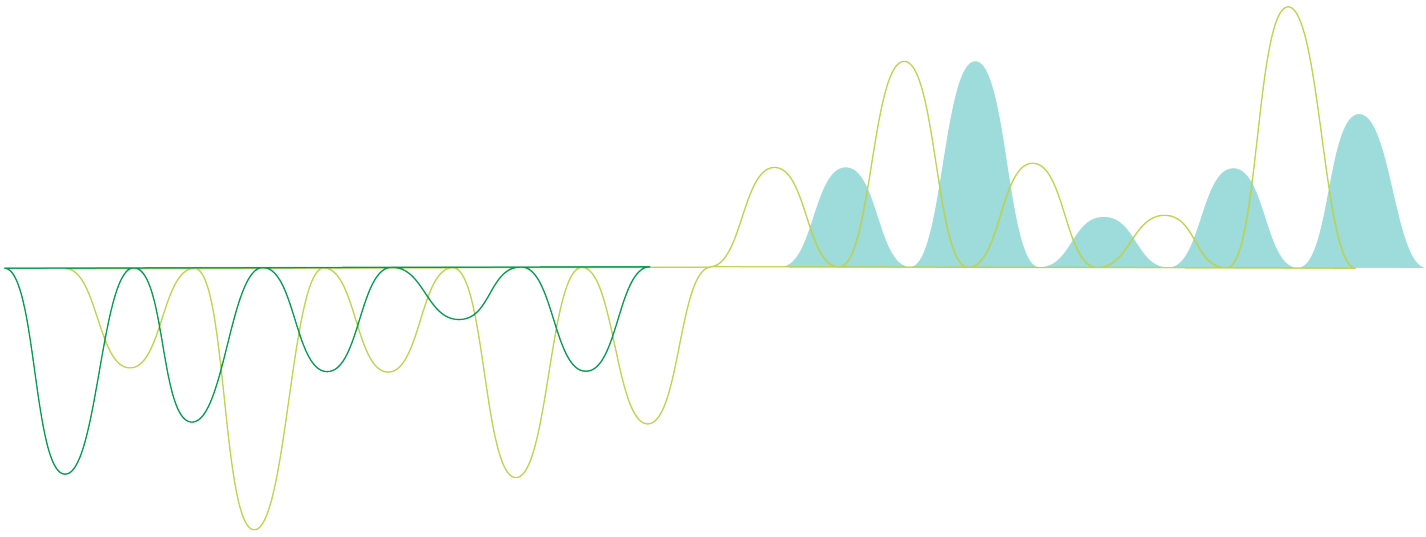
Skriptsyntax och diagramfunktioner

Qlik Sense®
May 2022



Contents

1 Vad är Qlik Sense?

Qlik Sense är en plattform för dataanalys. Med Qlik Sense kan du analysera data och göra upptäckter i dessa data på egen hand. Du kan dela kunskaper och analysera data i grupper och inom organisationer. Med Qlik Sense ställer du dina egna frågor och följer din egen väg mot upptäckter. Med Qlik Sense kan du även fatta gemensamma beslut tillsammans med kollegorna.

1.1 Vad kan du göra i Qlik Sense?

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

1.2 Hur fungerar Qlik Sense?

När du gör spontana analyser i Qlik Sense visas informationen på olika sätt. Qlik Sense kräver inga fördefinierade rapporter eller datavisualiseringar. Du är inte heller beroende av andra användare. Det enda du behöver göra är att klicka och lära dig längs vägen. Varje gång du klickar svarar Qlik Sense omedelbart och uppdaterar varje Qlik Sense-visualisering och vy i appen med aktuella uträknade data och visualiseringar som matchar dina urval perfekt.

Appmodellen

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

Den associativa analysen

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

Samarbete och mobilitet

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

1.3 Hur distribueras Qlik Sense?

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

Qlik Sense Desktop

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

Qlik Sense Enterprise

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

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

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

1.5 Utöka Qlik Sense och anpassa det efter dina syften

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

Bygga komplement och kombinationsprogram

Här kan webbutveckla med JavaScript för att bygga komplement för egna visualiseringar i Qlik Sense-appar eller använda API:er för kombinationsprogram för att bygga webbplatser med Qlik Sense-innehåll.

Bygga klienter

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

Bygga serververktyg

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

Koppla till andra datakällor

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

2 Översikt över skriptsyntax

2.1 Introduktion till skriptsyntax

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

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

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

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

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

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

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

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

2.2 Vad är Backus-Naur-formalism?

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

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

Symboler

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

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

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

Exempel:

Alias-satsen kan beskrivas på följande sätt:

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

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

Följande satser är korrekta:

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

Följande satser är inte korrekta:

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

2 Skriptsatser och nyckelord

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

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

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

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

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

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

2.3 Kontrollsatser i skriptet

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

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

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

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

Översikt av kontrollsatser i skriptet

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

Call

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

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

Do..loop

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

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

Exit script

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

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

For each ..next

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

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

For..next

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

```
For..next counter = expr1 to expr2 [ stepexpr3 ]
[statements]
[exit for [ ( when | unless ) condition ]
[statements]
Next [counter]
```

If..then

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

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

```
If..then..elseif..else..end if condition then
  [ statements ]
{ elseif condition then
  [ statements ] }
[ else
  [ statements ] ]
end if
```

Sub

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

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

Switch

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

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

Call

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

Syntax:

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

Argument:

Argument

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

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

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

Begränsningar:

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

Exempel:

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

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

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

Do..loop

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

Syntax:

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

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

Argument:

Argument

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

Exempel:

```
// LOAD files file1.csv..file9.csv
Set a=1;
Do while a<10
LOAD * from file$(a).csv;
Let a=a+1;
Loop
```

End

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

Exit

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

Exit script

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

Syntax:

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

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

Argument:

Argument

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

Exempel:

```
//Exit script
Exit Script;

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

For..next

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

Syntax:

```
For counter = expr1 to expr2 [ step expr3 ]
[statements]
[exit for [ ( when | unless ) condition ]
[statements]
Next [counter]
```

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

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

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

Argument:

Argument

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

Example 1: Läsa in ett antal filer i följd

```
// LOAD files file1.csv..file9.csv
for a=1 to 9

      LOAD * from file$(a).csv;

next
```

Example 2: Läsa in valfritt antal filer i

I detta exempel utgår vi från datafilerna *x1.csv, x3.csv, x5.csv, x7.csv* och *x9.csv*. Inläsningen stoppas vid valfri punkt med hjälp av villkoret `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

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

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

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

Argument:

Argument

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

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

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

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

Syntax:

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

Argument

Argument	Beskrivning
constant	Valfritt tal eller valfri sträng Observera att en sträng som skrivs direkt i skriptet måste omslutas av enkla citattecken. En sträng utan enkla citattecken tolkas som en variabel och då används värdet för variabeln. Tal behöver inte omslutas av enkla citattecken.
expression	Ett godtyckligt uttryck.
mask	En fil- eller katalognamnsmask som får innehålla valfria giltiga filnamnstecken samt standardjokertecknen ***** och **?**. Du kan använda absoluta sökvägar eller sökvägar till lib://.
condition	Ett logiskt uttryck som utvärderas till True eller False.
statements	Valfri grupp av en eller flera Qlik Sense-skriptsatser.
filelist mask	Denna syntax skapar en kommaavgränsad lista över alla filer i den aktuella mappen som matchar filnamnsmasken. *Argumentet har enbart stöd för bibliotekskopplingar i standardläget.*
dirlist mask	Denna syntax skapar en kommaavgränsad lista över alla kataloger i den aktuella katalog som matchar mappnamnsmasken. *Argumentet har enbart stöd för bibliotekskopplingar i standardläget.*
fieldvaluelist mask	Syntaxen itererar genom värdena i ett fält som redan har laddats in i Qlik Sense.

*Qlik Web Storage Provider Connectors och andra DataFiles-kopplingar stöder inte filtermasker som använder jokertecken (***** och **?**).*

Example 1: Ladda en fillista

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

Example 2: Skapa en fillista på disk

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

```
 sub DoDir (Root)    for each Ext in 'qvw', 'qva', 'qvo', 'qvs', 'qvc', 'qvf', 'qvd'
for each File in filelist (Root&'/*.' &Ext)           LOAD              '$(File)' as Name,
         FileSize( '$(File)' ) as Size,              FileTime( '$(File)' ) as FileTime
 autogenerate 1;        next File     next Ext    for each Dir in dirlist (Root&'/*' )
call DoDir (Dir)     next Dir  end sub  call DoDir ('lib://DataFiles')
```

Example 3: Itererar genom värdena i ett fält

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

```
 load * inline [ FIELD one two three ];  FOR Each a in FieldValueList('FIELD') LOAD '$(a)' &'-
'&RecNo() as NEWFIELD AutoGenerate 2; NEXT a
```

Den resulterande tabellen ser ut så här:

Example table

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

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

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

Kontrollsatser används normalt för att kontrollera skriptexekveringens flöde. I ett diagramuttryck använder du villkorsfunktionen **if** istället.

Syntax:

```
If  condition then
  [ statements ]
{ elseif condition then
  [ statements ] }
[ else
  [ statements ] ]
end if
```

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

Argument:

Argument

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

Example 1:

```
if a=1 then
      LOAD * from abc.csv;
      SQL SELECT e, f, g from tab1;
end if
```

Example 2:

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

Example 3:

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

Next

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

Sub..end sub

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

Syntax:

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

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

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

Argument:

Argument

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

Begränsningar:

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

Example 1:

```
Sub INCR (I,J)
I = I + 1
Exit Sub when I < 10
J = J + 1
End Sub
Call INCR (X,Y)
```

Example 2: - parameteröverföring

```
Sub ParTrans (A,B,C)
A=A+1
B=B+1
C=C+1
End Sub
A=1
X=1
C=1
Call ParTrans (A, (X+1)*2)
```

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

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

Switch..case..default..end switch

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

Syntax:

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

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

Argument:

Argument

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

Exempel:

```
Switch I
Case 1
LOAD '$(I): CASE 1' as case autogenerate 1;
Case 2
LOAD '$(I): CASE 2' as case autogenerate 1;
Default
LOAD '$(I): DEFAULT' as case autogenerate 1;
End Switch
```

To

Skriptnyckelordet **To** används i flera skriptsatser.

2.4 Skriptprefix

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

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

Skriptprefix – en översikt

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

Add

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

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

Buffer

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

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

Concatenate

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

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

Crosstable

Prefixet **crosstable** används för att omvandla en korstabell till en rak tabell, det vill säga en bred tabell med många kolumner omvandlas till en hög tabell där kolumnrubrikerna placeras i en enda attributkolumn.

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

First

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

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

Generic

generic-prefixet används för uppackning och laddning av generiska databaser.

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

Hierarchy

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

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

HierarchBelongsTo

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

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

Inner

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

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

IntervalMatch

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

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

Join

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

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

Keep

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

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

Left

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

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

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

Mapping

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

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

Merge

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

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

NoConcatenate

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

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

Outer

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

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

Partial reload

En fullständig laddning inleds alltid med att alla tabeller i den befintliga datamodellen tas bort, och sedan körs laddningsskriptet. En *Delvis laddning (page 60)* kommer inte att göra detta. Då behålls istället alla tabeller i datamodellen och sedan exekveras bara **Load**- och **Select**-satser med ett inledande **Add**-, **Merge**- eller **Replace**-prefix. Andra datatabeller påverkas inte av kommandot. **Only**-argumentet anger att satsen bara ska exekveras vid delvisa laddningar och ignoreras vid fullständiga laddningar. Följande tabell sammanfattar programutförandet för partiella och fullständiga ominläsningar.

Replace

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

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

Right

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

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

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

Sample

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

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

Semantic

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

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

Unless

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

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

When

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

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

Add

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

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

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

Syntax:

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

```
Add [only] mapstatement
```

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

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

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

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

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

Argument:

Argument

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

Exempel och resultat:

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

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

Buffer

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

Syntax:

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

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

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

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

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

Argument:

Argument

Argument	Beskrivning
inkrementell	Med alternativet incremental öppnas möjligheten att enbart läsa in delar av den underliggande filen. Filens tidigare storlek lagras i XML-huvudet i QVD-filen. Detta är särskilt användbart för loggfiler. Alla tidigare inlästa poster läses in från QVD-filen medan nyare poster läses in från originalkällan och en uppdaterad QVD-fil skapas. Alternativet incremental kan bara användas med **LOAD**-satser och textfiler. Inkrementell laddning kan inte användas där gamla data ändras eller raderas.
gammal [efter] belopp [(dagar | timmar)]	amount är ett tal som anger tidsperioden. Decimaler får användas. Om ingen enhet anges förutsätts dagar. Alternativet stale after används oftast med databaskällor vars originaldata är svåra att förse med en tidsmarkör. Istället anger man hur gammal en QVD-minnesdump kan vara för att få användas. En stale after-sats anger helt enkelt en tidsperiod efter att QVD-bufferten skapades, efter vilken den inte längre kommer att anses vara giltig. Dessförinnan används QVD-bufferten som källa för data och därefter används den ursprungliga datakällan. Därefter uppdateras QVD-buffertfilen automatiskt och en ny period påbörjas.

Begränsningar:

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

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

Om man vill konkatenera två tabeller som inte har samma fältuppsättning, kan man utföra en tvingad konkatenering med hjälp av **Concatenate**-prefixet. Denna sats åstadkommer en tvingad konkatenering med en befintlig namngiven tabell eller med den senast skapade

logiska tabellen.

Syntax:

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

Konkatenering är i princip samma sak som **SQL UNION**-satsen, men det finns två skillnader:

- **Concatenate**-prefixet kan användas oavsett om tabellerna har identiska fältnamn eller ej.
- Identiska poster tas inte bort med **Concatenate**-prefixet.

Argument:

Argument

Argument	Beskrivning
tablename	Namnet på den befintliga tabellen.

Exempel:

```
Concatenate LOAD * From file2.csv;
Concatenate SELECT * From table3;
tab1:
LOAD * From file1.csv;
tab2:
LOAD * From file2.csv;
.. .. ..
Concatenate (tab1) LOAD * From file3.csv;
```

Crosstable

Prefixet **crosstable** används för att omvandla en korstabell till en rak tabell, det vill säga en bred tabell med många kolumner omvandlas till en hög tabell där kolumnrubrikerna placeras i en enda attributkolumn.

Syntax:

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

Argument:

Argument

Argument	Beskrivning
attribute field name	Fältet som innehåller attributvärdena.
data field name	Fältet som innehåller datavärdena.
n	Antalet kvalificerande fält som föregår den tabell som ska omvandlas till generisk form. Standard är 1.

En korstabell är en vanlig tabelltyp, som består av en datamatris mellan två eller fler ortogonala listor av rubrikdata. En av dessa används för kolumnrubriker. Exempelvis skulle man kunna ha en kolumn per månad. Resultatet av **crosstable**-prefixet blir att kolumnrubrikerna (till exempel månadsnamn) lagras i ett fält - attributfältet - och kolumndata (månadsnumren) lagras i ett andra fält: datafältet.

Exempel

Laddningsskript

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

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

Resultat

Resultattabell

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

Förklaring

I det här exemplet demonstreras en korstabell med en kolumn för varje månad och en rad per produkt. I det nuvarande formatet är det inte lätt att analysera informationen. Det skulle vara mycket bättre att ha alla siffror i ett fält och alla månader i ett annat, det vill säga i en tabell med tre kolumner. Nu ska vi se hur du gör en sådan omvandling av korstabellen.

Omvandla korstabellen

Ta bort kommentaren från skriptet och kör det.

```
tmpData: Crosstable (MonthText, Sales) Load * inline [ Product, Jan 2021, Feb 2021, Mar 2021, Apr 2021, May 2021, Jun 2021 A, 100, 98, 103, 63, 108, 82 B, 284, 279, 297, 305, 294, 292 C, 50, 53, 50, 54, 49, 51];   Final: Load Product, Date(Date#(MonthText,'MMM YYYY'),'MMM YYYY') as Month, Sales  Resident tmpData;  Drop Table tmpData;
```

Resultat

Resultattabell

Produkt	Månad	Sales
A	Jan 2021	100
A	Feb 2021	98

Produkt	Månad	Sales
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

Förklaring

Korstabellen omvandlas till en enkel tabell med en kolumn för Month och en annan för Sales.

First

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

Syntax:

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

Argument:

Argument

Argument	Beskrivning
n	Ett godtyckligt uttryck, vars resultat är ett heltal som anger det maximala antalet poster som ska laddas. *n* kan omslutas av parenteser, såsom *(n)*, men detta är inget krav.

Exempel:

```
First 10 LOAD * from abc.csv;
First (1) SQL SELECT * from Orders;
```

Generic

generic-prefixet används för uppackning och laddning av generiska databaser.

Generiska databaser/datakällor innehåller strukturerade repetitiva data, till exempel en adresslista eller ett produktspecifikationsark där elementen upprepade gånger beskrivs med liknande attribut.

Syntax:

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

Exempel:

```
Generic LOAD * from abc.csv;
Generic SQL SELECT * from table1;
```

Tabeller som laddas med **generic**-satsen konkateneras inte automatiskt.

Exempel

Exempel 1

Laddningsskript

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

```
GenericDB:
Generic Load *;
Load * inline [
Region, Attribute, Value

US, Name, AAA
US, Address, A123
US, Phone, 001-123

US, Name, BBB
US, Address, B456
US, Phone, 002-456

SWE, Name, CCC
SWE, Address, C7789
SWE, Phone, 003-789 ];
```

Resultat

Resultattabell

Region	Namn	Address	Phone
SWE	CCC	C7789	003-789
US	AAA	A123	001-123
US	AAA	A123	002-456
US	AAA	B456	001-123
US	AAA	B456	002-456
US	BBB	A123	001-123
US	BBB	A123	002-456
US	BBB	B456	001-123
US	BBB	B456	002-456

Exempel 2

Laddningsskript

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

```
Sheet1:
Generic Load * inline [
object, attribute, value
ball, color, red
ball, diameter, 10 cm
ball, weight, 100 g
box, color, black
box, height, 16 cm
box, length, 20 cm
box, weight, 500 g
box, width, 10 cm ];
```

Resultat

Resultattabell

objekt	färg	diameter	längd	höjd	bredd	vikt
ball	red	10 cm	-	-	-	100 g
box	black	-	20 cm	16 cm	10 cm	500 g

Hierarchy

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

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

Syntax:

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

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

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

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

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

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

Argument:

Argument

Argument	Beskrivning
NodeID	Namnet på det fält som innehåller nodens ID. Detta fält måste förekomma i indatatabellen.
ParentID	Namnet på det fält som innehåller modernodens nod-ID. Detta fält måste förekomma i indatatabellen.
NodeName	Namnet på det fält som innehåller nodens namn. Detta fält måste förekomma i indatatabellen.
ParentName	En sträng som används för att namnge det nya **ParentName**-fältet. Om den utelämnas skapas ej detta fält.

Argument	Beskrivning
ParentSource	Namnet på det fält som innehåller namnet på den nod som används för att bygga nodens sökväg. Valfri parameter. Om den utelämnas används **NodeName**.
PathName	En sträng som används för att namnge det nya fältet **Path** som innehåller sökvägen från rot till nod. Valfri parameter. Om den utelämnas skapas ej detta fält.
PathDelimiter	En sträng som används som avgränsare i det nya **Path**-fältet. Valfri parameter. Om den utelämnas används '/'.
Depth	En sträng som används för att namnge det nya fältet **Depth** som innehåller djupet på noden i hierarkin. Valfri parameter. Om den utelämnas skapas ej detta fält.

Exempel:

```
Hierarchy(NodeID, ParentID, NodeName, ParentName, NodeName, PathName, '\', Depth) LOAD *
inline [
NodeID, ParentID, NodeName
1, 4, London
2, 3, Munich
3, 5, Germany
4, 5, UK
5, , Europe
];
```

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

HierarchyBelongsTo

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

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

Syntax:

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

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

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

Ett ytterligare fält som innehåller djupskillnaden mellan noderna kan skapas.

Argument:

Argument

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

Exempel:

```
HierarchyBelongsTo (NodeID, AncestorID, NodeName, AncestorID, AncestorName, DepthDiff) LOAD *
inline [
NodeID, AncestorID, NodeName
1, 4, London
2, 3, Munich
3, 5, Germany
4, 5, UK
5, , Europe
];
```

Results

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

Inner

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

Syntax:

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

Argument:

Argument

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

Exempel

Laddningsskript

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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 ];
```

Resultat

Resultattabell

Column1	Column2	Column3
A	B	C
1	aa	xx

Förklaring

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

IntervalMatch

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

Syntax:

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

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

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

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

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

Argument:

Argument

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

Example 1:

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

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

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

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

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

Table with additional column

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

Example 2: (med keyfield)

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

```
EventLog:
LOAD * Inline [
Time, Event, Comment, ProductionLine
00:00, 0, Start of shift 1, P1
01:00, 0, Start of shift 1, P2
01:18, 1, Line stop, P1
02:23, 2, Line restart 50%, P1
04:15, 3, Line speed 100%, P1
08:00, 4, Start of shift 2, P1
09:00, 4, Start of shift 2, P2
11:43, 5, End of production, P1
11:43, 5, End of production, P2
];

OrderLog:
LOAD * INLINE [
Start, End, Order, ProductionLine
01:00, 03:35, A, P1
02:30, 07:58, B, P1
03:04, 10:27, C, P1
07:23, 11:43, D, P2
];

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

En tabellbox kan du skapas enligt nedan:

Tablebox example

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

Join

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

Syntax:

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

Länkningen är en så kallad natural join, som görs över alla gemensamma fält. Join-satsen kan inledas med något av prefixen **inner**, **outer**, **left** eller **right**.

Argument:

Argument

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

Exempel:

```
Join SELECT * from table1;



tab1:

LOAD * from file1.csv;
```

```
tab2:

LOAD * from file2.csv;

.. .. ..

join (tab1) LOAD * from file3.csv;
```

Exempel

Laddningsskript

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

```
Table1: Load * inline [ Column1, Column2 A, B 1, aa 2, cc 3, ee ];  Table2: Join Load * inline
[ Column1, Column3 A, C 1, xx 4, yy ];
```

Resultattabell

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

Förklaring

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

Keep

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

Syntax:

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

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

Det explicita prefixet **join** i Qlik Senses skriptspråk skapar en fullständig länkning (full join) mellan de båda tabellerna. Resultatet är en tabell. I många fall resulterar sådana länkningar i mycket stora tabeller. En av finesserna med Qlik Sense är just att programmet gör associationer mellan flera tabeller istället för att skapa länkningar, vilket minskar minnesanvändningen avsevärt, ökar hastigheten och ger större flexibilitet. Explicita länkningar bör därför normalt undvikas i Qlik Sense-skript. Keep-funktionaliteten har utvecklats för att minska antalet fall där man måste använda länkning.

Argument:

Argument

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

Exempel:

```
Inner Keep LOAD * from abc.csv;
Left Keep SELECT * from table1;
tab1:
LOAD * from file1.csv;
tab2:
LOAD * from file2.csv;
.. .. ..
Left Keep (tab1) LOAD * from file3.csv;
```

Left

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

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

Letade du efter strängfunktionen med samma namn? Se: Left (page 788)

Syntax:

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

Argument:

Argument

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

Exempel

Laddningsskript

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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 ];
```

Resultat

Resultattabell

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

Förklaring

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

Mappning

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

Syntax:

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

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

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

Exempel:

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

```
// Load mapping table of country codes:
map1:
mapping LOAD *
```

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

Den resulterande tabellen ser ut så här:

Mapping table

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

Merge

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

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

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

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

Syntax:

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

Argument:

Argument

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

Allmän funktionalitet

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

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

Måltabell

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

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

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

Sekvensnummer

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

Operationer

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

Infoga: Fält som saknas i **Merge LOAD**, men som finns i måltabellen, får NULL i måltabellen.

Ta bort: Saknade fält påverkar inte resultatet. Relevanta poster raderas ändå.

