	Si la tabla es unidimensional o si el cualificador **TOTAL** se utiliza como argumento, el segmento de columna actual es siempre igual a la columna completa.

En la primera fila de un segmento de columna, se devuelve un valor NULL, ya que no hay ninguna fila encima de él.

Un segmento de columna se define como un subconjunto consecutivo de celdas que tienen los mismos valores para las dimensiones de la ordenación actual. Las funciones inter-registro se calculan en el segmento de columna excluida la dimensión más a la derecha del gráfico de tabla simple equivalente. Si solo hay una dimensión en el gráfico, o si se especifica el cualificador TOTAL, la expresión se evalúa en la tabla completa.

Si la tabla o el equivalente de tabla tiene múltiples dimensiones verticales, el segmento de columna actual incluirá solo filas con los mismos valores que la fila actual en todas las columnas de dimensión, excepto para la columna que muestra la última dimensión en el orden de campos interno.

Limitaciones:

- Las llamadas recursivas devolverán NULL.
- No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función.

Ejemplos y resultados:

Example 1:

Visualización de la tabla para el ejemplo 1

Customer	Sum([Sales])	Above(Sum(Sales))	Sum(Sales)+Above(Sum(Sales))	Above offset 3	Higher?
	2566	-	-	-	-
Astrida	587	-	-	-	-
Betacab	539	587	1126	-	-
Canutility	683	539	1222	-	Higher
Divadip	757	683	1440	1344	Higher

En la captura de pantalla de la tabla que se muestra en este ejemplo, la visualización de la tabla se crea a partir de la dimensión **Customer** y las medidas `Sum(Sales)` y `Above(Sum(Sales))`.

La columna Above(Sum(Sales)) devuelve NULL para la fila **Customer** que contiene a **Astrida**, porque no hay ninguna fila encima. El resultado para la fila **Betacab** muestra el valor de Sum(Sales) para **Astrida**, el resultado de **Canutility** muestra el valor de **Sum(Sales)** para **Betacab**, etc.

Para la columna etiquetada como Sum(Sales)+Above(Sum(Sales)), la fila de **Betacab** muestra el resultado de la suma de los valores **Sum(Sales)** para las filas **Betacab** + **Astrida** (539+587). El resultado de la fila **Canutility** muestra el resultado de la suma de los valores **Sum(Sales)** para **Canutility** + **Betacab** (683+539).

La medida etiquetada como Above offset 3 creada usando la expresión `Sum(Sales)+Above(Sum(Sales), 3)` tiene el argumento **offset** fijado en 3, y tiene el efecto de tomar el valor de la fila tres filas por encima de la fila actual. Añade el valor **Sum(Sales)** del actual cliente **Customer** al valor de cliente **Customer** tres filas por encima. Los valores devueltos para las tres primeras filas **Customer** son nulos.

La tabla también muestra medidas más complejas: una creada desde `Sum(Sales)+Above(Sum(Sales))` y otra etiquetada como **Higher?**, la cual se crea desde `IF(Sum(Sales)>Above(Sum(Sales)), 'Higher')`.

Esta función también puede utilizarse en gráficos distintos de tablas, por ejemplo en gráficos de barras.

Para otros tipos de gráficos, convierta el gráfico a la tabla simple equivalente para que pueda interpretar fácilmente con qué fila está relacionada la función.

Example 2:

En las capturas de pantalla de las tablas que se muestran en este ejemplo, se han agregado más dimensiones a las visualizaciones: **Month** y **Product**. Para los gráficos con más de una dimensión, los resultados de las expresiones que contienen las funciones **Above**, **Below**, **Top** y **Bottom** dependen del orden en que Qlik Sense ordena las dimensiones de columna. Qlik Sense evalúa las funciones basándose en los segmentos de columna que resultan de la dimensión que se ordena en último lugar. El criterio de ordenación de columnas se controla en el panel de propiedades bajo **Ordenación** y no es necesariamente el orden en que las columnas aparecen en una tabla.

En la siguiente captura de pantalla de la visualización de la tabla para el ejemplo 2, la última dimensión es **Month**, por lo que la función **Above** se evalúa en función de los meses. Hay una serie de resultados para cada valor **Product** de cada mes (**Jan** a **Aug**): un segmento de columna. A esto le sigue una serie para el siguiente segmento de columna: para cada **Month** para el próximo **Product**. Habrá un segmento de columna por cada valor de **Customer** para cada **Product**.

Visualización de la tabla para el ejemplo 2

Customer	Product	Month	Sum([Sales])	Above(Sum(Sales))
			2566	-
Astrida	AA	Jan	46	-
Astrida	AA	Feb	60	46
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

Example 3:

En la captura de pantalla de la visualización de tabla para el ejemplo 3, la última dimensión ordenada es **Product**. Esto se hace moviendo la dimensión Product a la posición 3 en la pestaña Ordenar del panel de propiedades. La función **Above** se evalúa para cada **Product**, y como solo hay dos productos, **AA** y **BB**, solo hay un resultado no nulo en cada serie. En la fila **BB** para el mes **Jan**, el valor de **Above(Sum(Sales))** es 46. Para la fila **AA**, el valor es nulo. El valor de cada fila **AA** para cualquier mes siempre será nulo, puesto que no hay ningún valor de **Product** por encima de AA. La segunda serie se evalúa en **AA** y **BB** para el mes **Feb**, para el valor **Customer**, **Astrida**. Cuando se han evaluado todos los meses para **Astrida**, la secuencia se repite para el segundo **Customer**Betacab, y así sucesivamente.

Visualización de la tabla para el ejemplo 3

Customer	Product	Month	Sum([Sales])	Above(Sum(Sales))
			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

Ejemplo 4

Example 4:	**Resultado**
La función Above se puede utilizar como entrada a las funciones de rango. Por ejemplo: `RangeAvg (Above(Sum(Sales),1,3))`.	En los argumentos de la función Above(), offset está fijado en 1 y count está fijado en 3. La función halla los resultados de la expresión Sum(Sales) en las tres filas inmediatamente por encima de la fila actual en el segmento de columna (donde haya una fila). Estos tres valores se utilizan como entrada a la función RangeAvg(), que encuentra el valor promedio en el rango de números proporcionado. Una tabla con Customer como dimensión da los siguientes resultados para la expresión RangeAvg(). Astrida - Betacab 587 Canutility 563 Divadip: 603

Datos utilizados en los ejemplos:

```
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 '|');
```

Vea también:

- *Below - función de gráfico (page 1297)*
- *Bottom - función de gráfico (page 1301)*
- *Top - función de gráfico (page 1330)*
- *RangeAvg (page 1352)*

Below - función de gráfico

Below() evalúa una expresión en una fila debajo de la fila actual dentro de un segmento de columna de una tabla. La fila para la que se calcula depende del valor de **offset**, si está presente, el valor predeterminado es la fila inmediatamente inferior. Para los gráficos que no sean tablas, **Below()** evalúa la fila de debajo de la columna actual en el equivalente de tabla simple del gráfico.

Sintaxis:

```
Below([TOTAL] expr [ , offset [,count ]])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
offset	Especificar un **offset**n mayor que 1 mueve la evaluación de la expresión n filas más hacia abajo de la fila actual. Especificar un offset de 0 evaluará la expresión en la fila actual. Especificar un número de offset negativo hace que la función **Below** opere como la función **Above** con el correspondiente número de offset positivo.
count	Especificando un tercer parámetro **count** mayor que 1, la función devolverá un rango de valores **count**, uno por cada una de las filas de tabla **count** contando hacia abajo desde la celda original. De esta manera, la función puede utilizarse como argumento en cualquiera de las funciones de rango especiales. *Funciones de rango (page 1349)*
TOTAL	Si la tabla es unidimensional o si el cualificador **TOTAL** se utiliza como argumento, el segmento de columna actual es siempre igual a la columna completa.

En la última fila de un segmento de columna, se devuelve un valor NULL, ya que no hay ninguna fila debajo de él.

Un segmento de columna se define como un subconjunto consecutivo de celdas que tienen los mismos valores para las dimensiones de la ordenación actual. Las funciones inter-registro se calculan en el segmento de columna excluida la dimensión más a la derecha del gráfico de tabla simple equivalente. Si solo hay una dimensión en el gráfico, o si se especifica el cualificador TOTAL, la expresión se evalúa en la tabla completa.

Si la tabla o el equivalente de tabla tiene múltiples dimensiones verticales, el segmento de columna actual incluirá solo filas con los mismos valores que la fila actual en todas las columnas de dimensión, excepto para la columna que muestra la última dimensión en el orden de campos interno.

Limitaciones:

- Las llamadas recursivas devolverán NULL.
- No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de

gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función.

Ejemplos y resultados:

Example 1:

Visualización de la tabla para el ejemplo 1

Customer	Sum([Sales])	Below(Sum(Sales))	Sum(Sales)+Below(Sum(Sales))	Below + Offset 3	Higher
	2566	-	-	-	-
Astrida	587	539	1126	1344	Higher
Betacab	539	683	1222	-	-
Canutility	683	757	1440	-	-
Divadip	757	-	-	-	-

En la tabla que se muestra en la captura de pantalla del ejemplo 1, la visualización de la tabla se crea a partir de la dimensión **Customer** y las medidas `Sum(Sales)` y `Below(Sum(Sales))`.

La columna **Below(Sum(Sales))** devuelve NULL para la fila **Customer** que contiene a **Divadip**, porque no hay ninguna fila por debajo de la misma. El resultado para la fila **Canutility** muestra el valor de Sum(Sales) para **Divadip**, el resultado de **Betacab** muestra el valor para **Sum(Sales)** para **Canutility**, etc.

La tabla también muestra medidas más complejas, que se pueden ver en las columnas etiquetadas: `Sum(Sales)+Below(Sum(Sales))`, **Below +Offset 3** y **Higher?**. Estas expresiones funcionan tal como se describe en los párrafos siguientes.

Para la columna etiquetada como **Sum(Sales)+Below(Sum(Sales))**, la fila para **Astrida** muestra el resultado de la suma de los valores **Sum(Sales)** para las filas **Betacab** + **Astrida** (539+587). El resultado de la fila **Betacab** muestra el resultado de la suma de los valores **Sum(Sales)** para **Canutility** + **Betacab** (539+683).

La medida etiquetada como **Below +Offset 3** creada utilizando la expresión `Sum(Sales)+Below(Sum(Sales), 3)` tiene el argumento **offset** configurado como 3 y tiene el efecto de tomar el valor de la fila tres filas por debajo de la fila actual. Agrega el valor **Sum(Sales)** del actual **Customer** al valor de **Customer** tres filas por debajo. Los valores de las tres filas **Customer** más bajas son nulos.

La medida etiquetada como **Higher?** se crea a partir de la expresión:`IF(Sum(Sales)>Below(Sum(Sales)), 'Higher')`. Esto compara los valores de la fila actual en la medida **Sum(Sales)** con la fila de debajo de ella. Si la fila actual es un valor mayor, devuelve el texto "Higher".

Esta función también puede utilizarse en gráficos distintos de tablas, por ejemplo en gráficos de barras.

Para otros tipos de gráficos, convierta el gráfico a la tabla simple equivalente para que pueda interpretar fácilmente con qué fila está relacionada la función.

Para los gráficos con más de una dimensión, los resultados de las expresiones que contienen las funciones **Above**, **Below**, **Top** y **Bottom** dependen del orden en que Qlik Sense ordena las dimensiones de columna. Qlik Sense evalúa las funciones basándose en los segmentos de columna que resultan de la dimensión que se ordena en último lugar. El criterio de ordenación de columnas se controla en el panel de propiedades bajo **Ordenación** y no es necesariamente el orden en que las columnas aparecen en una tabla.Le remitimos al Ejemplo: 2 en la función **Above** para más detalles.

Ejemplo 2

Example 2:	**Resultado**
La función **Below** se puede utilizar como entrada a las funciones de rango. Por ejemplo: `RangeAvg (Below(Sum(Sales),1,3))`.	En los argumentos de la función **Below()**, offset está configurado como 1 y count como 3. La función halla los resultados de la expresión **Sum(Sales)** en las tres filas inmediatamente inferiores a la fila actual del segmento de columna (donde haya una fila). Estos tres valores se utilizan como entrada para la función RangeAvg(), que encuentra el valor promedio en el rango de números proporcionado. Una tabla con **Customer** como dimensión da los siguientes resultados para la expresión RangeAvg().
	Astrida 659.67 Betacab 720 Canutility 757 Divadip: -

Datos utilizados en los ejemplos:

```
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 '|');
```

Vea también:

- *Above - función de gráfico (page 1293)*
- *Bottom - función de gráfico (page 1301)*
- *Top - función de gráfico (page 1330)*
- *RangeAvg (page 1352)*

Bottom - función de gráfico

Bottom() evalúa una expresión en la última fila (inferior) de un segmento de columna en una tabla. La fila para la que se calcula depende del valor de **offset**, si está presente, el valor predeterminado es la fila inferior. Para los gráficos que no sean tablas, la evaluación se realiza en la última fila de la columna actual en el equivalente de tabla simple del gráfico.

Sintaxis:

```
Bottom([TOTAL] expr [ , offset [,count ]])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
offset	Especificando un **offset**n mayor que 1 mueve la evaluación de la expresión hacia arriba n filas por encima de la fila inferior. Especificar un número de offset negativo hace que la función **Bottom** opere como la función **Top** con el correspondiente número de offset positivo.
count	Especificando un tercer parámetro **count** mayor que 1, la función devolverá no uno, sino un rango de valores **count**, uno por cada una de las **count** últimas filas del actual segmento de columna. De esta manera, la función puede utilizarse como argumento en cualquiera de las funciones de rango especiales. *Funciones de rango (page 1349)*
TOTAL	Si la tabla es unidimensional o si el cualificador **TOTAL** se utiliza como argumento, el segmento de columna actual es siempre igual a la columna completa.

Un segmento de columna se define como un subconjunto consecutivo de celdas que tienen los mismos valores para las dimensiones de la ordenación actual. Las funciones inter-registro se calculan en el segmento de columna excluida la dimensión más a la derecha del gráfico de tabla simple equivalente. Si solo hay una dimensión en el gráfico, o si se especifica el cualificador TOTAL, la expresión se evalúa en la tabla completa.

Si la tabla o el equivalente de tabla tiene múltiples dimensiones verticales, el segmento de columna actual incluirá solo filas con los mismos valores que la fila actual en todas las columnas de dimensión, excepto para la columna que muestra la última dimensión en el orden de campos interno.

Limitaciones:

- Las llamadas recursivas devolverán NULL.
- No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función.

Ejemplos y resultados:

Visualización de la tabla para el ejemplo 1

Customer	Sum([Sales])	Bottom(Sum(Sales))	Sum(Sales)+Bottom(Sum(Sales))	Bottom offset 3
	2566	**757**	**3323**	**3105**
Astrida	587	757	1344	1126
Betacab	539	757	1296	1078
Canutility	683	757	1440	1222
Divadip	757	757	1514	1296

En la captura de pantalla de la tabla que se muestra en este ejemplo, la visualización de la tabla se crea a partir de la dimensión **Customer** y las medidas `Sum(Sales)` y `Bottom(Sum(Sales))`.

La columna **Bottom(Sum(Sales))** devuelve 757 para todas las filas porque este es el valor de la fila inferior: **Divadip**.

La tabla también muestra medidas más complejas: una creada desde `Sum(Sales)+Bottom(Sum(Sales))` y otra etiquetada como **Bottom offset 3**, la cual se crea utilizando la expresión `Sum(Sales)+Bottom(Sum(Sales), 3)` y tiene el argumento **offset** definido en 3. Agrega el valor **Sum(Sales)** de la fila actual al valor de la tercera fila debajo de la fila inferior, es decir, la fila actual más el valor de **Betacab**.

Ejemplo: 2

En las capturas de pantalla de las tablas que se muestran en este ejemplo, se han agregado más dimensiones a las visualizaciones: **Month** y **Product**. Para los gráficos con más de una dimensión, los resultados de las expresiones que contienen las funciones **Above**, **Below**, **Top** y **Bottom** dependen del orden en que Qlik Sense ordena las dimensiones de columna. Qlik Sense evalúa las funciones basándose en los segmentos de columna que resultan de la dimensión que se ordena en último lugar. El criterio de ordenación de columnas se controla en el panel de propiedades bajo **Ordenación** y no es necesariamente el orden en que las columnas aparecen en una tabla.

En la primera tabla, la expresión se evalúa basándose en **Month** y en la segunda tabla se evalúa conforme a **Product**. La medida **End value** contiene la expresión `Bottom(Sum(Sales))`. La fila inferior para **Month** es Dec, y el valor para Dec en ambos valores de **Product** mostrados en la captura de imagen es 22. (Algunas filas se han editado fuera de la captura de pantalla para ahorrar espacio.)

Primera tabla para el ejemplo 2. El valor de Bottom para la medida End value basado en Month (Dec).

Customer	Product	Month	Sum(Sales)	End value
			2566	-
Astrida	AA	Jan	46	22
Astrida	AA	Feb	60	22
Astrida	AA	Mar	70	22
Astrida	AA	Sep	78	22
Astrida	AA	Oct	12	22
Astrida	AA	Nov	78	22
Astrida	AA	Dec	22	22
Astrida	BB	Jan	46	22

Segunda tabla para el ejemplo 2. El valor de Bottom para la medida End value basada en Product (BB para Astrida).

Customer	Product	Month	Sum(Sales)	End value
			2566	-
Astrida	AA	Jan	46	46
Astrida	BB	Jan	46	46
Astrida	AA	Feb	60	60
Astrida	BB	Feb	60	60
Astrida	AA	Mar	70	70
Astrida	BB	Mar	70	70
Astrida	AA	Apr	13	13
Astrida	BB	Apr	13	13

Le remitimos al Ejemplo: 2 en la función **Above** para más detalles.

Ejemplo 3

Ejemplo: 3	**Resultado**
La función **Bottom** se puede utilizar como entrada a las funciones de rango. Por ejemplo: `RangeAvg (Bottom(Sum(Sales),1,3))`.	En los argumentos de la función **Bottom()**, offset está definido en 1 y count en 3. La función halla los resultados de la expresión **Sum(Sales)** en las tres filas superiores comenzando por la fila de encima de la fila inferior en el segmento de columna (a causa de offset=1), y las dos filas por encima de esta (donde haya una fila). Estos tres valores se utilizan como entrada para la función RangeAvg(), que encuentra el valor promedio en el rango de números proporcionado. Una tabla con **Customer** como dimensión da los siguientes resultados para la expresión RangeAvg().
	Astrida 659.67 Betacab 659.67 Canutility 659.67 Divadip: 659.67

```
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 '|');
```

Vea también:

Top - función de gráfico (page 1330)

Column - función de gráfico

Column() devuelve el valor hallado en la columna correspondiente a **ColumnNo** en una tabla simple, sin tener en cuenta las dimensiones. Por ejemplo, **Column(2)** devuelve el valor de la segunda columna de medida.

Sintaxis:

```
Column(ColumnNo)
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
ColumnNo	Número de columna de una columna en la tabla que contiene una medida. *La función Column() ignora las columnas de dimensión.*

Limitaciones:

- Las llamadas recursivas devolverán NULL.
- Si **ColumnNo** remite a una columna para la que no hay ninguna medida, se devuelve un valor NULL.
- No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función.

Ejemplos y resultados:

Ejemplo: Porcentaje de ventas totales

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
A	AA	15	10	150	505	29.70
A	AA	16	4	64	505	12.67
A	BB	9	9	81	505	16.04

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
B	BB	10	5	50	505	9.90
B	CC	20	2	40	505	7.92
B	DD	25	-	0	505	0.00
C	AA	15	8	120	505	23.76
C	CC	19	-	0	505	0.00

Ejemplo: Porcentaje de ventas para el cliente seleccionado

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
A	AA	15	10	150	295	50.85
A	AA	16	4	64	295	21.69
A	BB	9	9	81	295	27.46

Ejemplos y resultados

Ejemplos	Resultados
Order Value se añade a la tabla como una medida con la expresión: `Sum (UnitPrice*UnitSales)`. Total Sales Value se añade como una medida con la expresión: `Sum(TOTAL UnitPrice*UnitSales)` % Sales se añade como una medida con la expresión `100*Column(1)/Column(2)`	El resultado de Column(1) se toma de la columna Order Value, porque esta es la primera columna de medida. El resultado de Column(2) se toma de Total Sales Value, porque esta es la segunda columna de medida. Vea los resultados en la columna % Sales en el ejemplo *Porcentaje de ventas totales (page 1305)*.
Haga la selección Customer A.	La selección cambia el Total Sales Value, y por lo tanto el %Sales. Vea el ejemplo *Porcentaje de ventas para el cliente seleccionado (page 1306)*.

Datos utilizados en los ejemplos:

```
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
Canutility|AA|8|15
Canutility|CC||19
```

```
] (delimiter is '|');
```

Dimensionality - función de gráfico

Dimensionality() devuelve el número de dimensiones de la fila actual. En el caso de las tablas pivotantes, la función devuelve el número total de columnas de dimensión que no tienen un contenido agregado, es decir, que no contienen sumas parciales o contenidos adicionales contraídos.

Sintaxis:

```
Dimensionality ( )
```

Tipo de datos que devuelve: Entero

Limitaciones:

Esta función solo está disponible en los gráficos. Para todo tipo de gráficos, excepto la tabla pivotante, devolverá el número de dimensiones de todas las filas excepto el total, que será 0.

No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función.

Ejemplo: Expresión de gráfico usando Dimensionality

Ejemplo: Expresión de gráfico
La función **Dimensionality()** se puede utilizar con una tabla pivotante como una expresión de gráfico donde desea aplicar un formato de celda diferente según el número de dimensiones en una fila que tiene datos no agregados. Este ejemplo utiliza la función Dimensionality() para aplicar un color de fondo a las celdas de la tabla que coinciden con una condición determinada.

Script de carga

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear el ejemplo de expresión de gráfico a continuación.

```
ProductSales:

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 ',');
```

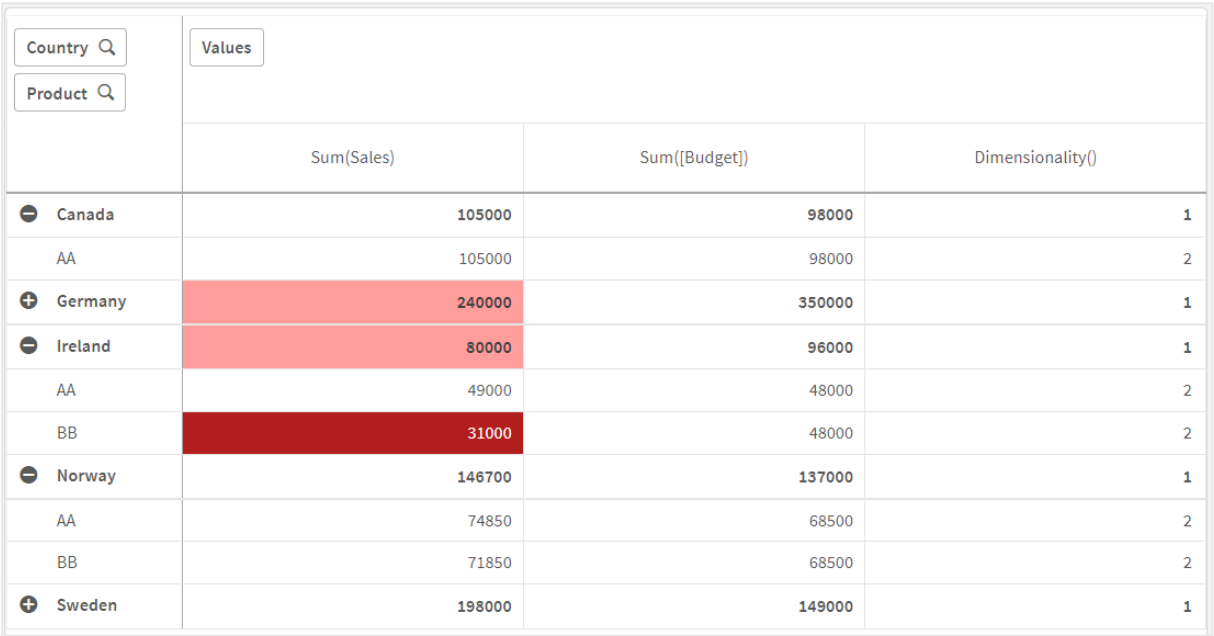
Expresión de gráfico

Cree una visualización de tabla pivotante en una hoja de Qlik Sense con **País** y **Producto** como dimensiones. Agregue **Sum(Sales)**, **Sum(Budget)** y **Dimensionality()** como medidas.

En el panel de **Propiedades**, escriba la siguiente expresión como **Expresión de color de fondo** para la medida **Sum(Sales)**.

```
If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156),
If(Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)
))
```

Resultado:



Country / Product	Sum(Sales)	Sum([Budget])	Dimensionality()
Canada	105000	98000	1
AA	105000	98000	2
Germany	240000	350000	1
Ireland	80000	96000	1
AA	49000	48000	2
BB	31000	48000	2
Norway	146700	137000	1
AA	74850	68500	2
BB	71850	68500	2
Sweden	198000	149000	1

Explicación

La expresión `If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156), If (Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)))` contiene sentencias condicionales que verifican el valor de dimensionalidad y Sum(Sales) y Sum(Budget) (la suma de ventas y suma de presupuesto) para cada producto. Si se cumplen las condiciones, se aplica un color de fondo al valor de Sum (Sales).

Exists

Exists() determina si un valor de campo específico ya se ha cargado en el campo en el script de carga de datos. La función devuelve TRUE o FALSE, así que se puede utilizar en la cláusula **where** de una sentencia **LOAD** o **IF**.

*También puede usar **Not Exists()** para determinar si un valor de campo no se ha cargado, pero se recomienda precaución si utiliza **Not Exists()** en una cláusula where. La función **Exists()** prueba las tablas previamente cargadas y los valores previamente cargados en la tabla actual. Por lo tanto, solo se cargará la primera instancia. Cuando se encuentra la segunda instancia, el valor ya está cargado. Vea los ejemplos para más información.*

Sintaxis:

```
Exists(field_name [, expr])
```

Tipo de datos que devuelve: Booleano

Argumentos:

Argumentos

Argumento	Descripción
field_name	El nombre del campo donde desea buscar un valor. Puede usar un nombre de campo explícito sin comillas. El campo ya debe haber sido cargado por el script. Eso significa que no puede hacer referencia a un campo que se cargue mediante una cláusula situada más abajo en el script.
expr	El valor que desea comprobar si existe. Puede usar un valor explícito o una expresión que haga referencia a uno o varios campos en la sentencia load actual. *No puede hacer referencia a campos que no estén incluidos en la sentencia load actual.* Este argumento es opcional. Si lo omite, la función verificará si el valor de **field_name** en el registro actual ya existe.

Ejemplos y resultados:

Ejemplo 1

```
Exists (Employee)
```

Devuelve -1 (True) si el valor del campo **Employee** en el registro actual ya existe en cualquier otro registro leído previamente que contenga ese campo.

Las sentencias `Exists (Employee, Employee)` y `Exists (Employee)` son equivalentes.

Ejemplo 2

```
Exists(Employee, 'Bill')
```

Devuelve -1 (True) si el valor del campo **'Bill'** se encuentra en el contenido actual del campo **Employee**.

Ejemplo 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;
```

Esto da como resultado una tabla que puede usar como una visualización de tabla utilizando las dimensiones Employee y Address.

La cláusula where, `where Exists (Employee)`, significa que solo se cargarán en la nueva tabla los nombres de la tabla Citizens que también estén en Employees. La sentencia Drop elimina la tabla Employees para evitar la confusión.

Resultados

Employee	Address
Bill	New York
John	Miami
Steve	Chicago

Ejemplo 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;
```

La cláusula where incluye not: `where not Exists (Employee)`.

Esto significa que solo se cargarán en la nueva tabla los nombres de la tabla Citizens que no estén en Employees.

Tenga en cuenta que hay dos valores para Lucy en la tabla Citizens, pero solo uno está incluido en la tabla de resultados. Cuando se carga la primera fila con el valor Lucy, este se incluye en el campo Employee. Por lo tanto, cuando se comprueba la segunda línea, el valor ya existe.

Resultados

Empleado	**Dirección**
Mary	London
Lucy	Madrid

Ejemplo 5

Este ejemplo muestra cómo cargar todos los valores.

```
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;
```

Para poder obtener todos los valores de Lucy, se han modificado dos cosas:

- Se ha insertado un load precedente a la tabla Employees donde Employee se ha renombrado a Name.
  `Load Employee As Name;`
- La condición Where en Citizens se ha cambiado a:
  `not Exists (Name, Employee).`

Esto crea campos para Name y Employee. Al comprobar la segunda fila de Lucy, todavía no existe en Name.

Resultados

Empleado	**Dirección**
Mary	London
Lucy	Madrid
Lucy	Paris

FieldIndex

FieldIndex() devuelve la posición del valor de campo **value** en el campo **field_name** (por orden de carga).

Sintaxis:

```
FieldIndex(field_name , value)
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	**Descripción**
field_name	Nombre del campo para el que se requiere el índice. Por ejemplo, la columna de una tabla. Debe especificarse como valor de cadena. Esto implica que el nombre del campo debe escribirse entre comillas simples.
value	El valor del campo **field_name**.

Limitaciones:

- Si no se logra encontrar **value** entre los valores del campo **field_name**, devuelve 0.
- No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función. Esta limitación no se aplica a la función de script equivalente.

Ejemplos y resultados:

Los ejemplos siguientes utilizan el campo: **First name** de la tabla **Names**.

Ejemplos y resultados

Ejemplos	**Resultados**
Añada los datos del ejemplo a su app y ejecútela.	Se carga la tabla **Names**, como en los datos de muestra.
Función de gráfico: En una tabla que contenga la dimensión First name, añada como medida.	
`FieldIndex ('First name','John')`	1, porque 'John' aparece en primer lugar en el orden de carga del campo **First name**. Observe que en un panel de filtrado **John** aparecería como el número 2 desde la parte superior puesto que está ordenado alfabéticamente y no por orden de carga.
`FieldIndex ('First name','Peter')`	4, porque **FieldIndex()** devuelve solo un valor, que es el primero en aparecer según el orden de carga.
Función de script: Dada la tabla **Names** cargada, como en los datos de ejemplo:	

`John1:`
`Load FieldIndex('First name','John') as MyJohnPos`
`Resident Names;`
	`MyJohnPos=1`, porque "John" aparece en primer lugar en el orden de carga del campo **First name**. Observe que en un panel de filtrado **John** aparecería como el número 2 desde la parte superior puesto que está ordenado alfabéticamente y no por orden de carga.

`Peter1:`
`Load FieldIndex('First name','Peter') as MyPeterPos`
`Resident Names;`
	`MyPeterPos=4`, porque **FieldIndex()** devuelve solo un valor, que es el primero en aparecer según el orden de carga.

Datos utilizados en el ejemplo:

```
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() devuelve el valor hallado en la posición **elem_no** del campo **field_name** (por orden de carga).

Sintaxis:

```
FieldValue(field_name , elem_no)
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
field_name	Nombre del campo para el que se requiere el valor. Por ejemplo, la columna de una tabla. Debe especificarse como valor de cadena. Esto implica que el nombre del campo debe escribirse entre comillas simples.
elem_no	El número de posición (elemento) del campo, siguiendo el orden de carga, para el que se devuelve el valor. Esto podría corresponder a la fila de una tabla, pero depende del orden en que los elementos (las filas) se carguen.

Limitaciones:

- Si **elem_no** es mayor que el número de valores, devuelve NULL.
- No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función. Esta limitación no se aplica a la función de script equivalente.

Ejemplo

Script de carga

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear el ejemplo a continuación.

```
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;
```

Crear una visualización

Cree una visualización de tabla en una hoja de Qlik Sense. Agregue los campos **First name**, **MyPos1** y **MyPos2** a la tabla.

Resultado

First name	MyPos1	MyPos2
Jane	John	Jane
John	John	Jane
Mark	John	Jane
Peter	John	Jane
Sue	John	Jane

Explicación

FieldValue('First name','1') da como resultado John como el valor de **MyPos1** para todos los nombres, porque John aparece primero en el orden de carga del campo **First name**. Obsérvese que en un panel de filtrado John aparecería como el número 2 empezando desde arriba, después de Jane, puesto que está ordenado alfabéticamente y no por orden de carga.

FieldValue('First name','5') da como resultado Jane como el valor de **MyPos2** para todos los nombres, porque Jane aparece en quinto lugar en el orden de carga del campo **First name**.

FieldValueCount

FieldValueCount() es una función de **entero** que devuelve el número de valores distintos de un campo.

Una carga parcial puede eliminar valores de los datos, que no se reflejarán en el número devuelto. El número devuelto corresponderá a todos los valores distintos que se cargaron en la carga inicial o en cualquier carga parcial posterior.

No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función. Esta limitación no se aplica a la función de script equivalente.

Sintaxis:

```
FieldValueCount(field_name)
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
field_name	Nombre del campo para el que se requiere el valor. Por ejemplo, la columna de una tabla. Debe especificarse como valor de cadena. Esto implica que el nombre del campo debe escribirse entre comillas simples.

Ejemplos y resultados:

Los ejemplos siguientes utilizan el campo **First name** de la tabla **Names**.

Ejemplos y resultados

Ejemplos	Resultados
Añada los datos del ejemplo a su app y ejecútela.	Se carga la tabla **Names**, como en los datos de muestra.
Función de gráfico: En una tabla que contenga la dimensión First name, añada como medida.	
`FieldValueCount('First name')`	5 como **Peter** aparece dos veces.
`FieldValueCount('Initials')`	6 como **Initials** solo tiene valores distintos.
Función de script: Dada la tabla **Names** cargada, como en los datos de ejemplo:	

`FieldCount1:`
`Load FieldValueCount('First name') as MyFieldCount1`
`Resident Names;`
	`MyFieldCount1=5`, porque "Peter" aparece dos veces.

`FieldCount2:`
`Load FieldValueCount('Initials') as MyInitialsCount1`
`Resident Names;`
	`MyFieldCount1=6`, porque "Initials" solo tiene valores distintos.

Datos utilizados en los ejemplos:

```
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() busca en una tabla que ya está cargada y devuelve el valor de **field_name** correspondiente a la primera vez que aparece el valor **match_field_value** en el campo **match_field_name**. La tabla puede ser la actual u otra cargada anteriormente.

Sintaxis:

```
lookup(field_name, match_field_name, match_field_value [, table_name])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
nombre_campo	Nombre del campo para el que se requiere el valor devuelto. El valor de entrada debe suministrarse como una cadena (por ejemplo, literales entrecomillados).
match_field_name	Nombre del campo que buscar en **match_field_value**. El valor de entrada debe suministrarse como una cadena (por ejemplo, literales entrecomillados).
match_field_value	Valor que buscar en el campo **match_field_name**.
table_name	Nombre de la tabla en la que buscar el valor. El valor de entrada debe suministrarse como una cadena (por ejemplo, literales entrecomillados). Si se omite **table_name**, se presupone la tabla actual.

Los argumentos sin comillas se refieren a la tabla actual. Para referirse a otras tablas, encierre un argumento entre comillas simples.

Limitaciones:

El orden de búsqueda es el orden de carga, a menos que la tabla sea el resultado de operaciones complejas, como uniones entre diversas tablas, en cuyo caso el orden no está bien definido. Tanto **field_name** como **match_field_name** deben ser campos en la misma tabla, especificados por **table_name**.

Si no encuentra ningún resultado, devuelve NULL.

Ejemplo

Script de carga

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear el ejemplo a continuación.

```
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;
```

Crear una visualización

Cree una visualización de tabla en una Qlik Sense de Qlik Cloud. Agregue los campos **ProductID**, **InvoiceID**, **CustomerID**, **Units** y **CategoryID** a la tabla.

Resultado

Tabla resultante

ProductID	InvoiceID	IDCliente	Unidades:	CategoryID
1	1	Astrida	8	1
2	1	Astrida	6	1
3	2	Betacab	10	2

ProductID	InvoiceID	IDCliente	Unidades:	CategoryID
3	3	Divadip	5	2
4	4	Divadip	10	3

Explicación

Los datos de muestra utilizan la función **Lookup()** de la siguiente forma:

```
Lookup('Category', 'ProductID', ProductID, 'ProductList')
```

La tabla **ProductList** se carga primero.

La función **Lookup()** se utiliza para construir la tabla **OrderData**. Especifica el tercer argumento como **ProductID**. Este es el campo para el que se buscará el valor en el segundo argumento **'ProductID'** en **ProductList**, como se indica por las comillas simples que lo incluyen.

La función devuelve el valor de '**Category**' (en la tabla **ProductList**), cargado como **CategoryID**.

La sentencia **drop** elimina la tabla **ProductList** del modelo de datos, porque no es necesaria, lo que nos deja la tabla **OrderData** resultante:

La función Lookup() es flexible y puede acceder a cualquier tabla cargada previamente. No obstante, es lento si se compara con la función Applymap().

Vea también:

ApplyMap (page 1342)

NoOfRows - función de gráfico

NoOfRows() devuelve el número de filas del segmento de columna actual de una tabla. Para los gráficos de mapa de bits, **NoOfRows()** devuelve el número de filas en el equivalente de tabla simple del gráfico.

Si la tabla o el equivalente de tabla tiene múltiples dimensiones verticales, el segmento de columna actual incluirá solo filas con los mismos valores que la fila actual en todas las columnas de dimensión, excepto para la columna que muestra la última dimensión en el orden de campos interno.

No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función.

Sintaxis:

```
NoOfRows([TOTAL])
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
TOTAL	Si la tabla es unidimensional o si el cualificador **TOTAL** se utiliza como argumento, el segmento de columna actual es siempre igual a la columna completa.

Ejemplo: Expresión de gráfico usando NoOfRows

Ejemplo: expresión de gráfico

Script de carga

Cargue los datos siguientes como una carga inline en el editor de carga de datos para crear los ejemplos de expresión del gráfico a continuación.

```
        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 '|');
```

Expresión de gráfico

Cree una visualización de tabla en una hojade Qlik Sense con **Region** y **SubRegion** como dimensiones. Añada `RowNo( )`, `NoOfRows()` y `NoOfRows(Total)` como medidas.

Resultado

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

Region	SubRegion	RowNo()	NoOfRows()	NoOfRows (Total)
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

Explicación

En este ejemplo, el orden de clasificación es por la primera dimensión Región. Como resultado, cada segmento de columna se compone de un grupo de regiones que tiene el mismo valor, por ejemplo, África.

La columna **RowNo()** muestra los números de fila para cada segmento de columna, por ejemplo, hay dos filas para la región de África. La numeración de filas comienza de nuevo en 1 para el siguiente segmento de columna, que es Americas.

La columna **NoOfRows()** cuenta el número de filas en cada segmento de columna, por ejemplo, Europa tiene tres filas en el segmento de columna.

La columna **NoOfRows(Total)** ignora las dimensiones debido al argumento `TOTAL` de `NoOfRows()` y cuenta las filas de la tabla.

Si la tabla se ordenara por la segunda dimensión, Subregión, los segmentos de columna se basarían en esa dimensión, por lo que la numeración de las filas cambiaría para cada subregión.

Vea también:

RowNo - función de gráfico (page 601)

Peek

Peek() devuelve el valor de un campo en una tabla para una fila que ya se ha cargado o que existe en la memoria interna. El número de fila se puede especificar, así como la tabla. Si no se especifica un número de fila, se utilizará el último registro cargado anteriormente.

La función peek() se utiliza con mayor frecuencia para encontrar los límites relevantes en una tabla previamente cargada, es decir, el primer valor o el último valor de un determinado campo. En la mayoría de los casos, este valor se almacena en una variable para su uso posterior, por ejemplo, como una condición en un bucle do-while.

Sintaxis:

```
Peek (

field_name
```

```
[, row_no[, table_name ] ])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
nombre_campo	Nombre del campo para el que se requiere el valor devuelto.El valor de entrada debe suministrarse como una cadena (por ejemplo, literales entrecomillados).
row_no	La fila de la tabla que especifica el campo requerido. Puede ser una expresión, pero debe resolverse en un entero. 0 denota el primer registro, 1 el segundo, y así sucesivamente. Los números negativos indican el orden desde el final de la tabla. -1 denota el último registro leído. Si no se especifica **row_no**, se presupone -1.
table_name	Una etiqueta de tabla que no finaliza en dos puntos. Si no se especifica **table_name**, se presupone la tabla actual. Si se usa fuera de la sentencia **LOAD** o se refiere a otra tabla, debe incluirse **table_name**.

Limitaciones:

La función solo puede devolver valores de registros ya cargados. Esto significa que en el primer registro de una tabla, una llamada que use -1 como row_no devolverá NULL.

Ejemplos y resultados:

Ejemplo 1

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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;
```

Tabla resultante

Código de empleado	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 porque `Peek('EmployeeCode',0, 'EmployeeDates')` devuelve el primer valor de EmployeeCode en la tabla EmployeeDates.

LastCode = 106 porque `Peek('EmployeeCode',-1, 'EmployeeDates')` devuelve el último valor de EmployeeCode en la tabla EmployeeDates.

Al sustituir el valor del argumento **row_no**, devuelve los valores de otras filas de la tabla, del siguiente modo:

`Peek('EmployeeCode',2, 'EmployeeDates')` devuelve el tercer valor, 103, de la tabla, como el primer FirstCode.

No obstante, tenga en cuenta que sin especificar la tabla como tercer argumento **table_name** en estos ejemplos, la función hace referencia a la tabla actual (en este caso, tabla interna).

Ejemplo 2

Si desea acceder a los datos más inferiores de una tabla, debe hacerlo en dos pasos: primero, cargue toda la tabla en una tabla temporal y luego vuelva a ordenarla cuando use **Peek()**.

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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 app with **ID**, **List**, and **Value** as the dimensions.

Tabla resultante

ID	Lista	Valor
1	3,4	4
1	3,4,6	6
1	3	3
2	1,11	11
2	1	1
3	7,8	8
3	7	7
5	2,78	78
5	2,78,13	13
5	2	2

La sentencia **IF()** se crea desde la tabla temporal T1.
`Peek('ID')` hace referencia al campo ID en la fila anterior en la tabla actual T2.
`Peek('List')` hace referencia al campo List en la fila anterior de la tabla T2, que se está construyendo a medida que se evalúa la expresión.

La sentencia se evalúa de la siguiente manera:
Si el valor actual de ID es el mismo que el valor anterior de ID, entonces escriba el valor de Peek('List') concatenado con el valor actual de Value. Si no, escriba el valor actual de Value solo.

Si Peek('List') ya contiene un resultado concatenado, el nuevo resultado de Peek('List') se le concatenará.

> *Observe la cláusula **Order by**. Esto especifica cómo se ordena la tabla (por ID en orden ascendente). Sin esto, la función Peek() utilizará cualquier orden arbitrario que tenga la tabla interna, lo que puede llevar a resultados impredecibles.*

Ejemplo 3

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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];
```

Tabla resultante

Amount	AmountMonthBefore	Month
1	4	2022-06
2	-	2022-01
3	2	2022-02
4	9	2022-05
7	3	2022-03
9	7	2022-04

El campo AmountMonthBefore retendrá el importe del mes anterior.

Aquí, se omiten los parámetros row_no y table_name, por lo que se utilizan los valores predeterminados. En este ejemplo, las siguientes tres llamadas a funciones son equivalentes:

- Peek(Amount)
- Peek(Amount,-1)
- Peek(Amount,-1,'Amounts')

Usar -1 como row_no significa que se usará el valor de la fila anterior. Al sustituir este valor, se pueden recuperar los valores de otras filas de la tabla:

Peek(Amount,2) devuelve el tercer valor de la tabla: 7.

Ejemplo 4

Los datos deben ordenarse correctamente para obtener los resultados correctos pero, desafortunadamente, no siempre es así. Además, la función Peek() no se puede utilizar para hacer referencia a datos que aún no se han cargado. Mediante el uso de tablas temporales y desplazándose varias veces por los datos, se pueden evitar estos problemas.

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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;
```

Explicación

La tabla inicial está ordenada según el mes, lo que significa que la función peek() en muchos casos devolvería la cantidad del producto incorrecto. De ahí que se deba reordenar esta tabla. Esto se hace en un segundo paso en los datos creando una nueva tabla tmp2Amounts. Observe la cláusula Order by. Ordena los registros primero por producto, luego por mes, en orden ascendente.

La función If() es necesaria ya que AmountMonthBefore solo debe calcularse si la fila anterior contiene los datos del mismo producto pero para el mes anterior. Al comparar el producto de la fila actual con el producto de la fila anterior, se puede validar esta condición.

Cuando se crea la segunda tabla, la primera tabla tmp1Amounts se elimina mediante una instrucción Drop Table.

Finalmente, se realiza una tercera pasada por los datos, pero ahora con los meses ordenados por orden inverso. De esta forma, también se puede calcular AmountMonthAfter.

Las cláusulas Order by especifican cómo se debe ordenar la tabla; sin estas, la función Peek() usaría cualquier orden arbitrario que tenga la tabla interna, lo que puede conducir a resultados impredecibles.

Resultado

Tabla resultante

Month	Producto	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
2022-05	A	4	7	-
2022-06	B	8	6	-

Ejemplo 5

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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;
```

Resultado

Tabla resultante

Trimestre	SumVal	AccSumVal
2003q1	65000	65000
2003q2	132450	197450
2003q3	131840	329290
2003q4	9000	338290
2004q1	5250	343540
2004q2	24240	367780
2004q3	88150	455930
2004q4	220650	676580

Explicación

La sentencia de carga **Load *, rangesum(SumVal,peek('AccSumVal')) as AccSumVal** incluye una llamada recursiva donde los valores anteriores se suman al valor actual. Esta operación sirve para calcular una acumulación de valores en el script.

Vea también:

Previous

Previous() halla el valor de la expresión **expr** utilizando datos del registro de entrada anterior que no se han descartado debido a una cláusula **where**. En el primer registro de una tabla interna, la función devolverá NULL.

Sintaxis:

```
Previous(expr)
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir. La expresión puede contener funciones **previous()** anidadas para acceder a los registros situados más atrás. Los datos se recuperan directamente de la fuente de entrada; esto también hace posible consultar los campos que no se hayan cargado en Qlik Sense, es decir, aunque no se hayan almacenado en la base de datos asociativa.

Limitaciones:

En el primer registro de una tabla interna, la función devuelve NULL.

Ejemplo:

Inserte lo siguiente en su script de carga:

```
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 '|');
```

Usando la función **Previous()** en la sentencia **Load**, podemos comparar el valor actual de Sales (Ventas) con el valor anterior y usarlo en un tercer campo, Increase (Incremento).