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

- 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 ...;

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

> *Merge LOAD kan inte användas på poster med jokertecken, till exempel en Section Access-tabell med en asterisk, för att ange alla värden.*

Exempel

Exempel 1: Enkel sammanslagning med specificerad tabell

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

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

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

Laddningsskript

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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 ];
```

Resultat

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

Resulting table

Name	Number
Jake	3
Jill	2
Steven	3

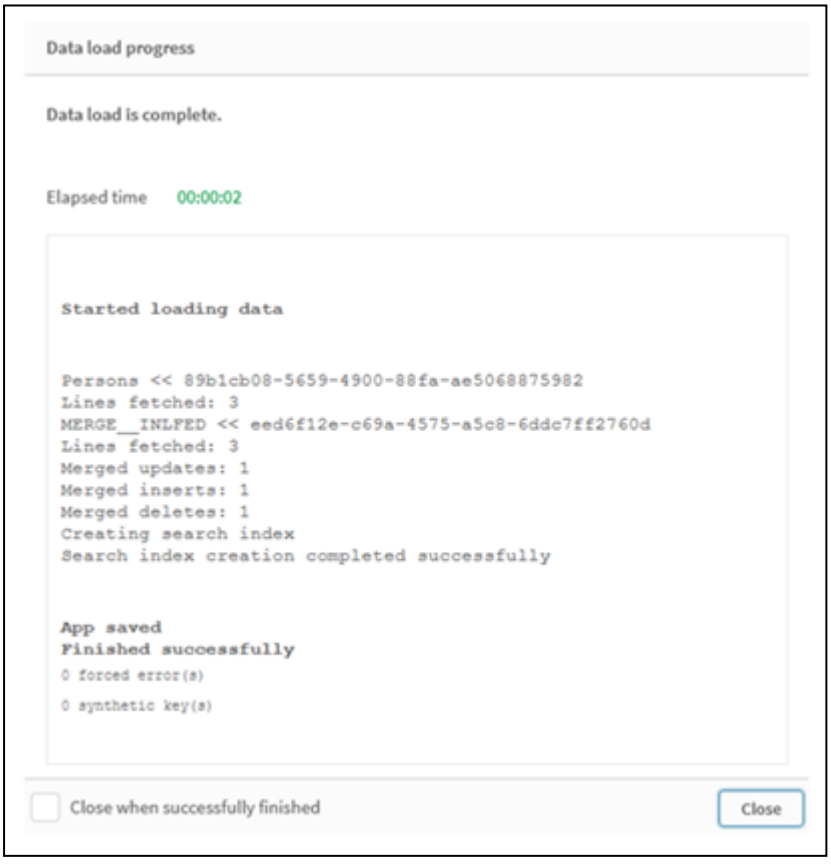
Efter **Merge Load** ser tabellen ut så här:

Resulting table

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

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

Dataladdningsdialogrutan



Exempel 2: Dataladdningsskript med saknade fält

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

Merge ändrar tabellen på följande sätt:

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

Laddningsskript

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

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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 ];
```

Resultat

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

Resulting table

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

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

Exempel 3: Dataladdningsskript – Delvis laddning med hjälp av en Where-sats med ChangeDate

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

Laddningsskript

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

NoConcatenate

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

Syntax:

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

Exempel:

```
LOAD A,B from file1.csv;
NoConcatenate LOAD A,B from file2.csv;
```

Only

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

Outer

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

Syntax:

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

Argument:

Argument

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

Exempel

Laddningsskript

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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 ];
```

Resultattabell

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

Förklaring

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

Delvis laddning

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

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

Sats	Fullständig omladdning	Delvis laddning
Load ...	Sats körs	Sats körs inte
Add/Replace/Merge Load ...	Sats körs	Sats körs
Add/Replace/Merge Only Load ...	Sats körs inte	Sats körs

Delvisa laddningar har flera fördelar jämfört med fullständiga laddningar:

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

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

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

Exempel

Exempel 1

Laddningsskript

Lägg till exempelskriptet i din app och gör en delvis omladdning. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

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

Resultat

Resulting table

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

Förklaring

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

Exempel 2

Laddningsskript

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

Resultat

Output table after full reload

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

Output table after partial reload

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

Förklaring

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

Replace

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

Replace

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

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

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

Syntax:

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

```
Replace [only] mapstatement
```

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

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

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

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

Argument:

Argument

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

Exempel och resultat:

Exempel	Resultat

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

`Tab1:`
`Replace only`
`LOAD * from`
`File1.csv;`
	Vid vanlig laddning beaktas inte denna sats.
	Vid partiell laddning utgår alla Qlik Sense-tabeller som tidigare hette Tab1inledningsvis. Sedan läses nya data in från File1.csv och lagras i Tab1.

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

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

Right

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

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

Letade du efter strängfunktionen med samma namn? Se: Right (page 797)

Syntax:

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

Argument:

Argument

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

Exempel

Laddningsskript

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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 ];
```

Resultat

Resultattabell

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

Förklaring

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

Sample

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

Syntax:

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

Argument:

Argument

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

Exempel:

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

Parenteser är tillåtna men inte obligatoriska.

Semantic

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

Syntax:

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

Semantic Load skapar semantiska fält som kan visas i filterrutor som kan användas för navigering bland data.

Tabeller som laddas med **semantic**-satsen kan inte konkateneras.

Exempel:

```
Semantic LOAD * from abc.csv;
Semantic SELECT Object1, Relation, Object2, InverseRelation from table1;
```

Unless

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

Syntax:

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

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

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

Argument:

Argument

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

Exempel:

```
exit script unless A=1;
unless A=1 LOAD * from myfile.csv;
unless A=1 when B=2 drop table Tab1;
```

When

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

Syntax:

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

statement eller **exitstatement** exekveras bara om villkoret utvärderas till True.

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

Syntax:

Argument

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

Example 1:

```
exit script when A=1;
```

Example 2:

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

Example 3:

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

2.5 Vanliga skriptsatser

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

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

Översikt av vanliga skriptsatser

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

Alias

Satsen **alias** används för att ställa in ett alias enligt vilket ett fält ska döpas om när det förekommer i skriptet som följer.

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

Autonumber

Denna sats returnerar ett unikt heltal för varje distinkt utvärderat värde i ett fält som påträffas under skriptexekveringen.

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

Binary

binary-satsen används för att ladda data från ett annat QlikView-dokument, inklusive Section Access-data.

```
Binary  [path] filename
```

comment

Gör det möjligt att visa fältkommentarerna (metadata) från databaser eller kalkylblad. Fältnamn som inte finns i appen ignoreras. Om flera förekomster av ett fältnamn hittas används det senaste värdet.

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

comment table

Gör det möjligt att visa tabellkommentarerna (metadata) från databaser eller kalkylblad.

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

Connect

Den här funktionen är inte tillgänglig i Qlik Sense SaaS.

CONNECT-satsen används för att ange Qlik Sense-åtkomst till en allmän databas via OLE DB/ODBC-gränssnittet. För ODBC måste datakällan först anges med hjälp av ODBC-administratören.

```
ODBC Connect TO connect-string [ ( access_info ) ]
OLEDB CONNECT TO connect-string [ ( access_info ) ]
CUSTOM CONNECT TO connect-string [ ( access_info ) ]
LIB CONNECT TO connection
```

Declare

Declare-satsen används för att skapa fältdefinitioner där du kan definiera relationer mellan fält eller funktioner. En uppsättning fältdefinitioner kan användas för att automatiskt generera härledda fält, vilka kan användas som dimensioner. Du kan till exempel skapa en kalenderdefinition och använda den för att generera relaterade dimensioner, som år, månad, vecka och dag, från ett datumfält.

```
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

Derive-satsen används för att generera härledda fält baserat på en fältdefinition som skapats med en **Declare**-sats. Du kan antingen specifikt ange vilka datafält du vill härleda fält från, eller härleda dem explicit eller implicit baserat på fälttaggar.

```
Derive [Field[s]] From [Field[s]] field_list Using definition
Derive [Field[s]] From Explicit [Tag[s]] (tag_list) Using definition
Derive [Field[s]] From Implicit [Tag[s]] Using definition
```

Direct Query

Med **DIRECT QUERY**-satsen kan du komma åt tabeller genom enODBC eller OLE DB-koppling med hjälp av funktionen Direct Discovery.

```
Direct Query [path]
```

Directory

Directory-satsen anger i vilken katalog datafilerna ska sökas i följande **LOAD**-satser tills en ny **Directory**-sats anges.

```
Directory [path]
```

Disconnect

Disconnect-satsen avslutar den aktuella ODBC-kopplingen/OLE DB-kopplingen/anpassade kopplingen. Denna sats är valfri.

```
Disconnect
```

drop field

Ett eller flera Qlik Sense-fält kan avlägsnas från datamodellen, och därigenom från minnet, när som helst under skriptexekveringen med hjälp av en **drop field**-sats.

> *Både **drop field** och **drop fields** är tillåtna och har samma betydelse. Om ingen tabell har angetts avlägsnas fältet från alla tabeller där det förekommer.*

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

drop table

Genom att använda en **drop table**-sats kan man när som helst under skriptexekveringen avlägsna en eller flera interna tabeller i Qlik Sense ur datamodellen och därmed ur minnet.

*Formerrna **drop table** och **drop tables** accepteras båda.*

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

Execute

Execute-satsen använda för att köra andra program medan Qlik Sense laddar data, exempelvis för att göra de konverteringar som krävs.

```
Execute commandline
```

FlushLog

Satsen **FlushLog** tvingar Qlik Sense att skriva innehållet i skriptbufferten till skriptets loggfil.

```
FlushLog
```

Force

force-satsen tvingar Qlik Sense att tolka fältnamn och fältvärden i efterföljande **LOAD**- och **SELECT**-satser som att de är skrivna med enbart versaler, enbart gemener, alltid med inledande versaler eller som de uppträder (blandat). Med hjälp av denna sats är det möjligt att koppla fält från tabeller med olika konventioner.

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

LOAD

LOAD-satsen laddar fält från en fil, direkt från data i skriptet, från en tidigare inläst tabell, från en webbsida, från resultatet av en efterföljande **SELECT**-sats eller genom att generera data automatiskt.Det går även att ladda data från analytiska kopplingar.

```
 Load [ distinct ] *fieldlist
[( from file [ format-spec ] |
from_field fieldassource [format-spec]
inline data [ format-spec ] |
resident table-label |
autogenerate size )]
[ where criterion | while criterion ]
[ group_by groupbyfieldlist ]
[order_by orderbyfieldlist ]
[extension pluginname.functionname(tabledescription)]
```

Let

let-satsen är ett komplement till **set**-satsen som används för att definiera skriptvariabler. I motsats till **set**-satsen utvärderas **let**-satsens uttryck till höger om '=' när skriptet körs, innan det tilldelas variabeln.

```
Let variablename=expression
```

Loosen Table

En eller flera av Qlik Senses interna datatabeller kan explicit förklaras vara löst kopplade under skriptexekvering med hjälp av satsen **Loosen Table**. När en tabell är löst kopplad tas alla associationer mellan fältvärdena i tabellen bort. Man kan åstadkomma en liknande effekt genom att ladda vart och ett av fälten i den löst kopplade tabellen som fristående, okopplade tabeller. Löst kopplad kan vara användbart under tester för att tillfälligt isolera olika delar av datastrukturen. En löst kopplad tabellen kan identifieras i tabellvyn genom de streckade linjerna. Om en eller flera **Loosen Table**-satser används i skriptet gör detta att Qlik Sense ignorerar alla inställningar för löst kopplade tabeller gjorda innan skriptexekveringen.

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

Map ... using

Satsen **map ... using** används för att mappa ett visst fältvärde eller uttryck till värdena i en specifik mappningstabell. Mappningstabellen skapas via satsen **Mapping**.

```
Map  *fieldlist Using  mapname
```

NullAsNull

Med **NullAsNull**-satsen upphävs konvertering av NULL-värden till strängvärden som dessförinnan har ställts in med **NullAsValue**-satsen.

```
NullAsNull *fieldlist
```

NullAsValue

Satsen **NullAsValue** anger för vilka fält de NULL-värdena ska konverteras till värden.

```
NullAsValue *fieldlist
```

Qualify

Satsen **Qualify** används för att aktivera bestämning av fältnamn, där fältnamn får tabellnamnet som prefix.

```
Qualify *fieldlist
```

Rem

rem-satsen används för att infoga kommentarer i skriptet, eller för att tillfälligt avaktivera skriptsatser utan att ta bort dem.

```
Rem string
```

Rename Field

Denna skriptfunktion döper om ett eller flera Qlik Sense-fält efter att de har lästs in.

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

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

Rename Table

Denna skriptfunktion döper om ett eller flera interna tabeller i Qlik Sense efter att de har lästs in.

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

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

Section

Med en **section**-sats är det möjligt att definiera om efterföljande **LOAD**- och **SELECT**-satser ska betraktas som data eller som en definition av behörighet.

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

Select

Val av fält från en ODBC-datakälla eller en OLE DB-drivrutin görs via vanliga SQL **SELECT**-satser. Om **SELECT**-satserna accepteras eller ej beror framför allt på den ODBC-drivrutin eller OLE DB-drivrutin som används.

```
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

Satsen **set** används för att definiera skriptvariabler. Dessa kan användas för att ersätta strängar, sökvägar, drivrutiner med mera.

```
Set variablename=string
```

Sleep

sleep-satsen pausar skriptexekveringen under en angiven tidsperiod.

```
Sleep n
```

SQL

Med **SQL**-satsen kan du skicka ett godtyckligt SQL-kommando via en ODBC- eller OLE DB-koppling.

```
SQL sql_command
```

SQLColumns

Satsen **sqlcolumns** returnerar ett antal fält som beskriver kolumnerna i den ODBC- eller OLE DB-datakälla som man gjort en koppling, **connect**, till.

```
SQLColumns
```

SQLTables

Satsen **sqltables** returnerar ett antal fält som beskriver tabellerna i den ODBC- eller OLE DB-datakälla som man gjort en koppling, **connect**, till.

```
SQLTables
```

SQLTypes

Satsen **sqltypes** returnerar ett antal fält som beskriver typerna i den ODBC- eller OLE DB-datakälla som man gjort en koppling, **connect**, till.

```
SQLTypes
```

Star

Den textsträng som används för att representera den totala uppsättningen av alla värden i ett fält i databasen kan ställas in med hjälp av **star**-satsen. Den påverkar efterföljande **LOAD**- och **SELECT**-satser.

```
Star is [ string ]
```

Store

Store-satsen skapar en QVD-, CSV- eller text-fil.

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

Tag

Den här skriptsatsen erbjuder ett sätt att tilldela taggar till ett eller flera fält eller tabeller. Om du försöker tagga ett fält eller en tabell som inte finns i appen ignoreras taggningen. Om det finns flera förekomster av ett fältnamn eller taggnamn används det senaste värdet.

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

Trace

trace-satsen skriver en sträng till fönstret **Skriptexekvering** och skriptets loggfil när den används. Detta är mycket användbart vid felsökning. Med hjälp av dollarteckenexpansion av variabler som beräknas innan **trace**-satsen används kan man anpassa meddelandet.

```
Trace string
```

Unmap

Satsen **Unmap** avaktiverar fältvärdesmappningar, som har gjorts med hjälp av en tidigare **Map … Using**-sats för fält som läses in efteråt.

```
Unmap *fieldlist
```

Unqualify

Unqualify-satsen upphäver tabellbestämning av fältnamn som tidigare definierats av en **Qualify**-sats.

```
Unqualify *fieldlist
```

Untag

Den här skriptsatsen erbjuder ett sätt att ta bort taggar från fält eller tabeller. Om du försöker ta bort taggar från ett fält eller en tabell som inte finns i appen ignoreras försöket.

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

Alias

Satsen **alias** används för att ställa in ett alias enligt vilket ett fält ska döpas om när det förekommer i skriptet som följer.

Syntax:

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

Argument:

Argument

Argument	Beskrivning
fieldname	Fältets namn i dina källdata
aliasname	Ett alias som du vill använda i stället

Exempel och resultat:

Exempel	Resultat
`Alias ID_N as NameID;`	
`Alias A as Name, B as Number, C as Date;`	Namnändringarna som definieras i denna sats tillämpas på alla följande **SELECT**- och **LOAD**-satser. Ett nytt alias för ett fältnamn kan definieras med en ny **alias**-sats var som helst i skriptet.

AutoNumber

Denna sats returnerar ett unikt heltal för varje distinkt utvärderat värde i ett fält som påträffas under skriptexekveringen.

Du kan också använda *autonumber (page 433)* funktionen i ett **LADDA** påstående men den ar vissa begränsningar när du vill använda en optimerad laddning. Du kan skapa en optimerad laddning genom att ladda data från en **QVD**-fil först och sedan använda **Autonummer** påståendet för att konvertera värden till symbolnycklar.

Syntax:

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

Argument:

Argument

Argument	Beskrivning
*fältlista	En kommaseparerad lista med fält där värdena ska ersättas med ett unikt heltal. Du kan använda jokertecken ? och * i fältnamnen för att inkludera alla fält med matchande namn. Du kan också använda * för att inkludera alla fält. Du måste citera fältnamn när du använder jokertecken.
namnområde	**Användandet av** namnområde är valfritt. Du kan använda det här alternativet om du vill skapa ett namnområde, där identiska värden i olika fält delar samma nyckel. Om du inte använder det här alternativet, har alla fält separat nyckelindex.

Begränsningar:

Om du har flera **LADDA** meddelanden i skriptet måste du placera **Autonummer**-meddelandet efter det sista **LADDA** meddelandet.

Exempel – skript med AutoNumber

Skriptexempel

I det här exemplet laddas data först utan satsen **AutoNumber**. Sedan läggs **AutoNumber**-satsen till för att visa effekten.

Data som används i exemplet

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa skriptexemplet nedan. Låt **AutoNumber**-satsen vara bortkommenterad så länge.

```
RegionSales: LOAD *,  Region &'|'& Year &'|'& Month as KeyToOtherTable INLINE [ Region, Year,
Month, Sales North,     2014,   May,    245 North,      2014,   May,    347 North,      2014,   June,   127 S
June,   645 South,      2013,   May,    367 South,      2013,   May,    221 ];
&'|'& Year &'|'& Month as KeyToOtherTable INLINE [Region, Year, Month, Budget North,  2014,
May,    200 North,  2014,   May,    350 North,  2014,   June,   150 South,  2014,   June,
500 South,  2013,   May,    300 South,  2013,   May,    200 ];  //AutoNumber KeyToOtherTable;
```

Skapa visualiseringar

Skapa två tabellvisualiseringar i ett Qlik Sense-ark. Lägg till **KeyToOtherTable**, **Region**, **Year**, **Month** och **Sales** som dimensioner i den första tabellen. Lägg till **KeyToOtherTable**, **Region**, **Year**, **Month** och **Budget** som dimensioner i den andra tabellen.

Resultat

Tabellen RegionSales

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

KeyToOtherTable	Region	Year	Month	Sales
North|2014|May	North	2014	May	245
North|2014|May	North	2014	May	347
South|2013|May	South	2013	May	221
South|2013|May	South	2013	May	367
South|2014|June	South	2014	June	645

Tabellen Budget

KeyToOtherTable	Region	Year	Month	Budget
North|2014|June	North	2014	June	150
North|2014|May	North	2014	May	200
North|2014|May	North	2014	May	350
South|2013|May	South	2013	May	200
South|2013|May	South	2013	May	300
South|2014|June	South	2014	June	500

Förklaring

I exemplet visas det sammansatta fältet **KeyToOtherTable** som länkar ihop de båda tabellerna. **AutoNumber** används inte. Observera längden på **KeyToOtherTable**-värdena.

Lägga till AutoNumber-satsen

Ta bort kommentarsmarkeringen på **AutoNumber**-satsen i laddningsskriptet.

```
AutoNumber KeyToOtherTable;
```

Resultat

Tabellen RegionSales

KeyToOtherTable	Region	Year	Month	Sales
1	North	2014	June	127
1	North	2014	May	245
2	North	2014	May	347
3	South	2013	May	221
4	South	2013	May	367
4	South	2014	June	645

Tabellen Budget

KeyToOtherTable	Region	Year	Month	Budget
1	North	2014	June	150
1	North	2014	May	200
2	North	2014	May	350
3	South	2013	May	200
4	South	2013	May	300
4	South	2014	June	500

Förklaring

Fältvärdena i **KeyToOtherTable** har ersatts med unika heltalsvärden och därmed reduceras även längden på fältvärdena, vilket sparar minne. Nyckelfälten i de båda tabellerna påverkas av **AutoNumber** och tabellerna förblir länkade. Exemplet är kortfattat eftersom det är avsett som en illustration, men blir meningsfullt med en tabell som innehåller ett stort antal rader.

Binary

binary-satsen används för att ladda data från en annan Qlik Sense-app eller ett QlikView-dokument, inklusive section access-data. Övriga element i appen ingår inte, exempelvis ark, berättelser, visualiseringar, original eller variabler.

Endast en **binary**-sats tillåts i skriptet. **binary**-satsen måste vara den första satsen i skriptet, till och med före SET-satserna som oftast finns i början av skriptet.

Syntax:

```
binary [path] filename
```

Argument:

Argument

Argument	Beskrivning
path	Sökvägen till filen, som ska vara en referens till en mappdatakoppling. Detta krävs om filen inte finns i Qlik Sense-arbetskatalogen. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till den app som innehåller skriptraden. **Exempel:** *data*
filename	Filens namn, inklusive tillägget .qvw eller .qvf.

Begränsningar:

Du kan inte använda **binary** för att ladda data från en app på samma Qlik Sense Enterprise-driftsättning genom att referera till appens ID. Du kan bara ladda från en *.qvf*-fil.

Exempel

Sträng	Beskrivning
`Binary lib://DataFolder/customer.qvw;`	I det här exemplet måste filen finnas i **Mapp**-datakopplingen. Det kan till exempel vara en mapp som administratören skapar på Qlik Sense-servern. Klicka på **Skapa ny koppling** i Skriptredigeraren och välj sedan **Mapp** under **Filplatser**.
`Binary customer.qvf;`	I det här exemplet måste filen finnas i Qlik Sense-arbetskatalogen.
`Binary c:\qv\customer.qvw;`	Det här exemplet, som använder en absolut sökväg, fungerar bara i bakåtkompatibelt skriptläge.

Comment field

Gör det möjligt att visa fältkommentarerna (metadata) från databaser eller kalkylblad. Fältnamn som inte finns i appen ignoreras. Om flera förekomster av ett fältnamn hittas används det senaste värdet.

Syntax:

```
comment [fields] *fieldlist using mapname
comment [field] fieldname with comment
```

Mappningstabellen som används bör ha två kolumner: en med fältnamn och en annan med kommentarer.

Argument:

Argument

Argument	Beskrivning
**fieldlist*	En kommaavgränsad lista över fält som ska kommenteras. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken.
mapname	Namnet på en mappningstabell som tidigare lästs in via en **LOAD**- eller **SELECT**-sats för mappning.
fieldname	Namnet på det fält som ska kommenteras.
comment	Den kommentar som ska läggas till i fältet.

Example 1:

```
commentmap:
mapping LOAD * inline [
a,b
Alpha,This field contains text values
Num,This field contains numeric values
];
comment fields using commentmap;
```

Example 2:

```
comment field Alpha with AFieldContainingCharacters;
comment field Num with '*A field containing numbers';
comment Gamma with 'Mickey Mouse field';
```

Comment table

Gör det möjligt att visa tabellkommentarerna (metadata) från databaser eller kalkylblad.

Tabellnamn som inte finns i appen ignoreras. Om flera förekomster av ett tabellnamn hittas används det senaste värdet. Nyckelordet kan användas för att läsa kommentarer från en datakälla.

Syntax:

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

Argument:

Argument

Argument	Beskrivning
tablelist	(table{,table})
mapname	Namnet på en mappningstabell som tidigare lästs in via en **LOAD**- eller **SELECT**-sats för mappning.
tablename	Namnet på den tabell som ska kommenteras.
comment	Den kommentar som ska läggas till i tabellen.

Example 1:

```
Commentmap:
mapping LOAD * inline [
a,b
Main,This is the fact table
Currencies, Currency helper table
];
comment tables using Commentmap;
```

Example 2:

```
comment table Main with 'Main fact table';
```

Connect

CONNECT-satsen används för att ange Qlik Sense-åtkomst till en allmän databas via OLE DB/ODBC-gränssnittet. För ODBC måste datakällan först anges med hjälp av ODBC-administratören.

Den här funktionen är inte tillgänglig i Qlik Sense SaaS.

Satsen har enbart stöd för mappdatakopplingar i standardläget.

Syntax:

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

Argument:

Argument

Argument	Beskrivning

connect-string	`connect-string ::= datasourcename { ; conn-spec-item }`
	Kopplingssträngen består av namnet på datakällan och om du vill en lista över en eller flera kopplingsspecifikatorer. Om namnet på datakällan innehåller blanktecken eller kopplingsspecifikatorer, måste strängen stå inom citationstecken.
	datasourcename måste vara en angiven ODBC-datakälla eller en sträng som anger en OLE DB-leverantör.
	`conn-spec-item ::=`**`DBQ=`**`database_specifier |`**`DriverID=`**`driver_specifier |`**`UID=`**`userid |`**`PWD=`**`password`
	Vilka kopplingsspecifikatorer som är möjliga varierar mellan olika databaser. För vissa databaser är ytterligare specifikatorer tillåtna. För OLE DB är vissa kopplingsrelaterade poster obligatoriska.

connection	Namnet på en datakoppling som finns sparad i Skriptredigeraren.

Om **ODBC** skrivs framför **CONNECT** används ODBC-gränssnittet. Annars används OLE DB.

Med **LIB CONNECT TO** kopplar du upp dig till en databas med hjälp av en lagrad datakoppling som har skapats i Skriptredigeraren.

Example 1:

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

Datakällan som specificeras genom denna sats används av alla följande **Select (SQL)**-satser tills en ny **CONNECT**-sats påträffas.

Example 2:

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

Connect32

Den här satsen används likadant som **CONNECT**-satsen, men den tvingar ett 64-bitarssystem att använda en 32-bitars ODBC/OLE DB-provider. Kan ej användas för anpassade uppkopplingar.

Connect64

Den här satsen används likadant som **CONNECT**-satsen, men den kräver att en 64-bitars drivrutin används. Kan ej användas för anpassade uppkopplingar.

Declare

Declare-satsen används för att skapa fältdefinitioner där du kan definiera relationer mellan fält eller funktioner. En uppsättning fältdefinitioner kan användas för att automatiskt generera härledda fält, vilka kan användas som dimensioner. Du kan till exempel skapa en kalenderdefinition och använda den för att generera relaterade dimensioner, som år, månad, vecka och dag, från ett datumfält.

Du kan använda **Declare** antingen för att skapa en ny fältdefinition eller en fältdefinition baserad på en befintlig definition.

Skapa en ny fältdefinition

Syntax:

```
definition_name:
Declare [Field[s]] Definition [Tagged tag_list ]
[Parameters parameter_list ]
Fields field_list
```

Argument:

Argument	Beskrivning
definition_name	Fältdefinitionens namn, avslutad med kolon. *Använd inte autoCalendar som namn på fältdefinitioner eftersom detta namn är reserverat för automatiskt genererade kalendermallar.* **Exempel:** `Calendar:`
tag_list	En kommaavgränsad lista över taggar att tillämpa på fält som härletts från fältdefinitionen. Det är valfritt att tillämpa taggar, men om du inte tillämpar taggar för att specificera sorteringsordning, som t.ex. $date, $numeric eller $text, kommer det härledda fältet som standard att sorteras efter laddningsordning. **Exempel:** `'$date'Thank you for bringing this to our attention, and apologies for the inconvenience.`

Argument	Beskrivning
parameter_list	En kommaavgränsad lista över parametrar. En parameter definieras i formen `name=value` och tilldelas ett startvärde, vilket kan åsidosättas när en fältdefinition återanvänds. Valbart. **Exempel:** `first_month_of_year = 1`
field_list	En kommaavgränsad lista över fält att generera när fältdefinitionen används. Ett fält definieras i formen `<expression> As field_name tagged tag`. Använd `$1` för att referera till datafältet som de härledda fälten ska genereras från. **Exempel:** `Year($1) As Year tagged ('$numeric')`

Exempel:

```
Calendar:
DECLARE FIELD DEFINITION TAGGED '$date'
   Parameters
      first_month_of_year = 1
   Fields
      Year($1) As Year Tagged ('$numeric'),
      Month($1) as Month Tagged ('$numeric'),
      Date($1) as Date Tagged ('$date'),
      Week($1) as Week Tagged ('$numeric'),
      Weekday($1) as Weekday Tagged ('$numeric'),
      DayNumberOfYear($1, first_month_of_year) as DayNumberOfYear Tagged ('$numeric')
;
```

Kalendern är nu definierad och du kan tillämpa den på datumfälten som du har laddat, i det här fallet OrderDate och ShippingDate, med en**Derive**-sats.

Återanvända en befintlig fältdefinition

Syntax:

```
<definition name>:
Declare [Field][s] Definition
Using <existing_definition>
[With <parameter_assignment> ]
```

Argument:

Argument	Beskrivning
definition_name	Fältdefinitionens namn, avslutad med kolon. **Exempel:** `MyCalendar:`
existing_definition	Fältdefinitionen som ska återanvändas för att skapa den nya fältdefinitionen. Den nya fältdefinitionen kommer att fungera på samma sätt som definitionen den är baserad på, med undantag för om du använder parameter_assignment för att ändra värdet som används i fältuttrycken. **Exempel:** `Using Calendar`
parameter_assignment	En kommaavgränsad lista över parametertilldelningar. En parametertilldelning definieras i formen `name=value` och åsidosätter parametervärdet som är angett i basfältdefinitionen. Valbart. **Exempel:** `first_month_of_year = 4`

Exempel:

I det här exemplet återanvänder vi kalenderdefinitionen som skapades i föregående exempel. I det här fallet vill vi använda ett budgetår som börjar i april. Detta uppnår vi genom att tilldela värdet 4 till first_month_of_year-parametern, vilket påverkar DayNumberOfYear-fältet som definieras.

I exemplet antas att du använder samma exempeldata och fältdefinition som i föregående exempel.

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

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

När du har laddat dataskriptet blir de genererade fälten tillgängliga i arkredigeraren med namnen OrderDate.MyCalendar.* och ShippingDate.MyCalendar.*.

Derive

Derive-satsen används för att generera härledda fält baserat på en fältdefinition som skapats med en **Declare**-sats. Du kan antingen specifikt ange vilka datafält du vill härleda fält från, eller härleda dem explicit eller implicit baserat på fälttaggar.

Syntax:

```
Derive [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
```

Argument:

Argument

Argument	Beskrivning
definition	Namnet på fältdefinitionen som ska användas för att härleda fält. **Exempel:** `Calendar`
field_list	En kommaavgränsad lista över datafält som de härledda fälten ska genereras från, baserat på fältdefinitionen. Datafälten ska vara fält som du redan har laddat i skriptet. **Exempel:** `OrderDate, ShippingDate`
tag_list	En kommaavgränsad lista över taggar. Härledda fält kommer att genereras för alla datafält med någon av taggarna i listan. Listan över taggar ska omslutas av runda parenteser. **Exempel:** `('$date','$timestamp')`

Exempel:

- Härleda fält för specifika datafält.
  I det här fallet anger vi OrderDate- och ShippingDate-fälten.
  `DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING Calendar;`
- Härleda fält för alla fält med en specifik tagg.
  I det här fallet härleder vi fält baserat på Calendar för alla fält med en `$date`-tagg.
  `DERIVE FIELDS FROM EXPLICIT TAGS ('$date') USING Calendar;`
- Härleda fält för alla fält med fältdefinitionstaggen.
  I det här fallet härleder vi fält för alla datafält med samma tagg som Calendar-fältsdefinitionen, vilken i detta fall är `$date`.
  `DERIVE FIELDS FROM IMPLICIT TAG USING Calendar;`

Direct Query

Med **DIRECT QUERY**-satsen kan du komma åt tabeller genom enODBC eller OLE DB-koppling med hjälp av funktionen Direct Discovery.

Syntax:

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

Nyckelorden **DIMENSION**, **MEASURE** och **DETAIL** kan användas i valfri ordning.

Nyckelordssatserna **DIMENSION** och **FROM** krävs i alla **DIRECT QUERY**-satser. Nyckelordet **FROM** måste komma efter nyckelordet **DIMENSION**.

Fälten som anges direkt efter nyckelordet **DIMENSION** laddas i minnet och kan användas för att skapa associationer mellan data i minnet och Direct Discovery-data.

> *DIRECT QUERY-satsen får inte innehålla DISTINCT- eller GROUP BY-satser.*

Med hjälp av nyckelordet **MEASURE** kan du definiera fält som Qlik Sense kan tolka på "metanivå". Faktiska data för ett measure-fält finns enbart i databasen under dataladdningsprocessen, och hämtas vid behov utifrån de diagramuttryck som används i en visualisering.

Normalt bör fält med diskreta värden som ska användas som dimensioner laddas in med nyckelordet **DIMENSION**, medan tal som ska användas i aggregeringar enbart bör markeras med nyckelordet **MEASURE**.

DETAIL fälten innehåller information eller detaljer, exempelvis fälten ”Comment”, som en användare kan välja att visa i en detaljerad tabellbox. **DETAIL**-fält kan inte användas i diagramuttryck.

DIRECT QUERY-satsen är utformad för att vara datakällsneutral för datakällor som har stöd för SQL. Tack vare detta kan samma **DIRECT QUERY**-sats användas för olika SQL-databaser utan att behöva ändras. Direct Discovery genererar frågor lämpliga för databasen när så krävs.

Programegen syntax för datakällan kan användas när användaren vet vilken databas frågan ska skickas till och vill utnyttja databasspecifika komplement till SQL. Programegen syntax för datakällan stöds:

- Som fältuttryck i **DIMENSION**- och **MEASURE**-satser
- Som innehåll i **WHERE**-satsen

Exempel:

```
DIRECT QUERY
      DIMENSION Dim1, Dim2

      MEASURE
            NATIVE ('X % Y') AS X_MOD_Y
FROM TableName

DIRECT QUERY
      DIMENSION Dim1, Dim2

      MEASURE X, Y

      FROM TableName

      WHERE NATIVE ('EMAIL MATCHES "\*.EDU"')
```

> *Följande termer används som nyckelord och kan inte användas som kolumn- eller fältnamn utan att placeras inom citationstecken: and, as, detach, detail, dimension, distinct, from, in, is, like, measure, native, not, or, where*

Argument:

Argument	Beskrivning
fieldlist	En kommaavgränsad lista med fältspecifikationer, *fieldname {, fieldname}*. En fältspecifikation kan vara ett fältnamn, och i så fall används samma namn som databaskolumnens namn och Qlik Sense-fältnamn. En fältspecifikation kan även vara ett ”fältalias”. Då får ett databasuttryck eller ett kolumnnamn ett Qlik Sense-fältnamn.
tablelist	En lista över namnen på tabeller eller vyer i databasen som data kommer att läsas in från. Normalt är det vyer som innehåller en JOIN som utförs på databasen.

where_ clause	Den fullständiga syntaxen för **WHERE**-databassatser anges inte här, men de flesta ”SQL- relationsuttryck” är tillåtna, inklusive användning av funktionsanrop, **LIKE**-operatorn för strängar, **IS NULL** och **IS NOT NULL** och **IN. BETWEEN** ingår inte.
	
	NOT är en unär operator i motsats till en modifierare för vissa nyckelord.
	
	Exempel:
	
	`WHERE x > 100 AND "Region Code" IN ('south', 'west')`
	
	`WHERE Code IS NOT NULL and Code LIKE '%prospect'`
	
	`WHERE NOT X in (1,2,3)`
	Det sista exemplet kan inte skrivas som:
	
	`WHERE X NOT in (1,2,3)`

Exempel:

I det här exemplet används en databastabell som heter TableName och innehåller fälten Dim1, Dim2, Num1, Num2 och Num3.Dim1 och Dim2 laddas in i datauppsättningen Qlik Sense.

```
DIRECT QUERY DIMENSTION Dim1, Dim2 MEASURE Num1, Num2, Num3 FROM TableName ;
```

Dim1 och Dim2 blir tillgängliga att använda som dimensioner. Num1, Num2 och Num3 blir tillgängliga för aggregeringar. Dim1 och Dim2 är också tillgängliga för aggregeringar. Vilken typ av aggregeringar Dim1 och Dim2 kan användas för beror på deras datatyper. I många fall innehåller till exempel **DIMENSION**-fält strängdata som namn eller kontonummer. Dessa fält kan inte läggas ihop, men de kan räknas: `count (Dim1)`.

__DIRECT QUERY__-satser skrivs direkt i Skriptredigeraren. För att underlätta vid konstruktionen av __DIRECT QUERY__-satser kan du generera en __SELECT__-sats från en dataanslutning och sedan redigera det genererade skriptet för att ändra det till en __DIRECT QUERY__-sats.
Till exempel kan __SELECT__-satsen:

```
SQL SELECT
 SalesOrderID,
 RevisionNumber,
 OrderDate,
 SubTotal,
 TaxAmt
FROM MyDB.Sales.SalesOrderHeader;
```

ändras till följande __DIRECT QUERY__-sats:

```
DIRECT QUERY
 DIMENSION
 SalesOrderID,
 RevisionNumber

MEASURE
 SubTotal,
 TaxAmt

DETAIL
OrderDate

FROM MyDB.Sales.SalesOrderHeader;
```

Direct Discovery fältlistor

En fältlista är en kommaseparerad lista med fältspecifikationer, *fieldname {, fieldname}*. En fältspecifikation kan vara ett fältnamn, och i så fall används samma namn som databaskolumnens namn och fältnamn. En fältspecifikation kan även vara ett fältalias. Då får ett databasuttryck eller ett kolumnnamn ett Qlik Sense-fältnamn.

Fältnamn kan vara antingen enkla namn eller namn inom citationstecken. Ett enkelt namn börjar med ett alfabetiskt Unicode-tecken och följs av valfri kombination av alfabetiska eller numeriska tecken eller understreck. Namn inom citationstecken börjar med dubbla citationstecken och innehåller en valfri följd av tecken. Om ett namn inom citationstecken innehåller citationstecken, representeras dessa av två citationstecken bredvid varandra.

Qlik Sense-fältnamn är skiftlägeskänsliga. Databasfältnamn kan vara skiftlägeskänsliga eller inte, beroende på databasen. En Direct Discovery-fråga bevarar skiftläget för alla fältidentifierare och alias. I följande exempel används aliaset "MyState" internt för att lagra data från databaskolumnen "STATEID".

```
DIRECT QUERY Dimension STATEID as MyState Measure AMOUNT from SALES_TABLE;
```

Detta skiljer sig från resultatet av en **SQL Select**-sats med ett alias. Om inte aliaset explicit är inom citationstecken innehåller resultatet standardskiftläget för kolumnen som returneras av måldatabasen. I följande exempel skapar **SQL Select**-satsen till en Oracle-databas "MYSTATE," med endast stora bokstäver som det interna Qlik Sense-aliaset, trots att aliaset är angivet med både stora och små bokstäver. **SQL Select**-satsen använder kolumnnamnet som returneras av databasen, och för Oracle har det bara stora bokstäver.

```
SQL Select STATEID as MyState, STATENAME from STATE_TABLE;
```

För att undvika detta kan du använda LOAD-satsen för att ange aliaset.

```
Load STATEID as MyState, STATENAME;
SQL Select STATEID, STATEMENT from STATE_TABLE;
```

I det här exemplet lagras "STATEID"-kolumnen internt av Qlik Sense som "MyState".

De flesta skalära databasuttryck är tillåtna som fältspecifikationer. Funktionsanrop kan också användas i fältspecifikationer. Uttryck kan innehålla konstanter som är booleska, numeriska eller strängar inom enkla citationstecken (inbäddade enkla citationstecken representeras av enkla citationstecken bredvid varandra).

Exempel:

```
DIRECT QUERY

   DIMENSION

      SalesOrderID, RevisionNumber

   MEASURE

      SubTotal AS "Sub Total"

FROM AdventureWorks.Sales.SalesOrderHeader;


DIRECT QUERY

   DIMENSION

      "SalesOrderID" AS "Sales Order ID"

   MEASURE

      SubTotal,TaxAmt,(SubTotal-TaxAmt) AS "Net Total"

FROM AdventureWorks.Sales.SalesOrderHeader;


DIRECT QUERY

   DIMENSION
```

```
            (2*Radius*3.14159) AS Circumference,

            Molecules/6.02e23 AS Moles

         MEASURE

            Num1 AS numA

      FROM TableName;
```

```
DIRECT QUERY
   DIMENSION
      concat(region, 'code') AS region_code
   MEASURE
      Num1 AS NumA
FROM TableName;
```

Direct Discovery stöder inte användning av aggregeringar i **LOAD**-satser. Om aggregeringar används blir resultaten oförutsägbara. En **LOAD**-sats som följande bör inte användas:

```
DIRECT QUERY DIMENSION stateid, SUM(amount*7) AS MultiFirst MEASURE amount FROM sales_table;
```

SUM bör inte finnas i **LOAD**-satsen.

Direct Discovery stöder inte heller Qlik Sense-funktioner i **Direct Query**-satser. Till exempel resulterar följande specifikation för ett **DIMENSION**-fält i misslyckande när "Mth"-fältet används som en dimension i en visualisering:

```
month(ModifiedDate) as Mth
```

Directory

Directory-satsen anger i vilken katalog datafilerna ska sökas i följande **LOAD**-satser tills en ny **Directory**-sats anges.

Syntax:

```
Directory[path]
```

Om **Directory**-satsen används utan någon **path** eller utelämnas kommer Qlik Sense att leta i Qlik Sense-arbetskatalogen.

Argument:

Argument

Argument	Beskrivning
path	En text som kan tolkas som sökvägen till data-filen. Sökvägen är sökvägen till filen, antingen: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense-appens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*

Exempel:

```
DIRECTORY C:\userfiles\data; // OR -> DIRECTORY data\

LOAD * FROM
[data1.csv] // ONLY THE FILE NAME CAN BE SPECIFIED HERE (WITHOUT THE FULL PATH)
(ansi, txt, delimiter is ',', embedded labels);

LOAD * FROM
[data2.txt] // ONLY THE FILE NAME CAN BE SPECIFIED HERE UNTIL A NEW DIRECTORY STATEMENT IS
MADE
(ansi, txt, delimiter is '\t', embedded labels);
```

Disconnect

Disconnect-satsen avslutar den aktuella ODBC-kopplingen/OLE DB-kopplingen/anpassade kopplingen. Denna sats är valfri.

Syntax:

```
Disconnect
```

Kopplingen bryts automatiskt när en ny **connect**-sats exekveras eller när skriptexekveringen är avslutad.

Exempel:

```
Disconnect;
```

Drop

Skriptnyckelordet **Drop** kan användas för att avlägsna tabeller eller fält från databasen.

Drop field

Ett eller flera Qlik Sense-fält kan avlägsnas från datamodellen, och därigenom från minnet, när som helst under skriptexekveringen med hjälp av en **drop field**-sats.

> *Både **drop field** och **drop fields** är tillåtna och har samma betydelse. Om ingen tabell har angetts avlägsnas fältet från alla tabeller där det förekommer.*

Syntax:

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

Exempel:

```
Drop field A;
Drop fields A,B;
Drop field A from X;
Drop fields A,B from X,Y;
```

Drop table

Genom att använda en **drop table**-sats kan man när som helst under skriptexekveringen avlägsna en eller flera interna tabeller i Qlik Sense ur datamodellen och därmed ur minnet.

Syntax:

```
drop table  tablename {, tablename2 ...}
drop tables tablename {, tablename2 ...}
```

> *Formerrna **drop table** och **drop tables** accepteras båda.*

Följande går förlorat till följd av:

- De faktiska tabellerna.
- Alla fält som inte ingår i kvarvarande tabeller.
- Fältvärden i kvarvarande fält som härstammar enbart från borttagna tabeller

Exempel och resultat:

Exempel	Resultat
`drop table Orders, Salesmen, T456a;`	Resulterar i att tre tabeller avlägsnas från minnet.

`Tab1:`
`Load * Inline [`
`Customer, Items, UnitPrice`
`Bob, 5, 1.50`
`];`
 	
`Tab2:`
`LOAD Customer, Sum( Items * UnitPrice ) as Sales`
`resident Tab1`
`group by Customer;`
 	
`drop table Tab1;`
	När tabellen *Tab2* har skapats utelämnas tabell *Tab1*.

Drop table

Genom att använda en **drop table**-sats kan man när som helst under skriptexekveringen avlägsna en eller flera interna tabeller i Qlik Sense ur datamodellen och därmed ur minnet.

Syntax:

```
drop table  tablename {, tablename2 ...}
drop tables tablename {, tablename2 ...}
```

*Formerrna **drop table** och **drop tables** accepteras båda.*

Följande går förlorat till följd av:

- De faktiska tabellerna.
- Alla fält som inte ingår i kvarvarande tabeller.
- Fältvärden i kvarvarande fält som härstammar enbart från borttagna tabeller

Exempel och resultat:

Exempel	Resultat
`drop table Orders, Salesmen, T456a;`	Resulterar i att tre tabeller avlägsnas från minnet.

Exempel	Resultat
Tab1: Load * Inline [Customer, Items, UnitPrice Bob, 5, 1.50]; Tab2: LOAD Customer, Sum(Items * UnitPrice) as Sales resident Tab1 group by Customer; drop table Tab1;	När tabellen *Tab2* har skapats utelämnas tabell *Tab1*.

Execute

Execute-satsen använda för att köra andra program medan Qlik Sense laddar data, exempelvis för att göra de konverteringar som krävs.

Den här funktionen är inte tillgänglig i Qlik Sense SaaS.

Satsen stöds inte i standardläget.

Syntax:

```
execute commandline
```

Argument:

Argument

Argument	Beskrivning
commandline	En text som kan tolkas som en kommandorad av operativsystemet. Du kan hänvisa till en absolut sökväg eller en sökväg till katalogen lib://.

Om du vill använda **Execute** måste följande villkor vara uppfyllda:

- Du måste köra i bakåtkompatibelt läge (tillämpligt för Qlik Sense och Qlik Sense Desktop).
- Du måste ställa in OverrideScriptSecurity på 1 i *Settings.ini* (gäller för Qlik Sense). *Settings.ini* finns under *C:\ProgramData\Qlik\Sense\Engine* och är normalt en tom fil.

*Om du ställer in OverrideScriptSecurity för att aktivera **Execute** kan alla användare exekvera filer på servern. En användare kan exempelvis bifoga en exekverbar fil till en app, och sedan exekvera filen i dataladdningsskriptet.*

Gör följande:

1. Gör en kopia av *Settings.ini* och öppna den i en textredigerare.
2. Kontrollera att filen innehåller *[Settings 7]* på första raden.
3. Infoga en ny rad och skriv in *OverrideScriptSecurity=1*.
4. Lägg till en ny rad i slutet av filen.
5. Spara filen.
6. Byt ut *Settings.ini* mot den redigerade filen.
7. Starta om Qlik Sense Engine Service (QES).

Om Qlik Sense körs som en tjänst kan en del kommandon ha ett annat beteende än väntat.

Exempel:

```
Execute C:\Program Files\Office12\Excel.exe;

Execute lib://win\notepad.exe // win is a folder connection referring to c:\windows
```

Field/Fields

Skriptnyckelorden **Field** och **Fields** används i **Declare**-, **Derive**-, **Drop**-, **Comment**-, **Rename**- och **Tag/Untag**-satser.

FlushLog

Satsen **FlushLog** tvingar Qlik Sense att skriva innehållet i skriptbufferten till skriptets loggfil.

Syntax:

```
FlushLog
```

Innehållet i bufferten skrivs till loggfilen. Detta kommando kan vara användbart för felsökning eftersom du får data som annars hade kunnat gå förlorade när en skriptexekvering inte lyckas.

Exempel:

```
FlushLog;
```

Force

force-satsen tvingar Qlik Sense att tolka fältnamn och fältvärden i efterföljande **LOAD**- och **SELECT**-satser som att de är skrivna med enbart versaler, enbart gemener, alltid med inledande versaler eller som de uppträder (blandat). Med hjälp av denna sats är det möjligt att koppla fält från tabeller med olika konventioner.

Syntax:

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

Standard är force case mixed. Konventionen som specificeras genom denna sats används fram tills en ny force-sats påträffas.

I behörighetssektionen har **force**-satsen ingen effekt: ingen skillnad görs här mellan versaler och gemener vid laddning av fälten.

Exempel och resultat

Exempel	Resultat

Det här exemplet visar hur du tvingar fram inledande versal.
`FORCE Capitalization;`
`Capitalization:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Tabellen **Capitalization** innehåller följande värden:
	`Ab`
	`Cd`
	`Ef`
	`Gh`
	Alla värden inleds med versal.

Det här exemplet visar hur du tvingar fram versaler.
`FORCE Case Upper;`
`CaseUpper:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	CaseUpper-tabellen innehåller följande värden:
	`AB`
	`CD`
	`EF`
	`GH`
	Alla värden är versala.

Det här exemplet visar hur du tvingar fram gemener.
`FORCE Case Lower;`
`CaseLower:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	CaseLower-tabellen innehåller följande värden:
	`ab`
	`cd`
	`ef`
	`gh`
	Alla värden är gemena.

Det här exemplet visar hur du tvingar fram både gemener och versaler.
`FORCE Case Mixed;`
`CaseMixed:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	CaseMixed-tabellen innehåller följande värden:
	`ab`
	`Cd`
	`eF`
	`GH`
	Alla värden ser ut som de visas i skriptet.

Se även:

From

Skriptnyckelordet **From** används i **Load**-satserna för att hänvisa till en fil och i **Select**-satserna för att hänvisa till en datatabell eller -vy.

Load

LOAD-satsen laddar fält från en fil, direkt från data i skriptet, från en tidigare inläst tabell, från en webbsida, från resultatet av en efterföljande **SELECT**-sats eller genom att generera data automatiskt. Det går även att ladda data från analyskopplingar.

Syntax:

```
LOAD [ distinct ] fieldlist
[( from file [ format-spec ] |
from_field fieldassource [format-spec]|
inline data [ format-spec ] |
resident table-label |
autogenerate size ) |extension pluginname.functionname([script]
tabledescription)]
[ where criterion | while criterion ]
[ group by groupbyfieldlist ]
[order by orderbyfieldlist ]
```

Argument:

Argument

Argument	Beskrivning
distinct	Du kan använda **distinct** som predikat om du bara vill ladda unika poster. Om det finns dubblettposter laddas den första. Om du använder föregående load måste du placera **distinct** i den första load-satsen, eftersom **distinct** bara inverkar på destinationstabellen.