Tabla resultante

Mes	Ventas	Incremento
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
10	35	3
11	40	5
12	41	1

Top - función de gráfico

Top() evalúa una expresión en la primera fila (superior) de un segmento de columna en una tabla. La fila para la que se calcula depende del valor de **offset**, si está presente, el valor predeterminado es la fila superior. Para los gráficos que no sean tablas, la evaluación **Top()** se realiza en la primera fila de la columna actual en el equivalente de tabla simple del gráfico.

Sintaxis:

```
Top([TOTAL] expr [ , offset [,count ]])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.

Argumento	Descripción
offset	Especificar un **offset** de n mayor que 1 mueve la evaluación de la expresión hacia abajo n filas debajo de la fila superior. Especificar un número de desplazamiento (offset) negativo hace que la función **Top** opere como la función **Bottom** con el correspondiente número de offset positivo.
count	Especificando un tercer parámetro **count** mayor que 1, la función devolverá un rango de valores **count**, uno por cada una de las últimas filas **count** del actual segmento de columna. De esta manera, la función puede utilizarse como argumento en cualquiera de las funciones de rango especiales. *Funciones de rango (page 1349)*
TOTAL	Si la tabla es unidimensional o si el cualificador **TOTAL** se utiliza como argumento, el segmento de columna actual es siempre igual a la columna completa.

Un segmento de columna se define como un subconjunto consecutivo de celdas que tienen los mismos valores para las dimensiones de la ordenación actual. Las funciones inter-registro se calculan en el segmento de columna excluida la dimensión más a la derecha del gráfico de tabla simple equivalente. Si solo hay una dimensión en el gráfico, o si se especifica el cualificador TOTAL, la expresión se evalúa en la tabla completa.

Si la tabla o el equivalente de tabla tiene múltiples dimensiones verticales, el segmento de columna actual incluirá solo filas con los mismos valores que la fila actual en todas las columnas de dimensión, excepto para la columna que muestra la última dimensión en el orden de campos interno.

Limitaciones:

- Las llamadas recursivas devolverán NULL.
- No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función.

Ejemplos y resultados:

Ejemplo: 1

En la captura de pantalla de la tabla que se muestra en este ejemplo, la visualización de la tabla se crea a partir de la dimensión **Customer** y las medidas `Sum(Sales)` y `Top(Sum(Sales))`.

La columna **Top(Sum(Sales))** devuelve 587 para todas las filas porque este es el valor de la fila superior: **Astrida**.

La tabla también muestra medidas más complejas: una creada desde `Sum(Sales)+Top(Sum(Sales))` y otra etiquetada como **Top offset 3**, la cual se crea usando la expresión `Sum(Sales)+Top(Sum(Sales), 3)` y tiene el argumento **offset** configurado en 3. Agrega el valor **Sum(Sales)** de la fila actual al valor de la tercera fila desde la fila superior, es decir, la fila actual más el valor de **Canutility**.

Ejemplo 1

Top and Bottom

Customer	Sum(Sales)	Top(Sum(Sales))	Sum(Sales)+Top(Sum(Sales))	Top offset 3
Totals	2566	587	3153	3249
Astrida	587	587	1174	1270
Betacab	539	587	1126	1222
Canutility	683	587	1270	1366
Divadip	757	587	1344	1440

Ejemplo: 2

En las capturas de pantalla de las tablas que se muestran en este ejemplo, se han agregado más dimensiones a las visualizaciones: **Month** y **Product**. Para los gráficos con más de una dimensión, los resultados de las expresiones que contienen las funciones **Above**, **Below**, **Top** y **Bottom** dependen del orden en que Qlik Sense ordena las dimensiones de columna. Qlik Sense evalúa las funciones basándose en los segmentos de columna que resultan de la dimensión que se ordena en último lugar. El criterio de ordenación de columnas se controla en el panel de propiedades bajo **Ordenación** y no es necesariamente el orden en que las columnas aparecen en una tabla.

Primera tabla para el ejemplo 2. El valor de Top para la medida First value basado en Month (Jan).

Customer	Product	Month	Sum(Sales)	First value
			2566	-
Astrida	AA	Jan	46	46
Astrida	AA	Feb	60	46
Astrida	AA	Mar	70	46
Astrida	AA	Apr	13	46
Astrida	AA	May	78	46
Astrida	AA	Jun	20	46
Astrida	AA	Jul	45	46
Astrida	AA	Aug	65	46
Astrida	AA	Sep	78	46
Astrida	AA	Oct	12	46
Astrida	AA	Nov	78	46
Astrida	AA	Dec	22	46

Segunda tabla para el ejemplo 2. El valor de Top para la medida First value basada en Product (AA para Astrida).

Customer	Product	Month	Sum(Sales)	First value
			2566	-
Astrida	AA	Jan	46	46
Astrida	BB	Jan	46	46
Astrida	AA	Feb	60	60
Astrida	BB	Feb	60	60
Astrida	AA	Mar	70	70
Astrida	BB	Mar	70	70
Astrida	AA	Apr	13	13
Astrida	BB	Apr	13	13

Le remitimos al Ejemplo: 2 en la función **Above** para más detalles.

Ejemplo 3

Ejemplo: 3	**Resultado**
La función **Top** se puede utilizar como entrada a las funciones de rango. Por ejemplo: `RangeAvg (Top (Sum(Sales),1,3))`.	En los argumentos de la función **Top()**, offset se fija en 1 y count en 3. La función encuentra los resultados de la expresión **Sum(Sales)** en las tres filas que comienzan con la fila de debajo de la fila inferior en el segmento de columna (porque offset=1), y las dos filas debajo de eso (donde haya una fila). Estos tres valores se utilizan como entrada para la función RangeAvg(), que encuentra la media de los valores en el rango de números proporcionado. Una tabla con **Customer** como dimensión da los siguientes resultados para la expresión RangeAvg().
	Astrida 603 Betacab 603 Canutility 603 Divadip: 603

```
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 '|');
```

Vea también:

- *Bottom - función de gráfico (page 1301)*
- *Above - función de gráfico (page 1293)*
- *Sum - función de gráfico (page 354)*
- *RangeAvg (page 1352)*
- *Funciones de rango (page 1349)*

SecondaryDimensionality - función de gráfico

SecondaryDimensionality() devuelve el número de filas de la tabla pivotante de dimensión que tienen contenido no agregado, es decir, que no contienen sumas parciales o agregados contraídos. Esta función es equivalente a la función **dimensionality()** para las dimensiones de tabla pivotante horizontales.

Sintaxis:

```
SecondaryDimensionality( )
```

Tipo de datos que devuelve: Entero

Limitaciones:

- A menos que se utilice en tablas pivotantes, la función **SecondaryDimensionality** siempre devuelve 0.
- No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función.

After - función de gráfico

After() devuelve el valor de una expresión evaluada con los valores de dimensión de una tabla pivotante tal y como aparecen en la columna tras la columna actual dentro de un segmento de fila en la tabla pivotante.

Sintaxis:

```
after([TOTAL] expr [, offset [, count ]])
```

No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función.

Esta función devuelve NULL en todos los tipos de gráfico excepto en las tablas pivotantes.

Argumentos:

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
offset	Especificar un **offset** n mayor que 1 mueve la evaluación de la expresión n filas más a la derecha de la fila actual. Especificar un offset de 0 evaluará la expresión en la fila actual. Especificar un número offset negativo hace que la función **After** opere como la función **Before** con el correspondiente número de offset positivo.
count	Especificando un tercer parámetro **count** mayor que 1, la función devolverá un rango de valores, uno por cada fila de la tabla hasta el valor de **count**, contando hacia la derecha de la celda original.
TOTAL	Si la tabla es unidimensional o si el cualificador **TOTAL** se utiliza como argumento, el segmento de columna actual es siempre igual a la columna completa.

En la última columna de un segmento de fila se devolverá el valor NULL, puesto que ya no hay ninguna columna después de esta.

Si la tabla pivotante tiene múltiples dimensiones horizontales, el segmento de fila actual incluirá solo columnas con los mismos valores que la columna actual en todas las filas de dimensión excepto la fila que muestra la última dimensión horizontal del orden de campo. El criterio de ordenación entre campos para dimensiones horizontales en tablas pivotantes viene definido simplemente por el orden de las dimensiones de arriba a abajo..

Ejemplo:

```
after( sum( Sales ))
after( sum( Sales ), 2 )
after( total sum( Sales ))
```

`rangeavg (after(sum(x),1,3))` devuelve una media de los tres resultados de la función **sum(x)** evaluada en las tres columnas situadas inmediatamente a la derecha de la columna actual.

Before - función de gráfico

Before() devuelve el valor de una expresión evaluada con los valores de dimensión de una tabla pivotante tal y como aparecen en la columna anterior a la columna actual dentro de un segmento de fila en la tabla pivotante.

Sintaxis:

```
before([TOTAL] expr [, offset [, count]])
```

Esta función devuelve NULL en todos los tipos de gráfico excepto en las tablas pivotantes.

No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función.

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
offset	Especificar un **offset** n mayor que 1 mueve la evaluación de la expresión n filas más a la izquierda de la fila actual. Especificar un offset de 0 evaluará la expresión en la fila actual. Especificar un número offset negativo hace que la función **Before** opere como la función **After** con el correspondiente número de offset positivo.
count	Especificando un tercer parámetro **count** mayor que 1, la función devolverá un rango de valores, uno por cada fila de la tabla hacia arriba hasta el valor de **count**, contando hacia la izquierda desde la celda original.
TOTAL	Si la tabla es unidimensional o si el cualificador **TOTAL** se utiliza como argumento, el segmento de columna actual es siempre igual a la columna completa.

En la primera columna de un segmento de fila devolverá un valor NULL, puesto que ya no hay ninguna columna antes de esta.

Si la tabla pivotante tiene múltiples dimensiones horizontales, el segmento de fila actual incluirá solo columnas con los mismos valores que la columna actual en todas las filas de dimensión excepto la fila que muestra la última dimensión horizontal del orden de campo.El criterio de ordenación entre campos para dimensiones horizontales en tablas pivotantes viene definido simplemente por el orden de las dimensiones de arriba a abajo..

Ejemplos:

```
before( sum( Sales ))
before( sum( Sales ), 2 )
before( total sum( Sales ))
```

`rangeavg (before(sum(x),1,3))` devuelve una media de los tres resultados de la función **sum(x)** evaluada en las tres columnas situadas inmediatamente a la izquierda de la columna actual.

First - función de gráfico

First() devuelve el valor de una expresión evaluada con los valores de dimensión de una tabla pivotante tal y como aparecen éstos en la primera columna del segmento de fila actual en la tabla pivotante. Esta función devuelve NULL en todos los tipos de gráfico excepto en las tablas pivotantes.

No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función.

Sintaxis:

```
first([TOTAL] expr [, offset [, count]])
```

Argumentos:

Argumentos

Argumento	Descripción
expression	La expresión o el campo que contiene los datos que se han de medir.
offset	Especificar un **offset** n mayor que 1 mueve la evaluación de la expresión n filas más a la derecha de la fila actual. Especificar un offset de 0 evaluará la expresión en la fila actual. Especificar un número offset negativo hace que la función **First** opere como la función **Last** con el correspondiente número de offset positivo.
count	Especificando un tercer parámetro **count** mayor que 1, la función devolverá un rango de valores, uno por cada fila de la tabla hasta el valor de **count**, contando hacia la derecha de la celda original.
TOTAL	Si la tabla es unidimensional o si el cualificador **TOTAL** se utiliza como argumento, el segmento de columna actual es siempre igual a la columna completa.

Si la tabla pivotante tiene múltiples dimensiones horizontales, el segmento de fila actual incluirá solo columnas con los mismos valores que la columna actual en todas las filas de dimensión excepto la fila que muestra la última dimensión horizontal del orden de campo.El criterio de ordenación entre campos para dimensiones horizontales en tablas pivotantes viene definido simplemente por el orden de las dimensiones de arriba a abajo..

Ejemplos:

```
first( sum( Sales ))
first( sum( Sales ), 2 )
first( total sum( Sales )
```

`rangeavg (first(sum(x),1,5))` devuelve una media de los resultados de la función **sum(x)** evaluada en las cinco columnas situadas más a la izquierda del segmento de fila actual.

Last - función de gráfico

Last() devuelve el valor de una expresión evaluada con los valores de dimensión de una tabla pivotante tal y como aparecen éstos en la última columna del segmento de fila actual en la tabla pivotante. Esta función devuelve NULL en todos los tipos de gráfico excepto en las tablas pivotantes.

No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función.

Sintaxis:

```
last([TOTAL] expr [, offset [, count]])
```

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
offset	Especificar un **offset** n mayor que 1 mueve la evaluación de la expresión n filas más a la izquierda de la fila actual. Especificar un offset de 0 evaluará la expresión en la fila actual. Especificar un número offset negativo hace que la función **First** opere como la función **Last** con el correspondiente número de offset positivo.
count	Especificando un tercer parámetro **count** mayor que 1, la función devolverá un rango de valores, uno por cada fila de la tabla hacia arriba hasta el valor de **count**, contando hacia la izquierda desde la celda original.
TOTAL	Si la tabla es unidimensional o si el cualificador **TOTAL** se utiliza como argumento, el segmento de columna actual es siempre igual a la columna completa.

Si la tabla pivotante tiene múltiples dimensiones horizontales, el segmento de fila actual incluirá solo columnas con los mismos valores que la columna actual en todas las filas de dimensión excepto la fila que muestra la última dimensión horizontal del orden de campo.El criterio de ordenación entre campos para dimensiones horizontales en tablas pivotantes viene definido simplemente por el orden de las dimensiones de arriba a abajo..

Ejemplo:

```
last( sum( Sales ))
last( sum( Sales ), 2 )
last( total sum( Sales )
```

`rangeavg (last(sum(x),1,5))` devuelve una media de los resultados de la función **sum(x)** evaluada en las cinco columnas situadas más a la derecha del segmento de fila actual.

ColumnNo - función de gráfico

ColumnNo() devuelve el número de la columna actual dentro del segmento de fila actual en una tabla pivotante. La primera columna es la número 1.

Sintaxis:

```
ColumnNo([total])
```

Argumentos:

Argumentos

Argumento	Descripción
TOTAL	Si la tabla es unidimensional o si el cualificador **TOTAL** se utiliza como argumento, el segmento de columna actual es siempre igual a la columna completa.

Si la tabla pivotante tiene múltiples dimensiones horizontales, el segmento de fila actual incluirá solo columnas con los mismos valores que la columna actual en todas las filas de dimensión excepto la fila que muestra la última dimensión horizontal del orden de campo.El criterio de ordenación entre campos para dimensiones horizontales en tablas pivotantes viene definido simplemente por el orden de las dimensiones de arriba a abajo..

No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función.

Ejemplo:

```
if( ColumnNo( )=1, 0, sum( Sales ) / before( sum( Sales )))
```

NoOfColumns - función de gráfico

NoOfColumns() devuelve el número de columnas que hay en el segmento de fila actual de una tabla pivotante.

> *No se permite ordenar por valores Y en gráficos ni ordenar por columnas de expresión en tablas cuando esta función de gráfico se usa en cualquiera de las expresiones del gráfico. Estas alternativas de ordenación están por lo tanto automáticamente deshabilitadas. Cuando utiliza esta función de gráfico en una visualización o tabla, el orden de la visualización volverá a la entrada ordenada de esta función.*

Sintaxis:

```
NoOfColumns([total])
```

Argumentos:

Argumentos

Argumento	Descripción
TOTAL	Si la tabla es unidimensional o si el cualificador **TOTAL** se utiliza como argumento, el segmento de columna actual es siempre igual a la columna completa.

Si la tabla pivotante tiene múltiples dimensiones horizontales, el segmento de fila actual incluirá solo columnas con los mismos valores que la columna actual en todas las filas de dimensión excepto en la fila que muestra la última dimensión en el orden de campo.El criterio de ordenación entre campos para dimensiones horizontales en tablas pivotantes viene definido simplemente por el orden de las dimensiones de arriba a abajo..

Ejemplo:

```
if( ColumnNo( )=NoOfColumns( ), 0, after( sum( Sales )))
```

8.17 Funciones lógicas

En esta sección se describen funciones de gestión de operaciones lógicas. Todas las funciones pueden utilizarse tanto en el script de carga de datos como en las expresiones de gráficos.

IsNum

Devuelve -1 (True) si la expresión puede interpretarse como un número, de lo contrario devuelve 0 (False).

```
IsNum( expr )
```

IsText

Devuelve -1 (True) si la expresión tiene una representación de texto, de lo contrario devuelve 0 (False).

```
IsText( expr )
```

*Tanto **IsNum** como **IsText** devuelven 0 si la expresión es NULL.*

Ejemplo:

El ejemplo a continuación carga una tabla inline con valores de texto y valores numéricos, y añade dos campos para verificar si el valor es un valor numérico o de texto.

```
Load *, IsNum(Value), IsText(Value)
Inline [
Value
23
Green
Blue
12
33Red];
```

La tabla resultante tiene el siguiente aspecto:

Resulting table

Value	IsNum(Value)	IsText(Value)
23	-1	0
Green	0	-1
Blue	0	-1
12	-1	0
33Red	0	-1

8.18 Funciones de correspondencia

En esta sección se describen funciones de gestión de tablas de correspondencia. Las tablas de correspondencias pueden utilizarse para reemplazar valores de campo o nombres de campo durante la ejecución de script.

Las funciones de correspondencia solo pueden utilizarse en el script de carga de datos

Descripción general de las funciones de correspondencia

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

ApplyMap

La función de script **ApplyMap** sirve para enlazar (o mapear) el resultado de una expresión con una tabla de correspondencia previamente cargada.

```
ApplyMap ('mapname', expr [ , defaultexpr ] )
```

MapSubstring

La función de script **MapSubstring** sirve para hacer corresponder (mapear) partes de cualquier expresión con una tabla de correspondencias previamente cargada. La correspondencia (o mapeado) es sensible a mayúsculas y no repetitivo y las subcadenas se asocian de izquierda a derecha.

```
MapSubstring ('mapname', expr)
```

ApplyMap

La función de script **ApplyMap** sirve para enlazar (o mapear) el resultado de una expresión con una tabla de correspondencia previamente cargada.

Sintaxis:

```
ApplyMap('map_name', expression [ , default_mapping ] )
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
map_name	El nombre de una tabla de asignación que se creó previamente a través de la sentencia **mapping load** o **mapping select**. Su nombre debe ir entre comillas simples. *Si utiliza esta función en una variable expandida de macros y hace referencia a una tabla de asignación que no existe, la llamada de función falla y no se crea un campo.*
expression	La expresión, el resultado de lo que debe mapearse o asignarse.
default_ mapping	Si se establece, este valor se usará como valor predeterminado si la tabla de asignación no contiene un valor coincidente para expression. Si no se establece, el valor de expression se devolverá tal cual es.

El campo de salida de ApplyMap no debe tener el mismo nombre que uno de sus campos de entrada. Esto podría ocasionar resultados inesperados. Ejemplo de no utilización: `ApplyMap ('Map', A) as A`.

Ejemplo:

En este ejemplo, se ha cargado una lista de comerciales con un código de país que representa su país de residencia. Se utiliza una tabla que asigna un código de país a un país para reemplazar el código de país por el nombre del país. Solo tres países están definidos en la tabla de asignación, otros códigos de país están asignados a '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';
```

La tabla resultante (Vendedores) tiene el siguiente aspecto:

Resulting table

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

MapSubstring

La función de script **MapSubstring** sirve para hacer corresponder (mapear) partes de cualquier expresión con una tabla de correspondencias previamente cargada. La correspondencia (o mapeado) es sensible a mayúsculas y no repetitivo y las subcadenas se asocian de izquierda a derecha.

Sintaxis:

```
MapSubstring('map_name', expression)
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
map_name	El nombre de una tabla de asignación previamente leída por una sentencia **mapping load** o **mapping select**. El nombre debe ir entre comillas simples rectas. *Si utiliza esta función en una variable expandida de macros y hace referencia a una tabla de asignación que no existe, la llamada de función falla y no se crea un campo.*
expression	La expresión cuyo resultado debe ser enlazado por las subcadenas.

Ejemplo:

En este ejemplo cargamos una lista de modelos de producto. Cada modelo tiene un conjunto de atributos que se describen mediante un código compuesto. Usando la tabla de asignación con MapSubstring, podemos ampliar los códigos de atributo a una descripción.

```
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';
```

La tabla resultante tiene el siguiente aspecto:

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 Funciones matemáticas

En esta sección se describen funciones para constantes matemáticas y valores Booleanos. Estas funciones no tienen ningún parámetro, pero los paréntesis siempre son necesarios.

Todas las funciones pueden utilizarse tanto en el script de carga de datos como en las expresiones de gráficos.

e
La función devuelve la base de los logaritmos naturales, **e** (2.71828...).

```
e( )
```

false
La función devuelve un valor dual con valor de texto 'False' y valor numérico 0, que se puede usar como un false lógico en expresiones.

```
false( )
```

pi
La función devuelve el valor de π (3.14159...).

```
pi( )
```

rand
La función devuelve un número aleatorio entre 0 y 1. Esto se puede usar para crear datos de muestra.

```
rand( )
```

Ejemplo:

Este script a modo de ejemplo crea una tabla de 1000 registros con caracteres en mayúscula seleccionados de manera aleatoria, es decir, caracteres en el rango de 65 a 91 (65+26).

```
Load
     Chr( Floor(rand() * 26) + 65) as UCaseChar,
     RecNo() as ID
     Autogenerate 1000;
```

true
La función devuelve un valor dual con valor de texto 'True' y valor numérico -1, que se puede usar como un true lógico en expresiones.

```
true( )
```

8.20 Funciones NULL

Esta sección describe funciones para devolver o detectar valores NULL.

Todas las funciones pueden utilizarse tanto en el script de carga de datos como en las expresiones de gráficos.

Vista general de las funciones NULL

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

EmptyIsNull
La función **EmptyIsNull** convierte cadenas vacías en NULL. Es decir, devuelve NULL si el parámetro es una cadena vacía, de lo contrario devuelve el parámetro.

```
EmptyIsNull (expr )
```

IsNull
La función **IsNull** comprueba si el valor de una expresión es NULL y, si lo es, devuelve -1 (True), de lo contrario, devuelve 0 (False).

```
IsNull (expr )
```

Null
La función **Null** devuelve un valor NULL.

```
NULL( )
```

EmptyIsNull

La función **EmptyIsNull** convierte cadenas vacías en NULL. Es decir, devuelve NULL si el parámetro es una cadena vacía, de lo contrario devuelve el parámetro.

Sintaxis:

```
EmptyIsNull(exp )
```

Ejemplos y resultados:

Ejemplos de script

Ejemplo	Resultado
`EmptyIsNull(AdditionalComments)`	Esta expresión devolverá como nulo cualquier valor de cadena vacía del campo *AdditionalComments*, en lugar de cadenas vacías. Se devuelven cadenas y números no vacíos.
`EmptyIsNull(PurgeChar (PhoneNumber, ' -()'))`	Esta expresión eliminará los guiones, espacios y paréntesis del campo *PhoneNumber* . Si no quedan caracteres, la función EmptyIsNull devuelve la cadena vacía como nula; un número de teléfono vacío es lo mismo que ningún número de teléfono.

IsNull

La función **IsNull** comprueba si el valor de una expresión es NULL y, si lo es, devuelve -1 (True), de lo contrario, devuelve 0 (False).

Sintaxis:

```
IsNull(expr )
```

*Una cadena con longitud cero no se considera NULL y hará que **IsNull** devuelva False.*

Ejemplo: Script de carga de datos

En este ejemplo, se carga una tabla inline con cuatro filas, donde las tres primeras líneas no contienen nada, o bien contienen - o 'NULL' en Value o en la columna. Convertimos estos valores en representaciones verdaderas del valor NULL con el medio precedente **LOAD** usando la función **Null**.

El **LOAD** precedente añade un campo que verifica si el valor es NULL, usando la función **IsNull**.

```
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,Value];
```

Esta es la tabla resultante. En la columna ValueNullConv, los valores NULL están representados por -.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

NULL

La función **Null** devuelve un valor NULL.

Sintaxis:

```
Null( )
```

Ejemplo: Script de carga de datos

En este ejemplo, se carga una tabla inline con cuatro filas, donde las tres primeras líneas no contienen nada, o bien contienen - o 'NULL' en Value o en la columna. Queremos convertir estos valores en representaciones de valores NULL verdaderos.

La sentencia **LOAD** precedente en el medio realiza la conversión utilizando la función **Null**.

El primer **LOAD** precedente añade un campo que comprueba si el valor es NULL, solo con fines ilustrativos en este ejemplo.

```
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,Value];
```

Esta es la tabla resultante. En la columna ValueNullConv, los valores NULL están representados por -.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

8.21 Funciones de rango

Las funciones de rango son funciones que toman un conjunto de valores y producen un único valor como resultado. Todas las funciones de rango pueden utilizarse tanto en el script de carga de datos como en las expresiones de gráficos.

Por ejemplo, en una visualización, una función de rango puede calcular un valor único a partir de un conjunto entre registros. En el script de carga de datos, una función de rango puede calcular un único valor a partir de un conjunto de valores en una tabla interna.

Las funciones de rango reemplazan a las siguientes funciones numéricas generales: ***numsum****,* ***numavg****,* ***numcount****,* ***nummin*** *y* ***nummax****, que ahora han quedado obsoletas.*

Funciones de rango básicas

RangeMax

RangeMax() devuelve los valores numéricos más altos encontrados dentro de la expresión o campo.

```
RangeMax (first_expr[, Expression])
```

RangeMaxString

RangeMaxString() devuelve el último valor, según el criterio de ordenación del texto, que encuentra en la expresión o campo.

```
RangeMaxString (first_expr[, Expression])
```

RangeMin

RangeMin() devuelve los valores numéricos más bajos encontrados dentro de la expresión o campo.

```
RangeMin (first_expr[, Expression])
```

RangeMinString

RangeMinString() devuelve el primer valor, según el criterio de ordenación del texto, que encuentra en la expresión o campo.

```
RangeMinString (first_expr[, Expression])
```

RangeMode

RangeMode() halla el valor más frecuente (valor de moda) en la expresión o campo.

```
RangeMode (first_expr[, Expression])
```

RangeOnly

RangeOnly() es una función dual que devuelve un valor si la expresión se evalúa como un valor único. Si este no es el caso devuelve **NULL**.

```
RangeOnly (first_expr[, Expression])
```

RangeSum

RangeSum() devuelve la suma de un rango de valores. Todos los valores no numéricos se tratan como 0.

```
RangeSum (first_expr[, Expression])
```

Funciones de rango de contador

RangeCount

RangeCount() devuelve el número de valores, tanto de texto como numéricos, que hay en la expresión o campo.

```
RangeCount (first_expr[, Expression])
```

RangeMissingCount

RangeMissingCount() devuelve el número de valores no numéricos (incluidos los valores NULL) que hay en la expresión o campo.

```
RangeMissingCount (first_expr[, Expression])
```

RangeNullCount

RangeNullCount() halla el número de valores NULL en la expresión o campo.

```
RangeNullCount (first_expr[, Expression])
```

RangeNumericCount

RangeNumericCount() halla el número de valores numéricos en la expresión o campo.

```
RangeNumericCount (first_expr[, Expression])
```

RangeTextCount

RangeTextCount() devuelve el número de valores de texto en una expresión o campo.

```
RangeTextCount (first_expr[, Expression])
```

Funciones de rango estadísticas

RangeAvg

RangeAvg() devuelve el promedio de un rango. Lo introducido en la función puede ser o bien un rango de valores o una expresión.

```
RangeAvg (first_expr[, Expression])
```

RangeCorrel

RangeCorrel() devuelve el coeficiente de correlación de dos conjuntos de datos. El coeficiente de correlación es una medida de la relación entre los conjuntos de datos.

```
RangeCorrel (x_values , y_values[, Expression])
```

RangeFractile

RangeFractile() devuelve el valor que corresponde al enésimo **fractil** (cuantil) de un rango de números.

```
RangeFractile (fractile, first_expr[ ,Expression])
```

RangeKurtosis

RangeKurtosis() devuelve el valor que corresponde a la kurtosis de un rango de números.

```
RangeKurtosis (first_expr[, Expression])
```

RangeSkew

RangeSkew() devuelve el valor que corresponde a la asimetría de un rango de números.

```
RangeSkew (first_expr[, Expression])
```

RangeStdev

RangeStdev() halla la desviación estándar de un rango de números.

```
RangeStdev (expr1[, Expression])
```

Funciones de rango financieras

RangeIRR

RangeIRR() devuelve la tasa de rendimiento interno de una serie de flujos de caja representados por los valores de entrada.

```
RangeIRR (value[, value][, Expression])
```

RangeNPV

RangeNPV() devuelve el valor actual neto de una inversión basada en un tipo de descuento y una serie de pagos periódicos futuros (valores negativos) e ingresos (valores positivos). El resultado tiene un formato numérico predeterminado **money**.

```
RangeNPV (discount_rate, value[, value][, Expression])
```

RangeXIRR

RangeXIRR() devuelve la tasa de rendimiento interno (anual) de una planificación de flujos de efectivo que no tienen por qué ser necesariamente periódicos. Para calcular la tasa interna de rendimiento de una serie de flujos de efectivo periódicos, utilice la función **RangeIRR**.

```
RangeXIRR (values, dates[, Expression])
```

RangeXNPV

RangeXNPV() devuelve el valor actual neto de una planificación de flujos de efectivo (no necesariamente periódicos) representado por números pareados en las expresiones dadas por **pmt** y **date**. Todos los pagos se descuentan según una base anual de 365 días.

```
RangeXNPV (discount_rate, values, dates[, Expression])
```

Vea también:

Funciones inter-registro (page 1289)

RangeAvg

RangeAvg() devuelve el promedio de un rango. Lo introducido en la función puede ser o bien un rango de valores o una expresión.

Sintaxis:

```
RangeAvg(first_expr[, Expression])
```

Tipo de datos que devuelve: numérico

Argumentos:

Los argumentos de esta función pueden contener funciones inter-registro las cuales devuelven un rango de valores por sí mismas.

Argumentos

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de medir.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Limitaciones:

Si no encuentra ningún valor numérico, devuelve NULL.

Ejemplos y resultados:

Ejemplos de script

Ejemplos	Resultados
`RangeAvg (1,2,4)`	Devuelve 2,33333333
`RangeAvg (1,'xyz')`	Devuelve 1
`RangeAvg (null( ), 'abc')`	Devuelve NULL

Ejemplo:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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
```

```
];
```

La tabla resultante muestra los valores que devuelve MyRangeAvg por cada uno de los registros de la tabla.

Tabla resultante

RangeID	MyRangeAvg
1	7
2	4
3	6
4	12.666
5	6.333
6	5

Ejemplo con expresión:

```
RangeAvg (Above(MyField),0,3))
```

Devuelve una media deslizante del resultado del rango de tres valores de **MyField** calculados en la fila actual y dos filas por encima de la fila actual. Especificando el tercer argumento como 3, la función **Above()** devuelve tres valores, donde haya suficientes filas por encima, que se toman como entrada para la función **RangeAvg()**.

Datos utilizados en los ejemplos:

*Deshabilite la ordenación de **MyField** para garantizar que el ejemplo funcione según lo esperado.*

Datos de muestra

MyField	RangeAvg (Above (MyField,0,3))	Comments
10	10	Como ésta es la fila superior, el rango consiste en un valor solamente.
2	6	Solo hay una fila por encima de esta fila, por lo que el rango es: 10,2.
8	6.6666666667	El equivalente a RangeAvg(10,2,8)
18	9.333333333	-
5	10. 333333333	-
9	10.6666666667	-

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
```

```
5
9
] ;
```

Vea también:

- *Avg - función de gráfico (page 411)*
- *Count - función de gráfico (page 359)*

RangeCorrel

RangeCorrel() devuelve el coeficiente de correlación de dos conjuntos de datos. El coeficiente de correlación es una medida de la relación entre los conjuntos de datos.

Sintaxis:

```
RangeCorrel(x_value , y_value[, Expression])
```

Tipo de datos que devuelve: numérico

Las series de datos deben introducirse como pares (x,y). Por ejemplo, para evaluar dos series de datos, la matriz 1 y la matriz 2, donde la matriz 1 = 2,6,9 y la matriz 2 = 3,8,4 se escribiría `RangeCorrel (2,3,6,8,9,4)`, lo cual devuelve 0,269.

Argumentos:

Argumentos

Argumento	Descripción
x-value, y-value	Cada valor representa un valor único o un rango de valores devueltos por una función inter-registro con un tercer parámetro opcional. Cada valor o rango de valores debe corresponder a un valor **x-value** o a un rango de valores **y-values**.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Limitaciones:

La función necesita al menos dos pares de coordenadas para poder calcularse.

Los valores de texto, valores nulos y valores perdidos devuelven NULL.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplos	Resultados
`RangeCorrel (2,3,6,8,9,4,8,5)`	Devuelve 0,2492. Esta función puede cargarse en el script o añadirse a una visualización en el editor de expresiones.

Ejemplo:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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 '|');
```

En una tabla con ID1 como dimensión y la medida RangeCorrel(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6)), la función **RangeCorrel()** encuentra el valor de **Correl** en el rango de seis pares x,y, por cada uno de los valores ID1.

Tabla resultante

ID1	MyRangeCorrel
01	-0.9517
02	-0.5209
03	-0.5209
04	-0.1599

Ejemplo:

```
XY:
LOAD recno() as RangeID, * Inline [
X|Y
2|3
6|8
9|4
8|5
](delimiter is '|');
```

En una tabla con RangeID como dimensión y la medida: RangeCorrel(Below(X,0,4,BelowY,0,4)), la función **RangeCorrel()** utiliza los resultados de las funciones **Below()**, las cuales, debido a que el tercer argumento (count) está fijado en 4, producen un rango de cuatro x-y valores desde la tabla cargada XY.

Tabla resultante

RangeID	MyRangeCorrel2
01	0.2492
02	-0.9959
03	-1.0000
04	-

El valor de RangeID 01 es el mismo que introducir manualmente RangeCorrel(2,3,6,8,9,4,8,5). Para los demás valores de RangeID, la serie producida por la función Below() son: (6,8,9,4,8,5), (9,4,8,5) y (8,5), el último de los cuales produce un resultado nulo.

Vea también:

Correl - función de gráfico (page 414)

RangeCount

RangeCount() devuelve el número de valores, tanto de texto como numéricos, que hay en la expresión o campo.

Sintaxis:

```
RangeCount(first_expr[, Expression])
```

Tipo de datos que devuelve: Entero

Argumentos:

Los argumentos de esta función pueden contener funciones inter-registro las cuales devuelven un rango de valores por sí mismas.

Argumentos

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de contar.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de contar.

Limitaciones:

Los valores NULL no se cuentan.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplos	Resultados
`RangeCount (1,2,4)`	Devuelve 3
`RangeCount (2,'xyz')`	Devuelve 2
`RangeCount (null( ))`	Devuelve 0
`RangeCount (2,'xyz', null())`	Devuelve 2

Ejemplo:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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
];
```

La tabla resultante muestra los valores que devuelve MyRangeCount por cada uno de los registros de la tabla.

Tabla de resultados

RangeID	MyRangeCount
1	3
2	3
3	3
4	3
5	3
6	3

Ejemplo con expresión:

```
RangeCount (Above(MyField,1,3))
```

Devuelve el número de valores contenidos en los tres resultados de **MyField**. Especificando el primer argumento de la función **Above()** como 1 y el segundo argumento como 3, devuelve los valores de los primeros tres campos por encima de la fila actual, donde hay suficientes filas, las cuales se toman como entrada para la función **RangeCount()**.

Datos utilizados en los ejemplos:

Datos de muestra

MyField	RangeCount(Above(MyField,1,3))
10	0
2	1
8	2
18	3
5	3
9	3

Datos utilizados en los ejemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Vea también:

Count - función de gráfico (page 359)

RangeFractile

RangeFractile() devuelve el valor que corresponde al enésimo **fractil** (cuantil) de un rango de números.

RangeFractile() utiliza la interpolación lineal entre los rangos más cercanos al calcular el fractil.

Sintaxis:

```
RangeFractile(fractile, first_expr[, Expression])
```

Tipo de datos que devuelve: numérico

Argumentos:

Los argumentos de esta función pueden contener funciones inter-registro las cuales devuelven un rango de valores por sí mismas.

Argumentos

Argumento	Descripción
fractile	Un número entre 0 y 1 correspondiente al percentil (cuantil expresado como fracción) que se debe calcular.
first_expr	La expresión o el campo que contiene los datos que se han de medir.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplos	Resultados
`RangeFractile (0.24,1,2,4,6)`	Devuelve 1,72
`RangeFractile(0.5,1,2,3,4,6)`	Devuelve 3
`RangeFractile (0.5,1,2,5,6)`	Devuelve 3,5

Ejemplo:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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
];
```

La tabla resultante muestra los valores que devuelve MyRangeFrac por cada uno de los registros de la tabla.

Tabla resultante

RangeID	MyRangeFrac
1	6
2	3
3	8
4	11
5	5
6	4

Ejemplo con expresión:

```
RangeFractile (0.5, Above(Sum(MyField),0,3))
```

En este ejemplo, la función inter-registro **Above()** contiene los argumentos opcionales offset y count. Esto produce un rango de resultados que puede utilizarse como entrada para cualquiera de las funciones de rango. En este caso, `Above(Sum(MyField),0,3)` devuelve los valores de `MyField` para la fila actual y las dos filas superiores. Estos valores proporcionan la entrada para la función **RangeFractile()**. Por lo tanto, para la fila inferior de la tabla a continuación, esto es el equivalente de `RangeFractile(0.5, 3,4,6)` es decir, calcular el fractil de 0,5 para las series 3, 4 y 6. Las primeras dos filas de la tabla a continuación, el número de valores en el rango se reduce en consecuencia, donde no hay filas por encima de la fila actual. Se obtendrán resultados similares para las demás funciones inter-registro.

Datos de muestra

MyField	RangeFractile(0.5, Above(Sum(MyField),0,3))
1	1
2	1.5
3	2
4	3
5	4
6	5

Datos utilizados en los ejemplos:

```
RangeTab:
LOAD * INLINE [
MyField
1
2
3
4
5
6
] ;
```

Vea también:

- *Above - función de gráfico (page 1293)*
- *Fractile - función de gráfico (page 418)*

RangeIRR

RangeIRR() devuelve la tasa de rendimiento interno de una serie de flujos de caja representados por los valores de entrada.

La tasa interna de devolución es el último tipo de interés recibido para una inversión consistente en pagos (valores negativos) e ingresos (valores positivos) que se suceden durante períodos regulares.

Esta función utiliza una versión simplificada del método de Newton para calcular la tasa interna de retorno (TIR).

Sintaxis:

```
RangeIRR(value[, value][, Expression])
```

Tipo de datos que devuelve: numérico

Argumentos

Argumento	Descripción
value	Un único valor, o un rango de valores, devueltos por una función inter-registro con un tercer parámetro opcional. La función necesita al menos un valor positivo y otro negativo para poder calcular.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos se descartan.

Tabla de ejemplo

Ejemplos	Resultados
`RangeIRR(-70000,12000,15000,18000,21000,26000)`	Devuelve 0,0866

Ejemplos	Resultados

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.
`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 '|');`
	La tabla resultante muestra los valores que devuelve RangeIRR por cada uno de los registros de la tabla.
	RangeID RangeIRR
	1 0.0639
	2 0.8708
	3 -
	4 5.8419
	5 0.9318
	6 -0.2566

Vea también:

- *Funciones inter-registro (page 1289)*

RangeKurtosis

RangeKurtosis() devuelve el valor que corresponde a la kurtosis de un rango de números.

Sintaxis:

```
RangeKurtosis(first_expr[, Expression])
```

Tipo de datos que devuelve: numérico

Argumentos:

Los argumentos de esta función pueden contener funciones inter-registro las cuales devuelven un rango de valores por sí mismas.

Argumentos

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de medir.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Limitaciones:

Si no encuentra ningún valor numérico, devuelve NULL.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplos	Resultados
`RangeKurtosis (1,2,4,7)`	Devuelve -0,28571428571429

Vea también:

Kurtosis - función de gráfico (page 426)

RangeMax

RangeMax() devuelve los valores numéricos más altos encontrados dentro de la expresión o campo.

Sintaxis:

```
RangeMax(first_expr[, Expression])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de medir.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Limitaciones:

Si no encuentra ningún valor numérico, devuelve NULL.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplos	Resultados
`RangeMax (1,2,4)`	Devuelve 4
`RangeMax (1,'xyz')`	Devuelve 1
`RangeMax (null( ), 'abc')`	Devuelve NULL

Ejemplo:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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
];
```

La tabla resultante muestra los valores que devuelve MyRangeMax por cada uno de los registros de la tabla.

Tabla resultante

RangeID	MyRangeMax
1	10
2	7
3	8
4	18
5	9
6	9

Ejemplo con expresión:

```
RangeMax (Above(MyField,0,3))
```

Devuelve el valor máximo en el rango de tres valores de **MyField** calculados en la fila actual y dos filas por encima de la fila actual. Especificando el tercer argumento como 3, la función **Above()** devuelve tres valores, donde haya suficientes filas por encima, que se toman como entrada para la función **RangeMax()**.

Datos utilizados en los ejemplos:

*Deshabilite la ordenación de **MyField** para garantizar que el ejemplo funcione según lo esperado.*

Datos de muestra

MyField	RangeMax (Above(Sum(MyField),1,3))
10	10
2	10
8	10
18	18
5	18
9	18

Datos utilizados en los ejemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

RangeMaxString

RangeMaxString() devuelve el último valor, según el criterio de ordenación del texto, que encuentra en la expresión o campo.

Sintaxis:

```
RangeMaxString(first_expr[, Expression])
```

Tipo de datos que devuelve: cadena

Argumentos:

Los argumentos de esta función pueden contener funciones inter-registro las cuales devuelven un rango de valores por sí mismas.

Argumentos

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de medir.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplos	Resultados
`RangeMaxString (1,2,4)`	Devuelve 4
`RangeMaxString ('xyz','abc')`	Devuelve 'xyz'
`RangeMaxString (5,'abc')`	Devuelve 'abc'
`RangeMaxString (null( ))`	Devuelve NULL

Ejemplo con expresión:

```
RangeMaxString (Above(MaxString(MyField),0,3))
```

Devuelve el último (en el orden del texto) de los tres resultados de la función **MaxString(MyField)** evaluada en la fila actual y dos filas por encima de la fila actual.

Datos utilizados en los ejemplos:

*Deshabilite la ordenación de **MyField** para garantizar que el ejemplo funcione según lo esperado.*

Datos de muestra

MyField	RangeMaxString(Above(MaxString(MyField),0,3))
10	10
abc	abc
8	abc
def	def
xyz	xyz
9	xyz

Datos utilizados en los ejemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

Vea también:

MaxString - función de gráfico (page 552)

RangeMin

RangeMin() devuelve los valores numéricos más bajos encontrados dentro de la expresión o campo.

Sintaxis:

```
RangeMin(first_expr[, Expression])
```

Tipo de datos que devuelve: numérico

Argumentos:

Argumentos

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de medir.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Limitaciones:

Si no encuentra ningún valor numérico, devuelve NULL.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplos	Resultados
`RangeMin (1,2,4)`	Devuelve 1
`RangeMin (1,'xyz')`	Devuelve 1
`RangeMin (null( ), 'abc')`	Devuelve NULL

Ejemplo:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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
];
```

La tabla resultante muestra los valores que devuelve MyRangeMin por cada uno de los registros de la tabla.

Tabla resultante

RangeID	MyRangeMin
1	5
2	2
3	2
4	9
5	5
6	2

Ejemplo con expresión:

```
RangeMin (Above(MyField,0,3)
```

Devuelve el valor mínimo en el rango de tres valores de **MyField** calculados en la fila actual y dos filas por encima de la fila actual. Especificando el tercer argumento como 3, la función **Above()** devuelve tres valores, donde haya suficientes filas por encima, que se toman como entrada para la función **RangeMin()**.

Datos utilizados en los ejemplos:

Datos de muestra

MyField	RangeMin(Above(MyField,0,3))
10	10
2	2
8	2
18	2
5	5
9	5

Datos utilizados en los ejemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Vea también:

Min - función de gráfico (page 346)

RangeMinString

RangeMinString() devuelve el primer valor, según el criterio de ordenación del texto, que encuentra en la expresión o campo.

Sintaxis:

```
RangeMinString(first_expr[, Expression])
```

Tipo de datos que devuelve: cadena

Argumentos:

Los argumentos de esta función pueden contener funciones inter-registro las cuales devuelven un rango de valores por sí mismas.

Argumentos

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de medir.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplos	Resultados
`RangeMinString (1,2,4)`	Devuelve 1
`RangeMinString ('xyz','abc')`	Devuelve 'abc'
`RangeMinString (5,'abc')`	Devuelve 5
`RangeMinString (null( ))`	Devuelve NULL

Ejemplo con expresión:

```
RangeMinString (Above(MinString(MyField),0,3))
```

Devuelve el primero (en el orden del texto) de los tres resultados de la función **MinString(MyField)** evaluada en la fila actual y dos filas por encima de la fila actual.

Datos utilizados en los ejemplos:

*Deshabilite la ordenación de **MyField** para garantizar que el ejemplo funcione según lo esperado.*

Datos de muestra

MyField	RangeMinString(Above(MinString(MyField),0,3))
10	10
abc	10
8	8
def	8
xyz	8
9	9

Datos utilizados en los ejemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

Vea también:

MinString - función de gráfico (page 554)

RangeMissingCount

RangeMissingCount() devuelve el número de valores no numéricos (incluidos los valores NULL) que hay en la expresión o campo.

Sintaxis:

```
RangeMissingCount(first_expr[, Expression])
```

Tipo de datos que devuelve: Entero

Argumentos:

Los argumentos de esta función pueden contener funciones inter-registro las cuales devuelven un rango de valores por sí mismas.

Argumentos

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de contar.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de contar.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplos	Resultados
`RangeMissingCount (1,2,4)`	Devuelve 0
`RangeMissingCount (5,'abc')`	Devuelve 1
`RangeMissingCount (null( ))`	Devuelve 1

Ejemplo con expresión:

```
RangeMissingCount (Above(MinString(MyField),0,3))
```

Devuelve la cantidad de valores no numéricos en los tres resultados de la función **MinString(MyField)** evaluada en la fila actual y dos filas sobre la fila actual.