Argument	Beskrivning
fieldlist	*fieldlist ::= (* | field {, * | field })* En lista på de fält som ska läsas in. Genom att använda * som fältlista anger man alla fält i tabellen. *field ::= (fieldref | expression)* [**as** *aliasname]* Fältdefinitionen måste alltid innehålla en litteral, en referens till ett befintligt fält eller ett uttryck. *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [**I** | **U** | **R** | **B** | **T**])* *fieldname* är en text som är identisk med fältnamnet i tabellen. Observera att fältnamnet måste omslutas av raka, dubbla citationstecken eller av hakparenteser om det innehåller exempelvis mellanslag. Ibland är fältnamn inte explicit tillgängliga. Då används en annan metod: *@fieldnumber* representerar fältnumret i en avgränsad tabellfil. Det måste vara ett positivt heltal som föregås av "@". Numreringen går alltid från 1 och upp till antalet fält. *@startpos*:*endpos* representerar första och sista teckenposition för fältet i en fixpostfil med fasta fältpositioner. Positionerna måste vara positiva heltal. De två talen måste föregås av "@" och avgränsas av kolon. Numreringen går alltid från 1 och upp till antalet positioner. I det sista fältet används **n** som slutposition. • Om*@startpos*:*endpos* omedelbart följs av tecknen **I** eller**U**, tolkas inlästa bytes som binärt signerade (**I**) eller osignerade (**U**) heltal (Intel byte order). Antalet positioner som läses måste vara 1, 2 eller 4. • Om *@startpos*:*endpos* omedelbart följs av tecknet **R**, tolkas inlästa bytes som ett binärt reellt tal (IEEE 32-bitars eller 64-bitars floating point). Antalet positioner som läses in måste vara 4 eller 8. • Om *@startpos*:*endpos* omedelbart följs av tecknet **B**, tolkas inlästa bytes som BCD (Binary Coded Decimal)-tal enligt COMP-3-standard. Valfritt antal bytes kan anges. *expression* kan vara en numerisk funktion eller en strängfunktion baserad på ett eller flera andra fält i samma tabell. För ytterligare information, se uttryckens syntax. **as** används för att döpa om fält.

Argument	Beskrivning
from	**from** används om data ska läsas in från en fil med en mapp eller webbfilsdatakoppling. *file ::= [path] filename* **Exempel:** ***'lib://Table Files/'*** Om sökvägen utelämnas söker Qlik Sense efter filen i den mapp som specificerats av en **Directory**-sats. Om det inte finns någon **Directory**-sats söker Qlik Sense i arbetskatalogen, *C:\Users\{user}\Documents\Qlik\Sense\Apps*. *I en Qlik Sense-serverinstallation anges arbetskatalogen i Qlik Sense Repository Service. Som standard är det C:\ProgramData\Qlik\Sense\Apps.* *filename* kan innehålla standardiserade jokertecken från DOS (* och ?). Detta får alla matchande filer i den angivna katalogen att läsas in. *format-spec ::= (fspec-item { , fspec-item })* Formatspecifikationen består av en lista med flera formatspecifikatorer inom parentes. **Bakåtkompatibel skriptkod** I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** ***c:\data*** • relativ till Qlik Sense-appens arbetskatalog. **Exempel:** ***data*** • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** ***http://www.qlik.com***
from_field	**from_field** används om data ska läsas in från ett tidigare inläst fält. *fieldassource::=***(***tablename, fieldname***)*** Fältet är namnet på tidigare inlästa *tablename* och *fieldname*. *format-spec ::= (fspec-item {, fspec-item })* Formatspecifikationen består av en lista med flera formatspecifikatorer inom parentes.

Argument	Beskrivning
inline	**inline** används om data ska skrivas i skriptet, och inte läsas in från en fil. *data ::= [text]* Data som läses in med en **inline**-sats måste inledas och avslutas med dubbla citationstecken eller hakparenteser. Texten mellan dessa tolkas som om den vore skriven i en fil. Precis som du infogar en ny rad i en textfil bör du göra det även i texten i en **inline**-sats. Klicka på vanligt sätt på returtangenten när du skriver skriptet. Antalet kolumner definieras av den första raden. *format-spec ::= (fspec-item {, fspec-item })* Formatspecifikationen består av en lista med flera formatspecifikatorer inom parentes.
resident	**resident** används om data ska läsas in från en tidigare inläst tabell. *table label* är en etikett som föregår de **LOAD**- eller **SELECT**-satser som skapade den ursprungliga tabellen. Etiketten ska avslutas med kolon.
autogenerate	**autogenerate** används när data ska genereras automatiskt av Qlik Sense. *size ::= number* *Number* är ett heltal som anger antalet poster som ska genereras. Fältlistan får inte innehålla uttryck som behöver data från en extern datakälla eller från en tidigare laddad tabell, om du inte hänvisar till ett enda fältvärde i en tidigare laddad tabell med hjälp av **Peek**-funktionen.

Argument	Beskrivning
extension	Du kan ladda data från analytiska kopplingar. Du måste använda **extension**-satsen för att anropa en funktion som definierats i SSE-insticksprogrammet (för komplement på serversidan) eller utvärdera ett skript. Du kan skicka en enstaka tabell till SSE-insticksprogrammet. En datatabell returneras. Fälten kallas Field1, Field2 och så vidare om insticksprogrammet inte specificerar namnen på de fält som returneras. `Extension pluginname.functionname( tabledescription );` • Ladda data med en funktion i ett SSE-insticksprogram *tabledescription ::= (table { ,tablefield})* Om du inte anger tabellfält används fälten i laddningsordning. • Ladda data genom att utvärdera ett skript i ett SSE-insticksprogram *tabledescription ::= (script, table { ,tablefield})* **Hantering av datatyp i tabellfältdefinitionen** Datatyper identifieras automatiskt i analytiska kopplingar. Om data inte har numeriska värden och minst en icke-NULL textsträng betraktas fältet som text. I alla andra fall betraktas det som numeriskt. Du kan tvinga datatypen genom att innesluta ett fältnamn med **String()** eller **Mixed()**. • **String()** tvingar fältet att vara text. Om fältet är numeriskt extraheras textdelen av det duala värdet. Ingen konvertering görs. • **Mixed()** tvingar fältet att vara dualt. **String()** eller **Mixed()** kan inte användas utanför **extension**-tabellfältdefinitioner och du kan inte använda andraQlik Sense-funktioner i en tabellfältdefinition. **Mer om analytiska kopplingar** Du måste konfigurera analytiska kopplingar innan du kan använda dem.
where	**where** är ett tillägg som används för att tala om huruvida en post ska inkluderas i valet eller inte. Valet inkluderas om *criterion* är True. *criterion* är ett logiskt uttryck.
while	**while** är en sats som används för att tala om när en post ska läsas in upprepade gånger. Samma post läses in så länge *criterion* är True. För att vara användbar måste en **while**-sats typiskt sett innehålla en **IterNo( )**-funktion. *criterion* är ett logiskt uttryck.

Argument	Beskrivning
group by	**group by** används för att definiera över vilka fält data ska aggregeras (grupperas). Aggregeringsfälten ska på något sätt inlemmas i de uttryck som läses in. Inga andra fält än aggregeringsfälten får användas utanför aggregeringsfunktionerna i inläsningsuttrycken. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	**order by** är en sats som används för att sortera poster i en resident tabell innan de bearbetas av **load**-satsen. Den residenta tabellen kan sorteras efter ett eller flera fält, i stigande eller sjunkande ordning. Sortering görs i första hand efter talvärde, i andra hand efter nationell sorteringsspråkvariant. Satsen är endast användbar när datakällan är en resident tabell. Ordningsföljdsfälten anger efter vilket fält resident-tabellen sorteras. Ange fältets namn eller dess nummer i resident-tabellen (det första fältet får nummer 1). orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } *sortorder* är antingen *asc* för stigande eller *desc* för fallande. Om ingen *sortorder* anges, antas *asc*. *fieldname*, *path*, *filename* och *aliasname* är textsträngar som representerar det namnen antyder. Valfritt fält i källtabellen kan användas som *fieldname*. Däremot hamnar fält som har skapats med hjälp av as-satsen (*aliasname*) utanför och kan inte användas inuti samma **load**-sats.

Om ingen datakälla anges genom **from**-, **inline**-, **resident**-, **from_field, extension**- eller **autogenerate**-satser, kommer data att läsas in från resultatet av närmast efterföljande **SELECT**- eller **LOAD**-sats. Den efterföljande satsen får inte ha något prefix.

Exempel:

Läsa in olika filformat
Ladda en avgränsad data fil utan standardalternativ:

```
LOAD * from data1.csv;
```

Ladda en avgränsad datafil från en bibliotekskoppling (DataFiles):

```
LOAD * from 'lib://DataFiles/data1.csv';
```

Ladda alla avgränsade datafiler från en bibliotekskoppling (DataFiles):

```
LOAD * from 'lib://DataFiles/*.csv';
```

Ladda en avgränsad fil, med komma som angiven avgränsare och inbäddade etiketter:

```
LOAD * from 'c:\userfiles\data1.csv' (ansi, txt, delimiter is ',', embedded labels);
```

Ladda en avgränsad fil, med tabb som angiven avgränsare och inbäddade etiketter:

```
LOAD * from 'c:\userfiles\data2.txt' (ansi, txt, delimiter is '\t', embedded labels);
```

Ladda en dif-fil med inbäddade rubriker:

```
LOAD * from file2.dif (ansi, dif, embedded labels);
```

Ladda tre fält från en fil med fasta poster utan rubriker:

```
LOAD @1:2 as ID, @3:25 as Name, @57:80 as City from data4.fix (ansi, fix, no labels, header is
0, record is 80);
```

Ladda en QVX-fil som specificerar en absolut sökväg:

```
LOAD * from C:\qdssamples\xyz.qvx (qvx);
```

Ladda webbfiler
Ladda från en standard-URL som anges i webbfildatakopplingen:

```
LOAD * from [lib://MyWebFile];
```

Ladda från en specifik URL och åsidosätt URL:en som anges i webbfildatakopplingen:

```
LOAD * from [lib://MyWebFile] (URL is 'http://localhost:8000/foo.bar');
```

Ladda från en specifik URL som anges i en variabel med dollarteckenexpansion:

```
SET dynamicURL = 'http://localhost/foo.bar';
LOAD * from [lib://MyWebFile] (URL is '$(dynamicURL)');
```

Välja vissa fält, döpa om och beräkna fält
Ladda endast tre specifika fält från en avgränsad fil:

```
LOAD FirstName, LastName, Number from data1.csv;
```

Döp om fet första fältet som A och det andra fältet som B när du laddar en fil utan etiketter:

```
LOAD @1 as A, @2 as B from data3.txt (ansi, txt, delimiter is '\t', no labels);
```

Ladda Name som en konkatenering av FirstName, ett blanksteg och LastName:

```
LOAD FirstName&' '&LastName as Name from data1.csv;
```

Ladda Quantity, Price och Value (produkten av Quantity och Price):

```
LOAD Quantity, Price, Quantity*Price as Value from data1.csv;
```

Läsa in vissa poster
Ladda endast unika poster, dubblettposter kommer att uteslutas:

```
LOAD distinct FirstName, LastName, Number from data1.csv;
```

Ladda endast poster där fältet Litres har ett värde över noll:

```
LOAD * from Consumption.csv where Litres>0;
```

Ladda data som inte finns i filer samt automatiskt genererade data
Ladda en tabell med inline-data, två fält som kallas CatID och Category:

```
LOAD * Inline
[CatID, Category
0,Regular
1,Occasional
2,Permanent];
```

Ladda en tabell med inline-data, tre fält som kallas UserID, Password och Access:

```
LOAD * Inline [UserID, Password, Access
A, ABC456, User
B, VIP789, Admin];
```

Ladda en tabell med 10 000 rader. Fält A kommer innehålla talet för posten som ska laddas (1,2,3,4,5...) och fältet B innehåller ett slumpmässigt nummer mellan 0 och 1:

```
LOAD RecNo( ) as A, rand( ) as B autogenerate(10000);
```

Parentesen efter autogenerate är tillåten men inte obligatorisk.

Läsa in data från en tidigare inläst tabell
Först laddar vi en avgränsad tabellfil och kallar dentab1:

```
tab1:
SELECT A,B,C,D from 'lib://DataFiles/data1.csv';
```

Ladda fält från den redan laddade tabellen tab1 som tab2:

```
tab2:
LOAD A,B,month(C),A*B+D as E resident tab1;
```

Ladda fält från den redan laddade tabellen tab1 men bara poster där A är större än B:

```
tab3:
LOAD A,A+B+C resident tab1 where A>B;
```

Ladda fält som den redan laddade tabellentab1 ordnade efter A:

```
LOAD A,B*C as E resident tab1 order by A;
```

Ladda fält från den redan laddade tabellen tab1, ordnade efter det första fältet, sedan efter det andra fältet:

```
LOAD A,B*C as E resident tab1 order by 1,2;
```

Ladda fält från den redan laddade tabellentab1 ordnade efter C i fallande ordning, sedan efter B i stigande ordning och därefter efter det första fältet i fallande ordning:

```
LOAD A,B*C as E resident tab1 order by C desc, B asc, 1 desc;
```

Läsa in data från tidigare inlästa fält
Ladda fältet Types från den tidigare laddade tabellen Characters som A:

```
LOAD A from_field (Characters, Types);
```

Ladda data från en efterföljande tabell (föregående laddning)

Ladda A, B och de beräknade fälten X och Y frånTable1 som laddas i efterföljande **SELECT**-sats:

```
LOAD A, B, if(C>0,'positive','negative') as X, weekday(D) as Y;
SELECT A,B,C,D from Table1;
```

Gruppera data

Ladda fält grupperade (aggregerade) efter ArtNo:

```
LOAD ArtNo, round(Sum(TransAmount),0.05) as ArtNoTotal from table.csv group by ArtNo;
```

Ladda fält grupperade (aggregerade) efter Week och ArtNo:

```
LOAD Week, ArtNo, round(Avg(TransAmount),0.05) as WeekArtNoAverages from table.csv group by
Week, ArtNo;
```

Upprepad inläsning av en post

I detta exempel har vi indatafilen Grades.csv som innehåller betygen för varje student, kondenserat till ett fält:

```
Student,Grades
Mike,5234
John,3345
Pete,1234
Paul,3352
```

Betygen, i en skala från 1 till 5, motsvarar Math, English, Science ochHistory. Vi kan dela upp betygen i separata värden genom att läsa in varje post flera gånger med en **while**-sats och använda **IterNo( )**-funktionen som räknare. Vid varje inläsning extraheras betyget med funktionen **Mid** och lagras i Grade och ämnet väljs med hjälp av funktionen **pick** och lagras i Subject. Den slutliga **while**-satsen innehåller testet för att kontrollera om alla betyg har lästs (fyra per student i det här fallet), vilket innebär att nästa studentpost ska läsas.

```
MyTab:
LOAD Student,
mid(Grades,IterNo( ),1) as Grade,
pick(IterNo( ), 'Math', 'English', 'Science', 'History') as Subject from Grades.csv
while IsNum(mid(Grades,IterNo(),1));
```

Resultatet blir en tabell med dessa data:

Student	Subject	Grade
John	English	3
John	History	5
John	Math	3
John	Science	4
Mike	English	2
Mike	History	4
Mike	Math	5
Mike	Science	3
Paul	English	3
Paul	History	2
Paul	Math	3
Paul	Science	5
Pete	English	2
Pete	History	4
Pete	Math	1
Pete	Science	3

Ladda från analytiska kopplingar
Följande exempeldata används.

```
Values:
Load
 Rand() as A,
 Rand() as B,
 Rand() as C
AutoGenerate(50);
```

Ladda data med en funktion

I dessa exempel antar vi att vi har ett instickspogram med analytisk koppling som heter *P* som innehåller den anpassade funktionen *Calculate(Parameter1, Parameter2)*. Funktionen returnerar tabellen *Results* som innehåller fälten *Field1* och *Field2*.

```
Load * Extension P.Calculate( Values{A, C} );
```
Ladda alla fält som returneras när fälten A och C skickas till funktionen.

```
Load Field1 Extension P.Calculate( Values{A, C} );
```
Ladda bara Field1-fältet när fälten A och C skickas till funktionen.

```
Load * Extension P.Calculate( Values );
```
Ladda alla fält som returneras när fälten A och B skickas till funktionen. Eftersom fält inte är angivna används A och B eftersom de står först i tabellordningen.

```
Load * Extension P.Calculate( Values {C, C});
```
Ladda alla fält som returneras när fältet C skickas till funktionens båda parametrar.

```
Load * Extension P.Calculate( Values {String(A), Mixed(B)});
```
Ladda alla fält som returneras när fält A skickas tvingad till sträng och fält B skickas tvingad till numerisk till funktionen.

Ladda data genom att utvärdera ett skript

`Load A as A_echo, B as B_echo Extension R.ScriptEval( 'q;', Values{A, B} );`
Ladda tabellen som returneras av q-skriptet när värdena av A och B skickas.

`Load * Extension R.ScriptEval( '$(My_R_Script)', Values{A, B} );`
Ladda tabellen som returneras av skriptet som lagras i My_R_Script-variabeln när värdena av A och B skickas.

`Load * Extension R.ScriptEval( '$(My_R_Script)', Values{B as D, *} );`
Ladda tabellen som returneras av skriptet som lagras i My_R_Script-variabeln när värdena av B som döpts om till D, A och C skickas. Använd * för att skicka återstående orefererade fält.

Filtillägget för DataFiles-kopplingar är skiftlägeskänsligt. Till exempel: .qvd.

Formatspecifikatorer

Varje formatspecifikator definierar en viss egenskap i tabellfilen:

fspec-item ::= [**ansi | oem | mac | UTF-8 | Unicode | txt | fix | dif | biff | ooxml | html | xml | kml | qvd | qvx | delimiter is** `char` | **no eof | embedded labels | explicit labels | no labels | table is [**`tablename`**] | header is** `n` | **header is** `line` | **header is** `n` **lines** | **comment is** `string` | **record is** `n` | **record is** `line` | **record is** `n` **lines** | **no quotes |msq | URL is** `string` | **userAgent is** `string`]

Teckenuppsättning

Teckenuppsättning är en filspecifikator för **LOAD**-satsen som anger vilken teckenuppsättning som används i filen.

Specifikatorerna **ansi**, **oem** och **mac** användes i QlikView och fungerar fortfarande. De kommer dock inte att genereras när du skapar **LOAD**-satsen i Qlik Sense.

Syntax:

```
utf8 | unicode | ansi | oem | mac | codepage is
```

Argument:

Argument

Argument	Beskrivning
utf8	UTF-8-teckenuppsättning
unicode	Unicode-teckenuppsättning
ansi	Windows (kodsida 1252)
oem	DOS, OS/2, AS400 med mera
mac	Kodsida 10000
codepage is	Med specifikatorn **codepage** går det att använda valfri kodsida för Windows som *N*.

Begränsningar:

Konvertering från **oem**-teckenuppsättningen är inte implementerad för MacOS. Om ingenting anges antas 1252 under Windows.

Exempel:

```
LOAD * from a.txt (utf8, txt, delimiter is ',' , embedded labels)
LOAD * from a.txt (unicode, txt, delimiter is ',' , embedded labels)
LOAD * from a.txt (codepage is 10000, txt, delimiter is ',' , no labels)
```

Se även:

Load (page 97)

Table format

Tabellformatet är en filspecifikator för **LOAD**-satsen som anger vilken filtyp det gäller. Om inget annat är angivet antas standardformatet *.txt*.

Tabellformatstyper

Typ	Beskrivning
txt	I en avgränsad textfil är kolumnerna avgränsade av ett avgränsartecken.
fix	I en textfil med fasta postlängder har varje kolumn en bredd på ett visst antal tecken. Vanligtvis innehåller filer med fasta postlängder poster som avgränsas av en radmatning, men det finns mer avancerade alternativ för att ange poststorlek i byte eller för att låta dem sträcka sig över mer än en rad med hjälp av **Record is**. *Om data innehåller tecken på flera byte kan fältbrytningarna bli feljusterade eftersom formatet baserar på en fast längd i byte.*
dif	I en *.dif*-fil (Data Interchange Format) används ett speciellt format för att definiera tabellen.
biff	Qlik Sense kan även tolka data i standard-Excel-filer med hjälp av *biff*-formatet (Binary Interchange File Format).
ooxml	Excel 2007 och senare versioner använder ooxml *.xslx*-formatet.
html	Om tabellen ingår i en html-sida eller fil bör html användas.
xml	xml (Extensible Markup Language) är ett vanligt märkspråk som används för att representera datastrukturer i textformat.
qvd	Formatet *qvd* är det egna filformatet QVD, exporterat från en Qlik Sense-app.

Typ	Beskrivning
qvx	*qvx* är ett filformat/streamingformat för att leverera utdata med hög prestanda till Qlik Sense.

Delimiter is

För avgränsade tabellfiler kan en valfri avgränsare anges med hjälp av **delimiter is**-specifikator. Denna specifikator är endast relevant för avgränsade .txt-filer.

Syntax:

```
delimiter is char
```

Argument:

Argument

Argument	Beskrivning
char	Anger ett enstaka tecken av de 127 ASCII-tecknen.

Dessutom kan följande värden användas:

Valfria värden

Värde	Beskrivning
'\t'	motsvarar ett tabbtecken, med eller utan citattecken.
'\\'	motsvarar ett omvänt snedstreck (\).
'spaces'	motsvarar alla kombinationer av ett eller flera mellanslag. Tecken som inte kan skrivas ut och har ett ASCII-värde under 32, undantaget CR och LF, tolkas som mellanslag.

Om inget anges, antas **delimiter is ','**.

Exempel:

```
LOAD * from a.txt (utf8, txt, delimiter is ',' , embedded labels);
```

Se även:

Load (page 97)

No eof

Specifikatorn **no eof** används för att ignorera tecknet som betecknar filslutsmarkering när du läser in avgränsade **.txt**-filer.

Syntax:

```
no eof
```

Om **no eof**-specificeraren används ignoreras tecknen med kodpunkten 26, som annars markerar slutet på filen och kan vara del av ett fältvärde.

Endast relevant för avgränsade textfiler.

Exempel:

```
LOAD * from a.txt (txt, utf8, embedded labels, delimiter is ' ', no eof);
```

Se även:

Load (page 97)

Labels

Labels är en filspecifikator för **LOAD**-satsen som anger var i en fil fältnamnen kan hittas.

Syntax:

```
embedded labels|explicit labels|no labels
```

Fältnamnen kan finnas på olika ställen i filen. Om filens första post innehåller fältnamnen, ska **embedded labels** användas. Om fältnamnen inte finns lagrade i filen, ska **no labels** användas. I *dif*-filer finns ibland en särskild inledning (header, se nedan), skild från datasektionen. I sådana fall bör man använda **explicit labels**. Om ingenting anges antas **embedded labels**, även för *dif*-filer.

Example 1:

```
LOAD * from a.txt (unicode, txt, delimiter is ',' , embedded labels
```

Example 2:

```
LOAD * from a.txt (codePage is 1252, txt, delimiter is ',' , no labels)
```

Se även:

Load (page 97)

Header is

Anger filhuvudets storlek i tabellfiler. En godtycklig inledningslängd kan anges genom specifikatorn **header is**. En sådan inledning är en sektion som inte används i Qlik Sense.

Syntax:

```
header is n
header is line
header is n lines
```

Rubriklängden kan anges i bytes (**header is n**), eller i linjer (**header is line** eller **header is n lines**). **n** måste vara ett positivt heltal som representerar rubriklängden. Om inget anges, antas **header is 0**. **header is**-specifikatorn är endast relevant för tabellfiler.

Exempel:

Det här är ett exempel på en datakälla i tabellformat som innehåller en rad med rubriktext som inte ska tolkas som data av Qlik Sense.

```
*Header line
Col1,Col2
a,B
c,D
```

Användning av specifikatorn **header is 1 lines** gör att den första raden inte blir inläst som data. I det här exemplet anger specifikatorn **embedded labels** att Qlik Sense ska tolka den första icke-uteslutna raden som att den innehåller fältetiketter.

```
LOAD Col1, Col2
FROM 'lib://files/header.txt'
(txt, embedded labels, delimiter is ',', msq, header is 1 lines);
```

Resultatet är en tabell med två fält, Col1 and Col2.

Se även:

Load (page 97)

Record is

För fixpostfiler måste postlängden anges med specifikatorn **record is**.

Syntax:

```
Record is n
Record is line
Record is n lines
```

Argument:

Argument

Argument	Beskrivning
n	Anger postlängden i bytes.
line	Anger postlängden som en rad.
n lines	Anger postlängden i rader där n är ett positivt heltal som motsvarar postlängden.

Begränsningar:

record is-specifikatorn är endast relevant för **fix**-filer.

Se även:

Load (page 97)

Quotes

Quotes är en filspecifikator för **LOAD**-satsen som anger om citationstecken kan användas och hur citationstecken och avgränsare prioriteras. Gäller endast textfiler.

Syntax:

```
no quotes
msq
```

Om specifikatorn utelämnas används standardinställningen. Då accepteras både " " och ' ', men bara som första eller sista icke-blankt tecken i ett fältvärde.

Argument:

Argument

Argument	Beskrivning
no quotes	Används om citationstecken inte ska accepteras i en textfil.
msq	Används för att ange Modern Style Quoting, som tillåter flerradigt innehåll i fält. Fält som innehåller radslutstecken måste omslutas av dubbla citattecken. En begränsning av msq-alternativet är att enstaka dubbelcitattecken som visas som första eller sista tecken i fältinnehåll tolkas som början eller slutet på flerradigt innehåll, vilket kan leda till oförutsedda resultat i den inlästa datauppsättningen. I detta fall bör standardcitationstecken användas i stället och specifikatorn bör utelämnas.

XML

Denna skriptspecifikator används när du läser in xml-filer.Giltiga alternativ för **XML**-specifikatorn syns i listan i syntaxen.

Du kan inte ladda DTD-filer i Qlik Sense.

Syntax:

```
xmlsimple
```

Se även:

Load (page 97)

KML

Denna skriptspecifikator används för att läsa in KML-filer för användning i en kartvisualisering.

Syntax:

```
kml
```

KML-filen kan representera antingen ytdata (till exempel länder eller regioner) som representeras av polygoner, linjedata (till exempel spår eller vägar), eller datapunkter (till exempel städer eller platser) som representeras i formatet [long, lat].

URL is

Den här skriptspecifikatorn används för att ange URL:en för en webbfils dataanslutning när den laddar en webbfil.

Syntax:

```
URL is string
```

Argument:

Argument

Argument	Beskrivning
string	Anger URL för den fil som ska laddas. Det åsidosätter den URL som anges i den webbfilskoppling som används.

Begränsningar:

URL is-specifikatorn är endast relevant för webbfiler. Du måste använda en befintlig webbfilsdatakoppling.

Se även:

Load (page 97)

userAgent is

Den här skriptspecifikatorn används för att ange webbläsarens användaragent när den laddar en webbfil.

Syntax:

```
userAgent is string
```

Argument:

Argument

Argument	Beskrivning
string	Anger användaragentsträng för webbläsaren. Den åtsidosätter den standardagentsträngen för webbläsaren "Mozilla/5.0".

Begränsningar:

userAgent is-specifikatorn är endast relevant för webbfiler.

Se även:

Load (page 97)

Let

let-satsen är ett komplement till **set**-satsen som används för att definiera skriptvariabler. I motsats till **set**-satsen utvärderas **let**-satsens uttryck till höger om '=' när skriptet körs, innan det tilldelas variabeln.

Syntax:

```
Let variablename=expression
```

Exempel och resultat:

Exempel	Resultat

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	`$(x)` utvärderas som ' 3+4 '
	
	`$(y)` utvärderas som ' 7 '
	
	`$(z)` utvärderas som ' 8 '
	
	Observera skillnaden mellan **Set**- och **Let**-satserna. **Set**-satsen tilldelar strängen '3+4' till variabeln, medan **Let**-satsen utvärderar strängen och tilldelar variabeln värdet 7.

Let T=now( );	`$(T)` får ett värde som motsvarar aktuell tid.

Loosen Table

En eller flera av Qlik Senses interna datatabeller kan explicit förklaras vara löst kopplade under skriptexekvering med hjälp av satsen **Loosen Table**. När en tabell är löst kopplad tas alla associationer mellan fältvärdena i tabellen bort. Man kan åstadkomma en liknande effekt genom att ladda vart och ett av fälten i den löst kopplade tabellen som fristående, okopplade tabeller. Löst kopplad kan vara användbart under tester för att tillfälligt isolera olika delar av datastrukturen. En löst kopplad tabellen kan identifieras i tabellvyn genom de streckade linjerna. Om en eller flera **Loosen Table**-satser används i skriptet gör detta att Qlik Sense ignorerar alla inställningar för löst kopplade tabeller gjorda innan skriptexekveringen.

Syntax:

```
Loosen Tabletablename [ , tablename2 ...]
Loosen Tablestablename [ , tablename2 ...]
```

Antingen syntaxen **Loosen Table** eller **Loosen Tables** kan användas.

Om Qlik Sense hittar cirkelreferenser i datastrukturen som inte kan brytas genom att tabeller interaktivt eller explicit förklaras vara löst kopplade i skriptet, kommer en eller flera ytterligare tabeller att tvingas vara löst kopplade tills inga cirkelreferenser kvarstår. När detta inträffar får man en varning via dialogen ***Varning cirkulära referenser****.*

Exempel:

```
Tab1:
SELECT * from Trans;
Loosen Table Tab1;
```

Map

Satsen **map ... using** används för att mappa ett visst fältvärde eller uttryck till värdena i en specifik mappningstabell. Mappningstabellen skapas via satsen **Mapping**.

Syntax:

```
Map fieldlist Using  mapname
```

Automatisk mappning görs för fält som är inlästa efter **Map … Using**-satsen fram till skriptets slut eller tills en **Unmap**-sats påträffas.

Mappningen utförs sist i den kedja av händelser som leder fram till att fältet lagras i den interna Qlik Sense-tabellen. Detta innebär att mappning inte görs varje gång ett fältnamn påträffas som del av ett uttryck, utan när värdet lagras under fältnamnet i den interna tabellen. Om mappning på uttrycksnivå krävs, måste funktionen **Applymap()** användas istället.

Argument:

Argument

Argument	Beskrivning
fieldlist	En kommaavgränsad lista över de fält som ska mappas fr.o.m. den aktuella positionen i skriptet. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken.
mapname	Namnet på en mappningstabell som tidigare lästs in via en **mapping load**- eller **mapping select**-sats.

Exempel och resultat:

Exempel	Resultat
Map Country Using Cmap;	Möjliggör mappning av fältet Country med hjälp av Cmap.
Map A, B, C Using X;	Möjliggör mappning av fälten A, B och C med hjälp av X.
Map * Using GenMap;	Startar mappning av alla fält med hjälp av GenMap.

NullAsNull

Med **NullAsNull**-satsen upphävs konvertering av NULL-värden till strängvärden som dessförinnan har ställts in med **NullAsValue**-satsen.

Syntax:

```
NullAsNull *fieldlist
```

Satsen **NullAsValue** kan slås på eller av flera gånger i skriptet med hjälp av en **NullAsValue**- eller **NullAsNull**-sats.

Argument:

Argument

Argument	Beskrivning
*fieldlist	En kommaavgränsad lista över fält för vilka **NullAsNull** ska aktiveras. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken.

Exempel:

```
NullAsNull A,B;
LOAD A,B from x.csv;
```

NullAsValue

Satsen **NullAsValue** anger för vilka fält de NULL-värdena ska konverteras till värden.

Syntax:

```
NullAsValue *fieldlist
```

Qlik Sense betraktar normalt NULL-värden som saknade eller ej angivna enheter. I vissa databasprocesser är det däremot underförstått att NULL-värden ska tolkas som speciella värden och inte som värden som saknas. Det faktum att NULL-värden normalt inte tillåts koppla till andra NULL-värden kan dock upphävas med hjälp av **NullAsValue**-satsen.

Satsen **NullAsValue** påverkar alla efterföljande inläsningssatser. Den kan stängas av igen med **NullAsNull**-satsen.

Argument:

Argument

Argument	Beskrivning
*fieldlist	En kommaavgränsad lista över fält för vilka **NullAsValue** ska aktiveras. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken.

Exempel:

```
NullAsValue A,B;
Set NullValue = 'NULL';
LOAD A,B from x.csv;
```

Qualify

Satsen **Qualify** används för att aktivera bestämning av fältnamn, där fältnamn får tabellnamnet som prefix.

Syntax:

```
Qualify *fieldlist
```

Det är möjligt att förhindra automatiska länkningar mellan fält med samma namn i olika tabeller genom att använda en **qualify**-sats. Denna kvalificerar fältnamnet med dess tabellnamn. Fältnamn som förekommer i qualify-satsen döps om när de påträffas i tabeller. Det nya namnet är i formen av *tablename.fieldname*. *Tablename* motsvarar den aktuella tabellens etikett; finns det inget sådant, används istället det namn som förekommer efter **from** i **LOAD**- och **SELECT**-satser.

Bestämningen görs för alla fält som läses in efter **qualify**-satsen.

Som standard gäller att tabellbestämningen är inaktiverad i början av varje skriptexekvering. Tabellbestämningen kan när som helst aktiveras med hjälp av en **qualify**-sats. Bestämningen kan när som helst upphävas med hjälp av en **Unqualify**-sats.

***qualify**-satsen kan inte användas tillsammans med partiell laddning!*

Argument:

Argument

Argument	Beskrivning
*fieldlist	En kommaavgränsad lista över fält för vilka tabellbestämningen ska aktiveras. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken.

Example 1:

```
Qualify B;
LOAD A,B from x.csv;
LOAD A,B from y.csv;
```

De båda tabellerna **x.csv** och **y.csv** associeras endast genom **A**. Resultatet blir tre fält: A, x.B, y.B.

Example 2:

När man arbetar med en okänd databas är det ofta bra att till en början associera endast ett eller ett fåtal fält, som i följande exempel:

```
qualify *;
unqualify TransID;
SQL SELECT * from tab1;
SQL SELECT * from tab2;
SQL SELECT * from tab3;
```

Endast **TransID** kommer att användas för att associera tabellerna *tab1*, *tab2* och *tab3*.

Rem

rem-satsen används för att infoga kommentarer i skriptet, eller för att tillfälligt avaktivera skriptsatser utan att ta bort dem.

Syntax:

```
Rem string
```

Allting mellan **rem** och nästa semikolon **;** betraktas som kommentar.

Du kan skapa kommentarer i skriptet på två olika sätt.

1. Du kan skapa en kommentar var som helst i skriptet, förutom mellan två citationstecken, genom att placera avsnittet i fråga mellan **/*** och ***/**.
2. När **//** anges i skriptet, blir all text som följer till höger på samma rad en kommentar. (Observera undantaget //: som kan användas som en del i en Internetadress.)

Argument:

Argument

Argument	Beskrivning
string	En godtycklig text.

Exempel:

```
Rem ** This is a comment **;
/* This is also a comment */
// This is a comment as well
```

Rename

Skriptnyckelordet **Rename** kan användas för att byta namn på tabeller eller fält som redan laddats.

Rename field

Denna skriptfunktion döper om ett eller flera Qlik Sense-fält efter att de har lästs in.

> *Det rekommenderas inte att du ger en variabel samma namn som ett fält eller en funktion i Qlik Sense.*

Antingen syntaxen **rename field** eller **rename fields** kan användas.

Syntax:

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

Argument:

Argument	Beskrivning
mapname	Namnet på en tidigare inläst mappningstabell som innehåller ett eller flera par gamla och nya fältnamn.
oldname	Det gamla fältnamnet.
newname	Det nya fältnamnet.

Begränsningar:

Det går inte att byta namn på två fält så att de får samma namn.

Example 1:

```
Rename Field XAZ0007 to Sales;
```

Example 2:

```
FieldMap:
Mapping SQL SELECT oldnames, newnames from datadictionary;
Rename Fields using FieldMap;
```

Rename table

Denna skriptfunktion döper om ett eller flera interna tabeller i Qlik Sense efter att de har lästs in.

Antingen syntaxen **rename table** eller **rename tables** kan användas.

Syntax:

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

Argument:

Argument

Argument	Beskrivning
mapname	Namnet på en tidigare inläst mappningstabell som innehåller ett eller flera par gamla och nya tabellnamn.
oldname	Det gamla tabellnamnet.
newname	Det nya tabellnamnet.

Begränsningar:

Två tabeller med olika namn kan inte byta namn till samma namn. Skriptet genererar ett fel om du försök byta namn på en tabell till samma namn som en befintlig tabell.

Example 1:

```
Tab1:
SELECT * from Trans;
Rename Table Tab1 to Xyz;
```

Example 2:

```
TabMap:
Mapping LOAD oldnames, newnames from tabnames.csv;
Rename Tables using TabMap;
```

Search

Search-satsen används för att inkludera eller utesluta fält i smartsökningen.

Syntax:

```
Search Include *fieldlist
Search Exclude *fieldlist
```

Du kan använda flera Search-satser för att förfina urvalet av fält som ska tas med. Avsnitten utvärderas uppifrån och ned.

Argument:

Argument

Argument	Beskrivning
*fieldlist	En kommaavgränsad lista över fält som ska inkluderas eller uteslutas från sökningar i smartsökningsverktyget. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken.

Exempel:

Search-exempel

Sats	Beskrivning
`Search Include *;`	Inkludera alla fält i sökningar i smartsökningsverktyget.
`Search Exclude [*ID];`	Uteslut alla fält som slutar med ID från sökningar i smartsökningsverktyget.
`Search Exclude '*ID';`	Uteslut alla fält som slutar med ID från sökningar i smartsökningsverktyget.
`Search Include ProductID;`	Inkludera fältet ProductID i sökningar i smartsökningsverktyget.

Det kombinerade resultatet av dessa tre satser, i den här ordningen, är att alla fält som slutar med ID utom ProductID utesluts från sökningar i smartsökningsverktyget.

Section

Med en **section**-sats är det möjligt att definiera om efterföljande **LOAD**- och **SELECT**-satser ska betraktas som data eller som en definition av behörighet.

Syntax:

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

Om inget anges, antas **section application**. **section**-definitionen används fram tills en ny **section**-sats påträffas.

Exempel:

```
Section access;
Section application;
```

Select

Val av fält från en ODBC-datakälla eller en OLE DB-drivrutin görs via vanliga SQL **SELECT**-satser. Om **SELECT**-satserna accepteras eller ej beror framför allt på den ODBC-drivrutin eller OLE DB-drivrutin som används. Användning av satsen **SELECT** kräver en öppen datakoppling till källan.

Syntax:

```
Select [all | distinct | distinctrow | top n [percent] ] fieldlist

From tablelist

[where criterion ]

[group by fieldlist [having criterion ] ]

[order by fieldlist [asc | desc] ]

[ (Inner | Left | Right | Full) join tablename on fieldref = fieldref ]
```

Dessutom kan flera **SELECT**-satser ibland konkateneras till en enda sats med hjälp av en **union**-operator:

```
selectstatement Union selectstatement
```

SELECT-satsen tolkas av ODBC-drivrutinen eller OLE DB-leverantören, så avvikelser från den allmänna SQL-syntaxen kan uppstå beroende på funktionerna hos ODBC-drivrutinerna eller OLE DB-leverantören, exempelvis:

- **as** är ibland inte tillåtet, dvs. *aliasname* måste följa omedelbart efter *fieldname*.
- **as** är ibland obligatoriskt om ett *aliasname* används.
- **distinct**, **as**, **where**, **group by**, **order by** eller **union** stöds ibland inte.
- ODBC-drivrutinen saknar ibland stöd för några av de citationstecken som beskrivs ovan.

*Detta är ingen komplett beskrivning av SQL **SELECT**-satsen! **SELECT**-satser kan t.ex. kapslas, en **SELECT**-sats kan innehålla flera join-satser, ibland tillåts ett stort antal funktioner i uttrycken osv.*

Argument:

Argument

Argument	Beskrivning
distinct	**distinct** används om duplicerade kombinationer av värden i de valda fälten bara ska läsas in en gång.
distinctrow	**distinctrow** används om duplicerade poster i ursprungstabellen bara ska läsas in en gång.
fieldlist	**fieldlist ::= (*| field) {, field }** En lista över de fält som ska väljas. Genom att använda * som fältlista anger man alla fält i tabellen. **fieldlist ::= field {, field }** En lista över ett eller flera kommaavgränsade fält. **field ::= (fieldref| expression) [as aliasname]** Uttrycket kan exempelvis vara ett numeriskt uttryck eller en teckensträng från ett eller flera olika fält. Några av de operatorer och funktioner som normalt kan användas: +, -, *, /, & (konkatenering av strängar), sum(fieldname), count(fieldname), avg(fieldname) (average), month(fieldname), osv. Se dokumentationen till ODBC-drivrutinen för mer information. **fieldref ::= [tablename.] fieldname** **tablename** och **fieldname** är textsträngar som motsvarar det de beskriver. De måste stå inom raka, dubbla citationstecken om de innehåller exempelvis blanktecken. **as**-tillägget används för att tilldela fältet ett nytt namn.
from	**tablelist ::= table {, table }** En lista över de tabeller från vilka fälten ska hämtas. **table ::= tablename [[as] aliasname]** **tablename** kan sättas inom citationstecken.
where	**where** är ett tillägg som används för att tala om huruvida en post ska inkluderas i valet eller inte. **criterion** är ett logiskt uttryck, som ibland kan vara mycket komplext. En del av de operatorer som godtas är: numeriska operatorer och funktioner, **=**, **<>** eller **#(**ej lika med), **>**, **>=**, **<**, **<=**, **and**, **or**, **not**, **exists**, **some**, **all**, **in** och även nya **SELECT**-satser. Se dokumentationen till ODBC-drivrutinen eller OLE DB-providern för mer information.
group by	**group by** är en sats som används för att aggregera (gruppera) flera poster till en. Inom en grupp, för ett visst fält, måste alla poster ha samma värde, eller så kan fältet endast förekomma i uttryck i vilka kollektiva egenskaper räknas ut, exempelvis summa eller medelvärde. Uttrycket, som baseras på ett eller flera fält, definieras i fältsymboluttrycket.
having	**having** är en sats som används för att kvalificera grupper på ett liknande sätt som satsen **where** används för att kvalificera poster.

Argument	Beskrivning
order by	**order by** är en sats som specificerar sorteringsordningen för den tabell som **SELECT**-satsen resulterar i.
join	**join** är en kvalificerare som talar om att flera tabeller ska länkas samman till en enda. Fältnamn och tabellnamn måste sättas inom citationstecken om de innehåller blanktecken eller å,ä,ö. Om skriptet genereras automatiskt av Qlik Sense, används vanligen de citationstecken som ODBC-drivrutinen eller OLE DB-providern specificerar i datakällans definition av datakällan i **Connect**-satsen.

Example 1:

```
SELECT * FROM `Categories`;
```

Example 2:

```
SELECT `Category ID`, `Category Name` FROM `Categories`;
```

Example 3:

```
SELECT `Order ID`, `Product ID`,

`Unit Price` * Quantity * (1-Discount) as NetSales

FROM `Order Details`;
```

Example 4:

```
SELECT `Order Details`.`Order ID`,

Sum(`Order Details`.`Unit Price` * `Order Details`.Quantity) as `Result`

FROM `Order Details`, Orders

where Orders.`Order ID` = `Order Details`.`Order ID`

group by `Order Details`.`Order ID`;
```

Set

Satsen **set** används för att definiera skriptvariabler. Dessa kan användas för att ersätta strängar, sökvägar, drivrutiner med mera.

Syntax:

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Sleep

sleep-satsen pausar skriptexekveringen under en angiven tidsperiod.

Syntax:

```
Sleep n
```

Argument:

Argument	Beskrivning
n	Anges i millisekunder, där *n* är ett positivt heltal som inte är större än 3600000 (dvs. 1 timme). Värdet kan vara ett uttryck.

Example 1:

```
Sleep 10000;
```

Example 2:

```
Sleep t*1000;
```

SQL

Med **SQL**-satsen kan du skicka ett godtyckligt SQL-kommando via en ODBC- eller OLE DB-koppling.

Syntax:

```
SQL sql_command
```

Om du skickar SQL-satser som uppdaterar databasen kommer ett fel att returneras om Qlik Sense har öppnat ODBC-källan i skrivskyddat läge.

Syntax:

```
SQL SELECT * from tab1;
```

godtas och är den allmänt gällande syntaxen för **SELECT** av konsekvensskäl. SQL-prefixet kommer dock att förbli frivilligt för **SELECT**-satser.

Argument:

Argument	Beskrivning
sql_command	Ett giltigt SQL-kommando.

Example 1:

```
SQL leave;
```

Example 2:

```
SQL Execute <storedProc>;
```

SQLColumns

Satsen **sqlcolumns** returnerar ett antal fält som beskriver kolumnerna i den ODBC- eller OLE DB-datakälla som man gjort en koppling, **connect**, till.

Syntax:

```
SQLcolumns
```

Fälten kan kombineras med fält som genererats av **sqltables**- och **sqltypes**-kommandona för att ge en bra överblick över en viss databas. De tolv standardfälten är följande:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

COLUMN_NAME

DATA_TYPE

TYPE_NAME

PRECISION

LENGTH

SCALE

RADIX

NULLABLE

REMARKS

För en detaljerad beskrivning av fälten, se en referenshandbok om ODBC.

Exempel:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLcolumns;
```

Vissa ODBC-drivrutiner kanske inte stöder det här kommandot. Vissa ODBC-drivrutiner kan ge ytterligare fält.

SQLTables

Satsen **sqltables** returnerar ett antal fält som beskriver tabellerna i den ODBC- eller OLE DB-datakälla som man gjort en koppling, **connect**, till.

Syntax:

```
SQLTables
```

Fälten kan kombineras med fält som genererats av **sqlcolumns**- och **sqltypes**-kommandona för att ge en bra överblick över en viss databas. De fem standardfälten är följande:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

TABLE_TYPE

REMARKS

För en detaljerad beskrivning av fälten, se en referenshandbok om ODBC.

Exempel:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTables;
```

Vissa ODBC-drivrutiner kanske inte stöder det här kommandot. Vissa ODBC-drivrutiner kan ge ytterligare fält.

SQLTypes

Satsen **sqltypes** returnerar ett antal fält som beskriver typerna i den ODBC- eller OLE DB-datakälla som man gjort en koppling, **connect**, till.

Syntax:

```
SQLTypes
```

Fälten kan kombineras med fält som genererats av **sqlcolumns**- och **sqltables**-kommandona för att ge en bra överblick över en viss databas. De femton standardfälten är följande:

TYPE_NAME

DATA_TYPE

PRECISION

LITERAL_PREFIX

LITERAL_SUFFIX

CREATE_PARAMS

NULLABLE

CASE_SENSITIVE

SEARCHABLE

UNSIGNED_ATTRIBUTE

MONEY

AUTO_INCREMENT

LOCAL_TYPE_NAME

MINIMUM_SCALE

MAXIMUM_SCALE

För en detaljerad beskrivning av fälten, se en referenshandbok om ODBC.

Exempel:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTypes;
```

Vissa ODBC-drivrutiner kanske inte stöder det här kommandot. Vissa ODBC-drivrutiner kan ge ytterligare fält.

Star

Den textsträng som används för att representera den totala uppsättningen av alla värden i ett fält i databasen kan ställas in med hjälp av **star**-satsen. Den påverkar efterföljande **LOAD**- och **SELECT**-satser.

Syntax:

```
Star is[ string ]
```

Argument:

Argument

Argument	Beskrivning
string	En godtycklig text. Observera att strängen måste sättas inom citationstecken om den innehåller blanktecken. Om ingenting anges, antas **star is;**, dvs. stjärnsymbolen måste specificeras explicit för att vara tillgänglig. Definitionen används fram tills en ny **star**-sats skapas.

Om section access används bör inte **Star is**-satsen användas i datadelen av skriptet (under **Section Application**). Däremot kan stjärntecknet användas i de skyddade fälten i **Section Access**-delen av skriptet. I så fall behöver du inte använda den explicita **Star is**-satsen eftersom denna alltid är implicit i section access.

Begränsningar

- Du kan inte använda stjärntecknet i nyckelfält, dvs. fält som länkar tabeller.
- Du kan inte använda stjärntecknet i några fält som påverkas av **Unqualify**-satsen eftersom detta kan påverka fält som länkar tabeller.
- Du kan inte använda stjärntecknet i icke-logiska tabeller, till exempel tabeller av typerna info-load eller mapping-load.
- När stjärntecknet används i ett reduceringsfält (ett fält som länkar till data) i section access representerar det värdena som listas i detta fält i section access. Det representerar inte andra värden som kan ingå i data, men som inte är listade i section access.
- Du kan inte använda stjärntecknet med fält som påverkas av någon typ av datareduktion utanför **Section Access**-området.

Exempel

I exemplet nedan finns ett utdrag av ett dataladdningsskript med section access.