*Deshabilite la ordenación de **MyField** para garantizar que el ejemplo funcione según lo esperado.*

Datos de muestra

MyField	RangeMissingCount (Above(MinString (MyField),0,3))	Explanation
10	2	Devuelve 2 porque no hay filas por encima de esta fila así que 2 de los 3 valores faltan.
abc	2	Devuelve 2 porque solo hay 1 fila por encima de la fila actual y la fila actual no es numérica ('abc').
8	1	Devuelve 1 porque 1 de las 3 filas incluye un ('abc') no numérico.
def	2	Devuelve 2 porque 2 de las 3 filas incluyen valores no numéricos ('def' y 'abc').
xyz	2	Devuelve 2 porque 2 de las 3 filas incluyen valores no numéricos (' xyz' y 'def').
9	2	Devuelve 2 porque 2 de las 3 filas incluyen valores no numéricos (' xyz' y 'def').

Datos utilizados en los ejemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
```

```
9
] ;
```

Vea también:

MissingCount - función de gráfico (page 363)

RangeMode

RangeMode() halla el valor más frecuente (valor de moda) en la expresión o campo.

Sintaxis:

```
RangeMode(first_expr {, Expression})
```

Tipo de datos que devuelve: numérico

Argumentos:

Los argumentos de esta función pueden contener funciones inter-registro las cuales devuelven un rango de valores por sí mismas.

Argumentos

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de medir.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Limitaciones:

Si más de un valor comparte la frecuencia más alta, devuelve NULL.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplos	Resultados
`RangeMode (1,2,9,2,4)`	Devuelve 2
`RangeMode ('a',4,'a',4)`	Devuelve NULL
`RangeMode (null( ))`	Devuelve NULL

Ejemplo:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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
];
```

La tabla resultante muestra los valores que devuelve **MyRangeMode** por cada uno de los registros de la tabla.

Tabla de resultados

RangeID	**MyRangMode**
1	-
2	-
3	8
4	-
5	5
6	-

Ejemplo con expresión:

```
RangeMode (Above(MyField,0,3))
```

Devuelve el valor más frecuente en los tres resultados de **MyField** evaluados en la fila actual y dos filas por encima de la fila actual. Especificando el tercer argumento como 3, la función **Above()** devuelve tres valores, donde haya suficientes filas por encima, que se toman como entrada para la función **RangeMode()**.

Datos utilizados en el ejemplo:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

*Deshabilite la ordenación de **MyField** para garantizar que el ejemplo funcione según lo esperado.*

Datos de muestra

MyField	**RangeMode(Above(MyField,0,3))**
10	Devuelve 10 porque no hay filas por encima así que el valor único es el que aparece con mayor frecuencia.

MyField	RangeMode(Above(MyField,0,3))
2	-
8	-
18	-
5	-
9	-

Vea también:

Mode - función de gráfico (page 349)

RangeNPV

RangeNPV() devuelve el valor actual neto de una inversión basada en un tipo de descuento y una serie de pagos periódicos futuros (valores negativos) e ingresos (valores positivos). El resultado tiene un formato numérico predeterminado **money**.

Para flujos de caja no necesariamente periódicos, vea *RangeXNPV (page 1387)*.

Sintaxis:

```
RangeNPV(discount_rate, value[,value][, Expression])
```

Tipo de datos que devuelve: numérico

Argumentos

Argumento	Descripción
discount_rate	Es el tipo de interés por periodo.
value	Es un pago o ingreso que tiene lugar al final de cada periodo. Cada valor puede ser un valor único o un rango de valores devueltos por una función inter-registro con un tercer parámetro opcional.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Limitaciones:

Los valores de texto, valores NULL y valores perdidos se descartan.

Ejemplos	Resultados
`RangeNPV(0.1,-10000,3000,4200,6800)`	Devuelve 1188,44

Ejemplos	Resultados

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.
`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 '|');`
	La tabla resultante muestra los valores que devuelve RangeNPV por cada uno de los registros de la tabla.
	RangeID: 1, RangeNPV: $-49.13
	RangeID: 2, RangeNPV: $777.78
	RangeID: 3, RangeNPV: $98.77
	RangeID: 4, RangeNPV: $25.51
	RangeID: 5, RangeNPV: $250.83
	RangeID: 6, RangeNPV: $20.40

Vea también:

- *Funciones inter-registro (page 1289)*

RangeNullCount

RangeNullCount() halla el número de valores NULL en la expresión o campo.

Sintaxis:

```
RangeNullCount(first_expr [, Expression])
```

Tipo de datos que devuelve: Entero

Argumentos:

Los argumentos de esta función pueden contener funciones inter-registro las cuales devuelven un rango de valores por sí mismas.

Argumentos

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de medir.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplos	Resultados
`RangeNullCount (1,2,4)`	Devuelve 0
`RangeNullCount (5,'abc')`	Devuelve 0
`RangeNullCount (null( ), null( ))`	Devuelve 2

Ejemplo con expresión:

```
RangeNullCount (Above(Sum(MyField),0,3))
```

Devuelve el número de valores NULL en los tres resultados de la función **Sum(MyField)** evaluada en la fila actual y dos filas por encima de la fila actual.

*Copiar **MyField** en el ejemplo a continuación no dará como resultado un valor NULL.*

Datos de muestra

MyField	RangeNullCount(Above(Sum(MyField),0,3))
10	Devuelve 2 porque no hay filas por encima de esta fila, así que 2 de los 3 valores faltan (=NULL).
'abc'	Devuelve 1 porque solo hay una fila por encima de la fila actual, por lo que falta uno de los tres valores (=NULL).
8	Devuelve 0 porque ninguna de las tres filas es un valor NULL.

Datos utilizados en los ejemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
] ;
```

Vea también:

NullCount - función de gráfico (page 366)

RangeNumericCount

RangeNumericCount() halla el número de valores numéricos en la expresión o campo.

Sintaxis:

```
RangeNumericCount(first_expr[, Expression])
```

Tipo de datos que devuelve: Entero

Argumentos:

Los argumentos de esta función pueden contener funciones inter-registro las cuales devuelven un rango de valores por sí mismas.

Argumentos

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de medir.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplos	Resultados
`RangeNumericCount (1,2,4)`	Devuelve 3
`RangeNumericCount (5,'abc')`	Devuelve 1
`RangeNumericCount (null( ))`	Devuelve 0

Ejemplo con expresión:

```
RangeNumericCount (Above(MaxString(MyField),0,3))
```

Devuelve la cantidad de valores numéricos en los tres resultados de la función **MaxString(MyField)** evaluada en la fila actual y dos filas sobre la fila actual.

*Deshabilite la ordenación de **MyField** para garantizar que el ejemplo funcione según lo esperado.*

Datos de muestra

MyField	RangeNumericCount(Above(MaxString(MyField),0,3))
10	1
abc	1
8	2
def	1
xyz	1
9	1

Datos utilizados en los ejemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
def
xyz
9
] ;
```

Vea también:

NumericCount - función de gráfico (page 369)

RangeOnly

RangeOnly() es una función dual que devuelve un valor si la expresión se evalúa como un valor único. Si este no es el caso devuelve **NULL**.

Sintaxis:

```
RangeOnly(first_expr[, Expression])
```

Tipo de datos que devuelve: dual

Argumentos:

Los argumentos de esta función pueden contener funciones inter-registro las cuales devuelven un rango de valores por sí mismas.

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de medir.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Ejemplos y resultados:

Ejemplos	Resultados
`RangeOnly (1,2,4)`	Devuelve NULL
`RangeOnly (5,'abc')`	Devuelve NULL
`RangeOnly (null( ), 'abc')`	Devuelve 'abc'
`RangeOnly(10,10,10)`	Devuelve 10

Vea también:

Only - función de gráfico (page 352)

RangeSkew

RangeSkew() devuelve el valor que corresponde a la asimetría de un rango de números.

Sintaxis:

```
RangeSkew(first_expr[, Expression])
```

Tipo de datos que devuelve: numérico

Argumentos:

Los argumentos de esta función pueden contener funciones inter-registro las cuales devuelven un rango de valores por sí mismas.

Argumentos

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de medir.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Limitaciones:

Si no encuentra ningún valor numérico, devuelve NULL.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplos	Resultados
`rangeskew (1,2,4)`	Devuelve 0,93521952958283
`rangeskew (above (SalesValue,0,3))`	Devuelve un sesgo deslizante del rango de tres valores devueltos por la función above() calculada en la fila actual y las dos filas por encima de la fila actual.

Datos utilizados en el ejemplo:

Datos de muestra

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
] ;
```

Vea también:

Skew - función de gráfico (page 458)

RangeStdev

RangeStdev() halla la desviación estándar de un rango de números.

Sintaxis:

```
RangeStdev(first_expr[, Expression])
```

Tipo de datos que devuelve: numérico

Argumentos:

Los argumentos de esta función pueden contener funciones inter-registro las cuales devuelven un rango de valores por sí mismas.

Argumentos

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de medir.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Limitaciones:

Si no encuentra ningún valor numérico, devuelve NULL.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplos	Resultados
`RangeStdev (1,2,4)`	Devuelve 1,5275252316519
`RangeStdev (null( ))`	Devuelve NULL
`RangeStdev (above (SalesValue),0,3))`	Devuelve un estándar deslizante del rango de tres valores devueltos por la función above() calculada en la fila actual y las dos filas sobre la fila actual.

Datos utilizados en el ejemplo:

Datos de muestra

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
] ;
```

Vea también:

- *Stdev - función de gráfico (page 461)*

RangeSum

RangeSum() devuelve la suma de un rango de valores. Todos los valores no numéricos se tratan como 0.

Sintaxis:

```
RangeSum(first_expr[, Expression])
```

Tipo de datos que devuelve: numérico

Argumentos:

Los argumentos de esta función pueden contener funciones inter-registro las cuales devuelven un rango de valores por sí mismas.

Argumentos

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de medir.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Limitaciones:

La función **RangeSum** trata todos los valores no numéricos como 0.

Ejemplos y resultados:

Ejemplos

Ejemplos	Resultados
`RangeSum (1,2,4)`	Devuelve 7
`RangeSum (5,'abc')`	Devuelve 5
`RangeSum (null( ))`	Devuelve 0

Ejemplo:

Agregue el script de ejemplo en su app y ejecútelo. Para ver el resultado, agregue los campos enumerados en la columna de resultados a una hoja de su app.

```
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
];
```

La tabla resultante muestra los valores que devuelve MyRangeSum por cada uno de los registros de la tabla.

Tabla resultante

RangeID	MyRangeSum
1	21
2	12
3	18
4	38
5	19
6	15

Ejemplo con expresión:

```
RangeSum (Above(MyField,0,3))
```

Devuelve la suma de los tres valores de **MyField)**: desde la fila actual y dos filas por encima de la fila actual. Especificando el tercer argumento como 3, la función **Above()** devuelve tres valores, donde haya suficientes filas por encima, que se toman como entrada para la función **RangeSum()**.

Datos utilizados en los ejemplos:

*Deshabilite la ordenación de **MyField** para garantizar que el ejemplo funcione según lo esperado.*

Datos de muestra

MyField	RangeSum(Above(MyField,0,3))
10	10
2	12
8	20

MyField	RangeSum(Above(MyField,0,3))
18	28
5	31
9	32

Datos utilizados en los ejemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Vea también:

- *Sum - función de gráfico (page 354)*
- *Above - función de gráfico (page 1293)*

RangeTextCount

RangeTextCount() devuelve el número de valores de texto en una expresión o campo.

Sintaxis:

```
RangeTextCount(first_expr[, Expression])
```

Tipo de datos que devuelve: Entero

Argumentos:

Los argumentos de esta función pueden contener funciones inter-registro las cuales devuelven un rango de valores por sí mismas.

Argumento

Argumento	Descripción
first_expr	La expresión o el campo que contiene los datos que se han de medir.
Expression	Las expresiones o campos opcionales que contienen el rango de datos que se han de medir.

Ejemplos y resultados:

Ejemplos de funciones

Ejemplos	Resultados
`RangeTextCount (1,2,4)`	Devuelve 0
`RangeTextCount (5,'abc')`	Devuelve 1
`RangeTextCount (null( ))`	Devuelve 0

Ejemplo con expresión:

```
RangeTextCount (Above(MaxString(MyField),0,3))
```

Devuelve el número de valores de texto dentro de los tres resultados de la función **MaxString(MyField)** evaluada en la fila actual y dos filas por encima de la fila actual.

Datos utilizados en los ejemplos:

*Deshabilite la ordenación de **MyField** para garantizar que el ejemplo funcione según lo esperado.*

Datos de ejemplo

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

Datos utilizados en los ejemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
null()
'xyz'
9
] ;
```

Vea también:

TextCount - función de gráfico (page 373)

RangeXIRR

RangeXIRR() devuelve la tasa de rendimiento interno (anual) de una planificación de flujos de efectivo que no tienen por qué ser necesariamente periódicos. Para calcular la tasa interna de rendimiento de una serie de flujos de efectivo periódicos, utilice la función **RangeIRR**.

La funcionalidad XIRR de Qlik (funciones **XIRR()** y **RangeXIRR()**) utiliza la siguiente ecuación, resolviendo para el valor `Rate`, a fin de determinar el valor XIRR correcto:

```
XNPV(Rate, pmt, date) = 0
```

La ecuación se resuelve utilizando una versión simplificada del método de Newton.

Sintaxis:

```
RangeXIRR(value, date{, value, date})
```

Tipo de datos que devuelve: numérico

Argumentos

Argumento	Descripción
value	Un flujo de caja o una serie de flujos de caja correspondientes a una planificación de pagos por fechas. La serie de valores deberá contener al menos un valor positivo y otro negativo.
date	Una fecha de pago o una planificación de pagos en varias fechas, que se corresponde con los pagos de los flujos de caja.

Al trabajar con esta función, se aplican las siguientes limitaciones:

- Los valores de texto, valores NULL y valores perdidos se descartan.
- Todos los pagos son descontados según una base de un año de 365 días.
- Esta función requiere al menos un pago negativo válido y al menos un pago positivo válido (con las fechas válidas correspondientes). Si no se proporcionan estos pagos, devuelve un valor NULL.

Los temas siguientes le ayudarán a trabajar con esta función:

- *RangeXNPV (page 1387)*: Utilice esta función para calcular el valor actual neto de una planificación de flujos de caja que no tienen que ser necesariamente periódicos.
- *XIRR (page 387)*: La función **XIRR()** calcula la tasa de rendimiento interno agregado (anual) de una planificación de flujos de efectivo que no tienen por qué ser necesariamente periódicos.

En diferentes versiones de Qlik Sense administrado por el cliente, hay variaciones en el algoritmo que utiliza esta función. Para más información sobre actualizaciones recientes del algoritmo, consulte el artículo de ayuda Reparación y actualización de la función XIRR*.*

Ejemplos y resultados:

Ejemplos y resultados

Ejemplos	Resultados
`RangeXIRR(-2500,'2008-01-01',2750,'2008-09-01')`	Devuelve 0,1532

Vea también:

- *RangeIRR (page 1360)*
- *RangeXNPV (page 1387)*
- *XIRR (page 387)*
- XIRR function Fix and Update

RangeXNPV

RangeXNPV() devuelve el valor actual neto de una planificación de flujos de efectivo (no necesariamente periódicos) representado por números pareados en las expresiones dadas por **pmt** y **date**. Todos los pagos se descuentan según una base anual de 365 días.

Sintaxis:

```
RangeXNPV(discount_rate, value, date{, value, date})
```

Tipo de datos que devuelve: numérico

Argumentos

Argumento	Descripción
discount_rate	**discount_rate** es la tasa anual a la que se deben descontar los pagos.
value	Un flujo de caja o una serie de flujos de caja correspondientes a una planificación de pagos por fechas. Cada valor puede ser un valor único o un rango de valores devueltos por una función inter-registro con un tercer parámetro opcional. La serie de valores deberá contener al menos un valor positivo y otro negativo.
date	Una fecha de pago o una planificación de pagos en varias fechas, que se corresponde con los pagos de los flujos de caja.

Al trabajar con esta función, se aplican las siguientes limitaciones:

- Los valores de texto, valores NULL y valores perdidos se descartan.
- Todos los pagos son descontados según una base de un año de 365 días.

Ejemplo: script

Script de carga y resultados

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Datos financieros contenidos en una tabla denominada `RangeTab3`.
- El uso de la función **RangeXNPV()** para calcular el valor actual neto.

Script de carga

```
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 '|');
```

Resultados

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue estos campos como dimensiones:

- `RangeID`
- `RangeXNPV`

Tabla de resultados

RangeID	RangeXNPV
1	-$9.09
2	-$0.00
3	$13.64

Ejemplo: expresión de gráfico

Script de carga y expresión de gráfico

Vista general

Abra el editor de carga de datos y agregue el script de carga a continuación a una nueva pestaña.

El script de carga contiene:

- Datos financieros contenidos en una tabla denominada `RangeTab3`.
- El uso de la función **RangeXNPV()** para calcular el valor actual neto.

Script de carga

```
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 '|');
```

Resultados

Haga lo siguiente:

Cargue los datos y abra una hoja. Cree una nueva tabla y agregue el siguiente cálculo como medida.

```
=RangeXNPV(0.1, -2500,'2008-01-01',2750,'2008-09-01')
```

Tabla de resultados

=XIRR(Payments, Date)
$80.25

Vea también:

XNPV (page 394)

8.22 Funciones relacionales

Este es un grupo de funciones que calculan propiedades de valores dimensionales individuales en un gráfico, utilizando números ya agregados.

Las funciones son relacionales en el sentido de que el resultado de la función depende no solo del valor del punto de datos en sí, sino también de la relación del valor con otros puntos de datos. Por ejemplo, un rango no se puede calcular sin una comparación con otros valores dimensionales.

Estas funciones solo pueden emplearse en expresiones de gráficos. No se pueden utilizar en el script de carga.

Se necesita una dimensión en el gráfico, ya que define los otros puntos de datos necesarios para la comparación. En consecuencia, una función relacional no tiene sentido en un gráfico adimensional (por ejemplo, un objeto KPI).

Funciones de ranking

Suprimir valores cero se desactiva automáticamente cuando se utilizan dichas funciones. Los valores NULL se descartan.

Rank

Rank() evalúa las filas del gráfico en la expresión, y para cada fila muestra la posición relativa del valor de la dimensión evaluada en la expresión. Esta función, cuando evalúa la expresión, compara el resultado con el resultado de las otras filas que contienen el segmento de columna actual y devuelve la clasificación de la fila actual dentro del segmento.

```
Rank - función de gráfico([TOTAL [<fld {, fld}>]] expr[, mode[, fmt]])
```

HRank

HRank() evalúa la expresión y compara el resultado con el resultado de las otras columnas que contienen el segmento de fila actual de una tabla pivotante. La función devuelve el ranking de la columna actual dentro del segmento.

```
HRank - función de gráfico([TOTAL] expr[, mode[, fmt]])
```

Funciones de agrupamiento

KMeans2D

El grupo de propiedades **Licencia de sitio** contiene propiedades relacionadas con la licencia del sistema Qlik Sense. Todos los campos son obligatorios y no deben estar vacíos.

Propiedades de la licencia del sitio

Nombre de la propiedad	Descripción
Nombre del propietario	El nombre de usuario del propietario del producto Qlik Sense.
Empresa del propietario	El nombre de la organización a la que pertenece el propietario del producto Qlik Sense.
Número de serie	El número de serie asignado al software de Qlik Sense.
Número de control	El número de control asignado al software de Qlik Sense.
Acceso LEF	El Archivo habilitador de licencia (LEF) asignado al software de Qlik Sense.

KMeans2D() evalúa las filas del gráfico aplicando agrupación en clústeres k-means, y para cada fila del gráfico muestra el ID del grupo al que se ha asignado este punto de datos. Las columnas que utiliza el algoritmo de agrupamiento están determinadas por los parámetros coordinate_1 y coordinate_2, respectivamente. Ambas son agregaciones. El número de clústeres que se crea viene determinado por el parámetro num_clusters. Los datos se pueden normalizar opcionalmente mediante el parámetro norma.

```
KMeans2D - función de gráfico(num_clusters, coordinate_1, coordinate_2 [,
norm])
```

KMeansND

KMeansND() evalúa las filas del gráfico aplicando agrupación en clústeres k-means, y para cada fila del gráfico muestra el ID del grupo al que se ha asignado este punto de datos. Las columnas que utiliza el algoritmo de agrupamiento vienen determinadas por los parámetros coordinate_1 y coordinate_2, etc., hasta n columnas. Esto son todo agregaciones. El número de clústeres que se crea viene determinado por el parámetro num_clusters.

```
KMeansND - función de gráfico(num_clusters, num_iter, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

KMeansCentroid2D

KMeansCentroid2D() evalúa las filas del gráfico aplicando agrupación en clústeres k-means y para cada fila del gráfico muestra la coordenada deseada del clúster al que se haya asignado este punto de datos. Las columnas que utiliza el algoritmo de agrupamiento vienen determinadas por los parámetros coordinate_1 y coordinate_2, respectivamente. Ambas son agregaciones. El número de clústeres que se crea viene determinado por el parámetro num_clusters. Los datos se pueden normalizar opcionalmente mediante el parámetro norma.

```
KMeansCentroid2D - función de gráfico(num_clusters, coordinate_no,
coordinate_1, coordinate_2 [, norm])
```

KMeansCentroidND

KMeansCentroidND() evalúa las filas del gráfico aplicando agrupación en clústeres k-means, y para cada fila del gráfico muestra la coordenada deseada del clúster al que se ha asignado este punto de datos. Las columnas que utiliza el algoritmo de agrupamiento vienen determinadas por los parámetros coordinate_1 y coordinate_2, etc., hasta n columnas. Esto son todo agregaciones. El número de clústeres que se crea viene determinado por el parámetro num_clusters.

```
KMeansCentroidND - función de gráfico(num_clusters, num_iter, coordinate_no,
coordinate_1, coordinate_2 [,coordinate_3 [, ...]])
```

Funciones de descomposición de series de tiempo

STL_Trend

STL_Trend es una función de descomposición de series de tiempo. Junto con **STL_Seasonal** y **STL_Residual**, esta función se utiliza para descomponer una serie temporal en componentes estacionales, de tendencia y residuales. En el contexto del algoritmo STL, la descomposición de series de tiempo se utiliza para identificar tanto un patrón estacional recurrente como una tendencia general, dada una métrica de entrada y otros parámetros. La función **STL_Trend** identifica una tendencia general, independiente de patrones o ciclos estacionales, a partir de datos de series de tiempo.

```
STL_Trend - función de gráfico(target_measure, period_int [,seasonal_smoother
[,trend_smoother]])
```

STL_Seasonal

STL_Seasonal es una función de descomposición de series de tiempo. Junto con **STL_Trend** y **STL_Residual**, esta función se utiliza para descomponer una serie temporal en componentes estacionales, de tendencia y residuales. En el contexto del algoritmo STL, la descomposición de series de tiempo se utiliza para identificar tanto un patrón estacional recurrente como una tendencia general, dada una métrica de entrada y otros parámetros. La función **STL_Seasonal** puede identificar un patrón estacional dentro de una serie temporal, separándolo de la tendencia general que muestran los datos.

```
STL_Seasonal - función de gráfico(target_measure, period_int [,seasonal_
smoother [,trend_smoother]])
```

STL_Residual

STL_Residual es una función de descomposición de series de tiempo. Junto con **STL_Seasonal** y **STL_Trend**, esta función se utiliza para descomponer una serie temporal en componentes estacionales, de tendencia y residuales. En el contexto del algoritmo STL, la descomposición de series de tiempo se utiliza para identificar tanto un patrón estacional recurrente como una tendencia general, dada una métrica de entrada y otros parámetros. Al realizar esta operación, parte de la variación en la métrica de entrada no encajará en el componente estacional ni en el de tendencia, y se definirá como el componente residual. La función de gráfico **STL_Residual** captura esta parte del cálculo.

```
STL_Residual - función de gráfico(target_measure, period_int [,seasonal_
smoother [,trend_smoother]])
```

Rank - función de gráfico

Rank() evalúa las filas del gráfico en la expresión, y para cada fila muestra la posición relativa del valor de la dimensión evaluada en la expresión. Esta función, cuando evalúa la expresión, compara el resultado con el resultado de las otras filas que contienen el segmento de columna actual y devuelve la clasificación de la fila actual dentro del segmento.

Segmentos de columna

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

Para gráficos que no sean tablas, se define el segmento de columna actual tal como aparece en su equivalente en la tabla simple del gráfico.

Sintaxis:

```
Rank([TOTAL] expr[, mode[, fmt]])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
mode	Especifica la representación numérica del resultado de la función.
fmt	Especifica la representación de texto del resultado de la función.
TOTAL	Si el gráfico es unidimensional o si la expresión va precedida por el calificador **TOTAL**, la función se evalúa a lo largo de toda la columna. Si la tabla o el equivalente de tabla tiene múltiples dimensiones verticales, el segmento de columna actual incluirá sólo filas con los mismos valores que la fila actual en todas las columnas de dimensión, excepto para la columna que muestra la última dimensión en el orden de campos interno.

El ranking se devuelve como un valor dual, el cual, en el caso de que cada fila tenga una clasificación única será un entero entre 1 y el número de filas del segmento de columna actual.

En caso de que varias filas compartan el mismo ranking, el texto y la representación numérica se pueden controlar con los parámetros **mode** y **fmt**.

mode

El segundo argumento, **mode**, puede tomar los siguientes valores:

Ejemplos de **mode**

Valor	Descripción
0 (por defecto)	Si todos los rangos dentro del grupo compartido caen dentro del extremo más bajo del valor de en medio del ranking entero, todas las filas reciben el rango menor dentro del grupo compartido. Si todos los rangos dentro del grupo compartido caen dentro del extremo más alto del valor de en medio del ranking entero, todas las filas reciben el rango menor dentro del grupo compartido. Si todos los rangos dentro del grupo compartido se salen del valor central del ranking completo, todas las filas reciben el valor correspondiente a la media del ranking superior e inferior en el segmento de columna completo.
1	Rango más bajo en todas las filas.
2	Rango promedio en todas las filas.
3	Rango más alto en todas las filas.
4	Rango más bajo en la primera fila, incrementado por uno para cada fila.

fmt

El tercer argumento, **fmt**, puede tomar los siguientes valores:

Ejemplos de **fmt**

Valor	Descripción
0 (por defecto)	Valor bajo - valor alto en todas las filas (por ejemplo 3 - 4).
1	Valor bajo en todas las filas.
2	Valor más bajo en la primera fila, espacio en blanco en las siguientes filas.

El orden de las filas para **mode** 4 y **fmt** 2 viene determinado por el orden de clasificación de las dimensiones del gráfico.

Ejemplos y resultados:

Cree dos visualizaciones a partir de las dimensiones Product y Sales y otra a partir de Product y UnitSales. Agregue medidas tal como se muestra en la tabla siguiente.

Ejemplos de rangos

Ejemplos	Resultados
Ejemplo 1. Cree una tabla con las dimensiones `Customer` y `Sales` y la medida `Rank(Sales)`	El resultado depende del orden de las dimensiones. Si la tabla se ordena por Customer, la tabla muestra todos los valores de Sales para Astrida, luego Betacab, y así sucesivamente. Los resultados de Rank(Sales) mostrarán 10 para el valor Sales 12, 9 para el valor Sales 13, etc., con el valor de rango de 1 devuelto para el valor Sales 78. El siguiente segmento de columna comienza con Betacab, para el que el primer valor de Sales en el segmento es 12. El valor de rango de Rank(Sales) para esto se proporciona como 11. Si la tabla se ordena por Sales, los segmentos de columna consistirán en los valores de Sales y el correspondiente Customer. Debido a que hay dos valores Sales de 12 (para Astrida y Betacab), el valor de Rank(Sales) para ese segmento de columna es 1-2, por cada valor de Customer. Esto se debe a que hay dos valores de Customer para el valor Sales 12. Si hubiera habido 4 valores, el resultado sería 1-4 para todas las filas. Esto muestra cómo se ve el resultado para el valor predeterminado (0) del argumento fmt.
Ejemplo 2. Reemplace la dimensión Cliente por Producto y agregue la medida `Rank(Sales,1,2)`	Esto devuelve 1 en la primera fila en cada segmento de columna y deja en blanco todas las demás filas, porque los argumentos **mode** y **fmt** se fijan en 1 y 2 respectivamente.

Resultados para el ejemplo 1, con tabla ordenada por Customer:

Tabla de resultados

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
Astrida	78	1
Betcab	12	11

Resultados para el ejemplo 1, con tabla ordenada por Sales:

Tabla de resultados

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

Datos utilizados en los ejemplos:

```
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 '|');
```

Vea también:

Sum - función de gráfico (page 354)

HRank - función de gráfico

HRank() evalúa la expresión y compara el resultado con el resultado de las otras columnas que contienen el segmento de fila actual de una tabla pivotante. La función devuelve el ranking de la columna actual dentro del segmento.

Sintaxis:

```
HRank([ TOTAL ] expr [ , mode [, fmt ] ])
```

Tipo de datos que devuelve: dual

Esta función sólo funciona en tablas pivotantes. En todos los demás tipos de gráficos devuelve NULL.

Argumentos:

Argumentos

Argumento	Descripción
expr	La expresión o el campo que contiene los datos que se han de medir.
mode	Especifica la representación numérica del resultado de la función.
fmt	Especifica la representación de texto del resultado de la función.
TOTAL	Si el gráfico es unidimensional o si la expresión va precedida por el calificador **TOTAL**, la función se evalúa a lo largo de toda la columna. Si la tabla o el equivalente de tabla tiene múltiples dimensiones verticales, el segmento de columna actual incluirá sólo filas con los mismos valores que la fila actual en todas las columnas de dimensión, excepto para la columna que muestra la última dimensión en el orden de campos interno.

Si la tabla pivotante es unidimensional o si la expresión va precedida por el cualificador **total**, el segmento de la fila actual siempre es igual a la fila completa. Si la tabla pivotante tiene múltiples dimensiones horizontales, el segmento de fila actual incluirá sólo columnas con los mismos valores que la columna actual en todas las filas de dimensión excepto la fila que muestra la última dimensión horizontal del orden de campo.

El ranking se devuelve como valor dual, el cual, en el caso de que cada columna tenga un ranking único, será entre 1 y el número de columnas del segmento de columna actual.

En el caso de que varias columnas compartan la misma clasificación, el texto y la representación numérica se pueden controlar con los argumentos **mode** y **format**.

El segundo argumento, **mode**, especifica la representación numérica del resultado de la función:

Ejemplos de **mode**

Valor	Descripción
0 (por defecto)	Si todos los puestos de ranking dentro del grupo compartido caen dentro del extremo más bajo del valor intermedio del ranking completo, todas las filas tendrán el rango más bajo dentro del grupo compartido. Si todos los puestos del ranking dentro del grupo compartido caen dentro del extremo más alto del valor intermedio del ranking completo, todas las columnas recibirán el rango más alto dentro del grupo compartido. Si todos los rangos dentro del grupo compartido se salen del valor central del ranking completo, todas las filas reciben el valor correspondiente a la media del ranking superior e inferior en el segmento de columna completo.
1	Rango más bajo en todas las columnas del grupo.
2	Rango medio en todas las columnas del grupo.
3	Rango más alto en todas las columnas del grupo.
4	Rango más bajo en la primera columna, luego incrementado en uno por cada columna del grupo

El tercer argumento, **format**, especifica la representación de texto del resultado de la función:

Ejemplos de **format**

Valor	Descripción
0 (por defecto)	Valor bajo **&' - '&**valor alto de todas las columnas del grupo (por ejemplo 3 - 4).
1	Rango más bajo en todas las columnas del grupo.
2	valor bajo en la primera columna, vacío en las siguientes columnas del grupo.

El orden de las columnas para **mode** 4 y **format** 2 viene determinado por el criterio de ordenación de las dimensiones del gráfico.

Ejemplos:

```
HRank( sum( Sales ))
```

```
HRank( sum( Sales ), 2 )
```

```
HRank( sum( Sales ), 0, 1 )
```

Optimizar con k-means: Un ejemplo del mundo real

El ejemplo siguiente ilustra un caso de uso en el mundo real en el que las funciones de agrupamiento en clústeres y centroide de KMeans se aplican a un conjunto de datos. La función KMeans segrega los puntos de datos en grupos que comparten similitudes. Los clústeres se vuelven más compactos y diferenciados a medida que se aplica el algoritmo KMeans en un número configurable de iteraciones.

KMeans se utiliza en muchos campos para una amplia variedad de casos de uso. Algunos ejemplos de utilización de la agrupación en clústeres son la segmentación de clientes, la detección de fraudes, la predicción de pérdida de cuentas, la creación de incentivos para los clientes, la identificación de delitos cibernéticos y la optimización de las rutas de entrega. El algoritmo de agrupación en clústeres K Means se utiliza cada vez más cuando las empresas intentan inferir patrones y optimizar sus ofertas de servicios.

Qlik Sense Funciones KMeans y Centroide

Qlik Sense proporciona dos funciones KMeans que agrupan puntos de datos en grupos en función de la similitud. Vea *KMeans2D - función de gráfico (page 1407)* y *KMeansND - función de gráfico (page 1422)*. La función **KMeans2D** acepta dos dimensiones y funciona bien para visualizar resultados a través de un **gráfico de dispersión**. La función **KMeansND** acepta más de dos dimensiones. Como es fácil conceptualizar un resultado 2D en gráficos estándar, la siguiente demostración aplica KMeans en un **gráfico de dispersión** utilizando dos dimensiones. La agrupación de KMeans se puede visualizar coloreando por expresión o por dimensión, como se describe en este ejemplo.

Las funciones de centroide de Qlik Sense determinan la posición media aritmética de todos los puntos de datos en el grupo e identifican un punto central o centroide para ese grupo. Para cada fila del gráfico (o registro), la función de centroide muestra la coordenada del grupo al que se ha asignado este punto de datos. Vea *KMeansCentroid2D - función de gráfico (page 1437)* y *KMeansCentroidND - función de gráfico (page 1438)*.

Resumen de casos de uso y ejemplos

El ejemplo siguiente muestra un escenario simulado que podría darse en el mundo real. Una empresa textil en el estado de Nueva York, EE. UU., debe reducir sus gastos minimizando los costes de envío. Una forma de hacer esto es reubicando los almacenes más cerca de sus distribuidores. La empresa emplea a 118 distribuidores en todo el estado de Nueva York. La demostración siguiente simula cómo un gerente de operaciones podría segmentar a los distribuidores en cinco geografías agrupadas utilizando la función KMeans y luego identificar cinco ubicaciones óptimas de almacén, centrales para esos grupos, utilizando la función centroide. El objetivo es descubrir coordenadas cartográficas que se puedan utilizar para identificar cinco ubicaciones de almacén central.

El conjunto de datos

El conjunto de datos se basa en nombres y direcciones generados aleatoriamente en el estado de Nueva York con coordenadas reales de latitud y longitud. El conjunto de datos contiene las siguientes diez columnas: id, nombre, apellido, teléfono, dirección, ciudad, estado, código postal, latitud, longitud. El conjunto de datos está disponible a continuación como un archivo que puede descargar localmente y luego cargarlo a Qlik Sense o de forma inline para el editor de carga de datos. La app creada se denomina *Distributors KMeans and Centroid* y la primera hoja de la app se llama *Distribution cluster analysis*.

Seleccione el enlace siguiente para descargar el archivo de datos de muestra: DistributorData.csv

Conjunto de datos del Distributor : Carga inline para el editor de carga de datos en Qlik Sense (page 1405)

Título: DistributorData

Número total de registros: 118

Aplicar la función KMeans2D

En este ejemplo, la configuración de un **gráfico de dispersión** se demuestra mediante el conjunto de datos *DistributorData*, se aplica la función **KMeans2D** y el gráfico se colorea por dimensión.

Tenga en cuenta que las funciones KMeans de Qlik Sense admiten la agrupación automática mediante un método llamado diferencia de profundidad (DeD). Cuando un usuario define 0 como el número de clústeres, se determina el número óptimo de clústeres para ese conjunto de datos. Sin embargo, para este ejemplo se crea una variable para el argumento **num_clusters** (consulte la sintaxis en *KMeans2D - función de gráfico (page 1407)*). Por lo tanto, el número deseado de clústeres (k = 5) se especifica mediante una variable.

1. Se arrastra un **gráfico de dispersión** a la hoja y se denomina *Distribuidores (por dimensión)*.
2. Se crea una **variable** para especificar el número de clústeres. La **variable** se denomina *vDistClusters*. Como **Definición** de la variable se introduce *5*.
3. Configuración de **datos** para el gráfico:
    a. En **Dimensiones**, se selecciona el campo *id* como **Burbuja**. *Cluster ID* se escribe como **Etiqueta**.
    b. En **Medidas**, *Avg([latitude])* es la expresión para el **Eje X**.
    c. En **Medidas**, *Avg([longitude])* es la expresión para el **Eje Y**.
4. Configuración de **Aspecto**:

a. En **Colores y leyenda**, elija **Personalizado** para **Colores**.
b. Seleccione **Por dimensión** para dar color al gráfico.
c. Escriba la siguiente expresión: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only (longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
d. Seleccione la casilla **Colores persistentes**.

Gráfico de dispersión antes de aplicar el color de KMeans por dimensión

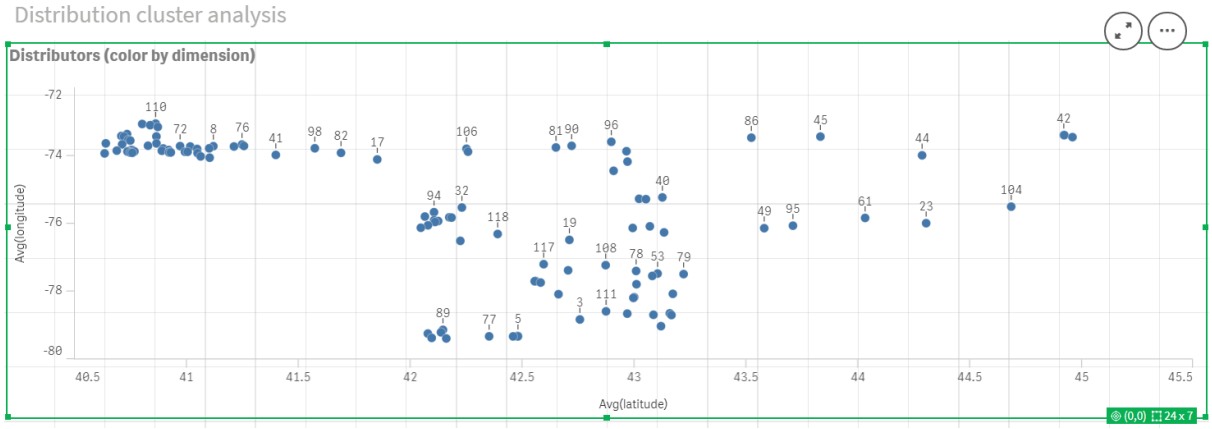
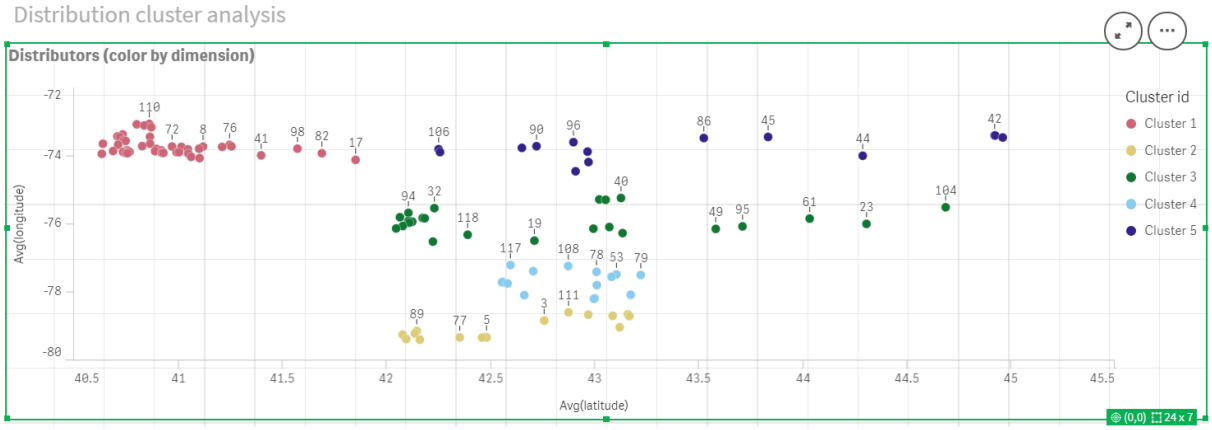


Gráfico de dispersión después de aplicar el color de KMeans por dimensión



Agregar una **tabla** : *Distribuidores*

Puede resultar útil tener una tabla a mano para acceder rápidamente a los datos relevantes. El **gráfico de dispersión** muestra los *id* aunque se agrega una tabla con los nombres de los distribuidores correspondientes como referencia.

1. Arrastre una **tabla** denominada *Distribuidores* a la hoja con las siguientes **Columnas** (Dimensiones) agregadas: *id*, *first_name* y *last_name*.

Tabla Nombres de distribuidores

Distributors

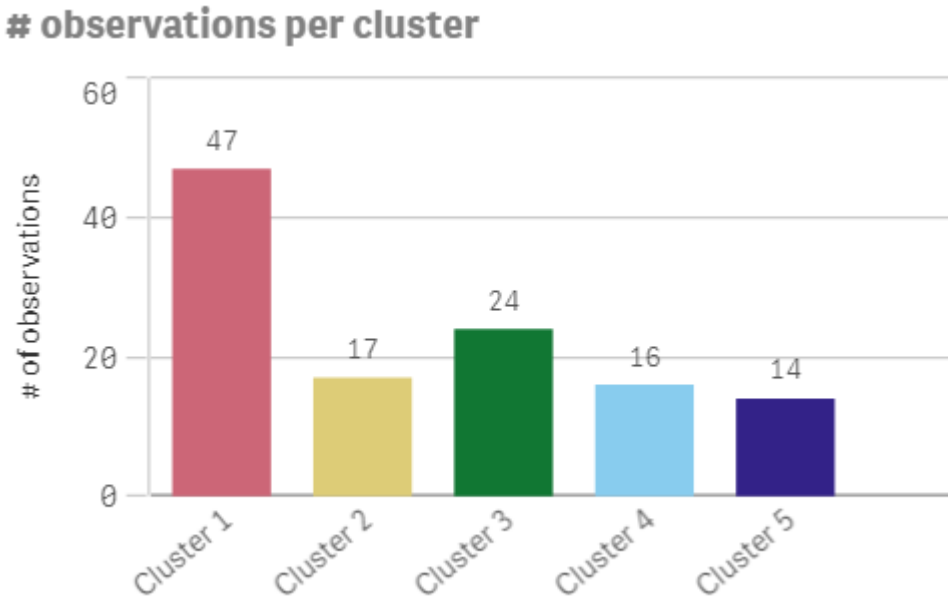
id	first_name	last_name
1	Kaiya	Snow
2	Dean	Roy
3	Eden	Paul
4	Bryanna	Higgins
5	Elisabeth	Lee
6	Skylar	Robinson
7	Cody	Bailey
8	Dario	Sims
9	Deacon	Hood

Agregar un **gráfico de barras**: *# observaciones por clúster*

Para el escenario de distribución de almacén, es útil saber cuántos distribuidores atenderá cada almacén. Por lo tanto, se crea un **gráfico de barras** que mide cuántos distribuidores se asignan a cada grupo.

1. Arrastre un **gráfico de barras** a la hoja. Denomine al gráfico: *# observaciones por clúster*.
2. Configuración de **Datos** para el **gráfico de barras**:
    a. Agregue una **Dimensión** denominada *Clústeres* (la etiqueta se puede agregar después de aplicar la expresión). Escriba la siguiente expresión: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    b. Agregue una **Medida** denominada *# de observaciones*. Escriba la siguiente expresión: *=count(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id))*
3. Configuración de **Aspecto**:
    a. En **Colores y leyenda**, elija **Personalizado** para **Colores**.
    b. Seleccione **Por dimensión** para dar color al gráfico.
    c. Escriba la siguiente expresión: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    d. Seleccione la casilla **Colores persistentes**.
    e. **Mostrar leyenda** está desactivado.
    f. En **Presentación**, **Etiquetas de valores** está en **Automático**.
    g. En **Eje X**: **Clústeres**, **Solo etiquetas** está seleccionado.

Gráfico de barras: # observaciones por clúster



Aplicar la función **Centroid2D**

Agregue una segunda tabla para la función **Centroid2D** que identificará las coordenadas para posibles ubicaciones de almacén. Esta tabla muestra la ubicación central (valores de centroide) para los cinco grupos de distribuidores identificados.

1. Arrastre una **Tabla** a la hoja y denomínela *Centroides de clúster* con las siguientes columnas agregadas:
    a. Agregue una **Dimensión** denominada *Clústeres*. Escriba la siguiente expresión: *=pick(aggr (KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1,'Warehouse 1','Warehouse 2','Warehouse 3','Warehouse 4','Warehouse 5'*)
    b. Agregue una **Medida** denominada *latitude (D1)*. Escriba la siguiente expresión: *=only(aggr (KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id))*
    Observe que el parámetro **coordinate_no** corresponde a la primera dimensión (0). En este caso, la *latitud* de la dimensión se traza contra el eje x. Si estuviéramos trabajando con la función **CentroidND** y hubiera hasta seis dimensiones, estas entradas de parámetros podrían ser cualquiera de estos seis valores: 0, 1, 2, 3, 4 o 5.
    c. Agregue una **Medida** denominada *longitude (D2)*. Escriba la siguiente expresión:*=only(aggr (KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id))*
    El parámetro **coordinate_no** en esta expresión corresponde a la segunda dimensión(1). La dimensión *longitude* se traza en el eje y.

Tabla: Cálculos de centroides de clúster

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

Correspondencias de centroides

El paso siguiente es asignar los centroides. Depende del desarrollador de la app si prefiere colocar la visualización en hojas separadas.

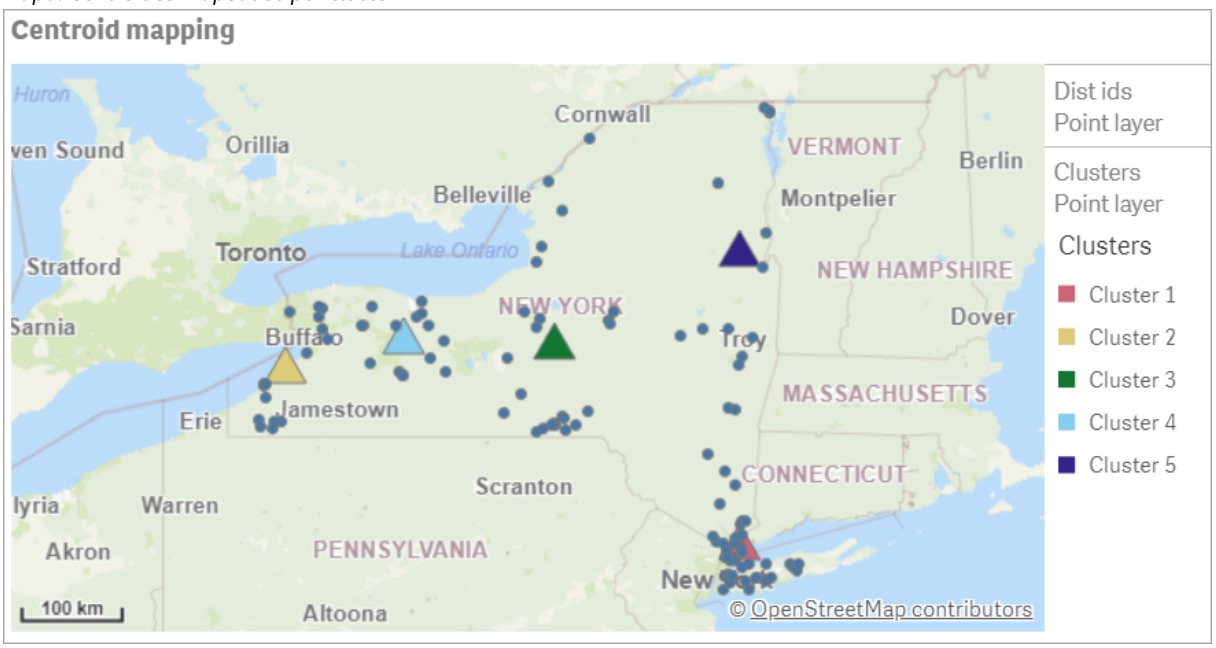
1. Arrastre un **mapa** denominado *Centroid mapping* a la hoja.
2. En la sección **Capas**. **Añadir capa** está seleccionado, luego seleccione **Capa de puntos**.
    a. Seleccione el *id* de **Campo** y añada la **Etiqueta** *Dist ids*.
    b. En la sección **Ubicación**, marque la casilla **Campos de latitud y longitud**.
    c. En **Latitud**, el campo *latitud* debe estar seleccionado.
    d. En **Longitud**, el campo *longitud* debe estar seleccionado.
    e. En la sección **Tamaño y forma**, seleccione **Burbuja** como **Forma** y el **Tamaño** se reduce al gusto en el control deslizante.
    f. En la sección **Colores**, seleccione **Color único** y seleccione azul como **Color** y gris para el **Color de borde** (según sus preferencias).
3. En la sección **Capas**, añada una segunda **Capa de puntos** seleccionando **Añadir capa** y luego seleccionando **Capa de puntos**.
    a. Escriba la siguiente expresión: *=aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)*
    b. Añada la **Etiqueta** *Clústeres*.
    c. En la sección **Ubicación**, marque la casilla **Campos de latitud y longitud**.
    d. En **Latitud** que en este caso se traza a lo largo del eje X, agregue la siguiente expresión: *=aggr (KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id)*
    e. En **Longitud** que en este caso se traza a lo largo del eje Y, agregue la siguiente expresión: *=aggr(KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id)*
    f. En la sección **Tamaño y forma**, seleccione **Triángulo** como **Forma** y reduzca el **Tamaño** a su gusto en el control deslizante.
    g. En **Colores y leyenda**, seleccione **Personalizado** en **Colores**.

h. Elija **Por dimensión** para dar color al gráfico. Escriba la siguiente expresión: *=pick(aggr (KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1,'Cluster 1','Cluster 2','Cluster 3','Cluster 4','Cluster 5')*

i. Denomine a la dimensión *Clústeres*.

4. En **Configuraciones de mapa**, seleccione **Adaptativo** en **Proyección**. Seleccione **Métrica** como **Unidades de medida.**

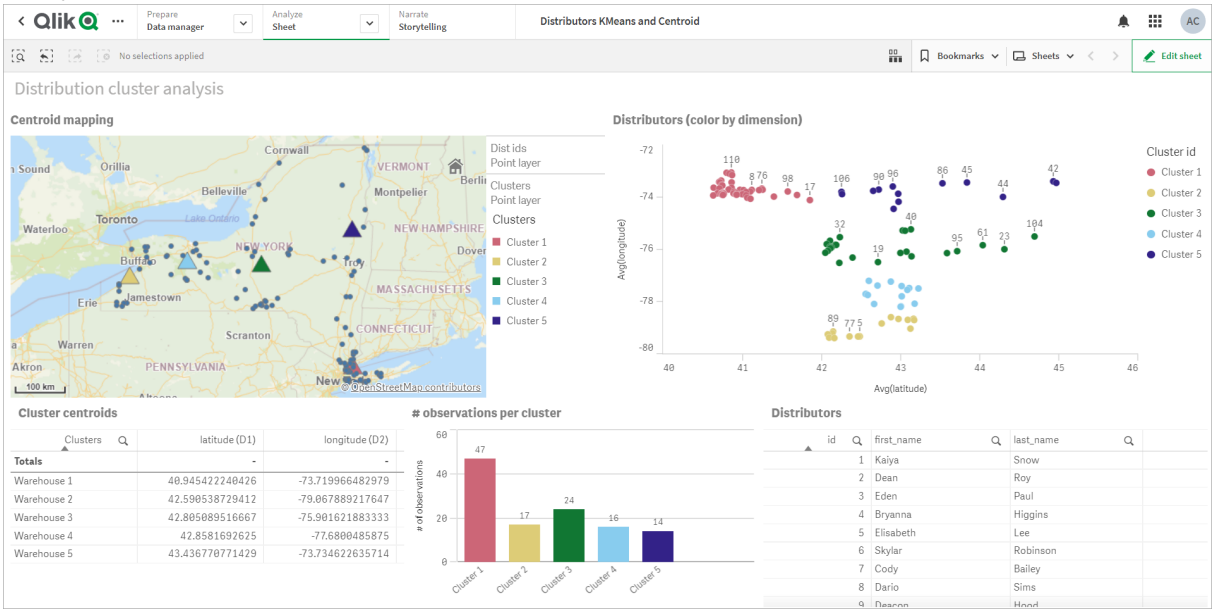
Mapa: Centroides mapeados por clúster



Conclusión

Utilizando la función KMeans para este escenario del mundo real, los distribuidores se han segmentado en grupos o clústeres similares en función de la similitud; en este caso, proximidad entre sí. La función Centroide se aplicó a esos grupos para identificar cinco coordenadas de mapas. Esas coordenadas proporcionan una ubicación central inicial en la que construir o ubicar almacenes. La función de centroide se aplica al gráfico de **mapa**, de modo que los usuarios de la app puedan visualizar dónde están ubicados los centroides en relación con los puntos de datos del clúster circundante. Las coordenadas resultantes representan ubicaciones potenciales de almacenes que podrían minimizar los costes de entrega a los distribuidores en el estado de Nueva York.

App: Ejemplo de análisis de KMeans y centroide



Conjunto de datos del Distributor : Carga inline para el editor de carga de datos en 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 - función de gráfico

KMeans2D() evalúa las filas del gráfico aplicando agrupación en clústeres k-means, y para cada fila del gráfico muestra el ID del grupo al que se ha asignado este punto de datos. Las columnas que utiliza el algoritmo de agrupamiento están determinadas por los parámetros coordinate_1 y coordinate_2, respectivamente. Ambas son agregaciones. El número de clústeres que se crea viene determinado por el parámetro num_clusters. Los datos se pueden normalizar opcionalmente mediante el parámetro norma.

KMeans2D devuelve un valor por punto de datos. El valor que devuelve es dual y es el valor del entero correspondiente al clúster al que se ha asignado cada punto de datos.

Sintaxis:

```
KMeans2D(num_clusters, coordinate_1, coordinate_2 [, norm])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
num_clusters	Entero que especifica el número de clústeres.
coordinate_1	La agregación que calcula la primera coordenada, generalmente el eje x del gráfico de dispersión que se puede hacer a partir del gráfico. El parámetro adicional, coordenada_2, calcula la segunda coordenada.
norm	El método de normalización opcional aplicado a los conjuntos de datos antes de la agrupación en clústeres KMeans. Valores posibles: 0 o "ninguno" para ninguna normalización 1 o "zscore" para una normalización de puntuación z 2 o "minmax" para la normalización mínima-máxima Si no se proporciona ningún parámetro o si el parámetro proporcionado es incorrecto, no se aplica ninguna normalización. Z-score normaliza los datos según la media de la característica y la desviación estándar. Z-score no asegura que cada característica tenga la misma escala, pero es un mejor enfoque que min-max cuando se trata de valores atípicos. La normalización mínimo-máximo asegura que las entidades tengan la misma escala tomando los valores mínimo y máximo de cada uno y recalculando cada punto de datos.

Ejemplo: Expresión de gráfico
En este ejemplo, creamos un gráfico de diagrama de dispersión utilizando el conjunto de datos *Iris* y luego usamos KMeans para colorear los datos por expresión.

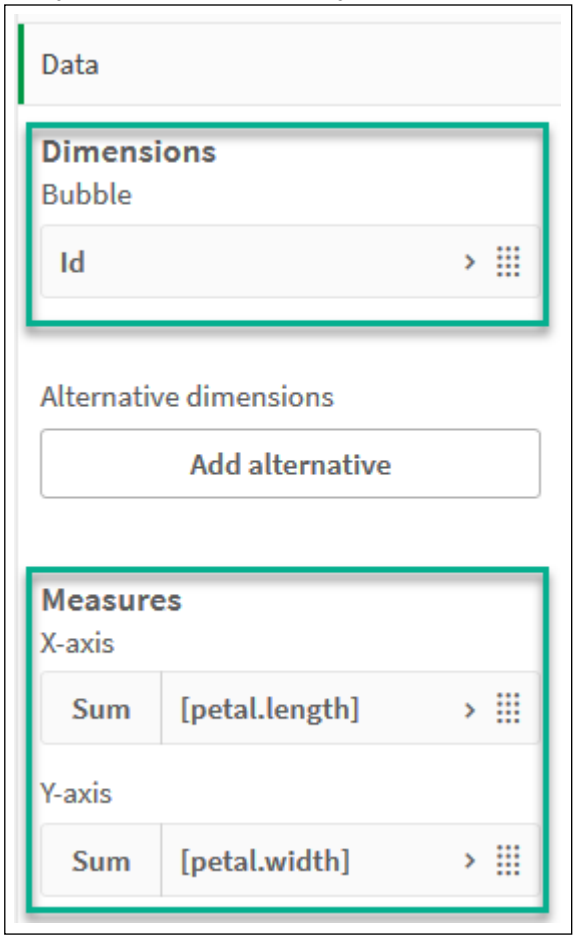
También creamos una variable para el argumento *num_clusters* y luego usamos un cuadro de entrada de variable para cambiar el número de clústeres.

El conjunto de datos *Iris* está disponible públicamente en una variedad de formatos. Hemos proporcionado los datos como una tabla inline para cargarla usando el editor de carga de datos de Qlik Sense. Tenga en cuenta que agregamos una columna *Id* a la tabla de datos para este ejemplo.

Tras cargar los datos en Qlik Sense, hacemos lo siguiente:

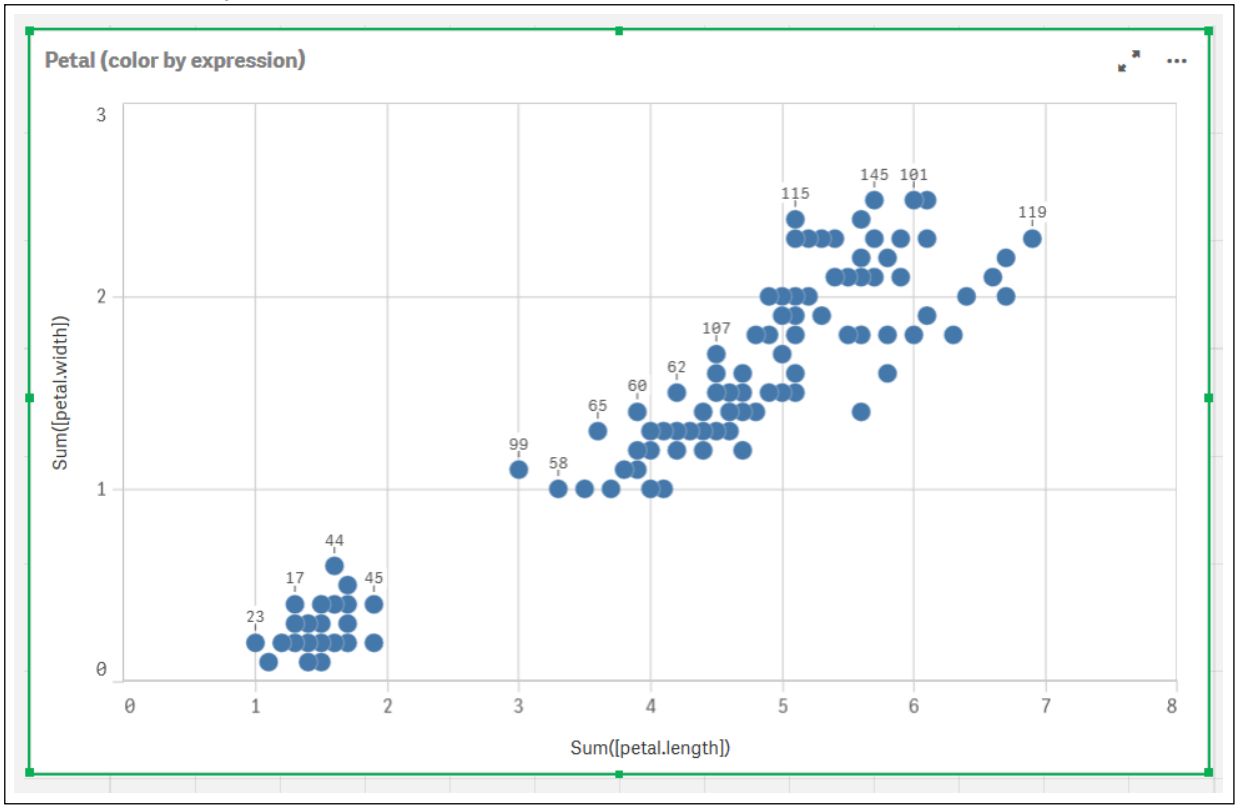
1. Arrastre un **Gráfico de dispersión** a una nueva hoja. Denomine el gráfico *Pétalo (colorear por expresión)*.
2. Cree una variable para especificar el número de clústeres. Para la variable **Nombre**, escriba *KmeansPetalClusters*. Para la variable **Definición**, escriba *=2*.
3. Configurar **Datos** para el gráfico:
    i. En **Dimensiones**, elija *ID* para el campo de **Burbuja**. Escriba el ID del clúster para la etiqueta.
    ii. En **Medidas**, elija *Sum([petal.length])* para la expresión para el **eje X**.
    iii. En **Medidas**, elija *Sum([petal.width])* para la expresión para el **eje Y**.

    Configuraciones de datos para el gráfico Pétalo (Colorear por expresión)



Los puntos de datos se trazan en el gráfico.

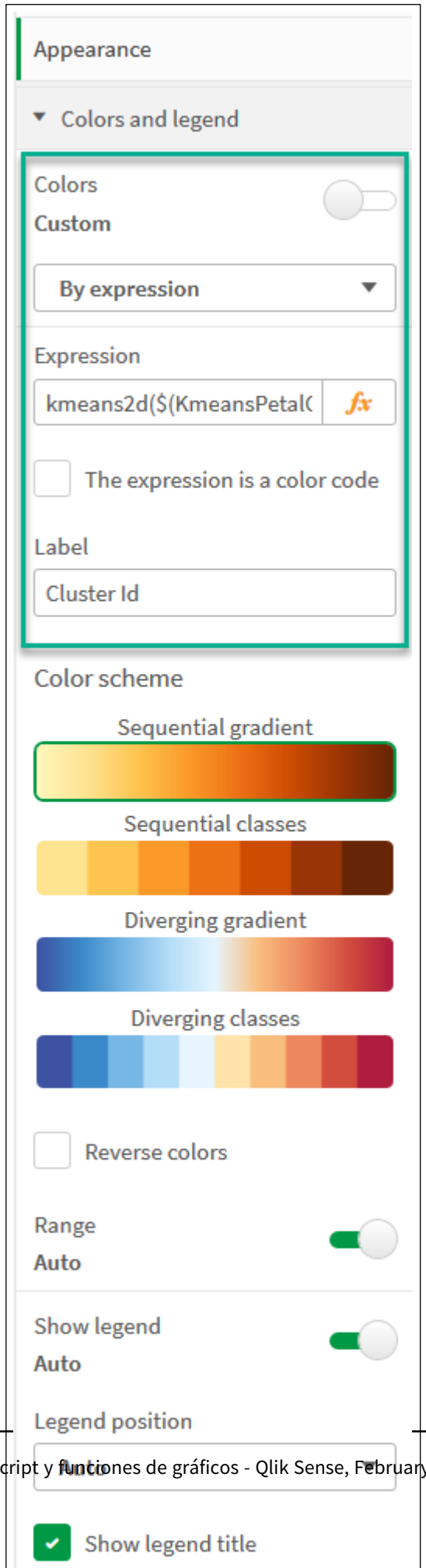
Puntos de datos en el gráfico Pétalo (Colorear por expresión)



4. Configurar **Aspecto** para el gráfico:
    i. En **Colores y leyenda**, elija **Personalizado** para **Colores**.
    ii. Elija colorear el gráfico **Por expresión**.
    iii. Inserte lo siguiente para **Expresión**: *kmeans2d($(KmeansPetalClusters), Sum([petal.length]), Sum([petal.width]))*
    Observe que *KmeansPetalClusters* es la variable que configuramos en *2*.
    Alternativamente, inserte lo siguiente: *kmeans2d(2, Sum([petal.length]), Sum([petal.width]))*
    iv. Desmarque la casilla de verificación para **La expresión es un código de color**.

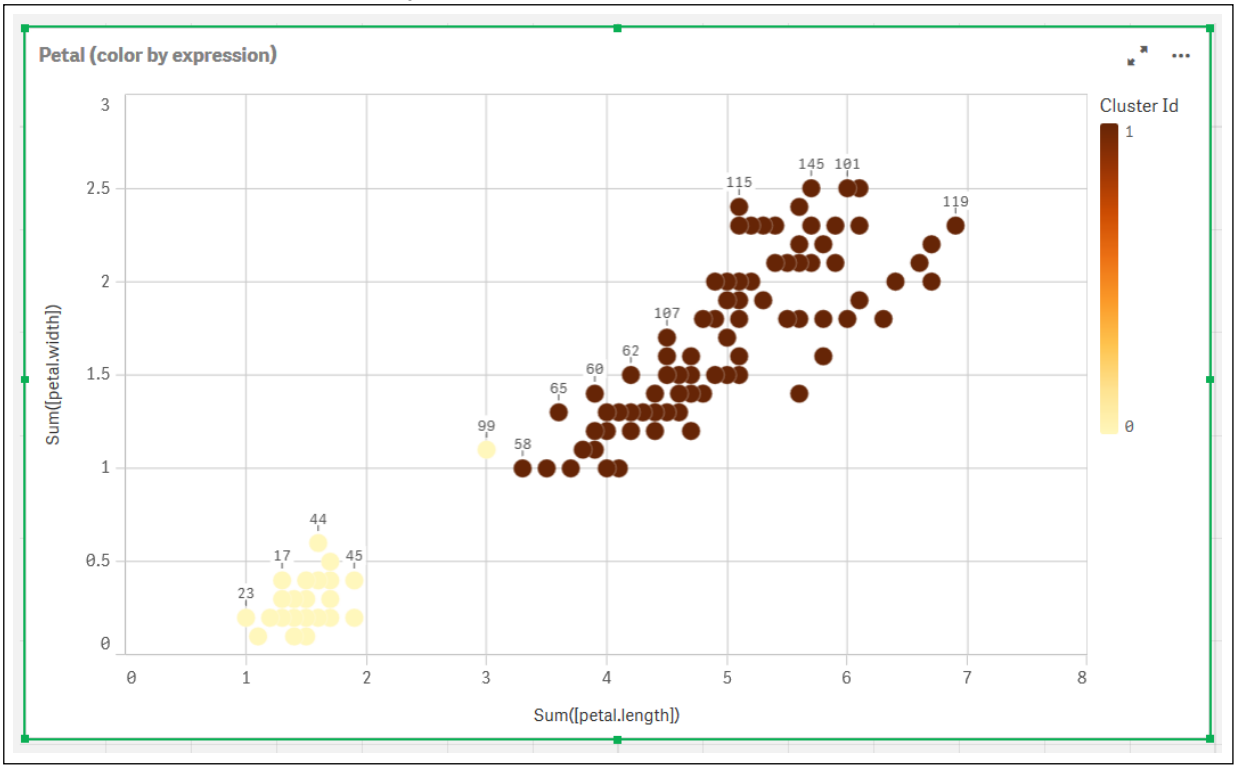
v. Escriba lo siguiente para **Etiqueta**: *Cluster ID*

Configuraciones de aspecto para el gráfico Pétalo (colorear por expresión)


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

Los dos grupos del gráfico están coloreados por la expresión KMeans.

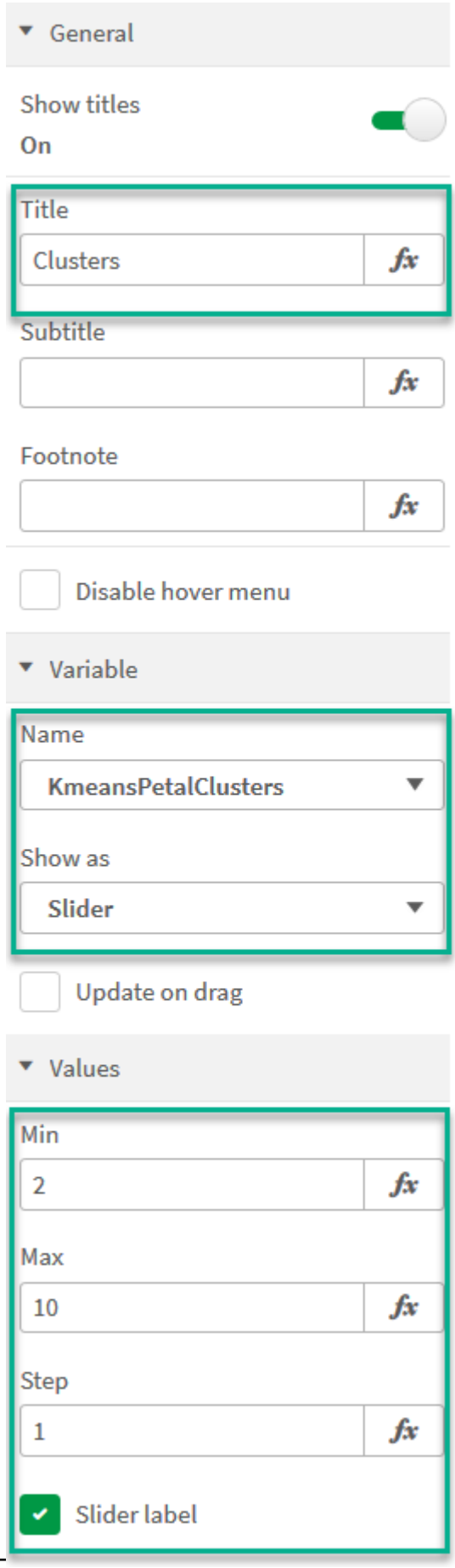
Clústeres coloreados por expresión en el gráfico Pétalo (colorear por expresión)



5. Agregue un cuadro de **Entrada de variable** para el número de clústeres.
    i. En **Objetos personalizados** en el panel de **Activos**, elija **Qlik Dashboard bundle**. Si no tuviéramos acceso al paquete Dashboard bundle, todavía podemos cambiar el número de clústeres usando la variable que creamos o directamente introduciendo un número entero en la expresión.
    ii. Arrastre un cuadro de **Entrada de variable** a la hoja.
    iii. En **Aspecto**, haga clic en **General**.
    iv. Escriba lo siguiente como **Título**: *Clústeres*
    v. Haga clic en **Variable**.
    vi. Elija la siguiente variable como **Nombre**: *KmeansPetalClusters*.
    vii. Elija **Deslizador** en **Mostrar como**.

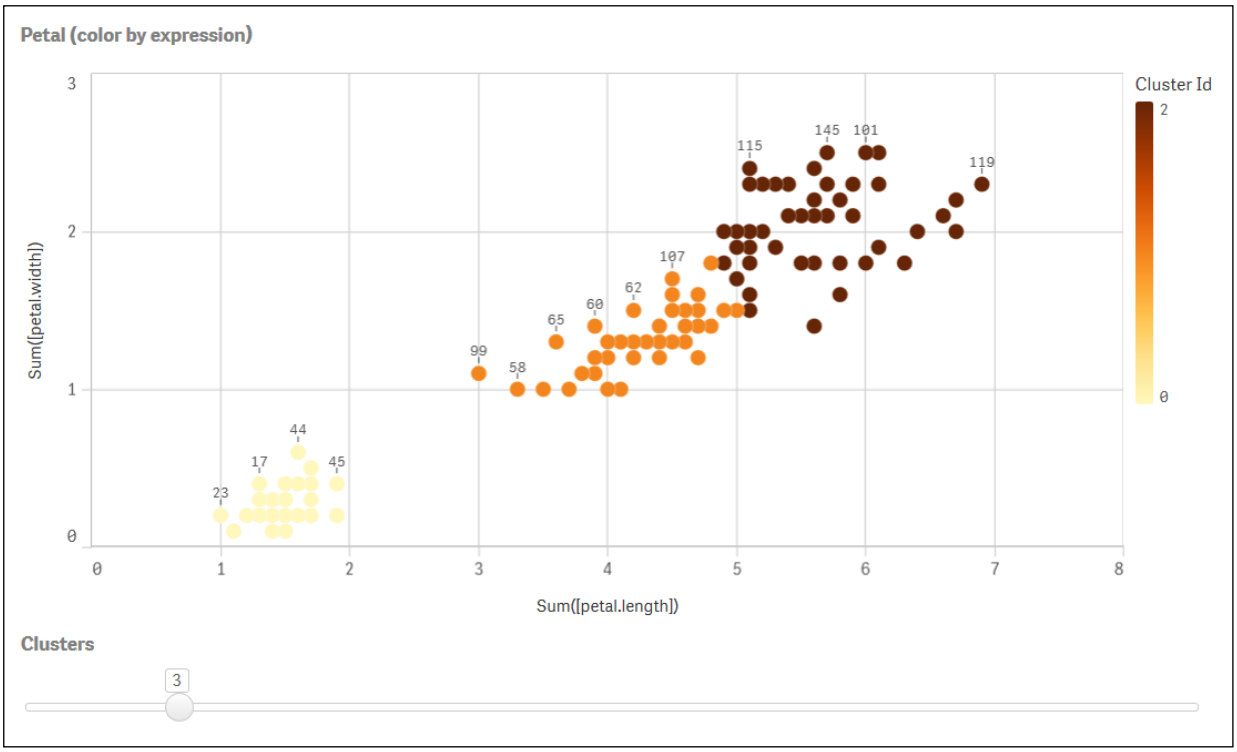
viii. Elija **Valores** y configure los ajustes según sea necesario.

Aspecto del cuadro de entrada de variable Clústeres


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

Cuando termine de editar, puede cambiar el número de clústeres usando el control deslizante en el cuadro de entrada de la variable *Clústeres*.

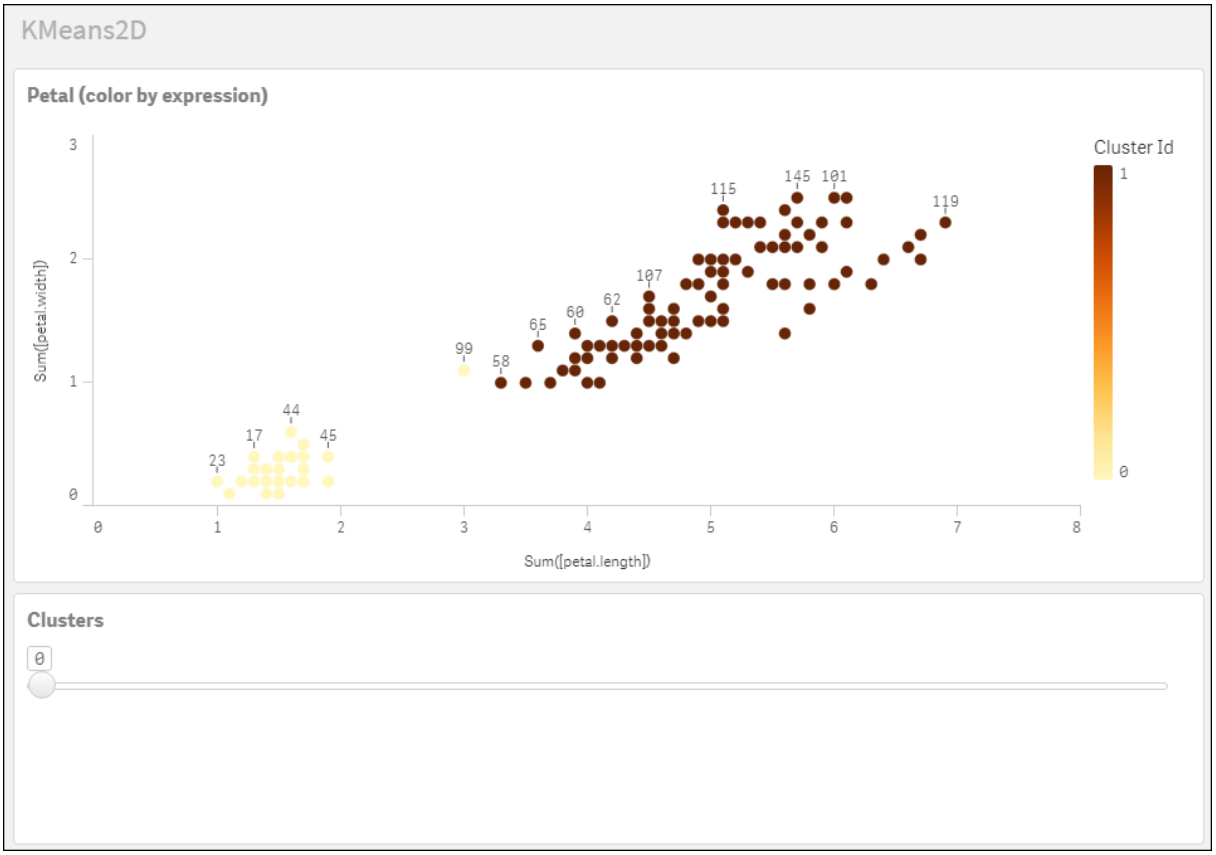
Clústeres coloreados por expresión en el gráfico Pétalo (colorear por expresión)



Agrupamiento automático

Las funciones **KMeans** admiten la agrupación automática mediante un método llamado diferencia de profundidad (DeD). Cuando un usuario define 0 como el número de clústeres, se determina un número óptimo de clústeres para ese conjunto de datos. Tenga en cuenta que, si bien no se devuelve explícitamente un número entero para el número de clústeres (*k*), se calcula dentro del algoritmo KMeans. Por ejemplo, si se especifica 0 en la función para el valor de *KmeansPetalClusters* o se establece a través de un cuadro de entrada variable, las asignaciones de clústeres se calculan automáticamente para el conjunto de datos en función de un número óptimo de clústeres.

El método de diferencia de profundidad de Kmeans determina el número óptimo de grupos cuando (k) se establece en 0



Conjunto de datos Iris de : Carga inline para el editor de carga de datos en 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 - función de gráfico

KMeansND() evalúa las filas del gráfico aplicando agrupación en clústeres k-means, y para cada fila del gráfico muestra el ID del grupo al que se ha asignado este punto de datos. Las columnas que utiliza el algoritmo de agrupamiento vienen determinadas por los parámetros coordinate_1 y coordinate_2, etc., hasta n columnas. Esto son todo agregaciones. El número de clústeres que se crea viene determinado por el parámetro num_clusters.

KMeansND devuelve un valor por punto de datos. El valor que devuelve es dual y es el valor del entero correspondiente al clúster al que se ha asignado cada punto de datos.

Sintaxis:

```
KMeansND(num_clusters, num_iter, coordinate_1, coordinate_2 [,coordinate_3 [,
...]])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
num_clusters	Entero que especifica el número de clústeres.
num_iter	El número de repeticiones de agrupación con centros de agrupación (clústeres) reinicializados.
coordinate_1	La agregación que calcula la primera coordenada, generalmente el eje x (de un gráfico de dispersión que se puede hacer a partir del gráfico). Los parámetros adicionales calculan la segunda, tercera y cuarta coordenadas, etc.

Ejemplo: Expresión de gráfico
En este ejemplo, creamos un gráfico de diagrama de dispersión utilizando el conjunto de datos *Iris* y luego usamos KMeans para colorear los datos por expresión.

También creamos una variable para el argumento *num_clusters* y luego usamos un cuadro de entrada de variable para cambiar el número de clústeres.

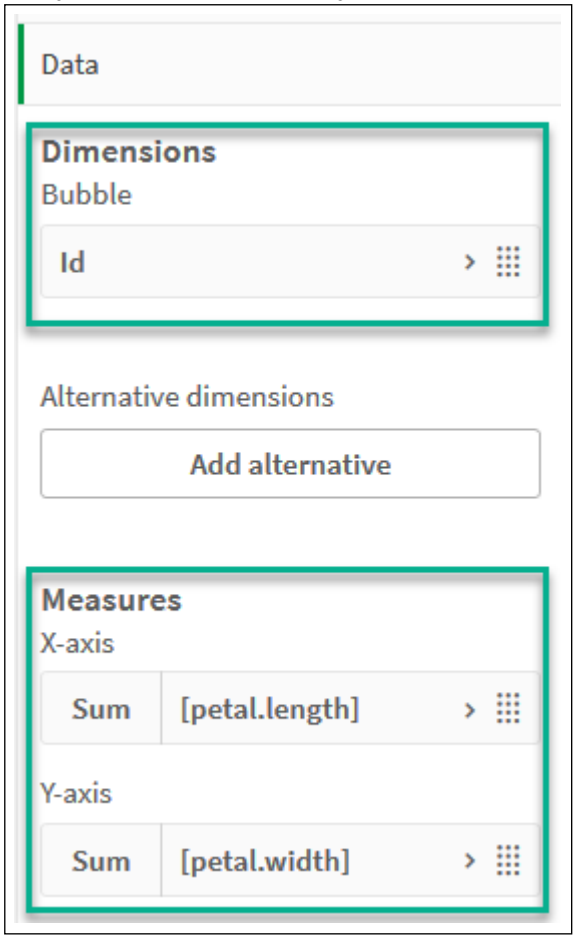
También creamos una variable para el argumento *num_iter* y luego usamos un segundo cuadro de entrada de variable para cambiar el número de iteraciones.

El conjunto de datos *Iris* está disponible públicamente en una variedad de formatos. Hemos proporcionado los datos como una tabla inline para cargarla usando el editor de carga de datos de Qlik Sense. Tenga en cuenta que agregamos una columna *Id* a la tabla de datos para este ejemplo.

Tras cargar los datos en Qlik Sense, hacemos lo siguiente:

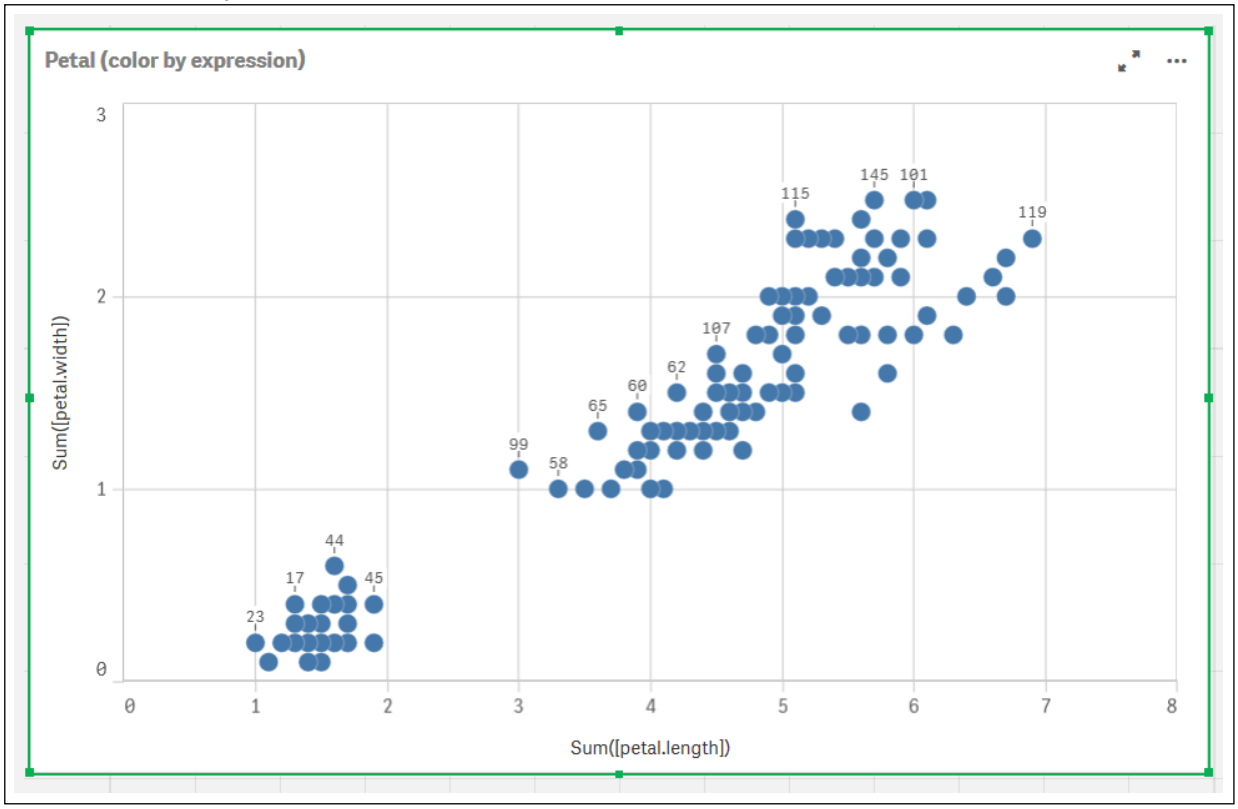
1. Arrastre un **Gráfico de dispersión** a una nueva hoja. Denomine el gráfico *Pétalo (colorear por expresión)*.
2. Cree una variable para especificar el número de clústeres. Para la variable **Nombre**, escriba *KmeansPetalClusters*. Para la variable **Definición**, escriba *=2*.
3. Cree una variable para especificar el número de iteraciones. Para la variable **Nombre**, escriba *KmeansNumberIterations*. Para la variable **Definición**, escriba *=1*.
4. Configurar **Datos** para el gráfico:
    i. En **Dimensiones**, elija *ID* para el campo de **Burbuja**. Escriba el ID del clúster para la etiqueta.
    ii. En **Medidas**, elija *Sum([petal.length])* para la expresión para el **eje X**.
    iii. En **Medidas**, elija *Sum([petal.width])* para la expresión para el **eje Y**.

Configuraciones de datos para el gráfico Pétalo (Colorear por expresión)



Los puntos de datos se trazan en el gráfico.

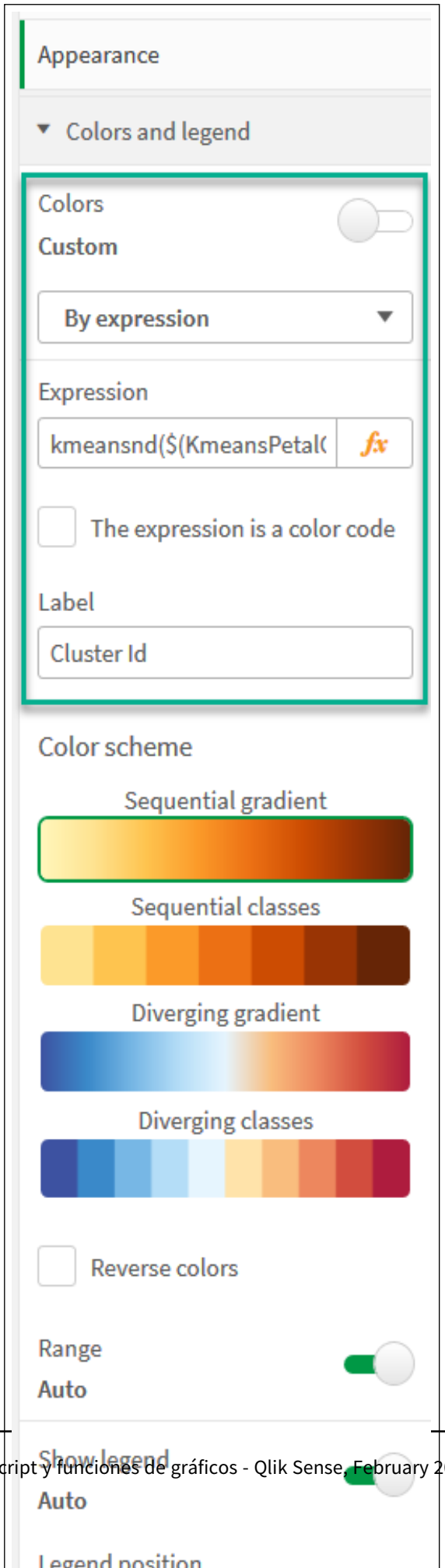
Puntos de datos en el gráfico Pétalo (Colorear por expresión)



5. Configurar **Aspecto** para el gráfico:
    i. En **Colores y leyenda**, elija **Personalizado** para **Colores**.
    ii. Elija colorear el gráfico **Por expresión**.
    iii. Inserte lo siguiente para **Expresión**: *kmeansnd ($(KmeansPetalClusters),$(KmeansNumberIterations), Sum([petal.length]), Sum ([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
    Observe que *KmeansPetalClusters* es la variable que configuramos en *2*.
    KmeansNumberIterations es la variable que configuramos en *1*.
    Alternativamente, inserte lo siguiente: *kmeansnd(2, 2, Sum([petal.length]), Sum ([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
    iv. Desmarque la casilla de verificación para **La expresión es un código de color**.

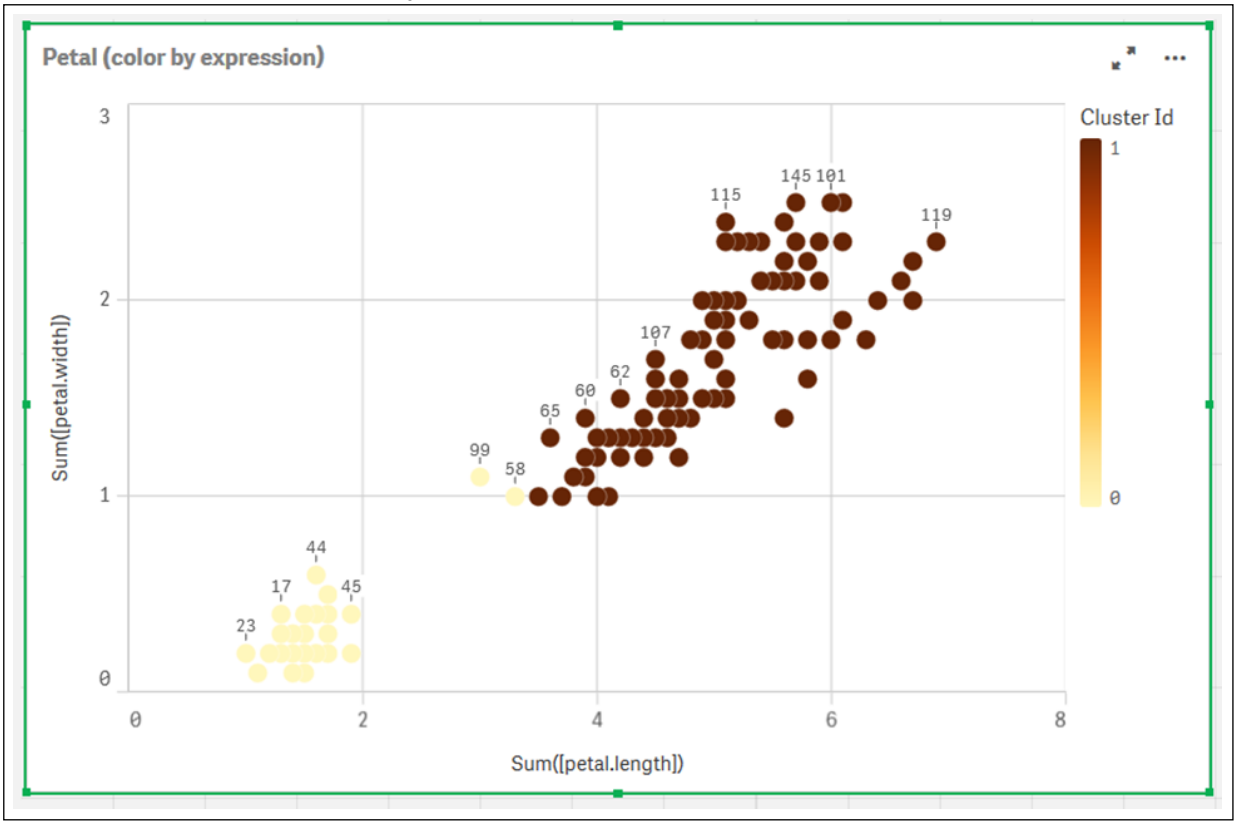
v. Escriba lo siguiente para **Etiqueta**: *Cluster ID*

Configuraciones de aspecto para el gráfico Pétalo (colorear por expresión)


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

Los dos grupos del gráfico están coloreados por la expresión KMeans.

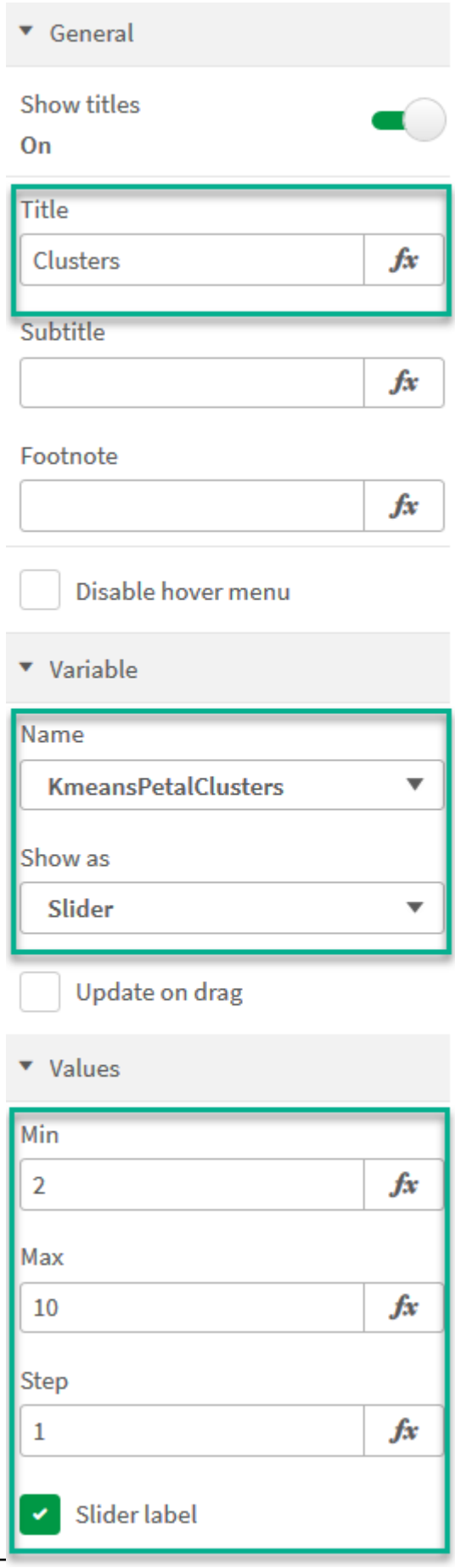
Clústeres coloreados por expresión en el gráfico Pétalo (colorear por expresión)



6. Agregue un cuadro de **Entrada de variable** para el número de clústeres.
    i. En **Objetos personalizados** en el panel de **Activos**, elija **Qlik Dashboard bundle**. Si no tuviéramos acceso al paquete Dashboard bundle, todavía podemos cambiar el número de clústeres usando la variable que creamos o directamente introduciendo un número entero en la expresión.
    ii. Arrastre un cuadro de **Entrada de variable** a la hoja.
    iii. En **Aspecto**, haga clic en **General**.
    iv. Escriba lo siguiente como **Título**: *Clústeres*
    v. Haga clic en **Variable**.
    vi. Elija la siguiente variable como **Nombre**: *KmeansPetalClusters*.
    vii. Elija **Deslizador** en **Mostrar como**.

viii. Elija **Valores** y configure los ajustes según sea necesario.

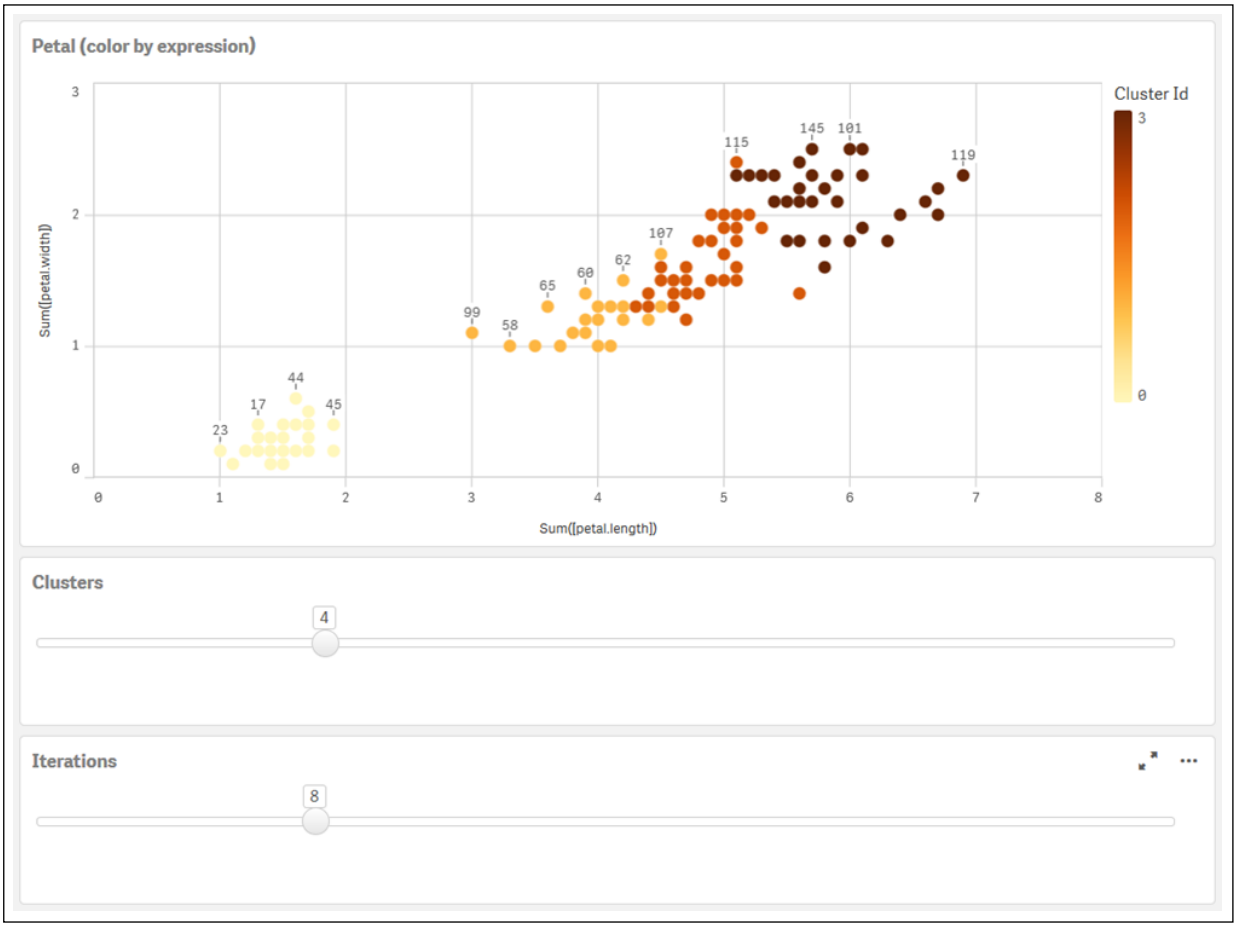
Aspecto del cuadro de entrada de variable Clústeres


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. Agregue un cuadro de **Entrada de variable** para el número de iteraciones.
    i. Arrastre un cuadro de **Entrada de variable** a la hoja.
    ii. En **Aspecto**, elija **General**.
    iii. Escriba lo siguiente como **Título**: *Iteraciones*
    iv. En **Aspecto**, elija **Variable**.
    v. Elija la siguiente variable en **Nombre**: *KmeansNumberIterations*.
    vi. Configure los ajustes adicionales según sea necesario.

    Ahora podemos cambiar el número de clústeres e iteraciones usando los controles deslizantes de los cuadros de entrada de variables.

    Clústeres coloreados por expresión en el gráfico Pétalo (colorear por expresión)

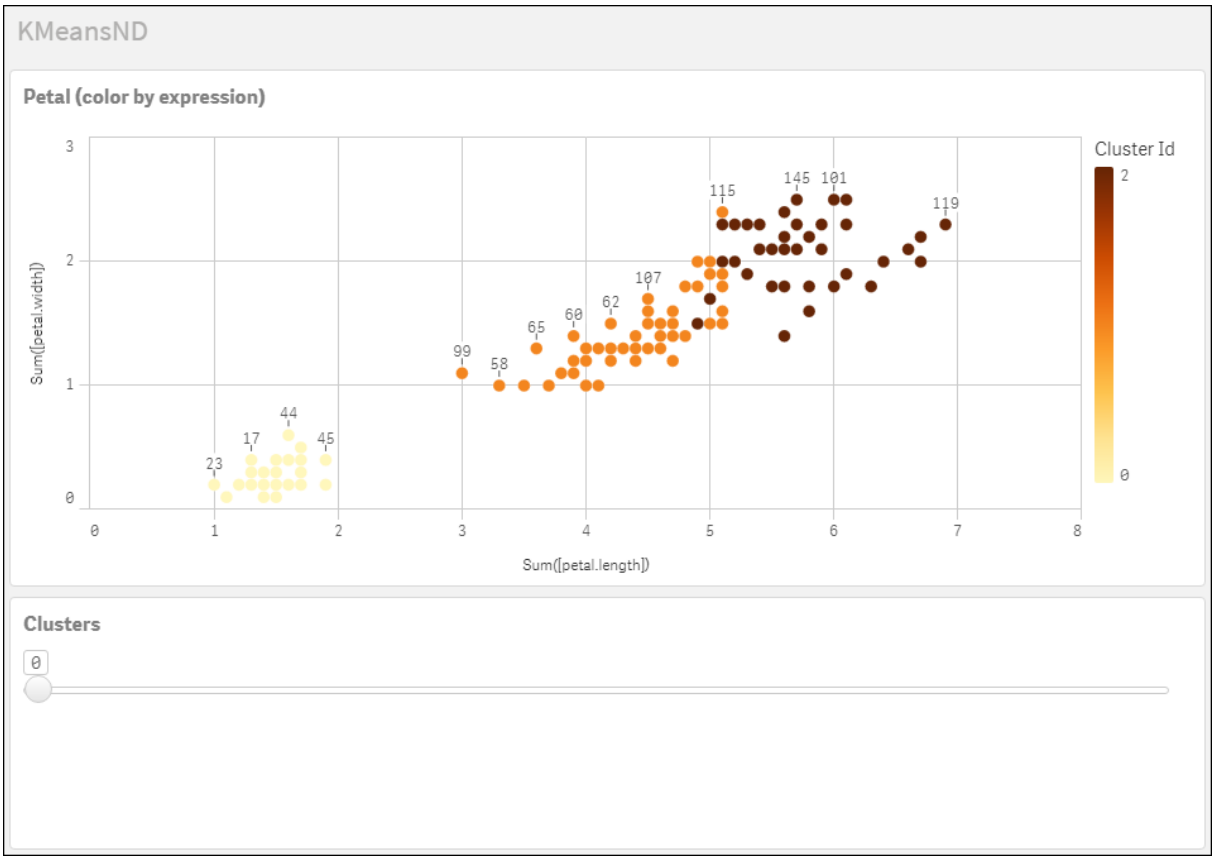


Agrupamiento automático

Las funciones **KMeans** admiten la agrupación automática mediante un método llamado diferencia de profundidad (DeD). Cuando un usuario define 0 como el número de clústeres, se determina un número óptimo de clústeres para ese conjunto de datos. Tenga en cuenta que, si bien no se devuelve explícitamente un número entero para el número de clústeres (*k*), se calcula dentro del algoritmo KMeans. Por ejemplo, si se especifica 0 en la función para el valor de *KmeansPetalClusters* o se establece a través de un cuadro de

entrada variable, las asignaciones de clústeres se calculan automáticamente para el conjunto de datos en función de un número óptimo de clústeres. Dado el conjunto de datos Iris, si se selecciona 0 para el número de clústeres, el algoritmo determinará (agrupamiento automático) un número óptimo de grupos (3) para este conjunto de datos.

El método de diferencia de profundidad de KMeans determina el número óptimo de grupos cuando (k) se establece en 0.



Conjunto de datos Iris de : Carga inline para el editor de carga de datos en 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 - función de gráfico

KMeansCentroid2D() evalúa las filas del gráfico aplicando agrupación en clústeres k-means y para cada fila del gráfico muestra la coordenada deseada del clúster al que se haya asignado este punto de datos. Las columnas que utiliza el algoritmo de agrupamiento vienen determinadas por los parámetros coordinate_1 y coordinate_2, respectivamente. Ambas son agregaciones. El número de clústeres que se crea viene determinado por el parámetro num_clusters. Los datos se pueden normalizar opcionalmente mediante el parámetro norma.

KMeansCentroid2D devuelve un valor por punto de datos. El valor que devuelve es dual y es una de las coordenadas de la posición correspondiente al centro de agrupación al que se ha asignado el punto de datos.

Sintaxis:

```
KMeansCentroid2D(num_clusters, coordinate_no, coordinate_1, coordinate_2 [, norm])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
num_clusters	Entero que especifica el número de clústeres.

Argumento	Descripción
coordinate_no	El número de coordenadas deseado de los centroides (correspondiente, por ejemplo, al eje x, y o z).
coordinate_1	La agregación que calcula la primera coordenada, generalmente el eje x del gráfico de dispersión que se puede hacer a partir del gráfico. El parámetro adicional, coordenada_2, calcula la segunda coordenada.
norm	El método de normalización opcional aplicado a los conjuntos de datos antes de la agrupación en clústeres KMeans. Valores posibles: 0 o "ninguno" para ninguna normalización 1 o "zscore" para una normalización de puntuación z 2 o "minmax" para la normalización mínima-máxima Si no se proporciona ningún parámetro o si el parámetro proporcionado es incorrecto, no se aplica ninguna normalización. Z-score normaliza los datos según la media de la característica y la desviación estándar. Z-score no asegura que cada característica tenga la misma escala, pero es un mejor enfoque que min-max cuando se trata de valores atípicos. La normalización mínimo-máximo asegura que las entidades tengan la misma escala tomando los valores mínimo y máximo de cada uno y recalculando cada punto de datos.

Agrupamiento automático

Las funciones **KMeans** admiten la agrupación automática mediante un método llamado diferencia de profundidad (DeD). Cuando un usuario define 0 como el número de clústeres, se determina un número óptimo de clústeres para ese conjunto de datos. Tenga en cuenta que, si bien no se devuelve explícitamente un número entero para el número de clústeres (*k*), se calcula dentro del algoritmo KMeans. Por ejemplo, si se especifica 0 en la función para el valor de *KmeansPetalClusters* o se establece a través de un cuadro de entrada variable, las asignaciones de clústeres se calculan automáticamente para el conjunto de datos en función de un número óptimo de clústeres.

KMeansCentroidND - función de gráfico

KMeansCentroidND() evalúa las filas del gráfico aplicando agrupación en clústeres k-means, y para cada fila del gráfico muestra la coordenada deseada del clúster al que se ha asignado este punto de datos. Las columnas que utiliza el algoritmo de agrupamiento vienen determinadas por los parámetros coordinate_1 y coordinate_2, etc., hasta n columnas. Esto son todo agregaciones. El número de clústeres que se crea viene determinado por el parámetro num_clusters.

KMeansCentroidND devuelve un valor por fila. El valor que devuelve es dual y es una de las coordenadas de la posición correspondiente al centro de clúster al que se ha asignado el punto de datos.

Sintaxis:

```
KMeansCentroidND((num_clusters, num_iter, coordinate_no, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
num_clusters	Entero que especifica el número de clústeres.
num_iter	El número de repeticiones de agrupación con centros de agrupación (clústeres) reinicializados.
coordinate_no	El número de coordenadas deseado de los centroides (correspondiente, por ejemplo, al eje x, y o z).
coordinate_1	La agregación que calcula la primera coordenada, generalmente el eje x (de un gráfico de dispersión que se puede hacer a partir del gráfico). Los parámetros adicionales calculan la segunda, tercera y cuarta coordenadas, etc.

Agrupamiento automático

Las funciones **KMeans** admiten la agrupación automática mediante un método llamado diferencia de profundidad (DeD). Cuando un usuario define 0 como el número de clústeres, se determina un número óptimo de clústeres para ese conjunto de datos. Tenga en cuenta que, si bien no se devuelve explícitamente un número entero para el número de clústeres (*k*), se calcula dentro del algoritmo KMeans. Por ejemplo, si se especifica 0 en la función para el valor de *KmeansPetalClusters* o se establece a través de un cuadro de entrada variable, las asignaciones de clústeres se calculan automáticamente para el conjunto de datos en función de un número óptimo de clústeres.

STL_Trend - función de gráfico

STL_Trend es una función de descomposición de series de tiempo. Junto con **STL_Seasonal** y **STL_Residual**, esta función se utiliza para descomponer una serie temporal en componentes estacionales, de tendencia y residuales. En el contexto del algoritmo STL, la descomposición de series de tiempo se utiliza para identificar tanto un patrón estacional recurrente como una tendencia general, dada una métrica de entrada y otros parámetros. La función **STL_Trend** identifica una tendencia general, independiente de patrones o ciclos estacionales, a partir de datos de series de tiempo.

Las tres funciones STL están relacionadas con la métrica de entrada a través de una simple suma:

STL_Trend + **STL_Seasonal** + **STL_Residual** = Métrica de entrada

STL (descomposición estacional y de tendencia usando Loess) emplea técnicas de suavizado de datos y, a través de sus parámetros de entrada, permite al usuario ajustar la periodicidad de los cálculos que realiza. Esta periodicidad determina cómo se segmenta en el análisis la dimensión temporal de la métrica de entrada (una medida).

Como mínimo, **STL_Trend** toma una métrica de entrada (`target_measure`) y un valor entero para su `period_int`, devolviendo un valor de punto flotante. La métrica de entrada tendrá la forma de una agregación que varía a lo largo de la dimensión de tiempo. Opcionalmente, puede incluir valores para `seasonal_smoother` y `trend_smoother`, para ajustar el algoritmo de suavizado.

Puede trabajar con esta función insertándola directamente en el editor de expresiones de un gráfico o agregando el modificador de **Descomposición de series de tiempo** a una medida. También está disponible para su uso en Insight Advisor como tipo de análisis.

Sintaxis:

```
STL_Trend(target_measure, period_int [,seasonal_smoother [,trend_smoother]])
```

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
target_measure	La medida para descomponer en componentes estacionales y de tendencia. Esta debería ser una medida como Sum(Sales) o Sum(Passengers) que varía a lo largo de la dimensión del tiempo. Este no debe ser un valor constante.
period_int	La periodicidad del conjunto de datos. Este parámetro es un valor entero que representa el número de pasos discretos que componen un período, o ciclo estacional, de la señal. Por ejemplo, si la serie temporal está segmentada en una sección para cada trimestre del año, debe establecer el **period_int** en un valor de 4 para definir la periodicidad como Año.
seasonal_smoother	Longitud del suavizador estacional. Debe ser un entero impar. El suavizador estacional utiliza datos para una fase en particular de la variación estacional, durante varios períodos. Se utiliza un paso discreto de la dimensión temporal de cada período. El suavizador estacional indica el número de períodos utilizados para suavizar. Por ejemplo, si la dimensión de tiempo está segmentada por mes y el período es Año (12), el componente estacional se calculará de modo que cada mes particular de cada año se calcule a partir de los datos del mismo mes, tanto en ese año como en años adyacentes. El valor de **seasonal_smoother** es el número de años empleados para suavizar.
trend_smoother	Longitud del suavizador de tendencias. Debe ser un entero impar. El suavizador de tendencias utiliza la misma escala de tiempo que el parámetro **period_int** y su valor es el número de gránulos usados para suavizar. Por ejemplo, si una serie de tiempo está segmentada por mes, la tendencia más suave será la cantidad de meses utilizados para suavizar.

La función de gráfico **STL_Trend** se utiliza a menudo en combinación con las siguientes funciones:

Funciones relacionadas

Función	Interacción
STL_Seasonal - función de gráfico (page 1441)	Esta es la función que se utiliza para calcular el componente estacional de una serie temporal.
STL_Residual - función de gráfico (page 1443)	Al desglosar una métrica de entrada en componente estacional y de tendencia, parte de la variación de la medida no se ajustará a ninguno de los dos componentes principales. La función **STL_ Residual** calcula esta parte de la descomposición.

Para ver un tutorial con un ejemplo completo que muestra cómo usar esta función, consulte *Tutorial: Descomposición de series de tiempo en Qlik Sense (page 1445)*.

STL_Seasonal - función de gráfico

STL_Seasonal es una función de descomposición de series de tiempo. Junto con **STL_Trend** y **STL_Residual**, esta función se utiliza para descomponer una serie temporal en componentes estacionales, de tendencia y residuales. En el contexto del algoritmo STL, la descomposición de series de tiempo se utiliza para identificar tanto un patrón estacional recurrente como una tendencia general, dada una métrica de entrada y otros parámetros. La función **STL_Seasonal** puede identificar un patrón estacional dentro de una serie temporal, separándolo de la tendencia general que muestran los datos.

Las tres funciones STL están relacionadas con la métrica de entrada a través de una simple suma:

STL_Trend + **STL_Seasonal** + **STL_Residual** = Métrica de entrada

STL (descomposición estacional y de tendencia usando Loess) emplea técnicas de suavizado de datos y, a través de sus parámetros de entrada, permite al usuario ajustar la periodicidad de los cálculos que realiza. Esta periodicidad determina cómo se segmenta en el análisis la dimensión temporal de la métrica de entrada (una medida).

Como mínimo, **STL_Seasonal** toma una métrica de entrada (`target_measure`) y un valor entero para su `period_int`, devolviendo un valor de punto flotante. La métrica de entrada tendrá la forma de una agregación que varía a lo largo de la dimensión de tiempo. Opcionalmente, puede incluir valores para `seasonal_smoother` y `trend_smoother`, para ajustar el algoritmo de suavizado.

Puede trabajar con esta función insertándola directamente en el editor de expresiones de un gráfico o agregando el modificador de **Descomposición de series de tiempo** a una medida. También está disponible para su uso en Insight Advisor como tipo de análisis.

Sintaxis:

```
STL_Seasonal(target_measure, period_int [,seasonal_smoother [,trend_
smoother]])
```

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
target_measure	La medida para descomponer en componentes estacionales y de tendencia. Esta debería ser una medida como Sum(Sales) o Sum(Passengers) que varía a lo largo de la dimensión del tiempo. Este no debe ser un valor constante.
period_int	La periodicidad del conjunto de datos. Este parámetro es un valor entero que representa el número de pasos discretos que componen un período, o ciclo estacional, de la señal. Por ejemplo, si la serie temporal está segmentada en una sección para cada trimestre del año, debe establecer el **period_int** en un valor de 4 para definir la periodicidad como Año.
seasonal_smoother	Longitud del suavizador estacional. Debe ser un entero impar. El suavizador estacional utiliza datos para una fase en particular de la variación estacional, durante varios períodos. Se utiliza un paso discreto de la dimensión temporal de cada período. El suavizador estacional indica el número de períodos utilizados para suavizar. Por ejemplo, si la dimensión de tiempo está segmentada por mes y el período es Año (12), el componente estacional se calculará de modo que cada mes particular de cada año se calcule a partir de los datos del mismo mes, tanto en ese año como en años adyacentes. El valor de **seasonal_smoother** es el número de años empleados para suavizar.
trend_smoother	Longitud del suavizador de tendencias. Debe ser un entero impar. El suavizador de tendencias utiliza la misma escala de tiempo que el parámetro **period_int** y su valor es el número de gránulos usados para suavizar. Por ejemplo, si una serie de tiempo está segmentada por mes, la tendencia más suave será la cantidad de meses utilizados para suavizar.

La función de gráfico **STL_Seasonal** se utiliza a menudo en combinación con las siguientes funciones:

Funciones relacionadas

Función	Interacción
STL_Trend - función de gráfico (page 1439)	Esta es la función que se utiliza para calcular el componente de tendencia de una serie temporal.
STL_Residual - función de gráfico (page 1443)	Al desglosar una métrica de entrada en componente estacional y de tendencia, parte de la variación de la medida no se ajustará a ninguno de los dos componentes principales. La función **STL_ Residual** calcula esta parte de la descomposición.

Para ver un tutorial con un ejemplo completo que muestra cómo usar esta función, consulte *Tutorial: Descomposición de series de tiempo en Qlik Sense (page 1445)*.

STL_Residual - función de gráfico

STL_Residual es una función de descomposición de series de tiempo. Junto con **STL_Seasonal** y **STL_Trend**, esta función se utiliza para descomponer una serie temporal en componentes estacionales, de tendencia y residuales. En el contexto del algoritmo STL, la descomposición de series de tiempo se utiliza para identificar tanto un patrón estacional recurrente como una tendencia general, dada una métrica de entrada y otros parámetros. Al realizar esta operación, parte de la variación en la métrica de entrada no encajará en el componente estacional ni en el de tendencia, y se definirá como el componente residual. La función de gráfico **STL_Residual** captura esta parte del cálculo.

Las tres funciones STL están relacionadas con la métrica de entrada a través de una simple suma:

STL_Trend + **STL_Seasonal** + **STL_Residual** = Métrica de entrada

STL (descomposición estacional y de tendencia usando Loess) emplea técnicas de suavizado de datos y, a través de sus parámetros de entrada, permite al usuario ajustar la periodicidad de los cálculos que realiza. Esta periodicidad determina cómo se segmenta en el análisis la dimensión temporal de la métrica de entrada (una medida).

Dado que la descomposición de series de tiempo busca principalmente la estacionalidad y variaciones generales en los datos, la información del componente residual se considera la menos significativa de los tres componentes. Sin embargo, un componente residual sesgado o periódico puede ayudar a identificar problemas en el cálculo, como por ejemplo configuraciones de periodicidad incorrectas.

Como mínimo, **STL_Residual** toma una métrica de entrada (`target_measure`) y un valor entero para su `period_int`, devolviendo un valor de punto flotante. La métrica de entrada tendrá la forma de una agregación que varía a lo largo de la dimensión de tiempo. Opcionalmente, puede incluir valores para `seasonal_smoother` y `trend_smoother`, para ajustar el algoritmo de suavizado.

Puede trabajar con esta función insertándola directamente en el editor de expresiones de un gráfico o agregando el modificador de **Descomposición de series de tiempo** a una medida. También está disponible para su uso en Insight Advisor como tipo de análisis.

Sintaxis:

```
STL_Residual(target_measure, period_int [,seasonal_smoother [,trend_
smoother]])
```

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
target_measure	La medida para descomponer en componentes estacionales y de tendencia. Esta debería ser una medida como Sum(Sales) o Sum(Passengers) que varía a lo largo de la dimensión del tiempo. Este no debe ser un valor constante.
period_int	La periodicidad del conjunto de datos. Este parámetro es un valor entero que representa el número de pasos discretos que componen un período, o ciclo estacional, de la señal. Por ejemplo, si la serie temporal está segmentada en una sección para cada trimestre del año, debe establecer el **period_int** en un valor de 4 para definir la periodicidad como Año.
seasonal_smoother	Longitud del suavizador estacional. Debe ser un entero impar. El suavizador estacional utiliza datos para una fase en particular de la variación estacional, durante varios períodos. Se utiliza un paso discreto de la dimensión temporal de cada período. El suavizador estacional indica el número de períodos utilizados para suavizar. Por ejemplo, si la dimensión de tiempo está segmentada por mes y el período es Año (12), el componente estacional se calculará de modo que cada mes particular de cada año se calcule a partir de los datos del mismo mes, tanto en ese año como en años adyacentes. El valor de **seasonal_smoother** es el número de años empleados para suavizar.

Argumento	Descripción
trend_ smoother	Longitud del suavizador de tendencias. Debe ser un entero impar. El suavizador de tendencias utiliza la misma escala de tiempo que el parámetro **period_int** y su valor es el número de gránulos usados para suavizar. Por ejemplo, si una serie de tiempo está segmentada por mes, la tendencia más suave será la cantidad de meses utilizados para suavizar.

La función de gráfico **STL_Residual** se utiliza a menudo en combinación con las siguientes funciones:

Funciones relacionadas

Función	Interacción
STL_Seasonal - función de gráfico (page 1441)	Esta es la función que se utiliza para calcular el componente estacional de una serie temporal.
STL_Trend - función de gráfico (page 1439)	Esta es la función que se utiliza para calcular el componente de tendencia de una serie temporal.

Para ver un tutorial con un ejemplo completo que muestra cómo usar esta función, consulte *Tutorial: Descomposición de series de tiempo en Qlik Sense (page 1445)*.

Tutorial: Descomposición de series de tiempo en Qlik Sense

Este tutorial demuestra el uso de tres funciones de gráfico para descomponer una serie de tiempo usando el algoritmo STL.

Este tutorial utiliza datos de series temporales para la cantidad de pasajeros que utilizan una aerolínea por mes para demostrar la funcionalidad del algoritmo STL. Las funciones de gráfico **STL_Trend**, **STL_Seasonal** y **STL_Residual** se utilizarán para crear las visualizaciones. Para más información sobre la descomposición de series de tiempo en Qlik Sense, vea *Funciones de descomposición de series de tiempo (page 1391)*.

Crear una app

Comience creando una nueva app e importando el conjunto de datos.

Descargue este conjunto de datos:

Tutorial: Descomposición de series de tiempo

Este archivo contiene datos sobre el número de pasajeros de una aerolínea al mes.

Haga lo siguiente:

1. Desde el centro de control, haga clic en **Crear nueva app**.
2. Abra la app y suelte el archivo *Tutorial - Time series decomposition.csv* sobre ella.

Preparar y cargar los datos

Para que Qlik Sense interprete correctamente el campo YearMonth, es posible que deba usar el Gestor de datos para reconocer el campo como un campo de fecha, no como un campo con valores de cadena de texto. Por lo general, este paso se maneja automáticamente, pero en este caso las fechas se presentan en el formato *AAAA-MM*, poco común.

1. En el Gestor de datos, seleccione la tabla y haga clic en .
2. Con el campo *YearMonth* seleccionado, haga clic en y configure el **Tipo de campo** como **Fecha**.
3. En **Formato de entrada**, escriba *AAAA-MM*.
4. En **Mostrar formato**, escriba *AAAA-MM* y haga clic en **Aceptar**.
   El campo ahora debería mostrar el icono de calendario.
5. Haga clic en **Cargar datos**.

Ahora está listo para comenzar a usar las funciones STL para representar visualmente sus datos.

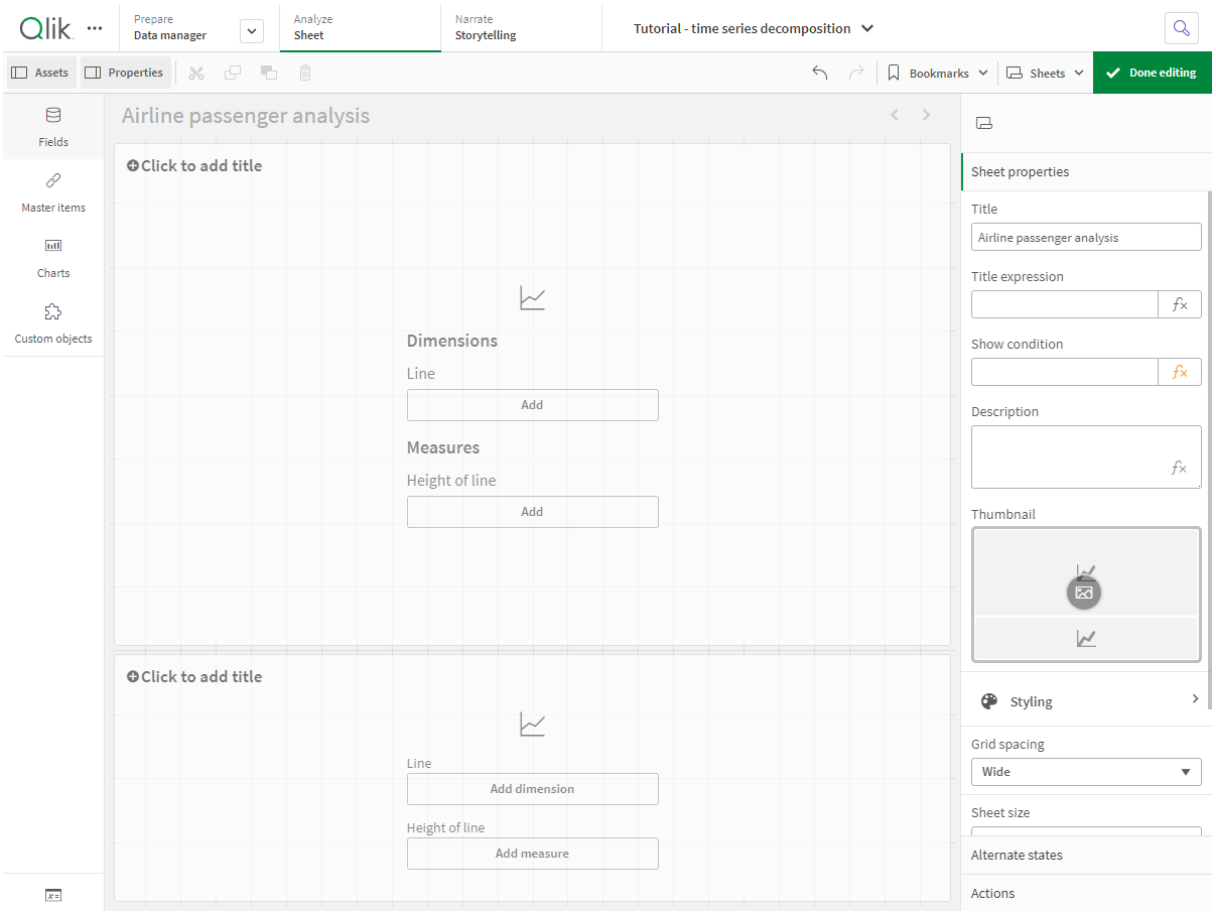
Crear las visualizaciones

A continuación, crearemos dos gráficos de líneas para demostrar la utilidad de las funciones de gráfico **STL_Trend**, **STL_Seasonal** y **STL_Residual**.

Abra una nueva hoja y asígnele un título.

Agregue dos gráficos de líneas a la hoja. Cambie el tamaño y la posición de los gráficos para que coincidan con la siguiente imagen.

Esquema de cuadrícula de Qlik Sense de la hoja de aplicación en blanco



Primer gráfico de líneas: Tendencia y componentes estacionales

Haga lo siguiente:

1. Agregue el título *Estacional y tendencia* al primer gráfico de líneas.
2. Agregue *YearMonth* como dimensión y denomínela *Fecha*.
3. Agregue la siguiente medida y denomínela *Pasajeros al mes*:
   =Sum(Pasajeros)
4. En **Datos**, expanda la medida *Pasajeros al mes* y haga clic en **Añadir línea de tendencia**.
5. Establezca el **Tipo** en **Lineal**.
   Ahora compararemos esta línea de tendencia con la salida suavizada del componente de tendencia.
6. Agregue la siguiente medida para trazar el componente de tendencia y denomínelo *Tendencia*:
   =STL_Trend(SUM(Pasajeros), 12)
7. A continuación, agregue la siguiente medida para trazar el componente estacional y denomínelo *Estacional*:
   =STL_Seasonal(SUM(Pasajeros), 12)
8. En **Aspecto** > **Presentación**, configure **Barra de desplazamiento** en **Ninguna**.
9. Mantenga los colores predeterminados o cámbielos para que se ajusten a sus preferencias.

Segundo gráfico de líneas: Componente residual

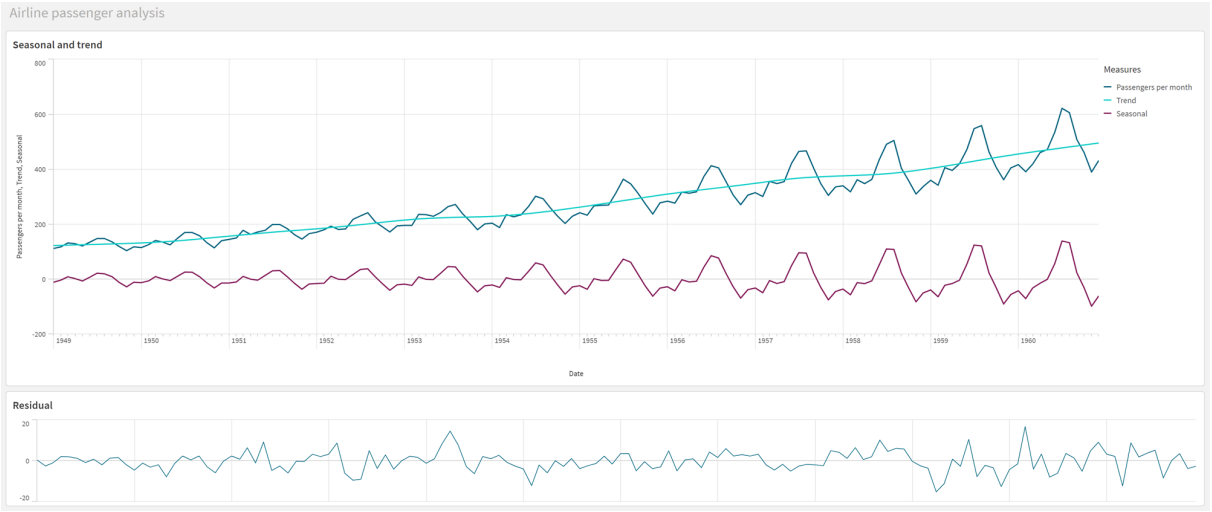
A continuación, configure el segundo gráfico de líneas. Esta visualización mostrará el componente residual de la serie temporal.

Haga lo siguiente:

1. Arrastre un gráfico de líneas a la hoja. Agregue el título *Residual*.
2. Agregue *Fecha* como dimensión.
3. Agregue la siguiente medida y denomínela *Residual*:
   =STL_Residual(SUM(Pasajeros), 12)
4. En **Aspecto** > **Presentación**, configure **Barra de desplazamiento** en **Ninguna**.

Su hoja ahora debería verse de la siguiente manera.

Hoja de Qlik Sense para análisis de pasajeros de líneas aéreas



Interpretar y explicar los datos

Con las funciones de gráfico STL, podemos obtener una serie de conocimientos de nuestros datos sobre series temporales.

Componente de tendencia

La información estadística en el componente de tendencia está desestacionalizada. Esto hace que sea más fácil ver fluctuaciones generales que no se repiten a lo largo del tiempo. En comparación con la línea de tendencia lineal y recta de *Pasajeros al mes*, el componente de tendencia STL captura las tendencias cambiantes. Muestra algunas desviaciones claras sin dejar de presentar la información de forma legible. Los comportamientos de suavizado en el algoritmo STL ayudaron a capturar esto.

Las caídas en el número de pasajeros de aerolíneas que son visibles en el gráfico de tendencia STL pueden explicarse como parte del impacto económico de las recesiones que ocurrieron durante la década de 1950.

Componente estacional

El componente estacional sin tendencia aisló las fluctuaciones recurrentes a lo largo de la serie temporal y eliminó la información de tendencia general de esa parte del análisis. Comenzamos con un conjunto de datos que consiste en agregaciones de año y mes. Con estos datos, queda implícito que estamos segmentando los datos en gránulos de un mes. Definiendo un valor de período de 12, configuramos el gráfico para modelar patrones estacionales en el transcurso de ciclos de un año (doce meses).

En los datos, hay un patrón estacional repetido de aumentos repentinos de pasajeros de aerolíneas en los meses de verano, seguidos de descensos en los meses de invierno. Esto va alineado con la idea de que en verano se suelen tomar vacaciones y viajar. También vemos que en el transcurso de la serie temporal, estos ciclos estacionales aumentan drásticamente en amplitud.

Componente residual

El gráfico para el componente residual muestra toda la información que no fue capturada en la descomposición de tendencia y estacional. El componente residual incluye ruido estadístico, pero también puede indicar una configuración incorrecta de la tendencia STL y los argumentos de la función estacional. Generalmente, si hay oscilaciones periódicas en el componente residual de la señal, o la información que se muestra claramente no es aleatoria, suele ser signo de que hay información en la serie temporal que actualmente no está capturada en los componentes estacionales o de tendencia. En ese caso, debe revisar sus definiciones de cada argumento de función y posiblemente cambiar la periodicidad.

Valores más suaves

Dado que no especificamos ningún valor para los suavizadores de tendencia y estacionales, la función utilizará los valores predeterminados para estos parámetros. En Qlik Sense, los valores predeterminados más suaves en el algoritmo STL producen resultados efectivos. Como resultado, en la mayoría de los casos, estos argumentos pueden quedar fuera de las expresiones.

Establecer los argumentos estacionales o de suavizado de tendencias en 0 en cualquiera de las tres funciones STL hace que el algoritmo use valores predeterminados, en lugar de valores de 0.

El valor de suavizado de tendencia utiliza la dimensión que se especifica en el gráfico. Dado que el campo *YearMonth* presenta los datos por meses, el valor más suave de la tendencia será el número de meses. El suavizador estacional reflejará la periodicidad definida. En este caso, dado que definimos un período de doce meses (un año), el valor estacional más suave es el número de años. Esto puede sonar confuso, pero en realidad significa que para encontrar la estacionalidad, debemos observar varias estaciones. Este número es el suavizador estacional.

Otra información útil

Dado que los ciclos estacionales aumentan en amplitud con el tiempo, un enfoque analítico más avanzado podría utilizar funciones logarítmicas para crear una descomposición multiplicativa. En la práctica, se puede crear una medida simple de amplitud relativa en Qlik Sense dividiendo el componente estacional por el componente de tendencia. Cuando se hace esto, notamos que con el tiempo, los picos de verano de cada ciclo aumentan en amplitud relativa. Sin embargo, la amplitud de los puntos bajos en invierno no aumenta con el tiempo.

8.23 Funciones de distribución estadística

Las funciones de distribución estadística devuelven las probabilidades de aparecer diferentes resultados posibles para una variable de entrada dada. Puede utilizar estas funciones para calcular los valores potenciales de sus puntos de datos.

Los tres grupos de funciones de distribución estadística que se describen a continuación se han implementado todos en Qlik Sense utilizando la biblioteca de funciones Cephes. Para obtener referencias y detalles sobre los algoritmos empleados, su precisión, etc. vea: Cephes library. La biblioteca de funciones Cephes se utiliza con permiso.

- Las funciones de probabilidad calculan la probabilidad en el punto de la distribución dado por el valor suministrado.
    - Las funciones de frecuencia se utilizan para distribuciones discretas.
    - Las funciones de densidad se utilizan para funciones continuas.
- Las funciones Dist calculan la probabilidad acumulada de la distribución en el punto de la distribución dado por el valor suministrado.
- Las funciones Inv calculan el valor inverso, dada la probabilidad acumulada de la distribución.

Todas las funciones pueden utilizarse tanto en el script de carga de datos como en las expresiones de gráficos.

Descripción general de las funciones de distribución estadística

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

BetaDensity

`BetaDensity()` devuelve la probabilidad de la distribución Beta.

```
BetaDensity (value, alpha, beta)
```

BetaDist

`BetaDist()` devuelve la probabilidad acumulada de la distribución Beta.

```
BetaDist (value, alpha, beta)
```

BetaInv

`BetaINV()` devuelve el inverso de la probabilidad acumulada de la distribución Beta.

```
BetaInv (prob, alpha, beta)
```

BinomDist

`BinomDist()` devuelve la probabilidad acumulada de la distribución binomial.

```
BinomDist (value, trials, trial_probability)
```

BinomFrequency

`BinomFrequency()` devuelve la distribución de probabilidad binomial.

```
BinomFrequency (value, trials, trial_probability)
```

BinomInv

`BinomInv()` devuelve el inverso de la probabilidad acumulada de la distribución binomial.

```
BinomInv (prob, trials, trial_probability)
```

ChiDensity

`ChiDensity()` devuelve la probabilidad de una cola de la distribución de chi^2. La función de densidad de chi^2 está asociada con una prueba de chi^2.

```
ChiDensity (value, degrees_freedom)
```

ChiDist

`ChiDist()` devuelve la probabilidad de una cola de la distribución de chi^2. La distribución de chi^2 va asociada con una prueba de chi^2.

```
ChiDist (value, degrees_freedom)
```

ChiInv

`ChiInv()` devuelve el inverso de la probabilidad de una cola de la distribución de chi^2.

```
ChiInv (prob, degrees_freedom)
```

FDensity

`FDensity()` devuelve la probabilidad de la distribución `F`.

```
FDensity (value, degrees_freedom1, degrees_freedom2)
```

FDist

`FDist()` devuelve la probabilidad acumulada de la distribución `F`.

```
FDist (value, degrees_freedom1, degrees_freedom2)
```

FInv

`FInv()` devuelve el inverso de la probabilidad acumulada de la distribución `F`.

```
FInv (prob, degrees_freedom1, degrees_freedom2)
```

GammaDensity

`GammaDensity()` devuelve la probabilidad de la distribución Gamma.

```
GammaDensity (value, k, θ)
```

GammaDist

`GammaDist()` devuelve la probabilidad acumulada de la distribución Gamma.

```
GammaDist (value, k, θ)
```

GammaInv

`GammaInv()` devuelve el inverso de la probabilidad acumulada de la distribución Gamma.

```
GammaInv (prob, k, θ)
```

NormDist

`NormDist()` devuelve la distribución normal acumulativa de la media y la desviación estándar especificadas. Si mean = 0 y standard_dev = 1, la función devuelve la distribución normal estándar.

```
NormDist (value, mean, standard_dev)
```

NormInv

`NormInv()` devuelve el inverso de la distribución acumulativa normal de la media y la desviación estándar especificadas.

```
NormInv (prob, mean, standard_dev)
```

PoissonDist

`PoissonDist()` devuelve la probabilidad acumulada de la distribución de Poisson.

```
PoissonDist (value, mean)
```

PoissonFrequency

`PoissonFrequency()` devuelve la distribución de probabilidad de Poisson.

```
PoissonFrequency (value, mean)
```

PoissonInv

`PoissonInv()` devuelve el inverso de la probabilidad acumulada de la distribución de Poisson.

```
PoissonInv (prob, mean)
```

TDensity

`TDensity()` devuelve el valor de la función de densidad `t` de estudiante, donde un valor numérico es un valor calculado de `t` para el que se calculará la probabilidad.

```
TDensity (value, degrees_freedom, tails)
```

TDist

`TDist()` devuelve la probabilidad de distribución `t` de estudiante, en la que un valor numérico es un valor calculado de `t` para el cual se ha de calcular la probabilidad.

```
TDist (value, degrees_freedom, tails)
```

TInv

`TInv()` devuelve el valor `t` de la distribución `t` de estudiante como una función de la probabilidad y los grados de libertad.