```
Star is *;



Section Access;

LOAD * INLINE [

ACCESS, USERID, OMIT

ADMIN, ADMIN,

USER, USER1, SALES

USER, USER2, WAREHOUSE

USER, USER3, EMPLOYEES
```

```
USER, USER4, SALES

USER, USER4, WAREHOUSE

USER, USER5, *

];



Section Application;

LOAD * INLINE [

SALES, WAREHOUSE, EMPLOYEES, ORDERS

1, 2, 3, 4

];
```

Följande gäller:

- *Star* -tecknet är *.
- Användaren *ADMIN* ser alla fält. Inget har utelämnats.
- Användaren *USER1* kan inte se fältet *SALES*.
- Användaren *USER2* kan inte se fältet *WAREHOUSE* .
- Användaren *USER3* kan inte se fältet *EMPLOYEES*.
- Användaren *USER4* läggs till två gånger i lösningen för att UTESLUTA två fält för den här användaren, *SALES* och *WAREHOUSE*.
- *USER5* har en ”*” som tillägg, vilket betyder att alla fält under OMIT är otillgängliga, d.v.s. användaren *USER5* kan inte se fälten *SALES*, *WAREHOUSE* och *EMPLOYEES* men den här användaren kan se fältet *ORDERS*.

Store

Store-satsen skapar en QVD-, CSV- eller text-fil.

Syntax:

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

Satsen skapar en explicit namngiven QVD-, CSV- eller TXT-fil.

Satsen kan endast exportera fält från en datatabell. Om fält från flera tabeller ska exporteras, måste du först göra en explicit join i skriptet för att skapa den datatabell som ska exporteras.

Textvärdena exporteras till CSV-filen i UTF-8-format. En avgränsare kan anges, se **LOAD**. Om **store** sparar till en CSV-fil stöds inte BIFF-export.

Argument:

Argument för kommandot Store

Argument	Beskrivning
*fieldlist::= (* | field) { , field })*	En lista över de fält som ska väljas. Genom att använda * som fältlista anger man alla fält. *field::= fieldname [***as** *aliasname]* *fieldname* är en text som är identisk med fältnamnet i *table*. (Observera att fältnamnet måste omslutas av raka, dubbla citationstecken eller av hakparenteser om det innehåller exempelvis mellanslag eller andra tecken som inte är standard.) *aliasname* är ett alternativt namn för det fält som ska användas i den färdiga QVD- eller CSV-filen.
table	En skriptetikett för en redan inläst tabell som ska användas som datakälla.
filename	Namnet på målfilen inkluderar en giltig sökväg till en befintlig mappdatakoppling. **Exempel:** ***'lib://Table Files/target.qvd'*** I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** ***c:\data\sales.qvd*** • relativ till Qlik Sense-appens arbetskatalog. **Exempel:** ***data\sales.qvd*** Om sökvägen utelämnas sparar Qlik Sense filen i den mapp som specificerats av en **Directory**-sats. Om det inte finns någon **Directory**-sats, lagrar Qlik Sense filen i arbetskatalogen, *C:\Users\{user}\Documents\Qlik\Sense\Apps*.
*format-spec ::=***(** (**txt** | **qvd**) **)**	Formatspecifikationen består av texten **txt** för textfiler, eller texten **qvd** för qvd-filer. Om formatspecifikationen utelämnas antas **qvd**.

Exempel:

```
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 * from mytable into 'lib://FolderConnection/myfile.qvd';
```

Filtillägget för DataFiles-kopplingar är skiftlägeskänsligt. Till exempel: .qvd.

Table/Tables

Skriptnyckelorden **Table** och **Tables** används i **Drop**-, **Comment**- och **Rename**-satser, liksom som en formatspecifikator i **Load**-satser.

Tag

Den här skriptsatsen erbjuder ett sätt att tilldela taggar till ett eller flera fält eller tabeller. Om du försöker tagga ett fält eller en tabell som inte finns i appen ignoreras taggningen. Om det finns flera förekomster av ett fältnamn eller taggnamn används det senaste värdet.

Syntax:

```
Tag [field|fields] fieldlist with tagname

Tag [field|fields] fieldlist using mapname

Tag table tablelist with tagname
```

Argument

Argument	Beskrivning
fieldlist	Ett eller flera fält som ska taggas, i en kommaavgränsad lista.
mapname	Namnet på en mappningstabell som tidigare laddats in via en **mapping Load**- eller **mapping Select**-sats.
tablelist	En kommaavgränsad lista över tabellerna som ska taggas.
tagname	Namnet på den tagg som ska användas på fältet.

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
```

```
];
tag fields using tagmap;
```

Example 2:

```
tag field Alpha with 'MyTag2';
```

Trace

trace-satsen skriver en sträng till fönstret **Skriptexekvering** och skriptets loggfil när den används. Detta är mycket användbart vid felsökning. Med hjälp av dollarteckenexpansion av variabler som beräknas innan **trace**-satsen används kan man anpassa meddelandet.

Syntax:

```
Trace string
```

Example 1:

Följande sats kan användas direkt efter Load-satsen som laddar Main-tabellen.

```
Trace Main table loaded;
```

Texten ”Main table loaded” (Huvudtabell har laddats) visas i skriptkörningsdialogen samt i loggfilen.

Example 2:

Följande satser kan användas direkt efter Load-satsen som laddar Main-tabellen.

```
Let MyMessage = NoOfRows('Main') & ' rows in Main table';

Trace $(MyMessage);
```

Då visas en text som visar antalet rader i skriptkörningsdialogen och i loggfilen, till exempel ”265,391 rows in Main table” (265 391 rader i huvudtabellen).

Unmap

Satsen **Unmap** avaktiverar fältvärdesmappningar, som har gjorts med hjälp av en tidigare **Map … Using**-sats för fält som läses in efteråt.

Syntax:

```
Unmap *fieldlist
```

Argument:

Argument

Argument	Beskrivning
*fieldlist	En kommaavgränsad lista över de fält som inte längre ska mappas från den aktuella positionen i skriptet. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken.

Exempel och resultat:

Exempel	Resultat
Unmap Country;	Inaktiverar mappning av fältet Country.
Unmap A, B, C;	Inaktiverar mappning av fälten A, B och C.
Unmap * ;	Inaktiverar mappning av alla fält.

Unqualify

Unqualify-satsen upphäver tabellbestämning av fältnamn som tidigare definierats av en **Qualify**-sats.

Syntax:

```
Unqualify *fieldlist
```

Argument:

Argument

Argument	Beskrivning
*fieldlist	En kommaavgränsad lista över fält för vilka tabellbestämningen ska aktiveras. Genom att använda * som fältlista anger du alla fält. Jokertecknen * och ? är tillåtna i fältnamn. När jokertecken används som delar av fältnamn är det ibland nödvändigt att sätta fältnamnen inom citationstecken. Se dokumentationen till **Qualify**-satsen för ytterligare information.

Example 1:

När man arbetar med en okänd databas är det ofta bra att till en början associera endast ett eller ett fåtal fält, som i följande exempel:

```
qualify *;
unqualify TransID;
SQL SELECT * from tab1;
SQL SELECT * from tab2;
SQL SELECT * from tab3;
```

Först aktiveras bestämning för alla fält.
Sedan inaktiveras bestämning för **TransID**.
Endast **TransID** kommer att användas för att associera tabellerna *tab1*, *tab2* och *tab3*. Alla andra fält kommer att bestämmas med tabellnamnet.

Untag

Den här skriptsatsen erbjuder ett sätt att ta bort taggar från fält eller tabeller. Om du försöker ta bort taggar från ett fält eller en tabell som inte finns i appen ignoreras försöket.

Syntax:

```
Untag [field|fields] fieldlist with tagname
```

```
Untag [field|fields] fieldlist using mapname
```

```
Untag table tablelist with tagname
```

Argument:

Argument

Argument	Beskrivning
fieldlist	Ett eller flera fält vars taggar ska tas bort, i en kommaavgränsad lista.
mapname	Namnet på en mappningstabell som tidigare har lästs in via en **LOAD**- eller **SELECT**-sats för mappning.
tablelist	En kommaavgränsad lista över tabellerna vars taggar ska tas bort.
tagname	Namnet på den tagg som ska tas bort från fältet.

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
Untag fields using tagmap;
```

Example 2:

```
Untag field Alpha with MyTag2;
```

2.6 Arbetskatalog

Om du refererar till en fil i en skriptsats och sökvägen utelämnas så söker Qlik Sense efter filen i följande ordning:

1. Katalogen som anges av en **Directory**-sats (stöds endast i bakåtkompatibelt skriptläge).
2. Om det inte finns någon **Directory**-sats söker Qlik Sense i arbetskatalogen.

Qlik Sense Desktop-arbetskatalog

I Qlik Sense Desktop är arbetskatalogen *C:\Users\{user}\Documents\Qlik\Sense\Apps*.

Qlik Sense-arbetskatalog

I en Qlik Sense-serverinstallation anges arbetskatalogen i Qlik Sense Repository Service. Som standard är det *C:\ProgramData\Qlik\Sense\Apps*. Se hjälpen till Qlik Management Console för mer information.

2 Arbeta med variabler i Skriptredigeraren

En variabel i Qlik Sense är en behållare som lagrar ett statiskt värde eller en beräkning, till exempel ett numeriskt eller ett alfanumeriskt värde. När du använder variabeln i appen kan alla förändringar som görs för variabeln tillämpas överallt där variabeln används. Du kan definiera variabler med hjälp av variabelöversikten eller i skript med hjälp av Skriptredigeraren. Du ställer in värdet för en variabel med **Let** eller **Set** uttalanden i dataladdningsskriptet.

Du kan även arbeta med Qlik Sense-variablerna från variabelöversikten när du redigerar ett ark.

2.7 Översikt

Om det första tecknet i ett variabelvärde är ett likhetstecken ”=” kommer Qlik Sense att försöka utvärdera värdet som en formel (Qlik Sense-uttryck) och därefter visa eller returnera resultatet snarare än visa den faktiska formeln.

När variabeln används ersätts den av sitt värde. Variabler kan användas för dollarteckenexpansion eller i olika kontrollsatser i skriptet. Det är väldigt praktiskt om samma sträng förekommer flera gånger i skriptet, t.ex. en sökväg.

Vissa speciella systemvariabler ställs dock in av Qlik Sense i början av skriptexekveringen oavsett vilket värde de tidigare hade i layouten.

2.8 Definiera en variabel

Variabler gör det möjligt att lagra statiska värden eller resultatet av en beräkning. När en variabel definieras använder du följande syntax:

```
set variablename = string
```

eller

```
let variable = expression
```

Satsen **Set** används för strängtilldelning. Den tilldelar texten till höger om likhetstecknet till variablerna. Satsen **Let** utvärderar ett uttryck till höger om likhetstecknet under skriptets körningstid och tilldelar resultatet av uttrycket till variabeln.

Variabler är skiftlägeskänsliga.

Det rekommenderas inte att du ger en variabel samma namn som ett fält eller en funktion i Qlik Sense.

Exempel:

`set x = 3 + 4; //`-variabeln får strängen '3 + 4' som värde.

`let x = 3 + 4; //` returnerar 7 som värde.

`set x = Today(); //` returnerar 'Today()' som värde.

`let x = Today(); //` returnerar dagens datum som värde, till exempel ‘9/27/2021’.

2.9 Ta bort en variabel

Om du tar bort en variabel från skriptet och laddar data igen stannar variabeln kvar i appen. Om du vill ta bort variabeln helt från appen måste du även ta bort variabeln från variabeldialogen.

2.10 Ladda ett variabelvärde som ett fältvärde

Om du vill ladda ett variabelvärde som ett fältvärde i en **LOAD** -sats och resultatet av dollarexpansionen är text i stället för tal eller ett uttryck måste du bifoga den utvidgade variabeln inom enkla citattecken.

Exempel:

Detta exempel laddar systemvariablen som innehåller listan över skriptfel till en tabell. Lägg märke till att expansionen av ScriptErrorCount i **If**-satsen inte kräver citattecken, men att expansionen av ScriptErrorList kräver citattecken.

```
IF $(ScriptErrorCount) >= 1 THEN

   LOAD '$(ScriptErrorList)' AS Error AutoGenerate 1; END IF
```

2.11 Beräkning med variabler

Det finns flera sätt att använda variabler med beräknade värden i Qlik Sense. Resultatet beror på hur du definierar den och hur du anropar den i ett uttryck.

I detta exempel laddar vi vissa data inline:

```
LOAD * INLINE [
    Dim, Sales
    A, 150
    A, 200
    B, 240
    B, 230
    C, 410
    C, 330
];
```

Nu ska vi definiera två variabler:

```
Let vSales  = 'Sum(Sales)' ;
Let vSales2  = '=Sum(Sales)' ;
```

I den andra variabeln lägger vi till ett likhetstecken framför uttrycket. Nu beräknas variabeln innan den expanderas och innan uttrycket utvärderas.

Om du använder variabeln vSales som den är, exempelvis i ett mått, blir resultatet strängen Sum(Sales). Med andra ord görs ingen beräkning.

Om du lägger till en dollarteckenexpansion och anropar $(vSales) i uttrycket, expanderas variabeln, och summan av Sales visas.

Om du i stället anropar $(vSales2) beräknas variabeln innan den expanderas. Detta innebär att resultatet som visas är totalsumman för Sales. Skillnaden mellan att använda =$(vSales) och =$(vSales2) som måttuttryck visas i diagrammet nedan. Resultat:

Resultat

Dim	$(vSales)	$(vSales2)
A	350	1560
B	470	1560
C	740	1560

Som du ser resulterar $(vSales) i delsumman för ett dimensionsvärde, medan $(vSales2) resulterar i totalsumman.

Följande skriptvariabler är tillgängliga:

- *Felvariabler (page 160)*
- *Variabler för tolkning av tal (page 146)*
- *Systemvariabler (page 138)*
- *Variabler för värdehantering (page 144)*

2.12 Systemvariabler

Systemvariabler, var av vissa är systemdefinierade, ger information om systemet och Qlik Sense-appen.

Översikt av systemvariabler

En del av funktionerna beskrivs mer ingående efter översikten. För de här funktionerna kan du klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Floppy

Returnerar enhetsbeteckningen för den första diskettenhet som påträffas, vanligen *a:*. Detta är en systemdefinierad variabel.

```
Floppy
```

Variabeln stöds inte i standardläget.

CD

Returnerar bokstaven på den första cd-rom-enhet som påträffas. Om ingen cd-rom-enhet påträffas, returneras *c:*. Detta är en systemdefinierad variabel.

```
CD
```

Variabeln stöds inte i standardläget.

Include

Variabeln **Include/Must_Include** specificerar en fil som innehåller text som ska inkluderas i skriptet och utvärderas som skriptkod. Den används inte för att lägga till data. Du kan spara delar av skriptkoden i en separat textfil och återanvända den i flera appar. Detta är en användardefinierad variabel.

```
$(Include=filename)
$(Must_Include=filename)
```

HidePrefix

Alla fältnamn som inleds av denna textsträng kommer att döljas på samma sätt som systemfälten. Detta är en användardefinierad variabel.

```
HidePrefix
```

HideSuffix

Alla fältnamn som avslutas av denna textsträng kommer att döljas på samma sätt som systemfälten. Detta är en användardefinierad variabel.

```
HideSuffix
```

QvPath

Returnerar sökströngen till Qlik Sense-programfilen. Detta är en systemdefinierad variabel.

```
QvPath
```

Variabeln stöds inte i standardläget.

QvRoot

Returnerar rotkatalogen till Qlik Sense-programfilen. Detta är en systemdefinierad variabel.

```
QvRoot
```

Variabeln stöds inte i standardläget.

QvWorkPath

Returnerar sökströngen till den aktuella Qlik Sense-appen. Detta är en systemdefinierad variabel.

```
QvWorkPath
```

Variabeln stöds inte i standardläget.

QvWorkRoot

Returnerar rotmappen till den aktuella Qlik Sense-appen. Detta är en systemdefinierad variabel.

```
QvWorkRoot
```

Variabeln stöds inte i standardläget.

StripComments

Om denna variabel är inställd på 0 blir kommentarer av typen /*..*/ och // otillåtna. Om variabeln inte är definierad strippas alltid kommentarer.

```
StripComments
```

Verbatim

Normalt töms alla fältvärden automatiskt på inledande och avslutande blanktecken (ASCII 32) innan de läses in i Qlik Sense-databasen. Om man sätter variabeln till 1 fördröjs rensningen på tomma blanktecken. Tecknen tabb (ASCII 9) och hårt mellanslag (ANSI 160) rensas aldrig bort.

```
Verbatim
```

OpenUrlTimeout

Denna variabel anger i sekunder den tidsgräns som Qlik Sense ska hålla när data hämtas från URL-källor (t.ex. HTML sidor). Om den utelämnas blir det en tidsgräns på ungefär 20 minuter.

```
OpenUrlTimeout
```

WinPath

Returnerar sökströngen till Windows. Detta är en systemdefinierad variabel.

```
WinPath
```

Variabeln stöds inte i standardläget.

WinRoot

Returnerar Windows rotmapp. Detta är en systemdefinierad variabel.

```
WinRoot
```

Variabeln stöds inte i standardläget.

CollationLocale
Anger vilken språkvariant som ska användas för sorteringsordning och sökmatchning. Värdet är kulturnamnet för en språkvariant, till exempel ”en-US”.Detta är en systemdefinierad variabel.

```
CollationLocale
```

CreateSearchIndexOnReload
Denna variabel definierar om sökindexfiler ska skapas under ominläsning av data.

```
CreateSearchIndexOnReload
```

CreateSearchIndexOnReload

Denna variabel definierar om sökindexfiler ska skapas under ominläsning av data.

Syntax:

```
CreateSearchIndexOnReload
```

Du kan definiera om sökindexfiler ska skapas vid omladdning av data eller om de ska skapas efter den första sökbegäran från användaren. Fördelen med att skapa sökindexfiler under omladdning av data är att du undviker den väntetid som den första användaren som gör en sökning annars får. Detta måste vägas mot den längre tiden för omladdning av data som krävs vid skapande av ett sökindex.

Om denna variabel utelämnas skapas inte sökindexfiler vid omladdning av data.

> *För sessionsappar skapas inte sökindexfiler vid omladdning av data oavsett denna variabels inställning.*

Example 1: Skapa sökindexfiler under omladdning av data.

```
set CreateSearchIndexOnReload=1;
```

Example 2: Skapa sökindexfiler efter första sökningsbegäran

```
set CreateSearchIndexOnReload=0;
```

HidePrefix

Alla fältnamn som inleds av denna textsträng kommer att döljas på samma sätt som systemfälten. Detta är en användardefinierad variabel.

Syntax:

```
HidePrefix
```

Exempel:

```
set HidePrefix='_' ;
```

Satsen medför att fältnamn som börjar med ett understrykningstecken inte kommer att visas i fältnamnslistorna om systemfälten är dolda.

HideSuffix

Alla fältnamn som avslutas av denna textsträng kommer att döljas på samma sätt som systemfälten. Detta är en användardefinierad variabel.

Syntax:

```
HideSuffix
```

Exempel:

```
set HideSuffix='%';
```

Satsen medför att fältnamn som avslutas av ett procenttecken inte kommer att visas i fältnamnslistorna om systemfälten är dolda.

Include

Variabeln **Include/Must_Include** specificerar en fil som innehåller text som ska inkluderas i skriptet och utvärderas som skriptkod. Den används inte för att lägga till data. Du kan spara delar av skriptkoden i en separat textfil och återanvända den i flera appar. Detta är en användardefinierad variabel.

Variabeln har enbart stöd för mappdatakopplingar i standardläget.

Syntax:

```
$(Include=filename)
```

```
$(Must_Include=filename)
```

Det finns två versioner av variabeln:

- **Include** genererar inte ett del om det inte går att hitta filen, den misslyckas i tysthet.
- **Must_Include** genererar ett fel om det inte går att hitta filen.

Om du inte anger en sökväg, blir filnamnet relativt till Qlik Sense-appens arbetskatalog. Du kan också ange en absolut sökväg till filen, eller en sökväg till lib://-katalogens anslutning. Likhetstecknet ska inte ha blanksteg före eller efter.

*Konstruktionen **set Include =**filename är inte tillämplig.*

Exempel:

```
$(Include=abc.txt);
```

```
$(Must_Include=lib://DataFiles/abc.txt);
```

Begränsningar

Begränsad kompatibilitet mellan Windows och Linux för UTF-8-kodade filer.

Det är valfritt att använda UTF-8 med BOM (Byte Order Mark). BOM kan störa användningen av UTF-8 i program som inte förväntar sig icke-ASCII-byte i början av en fil, men som annars skulle kunna hantera textströmmen.

- Windows-system använder BOM i UTF-8 för att identifiera att en fil är UTF-8-kodad, trots att det inte råder någon tveksamhet om hur lagring i byte ska ske.
- Unix/Linux använder UTF-8 för Unicode men använder inte BOM, eftersom detta stör syntaxen i kommandofiler.

Detta medför vissa konsekvenser för Qlik Sense.

- I Windows identifieras alla filer som börjar med en UTF-8 BOM som en UTF-8-skriptfil. Annars antas att ANSI-kodning används.
- I Linux är UTF-8 systemets standardkodsida för 8 bitar. Därför fungerar UTF-8 fastän den inte innehåller någon BOM.

Portabilitet kan därför inte garanteras. En fil som skapas i Windows kan inte alltid tolkas av Linux och vice versa. För UTF-8-kodade filer finns ingen tvärkompatibilitet mellan systemen eftersom de hanterar BOM på olika sätt.

OpenUrlTimeout

Denna variabel anger i sekunder den tidsgräns som Qlik Sense ska hålla när data hämtas från URL-källor (t.ex. HTML sidor). Om den utelämnas blir det en tidsgräns på ungefär 20 minuter.

Syntax:

```
OpenUrlTimeout
```

Exempel:

```
set OpenUrlTimeout=10;
```

StripComments

Om denna variabel är inställd på 0 blir kommentarer av typen /*..*/ och // otillåtna. Om variabeln inte är definierad strippas alltid kommentarer.

Syntax:

```
StripComments
```

Vissa databasdrivrutiner använder sig av /*..*/ som optimeringstips i **SELECT**-satser. Om detta är fallet bör kommentarerna inte rensas bort innan **SELECT**-satsen skickas till databasens drivrutin.

Det rekommenderas att variabeln återställs till 1 omedelbart efter satserna vid behov.

Exempel:

```
set StripComments=0;
SQL SELECT * /* <optimization directive> */ FROM Table ;
set StripComments=1;
```

Verbatim

Normalt töms alla fältvärden automatiskt på inledande och avslutande blanktecken (ASCII 32) innan de läses in i Qlik Sense-databasen. Om man sätter variabeln till 1 fördröjs rensningen på tomma blanktecken. Tecknen tabb (ASCII 9) och hårt mellanslag (ANSI 160) rensas aldrig bort.

Syntax:

```
Verbatim
```

Exempel:

```
set Verbatim = 1;
```

2.13 Variabler för värdehantering

Den här delen beskriver variabler som används för att hantera NULL och andra värden.

Översikt av variabler för värdehantering

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

NullDisplay

Den definierade symbolen kommer att ersätta alla NULL-värden från ODBC, samt kopplingarna, på den lägsta datanivån. Detta är en användardefinierad variabel.

```
NullDisplay
```

NullInterpret

Symbolen som definieras kommer att tolkas som NULL varje gång den förekommer i en textfil, Excel-fil eller i en inline-sats. Detta är en användardefinierad variabel.

```
NullInterpret
```

NullValue

Om **NullAsValue**-satsen används ersätter den definierade symbolen alla NULL-värden i **NullAsValue**-angivna fält med den angivna strängen.

```
NullValue
```

OtherSymbol
Definierar att en symbol som ska behandlas som "alla andra värden" före en **LOAD/SELECT**-sats. Detta är en användardefinierad variabel.

```
OtherSymbol
```

NullDisplay

Den definierade symbolen kommer att ersätta alla NULL-värden från ODBC, samt kopplingarna, på den lägsta datanivån. Detta är en användardefinierad variabel.

Syntax:

```
NullDisplay
```

Exempel:

```
set NullDisplay='<NULL>';
```

NullInterpret

Symbolen som definieras kommer att tolkas som NULL varje gång den förekommer i en textfil, Excel-fil eller i en inline-sats. Detta är en användardefinierad variabel.

Syntax:

```
NullInterpret
```

Exempel:

```
set NullInterpret=' ';
set NullInterpret =;
```

returnerar INTE NULL-värden för tomma värden i Excel, med gör det i en CSV-textfil.

```
set NullInterpret ='';
```

returnerar NULL-värden för tomma värden i Excel.

NullValue

Om **NullAsValue**-satsen används ersätter den definierade symbolen alla NULL-värden i **NullAsValue**-angivna fält med den angivna strängen.

Syntax:

```
NullValue
```

Exempel:

```
NullAsValue Field1, Field2;
set NullValue='<NULL>';
```

OtherSymbol

Definierar att en symbol som ska behandlas som "alla andra värden" före en **LOAD/SELECT**-sats. Detta är en användardefinierad variabel.

Syntax:

```
OtherSymbol
```

Exempel:

```
set OtherSymbol='+';
LOAD * inline
[X, Y
a, a
b, b];
LOAD * inline
[X, Z
a, a
+, c];
```

Fältvärdet Y='b' länkar nu till Z='c' via den andra symbolen.

2.14 Variabler för tolkning av tal

Variabler för tolkning av tal genereras av systemet, dvs. de genereras automatiskt i överensstämmelse med rådande regionala inställningar i operativsystemet när en ny app skapas. I Qlik Sense Desktop innebär detta enligt inställningarna i datorns operativsystem, och i Qlik Sense innebär det enligt operativsystemet på servern där Qlik Sense finns installerat.

Variablerna anges upptill i skriptet i den nya Qlik Sense-appen och ersätter operativsystemets standardinställningar för vissa talformatsinställningar när skriptet exekveras. De kan tas bort, redigeras eller dupliceras efter behag.

Om du vill skapa en app för ett visst språk är det enklaste sättet troligtvis att använda Qlik Sense Desktop på en dator med den önskade språkinställningen i operativsystemet för att skapa appen. Appen innehåller då de rätta regionala inställningarna för det språket och du kan flytta den till en Qlik Sense valfri server för ytterligare utveckling.

Variabler för tolkning av tal – en översikt

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

Valutaformat

MoneyDecimalSep

Decimalavgränsaren som definieras ersätter operativsystemets decimalavgränsare för valuta (regionala inställningar).

```
MoneyDecimalSep
```

MoneyFormat

Formatet som definieras ersätter operativsystemets valutaformat (regionala inställningar).

```
MoneyFormat
```

MoneyThousandSep

Tusentalsavgränsaren som definieras ersätter operativsystemets tusentalsavgränsare för valuta (regionala inställningar).

```
MoneyThousandSep
```

Talformat

DecimalSep

Decimalavgränsaren som definieras ersätter operativsystemets decimalavgränsare (regionala inställningar).

```
DecimalSep
```

ThousandSep

Tusentalsavgränsaren som definieras ersätter operativsystemets tusentalsavgränsare (regionala inställningar).

```
ThousandSep
```

NumericalAbbreviation

Den numeriska förkortningen ställer in vilken förkortning att använda för siffrors skalprefix, till exempel M för mega eller miljon (10^6) och µ för mikro (10^{-6}).

```
NumericalAbbreviation
```

Tidsformat

DateFormat

Miljövariabeln definierar datumformatet som används som standard i appen. Formatet används både för att tolka och formatera datum. Om variabeln inte är definierad kommer datumformatet enligt regioninställningarna i operativsystemet att hämtas när skriptet körs.

```
DateFormat
```

TimeFormat

Formatet som definieras ersätter operativsystemets format för tid (regionala inställningar).

```
TimeFormat
```

TimestampFormat

Formatet som definieras ersätter operativsystemets format för datum och tid (regionala inställningar).

```
TimestampFormat
```

MonthNames

Formatet som definieras ersätter operativsystemets format för månadsnamn (regionala inställningar).

```
MonthNames
```

LongMonthNames

Formatet som definieras ersätter operativsystemets format för långa månadsnamn (regionala inställningar).

```
LongMonthNames
```

DayNames

Formatet som definieras ersätter operativsystemets format för veckodagsnamn (regionala inställningar).

```
DayNames
```

LongDayNames

Formatet som definieras ersätter operativsystemets format för långa veckodagsnamn (regionala inställningar).

```
LongDayNames
```

FirstWeekDay

Heltal som definierar vilken dag som ska användas som den första dagen i veckan.

```
FirstWeekDay
```

BrokenWeeks

Den här inställningen definierar om veckor ska brytas eller inte.

```
BrokenWeeks
```

ReferenceDay

Inställningen definierar vilken dag i januari som ska ställas in som referensdag för att definiera vecka 1.

```
ReferenceDay
```

FirstMonthOfYear

Inställningen definierar vilken månad som ska användas som inledande månad på året, och kan användas för att definiera brutna räkenskapsår, exempelvis ett som inleds den 1 april.

Den här inställningen används för närvarande inte, utan är reserverad för framtida användning.

Giltiga inställningar är 1 (januari) till 12 (december). Standardinställningen är 1.

Syntax:

```
FirstMonthOfYear
```

Exempel:

```
Set FirstMonthOfYear=4; //Sets the year to start in April
```

BrokenWeeks

Den här inställningen definierar om veckor ska brytas eller inte.

Syntax:

```
BrokenWeeks
```

Som standard använder Qlik Sense-funktionerna obrutna veckor. Det betyder att:

- Vissa år börjar vecka 1 i december, och andra år fortsätter vecka 52 eller 53 in i januari.
- Vecka 1 har alltid minst 4 dagar i januari.

Alternativet är att använda brutna veckor:

- Vecka 52 eller 53 fortsätter inte in i januari.
- Vecka 1 börjar den 1 januari och är i de flesta fall inte en hel vecka.

Följande värden kan användas:

- 0 (= använd obrutna veckor)
- 1 (= använd brutna veckor)

Exempel:

```
Set BrokenWeeks=0; //(use unbroken weeks)

Set BrokenWeeks=1; //(use broken weeks)
```

DateFormat

Miljövariabeln definierar datumformatet som används som standard i appen. Formatet används både för att tolka och formatera datum. Om variabeln inte är definierad kommer datumformatet enligt regioninställningarna i operativsystemet att hämtas när skriptet körs.

Syntax:

```
DateFormat
```

Exempel:

```
Set DateFormat='M/D/YY'; //(US format)

Set DateFormat='DD/MM/YY'; //(UK date format)

Set DateFormat='YYYY-MM-DD'; //(ISO date format)
```

DayNames

Formatet som definieras ersätter operativsystemets format för veckodagsnamn (regionala inställningar).

Syntax:

```
DayNames
```

Exempel:

```
Set DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
```

DecimalSep

Decimalavgränsaren som definieras ersätter operativsystemets decimalavgränsare (regionala inställningar).

Syntax:

```
DecimalSep
```

Exempel:

```
Set DecimalSep='.';

Set DecimalSep=',';
```

FirstWeekDay

Heltal som definierar vilken dag som ska användas som den första dagen i veckan.

Syntax:

```
FirstWeekDay
```

Systemvariablerna Qlik Sense definierar `FirstWeekDay=6` som standard. Detta innebär att söndag är första dagen i veckan.

Värden kan därefter anges för FirstWeekDay

Värde	Dag
0	Måndag
1	Tisdag
2	Onsdag
3	Torsdag
4	Fredag
5	Lördag
6	Söndag

LongDayNames

Formatet som definieras ersätter operativsystemets format för långa veckodagsnamn (regionala inställningar).

Syntax:

```
LongDayNames
```

Exempel:

```
Set LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';
```

LongMonthNames

Formatet som definieras ersätter operativsystemets format för långa månadsnamn (regionala inställningar).

Syntax:

```
LongMonthNames
```

Exempel:

```
Set
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';
```

MoneyDecimalSep

Decimalavgränsaren som definieras ersätter operativsystemets decimalavgränsare för valuta (regionala inställningar).

Syntax:

```
MoneyDecimalSep
```

Exempel:

```
Set MoneyDecimalSep='.';
```

MoneyFormat

Formatet som definieras ersätter operativsystemets valutaformat (regionala inställningar).

Syntax:

```
MoneyFormat
```

Exempel:

```
Set MoneyFormat='$ #,##0.00; ($ #,##0.00)';
```

MoneyThousandSep

Tusentalsavgränsaren som definieras ersätter operativsystemets tusentalsavgränsare för valuta (regionala inställningar).

Syntax:

```
MoneyThousandSep
```

Exempel:

```
Set MoneyThousandSep=',';
```

MonthNames

Formatet som definieras ersätter operativsystemets format för månadsnamn (regionala inställningar).

Syntax:

```
MonthNames
```

Exempel:

```
Set MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';
```

NumericalAbbreviation

Den numeriska förkortningen ställer in vilken förkortning att använda för siffrors skalprefix, till exempel M för mega eller miljon (10^6) och µ för mikro (10^{-6}).

Syntax:

```
NumericalAbbreviation
```

Ställ in variabeln `Numerisk förkortning` till en sträng som innehåller en lista med förkortningsdefinitionspar, separerade med semikolon. Varje förkortningspar ska innehålla skalan (exponenten i decimalbas) och förkortningen separerad av ett kolon, till exempel `6:M` för en miljon.

Standardinställningen är `'3:k;6:M;9:G;12:T;15:P;18:E;21:Z;24:Y;-3:m;-6:µ;-9:n;-12:p;-15:f;-18:a;-21:z;-24:y'`.

Exempel:

Inställningen ändrar prefixet för tusen till t och prefixet för miljard till B. Det är användbart för ekonomiska applikationer där du kan förvänta dig förkortningar som t$, M$ och B$.

```
Set NumericalAbbreviation='3:t;6:M;9:B;12:T;15:P;18:E;21:Z;24:Y;-3:m;-6:µ;-9:n;-12:p;-15:f;-18:a;-21:z;-24:y';
```

ReferenceDay

Inställningen definierar vilken dag i januari som ska ställas in som referensdag för att definiera vecka 1.

Syntax:

```
ReferenceDay
```

Som standard använder Qlik Sense-funktioner 4 som referensdag. Det betyder att vecka 1 måste innehålla 4 januari, eller med andra ord, vecka 1 måste alltid bestå av minst 4 dagar i januari.

Följande värden kan användas för att ställa in en annan referensdag.

- 1 (= 1 januari)
- 2 (= 2 januari)
- 3 (= 3 januari)
- 4 (= 4 januari)
- 5 (= 5 januari)

- 6 (= 6 januari)
- 7 (= 7 januari)

Exempel:

```
Set ReferenceDay=3; //(set January 3 as the reference day)
```

ThousandSep

Tusentalsavgränsaren som definieras ersätter operativsystemets tusentalsavgränsare (regionala inställningar).

Syntax:

```
ThousandSep
```

Exempel:

```
Set ThousandSep=','; //(for example, seven billion must be specified as: 7,000,000,000)

Set ThousandSep=' ';
```

TimeFormat

Formatet som definieras ersätter operativsystemets format för tid (regionala inställningar).

Syntax:

```
TimeFormat
```

Exempel:

```
Set TimeFormat='hh:mm:ss';
```

TimestampFormat

Formatet som definieras ersätter operativsystemets format för datum och tid (regionala inställningar).

Syntax:

```
TimestampFormat
```

Exempel:

I följande exempel används *1983-12-14T13:15:30Z* som tidsmarkörsdata för att visa resultaten för olika **SET TimestampFormat**-satser. Datumformatet som används är **YYYYMMDD** och tidsformatet är **h:mm:ss TT**. Datumformatet anges i **SET DateFormat**-satsen och tidsformatet anges i **SET TimeFormat**-satsen längst upp i dataladdningsskriptet.

Resultat

Exempel	Resultat
`SET TimestampFormat='YYYYMMDD';`	19831214
`SET TimestampFormat='M/D/YY hh:mm:ss[.fff]';`	12/14/83 13:15:30
`SET TimestampFormat='DD/MM/YYYY hh:mm:ss[.fff]';`	14/12/1983 13:15:30
`SET TimestampFormat='DD/MM/YYYY hh:mm:ss[.fff] TT';`	14/12/1983 1:15:30 PM
`SET TimestampFormat='YYYY-MM-DD hh:mm:ss[.fff] TT';`	1983-12-14 01:15:30

Exempel: Laddningsskript

Exempel: Laddningsskript

I det första laddningsskriptet används *SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT'*. I det andra laddningsskriptet har formatet för tidsmarkören ändrats till *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'*. De olika resultaten visar hur **SET TimeFormat**-satsen fungerar med olika tidsformat.

Tabellen nedan visar datauppsättningen som används i laddningsskripten som följer. Tabellens andra kolumn innehåller formaten för varje tidsmarkör i datauppsättningen. De första fem tidsmarkörerna följer ISO 8601-normen, men det gör inte den sjätte.

Datauppsättning

Tabellen visar de tidsdata som används och formatet för varje tidsmarkör i datauppsättningen.

transaction_timestamp	time data format
2018-08-30	YYYY-MM-DD
20180830T193614.857	YYYYMMDDhhmmss.sss
20180830T193614.857+0200	YYYYMMDDhhmmss.sss±hhmm
2018-09-16T12:30-02:00	YYYY-MM-DDhh:mm±hh:mm
2018-09-16T13:15:30Z	YYYY-MM-DDhh:mmZ
9/30/18 19:36:14	M/D/YY hh:mm:ss

Gå till **Skriptredigeraren** och skapa ett nytt delavsnitt. Lägg sedan till exempelskriptet och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i din app för att se resultatet.

Laddningsskript

```
SET FirstWeekDay=0; SET BrokenWeeks=1; SET ReferenceDay=0; SET
DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun'; SET
LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday'; SET
DateFormat='YYYYMMDD'; SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT';   Transactions: Load
*, Timestamp(transaction_timestamp, 'YYYY-MM-DD hh:mm:ss[.fff]') as LogTimestamp ;   Load *
Inline [ transaction_id, transaction_timestamp, transaction_amount, transaction_quantity,
discount, customer_id, size, color_code 3750, 2018-08-30, 12423.56, 23, 0,2038593, L, Red
3751, 20180830T193614.857, 5356.31, 6, 0.1, 203521, m, orange 3752, 20180830T193614.857+0200,
```

```
15.75, 1, 0.22, 5646471, S, blue 3753, 2018-09-16T12:30-02:00, 1251, 7, 0, 3036491, l, Black
3754, 2018-09-16T13:15:30Z, 21484.21, 1356, 75, 049681, xs, Red 3755, 9/30/18 19:36:14, -
59.18, 2, 0.333333333333333, 2038593, M, Blue ];
```

Resultat

Qlik Sense-tabell som visar resultaten för TimestampFormat-tolkningsvariabeln som används i laddningsskriptet. Den sista tidsmarkören i datauppsättningen returnerar inte ett korrekt datum.

transaction_id	transaction_timestamp	LogTimeStamp
3750	2018-08-30	2018-08-30 00:00:00
3751	20180830T193614.857	2018-08-30 19:36:14
3752	20180830T193614.857+0200	2018-08-30 17:36:14
3753	2018-09-16T12:30-02:00	2018-09-16 14:30:00
3754	2018-09-16T13:15:30Z	2018-09-16 13:15:30
3755	9/30/18 19:36:14	-

Nästa laddningsskript använder samma datauppsättning. Det använder dock *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'* för att matcha formatet i den sjätte tidsmarkören, som inte följer ISO 8601-normen.

Gå till **Skriptredigeraren** och ersätt det tidigare exempelskriptet med skriptet nedan och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i din app för att se resultatet.

Laddningsskript

```
SET FirstWeekDay=0; SET BrokenWeeks=1; SET ReferenceDay=0; SET
DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun'; SET
LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday'; SET
DateFormat='YYYYMMDD'; SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]';   Transactions: Load
*, Timestamp(transaction_timestamp, 'YYYY-MM-DD hh:mm:ss[.fff]') as LogTimestamp ;   Load *
Inline [ transaction_id, transaction_timestamp, transaction_amount, transaction_quantity,
discount, customer_id, size, color_code 3750, 2018-08-30, 12423.56, 23, 0,2038593, L, Red
3751, 20180830T193614.857, 5356.31, 6, 0.1, 203521, m, orange 3752, 20180830T193614.857+0200,
15.75, 1, 0.22, 5646471, S, blue 3753, 2018-09-16T12:30-02:00, 1251, 7, 0, 3036491, l, Black
3754, 2018-09-16T13:15:30Z, 21484.21, 1356, 75, 049681, xs, Red 3755, 9/30/18 19:36:14, -
59.18, 2, 0.333333333333333, 2038593, M, Blue ];
```

Resultat

Qlik Sense-tabell som visar resultaten för TimestampFormat-tolkningsvariabeln som används i laddningsskriptet.

transaction_id	transaction_timestamp	LogTimeStamp
3750	2018-08-30	2018-08-30 00:00:00

transaction_id	transaction_timestamp	LogTimeStamp
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

2.15 Variabler i Direct Discovery

Systemvariabler i Direct Discovery

DirectCacheSeconds

Du kan ställa in en cachelagringsgräns till Direct Discovery-frågeresultaten för visualiseringar. När den här tidsgränsen har uppnåtts rensar Qlik Sense cacheminnet när nya Direct Discovery-frågor ställs. Qlik Sense ställer frågor om urval till datakällan och skapar cacheminnet igen för den designerade tidsgränsen. Resultatet för varje kombination av urval cachelagras fristående. Det innebär att cacheminnet uppdateras fristående för varje urval, så att ett urval uppdaterar cacheminnet enbart för de valda fälten, och ett andra urval uppdaterar cacheminnet för de berörda fälten. Om det andra urvalet omfattar fält som uppdaterades i det första urvalet uppdateras de inte i cacheminnet igen om inte cachelagringsgränsen har uppnåtts.

Direct Discovery-cacheminnet gäller inte för **Tabell**-visualiseringar. Tabellurvalen frågar datakällan varje gång.

Gränsvärdet måste anges i sekunder. Standardgränsen för cachelagring är 1 800 sekunder (30 minuter).

Det värde som används för **DirectCacheSeconds** är det värde som är inställt vid tidpunkten då **DIRECT QUERY**-satsen exekveras. Värdet kan inte ändras under körning.

Exempel:

```
SET DirectCacheSeconds=1800;
```

DirectConnectionMax

Du kan göra asynkrona, parallella anrop mot databasen med hjälp av funktionen för anslutningspoolning. Laddningsskriptsyntaxen för konfigurering av anslutningspoolning ser ut som följer:

```
SET DirectConnectionMax=10;
```

Den numeriska inställningen anger det maximala antalet databaskopplingar som Direct Discovery-koden ska använda vid uppdatering av ett ark. Standardinställningen är 1.

Den här variabeln bör användas försiktigt. Att ställa in den till ett större värde än 1 brukar orsaka problem vid anslutning till Microsoft SQL Server.

DirectUnicodeStrings
Direct Discovery kan stödja urval av utökade Unicode-data med hjälp av SQL -standardformatet för stränglitteraler med utökad teckensträng (N'<utökad sträng>') som krävs av vissa databaser (framför allt SQL Server). Användning av denna syntax kan aktiveras för Direct Discovery med hjälp av skriptvariabeln **DirectUnicodeStrings**.

Om variabeln anges till "true" kan du använda den standardmässiga breda ANSI-teckenmarkören "N" framför stränglitteralerna. Alla databaser stöder inte den här standarden. Standardinställningen är "false".

DirectDistinctSupport
Om ett **DIMENSION**-fältvärde väljs i ett Qlik Sense-objekt skapas en fråga för källdatabasen. Om frågan kräver gruppering använder Direct Discovery nyckelordet **DISTINCT** för att välja enbart unika värden. Vissa databaser kräver dock nyckelordet **GROUP BY**. Ställ in **DirectDistinctSupport** på 'false' för att generera **GROUP BY** i stället för **DISTINCT** i frågor för unika värden.

```
SET DirectDistinctSupport='false';
```
Om DirectDistinctSupport är inställt på true används **DISTINCT**. Om det inte är inställt är standardbeteendet att använda **DISTINCT**.

DirectEnableSubquery
I scenarier med hög kardinalitet och flera tabeller är det möjligt att generera delfrågor i SQL-frågan istället för att generera en lång IN-sats. Aktivera detta genom att ställa in **DirectEnableSubquery** på 'true'. Standardvärdet är 'false'.

> *Om du har aktiverat **DirectEnableSubquery** kan du inte ladda tabeller som inte är i läget Direct Discovery.*

```
SET DirectEnableSubquery='true';
```

Variabler för Teradata-query banding

Teradata Query Banding är en funktion som större organisationer kan använda mot den underliggande Teradata-databasen för bättre redovisning, prioritering och hantering av arbetsbelastningen. Med hjälp av query banding kan man omsluta en fråga med metadata, exempelvis med autentiseringsuppgifter för användare.

Det finns två varianter. Båda är strängar som utvärderas och skickas till databasen.

SQLSessionPrefix
Den här strängen skickas när en koppling till databasen skapas.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR SESSION;';
```
Om till exempel **OSuser()** returnerar *WA\sbt* utvärderas detta till `SET QUERY_BAND = 'Who=WA\sbt;' FOR SESSION;`, vilket skickas till databasen när kopplingen skapas.

SQLQueryPrefix
Strängen skickas för varje enskild fråga.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR TRANSACTION;';
```

Direct Discovery-teckenvariabler

DirectFieldColumnDelimiter

Du kan ställa in det tecken som används som fältavgränsare i **Direct Query**-satser för databaser som kräver ett annat tecken än komma som fältavgränsare. Det angivna tecknet måste omges av enkla citattecken i **SET**-satsen.

```
SET DirectFieldColumnDelimiter= '|'
```

DirectStringQuoteChar

Du kan ange ett tecken som ska användas för att förse strängarna i en genererad fråga med citattecken. Standardvärdet är enkla citattecken. Det angivna tecknet måste omges av enkla citattecken i **SET**-satsen.

```
SET DirectStringQuoteChar= '"';
```

DirectIdentifierQuoteStyle

Du kan ange att icke-ANSI-citatmarkering av identifierare ska användas i genererade frågor. För närvarande är den enda tillgängliga icke-ANSI-citatmarkeringen GoogleBQ. ANSI är standard. Versaler, gemener och versaler/gemener blandat kan användas (ANSI, ansi, Ansi).

```
SET DirectIdentifierQuoteStyle="GoogleBQ";
```

ANSI-citatmarkering används till exempel i följande **SELECT**-sats:

```
SELECT [Quarter] FROM [qvTest].[sales] GROUP BY [Quarter]
```

Om **DirectIdentifierQuoteStyle** är inställd på "GoogleBQ" skulle **SELECT**-satsen använda citatmarkering på följande sätt:

```
SELECT [Quarter] FROM [qvTest.sales] GROUP BY [Quarter]
```

DirectIdentifierQuoteChar

Du kan ange ett tecken för att kontrollera citatmarkeringen av identifierare i en genererad fråga. Detta kan ställas in till antingen ett tecken (exempelvis ett dubbelt citattecken) eller två (exempelvis ett par hakparenteser). Standardvärdet är dubbla citattecken.

```
SET DirectIdentifierQuoteChar='[]';
SET DirectIdentifierQuoteChar='``';
SET DirectIdentifierQuoteChar=' ';
SET DirectIdentifierQuoteChar='""';
```

DirectTableBoxListThreshold

När Direct Discovery-fält används i en **Tabell**-visualisering, ställs ett gränsvärde in för att begränsa det antal rader som visas. Standardgränsen är 1 000 poster. Standardinställningen för gränsvärdet kan ändras med hjälp av variabeln **DirectTableBoxListThreshold** i laddningsskriptet. Exempel:

```
SET DirectTableBoxListThreshold=5000;
```

Gränsinställningen gäller enbart för **Tabell**-visualiseringar som innehåller Direct Discovery-fält. **Tabell**-visualiseringar som innehåller endast minnesfält begränsas inte av inställningen **DirectTableBoxListThreshold**.

Inga fält visas i **Tabell**-visualiseringen förrän urvalet har färre poster än gränsinställningen.

Direct Discovery-variabler för tolkning av tal

DirectMoneyDecimalSep

Den definierade decimalavgränsaren ersätter decimalsymbolen för valuta i SQL-satsen som genereras för att ladda data med Direct Discovery. Det här tecknet måste matcha det tecken som används i **DirectMoneyFormat**.

Standardvärdet är `'.'`

Exempel:

```
Set DirectMoneyDecimalSep='.';
```

DirectMoneyFormat

Den definierade symbolen ersätter valutaformatet i den SQL-sats som genereras för att ladda data med hjälp av Direct Discovery. Valutasymbolen för tusentalsavgränsaren ska inte tas med.

Standardvärdet är `'#.0000'`

Exempel:

```
Set DirectMoneyFormat='#.0000';
```

DirectTimeFormat

Det definierade tidsformatet ersätter tidsformatet i den SQL-sats som genereras för att ladda data med hjälp av Direct Discovery.

Exempel:

```
Set DirectTimeFormat='hh:mm:ss';
```

DirectDateFormat

Det definierade datumformatet ersätter datumformatet i den SQL-sats som genereras för att ladda data med hjälp av Direct Discovery.

Exempel:

```
Set DirectDateFormat='MM/DD/YYYY';
```

DirectTimeStampFormat

Det definierade formatet ersätter datum- och tidsformatet i den SQL-sats som genereras för att ladda data med hjälp av Direct Discovery.

Exempel:

```
Set DirectTimestampFormat='M/D/YY hh:mm:ss[.fff]';
```

2.16 Felvariabler

Värdena i alla felvariablerna kvarstår efter skriptexekveringen. Den första variabeln, ErrorMode, är indata från användaren. De sista tre är utdata från Qlik Sense med information om fel i skriptet.

Felvariabler – en översikt

Varje variabel beskrivs ytterligare efter översikten. Du kan även klicka på namnet på variabeln i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika variabeln.

Se onlinehjälpen till Qlik Sense för mer information om variablerna.

ErrorMode
Denna felvariabel avgör hur Qlik Sense ska reagera när ett fel upptäcks under skriptexekveringen.

```
ErrorMode
```

ScriptError
Denna felvariabel returnerar den senast exekverade skriptsatsens felkod.

```
ScriptError
```

ScriptErrorCount
Denna felvariabel returnerar det totala antalet satser som har orsakat fel under den aktuella skriptexekveringen. Denna variabel återställs alltid till 0 när skriptexekvering påbörjas.

```
ScriptErrorCount
```

ScriptErrorList
Denna felvariabel innehåller en konkatenerad lista över de skriptfel som har inträffat under den senaste exekveringen. Felen avgränsas med en radmatning.