```
TInv (prob, degrees_freedom)
```

Vea también:

Funciones de agregación estadística (page 403)

BetaDensity

`BetaDensity()` devuelve la probabilidad de la distribución Beta.

Sintaxis:

```
BetaDensity(value, alpha, beta)
```

Tipo de datos que devuelve: número

Argumentos

Argumento	Descripción
`value`	Es el valor con el cual se desea evaluar la distribución. El valor debe estar entre 0 y 1.
`alpha`	Un número positivo que define el primer parámetro de forma. Es el exponente de la variable aleatoria
`beta`	Un número positivo que define el segundo parámetro de forma. Indica el número de grados de libertad del denominador.

BetaDist

`BetaDist()` devuelve la probabilidad acumulada de la distribución Beta.

Sintaxis:

```
BetaDist(value, alpha, beta)
```

Tipo de datos que devuelve: número

Argumentos

Argumento	Descripción
`value`	Es el valor con el cual se desea evaluar la distribución. El valor debe estar entre 0 y 1.
`alpha`	Un número positivo que define el primer parámetro de forma. Es el exponente de la variable aleatoria
`beta`	Un número positivo que define el segundo parámetro de forma. Es el exponente que controla la forma de la distribución.

Esta función se relaciona con la función `BetaInv` de la siguiente manera:
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BetaInv

`BetaINV()` devuelve el inverso de la probabilidad acumulada de la distribución Beta.

Sintaxis:

```
BetaInv(prob, alpha, beta)
```

Tipo de datos que devuelve: número

Argumentos

Argumento	Descripción
prob	Una probabilidad asociada a la distribución de probabilidad Beta. Debe ser un número ente 0 y 1.
alpha	Un número positivo que define el primer parámetro de forma. Es el exponente de la variable aleatoria
beta	Un número positivo que define el segundo parámetro de forma. Es el exponente que controla la forma de la distribución.

Esta función se relaciona con la función `BetaDist` de la siguiente manera:
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BinomDist

`BinomDist()` devuelve la probabilidad acumulada de la distribución binomial.

Sintaxis:

```
BinomDist(value, trials, trial_probability)
```

Tipo de datos que devuelve: número

Argumentos

Argumento	Descripción
value	Es el valor con el cual se desea evaluar la distribución. El valor debe ser un número entero no menor que cero y no mayor que el número de pruebas.
trials	Un número entero positivo que indica el número de pruebas.
trial_probability	La probabilidad de éxito de cada prueba. Siempre es un número entre 0 y 1.

Esta función se relaciona con la función `BinomInv` de la siguiente manera:
`If prob = BinomDIST(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value`

BinomFrequency

`BinomFrequency()` devuelve la distribución de probabilidad binomial.

Sintaxis:

```
BinomFrequency(value, trials, trial_probability)
```

Tipo de datos que devuelve: número

Argumentos

Argumento	Descripción
value	Es el valor con el cual se desea evaluar la distribución. El valor debe ser un número entero no menor que cero y no mayor que el número de pruebas.
trials	Un número entero positivo que indica el número de pruebas.
trial_probability	La probabilidad de éxito de cada prueba. Siempre es un número entre 0 y 1.

BinomInv

`BinomInv()` devuelve el inverso de la probabilidad acumulada de la distribución binomial.

Sintaxis:

```
BinomInv(prob, trials, trial_probability)
```

Tipo de datos que devuelve: número

Argumentos

Argumento	Descripción
prob	Una probabilidad asociada a la distribución de probabilidad Beta. Debe ser un número ente 0 y 1.
trials	Un número entero positivo que indica el número de pruebas.
trial_probability	La probabilidad de éxito de cada prueba. Siempre es un número entre 0 y 1.

Esta función se relaciona con la función `BinomDist` de la siguiente manera:
`If prob = BinomDist(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value`

ChiDensity

`ChiDensity()` devuelve la probabilidad de una cola de la distribución de chi^2. La función de densidad de chi^2 está asociada con una prueba de chi^2.

Sintaxis:

```
ChiDensity(value, degrees_freedom)
```

Tipo de datos que devuelve: número

Argumentos

Argumento	Descripción
value	Es el valor con el cual se desea evaluar la distribución. El valor no debe ser negativo.
degrees_freedom	Es un entero positivo que indica el número de grados de libertad del numerador.

ChiDist

`ChiDist()` devuelve la probabilidad de una cola de la distribución de chi^2. La distribución de chi^2 va asociada con una prueba de chi^2.

Sintaxis:

```
CHIDIST(value, degrees_freedom)
```

Tipo de datos que devuelve: número

Argumentos:

Argumentos

Argumento	Descripción
value	Es el valor con el cual se desea evaluar la distribución. El valor no debe ser negativo.
degrees_freedom	Es un entero positivo que indica el número de grados de libertad.

Esta función se relaciona con la función **ChiInv** de la siguiente manera:
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Limitaciones:

Todos los argumentos deben ser numéricos, de lo contrario devolverá NULL.

Ejemplos y resultados:

Ejemplo	Resultado
CHIDIST(8, 15)	Devuelve 0,9238

ChiInv

`ChiInv()` devuelve el inverso de la probabilidad de una cola de la distribución de chi^2.

Sintaxis:

```
CHIINV(prob, degrees_freedom)
```

Tipo de datos que devuelve: número

Argumentos:

Argumentos

Argumento	Descripción
prob	Una probabilidad asociada con la distribución chi^2. Debe ser un número ente 0 y 1.
degrees_freedom	Es un entero que indica el número de grados de libertad.

Esta función se relaciona con la función **ChiDist** de la siguiente manera:
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Limitaciones:

Todos los argumentos deben ser numéricos, de lo contrario devolverá NULL.

Ejemplos y resultados:

Ejemplo	Resultado
CHIINV(0.9237827, 15)	Devuelve 8,0000

FDensity

`FDensity()` devuelve la probabilidad de la distribución `F`.

Sintaxis:

```
FDensity(value, degrees_freedom1, degrees_freedom2)
```

Tipo de datos que devuelve: número

Argumentos

Argumento	Descripción
`value`	Es el valor con el cual se desea evaluar la distribución. El valor no debe ser negativo.
`degrees_freedom1`	Es un entero positivo que indica el número de grados de libertad del numerador.
`degrees_freedom2`	Es un entero positivo que indica el número de grados de libertad del denominador.

FDist

`FDist()` devuelve la probabilidad acumulada de la distribución `F`.

Sintaxis:

```
FDist(value, degrees_freedom1, degrees_freedom2)
```

Tipo de datos que devuelve: número

Argumentos:

Argumentos

Argumento	Descripción
value	Es el valor con el cual se desea evaluar la distribución. El valor no debe ser negativo.
degrees_freedom1	Es un entero positivo que indica el número de grados de libertad del numerador.
degrees_freedom2	Es un entero positivo que indica el número de grados de libertad del denominador.

Esta función se relaciona con la función **FInv** de la siguiente manera:
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Limitaciones:

Todos los argumentos deben ser numéricos, de lo contrario devolverá NULL.

Ejemplos y resultados:

Ejemplo	Resultado
FDIST(15, 8, 6)	Devuelve 0,0019

FInv

`FInv()` devuelve el inverso de la probabilidad acumulada de la distribución `F`.

Sintaxis:

```
FInv(prob, degrees_freedom1, degrees_freedom2)
```

Tipo de datos que devuelve: número

Argumentos:

Argumentos

Argumento	Descripción
prob	Una probabilidad asociada con la distribución de probabilidad F y debe ser un número entre 0 y 1.
degrees_ freedom	Es un entero que indica el número de grados de libertad.

Esta función se relaciona con la función **FDist** de la siguiente manera:
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Limitaciones:

Todos los argumentos deben ser numéricos, de lo contrario devolverá NULL.

Ejemplos y resultados:

Ejemplo	Resultado
FINV(0.0019369, 8, 6)	Devuelve 15,0000

GammaDensity

`GammaDensity()` devuelve la probabilidad de la distribución Gamma.

Sintaxis:

```
GammaDensity(value, k, θ)
```

Tipo de datos que devuelve: número

Argumentos

Argumento	Descripción
value	Es el valor con el cual se desea evaluar la distribución. El valor no debe ser negativo.
k	Un número positivo que define el parámetro de forma.
θ	Un número positivo que define el parámetro de escala.

GammaDist

`GammaDist()` devuelve la probabilidad acumulada de la distribución Gamma.

Sintaxis:

```
GammaDist(value, k, θ)
```

Tipo de datos que devuelve: número

Argumentos

Argumento	Descripción
value	Es el valor con el cual se desea evaluar la distribución. El valor no debe ser negativo.
k	Un número positivo que define el parámetro de forma.
θ	Un número positivo que define el parámetro de escala.

Esta función se relaciona con la función `GammaINV` de la siguiente manera:
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

GammaInv

`GammaInv()` devuelve el inverso de la probabilidad acumulada de la distribución Gamma.

Sintaxis:

```
GammaInv(prob, k, θ)
```

Tipo de datos que devuelve: número

Argumentos

Argumento	Descripción
prob	Una probabilidad asociada a la distribución de probabilidad Gamma. Debe ser un número ente 0 y 1.
k	Un número positivo que define el parámetro de forma.
θ	Un número positivo que define el parámetro de escala.

Esta función se relaciona con la función `GammaDist` de la siguiente manera:
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

NormDist

`NormDist()` devuelve la distribución normal acumulativa de la media y la desviación estándar especificadas. Si mean = 0 y standard_dev = 1, la función devuelve la distribución normal estándar.

Sintaxis:

```
NORMDIST(value, [mean], [standard_dev], [cumulative])
```

Tipo de datos que devuelve: número

Argumentos:

Argumentos

Argumento	Descripción
value	Es el valor con el cual se desea evaluar la distribución.
mean	Un valor opcional que indica la media aritmética de la distribución. Si no indica este argumento, el valor predeterminado es 0.
standard_dev	Un valor positivo opcional que muestra la desviación estándar de la distribución. Si no indica este argumento, el valor predeterminado es 1.

Argumento	Descripción
cumulative	De manera opcional, puede elegir entre usar una distribución normal estándar o una distribución acumulativa. 0 = distribución normal estándar 1 = distribución acumulativa (opción predeterminada)

Esta función se relaciona con la función **NormInv** de la siguiente manera:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Limitaciones:

Todos los argumentos deben ser numéricos, de lo contrario devolverá NULL.

Ejemplos y resultados:

Ejemplo	Resultado
NORMDIST(0.5, 0, 1)	Devuelve 0,6915

NormInv

`NormInv()` devuelve el inverso de la distribución acumulativa normal de la media y la desviación estándar especificadas.

Sintaxis:

```
NORMINV(prob, mean, standard_dev)
```

Tipo de datos que devuelve: número

Argumentos:

Argumentos

Argumento	Descripción
prob	Es una probabilidad asociada con la distribución normal. Debe ser un número ente 0 y 1.
mean	Es un valor que indica la media aritmética para la distribución.
standard_dev	Es un valor positivo que muestra la desviación estándar de la distribución.

Esta función se relaciona con la función **NormDist** de la siguiente manera:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Limitaciones:

Todos los argumentos deben ser numéricos, de lo contrario devolverá NULL.

Ejemplos y resultados:

Ejemplo	Resultado
NORMINV(0.6914625, 0, 1)	Devuelve 0,5000

PoissonDist

`PoissonDist()` devuelve la probabilidad acumulada de la distribución de Poisson.

Sintaxis:

```
PoissonDist(value, mean)
```

Tipo de datos que devuelve: número

Argumentos

Argumento	Descripción
`value`	Es el valor con el cual se desea evaluar la distribución. El valor no debe ser negativo.
`mean`	Un número positivo que define el resultado promedio.

Esta función se relaciona con la función `PoissonInv` de la siguiente manera:
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

PoissonFrequency

`PoissonFrequency()` devuelve la distribución de probabilidad de Poisson.

Sintaxis:

```
PoissonFrequency(value, mean)
```

Tipo de datos que devuelve: número

Argumentos

Argumento	Descripción
`value`	Es el valor con el cual se desea evaluar la distribución. El valor no debe ser negativo.
`mean`	Un número positivo que define el resultado promedio.

PoissonInv

`PoissonInv()` devuelve el inverso de la probabilidad acumulada de la distribución de Poisson.

Sintaxis:

```
PoissonInv(prob, mean)
```

Tipo de datos que devuelve: número

Argumentos

Argumento	Descripción
prob	Una probabilidad asociada a la distribución de probabilidad de Poisson. Debe ser un número ente 0 y 1.
mean	Un número positivo que define el resultado promedio.

Esta función se relaciona con la función PoissonDIST de la siguiente manera:
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

TDensity

`TDensity()` devuelve el valor de la función de densidad `t` de estudiante, donde un valor numérico es un valor calculado de `t` para el que se calculará la probabilidad.

Sintaxis:

```
TDensity(value, degrees_freedom)
```

Tipo de datos que devuelve: número

Argumentos

Argumento	Descripción
value	Es el valor con el cual se desea evaluar la distribución. El valor no debe ser negativo.
degrees_freedom	Es un entero positivo que indica el número de grados de libertad.

TDist

`TDist()` devuelve la probabilidad de distribución `t` de estudiante, en la que un valor numérico es un valor calculado de `t` para el cual se ha de calcular la probabilidad.

Sintaxis:

```
TDist(value, degrees_freedom, tails)
```

Tipo de datos que devuelve: número

Argumentos:

Argumentos

Argumento	Descripción
value	Es el valor con el cual se desea evaluar la distribución. El valor no debe ser negativo.
degrees_freedom	Es un entero positivo que indica el número de grados de libertad.
tails	Debe ser o bien 1 (distribución de una cola) o 2 (distribución de dos colas).

Esta función se relaciona con la función **TInv** de la siguiente manera:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value`

Limitaciones:

Todos los argumentos deben ser numéricos, de lo contrario devolverá NULL.

Ejemplos y resultados:

Ejemplo	Resultado
TDIST(1, 30, 2)	Devuelve 0,3253

TInv

`TInv()` devuelve el valor `t` de la distribución `t` de estudiante como una función de la probabilidad y los grados de libertad.

Sintaxis:

```
TINV(prob, degrees_freedom)
```

Tipo de datos que devuelve: número

Argumentos:

Argumentos

Argumento	Descripción
prob	Es una probabilidad de dos colas asociada con la distribución t. Debe ser un número ente 0 y 1.
degrees_ freedom	Es un entero que indica el número de grados de libertad.

Limitaciones:

Todos los argumentos deben ser numéricos, de lo contrario devolverá NULL.

Esta función se relaciona con la función **TDist** de la siguiente manera:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value`.

Ejemplos y resultados:

Ejemplo	Resultado
TINV(0.3253086, 30)	Devuelve 1,0000

8.24 Funciones de cadena

En esta sección se describen funciones para la gestión y manipulación de cadenas.

Todas las funciones pueden utilizarse tanto en el script de carga de datos como en las expresiones de gráficos, excepto **Evaluate** que solo puede utilizarse en el script de carga de datos.

Descripción general de las funciones de cadena

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Capitalize
Capitalize() devuelve la cadena con todas las palabras con su letra inicial en mayúscula.

```
Capitalize (text)
```

Chr
Chr() devuelve el carácter Unicode correspondiente al número entero introducido.

```
Chr (int)
```

Evaluate
Evaluate() halla si la cadena de texto introducida puede ser evaluada como una expresión Qlik Sense válida, y, si es así, devuelve el valor de la expresión en forma de cadena. Si la cadena de entrada no es una expresión válida, devuelve NULL.

```
Evaluate (expression_text)
```

FindOneOf
FindOneOf() busca una cadena a fin de hallar la posición de cualquier carácter de un conjunto de caracteres suministrados. Devuelve la posición de la primera vez que aparece cualquier carácter del conjunto especificado en la búsqueda, a menos que se suministre un tercer argumento (con un valor mayor que 1). Si no encuentra ningún resultado, devuelve **0**.

```
FindOneOf (text, char_set[, count])
```

Hash128
Hash128() devuelve un código hash de 128 bits de los valores de entrada combinados de la expresión. El resultado es una cadena de 22 caracteres.

```
Hash128 (expr{, expression})
```

Hash160
Hash160() devuelve un código hash de 160 bits de los valores de entrada combinados de la expresión. El resultado es una cadena de 27 caracteres.

```
Hash160 (expr{, expression})
```

Hash256
Hash256() devuelve un código hash de 256 bits de los valores de entrada combinados de la expresión. El resultado es una cadena de 43 caracteres.

```
Hash256 (expr{, expression})
```

Index

Index() busca una cadena para hallar la posición de inicio de la enésima vez que aparece una subcadena suministrada. Un tercer argumento opcional proporciona el valor de n, el cual se considera 1 si se omite. Un valor negativo busca desde el final de la cadena. Las posiciones en la cadena están numeradas de **1** en adelante.

```
Index  (text, substring[, count])
```

IsJson

IsJson() comprueba si una cadena especificada contiene datos JSON (JavaScript Object Notation) válidos. También puede validar un tipo de datos JSON específico.

```
IsJson (json [, type])
```

JsonGet

JsonGet() devuelve la ruta de una cadena de datos JSON (JavaScript Object Notation). Los datos deben ser datos JSON válidos pero pueden contener espacios extra o retornos.

```
JsonGet (json, path)
```

JsonSet

JsonSet() modifica una cadena especificada que contiene datos JSON (JavaScript Object Notation) válidos. Puede establecer o insertar un valor JSON con la nueva ubicación especificada por la ruta. Los datos deben ser datos JSON válidos pero pueden contener espacios extra o retornos.

```
JsonSet(json, path, value)
```

KeepChar

KeepChar() devuelve una cadena que consiste en la primera cadena, 'text', menos cualquiera de los caracteres NO contenidos en la segunda cadena, "keep_chars".

```
KeepChar (text, keep_chars)
```

Left

Left() devuelve una cadena que consiste en los primeros caracteres (situados más a la izquierda) de la cadena introducida, donde el número de caracteres viene determinado por el segundo argumento.

```
Left (text, count)
```

Len

Len() devuelve la longitud de la cadena introducida.

```
Len (text)
```

LevenshteinDist

LevenshteinDist() devuelve la distancia Levenshtein entre dos cadenas. Se define como el número mínimo de ediciones de un solo carácter (inserciones, eliminaciones o sustituciones) necesarias para cambiar una cadena por otra. La función es útil para comparaciones de cadenas difusas.

```
LevenshteinDist (text1, text2)
```

Lower

Lower() pone todos los caracteres de la cadena introducida en letra minúscula.

```
Lower (text)
```

LTrim

LTrim() devuelve la cadena introducida sin los espacios iniciales.

```
LTrim (text)
```

Mid

Mid() devuelve la parte de la cadena de entrada que comienza en la posición del carácter definido por el segundo argumento, 'start', y devolviendo el número de caracteres definidos por el tercer argumento, 'count'. Si 'count' se omite, devuelve el resto de la cadena de entrada. El primer carácter de la cadena de entrada se enumera como 1.

```
Mid (text, start[, count])
```

Ord

Ord() devuelve el número Unicode de punto de código del primer carácter de la cadena de entrada.

```
Ord (text)
```

PurgeChar

PurgeChar() devuelve una cadena que contiene los caracteres de la cadena introducida ('text'), menos los caracteres que aparecen en el segundo argumento ('remove_chars').

```
PurgeChar (text, remove_chars)
```

Repeat

Repeat() forma una cadena que consiste en la cadena introducida, repetida el número de veces definido por el segundo argumento.

```
Repeat (text[, repeat_count])
```

Replace

Replace() devuelve una cadena tras haber reemplazado todas las veces en que aparece una determinada subcadena dentro de la cadena introducida por otra subcadena. La función no es recursiva y funciona de izquierda a derecha.

```
Replace (text, from_str, to_str)
```

Right

Right() devuelve una cadena formada por los últimos caracteres (los situados más a la derecha) de la cadena de entrada, donde el número de caracteres viene determinado por el segundo argumento.

```
Right (text, count)
```

RTrim

RTrim() devuelve la cadena introducida libre de espacios finales.

```
RTrim (text)
```

SubField

SubField() se utiliza para extraer componentes de subcadenas de un campo de cadena principal, donde los campos de registro originales constan de dos o más partes separadas por un delimitador.

```
SubField (text, delimiter[, field_no ])
```

SubStringCount

SubStringCount() devuelve el número de veces que aparece la subcadena especificada en el texto de la cadena de entrada. Si no existe coincidencia alguna, devuelve 0.

```
SubStringCount (text, substring)
```

TextBetween

TextBetween() devuelve el texto de la cadena de entrada que se da entre los caracteres especificados como delimitadores.

```
TextBetween (text, delimiter1, delimiter2[, n])
```

Trim

Trim() devuelve la cadena introducida libre de todos los espacios iniciales y finales.

```
Trim (text)
```

Upper

Upper() convierte todos los caracteres de la cadena introducida en mayúscula para todos los caracteres de texto de la expresión. Los números y símbolos se ignoran.

```
Upper (text)
```

Capitalize

Capitalize() devuelve la cadena con todas las palabras con su letra inicial en mayúscula.

Sintaxis:

```
Capitalize(text)
```

Tipo de datos que devuelve: cadena

Ejemplo: Expresiones de gráfico

Ejemplo	Resultado
`Capitalize ( 'star trek' )`	Devuelve 'Star Trek'
`Capitalize ( 'AA bb cC Dd')`	Devuelve 'Aa Bb Cc Dd'

Ejemplo: Script de carga

```
Load  String, Capitalize(String) Inline [String rHode iSland washingTon d.C. new york];
```

Resultado

Cadena	Capitalize(String)
rHode iSland	Rhode Island
washingTon d.C.	Washington D.C.
new york	New York

Chr

Chr() devuelve el carácter Unicode correspondiente al número entero introducido.

Sintaxis:

```
Chr(int)
```

Tipo de datos que devuelve: cadena

Ejemplos y resultados:

Ejemplo	Resultado
Chr(65)	Devuelve la cadena 'A'
Chr(163)	Devuelve la cadena '£'
Chr(35)	Devuelve la cadena '#'

Evaluate

Evaluate() halla si la cadena de texto introducida puede ser evaluada como una expresión Qlik Sense válida, y, si es así, devuelve el valor de la expresión en forma de cadena. Si la cadena de entrada no es una expresión válida, devuelve NULL.

Sintaxis:

```
Evaluate(expression_text)
```

Tipo de datos que devuelve: dual

Esta función de script no puede utilizarse en expresiones de gráfico.

Ejemplos y resultados:

Ejemplo de función	Resultado
`Evaluate ( 5 * 8 )`	Devuelve '40'

Ejemplo de script de carga

```
Load  Evaluate(String) as Evaluated, String Inline [String 4 5+3 0123456789012345678 Today()
];
```

Resultado

Cadena	Evaluado
4	4
5+3	8
0123456789012345678	0123456789012345678
Today()	2022-02-02

FindOneOf

FindOneOf() busca una cadena a fin de hallar la posición de cualquier carácter de un conjunto de caracteres suministrados. Devuelve la posición de la primera vez que aparece cualquier carácter del conjunto especificado en la búsqueda, a menos que se suministre un tercer argumento (con un valor mayor que 1). Si no encuentra ningún resultado, devuelve **0**.

Sintaxis:

```
FindOneOf(text, char_set[, count])
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
text	La cadena original.
char_set	Un conjunto de caracteres para buscar en text.
count	Define qué ocurrencia de carácter buscar. Por ejemplo, un valor de 2 busca la segunda vez que aparece un carácter.

Ejemplo: Expresiones de gráfico

Ejemplo	Resultado
`FindOneOf( 'my example text string', 'et%s')`	Devuelve "4" porque "e" es el cuarto carácter en la cadena de ejemplo.
`FindOneOf( 'my example text string', 'et%s', 3)`	Devuelve "12" porque la búsqueda es de cualquiera de los caracteres e, t, % o s y "t" es la tercera aparición en la posición 12 de la cadena de ejemplo.

Ejemplo	Resultado
`FindOneOf( 'my example text string', '¤%&')`	Devuelve "0" porque ninguno de los caracteres ¤, % o & existe en la cadena de ejemplo.

Ejemplo: Script de carga

```
Load * Inline [SearchFor, Occurrence et%s,1 et%s,3 ¤%&,1]
```

Resultado

SearchFor	Occurrence	FindOneOf("mi cadena de texto de ejemplo", SearchFor, Occurrence)
et%s	1	4
et%s	3	12
¤%&	1	0

Hash128

Hash128() devuelve un código hash de 128 bits de los valores de entrada combinados de la expresión. El resultado es una cadena de 22 caracteres.

Sintaxis:

```
Hash128(expr{, expression})
```

Tipo de datos que devuelve: cadena

Ejemplo: Expresiones de gráfico

Ejemplo	Resultado
Hash128 ('abc', 'xyz', '123')	Devuelve "MA&5]6+3=:>:>G%S<U*S2+".
Hash128 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Devuelve "G7*=6GKPJ(Z+)^KM?<$'A+".

Ejemplo: Script de carga

```
Hash_128: Load *,  Hash128(Region, Year, Month) as Hash128; Load * inline [ Region, Year, Month abc, xyz, 123 EU, 2022, 01 UK, 2022, 02 US, 2022, 02 ];
```

Resultado

Región	Año	Mes	Hash128
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2+
EU	2022	01	B40^K&[T@!;VB'XR]<5=/$

Región	Año	Mes	Hash128
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!
US	2022	02	C6@#]4#_G-(]J7EQY#KRW0

Hash160

Hash160() devuelve un código hash de 160 bits de los valores de entrada combinados de la expresión. El resultado es una cadena de 27 caracteres.

Sintaxis:

```
Hash160(expr{, expression})
```

Tipo de datos que devuelve: cadena

Ejemplo: Expresiones de gráfico

Ejemplo	Resultado
Hash160 ('abc', 'xyz', '123')	Devuelve "MA&5]6+3=:>;>G%S<U*S2I:`=X*".
Hash160 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Devuelve "G7*=6GKPJ (Z+)^KM?<$'AI.)?U$".

Ejemplo: Script de carga

```
Hash_160: Load *,  Hash160(Region, Year, Month) as Hash160; Load * inline [ Region, Year, Month abc, xyz, 123 EU, 2022, 01 UK, 2022, 02 US, 2022, 02 ];
```

Resultado

Región	Año	Mes	Hash160
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

Hash256

Hash256() devuelve un código hash de 256 bits de los valores de entrada combinados de la expresión. El resultado es una cadena de 43 caracteres.

Sintaxis:

```
Hash256(expr{, expression})
```

Tipo de datos que devuelve: cadena

Ejemplo: Expresiones de gráfico

Ejemplo	Resultado
Hash256 ('abc', 'xyz', '123')	Devuelve "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.	Devuelve "G7*=6GKPJ(Z+)^KM?<$'AI.)?U$#X2RB [:0ZP=+Z`F:".

Ejemplo: Script de carga

```
Hash_256: Load *,  Hash256(Region, Year, Month) as Hash256; Load * inline [ Region, Year, Month abc, xyz, 123 EU, 2022, 01 UK, 2022, 02 US, 2022, 02 ];
```

Resultado

Región	Año	Mes	Hash256
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

Index

Index() busca una cadena para hallar la posición de inicio de la enésima vez que aparece una subcadena suministrada. Un tercer argumento opcional proporciona el valor de n, el cual se considera 1 si se omite. Un valor negativo busca desde el final de la cadena. Las posiciones en la cadena están numeradas de **1** en adelante.

Sintaxis:

```
Index(text, substring[, count])
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumentos

Argumento	Descripción
text	La cadena original.
substring	Una cadena de caracteres para buscar en text.
count	Define qué instancia del carácter **substring** buscar. Por ejemplo, un valor de 2 busca la segunda vez que aparece un carácter.

Ejemplos y resultados:

Ejemplo	Resultado
Index('abcdefg', 'cd')	Devuelve 3
Index('abcdabcd', 'b', 2)	Devuelve 6 (la segunda vez que aparece 'b')
Index('abcdabcd', 'b',-2)	Devuelve 2 (la segunda vez que aparece 'b' empezando por el final)
`Left( Date, Index( Date,'-') -1 )` where **Date** = 1997-07-14	Devuelve 1997
`Mid( Date, Index( Date, '-', 2 ) -2, 2 )` where **Date** = 1997-07-14	Devuelve 07

Ejemplo: Script

```
T1: 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 ];
```

IsJson

IsJson() comprueba si una cadena especificada contiene datos JSON (JavaScript Object Notation) válidos. También puede validar un tipo de datos JSON específico.

Sintaxis:

```
value IsJson(json [, type])
```

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
json	Cadena para pruebas. Puede contener espacios adicionales o saltos de línea.
type	Argumento opcional que especifica el tipo de datos JSON que se han de probar. • "value" (predeterminado) • "object" • "array" • "string" • "number" • "Boolean" • "null"

Ejemplo: JSON y tipo válidos

Ejemplo	Resultado
`IsJson('null')`	Devuelve -1 (true)
`IsJson('"abc"', 'value')`	Devuelve -1 (true)
`IsJson('"abc"', 'string')`	Devuelve -1 (true)
`IsJson(123, 'number')`	Devuelve -1 (true)

Ejemplo: JSON y tipo no válidos

Ejemplo	Resultado	Descripción
`IsJson('text')`	Devuelve 0 (false)	'text' no es un valor JSON válido
`IsJson('"text"', 'number')`	Devuelve 0 (false)	'"text"' no es un número JSON válido
`IsJson('"text"', 'text')`	Devuelve 0 (false)	'text' no es un tipo JSON válido

JsonGet

JsonGet() devuelve la ruta de una cadena de datos JSON (JavaScript Object Notation). Los datos deben ser datos JSON válidos pero pueden contener espacios extra o retornos.

Sintaxis:

```
value JsonGet(json, path)
```

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
json	Cadena que contiene datos JSON.
path	La ruta debe especificarse conforme a RFC 6901. Esto permitirá la búsqueda de propiedades dentro de los datos JSON, sin utilizar complejas funciones de subcadena o de índice.

Ejemplo: JSON y ruta válidos

Ejemplo	Resultado
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '')`	Devuelve '{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a')`	Devuelve '{"foo":"bar"}'

Ejemplo	Resultado
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a/foo')`	Devuelve '"bar"'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b')`	Devuelve '[123,"abc","ABC"]'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/0')`	Devuelve '123'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/1')`	Devuelve '"abc"'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/2')`	Devuelve '"ABC"'

Ejemplo: JSON y ruta no válidos

Ejemplo	Resultado	Descripción
`JsonGet('{"a":"b"}','/b')`	Devuelve null	La ruta no apunta a una parte válida de los datos JSON.
`JsonGet('{"a"}','/a')`	Devuelve null	Los datos JSON no son JSON válidos (el miembro "a" no tiene un valor).

JsonSet

JsonSet() modifica una cadena especificada que contiene datos JSON (JavaScript Object Notation) válidos. Puede establecer o insertar un valor JSON con la nueva ubicación especificada por la ruta. Los datos deben ser datos JSON válidos pero pueden contener espacios extra o retornos.

Sintaxis:

```
value JsonSet(json, path, value)
```

Tipo de datos que devuelve: dual

Argumentos

Argumento	Descripción
json	Cadena que contiene datos JSON.
path	La ruta debe especificarse conforme a RFC 6901. Esto permitirá construir propiedades dentro de los datos JSON sin utilizar complejas funciones de subcadena o de índice.
value	El nuevo valor de la cadena en formato JSON.

Ejemplo: JSON, ruta y valor válidos

Ejemplo	Resultado
`JsonSet('{}','/a','"b"')`	Devuelve '{"a":"b"}'
`JsonSet('[]','/0','"x"')`	Devuelve '["x"]'
`JsonSet('"abc"','','123')`	Devuelve 123

Ejemplo: JSON, ruta o valor no válidos

Ejemplo	Resultado	Descripción
`JsonSet('"abc"','/x','123')`	Devuelve null	La ruta no apunta a una parte válida de los datos JSON.
`JsonSet('{"a": {"b":"c"}}','a/b','"x"')`	Devuelve null	La ruta no es válida.
`JsonSet('{"a":"b"}','/a','abc')`	Devuelve null	El valor no es un JSON válido. Una cadena debe ir entre comillas.

KeepChar

KeepChar() devuelve una cadena que consiste en la primera cadena, 'text', menos cualquiera de los caracteres NO contenidos en la segunda cadena, "keep_chars".

Sintaxis:

```
KeepChar(text, keep_chars)
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
text	La cadena original.
keep_chars	Una cadena que contiene los caracteres text que deben conservarse.

Ejemplo: Expresiones de gráfico

Ejemplo	Resultado
`KeepChar ( 'a1b2c3','123' )`	Devuelve '123'.
`KeepChar ( 'a1b2c3','1234' )`	Devuelve '123'.
`KeepChar ( 'a1b22c3','1234' )`	Devuelve '1223'.
`KeepChar ( 'a1b2c3','312' )`	Devuelve '123'.

Ejemplo: Script de carga

```
T1:
Load
*,
keepchar(String1, String2) as KeepChar;
Load * inline [
String1, String2
'a1b2c3', '123'
];
```

Resultados

Tabla de Qlik Sense que muestra el resultado de usar la función *KeepChar* en el script de carga.

String1	String2	KeepChar
a1b2c3	123	123

Vea también:

PurgeChar (page 1484)

Left

Left() devuelve una cadena que consiste en los primeros caracteres (situados más a la izquierda) de la cadena introducida, donde el número de caracteres viene determinado por el segundo argumento.

Sintaxis:

```
Left(text, count)
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumento	Descripción
text	La cadena original.
count	Define el número de caracteres que se incluirán en la parte izquierda de la cadena **text**.

Ejemplo: Expresión de gráfico

Ejemplo	Resultado
`Left('abcdef', 3)`	Devuelve 'abc'

Ejemplo: Script de carga

```
T1: Load  *,  left(Text,Start) as Left;        Load * inline [ Text, Start 'abcdef', 3 '2021-07-14', 4 '2021-07-14', 2 ];
```

Resultado

Tabla de Qlik Sense que muestra el resultado de usar la función *Left* en el script de carga.

Texto	Inicio	Left
abcdef	3	abc
2021-07-14	4	2021
2021-07-14	2	20

Vea también *Index (page 1473)*, que permite un análisis de cadenas más complejo.

Len

Len() devuelve la longitud de la cadena introducida.

Sintaxis:

```
Len(text)
```

Tipo de datos que devuelve: entero

Ejemplo: Expresión de gráfico

Ejemplo	Resultado
`Len('Peter')`	Devuelve '5'

Ejemplo: Script de carga

```
T1: Load String, First&Second as NewString; Load *, mid(String,len(First)+1) as Second; Load *, upper(left(String,1)) as First; Load  * inline [ String this is a sample text string capitalize first letter only ];
```

Resultado

Cadena	NuevaCadena
this is a sample text string	This is a sample text string
capitalize first letter only	Capitalize first letter only

LevenshteinDist

LevenshteinDist() devuelve la distancia Levenshtein entre dos cadenas. Se define como el número mínimo de ediciones de un solo carácter (inserciones, eliminaciones o sustituciones) necesarias para cambiar una cadena por otra. La función es útil para comparaciones de cadenas difusas.

Sintaxis:

```
LevenshteinDist(text1, text2)
```

Tipo de datos que devuelve: entero

Ejemplo: Expresión de gráfico

Ejemplo	Resultado
`LevenshteinDist('Kitten','Sitting')`	Devuelve "3"

Ejemplo: Script de carga

Script de carga

```
T1: Load *, recno() as ID; Load 'Silver' as String_1,* inline [ String_2 Sliver SSiver SSiveer
];  T1: Load *, recno()+3 as ID; Load 'Gold' as String_1,* inline [ String_2 Bold Bool Bond ];
 T1: Load *, recno()+6 as ID; Load 'Ove' as String_1,* inline [ String_2 Ove Uve Üve ];  T1:
Load *, recno()+9 as ID; Load 'ABC' as String_1,* inline [ String_2 DEFG abc ビビビ ];  set
nullinterpret = '<NULL>'; T1: 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 T1;  Drop table T1;
```

Resultado

ID	Cadenatexto_1	Cadenatexto_2	LevenshteinDistance
1	Silver	Sliver	2
2	Silver	SSiver	2
3	Silver	SSiveer	3
4	Gold	Bold	1
5	Gold	Bool	3
6	Gold	Bond	2
7	Ove	Ove	0
8	Ove	Uve	1
9	Ove	Üve	1
10	ABC	DEFG	4
11	ABC	abc	3
12	ABC	ビビビ	3
13	X		1
14	X	-	1
15	X	1	1

Lower

Lower() pone todos los caracteres de la cadena introducida en letra minúscula.

Sintaxis:

```
Lower(text)
```

Tipo de datos que devuelve: cadena

Ejemplo: Expresión de gráfico

Ejemplo	Resultado
`Lower('abcD')`	Devuelve 'abcd'

Ejemplo: Script de carga

```
Load  String, Lower(String) Inline [String rHode iSland washingTon d.C. new york];
```

Resultado

Cadena	Lower(String)
rHode iSland	rhode island
washingTon d.C.	washington d.c.
new york	new york

LTrim

LTrim() devuelve la cadena introducida sin los espacios iniciales.

Sintaxis:

```
LTrim(text)
```

Tipo de datos que devuelve: cadena

Ejemplo: Expresiones de gráfico

Ejemplo	Resultado
`LTrim( ' abc' )`	Devuelve 'abc'
`LTrim( 'abc ' )`	Devuelve 'abc '

Ejemplo: Script de carga

```
Set verbatim=1;  T1:  Load *, len(LtrimString) as LtrimStringLength;       Load *, ltrim
(String) as LtrimString; Load *, len(String) as StringLength;                Load * Inline [
String '   abc    ' ' def  '];
```

La instrucción "Set verbatim=1" se incluye en el ejemplo para garantizar que los espacios no se recorten automáticamente antes de la demostración de la función ltrim. Vea Verbatim (page 210) si desea más información.

Resultado

Cadena	StringLength	LtrimStringLength
def	6	5
abc	10	7

Vea también:

RTrim (page 1487)

Mid

Mid() devuelve la parte de la cadena de entrada que comienza en la posición del carácter definido por el segundo argumento, 'start', y devolviendo el número de caracteres definidos por el tercer argumento, 'count'. Si 'count' se omite, devuelve el resto de la cadena de entrada. El primer carácter de la cadena de entrada se enumera como 1.

Sintaxis:

```
Mid(text, start[, count])
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
text	La cadena original.
start	Un número entero que define la posición del primer carácter que se ha de incluir en text.
count	Define la longitud de cadena de la cadena de salida. Si se omite, se incluyen todos los caracteres de la posición definida por **start**.

Ejemplo: Expresiones de gráfico

Ejemplo	Resultado
`Mid('abcdef',3 )`	Devuelve 'cdef'
`Mid('abcdef',3, 2 )`	Devuelve 'cd'

Ejemplo: Script de carga

```
T1: Load *,  mid(Text,Start) as Mid1,            mid(Text,Start,Count) as Mid2;     Load *
inline [ Text, Start, Count 'abcdef', 3, 2 'abcdef', 2, 3 '210714', 3, 2 '210714', 2, 3 ];
```

Resultado

Tabla de Qlik Sense que muestra el resultado de usar la función *Mid* en el script de carga.

Texto	Inicio	Mid1	Total	Mid2
abcdef	2	bcdef	3	bcd
abcdef	3	cdef	2	cd
210714	2	10714	3	107
210714	3	0714	2	07

Vea también:

Index (page 1473)

Ord

Ord() devuelve el número Unicode de punto de código del primer carácter de la cadena de entrada.

Sintaxis:

```
Ord(text)
```

Tipo de datos que devuelve: Entero

Ejemplos y resultados:

Ejemplo: Expresión de gráfico

Ejemplo	Resultado
`Ord('A')`	Devuelve el entero 65.
`Ord('Ab')`	Devuelve el entero 65.

Ejemplo: Script de carga

```
//Guqin (Chinese: 古琴) – 7-stringed zithers T2: Load *,  ord(Chinese) as OrdUnicode,
        ord(Western) as OrdASCII;                    Load * inline [ Chinese, Western 古琴,
Guqin ];
```

Resultado:

Chino	Occidental	OrdASCII	OrdUnicode
古琴	Guqin	71	21476

PurgeChar

PurgeChar() devuelve una cadena que contiene los caracteres de la cadena introducida ('text'), menos los caracteres que aparecen en el segundo argumento ('remove_chars').

Sintaxis:

```
PurgeChar(text, remove_chars)
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
text	La cadena original.
remove_chars	Una cadena que contiene los caracteres de text que deben eliminarse.

Tipo de datos que devuelve: cadena

Ejemplo: Expresiones de gráfico

Ejemplo	Resultado
`PurgeChar ( 'a1b2c3','123' )`	Devuelve "abc".
`PurgeChar ( 'a1b2c3','312' )`	Devuelve "abc".

Ejemplo: Script de carga

```
T1:
Load
*,
purgechar(String1, String2) as PurgeChar;
Load * inline [
String1, String2
'a1b2c3', '123'
];
```

Resultados

Tabla de Qlik Sense que muestra el resultado de usar la función *PurgeChar* en el script de carga.

String1	String2	PurgeChar
a1b2c3	123	abc

Vea también:

KeepChar (page 1477)

Repeat

Repeat() forma una cadena que consiste en la cadena introducida, repetida el número de veces definido por el segundo argumento.

Sintaxis:

```
Repeat(text[, repeat_count])
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
text	La cadena original.
repeat_count	Define el número de veces que se repetirán los caracteres de la cadena **text** en la cadena resultante.

Ejemplo: Expresión de gráfico

Ejemplo	Resultado
`Repeat( ' * ', rating )` when **rating** = 4	Devuelve '****'

Ejemplo: Script de carga

```
T1: Load *, repeat(String,2) as Repeat; Load * inline [ String hello world! hOw aRe you? ];
```

Resultado

Cadena	Repetir
hello world!	hello world!hello world!
hOw aRe you?	hOw aRe you?hOw aRe you?

Replace

Replace() devuelve una cadena tras haber reemplazado todas las veces en que aparece una determinada subcadena dentro de la cadena introducida por otra subcadena. La función no es recursiva y funciona de izquierda a derecha.

Sintaxis:

```
Replace(text, from_str, to_str)
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
text	La cadena original.
from_str	Una cadena que puede aparecer una o más veces dentro de la cadena de texto **text** introducida.
to_str	La cadena que reemplazará todas las instancias de **from_str** dentro de la cadena **text**.

Ejemplos y resultados:

Ejemplo	Resultado
`Replace('abccde','cc','xyz')`	Devuelve 'abxyzde'

Vea también:

Right

Right() devuelve una cadena formada por los últimos caracteres (los situados más a la derecha) de la cadena de entrada, donde el número de caracteres viene determinado por el segundo argumento.

Sintaxis:

```
Right(text, count)
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
text	La cadena original.
count	Define el número de caracteres que se incluirán en la parte derecha de la cadena **text**.

Ejemplo: Expresión de gráfico

Ejemplo	Resultado
`Right('abcdef', 3)`	Devuelve 'def'

Ejemplo: Script de carga

```
T1:
Load
*,
right(Text,Start) as Right;
Load * inline [
Text, Start
'abcdef', 3
'2021-07-14', 4
'2021-07-14', 2
];
```

Resultado

Tabla de Qlik Sense que muestra el resultado de usar la función *Right* en el script de carga.

Texto	Inicio	Right
abcdef	3	def
2021-07-14	4	7-14
2021-07-14	2	14

RTrim

RTrim() devuelve la cadena introducida libre de espacios finales.

Sintaxis:

```
RTrim(text)
```

Tipo de datos que devuelve: cadena

Ejemplo: Expresiones de gráfico

Ejemplo	Resultado
`RTrim( ' abc' )`	Devuelve ' abc'
`RTrim( 'abc ' )`	Devuelve 'abc'

Ejemplo: Script de carga

```
Set verbatim=1;    T1:  Load *, len(RtrimString) as RtrimStringLength;         Load *, rtrim
(String) as RtrimString; Load *, len(String) as StringLength;              Load * Inline [
String '   abc    ' ' def  '];
```

La instrucción "Set verbatim=1" se incluye en el ejemplo para garantizar que los espacios no se recorten automáticamente antes de la demostración de la función rtrim. Vea Verbatim (page 210) si desea más información.

Resultado

Cadena	StringLength	RtrimStringLength
def	6	4
abc	10	6

Vea también:

LTrim (page 1481)

SubField

SubField() se utiliza para extraer componentes de subcadenas de un campo de cadena principal, donde los campos de registro originales constan de dos o más partes separadas por un delimitador.

La función **Subfield()** debe utilizarse, por ejemplo, para extraer el nombre y apellido de una lista de registros que contienen nombres completos, las partes que componen el nombre de una ruta, o para extraer datos de tablas separadas por comas.

Si utiliza la función **Subfield()** en una sentencia **LOAD** con el parámetro opcional field_no omitido, se generará un registro completo para cada subcadena. Si se cargan varios campos utilizando **Subfield()** se crean los productos cartesianos de todas las combinaciones.

Sintaxis:

```
SubField(text, delimiter[, field_no ])
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumentos

Argumento	Descripción
text	La cadena original. Puede ser un texto escrito directamente en el código, una variable, una expansión de signo dólar u otra expresión.
delimiter	Un carácter dentro de los datos introducidos en **text** que divide la cadena en partes componentes.

Argumento	Descripción
field_no	El tercer argumento opcional es un entero que especifica cuál de las subcadenas de la cadena principal **text** se devolverá. Utilice el valor 1 para volver a la primera subcadena, 2 para volver a la segunda subcadena, etc. • Si **field_no** es un valor positivo, las subcadenas se extraen de izquierda a derecha. • Si **field_no** es un valor negativo, las subcadenas se extraen de derecha a izquierda.

SubField() se puede usar en lugar de utilizar combinaciones complejas de funciones como Len(), Right(), Left(), Mid() y otras funciones de cadena.

Ejemplos: Expresiones de scripts y gráficos que usan SubField

Ejemplos: expresiones de script y de gráfico

Ejemplos básicos

Ejemplo	Resultado
`SubField(S, ';' ,2)`	Devuelve 'cde' si **S** es 'abc;cde;efg'.
`SubField(S, ';' ,1)`	Devuelve una cadena vacía si **S** es una cadena vacía.
`SubField(S, ';' ,1)`	Devuelve una cadena vacía si **S** es ';'.
Supongamos que tiene una variable que contiene una ruta de archivo vMyPath, `Set vMyPath=\Users\ext_jrb\Documents\Qlik\Sense\Apps;.`	En un gráfico de texto e imagen, puede agregar una medida como: `SubField(vMyPath, '\',-3)`, lo cual da como resultado "Qlik", porque es la tercera subcadena desde el extremo derecho de la variable vMyPath.

Ejemplo de script 1

Script de carga

Cargue las siguientes expresiones de script y datos en el editor de carga de datos.

```
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;
```

Crear una visualización

Cree una visualización de tabla en una hoja de Qlik Sense con **Name**, **FirstName** y **SurName** como dimensiones.

Resultado

Name	FirstName	SurName
Dave Owen	Dave	Owen
Joe Tem	Joe	Tem

Explicación

La función **SubField()** extrae la primera subcadena de **Name** estableciendo el argumento **field_no** en 1. Dado que el valor de **field_no** es positivo, se sigue un orden de izquierda a derecha para extraer la subcadena. Una segunda llamada de función extrae la segunda subcadena configurando el argumento de **field _no** en -1, lo que extrae la subcadena siguiendo un orden de derecha a izquierda.

Ejemplo de script 2

Script de carga

Cargue las siguientes expresiones de script y datos en el editor de carga de datos.

```
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 '|');
```

Crear una visualización

Cree una visualización de tabla en una hoja de Qlik Sense con **Instrument**, **Player** y **Project**como dimensiones.

Resultado

Instrument	Player	Project
Guitar	Mike	Music

Instrument	Player	Project
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

Explicación

Este ejemplo muestra cómo usar múltiples instancias de la función **Subfield()**, cada una con el parámetro field_no omitido, desde dentro de la misma sentencia **LOAD** crea productos cartesianos de todas las combinaciones. La opción **DISTINCT** sirve para evitar crear registros duplicados.

SubStringCount

SubStringCount() devuelve el número de veces que aparece la subcadena especificada en el texto de la cadena de entrada. Si no existe coincidencia alguna, devuelve 0.

Sintaxis:

```
SubStringCount(text, sub_string)
```

Tipo de datos que devuelve: Entero

Argumentos:

Argumento	Descripción
text	La cadena original.
sub_string	Una cadena que puede aparecer una o más veces dentro de la cadena de **text** introducida.

Ejemplo: Expresiones de gráfico

Ejemplo	Resultado
`SubStringCount ( 'abcdefgcdxyz', 'cd' )`	Devuelve 2
`SubStringCount ( 'abcdefgcdxyz', 'dc' )`	Devuelve (0)

Ejemplo: Script de carga

```
T1: Load *,  substringcount(upper(Strings),'AB') as SubStringCount_AB; Load * inline [ Strings
ABC:DEF:GHI:AB:CD:EF:GH aB/cd/ef/gh/Abc/abandoned ];
```

Resultado

Cadenas de texto	SubStringCount_AB
aB/cd/ef/gh/Abc/abandoned	3
ABC:DEF:GHI:AB:CD:EF:GH	2

TextBetween

TextBetween() devuelve el texto de la cadena de entrada que se da entre los caracteres especificados como delimitadores.

Sintaxis:

```
TextBetween(text, delimiter1, delimiter2[, n])
```

Tipo de datos que devuelve: cadena

Argumentos:

Argumento	Descripción
text	La cadena original.
delimiter1	Especifica el primer carácter delimitador (o cadena) que buscar en **text**.
delimiter2	Especifica el segundo carácter delimitador (o cadena) que buscar en **text**.
n	Define entre qué aparición del par delimitador se ha de buscar. Por ejemplo, un valor de 2 devuelve los caracteres entre la segunda vez que aparece delimiter1 y la segunda vez que aparece delimiter2.

Ejemplo: Expresiones de gráfico

Ejemplo	Resultado
`TextBetween('<abc>', '<', '>')`	Devuelve 'abc'
`TextBetween('<abc><de>', '<', '>',2)`	Devuelve 'de'

Ejemplo	Resultado
`TextBetween('abc', '<', '>')` `TextBetween('<a<b', '<', '>')`	Ambos ejemplos devuelven NULL. Si alguno de los delimitadores no se encuentra en la cadena, se devuelve NULL.
`TextBetween('<>', '<', '>')`	Devuelve una cadena de longitud cero.
`TextBetween('<abc>', '<', '>', 2)`	Devuelve NULL, ya que n es mayor que el número de veces que aparecen los delimitadores.

Ejemplo: Script de carga

```
Load *, textbetween(Text,'<','>') as TextBetween, textbetween(Text,'<','>',2) as
SecondTextBetween; Load * inline [ Text <abc><de> <def><ghi><jkl> ];
```

Resultado

Texto	TextBetween	SecondTextBetween
<abc><de>	abc	de
<def><ghi><jkl>	def	ghi

Trim

Trim() devuelve la cadena introducida libre de todos los espacios iniciales y finales.

Sintaxis:

```
Trim(text)
```

Tipo de datos que devuelve: cadena

Ejemplos y resultados:

Ejemplo: Expresión de gráfico

Ejemplo	Resultado
`Trim( ' abc' )`	Devuelve 'abc'
`Trim( 'abc ' )`	Devuelve 'abc'
`Trim( ' abc ' )`	Devuelve 'abc'

Ejemplo: Script de carga

```
Set verbatim=1;                                          T1: Load *, len(TrimString) as TrimStringLength;
(String) as TrimString; Load *, len(String) as StringLength;                Load * inline [
String '   abc    ' ' def  '](delimiter is '\t');
```

La instrucción "Set verbatim=1" se incluye en el ejemplo para garantizar que los espacios no se recorten automáticamente antes de la demostración de la función trim. Vea Verbatim (page 210) si desea más información.

Resultado:

Cadena	StringLength	TrimStringLength
def	6	3
abc	10	3

Upper

Upper() convierte todos los caracteres de la cadena introducida en mayúscula para todos los caracteres de texto de la expresión. Los números y símbolos se ignoran.

Sintaxis:

```
Upper(text)
```

Tipo de datos que devuelve: cadena

Ejemplo: Expresión de gráfico

Ejemplo	Resultado
`Upper(' abcD')`	Devuelve 'ABCD'

Ejemplo: Script de carga

```
Load  String,Upper(String) Inline [String rHode iSland washingTon d.C. new york];
```

Resultado

Cadena	Upper(String)
rHode iSland	RHODE ISLAND
washingTon d.C.	WASHINGTON D.C.
new york	NEW YORK

8.25 Funciones de sistema

Las funciones de sistema permiten acceder a las propiedades del sistema, de un dispositivo y de la app de Qlik Sense.

Descripción general de las funciones de sistema

Algunas de las funciones se describen a continuación tras la vista genérica. Para esas funciones, puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Author()
Esta función devuelve una cadena que contiene la propiedad de autor de la actual app. Puede utilizarse tanto en el script de carga de datos como en una expresión de gráficos.

La propiedad de autor no puede fijarse en la versión actual de Qlik Sense. Si migra un documento QlikView, la propiedad de autor se mantendrá.

ClientPlatform()
Esta función devuelve la cadena de agente de usuario del navegador cliente. Puede utilizarse tanto en el script de carga de datos como en una expresión de gráficos.

Ejemplo:

```
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/35.0.1916.114 Safari/537.36
```

ComputerName
Esta función devuelve una cadena con el nombre del equipo informático, tal como lo devuelve el sistema operativo. Puede utilizarse tanto en el script de carga de datos como en una expresión de gráficos.

Si el nombre del equipo informático tiene más de 15 caracteres, la cadena solo contendrá los primeros 15 caracteres.

```
ComputerName( )
```

DocumentName
Esta función devuelve una cadena con el nombre de la app actual de Qlik Sense, sin la ruta de acceso pero sí con la extensión. Puede utilizarse tanto en el script de carga de datos como en una expresión de gráficos.

```
DocumentName( )
```

DocumentPath
Esta función devuelve una cadena que contiene la ruta completa a la app actual de Qlik Sense. Puede utilizarse tanto en el script de carga de datos como en una expresión de gráficos.

```
DocumentPath( )
```

Esta función no es posible en modo estándar.

DocumentTitle
Esta función devuelve una cadena que contiene el título de la app actual de Qlik Sense. Puede utilizarse tanto en el script de carga de datos como en una expresión de gráficos.

```
DocumentTitle( )
```

EngineVersion
Esta función devuelve la versión completa del motor Qlik Sense como una cadena.

```
EngineVersion ()
```

GetCollationLocale
Esta función de script devuelve el nombre del lugar de cotejo que se está utilizando. Si la variable CollationLocale no se ha definido, devuelve la configuración regional de la máquina del usuario real.

```
GetCollationLocale( )
```

GetObjectField
GetObjectField() devuelve el nombre de la dimensión. **Index** es un entero opcional que indica la dimensión que debe devolverse.

```
GetObjectField - función de gráfico([index])
```

GetRegistryString
Esta función devuelve el valor de una clave en el registro de Windows. Puede utilizarse tanto en el script de carga de datos como en una expresión de gráficos.

```
GetRegistryString(path, key)
```

Esta función no es posible en modo estándar.

GetSysAttr
Esta función devuelve los atributos de dominio de espacio y espacio empresarial para una app seleccionada. Puede utilizarse tanto en el script de carga de datos como en una expresión de gráficos.

```
GetSysAttr (name)
```

Si utiliza esta función en Qlik Sense administrado por el cliente, solo devuelve valores de datos vacíos.

IsPartialReload
Esta función devuelve 1 (True) si la recarga actual es parcial, de lo contrario devuelve 0 (False).

```
IsPartialReload ()
```

InObject
La función de gráfico **InObject()** evalúa si el objeto actual está contenido dentro de otro objeto con el ID especificado en el argumento de la función. El objeto puede ser una hoja o una visualización.

```
InObject - función de gráfico(id_str)
```

ObjectId

La función de gráfico **ObjectId()** devuelve el ID del objeto en el que se evalúa la expresión. La función toma un argumento opcional que especifica a qué tipo de objeto se refiere la función. El objeto puede ser una hoja o una visualización. Esta función solo está disponible en expresiones de gráfico.

```
ObjectId - función de gráfico([object_type_str])
```

OSUser

Esta función devuelve una cadena que contiene el nombre del usuario actualmente conectado. Puede utilizarse tanto en el script de carga de datos como en una expresión de gráficos.

```
OSUser( )
```

En Qlik Sense Desktop y Qlik Sense Mobile Administrado por el cliente, esta función siempre devuelve "Personal\Me".

ProductVersion

Esta función devuelve el número completo de la versión de Qlik Sense como una cadena.

Esta función está en desuso y ha sido reemplazada por **EngineVersion()**.

```
ProductVersion ()
```

ReloadTime

Esta función devuelve una fecha-hora indicando el momento en que finalizó la última carga de datos. Puede utilizarse tanto en el script de carga de datos como en una expresión de gráficos.

```
ReloadTime( )
```

StateName

StateName() devuelve el nombre del estado alterno de la visualización en la que se está usando. StateName Se puede usar, por ejemplo, para crear visualizaciones con texto dinámico y colores para reflejar cuándo se cambia el estado de una visualización. Esta función se puede utilizar en expresiones de gráfico, pero no puede utilizarse para determinar el estado al que se refiere la expresión.

```
StateName - función de gráfico()
```

EngineVersion

Esta función devuelve la versión completa del motor Qlik Sense como una cadena.

Sintaxis:

```
EngineVersion()
```

GetSysAttr

Esta función devuelve los atributos de dominio de espacio y espacio empresarial para una app seleccionada. Puede utilizarse tanto en el script de carga de datos como en una expresión de gráficos.

Si utiliza esta función en Qlik Sense administrado por el cliente, devuelve valores de datos vacíos. Por lo tanto, puede utilizar la función para desarrollar scripts de carga en Qlik Sense administrado por el cliente sin encontrar errores, para luego subir las aplicaciones a Qlik Cloud.

Para acceder a la documentación completa de la función Qlik Cloud, vea GetSysAttr - función de gráfico y script.

InObject - función de gráfico

La función de gráfico **InObject()** evalúa si el objeto actual está contenido dentro de otro objeto con el ID especificado en el argumento de la función. El objeto puede ser una hoja o una visualización.

Esta función se puede usar para mostrar la jerarquía de objetos en una hoja, desde el objeto de hoja de nivel superior hasta las visualizaciones anidadas dentro de otras visualizaciones. Esta función se puede usar junto con las funciones **if** y **ObjectId** para crear una navegación personalizada en sus aplicaciones.

Sintaxis:

```
InObject(id_str)
```

Tipo de datos que devuelve: Booleano

En Qlik Sense, el valor booleano verdadero viene representado por -1 y el valor falso está representado por 0.

Argumentos

Argumento	Descripción
id_str	Un valor de cadena de texto que representa el ID del objeto que se está evaluando.

El ID de la hoja se puede obtener de la URL de la aplicación. Para las visualizaciones, use las opciones de **Desarrollador** para identificar el ID del objeto y la cadena de texto del tipo de objeto.

Haga lo siguiente:

1. En el modo de análisis, agregue el siguiente texto a su URL:
   /options/developer
2. Haga clic con el botón derecho en una visualización y después haga clic en **Developer**.
3. En **Propiedades**, obtenga el ID de objeto del encabezado del cuadro de diálogo y el tipo de objeto de la propiedad **"qType**".

Limitaciones:

Esta función puede dar resultados inesperados cuando se invoca en un objeto (por ejemplo, un botón) dentro de un contenedor que es un elemento maestro. Esta limitación también se aplica a los elementos maestros del panel de filtrado, que son contenedores para varios cuadros de lista. Esto se debe a cómo los elementos maestros utilizan la jerarquía de objetos.

InObject() se utiliza a menudo en combinación con las siguientes funciones:

Funciones relacionadas

Función	Interacción
if (page 577)	Las funciones **if** y **ObjectId** se pueden utilizar juntas para crear expresiones condicionales. Por ejemplo, las visualizaciones pueden lograr colores condicionales a través de expresiones que usan estas funciones.
ObjectId - función de gráfico (page 1502)	Similar a **if**, **ObjectId** también se usa con **InObject** para crear expresiones condicionales.

Ejemplo 1: funcionalidad básica

Gráfico de expresión y resultados

El siguiente ejemplo básico demuestra cómo determinar si un objeto está contenido dentro de otro objeto. En este caso, comprobaremos si un objeto de **Texto e imagen** reside en un objeto de hoja utilizando el ID de la hoja como argumento.

Haga lo siguiente:

1. Abra una hoja nueva y arrastre un gráfico de **Texto e imagen** a la hoja.
2. En el panel de propiedades, haga clic en **Añadir medida**.
3. Haga clic en fx para abrir el editor de expresiones.
4. Pegue la siguiente expresión en el cuadro de diálogo:
   `=InObject()`
5. Modifique la expresión para incluir el ID de su hoja como una cadena de caracteres entre paréntesis. Por ejemplo, para una hoja con ID 1234-5678, usaría lo siguiente:
   `=InObject('1234-5678')`
6. Haga clic en **Aplicar**.

El valor `-1` se muestra en el gráfico, lo que indica que la expresión se evaluó como verdadera.

Ejemplo 2: Objetos con colores condicionales

Gráfico de expresión y resultados

General

El ejemplo siguiente muestra cómo crear botones de navegación personalizados que muestran diferentes colores para indicar la hoja que está abierta actualmente.

Comience creando una nueva aplicación y abriendo el Editor de carga de datos. Pegue el siguiente script de carga en una nueva pestaña: Tenga en cuenta que los datos en sí son un marcador de posición y no se utilizarán en el contenido de ejemplo.

Script de carga

```
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
];
```

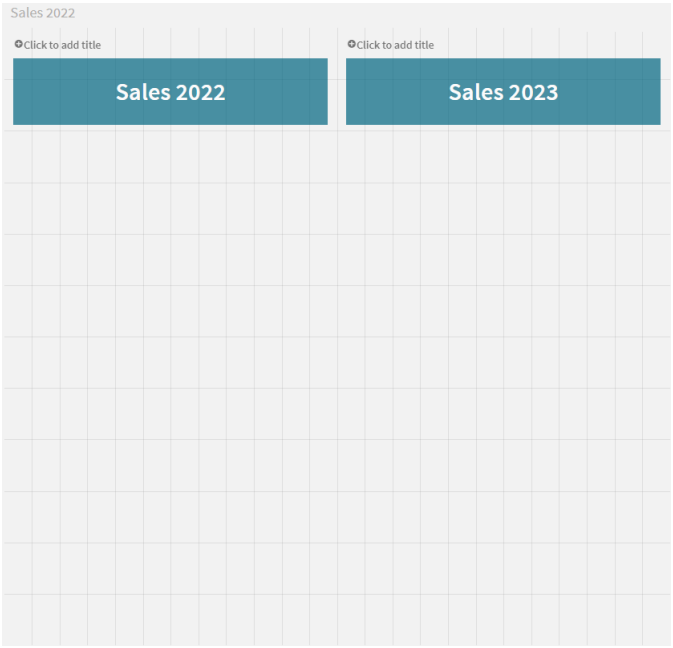
Crear las visualizaciones

Cargue los datos y cree dos hojas nuevas. Escriba los títulos *Ventas 2022* y *Ventas 2023* respectivamente.

A continuación, cree dos objetos de botón que se utilizarán para navegar entre las dos hojas.

Haga lo siguiente:

1. Agregue dos objetos de **Botón** a la hoja.
2. En **Aspecto** > **General**, establezca la **Etiqueta** de cada botón a *Ventas 2022* y *Ventas 2023*, respectivamente.
3. Organice los botones para que coincidan con la siguiente imagen.
   Ventas 2022 con dos botones de navegación



4. Seleccione el botón *Ventas 2022* y expanda **Acciones y navegación** en el panel de propiedades.
5. Haga clic en **Agregar acción** y, en **Navegación**, seleccione **Ir a una hoja**.
6. En **Hoja**, seleccione *Ventas 2022*.
7. Repita la configuración de esta acción de botón para vincular el botón **Ventas 2023** a la hoja *Ventas 2023*.
8. Convierta los botones en elementos maestros haciendo clic con el botón derecho sobre ellos y seleccionando **Añadir a elementos maestros**.

Ahora puede copiar cada botón y pegarlo en la hoja *Ventas 2023*, usando el mismo tamaño y disposición en la hoja.

Crear colores condicionales

A continuación, configure los botones para que sean azules si están vinculados a la hoja abierta actualmente y gris claro si están vinculados a la hoja que no está abierta.

Haga lo siguiente:

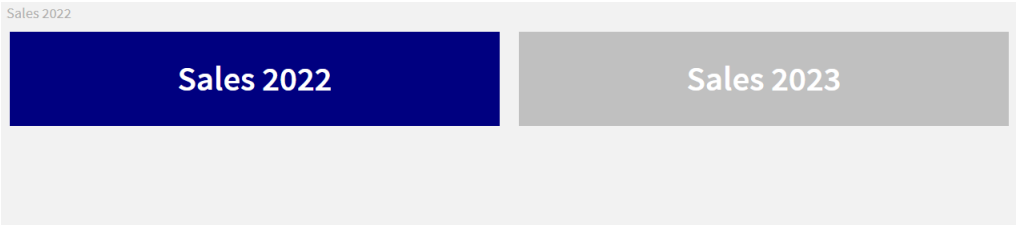
1. Abra la hoja *Ventas 2022* y obtenga el ID de la hoja de la URL. Mantenga abierta la hoja *Ventas 2022*.
2. Haga clic en el elemento maestro del botón **Ventas 2022** y seleccione **Editar** en el panel de propiedades.
3. En **Aspecto** > **Fondo**, seleccione dar color al botón **Por expresión**.
4. En **Expresión**, pegue el siguiente texto:
   =if(InObject(''), Blue(), LightGray())
5. Entre paréntesis en la expresión anterior, pegue el ID de hoja para la hoja *Ventas 2022*.

El botón ahora está configurado para mostrarse en azul cuando la hoja *Ventas 2022* esté abierta y en gris claro cuando no está abierta.

Repita las instrucciones anteriores para la hoja *Ventas 2023*, vinculando el elemento maestro del botón **Ventas 2023** al ID de la hoja *Ventas 2023*.

Cada hoja ahora debería tener dos botones que indiquen la hoja actualmente abierta con el color azul.

La hoja Ventas 2022 de color azul para indicar que actualmente se muestra Ventas 2022



IsPartialReload

Esta función devuelve 1 (True) si la recarga actual es parcial, de lo contrario devuelve 0 (False).

Sintaxis:

```
IsPartialReload()
```

ObjectId - función de gráfico

La función de gráfico **ObjectId()** devuelve el ID del objeto en el que se evalúa la expresión. La función toma un argumento opcional que especifica a qué tipo de objeto se refiere la función. El objeto puede ser una hoja o una visualización. Esta función solo está disponible en expresiones de gráfico.

Sintaxis:

```
ObjectId([object_type_str])
```

Tipo de datos que devuelve: cadena de texto

El único argumento de la función, **object_type_str**, es opcional y hace referencia a un valor de cadena de caracteres que representa el tipo del objeto.

Argumentos

Argumento	Descripción
object_type_str	Un valor de cadena que representa el tipo de objeto que se está evaluando.

Si no se especifica ningún argumento en la expresión de la función, **ObjectId()** devuelve el ID del objeto en el que se utiliza la expresión. Para devolver el ID del objeto de hoja en el que aparece la visualización, utilice *ObjectId('sheet')*.

En el caso de objetos de visualización anidados dentro de otros objetos de visualización, especifique el tipo de objeto deseado en el argumento de la función para obtener diferentes resultados. Por ejemplo, para un gráfico de **Texto e imagen** dentro de un contenedor, use *"text-image"* para devolver el objeto de **Texto e imagen** y *"contenedor"* para devolver el ID del contenedor.

Haga lo siguiente:

1. En el modo de análisis, agregue el siguiente texto a su URL:
   /options/developer
2. Haga clic con el botón derecho en una visualización y después haga clic en **Developer**.
3. En **Propiedades**, obtenga el ID de objeto del encabezado del cuadro de diálogo y el tipo de objeto de la propiedad **"qType**".

Limitaciones:

Esta función puede dar resultados inesperados cuando se invoca en un objeto (por ejemplo, un botón) dentro de un contenedor que es un elemento maestro. Esta limitación también se aplica a los elementos maestros del panel de filtrado, que son contenedores para varios cuadros de lista. Esto se debe a cómo los elementos maestros utilizan la jerarquía de objetos.

La expresión del gráfico *ObjectId('sheet')* devolverá una cadena vacía en esos casos, mientras que *ObjectId ('masterobject')* mostrará el identificador del elemento maestro propietario.

ObjectId() se utiliza a menudo en combinación con las siguientes funciones:

Funciones relacionadas

Función	Interacción
if (page 577)	Las funciones **if** y **ObjectId** se pueden utilizar juntas para crear expresiones condicionales. Por ejemplo, las visualizaciones pueden lograr colores condicionales a través de expresiones que usan estas funciones.
InObject - función de gráfico (page 1498)	Similar a **if**, **InObject** también se usa con **ObjectId** para crear expresiones condicionales.

Ejemplo 1: Devolver el ID de un objeto gráfico

Gráfico de expresión y resultados

El siguiente ejemplo básico demuestra cómo devolver el ID de una visualización.

Haga lo siguiente:

1. Abra una hoja nueva y arrastre un gráfico de **Texto e imagen** a la hoja.
2. En el panel de propiedades, haga clic en **Añadir medida**.
3. Haga clic en fx para abrir el editor de expresiones.
4. Pegue la siguiente expresión en el cuadro de diálogo:
   `=ObjectId()`
5. Haga clic en **Aplicar**.

El ID del objeto **Texto e imagen** se muestra en la visualización.

El mismo resultado se puede lograr con la siguiente expresión:

```
=ObjectId('text-image')
```

Ejemplo 2: ID de la hoja de devolución

Gráfico de expresión y resultados

El siguiente ejemplo básico demuestra cómo devolver el ID de la hoja en la que aparece una visualización.

Haga lo siguiente:

1. Abra una hoja nueva y arrastre un gráfico de **Texto e imagen** a la hoja.
2. En el panel de propiedades, haga clic en **Añadir medida**.
3. Haga clic en fx para abrir el editor de expresiones.
4. Pegue la siguiente expresión en el cuadro de diálogo:
   `=ObjectId('sheet')`
5. Haga clic en **Aplicar**.

El ID de la hoja se muestra en la visualización.

Ejemplo 3: expresión anidada

Gráfico de expresión y resultados

El siguiente ejemplo muestra cómo la función **ObjectId()** se puede anidar dentro de otras expresiones.

Haga lo siguiente:

1. Abra una hoja nueva y arrastre un gráfico de **Texto e imagen** a la hoja.
2. En el panel de propiedades, haga clic en **Añadir medida**.
3. Haga clic en fx para abrir el editor de expresiones.
4. Pegue la siguiente expresión en el cuadro de diálogo:
   =if(InObject(ObjectId('text-image')), 'In Text & image', 'Not in Text & image')
5. Haga clic en **Aplicar**.

El texto en *Texto e imagen* aparece en el gráfico, lo que indica que el objeto al que se hace referencia en la expresión es un gráfico de **Texto e imagen**.

Para ver un ejemplo más detallado del uso de la coloración condicional, vea el ejemplo en *InObject - función de gráfico (page 1498)*.

ProductVersion

Esta función devuelve el número completo de la versión de Qlik Sense como una cadena. Esta función está en desuso y ha sido reemplazada por **EngineVersion()**.

Sintaxis:

```
ProductVersion()
```

StateName - función de gráfico

StateName() devuelve el nombre del estado alterno de la visualización en la que se está usando. StateName Se puede usar, por ejemplo, para crear visualizaciones con texto dinámico y colores para reflejar cuándo se cambia el estado de una visualización. Esta función se puede utilizar en expresiones de gráfico, pero no puede utilizarse para determinar el estado al que se refiere la expresión.

Sintaxis:

```
StateName ()
```

Example 1:

Texto dinámico

```
='Region - ' & if(StateName() = '$', 'Default', StateName())
```

Example 2:

Colores dinámicos

```
if(StateName() = 'Group 1', rgb(152, 171, 206),
      if(StateName() = 'Group 2', rgb(187, 200, 179),
            rgb(210, 210, 210)
      )
)
```

8.26 Funciones de tabla

Las funciones de tabla devuelven información sobre la tabla de datos que se está leyendo en ese momento. Si no se especifica nombre de tabla alguno y la función se emplea dentro de una sentencia **LOAD**, se asume la tabla actual.

Todas las funciones pueden utilizarse en el script de carga de datos, mientras que **NoOfRows** solo puede utilizarse en una expresión de gráfico.

Vista general de las funciones de tabla

Algunas de las funciones se describen a continuación tras la vista genérica. Para esas funciones, puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

FieldName

La función de script **FieldName** devuelve el nombre del campo con el número especificado dentro de una tabla previamente cargada. Si la función se utiliza dentro de una sentencia **LOAD**, no debe hacer referencia a la tabla que se está cargando actualmente.

```
FieldName (field_number ,table_name)
```

FieldNumber
La función de script **FieldNumber** devuelve el número de un campo especificado dentro de una tabla previamente cargada. Si la función se utiliza dentro de una sentencia **LOAD**, no debe hacer referencia a la tabla que se está cargando actualmente.

```
FieldNumber (field_name ,table_name)
```

NoOfFields
La función de script **NoOfFields** devuelve el número de campos de una tabla previamente cargada. Si la función se utiliza dentro de una sentencia **LOAD**, no debe hacer referencia a la tabla que se está cargando actualmente.

```
NoOfFields (table_name)
```

NoOfRows
La función **NoOfRows** devuelve el número de filas (registros) de una tabla previamente cargada. Si la función se utiliza dentro de una sentencia **LOAD**, no debe hacer referencia a la tabla que se está cargando actualmente.

```
NoOfRows (table_name)
```

NoOfTables
Esta función de script devuelve el número de tablas previamente cargadas.

```
NoOfTables()
```

TableName
Esta función de script devuelve el nombre de la tabla con el número especificado.

```
TableName(table_number)
```

TableNumber
Esta función de script devuelve el número de la tabla especificada. La primera tabla tiene el número 0.

Si table_name no existe, devuelve NULL.

```
TableNumber(table_name)
```

Ejemplo:

En este ejemplo, queremos crear una tabla con información sobre las tablas y los campos que se han cargado.

Primero, cargamos algunos datos de ejemplo. Esto crea las dos tablas que usaremos para ilustrar las funciones de tabla descritas en esta sección.

```
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 ;
```

Después recorremos las tablas que se han cargado, usando la función **NoOfTables**, y luego a través de los campos de cada tabla, usando la función **NoOfFields**, y cargamos la información utilizando las funciones de la tabla.

```
//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;
```

La tabla Tables resultante tendrá el siguiente aspecto:

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

La función de script **FieldName** devuelve el nombre del campo con el número especificado dentro de una tabla previamente cargada. Si la función se utiliza dentro de una sentencia **LOAD**, no debe hacer referencia a la tabla que se está cargando actualmente.

Sintaxis:

```
FieldName(field_number ,table_name)
```

Argumentos:

Argumentos

Argumento	Descripción
field_number	El número de campo al que desee hacer referencia.
table_name	La tabla que contiene el campo al que desea hacer referencia.

Ejemplo:

```
LET a = FieldName(4,'tab1');
```

FieldNumber

La función de script **FieldNumber** devuelve el número de un campo especificado dentro de una tabla previamente cargada. Si la función se utiliza dentro de una sentencia **LOAD**, no debe hacer referencia a la tabla que se está cargando actualmente.

Sintaxis:

```
FieldNumber(field_name ,table_name)
```

Argumentos:

Argumentos

Argumento	Descripción
field_name	El nombre del campo.
table_name	El nombre de la tabla que contiene el campo.

Si el campo field_name no existe en table_name, o table_name no existe, la función devuelve 0.

Ejemplo:

```
LET a = FieldNumber('Customer','tab1');
```

NoOfFields

La función de script **NoOfFields** devuelve el número de campos de una tabla previamente cargada. Si la función se utiliza dentro de una sentencia **LOAD**, no debe hacer referencia a la tabla que se está cargando actualmente.

Sintaxis:

```
NoOfFields(table_name)
```

Argumentos:

Argumentos

Argumento	Descripción
table_name	El nombre de la tabla.

Ejemplo:

```
LET a = NoOfFields('tab1');
```

NoOfRows

La función **NoOfRows** devuelve el número de filas (registros) de una tabla previamente cargada. Si la función se utiliza dentro de una sentencia **LOAD**, no debe hacer referencia a la tabla que se está cargando actualmente.

Sintaxis:

```
NoOfRows(table_name)
```

Argumentos:

Argumentos

Argumento	Descripción
table_name	El nombre de la tabla.

Ejemplo:

```
LET a = NoOfRows('tab1');
```

8.27 Funciones trigonométricas e hiperbólicas

En esta sección se describen funciones para realizar operaciones trigonométricas e hiperbólicas. En todas las funciones, los argumentos son expresiones que devuelven ángulos medidos en radianes, donde **x** debe interpretarse como un número real.

Todos los ángulos se miden en radianes.

Todas las funciones pueden utilizarse tanto en el script de carga de datos como en las expresiones de gráficos.

cos
Coseno de **x**. El resultado es un número entre -1 y 1.

```
cos( x )
```

acos
Coseno inverso de **x**. La función solo se define si -1≤**x**≤1. El resultado es un número entre 0 y π.

```
acos( x )
```

sin
Seno de **x**. El resultado es un número entre -1 y 1.

```
sin( x )
```

asin
Seno inverso de **x**. La función solo se define si -1≤**x**≤1. El resultado es un número entre - π/2 y π/2.

```
asin( x )
```

tan
Tangente de **x**. El resultado es un número real.

```
tan( x )
```

atan
Tangente inversa de **x**. El resultado es un número entre - π/2 y π/2.

```
atan( x )
```

atan2
Generalización bidimensional de la función tangente inversa. Devuelve el ángulo entre el origen y el punto representados por las coordenadas **x** y **y**. El resultado es un número entre - π y + π.

```
atan2( y,x )
```

cosh
Coseno hiperbólico de **x**. El resultado es un número real positivo.

```
cosh( x )
```

sinh
Seno hiperbólico de **x**. El resultado es un número real.

```
sinh( x )
```

tanh
Tangente hiperbólica de **x**. El resultado es un número real.

```
tanh( x )
```

acosh
Coseno hiperbólico inverso de **x**. El resultado es un número real positivo.

```
acosh( x )
```

asinh
Seno hiperbólico inverso de **x**. El resultado es un número real.

```
asinh( x )
```

atanh
Tangente hiperbólica inversa de **x**. El resultado es un número real.

```
atanh( x )
```

Ejemplos:

El siguiente código de script carga una tabla de muestra y a continuación carga una tabla que contiene las operaciones trigonométricas e hiperbólicas sobre los valores.

```
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 Funciones de ventana

Las funciones de ventana realizan cálculos utilizando valores de varias filas para producir un valor para cada fila por separado. Las funciones de ventana solo se pueden calcular una vez que se lee toda la tabla.

Puede utilizar las funciones de ventana para realizar operaciones como:

- Comparar un valor numérico individual en una fila con el promedio, máximo o mínimo dentro de la columna.
- Calcular la clasificación de un valor individual, ya sea dentro de la columna o dentro de toda la tabla.

Las funciones de ventana no cambian la cantidad de registros de la tabla, pero pueden realizar tareas similares a las funciones de agregación o funciones relacionales y funciones de rango.

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Window

La función **Window** realiza cálculos a partir de varias filas, generando un valor para cada fila por separado.

```
Window - función de script(input_expr, [partition1, partition2, ...], [sort_
type, [sort_expr]],[filter_expr], [start_expr,end_expr])[row_window_size])
```

WRank

La función **WRank** realiza cálculos de clasificación o ranking dentro de **Window**.

```
WRank - función de script([TOTAL] expr[, mode[, fmt]])
```

Window - función de script

Window() realiza cálculos a partir de varias filas, generando un valor para cada fila por separado.

Puede utilizar las funciones **Window** para realizar operaciones como:

- Comparar un valor numérico individual en una fila con el promedio, máximo o mínimo dentro de la columna.
- Calcular la clasificación de un valor individual, ya sea dentro de la columna o dentro de toda la tabla.

La función **Window** no cambia la cantidad de registros en la tabla, pero aún puede realizar tareas similares como funciones de agregación, relacionales y de rango.

La función **Window** debe tener una caché dentro de la sentencia LOAD de la tabla con la que está trabajando para agregarlo a la tabla. Por ejemplo:

```
[Transactions]:
Load
       *,
       Window(avg(Expression1),[Num]);
LOAD
       TransLineID,
       TransID,
       "Num",
       Dim1,
       Dim2,
       Dim3,
       Expression1,
       Expression2,
       Expression3
FROM [lib://AttachedFiles/transactions.qvd] (qvd);
```

Window admite funciones generales, como redondeo u operaciones numéricas básicas. Por ejemplo:

```
Load *, Round(Window(Sum(Salary),Department)) as SumSalary
Load *, Window(Sum(Salary),Department) + 5 as SumSalary
```

Puede definir una ventana deslizante para la función **Window**. Esto establece el número de filas utilizadas al aplicar la función en el **Window** actual. Por ejemplo, puede configurar la ventana para que contenga las 3 filas anteriores y las 3 filas posteriores.

Sintaxis:

```
Window (input_expr, [partition1, partition2, ...], [sort_type, [sort_expr]],
[filter_expr], [start_expr,end_expr])
```

Tipo de datos que devuelve: Un nuevo campo agregado a la tabla resultante creada por la sentencia LOAD.

Argumentos:

Argumentos

Argumento	Descripción

input_expr	La expresión de entrada calculada y devuelta por la función. Debe ser cualquier expresión basada en una agregación, como `Median(Salary)`. Por ejemplo:
	`Window(Median(Salary)) as MedianSalary`
	El dato de entrada también puede ser un nombre de campo sin que se aplique ninguna agregación. En este caso, **Window** lo trata como la función **Only()** se aplica a ese campo. Por ejemplo:
	`Window(Salary,Department) as WSalary`
	
	Si lo desea también puede definir la partición con la expresión de entrada. La partición es la misma que la agrupación obtenida por la cláusula **group by**, con la diferencia de que el resultado se agrega como una nueva columna a la tabla de entrada. La partición no reduce la cantidad de registros de la tabla de entrada. Se pueden definir múltiples campos de partición.
	Ejemplo:
	`LOAD`
	`Window(Max(Sales), City, 'ASC', OrderDate, Sales > 300)`
	`+ AddMonths(OrderDate,-6) as MAX_Sales_City_Last_6_Mos,`
	`Window(Avg(Sales), City, 'ASC', OrderDate, City = 'Portland')`
	`+ AddMonths(OrderDate,-6) as Avg_Sales_Portland_Last_6_Mos,`
	`Window(Max(Sales), City, 'ASC', OrderDate, Sales > 300)`
	`+ AddMonths(OrderDate,-12) as MAX_Sales_City_Last_12_Mos;`
	`LOAD`
	`City,`
	`Sales,`
	`OrderDate`
	`FROM [lib://AttachedFiles/Sales Data.xlsx]`
	`(ooxml, embedded labels, table is [Sales Data]);`

partition1, partition2	Después de input_expr, puede definir cualquier número de particiones. Las particiones son campos que definen con qué combinaciones aplicar las agregaciones. La agregación se aplica por separado con cada partición. Por ejemplo: `Window(Avg(Salary), Unit, Department, Country) as AvgSalary` En el anterior, las particiones son *Unit*, *Department* y *Country*. Las particiones no son obligatorias, pero sí recomendables para que los campos tengan ventanas adecuadas.

Argumento	Descripción
sort_type, [sort_expr]]	Opcionalmente, especifique el tipo de ordenación y la expresión de orden. sort_type puede tener uno de estos dos valores: • ASC: Orden ascendente. • DESC: Orden descendente. Si define sort_type, necesita definir una expresión de orden. Esta es una expresión que decide el orden de las filas dentro de una partición. Por ejemplo: `Window(RecNo(), Department, 'ASC', Year)` En el ejemplo anterior, los resultados dentro de la partición están ordenados de forma ascendente por el campo *Year*. *El tipo de clasificación y la expresión de orden solo se requieren principalmente con las funciones **RecNo** y **WRank**.*
filter_expr	Opcionalmente, agregue una expresión de filtro. Esta es una expresión booleana que decide si el registro debe incluirse en el cálculo o no. Este parámetro se puede omitir por completo y el resultado debería ser que no hay filtro. Por ejemplo: `Window(avg(Salary), Department, 'ASC', Age, EmployeeID=3 Or EmployeeID=7) as wAvgSalary) as wAvgSalaryIfEmpIs3or7`

Argumento	Descripción

[start_expr,end_expr]	Opcionalmente, establezca el argumento para la funcionalidad de ventana deslizante. Una ventana deslizante requiere dos argumentos:
	• Expresión inicial: El número de filas anteriores a la fila actual que se incluirán en la ventana.
	• Expresión final: El número de filas después de la fila actual que se incluirán en la ventana.
	Por ejemplo, si desea incluir las 3 filas anteriores, la fila actual y la siguiente fila:
	`Window(concat(Text(Salary),'-'), Department, 'ASC', Age, Year>0,  -3, 1)`
	` as WSalaryDepartment`
	Para indicar todas las filas anteriores o todas las filas posteriores, puede utilizar la función **Unbounded()**. Por ejemplo, para incluir todas las filas anteriores, la fila actual y la fila siguiente:
	`Window(concat(Text(Salary),'-'), Department, 'ASC', Age, Year>0,  UNBOUNDED(), 1)`
	` as WSlidingSalaryDepartment`
	Por ejemplo, para incluir la tercera fila de la fila actual y todas las filas posteriores:
	`Window(concat(Text(Salary),'-'), Department, 'ASC', Age, Year>0,  3, UNBOUNDED())`
	` as WSlidingSalaryDepartment`

Ejemplo: agregar un campo que contiene una agregación

Ejemplo: Agregar un campo que contiene una agregación

Script de carga

Cree una nueva pestaña en el editor de carga de datos y luego cargue los siguientes datos como una carga inline. Cree la tabla siguiente en Qlik Sense para ver los resultados.

```
Transactions:
Load
*,
Window(Avg(transaction_amount),customer_id) as AvgCustTransaction;

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
];
```

Resultados

Resultados al agregar un campo que contiene una agregación

id_transacción	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_code	AvgCustTransaction
3750	20180830	23.56	2	2038593	L	Red	103.43
3751	20180907	556.31	6	203521	M	Orange	266.775
3752	20180916	5.75	1	5646471	S	Blue	42.935
3753	20180922	125.00	7	3036491	L	Black	64.21
3754	20180922	484.21	13	049681	XS	Red	273.88
3756	20180922	59.18	2	2038593	M	Blue	103.43
3757	20180923	177.42	21	203521	XL	Black	266.775
3758	20180924	153.42	14	2038593	L	Red	103.43
3759	20180925	7.42	5	203521	M	Orange	266.775
3760	20180925	80.12	18	5646471	M	Blue	42.935
3761	20180926	3.42	7	3036491	XS	Black	64.21
3763	20180926	63.55	12	049681	S	Red	273.88
3763	20180927	177.56	10	2038593	L	Blue	103.43
3764	20180927	325.95	8	203521	XL	Black	266.775

Ejemplo: agregar un campo que contiene una agregación con determinados valores filtrados

Ejemplo: Agregar un campo que contiene una agregación filtrada para valores específicos

Script de carga

Cree una nueva pestaña en el editor de carga de datos y luego cargue los siguientes datos como una carga inline. Cree la tabla siguiente en Qlik Sense para ver los resultados.

```
Transactions:
Load
*,
Window(Avg(transaction_amount),customer_id, color_code = 'Blue') as AvgCustTransaction;

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
];
```

Resultados

Resultados al agregar un campo que contiene una agregación filtrada para valores específicos

id_transacción	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_code	AvgCustTransaction
3750	20180830	23.56	2	2038593	L	Red	-
3751	20180907	556.31	6	203521	M	Orange	-
3752	20180916	5.75	1	5646471	S	Blue	42.94
3753	20180922	125.00	7	3036491	L	Black	-
3754	20180922	484.21	13	049681	XS	Red	-
3756	20180922	59.18	2	2038593	M	Blue	118.4
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	42.94
3761	20180926	3.42	7	3036491	XS	Black	-
3763	20180926	63.55	12	049681	S	Red	-

id_ transacción	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_ code	AvgCustTransaction
3763	20180927	177.56	10	2038593	L	Blue	118.4
3764	20180927	325.95	8	203521	XL	Black	-

Ejemplo: agregar un campo con una ventana deslizante

Ejemplo: Agregar un campo con una ventana deslizante

Script de carga

Cree una nueva pestaña en el editor de carga de datos y luego cargue los siguientes datos como una carga inline. Cree la tabla siguiente en Qlik Sense para ver los resultados.

```
Transactions:
Load
*,
Window(Avg(transaction_amount),customer_id, 'ASC', -1, 1, 0, 1) as AvgCustTransaction;

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
];
```

Resultados

Resultados al agregar un campo que contiene una agregación filtrada para valores específicos

id_ transacción	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_ code	AvgCustTransaction
3750	20180830	23.56	2	2038593	L	Red	41.37

id_ transacción	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_ code	AvgCustTransaction
3751	20180907	556.31	6	203521	M	Orange	366.865
3752	20180916	5.75	1	5646471	S	Blue	42.935
3753	20180922	125.00	7	3036491	L	Black	64.21
3754	20180922	484.21	13	049681	XS	Red	273.88
3756	20180922	59.18	2	2038593	M	Blue	106.3
3757	20180923	177.42	21	203521	XL	Black	92.42
3758	20180924	153.42	14	2038593	L	Red	165.49
3759	20180925	7.42	5	203521	M	Orange	166.685
3760	20180925	80.12	18	5646471	M	Blue	80.12
3761	20180926	3.42	7	3036491	XS	Black	3.42
3763	20180926	63.55	12	049681	S	Red	177.56
3763	20180927	177.56	10	2038593	L	Blue	63.55
3764	20180927	325.95	8	203521	XL	Black	325.95

Limitaciones

Window tiene las siguientes limitaciones:

- **Window** es una función que gasta muchos recursos, especialmente en términos de consumo de memoria.
- **Window** no es compatible con Qlik Sense Mobile.
- Las expresiones de gráfico no son compatibles con **Window**.
- No puede anidar funciones **Window** dentro de otras funciones **Window**.
- **Window** no se puede utilizar dentro de una función de agregación.
- **Window** necesita poder escanear toda la tabla.
- **WRank()**, **RecNo()** y **RowNo()** no se pueden usar con **Window** cuando se utiliza la función de ventana deslizante.

WRank - función de script

WRank() evalúa las filas de una tabla en el script de carga y, para cada fila, muestra la posición relativa del valor del campo evaluado en el script de carga. Cuando se evalúa la tabla, esta función compara el resultado con el resultado de las otras filas que contienen el segmento de columna actual y devuelve la clasificación de la fila actual dentro del segmento.

Particiones de una tabla

WRank solo se puede utilizar en una función **Window**. La función **Window** debe incluir un tipo de clasificación y una expresión de orden. El ranking o clasificación se aplica en la expresión de orden.

Sintaxis:

```
WRank ([mode[, fmt]])
```

Tipo de datos que devuelve: dual

Argumentos:

Argumentos

Argumento	Descripción
mode	Opionalmente, especifica la representación numérica del resultado de la función.
fmt	Opionalmente, especifica la representación textual del resultado de la función.
TOTAL	Si la tabla es unidimensional o si el script va precedido por el calificador **TOTAL**, la función se evalúa a lo largo de toda la columna. Si la tabla o el equivalente de tabla tiene múltiples dimensiones verticales, la partición actual incluirá únicamente las filas que tengan los mismos valores que la fila actual en todas las columnas de dimensión, excepto la columna que muestra la última dimensión en el orden de campos interno.

El número de clasificación o ranking se devuelve en la forma de un valor dual, el cual, en el caso de que cada fila tenga una clasificación única, será un entero entre 1 y el número de filas de la partición actual.

En caso de que varias filas compartan una misma clasificación o ranking, el texto y la representación numérica se pueden controlar mediante los parámetros **mode** y **fmt**.

mode

El primer argumento, **mode**, puede tomar los siguientes valores:

Valores **mode**

Valor	Descripción
0 (por defecto)	Si todos los rangos dentro del grupo compartido caen dentro del extremo más bajo del valor de en medio del ranking entero, todas las filas reciben el rango menor dentro del grupo compartido. Si todos los rangos dentro del grupo compartido caen dentro del extremo más alto del valor de en medio del ranking entero, todas las filas reciben el rango menor dentro del grupo compartido. Si todos los rangos dentro del grupo compartido se salen del valor central del ranking completo, todas las filas reciben el valor correspondiente a la media del ranking superior e inferior en la partición completa.
1	Rango más bajo en todas las filas.
2	Rango promedio en todas las filas.
3	Rango más alto en todas las filas.
4	Rango más bajo en la primera fila, incrementado por uno para cada fila.

fmt

El segundo argumento, **fmt**, puede tomar los siguientes valores:

Valores **fmt**

Valor	Descripción
0 (por defecto)	Valor bajo - valor alto en todas las filas (por ejemplo 3 - 4).
1	Valor bajo en todas las filas.
2	Valor más bajo en la primera fila, espacio en blanco en las siguientes filas.

El orden de las filas para **mode** 4 y **fmt** 2 viene determinado por el orden de carga de los campos de la tabla.

Ejemplo: agregar un campo clasificado

Ejemplo: Agregar un campo clasificado

Script de carga

Cree una nueva pestaña en el editor de carga de datos y luego cargue los siguientes datos como una carga inline. Cree la tabla siguiente en Qlik Sense para ver los resultados.

```
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
];
```

Resultados

Resultados para la agregación de un campo clasificado

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_code	TransactionRanking
3750	20180830	23.56	2	2038593	L	Rojo	4-4
3751	20180907	556.31	6	203521	M	Orange	1-1
3752	20180916	5.75	1	5646471	S	Blue	2-2
3754	20180922	484.21	13	049681	XS	Rojo	1-1
3756	20180922	59.18	2	2038593	M	Blue	3-3
3753	20180922	125.00	7	3036491	L	Black	1-1
3757	20180923	177.42	21	203521	XL	Black	3-3
3758	20180924	153.42	14	2038593	L	Rojo	2-2
3759	20180925	7.42	5	203521	M	Orange	4-4
3760	20180925	80.12	18	5646471	M	Blue	1-1
3763	20180926	63.55	12	049681	S	Rojo	2-2
3761	20180926	3.42	7	3036491	XS	Black	2-2
3764	20180927	325.95	8	203521	XL	Black	2-2
3763	20180927	177.56	10	2038593	L	Blue	1-1

Ejemplo: agregar un campo clasificado usando fmt para tener un resultado de un solo dígito

Ejemplo: Agregar un campo clasificado usando fmt para tener un resultado de un solo dígito

Script de carga

Cree una nueva pestaña en el editor de carga de datos y luego cargue los siguientes datos como una carga inline. Cree la tabla siguiente en Qlik Sense para ver los resultados.

```
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
];
```

Resultados

Resultados para agregar un campo clasificado usando fmt para tener un resultado de un solo dígito

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_code	TransactionRanking
3750	20180830	23.56	2	2038593	L	Rojo	4
3751	20180907	556.31	6	203521	M	Orange	1
3752	20180916	5.75	1	5646471	S	Blue	2
3754	20180922	484.21	13	049681	XS	Rojo	1
3756	20180922	59.18	2	2038593	M	Blue	3
3753	20180922	125.00	7	3036491	L	Black	1
3757	20180923	177.42	21	203521	XL	Black	3

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_code	TransactionRanking
3758	20180924	153.42	14	2038593	L	Rojo	2
3759	20180925	7.42	5	203521	M	Orange	4
3760	20180925	80.12	18	5646471	M	Blue	1
3763	20180926	63.55	12	049681	S	Rojo	2
3761	20180926	3.42	7	3036491	XS	Black	2
3764	20180927	325.95	8	203521	XL	Black	2
3763	20180927	177.56	10	2038593	L	Blue	1

Ejemplo: agregar un campo clasificado con múltiples particiones

Ejemplo: Agregar un campo clasificado con múltiples particiones

Script de carga

Cree una nueva pestaña en el editor de carga de datos y luego cargue los siguientes datos como una carga inline. Cree la tabla siguiente en Qlik Sense para ver los resultados.

```
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
];
```

Resultados

Resultados para agregar un campo clasificado usando fmt para tener un resultado de un solo dígito

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_code	TransactionRanking
3750	20180830	23.56	2	2038593	L	Rojo	2
3751	20180907	556.31	6	203521	M	Orange	1
3752	20180916	5.75	1	5646471	S	Blue	1
3754	20180922	484.21	13	049681	XS	Rojo	1
3756	20180922	59.18	2	2038593	M	Blue	1
3753	20180922	125.00	7	3036491	L	Black	1
3757	20180923	177.42	21	203521	XL	Black	2
3758	20180924	153.42	14	2038593	L	Rojo	1
3759	20180925	7.42	5	203521	M	Orange	2
3760	20180925	80.12	18	5646471	M	Blue	1
3763	20180926	63.55	12	049681	S	Rojo	1
3761	20180926	3.42	7	3036491	XS	Black	1
3764	20180927	325.95	8	203521	XL	Black	1
3763	20180927	177.56	10	2038593	L	Blue	1

Limitaciones

WRank tiene las siguientes limitaciones:

- Si su valor fmt es 0 y desea utilizar la parte de texto del resultado dual para **WRank**, debe utilizar **Text ()** con **Window(WRank)**. Por ejemplo: `Text(Window(WRank(0), Unit, 'DESC', Age)) as UnitWRankedByAgeText`.

9 Restricción de acceso al sistema de archivos

Por razones de seguridad, Qlik Sense en modo estándar no admite rutas en el script de carga de datos ni funciones o variables que expongan el sistema de archivos.

Sin embargo, dado que se admitían las rutas del sistema de archivos en QlikView, es posible deshabilitar el modo estándar y utilizar el modo de legado para poder reutilizar los scripts de carga de QlikView.

Deshabilitar el modo estándar puede crear un riesgo de seguridad al exponer el sistema de archivos.

Deshabilitar el modo estándar (page 1533)

9.1 Aspectos de seguridad relativos a la conexión con conexiones de datos ODBC y OLE DB basadas en archivos

Las conexiones de datos ODBC y OLE DB que utilizan controladores basados en archivos expondrán la ruta al archivo de datos conectado en la cadena de conexión. La ruta se puede exponer cuando se edite la conexión, en el cuadro de diálogo de selección de datos o en determinadas consultas SQL. Este es el caso tanto en el modo estándar como en el modo de legado.

Si exponer la ruta al archivo de datos supone un problema, se recomienda conectarse al archivo de datos empleando una conexión de datos de carpeta, si fuera posible.

9.2 Limitaciones en el modo estándar

Algunas sentencias, variables y funciones no se pueden utilizar o presentan limitaciones en el modo estándar. Utilizar sentencias no admitidas en el script de carga de datos produce un error cuando el script de carga se ejecuta. Pueden surgir mensajes de error en el archivo de registro del script. Utilizar variables y funciones no admitidas no produce mensajes de error o entradas en el archivo de registro, sino que la función devuelve NULL.

No hay indicación alguna de que una variable, sentencia o función no se admita cuando estemos editando el script de carga de datos.

Variables de sistema

Variables de sistema

Variable	Modo estándar	Modo de legado	Definición
Floppy	No admitido	Admitido	Devuelve la letra de la primera unidad de disco que encuentra, normalmente *a:*.

Variable	Modo estándar	Modo de legado	Definición
CD	No admitido	Admitido	Devuelve la letra de la primera unidad de CD-ROM que encuentre. Si no encuentra ningún CD-ROM, devuelve *c:*.
QvPath	No admitido	Admitido	Devuelve la cadena de búsqueda al ejecutable de Qlik Sense:
QvRoot	No admitido	Admitido	Devuelve el directorio raíz del ejecutable de Qlik Sense:
QvWorkPath	No admitido	Admitido	Devuelve la cadena de búsqueda a la app actual de Qlik Sense.
QvWorkRoot	No admitido	Admitido	Devuelve el directorio raíz de la app actual de Qlik Sense.
WinPath	No admitido	Admitido	Devuelve la cadena de exploración a Windows.
WinRoot	No admitido	Admitido	Devuelve el directorio raíz de Windows.
$(include=...)	Entrada compatible: Ruta que utiliza la conexión de la biblioteca	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos	La variable **Include/Must_Include** especifica un archivo que contiene texto que debe incluirse en el script y evaluarse como código de script. No se utiliza para añadir datos. Puede almacenar partes de su código de script en un archivo de texto aparte y reutilizarlo en diversas apps. Esta es una variable definida por el usuario.

Sentencias de script habituales

Sentencias de script habituales

Sentencia	Modo estándar	Modo de legado	Definición
Binary	Entrada compatible: Ruta que utiliza la conexión de la biblioteca	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos	La sentencia **binary** se usa para cargar datos desde otra app.
Connect	Entrada compatible: Ruta que utiliza la conexión de la biblioteca	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos	La sentencia **CONNECT** se utiliza para definir el acceso de Qlik Sense a una base de datos general mediante la interfaz OLE DB/ODBC. Para ODBC, primero se debe especificar la fuente de datos utilizando el administrador ODBC.
Directory	Entrada compatible: Ruta que utiliza la conexión de la biblioteca	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos	La sentencia **Directory** define qué directorio buscar en los archivos de datos en sentencias **LOAD** posteriores, hasta que se haga una nueva sentencia **Directory**.
Execute	No admitido	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos	La sentencia **Execute** se utiliza para ejecutar otros programas a la vez que Qlik Sense está cargando datos. Por ejemplo, para hacer las conversiones que sean necesarias.

Sentencia	Modo estándar	Modo de legado	Definición
LOAD from ...	Entrada compatible: Ruta que utiliza la conexión de la biblioteca	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos	La sentencia **LOAD** carga campos desde un archivo, desde datos definidos en el script, desde una tabla previamente cargada, desde una página web, desde el resultado de una sentencia **SELECT** posterior, o generando los datos automáticamente.
Store into ...	Entrada compatible: Ruta que utiliza la conexión de la biblioteca	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos	La sentencia **Store** crea un archivo QVD, Parquet, CSV o TXT.

Sentencias de control de script

Sentencias de control de script

Sentencia	Modo estándar	Modo de legado	Definición
For each... filelist mask/dirlist mask	Entrada compatible: Ruta que utiliza la conexión de la biblioteca Resultado obtenido: Conexión de biblioteca	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos Resultado obtenido: Conexión de la biblioteca o el sistema de archivos, dependiendo de lo introducido	La sintaxis filelist mask produce una lista separada por comas de todos los archivos del directorio actual que coincidan con **filelist mask**. La sintaxis **dirlist mask** produce una lista separada por comas de todos los directorios del directorio actual que coincidan con la máscara del nombre del directorio.

Funciones de archivo

Funciones de archivo

Función	Modo estándar	Modo de legado	Definición
Attribute()	Entrada compatible: Ruta que utiliza la conexión de la biblioteca	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos	Devuelve el valor de las meta etiquetas de diversos archivos de medios como texto.
ConnectString()	Resultado obtenido: Nombre de la conexión de librería	Nombre de la conexión de librería o conexión actual, dependiendo de la entrada	Devuelve la cadena de conexión activa de conexiones ODBC o OLE DB.
FileDir()	Resultado obtenido: Conexión de biblioteca	Resultado obtenido: Conexión de la biblioteca o el sistema de archivos, dependiendo de lo introducido	La función **FileDir** devuelve una cadena que contiene la ruta al directorio del archivo de tabla que se está leyendo en ese momento.
FilePath()	Resultado obtenido: Conexión de biblioteca	Resultado obtenido: Conexión de la biblioteca o el sistema de archivos, dependiendo de lo introducido	La función **FilePath** devuelve una cadena que contiene la ruta completa al archivo de tabla que se esté leyendo en ese momento.
FileSize()	Entrada compatible: Ruta que utiliza la conexión de la biblioteca	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos	La función **FileSize** devuelve un entero que contiene el tamaño en bytes del archivo filename o, si no se especifica ningún filename, del archivo de tabla que se esté leyendo en ese momento.

Función	Modo estándar	Modo de legado	Definición
FileTime()	Entrada compatible: Ruta que utiliza la conexión de la biblioteca	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos	La función **FileTime** devuelve una marca de tiempo en formato UTC de la última modificación de un archivo especificado. Si no se especifica un archivo, la función devuelve una marca de tiempo en UTC de la última modificación del archivo de tabla leído actualmente.
GetFolderPath()	No admitido	Resultado obtenido: Ruta absoluta	La función **GetFolderPath** devuelve el valor de la función Microsoft Windows *SHGetFolderPath*. Esta función toma como entrada el nombre de una carpeta de Microsoft Windows y devuelve la ruta completa de la carpeta.
QvdCreateTime()	Entrada compatible: Ruta que utiliza la conexión de la biblioteca	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos	Esta función de script devuelve la marca de tiempo del encabezado XML de un archivo QVD, si la hay, de lo contrario devuelve NULL. En la marca de tiempo, la hora se proporciona en UTC.
QvdFieldName()	Entrada compatible: Ruta que utiliza la conexión de la biblioteca	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos	Esta función de script devuelve el nombre del número de campo **fieldno** en un archivo QVD. Si el campo no existe, devuelve NULL.

Función	Modo estándar	Modo de legado	Definición
QvdNoOfFields()	Entrada compatible: Ruta que utiliza la conexión de la biblioteca	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos	Esta función de script devuelve el número de campos de un archivo QVD.
QvdNoOfRecords()	Entrada compatible: Ruta que utiliza la conexión de la biblioteca	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos	Esta función de script devuelve el número de registros que hay actualmente en un archivo QVD.
QvdTableName()	Entrada compatible: Ruta que utiliza la conexión de la biblioteca	Entrada compatible: Ruta que utiliza la conexión de la biblioteca o el sistema de archivos	Esta función de script devuelve el nombre de la tabla almacenada en un archivo QVD.

Funciones de sistema

Funciones de sistema

Función	Modo estándar	Modo de legado	Definición
DocumentPath()	No admitido	Resultado obtenido: Ruta absoluta	Esta función devuelve una cadena que contiene la ruta completa a la app actual de Qlik Sense.
GetRegistryString()	No admitido	Admitido	Devuelve el valor de una clave de registro nombrada, con una ruta de registro determinada. Esta función puede emplearse en los gráficos y en el script por igual.

9.3 Deshabilitar el modo estándar

Podemos deshabilitar el modo estándar, o en otras palabras, establecer un modo de legado, para poder reutilizar scripts de carga de QlikView que se refieren a rutas de archivo absolutas o relativas así como conexiones de librería.

Deshabilitar el modo estándar puede crear un riesgo de seguridad al exponer el sistema de archivos.

Qlik Sense

Para Qlik Sense, el modo estándar se puede deshabilitar en QMC utilizando la propiedad de **Modo estándar**.

Qlik Sense Desktop

En Qlik Sense Desktop, puede establecer el modo estándar/de legado en *Settings.ini*.

Si instaló Qlik Sense Desktop usando la ubicación de instalación predeterminada, *Settings.ini* se ubica en *C:\Users\{user}\Documents\Qlik\Sense\Settings.ini*. Si instaló Qlik Sense Desktop en una carpeta que seleccionó, *Settings.ini* se ubica en la carpeta *Engine* de la ruta de instalación.

Haga lo siguiente:

1. Abra *Settings.ini* en un editor de texto.
2. Cambie *StandardReload=1* a *StandardReload=0*.
3. Guarde el archivo e inicie Qlik Sense Desktop.

Qlik Sense Desktop se ejecuta ahora en modo de legado.

Configuración

Las configuraciones disponibles para la recarga estándar son:

- 1 (modo estándar)
- 0 (modo de legado)

10 Secuencias de script a nivel de gráfico

Al modificar los datos del gráfico, utiliza un subconjunto del script de Qlik Sense que consta de una serie de sentencias. Una sentencia puede ser de dos tipos, una sentencia normal de script o una sentencia de control de script. Ciertas sentencias pueden ir precedidas de prefijos.

Las sentencias más comunes se utilizan habitualmente para manipular datos de varias formas. Estas sentencias pueden escribirse sobre cualquier número de filas en el script y deben terminar siempre en punto y coma ";".

Las sentencias de control en cambio se suelen emplear para controlar el flujo de ejecución del script. Hay que mantener cada cláusula de una sentencia de control dentro de una línea en el script. Estas cláusulas pueden terminar en punto y coma, o en un final de línea.

La aplicación de prefijos es posible con sentencias habituales, pero nunca con las sentencias de control.

Todas las palabras clave del script pueden escribirse con cualquier combinación de caracteres en mayúscula o minúscula. Los nombres de campo y de variable utilizados en las sentencias, por supuesto, son sensibles a mayúsculas.

En esta sección encontrará una lista alfabética de todas las sentencias de script, sentencias de control y prefijos disponibles en el subconjunto de las secuencias de script utilizadas al modificar los datos del gráfico.

10.1 Sentencias de control

Al modificar los datos del gráfico, utiliza un subconjunto del script de Qlik Sense que consta de una serie de sentencias. Una sentencia puede ser de dos tipos, una sentencia normal de script o una sentencia de control de script.

Las sentencias de control en cambio se suelen emplear para controlar el flujo de ejecución del script. Cada cláusula de una sentencia de control debe hallarse dentro de una línea de script y puede acabar en punto y coma o un final de línea.

Los prefijos nunca se aplican a las sentencias de control.

Todas las palabras clave del script pueden escribirse con cualquier combinación de caracteres en mayúscula o minúscula.

Descripción general de las sentencias de control del modificador de gráfico

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Call

La sentencia de control **call** invoca una subrutina que debe ir definida por una sentencia **sub** anterior.

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

Do..loop

La sentencia de control **do..loop** es una construcción de iteración de script que ejecuta una o varias sentencias hasta que se cumple una condición lógica.

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

End

La palabra clave de script **End** se utiliza para concluir cláusulas **If**, **Sub** y **Switch**.

Exit

La palabra clave de script **Exit** forma parte de la sentencia **Exit Script**, pero también se puede usar para salir de cláusulas **Do**, **For** o **Sub**.

Exit script

Esta sentencia de control detiene la ejecución del script. Puede insertarse en cualquier parte del script.

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

For..next

La sentencia de control **for..next** es una construcción de iteración de script con un contador. Las sentencias dentro del bucle incluidas entre **for** y **next** se ejecutarán para cada valor de la variable de contador entre los límites alto y bajo especificados.

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

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

For each ..next

La sentencia de control **for each..next** es una construcción de iteración de script que ejecuta una o varias sentencias para cada valor en una lista separada por comas. Las sentencias dentro del bucle incluidas entre **for** y **next** se ejecutarán para cada valor de la lista.

```
For each..next var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

If..then

La sentencia de control **if..then** es una construcción de selección de script que obliga a la ejecución del script a seguir diferentes rutas dependiendo de una o varias condiciones lógicas.

> *Dado que la sentencia **if..then** es una sentencia de control y como tal finaliza con un punto y coma o un final de línea, cada una de sus cuatro cláusulas posibles (**if..then**, **elseif..then**, **else** y **end if**) no debe superar el límite de una línea.*

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

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

Next

La palabra clave de script **Next** se utiliza para cerrar bucles **For**.

Sub

La sentencia de control **sub..end sub** define una subrutina que puede invocarse desde una sentencia **call**.

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

Switch

La sentencia de control **switch** es una construcción de selección de script que obliga a la ejecución de script a seguir diferentes rutas dependiendo del valor de una expresión.

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

To

La palabra clave de script **To** se utiliza en diversas sentencias de script.

Call

La sentencia de control **call** invoca una subrutina que debe ir definida por una sentencia **sub** anterior.

Sintaxis:

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

Argumentos:

Argumentos

Argumento	Descripción
name	El nombre de la subrutina.
paramlist	Una lista separada por comas de los parámetros que se habrán de enviar a la subrutina. Cada elemento de la lista puede ser un nombre de campo, una variable o una expresión arbitraria.

La subrutina llamada por una sentencia **call** debe definirse mediante una sentencia **sub** que se encuentre anteriormente durante la ejecución del script.

Los parámetros se copian en la subrutina y, si el parámetro en la sentencia **call** es una variable y no una expresión, se copia nuevamente al salir de la subrutina.

Limitaciones:

- Puesto que la sentencia **call** es una sentencia de control y, como tal, finaliza con un punto y coma o un final de línea, no debe superar el límite de una línea.
- Cuando define una subrutina con `Sub..end sub` dentro de una sentencia de control, por ejemplo `if..then`, solo puede llamar a la subrutina desde dentro de esa misma sentencia de control.

Do..loop

La sentencia de control **do..loop** es una construcción de iteración de script que ejecuta una o varias sentencias hasta que se cumple una condición lógica.

Sintaxis:

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

*Dado que la sentencia **do..loop** es una sentencia de control y como tal finaliza con un punto y coma o un final de línea, cada una de sus tres posibles cláusulas (**do**, **exit do** y **loop**) no debe superar el límite de una línea.*

Argumentos:

Argumentos

Argumento	Descripción
condition	Una expresión lógica que devuelve True o False.

Argumento	Descripción
statements	Es cualquier grupo de una o varias sentencias de script de Qlik Sense.
while / until	La cláusula condicional **while** o **until** solo debe aparecer una vez en cualquier sentencia **do..loop**, es decir, o bien después de **do** o después de **loop**. Cada condición se interpreta sólo la primera vez que se encuentra pero se evalúa cada vez que se encuentra en el bucle.
exit do	Si se encuentra una cláusula **exit do** dentro del bucle, la ejecución del script se transferirá a la primera sentencia después de la cláusula **loop** que denota el final del bucle. Una cláusula **exit do** puede volverse condicional mediante el uso opcional de un sufijo **when** o **unless**.

End

La palabra clave de script **End** se utiliza para concluir cláusulas **If**, **Sub** y **Switch**.

Exit

La palabra clave de script **Exit** forma parte de la sentencia **Exit Script**, pero también se puede usar para salir de cláusulas **Do**, **For** o **Sub**.

Exit script

Esta sentencia de control detiene la ejecución del script. Puede insertarse en cualquier parte del script.

Sintaxis:

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

Puesto que la sentencia **exit script** es una sentencia de control y, como tal, finaliza con un punto y coma o un final de línea, no debe superar el límite de una línea.

Argumentos:

Argumentos

Argumento	Descripción
condition	Una expresión lógica que devuelve True o False.
when / unless	Una sentencia **exit script** puede volverse condicional mediante el uso opcional de **when** o una cláusula **unless**.

Ejemplos:

```
//Exit script
Exit Script;
```

```
//Exit script when a condition is fulfilled
Exit Script when a=1
```

For..next

La sentencia de control **for..next** es una construcción de iteración de script con un contador. Las sentencias dentro del bucle incluidas entre **for** y **next** se ejecutarán para cada valor de la variable de contador entre los límites alto y bajo especificados.

Sintaxis:

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

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

Las expresiones *expr1*, *expr2* y *expr3* solo se evalúan la primera vez que se entra en el bucle. El valor de la variable contador puede ser modificado por sentencias dentro del bucle, pero no es una buena práctica de programación.

Si se encuentra una cláusula **exit for** dentro del bucle, la ejecución del script se transferirá a la primera sentencia después de la cláusula **next** que denota el final del bucle. Una cláusula **exit for** puede volverse condicional mediante el uso opcional de un sufijo **when** o **unless**.

> *Dado que la sentencia **for..next** es una sentencia de control y como tal finaliza con un punto y coma o un final de línea, cada una de sus tres posibles cláusulas (**for..to..step**, **exit for** y **next**) no debe superar el límite de una línea.*

Argumentos:

Argumentos

Argumento	Descripción
counter	Es un nombre de variable. Si se especifica *counter* después de **next** debe ser el mismo nombre de variable que el que se encuentra después del correspondiente **for**.
expr1	Una expresión que determina el primer valor de la variable *counter* para la que se debe ejecutar el bucle.
expr2	Una expresión que determina el último valor de la variable *counter* para el que se debe ejecutar el bucle.

Argumento	Descripción
expr3	Una expresión que determina el valor que indica el incremento de la variable *counter* cada vez que se ha ejecutado el bucle.
condition	una expresión lógica que devuelve True o False.
statements	Es cualquier grupo de una o varias sentencias de script de Qlik Sense.

For each..next

La sentencia de control **for each..next** es una construcción de iteración de script que ejecuta una o varias sentencias para cada valor en una lista separada por comas. Las sentencias dentro del bucle incluidas entre **for** y **next** se ejecutarán para cada valor de la lista.

Sintaxis:
La sintaxis especial hace posible generar listas con los nombres de archivo y directorio en el directorio actual.

```
for each var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

Argumentos:

Argumentos

Argumento	Descripción
var	Es un nombre de variable de script que adquirirá un nuevo valor de lista para cada ejecución del bucle. Si se especifica **var** después de **next** debe ser el mismo nombre de variable que el que se encuentra después del correspondiente **for each**.

El valor de la variable **var** se puede modificar mediante sentencias dentro del bucle, pero no es una buena práctica de programación.

Si se encuentra una cláusula **exit for** dentro del bucle, la ejecución del script se transferirá a la primera sentencia después de la cláusula **next** que denota el final del bucle. Una cláusula **exit for** puede volverse condicional mediante el uso opcional de un sufijo **when** o **unless**.

*Dado que la sentencia **for each..next** es una sentencia de control y como tal finaliza con un punto y coma o un final de línea, cada una de sus tres posibles cláusulas (**for each**, **exit for** y **next**) no debe superar el límite de una línea.*

Sintaxis:

```
list := item { , item }
```

```
item := constant | (expression) | filelist mask | dirlist mask |
fieldvaluelist mask
```

Argumentos

Argumento	Descripción
constant	Es cualquier número o cadena. Obsérvese que una cadena introducida directamente en el script debe ir entre comillas simples. Una cadena sin entrecomillado simple se interpretará como una variable y entonces se utilizará el valor de dicha variable. Los números no tienen que ir entre comillas simples.
expression	Es una expresión cualquiera.
mask	Una máscara de nombre de archivo o carpeta que puede incluir cualquier carácter válido de nombre de archivo, así como los caracteres comodín estándar, ***** y **?**. Puede utilizar rutas de archivos absolutas o lib://.
condition	Una expresión lógica que devuelve True o False.
statements	Es cualquier grupo de una o varias sentencias de script de Qlik Sense.
filelist mask	Esta sintaxis produce una lista de todos los archivos incluidos en el directorio actual, separados por coma, que coincidan con la máscara de nombre de archivo. *Este argumento admite únicamente conexiones de biblioteca en modo estándar.*
dirlist mask	Esta sintaxis produce una lista con todas las carpetas de la carpeta actual (separadas por comas) que coincidan con la máscara de nombre de archivo. *Este argumento admite únicamente conexiones de biblioteca en modo estándar.*
fieldvaluelist mask	Esta sintaxis se repite a lo largo de los valores de un campo ya cargado en Qlik Sense.

*El Qlik Conectores de proveedores de almacenamiento web y otras conexiones de DataFiles no admiten máscaras de filtro que usen los caracteres comodín (***** y **?**).*

Example 1: Cargar una lista de archivos

```
// 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: Crear una lista de archivos en el disco

Este ejemplo carga una lista de todos los campos Qlik Sense relacionados en una carpeta.

```
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: Se repite a lo largo de los valores de un campo

Este ejemplo recorre toda la lista de valores cargados de FIELD y genera un nuevo campo, NEWFIELD. Por cada valor de FIELD, se crearán dos NEWFIELD registros.

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

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

La tabla resultante tiene el siguiente aspecto:

Example table

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

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

La sentencia de control **if..then** es una construcción de selección de script que obliga a la ejecución del script a seguir diferentes rutas dependiendo de una o varias condiciones lógicas.

Las sentencias de control se emplean habitualmente para controlar el flujo de ejecución del script. En una expresión de gráfico, utilice la función condicional **if** en su lugar.

Sintaxis:

```
If  condition then

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

Dado que la sentencia **if..then** es una sentencia de control y como tal finaliza con un punto y coma o un final de línea, cada una de sus cuatro cláusulas posibles (**if..then**, **elseif..then**, **else** y **end if**) no debe superar el límite de una línea.

Argumentos:

Argumentos

Argumento	Descripción
condition	Una expresión lógica que puede evaluarse como True o False.
statements	Es cualquier grupo de una o varias sentencias de script de Qlik Sense.

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

La palabra clave de script **Next** se utiliza para cerrar bucles **For**.

Sub..end sub

La sentencia de control **sub..end sub** define una subrutina que puede invocarse desde una sentencia **call**.

Sintaxis:

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

Los argumentos se copian en la subrutina y, si el parámetro real correspondiente en la sentencia **call** es un nombre de una variable, se copia nuevamente al salir de la subrutina.

Si una subrutina tiene parámetros más formales que los parámetros reales que pasan por una sentencia **call**, los parámetros adicionales se inicializarán en NULL y se podrán usar como variables locales dentro de la subrutina.

Argumentos:

Argumentos

Argumento	Descripción
name	El nombre de la subrutina.
paramlist	Una lista separada por comas con los nombres de variables de los parámetros formales de la subrutina. Estos pueden utilizarse como cualquier variable dentro de la subrutina.
statements	Es cualquier grupo de una o varias sentencias de script de Qlik Sense.

Limitaciones:

- Dado que la sentencia **sub** es una sentencia de control y como tal finaliza con un punto y coma o un final de línea, cada una de sus dos cláusulas (**sub** y **end sub**) no debe superar el límite de una línea.
- Cuando define una subrutina con `Sub..end sub` dentro de una sentencia de control, por ejemplo `if..then`, solo puede llamar a la subrutina desde dentro de esa misma sentencia de control.

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: - transferencia de parámetros

```
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)
```
El resultado de lo anterior será que, localmente, dentro de la subrutina, A se inicializará en 1, B se inicializará en 4 y C se inicializará en NULL.

Al salir de la subrutina, la variable global A obtendrá 2 como valor (copiado de la subrutina). El segundo parámetro real "(X+1)*2" no será copiado puesto que no es una variable. Por último, la variable global C no se verá afectada por la llamada de la subrutina.

Switch..case..default..end switch

La sentencia de control **switch** es una construcción de selección de script que obliga a la ejecución de script a seguir diferentes rutas dependiendo del valor de una expresión.

Sintaxis:

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

> *Dado que la sentencia **switch** es una sentencia de control y como tal finaliza con un punto y coma o un final de línea, cada una de sus cuatro cláusulas posibles (**switch**, **case**, **default** y **end switch**) no debe superar el límite de una línea.*

Argumentos:

Argumentos

Argumento	Descripción
expression	Es una expresión cualquiera.
valuelist	Una lista de valores separados por comas, con los que se compara el valor de expresión. La ejecución del script continuará con las sentencias del primer grupo que se haya hallado que contienen un valor en listavalores igual al valor de expresión. Cada valor de listavalores puede ser una expresión cualquiera. Si no se encuentra correspondencia en ninguna cláusula **case**, se ejecutarán las declaraciones bajo la cláusula **default**, si se especifica.
statements	Es cualquier grupo de una o varias sentencias de script de Qlik Sense.

Ejemplo:

```
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

La palabra clave de script **To** se utiliza en diversas sentencias de script.

10.2 Prefijos

La aplicación de prefijos es posible con sentencias habituales, pero nunca con las sentencias de control.

Todas las palabras clave del script pueden escribirse con cualquier combinación de caracteres en mayúscula o minúscula. Los nombres de campo y de variable utilizados en las sentencias, por supuesto, son sensibles a mayúsculas.

Descripción general de los prefijos modificadores de gráficos

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

Add
El prefijo **Add** se puede añadir a cualquier sentencia **LOAD** o **SELECT** en el script para especificar que debe agregar registros a otra tabla. También especifica que esta sentencia debe ejecutarse en una carga parcial. El prefijo **Add** también se puede usar en una sentencia **Map**.

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

Replace
El prefijo **Replace** se puede añadir a cualquier sentencia **LOAD** o **SELECT** en el script para especificar que la tabla cargada debe reemplazar a otra tabla. También especifica que esta sentencia debe ejecutarse en una carga parcial. El prefijo **Replace** también se puede usar en una sentencia **Map**.

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

Add

En un contexto de modificación de gráficos, el prefijo **Add** se utiliza con **LOAD** para agregar valores a la tabla *HC1*, que representa el hipercubo calculado por el Motor asociativo de Qlik. Puede especificar una o varias columnas. Los valores que faltan son llenados automáticamente por el motor Motor asociativo de Qlik.

Sintaxis:

```
Add loadstatement
```

Ejemplo:

Este ejemplo agrega dos filas a las columnas *Dates* y *Sales* de la sentencia inline

```
Add Load
x as Dates,
y as Sales
Inline
[
Dates,Sales
2001/09/1,1000
2001/09/10,-300
]
```

Replace

En un contexto de modificación de gráficos, el prefijo **Replace** cambia todos los valores de la tabla *HC1* con un valor calculado definido por el script.

Sintaxis:

```
Replace loadstatement
```

Ejemplo:

Este ejemplo sobrescribe todos los valores en la columna `z` con la suma de `x` y `y`.

```
Replace Load
x+y as z
Resident HC1;
```

10.3 Sentencias habituales

Las sentencias más comunes se utilizan habitualmente para manipular datos de varias formas. Estas sentencias pueden escribirse sobre cualquier número de filas en el script y deben terminar siempre en punto y coma ";".

Todas las palabras clave del script pueden escribirse con cualquier combinación de caracteres en mayúscula o minúscula. Los nombres de campo y de variable utilizados en las sentencias, por supuesto, son sensibles a mayúsculas.

Descripción general de las sentencias habituales modificadoras de gráficos

Cada función se define en detalle tras la vista general. También puede hacer clic en el nombre de la función en la sintaxis para acceder de inmediato a los detalles de dicha función.

LOAD

En un contexto de modificación de gráficos, la sentencia **LOAD** carga datos adicionales al hipercubo a partir de datos definidos en el script o desde una tabla previamente cargada. También se puede cargar datos desde conexiones analíticas.

*La sentencia **LOAD** debe llevar o bien el prefijo **Replace** o **Add**, o será rechazada.*

```
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
La sentencia **let** es un complemento a la sentencia **set**, utilizada para definir variables de script. La sentencia **let**, al contrario que la sentencia **set**, evalúa la expresión a la derecha del signo "=" en tiempo de ejecución de script antes de que se asigne a la variable.

```
Let variablename=expression
```

Set
La sentencia **set** se utiliza para definir variables de script. Éstas pueden servir para sustituir cadenas, rutas, unidades de disco, etc.

```
Set variablename=string
```

Put
La sentencia **Put** se utiliza para establecer algún valor numérico en el hipercubo.

HCValue
La sentencia **HCValue** se utiliza para recuperar valores en una fila de una columna especificada.

Load

En un contexto de modificación de gráficos, la sentencia LOAD carga datos adicionales al hipercubo a partir de datos definidos en el script o desde una tabla previamente cargada. También se puede cargar datos desde conexiones analíticas.

*La sentencia **LOAD** debe llevar o bien el prefijo **Replace** o **Add**, o será rechazada.*

Sintaxis:

```
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 ]
```

Argumentos:

Argumentos

Argumento	Descripción
fieldlist	*fieldlist ::= (* | field{**,** * | field })* Una lista de los campos que se van a cargar. Usar * como una lista de campos indica todos los campos de la tabla. *field ::= (fieldref | expression) [***as** *aliasname]* La definición de campo debe contener siempre una referencia literal a un campo existente, o a una expresión. *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [* ***I*** | ***U*** | ***R*** | ***B*** | **T**]) *fieldname* es un texto que es idéntico a un nombre de campo de la tabla. Tenga en cuenta que el nombre de campo debe ir entre comillas dobles rectas o corchetes si contiene por ejemplo espacios. A veces los nombres de campo no están disponibles de forma explícita. Entonces se usa una nomenclatura diferente: *@fieldnumber* representa el número de campo en un archivo de tabla delimitada. Debe ser un entero positivo precedido por "@". La numeración se hace siempre desde 1 hasta el número de campos. *@startpos*:*endpos* representa las posiciones de inicio y final de un campo en un archivo con registros de longitud fija. Las posiciones deben ser ambas números enteros positivos. Los dos números deben estar precedidos por "@" y separados por dos puntos. La numeración se hace siempre desde 1 hasta el número de posiciones. En el último campo, **n** se usa como posición final. • Si *@startpos*:*endpos* va seguido inmediatamente por los caracteres **I** o **U**, los bytes leídos se interpretarán como un binario firmado (**I**) o un entero no firmado (**U**) (orden de bytes de Intel). El número de posiciones leídas debe ser 1, 2 o 4. • Si *@startpos*:*endpos* va inmediatamente seguido por el carácter **R**, los bytes leídos se interpretarán como un número real binario (IEEE de 32 bits o coma flotante de 64 bits). El número de las posiciones leídas debe ser 4 u 8. • Si *@startpos*:*endpos* va inmediatamente seguido por el carácter **B**, los bytes leídos se interpretarán como números BCD (Binary Coded Decimal) según el estándar COMP-3. Se puede especificar cualquier número de bytes. *expression* puede ser una función numérica o una función de cadena basada en uno o varios campos más de la misma tabla. Para más información, vea la sintaxis de las expresiones. **as** se usa para asignar un nuevo nombre al campo.

Argumento	Descripción
inline	**inline** se usa si los datos se deben escribir dentro del script y no se cargan desde un archivo. *data ::= [text]* Los datos introducidos mediante una cláusula **inline** deben estar entre comillas dobles o entre corchetes. El texto en su interior se interpreta de la misma manera que el contenido de un archivo. Por lo tanto, cuando inserte una nueva línea en un archivo de texto, también debe hacerlo en el texto de una cláusula **inline**, es decir, pulsando la tecla Intro al escribir la secuencia de script. El número de columnas viene definido por la primera línea. *format-spec ::= (fspec-item {, fspec-item })* La especificación de formato consiste en una lista con varios elementos de especificación entre paréntesis. Para más información, vea *Elementos de especificación de formato (page 168)*.
resident	**resident** se usa si los datos se deben cargar desde una tabla previamente cargada. *table label* es una etiqueta que precede a la sentencia **LOAD** que creó la tabla original. La etiqueta debe ir seguida de dos puntos al final de la línea.

Argumento	Descripción
extension	Puede cargar datos desde conexiones analíticas. Necesita usar la cláusula **extension** para llamar a una función definida en el complemento plugin (SSE) de extensión del lado del servidor o evaluar un script. Puede enviar una única tabla al complemento SSE y devuelve una sola tabla de datos. Si el complemento plugin no especifica los nombres de los campos que se devuelven, los campos se denominarán Field1, Field2 y así sucesivamente. `Extension pluginname.functionname( tabledescription );` • Cargar datos utilizando una función en un complemento plugin SSE *tabledescription ::= (table { ,tablefield})* Si no indica campos de tabla, los campos se usarán en orden de carga. • Cargar los datos evaluando un script en un complemento plugin SSE *tabledescription ::= (script, table { ,tablefield})* **Manejo del tipo de datos en la definición del campo de la tabla** Los tipos de datos se detectan automáticamente en las conexiones analíticas. Si los datos no tienen valores numéricos y al menos una cadena de texto no nula, el campo se considera texto. En cualquier otro caso, se considera numérico. Puede forzar el tipo de datos encerrando un nombre de campo en **Cadena()** o **Mixto()**. • **Cadena()** obliga al campo a ser de texto. Si el campo es numérico, se extrae la parte de texto del valor dual, no se realiza ninguna conversión. • **Mixto()** obliga al campo a ser dual. **Cadena()** o **Mixto()** no se pueden usar fuera de las definiciones de campo de la tabla de **extensión** y no puede usar otras funciones Qlik Sense en una definición de campo de tabla.
where	**where** es una cláusula utilizada para indicar si un registro debe incluirse en la selección o no. La selección se incluye si *criterion* es True. *criterion* es una expresión lógica
while	**while** es una cláusula utilizada para indicar si un registro debe leerse repetidamente. Se lee el mismo registro siempre y cuando el *criterion* sea True. Para ser útil, una cláusula **while** debe incluir por lo general la función **IterNo( )**. *criterion* es una expresión lógica

Argumento	Descripción
group by	**group by** es una cláusula que sirve para definir sobre qué campos deben agregarse (agruparse) los datos. Los campos de agrupación deberán incluirse de alguna manera en las expresiones cargadas. Ningún otro campo más que los de agrupación deberá emplearse fuera de las funciones de agregación en las expresiones cargadas. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	**order by** es una cláusula utilizada para clasificar los registros de una tabla residente antes de que la sentencia **load** los procese. La tabla residente puede ordenarse por más de un campo en orden ascendente o descendente. La ordenación se hace principalmente por valores numéricos y secundariamente por valor de cotejo nacional. Esta cláusula solo puede utilizarse cuando al fuente de datos es una tabla residente. Los campos de ordenación especifican por qué campos está ordenada la tabla residente. El campo puede especificarse por su nombre o por su número en la tabla residente (el primer número de campo es el 1). orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } *sortorder* es o bien *asc* ascendente o *desc* descendente. Si no se especifica *sortorder*, se asume *asc*. *fieldname*, *path*, *filename* y *aliasname* son cadenas de texto que representan lo que sus respectivos nombres implican. Cualquier campo de la tabla fuente se puede usar como *fieldname*. Sin embargo, los campos creados a través de la cláusula (*aliasname*) están fuera del alcance y no se pueden usar dentro de la misma sentencia **load**.

Let

La sentencia **let** es un complemento a la sentencia **set**, utilizada para definir variables de script. La sentencia **let**, al contrario que la sentencia **set**, evalúa la expresión a la derecha del signo "=" en tiempo de ejecución de script antes de que se asigne a la variable.

Sintaxis:

```
Let variablename=expression
```

Ejemplos y resultados:

Ejemplo	Resultado

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	`$(x)` se evaluará como ' 3+4 '
	
	`$(y)` se evaluará como ' 7 '
	
	`$(z)` se evaluará como ' 8 '
	
	Observe la diferencia entre las sentencias **Set** y **Let**. La sentencia **Set** asigna la cadena '3+4' a la variable, mientras que la sentencia **Let** evalúa la cadena y asigna 7 a la variable.

Let T=now( );	`$(T)` recibirá el valor de la hora actual.

Set

La sentencia **set** se utiliza para definir variables de script. Éstas pueden servir para sustituir cadenas, rutas, unidades de disco, etc.

Sintaxis:

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Put

La sentencia **put** se utiliza para establecer algún valor numérico en el hipercubo.

El acceso a las columnas se puede realizar mediante etiquetas. También puede acceder a columnas y filas por orden de sentencia. Vea los ejemplos a continuación para obtener más detalles.

Sintaxis:

```
put column(position)=value
```

Example 1:

El acceso a las columnas se puede realizar mediante etiquetas.

Este ejemplo establecerá un valor de `1` en la primera posición de la columna denominada *Sales*.

```
Put Sales(1) = 1;
```

Example 2:

Puede acceder a las columnas de medida por orden de sentencia usando el formato `#hc1.measure` para medidas.

Este ejemplo devuelve el valor 1000 en la décima posición del hipercubo final ordenado.

```
Put #hc1.measure.2(10) = 1000;
```

Example 3:

Puede acceder a las filas de dimensión por orden de declaración usando el formato `#hc1.dimension` para dimensiones.

Este ejemplo pone el valor de la constante Pi en la quinta fila de la tercera dimensión declarada.

```
Put #hc1.dimension.3(5) = Pi();
```

Si no existen tales dimensiones o expresiones, en valor o etiquetas, devuelve un error que indica que no se encontró la columna. Si el índice de la columna está fuera de los límites, no devuelve un error.

HCValue

La función **HCValue** se utiliza para recuperar valores en una fila de una columna especificada.

Sintaxis:

```
HCValue(column,position)
```

Example 1:

Este ejemplo devuelve el valor en la primera posición de la columna con la etiqueta "Ventas".

```
HCValue(Sales,1)
```

Example 2:

Este ejemplo devuelve el valor en la décima posición del hipercubo ordenado.

```
HCValue(#hc1.measure2,10)
```

Example 3:

Este ejemplo devuelve el valor en la quinta fila de la tercera dimensión.

```
HCValue(#hc1.dimension.3,5)
```

Si no existen tales dimensiones o expresiones, en valor o etiquetas, devuelve un error que indica que no se encontró la columna. Si el índice de la columna está fuera de los límites, devuelve NULL.

11 Funciones y sentencias de QlikView no admitidas en Qlik Sense

La mayoría de las funciones y sentencias se pueden usar en scripts de carga de QlikView, y las expresiones de gráficos también son compatibles en Qlik Sense, aunque hay algunas excepciones, tal y como se describe aquí.

11.1 Sentencias de script no admitidas en Qlik Sense

QlikView Sentencias de script que no se admiten en Qlik Sense

Sentencia	Comentarios
Command	Use **SQL** en su lugar.
InputField	

11.2 Funciones no admitidas en Qlik Sense

Esta lista describe las sentencias de script de QlikView y las funciones de gráfico no admitidas en Qlik Sense.

- **GetCurrentField**
- **GetExtendedProperty**
- **Input**
- **InputAvg**
- **InputSum**
- **MsgBox**
- **NoOfReports**
- **ReportComment**
- **ReportId**
- **ReportName**
- **ReportNumber**

11.3 Prefijos no admitidos en Qlik Sense

Esta lista describe los prefijos de QlikView no admitidos en Qlik Sense.

- **Bundle**
- **Image_Size**
- **Info**

12 Funciones y sentencias no recomendadas en Qlik Sense

La mayoría de las funciones y sentencias que pueden utilizarse en los scripts de carga y las expresiones de gráficos de QlikView también se admiten en Qlik Sense, pero en algunas ocasiones su uso no se recomienda en Qlik Sense. También hay funciones y sentencias disponibles en versiones anteriores de Qlik Sense que han quedado en desuso.

Por razones de compatibilidad siguen funcionando de la forma prevista, pero se recomienda actualizar el código según las recomendaciones de esta sección, porque podrían eliminarse en versiones futuras.

12.1 Sentencias de script no recomendadas en Qlik Sense

Esta tabla contiene sentencias de script cuyo uso no se recomienda en Qlik Sense.

Sentencias de script que no se recomiendan

Sentencia	**Recomendación**
Command	Use **SQL** en su lugar.
CustomConnect	Use **Custom Connect** en su lugar.

12.2 Parámetros de sentencias de script no recomendados en Qlik Sense

Esta tabla describe parámetros de sentencias de script cuyo uso no se recomienda en Qlik Sense.

Parámetros de sentencias de script que no se recomiendan

Sentencia	**Parámetros**
Buffer	Use **Incremental** en vez de: • **Inc** (no recomendado) • **Incr** (no recomendado)

Sentencia	Parámetros
LOAD	Los asistentes de transformación de archivos de QlikView generan las siguientes palabras clave de parámetros. La funcionalidad se conserva cuando vuelven a cargarse los datos, pero Qlik Sense no ofrece soporte guiado/asistentes para generar la sentencia con estos parámetros: • **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 Funciones no recomendadas en Qlik Sense

Esta tabla describe las funciones de script y de gráfico cuyo uso no se recomienda en Qlik Sense.

Funciones que no se recomiendan

Función	Recomendación
NumAvg **NumCount** **NumMax** **NumMin** **NumSum**	Utilice funciones range en su lugar. *Funciones de rango (page 1349)*
Color() **QliktechBlue** **QliktechGray**	Utilice otras funciones de color en su lugar. **QliktechBlue()** puede ser reemplazado por **RGB(8, 18, 90)** y **QliktechGray** puede ser reemplazado por **RGB(158, 148, 137)** para obtener los mismos colores. *Funciones de color (page 566)*
QlikViewVersion	Use **EngineVersion** en su lugar. *EngineVersion (page 1497)*
ProductVersion	Use **EngineVersion** en su lugar. *EngineVersion (page 1497)*
QVUser	
Year2Date	Use **YearToDate** en su lugar.
Vrank	Use **Rank** en su lugar.
WildMatch5	Use **WildMatch** en su lugar.

El cualificador **ALL**

En QlikView, el cualificador **ALL** puede ir antes de una expresión. Esto equivale a usar **{1} TOTAL**. En tal caso, el cálculo se hará sobre todos los valores de campo del documento, descartando las dimensiones del gráfico y las selecciones actuales. Siempre devuelve el mismo valor, independientemente del estado lógico del documento. Si se utiliza el cualificador **ALL**, no se puede usar una expresión de conjunto, ya que el cualificador **ALL** define un conjunto por sí mismo. Por razones de legado, el cualificador **ALL** aún funcionará en esta versión de Qlik Sense, pero puede que se elimine en próximas versiones.