```
ScriptErrorList
```

ErrorMode

Denna felvariabel avgör hur Qlik Sense ska reagera när ett fel upptäcks under skriptexekveringen.

Syntax:

```
ErrorMode
```

Argument:

Argument

Argument	Beskrivning
ErrorMode=1	Standardinställningen. Skriptexekveringen avbryts och användaren ombeds att agera (non-batch mode).
ErrorMode =0	Qlik Sense ignorerar felet och fortsätter skriptexekveringen vid nästa skriptsats.

Argument	Beskrivning
ErrorMode =2	Qlik Sense utlöser ett felmeddelande "Skriptexekveringen misslyckades..." genast, utan att användaren uppmanas att agera i förväg.

Exempel:

```
set ErrorMode=0;
```

ScriptError

Denna felvariabel returnerar den senast exekverade skriptsatsens felkod.

Syntax:

```
ScriptError
```

Denna variabel återställs till 0 efter varje lyckad skriptexekvering. Om ett fel inträffar kommer en intern Qlik Sense-felkod att användas. Felkoden består av ett dualt värde, ett numeriskt värde och ett textvärde: Följande felkoder finns:

Skriptfelkoder

Felkod	Beskrivning
0	Inget fel. Text för dualt värde är tom.
1	Allmänt fel.
2	Syntaxfel.
3	Allmänt ODBC-fel.
4	Allmänt OLE DB-fel.
5	Generellt fel från anpassad databas.
6	Allmänt XML-fel.
7	Allmänt HTML-fel.
8	Det går inte att hitta filen.
9	Det går inte att hitta databasen.
10	Det går inte att hitta tabellen.
11	Det går inte att hitta fältet.

Felkod	Beskrivning
12	Fel filformat.
16	Semantiskt fel.

Exempel:

```
set ErrorMode=0;

LOAD * from abc.qvf;

if ScriptError=8 then

exit script;

//no file;

end if
```

ScriptErrorCount

Denna felvariabel returnerar det totala antalet satser som har orsakat fel under den aktuella skriptexekveringen. Denna variabel återställs alltid till 0 när skriptexekvering påbörjas.

Syntax:

```
ScriptErrorCount
```

ScriptErrorList

Denna felvariabel innehåller en konkatenerad lista över de skriptfel som har inträffat under den senaste exekveringen. Felen avgränsas med en radmatning.

Syntax:

```
ScriptErrorList
```

2 Skriptuttryck

Uttryck kan användas i både **LOAD**-satser och **SELECT**-satser. De uttryck och funktioner som beskrivs här gäller endast **LOAD**-satsen, och inte **SELECT**-satsen, eftersom den tolkas av ODBC-drivrutinen och inte av Qlik Sense. De flesta ODBC-drivrutiner kan dock tolka ett antal av de funktioner som beskrivs nedan.

Uttryck består av funktioner, fält och operatorer som kombineras i en syntax.

Alla uttryck i Qlik Sense-skriptet resulterar i ett tal och/eller en sträng, beroende på vad som är lämpligt. Logiska funktioner och operatorer returnerar 0 för False och -1 för True. Konverteringar från tal till textsträng eller vice versa är implicita. Logiska operatorer och funktioner tolkar 0 som False och alla andra som True.

Den allmänna syntaxen för ett uttryck är:

Allmän syntax

Uttryck	Fält	Operator
expression ::= (constant	constant	|
expression ::= (constant	fieldref	|
expression ::= (constant	operator1 expression	|
expression ::= (constant	expression operator2 expression	|
expression ::= (constant	function	|
expression ::= (constant	(expression)	)

där:

- **constant** är en sträng (en text, ett datum, en tid) inom enkla, raka citationstecken, eller ett tal. Konstanter skrivs utan tusentalsavgränsare och med decimalkomma som decimalavgränsaren.
- **fieldref** är ett fältnamn i den inlästa tabellen.
- **operator1** är en unär operator (som arbetar med ett uttryck, det till höger).
- **operator2** är en binär operator (som arbetar med två uttryck, ett på varje sida).
- **function ::= functionname(parameters)**
- **parameters ::= expression { , expression }**

Varken typen av parameter eller antalet parametrar är godtyckligt. utan beror på funktionen som används.

Uttryck och funktioner kan således nästlas efter behag, och så länge som uttrycket resulterar i ett värde som går att tolka kommer Qlik Sense inte att ge några felmeddelanden.

3 Diagramuttryck

Ett diagram (visualisering) är en kombination av funktioner, fält, matematiska operatorer (+ * / =) och andra mått. Uttryck används för att bearbeta data i appen i syfte att skapa ett resultat som kan ses i en visualisering. De är inte begränsade till användning i mått. Du kan bygga visualiseringar som är mer dynamiska och effektiva med uttryck för rubriker, underrubriker, fotnoter och till och med dimensioner.

Detta innebär till exempel att istället för att låta en visualiserings rubrik vara statisk text kan den skapas från ett uttryck som ger olika resultat beroende på vilka val som görs.

En detaljerad referens om skriptfunktioner och diagramfunktioner finns i Skriptsyntax och diagramfunktioner.

3.1 Definiera aggregeringens omfattning

Det finns vanligtvis två faktorer som tillsammans avgör vilka poster som används för att definiera värdet av en aggregering i ett uttryck. När vi arbetar med visualiseringar är dessa faktorer:

- Dimensionsvärde (för en aggregering i ett diagramuttryck)
- Val

Tillsammans definierar dessa faktorer omfattningen för aggregeringen. Du kan komma att stöta på situationer där du vill att beräkningen ska bortse från urvalet, dimensionen eller bådadera. I diagramfunktioner kan du åstadkomma detta genom att använda kvalificeraren TOTAL, set-analys eller en kombination av dessa.

Aggregering: Metod och beskrivning

Metod	Beskrivning
Kvalificeraren TOTAL	När du använder total-kvalificeraren i en aggregeringsfunktion ignoreras dimensionsvärdet. Aggregeringen utförs på alla möjliga fältvärden. Bestämningen **TOTAL** kan följas av en lista med ett eller flera fältnamn inom vinkelparenteser. Dessa fältnamn bör ingå i en underuppsättning av diagrammets dimensionsvariabler. Vid uträkningen beaktas i detta fall alla diagrammets dimensionsvariabler utom de som ingår i listan, d.v.s. ett värde returneras för varje kombination av fältvärden i dimensionsfälten som finns i listan. Även fält som för närvarande inte utgör en dimension i ett diagram kan ingå i listan. Detta är användbart när man arbetar med grupper av dimensioner där dimensionsfälten inte är fasta. Genom att lista alla variabler i gruppen säkerställs att funktionen fungerar när den hierarkiska nivån ändras.

Metod	Beskrivning
Set-analys	Vid användning av set-analys inuti aggregeringen åsidosätts urvalet. Aggregeringen utförs på alla värden delat över dimensionerna.
Kvalificeraren TOTAL och set-analys	När kvalificeraren **TOTAL** och set-analys används inuti din aggregering åsidosätts urvalet och dimensionerna ignoreras.

Kvalificeraren ALL	När kvalificeraren **ALL**används inuti aggregeringen ignoreras urvalet och dimensionerna. Motsvarande resultat kan uppnås med set-analys-satsen {1} och kvalificeraren **TOTAL** :
	`=sum(All Sales)`
	`=sum({1} Total Sales)`

Exempel: Kvalificeraren TOTAL

I följande exempel visas hur TOTAL kan användas för att beräkna en relativ andel. Anta att Q2 har valts. Om du använder TOTAL beräknas summan av alla värden utan hänsyn till dimensionerna.

Exempel: Kvalificeraren TOTAL

Year	Quarter	Sum(Amount)	Sum(TOTAL Amount)	Sum(Amount)/Sum(TOTAL Amount)
		3000	3000	100%
2012	Q2	1700	3000	56,7%
2013	Q2	1300	3000	43,3%

*Om du vill visa talen som en procentandel går du till egenskapspanelen för det mått som du vill visa som ett procentvärde. Under **Number formatting** väljer du **Number** och som **Formatting** väljer du **Simple** och något av procentformaten.*

Exempel: Set-analys

I följande exempel visas hur set-analys kan användas för att göra en jämförelse mellan datauppsättningar innan något urval har gjorts. Anta att Q2har valts. Vid användning av set-analys med set-definitionen {1} beräknas summan av alla värden utan hänsyn till urval men delat med dimensionerna.

Exempel: Set-analys

Year	Quarter	Sum(Amount)	Sum({1} Amount)	Sum(Amount)/Sum({1} Amount)
		3000	10800	27,8%
2012	Q1	0	1100	0%

Year	Quarter	Sum(Amount)	Sum({1} Amount)	Sum(Amount)/Sum({1} Amount)
2012	Q3	0	1400	0%
2012	Q4	0	1800	0%
2012	Q2	1700	1700	100%
2013	Q1	0	1000	0%
2013	Q3	0	1100	0%
2013	Q4	0	1400	0%
2013	Q2	1300	1300	100%

Exempel: Kvalificeraren TOTAL och set-analys

I följande exempel visas hur set-analys och kvalificeraren TOTAL kan kombineras för att göra en jämförelse mellan datauppsättningar innan något urval har gjorts och över alla dimensioner. Om man antar att Q2 har valts används set-analys med set-definitionen {1} och kvalificeraren TOTAL för att beräkna summan av alla värden utan hänsyn till urval och utan hänsyn till dimensionerna.

Exempel: Kvalificeraren TOTAL och set-analys

Year	Quarter	Sum (Amount)	Sum({1} TOTAL Amount)	Sum(Amount)/Sum({1} TOTAL Amount)
		3000	10800	27,8%
2012	Q2	1700	10800	15,7%
2013	Q2	1300	10800	12%

Data som används i exempel:

```
AggregationScope: LOAD * inline [ Year Quarter Amount 2012 Q1 1100 2012 Q2 1700 2012 Q3 1400
2012 Q4 1800 2013 Q1 1000 2013 Q2 1300 2013 Q3 1100 2013 Q4 1400] (delimiter is ' ');
```

3.2 Set-analys

När du gör ett urval i en app definierar du en delmängd av posterna i data. Aggregeringsfunktioner, som `Sum()`, `Max()`, `Min()`, `Avg()` och `Count()` beräknas baserat på den här delmängden.

Ditt urval definierar med andra ord aggregeringens omfattning; den definierar uppsättningen med poster som beräkningarna görs på.

Set-analys gör det möjligt att definiera en omfattning som är olik den uppsättning med poster som definieras av det aktuella urvalet. Den här nya omfattningen kan även betraktas som ett alternativt urval.

Det här kan vara användbart om du vill jämföra det aktuella urvalet med ett visst värde, till exempel förra årets värde eller den globala marknadsandelen.

Set-uttryck

Set-uttryck används i aggregeringsfunktioner och inom en klammerparentes. Exempel:

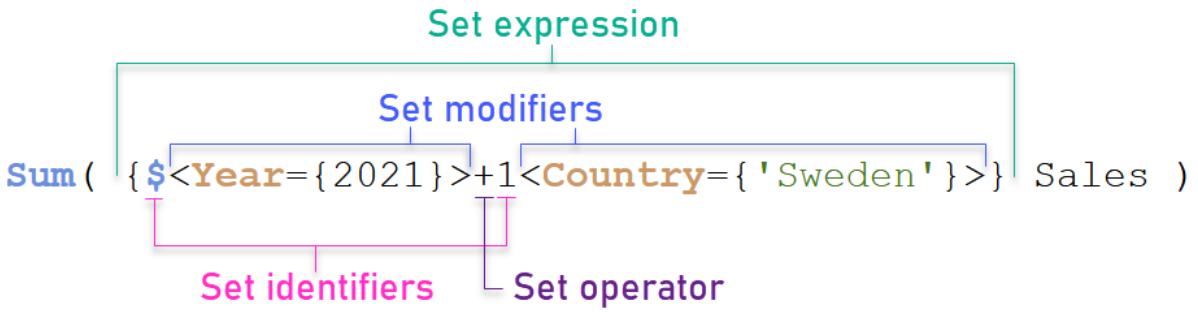
```
Sum( {$<Year={2021}>} Sales )
```

Ett set-uttryck består av en kombination av följande element:

- **Identifierare**. En set-identifierare motsvarar ett urval, som definieras på en annan plats. Den motsvarar även en specifik uppsättning med poster i data. Det kan vara det aktuella urvalet, ett urval från ett bokmärke eller ett urval från ett parallellt tillstånd. Ett enkelt set-uttryck består av en enda identifierare, som t.ex. dollartecknet {$}, vilket innebär alla poster i aktuellt urval.
  Exempel: `$`, `1`, `BookMark1`, `State2`
- **Operatorer**. En set-operator kan användas till att skapa unioner, differenser eller intersektioner mellan set-identifierare. På det sättet kan du skapa en överordnad uppsättning av urvalen som definierats av set-identifierarna.
  Exempel: `+`, `-`, `*`, `/`
- **Modifierare** En set-modifierare kan läggas till i set-identifieraren för att ändra dess urval. En modifierare kan även användas på egen hand och kommer sedan att modifiera standardidentifieraren. En modifierare måste omslutas av vinkelparenteser `<…>`.
  Exempel: `<Year={2020}>`, `<Supplier={ACME}>`

Elementen kombineras för att skapa set-uttryck.

Element i ett set-uttryck



Set-uttrycket ovan är till exempel byggt från aggregeringen `Sum(Sales)`.

Den första operanden returnerar försäljning för året `2021` för det aktuella urvalet, vilket indikeras med set-identifieraren `$` och modifieraren som innehåller urvalet för år `2021`. Den andra operanden returnerar `Sales` för `Sweden` och ignorerar det aktuella urvalet, vilket indikeras med set-identifieraren `1`.

Slutligen returnerar uttrycket en uppsättning som består av posterna som tillhör någon av de två set-operanderna, enligt indikation från set-operatorn `+`.

Exempel

Exempel som kombinerar set-uttryckselementen ovan är tillgängliga i följande ämnen:

Naturliga uppsättningar

Vanligen motsvarar ett set-uttryck både en uppsättning med poster i datamodellen och ett urval som definierar den här delmängden data. I detta fall kallas uppsättningen en naturlig uppsättning.

Set-identifierare, med eller utan set-modifierare, representerar alltid naturliga uppsättningar.

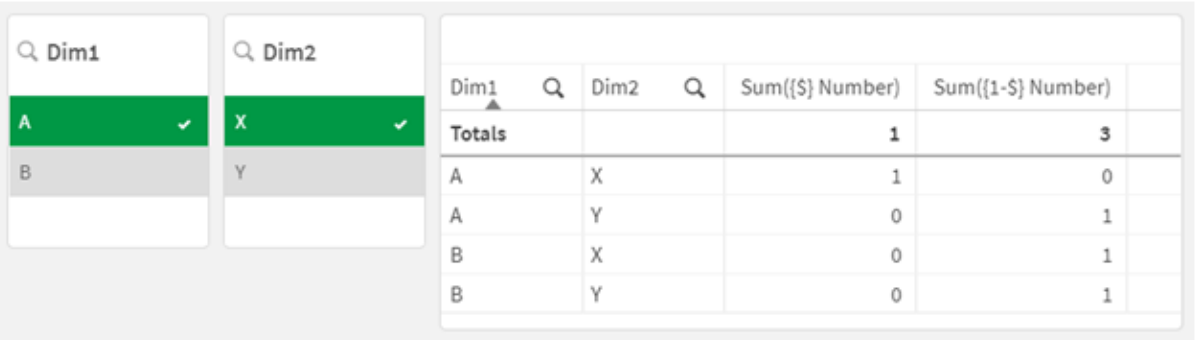
Ett set-uttryck som använder set-operatorer representerar dock även en delmängd av posterna, men kan generellt ändå inte beskrivas som ett urval med fältvärden. Ett sådant uttryck är en icke-naturlig uppsättning.

Till exempel kan uppsättningen som ges av `{1-$}` inte alltid definieras av ett urval. Det innebär att det inte är en naturlig uppsättning. Det går att visa genom att ladda följande data, lägga till dem i en tabell och sedan göra urvalen med filterpaneler.

```
Load * Inline [Dim1, Dim2, Number A, X, 1 A, Y, 1 B, X, 1 B, Y, 1];
```

Genom att göra urval för `Dim1` och `Dim2` får du vyn som visas i följande tabell.

Tabell med naturliga och icke-naturliga uppsättningar



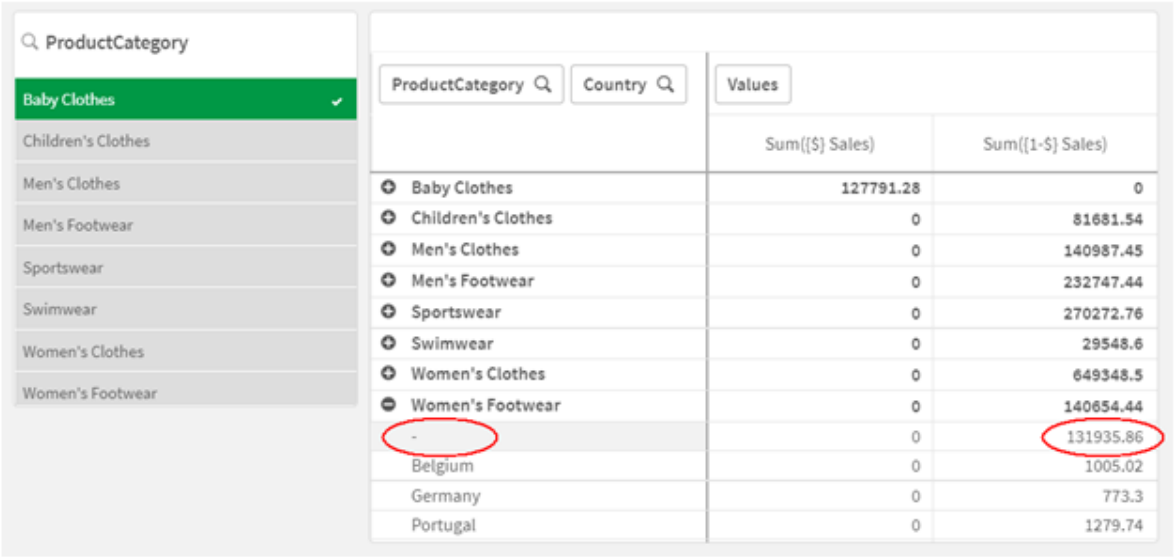
Set-uttrycket i den första åtgärden använder en naturlig uppsättning: det motsvarar urvalet som görs `{$}`.

Den andra åtgärden är annorlunda. Den använder `{1-$}`. Det går inte att göra ett urval som motsvarar den här uppsättningen, så det är en icke-naturlig uppsättning.

Den här distinktionen har ett flertal konsekvenser:

- Set-modifierare går endast att tillämpa på set-identifierare. De går inte att tillämpa på ett godtyckligt set-uttryck. Till exempel är det inte möjligt att använda ett set-uttryck som:
  `{ (BM01 * BM02) <Field={x,y}> }`
  Här betyder de vanliga (runda) parenteserna att intersektionen mellan `BM01` och `BM02` ska utvärderas innan set-modifieraren tillämpas. Skälet är att det inte finns någon uppsättning av element som kan modifieras.
- Du kan inte använda icke-naturliga uppsättningar i `P()`- och `E()`-elementfunktioner. Dessa funktioner returnerar en uppsättning av element, men det går inte att härleda en uppsättning från en icke-naturlig uppsättning.

- Ett mått som använder en icke-naturlig uppsättning kan inte alltid attribueras till rätt dimensionsvärde om datamodellen har flera tabeller. I följande diagram attribueras vissa uteslutna försäljningssiffror till rätt Country, medan andra har NULL som Country.
  Diagram med icke-naturlig uppsättning



  Huruvida tilldelningen är korrekt eller inte beror på datamodellen. I det här fallet kan talet inte tilldelas om det berör ett land som utesluts av urvalet.

Set-identifierare

En set-identifierare motsvarar en uppsättning med poster i data; antingen alla data eller en delmängd av data. Det är uppsättningen med poster som definieras av ett urval. Det kan vara det aktuella urvalet, alla data (inget urval), ett urval från ett bokmärke eller ett urval från ett parallellt tillstånd.

I det här exemplet `Sum( {$<Year = {2009}>} Sales )` är identifieraren dollartecknet: `$`. Det motsvarar det aktuella urvalet. Det representerar även alla möjliga poster. Den här uppsättningen kan sedan ändras med modifierardelen i set-uttrycket: urvalet `2009` i `Year` läggs till.

I ett mer komplicerat set-uttryck går det att använda två identifierare tillsammans med en operator för att bilda en union, en differens eller en skärningspunkt för de två uppsättningarna med poster.

Följande tabell visar några gemensamma identifierare.

Exempel med gemensamma identifierare

Identifierare	Beskrivning
1	Representerar hela uppsättningen av alla poster i applikationen, oavsett vilka urval som har gjorts.

Identifierare	Beskrivning
`$`	Representerar posterna i det aktuella urvalet i standardtillståndet. Set-uttrycket `{$}` är följaktligen vanligen motsvarigheten till att inte ange ett set-uttryck.
`$1`	Motsvarar det föregående urvalet i standardtillståndet. `$2` motsvarar det föregående urvalet utom ett och så vidare.
`$_1`	Representerar nästa urval (framåt). `$_2` representerar nästa urval utom ett och så vidare.
`BM01`	Du kan använda vilket bokmärkes-ID eller bokmärkesnamn du vill.
`AltState`	Du kan hänvisa till ett parallellt tillstånd med hjälp av tillståndets namn.
`AltState::BM01`	Ett bokmärke innehåller urval av alla tillstånd och du kan hänvisa till ett specifikt bokmärke genom att kvalificera bokmärkets namn.

Följande tabell visar exempel med olika identifierare.

Exempel med olika identifierare

Exempel	Resultat
`Sum ({1} Sales)`	Returnerar den totala försäljningen för appen, oavsett urval men inte oavsett dimensionen.
`Sum ({$} Sales)`	Returnerar försäljning för det aktuella urvalet, alltså detsamma som `Sum(Sales)`.
`Sum ({$1} Sales)`	Returnerar försäljning för föregående urval.
`Sum ({BM01} Sales)`	Returnerar försäljningen för bokmärket med namnet`BM01`.

Set-operatorer

Set-operatorer används till att inkludera, utesluta eller genomskära datauppsättningar. Alla operatorer använder uppsättningar som operander och returnerar en uppsättning som resultat.

Du kan använda set-operatorer i två olika situationer:

- För att utföra en set-åtgärd på set-identifierare, som motsvarar uppsättningar med poster i data.
- För att utföra en set-åtgärd på elementuppsättningar, på fältvärden eller i en set-modifierare.

Följande tabell visar operatorer som kan användas i set-uttryck.

Operatorer

Operator	Beskrivning
`+`	Union. Denna binära operation returnerar en uppsättning som består av de poster eller element som tillhör någon av de två set-operanderna.

Operator	Beskrivning
-	Exklusion. Denna binära operation returnerar en uppsättning som består av de poster eller element som enbart tillhör den första men inte den andra av de två set-operanderna. Den returnerar dessutom det komplementära setet när det används som en unär operator.
*	Intersektion. Denna binära operation returnerar en uppsättning som består av de poster eller element som tillhör båda set-operanderna.
/	Symmetrisk skillnad (XOR). Denna binära operation returnerar en uppsättning som består av de poster eller element som tillhör endera men inte båda set-operanderna.

Följande tabell visar exempel med operatorer.

Exempel med operatorer

Exempel	Resultat
`Sum ({1-$} Sales)`	Returnerar försäljningen för allt som det aktuella urvalet exkluderar.
`Sum ({$*BM01} Sales)`	Returnerar försäljningen för skärningspunkten mellan urvalet och bokmärket `BM01`.
`Sum ({-($+BM01)} Sales)`	Returnerar försäljning som exkluderas av urvalet och bokmärket `BM01`.
`Sum ({$<Year={2009}>+1<Country={'Sweden'}>} Sales)`	Returnerar försäljningen för år 2009 associerat till de aktuella urvalen och lägger till den fullständiga datauppsättningen associerad med landet `Sweden` för alla år.
`Sum ({$<Country={"S*"}+{"*land"}>} Sales)`	Returnerar försäljningen för länder som börjar med `S` eller slutar med `land`.

Set-modifierare

Set-uttryck används för att definiera omfattningen av en beräkning. Den centrala delen av set-uttrycket är set-modifieraren som specificerar ett urval. Det används till att ändra användarens urval eller urvalet i set-identifieraren och resultatet definierar en ny omfattning för beräkningen.

Set-modifieraren består av ett eller flera fältnamn, som vart och ett följs av ett urval som ska göras på just det fältet. Modifieraren omsluts av vinkelparenteser: `< >`

Exempel:

- `Sum ( {$<Year = {2015}>} Sales )`
- `Count ( {1<Country = {Germany}>} distinct OrderID )`
- `Sum ( {$<Year = {2015}, Country = {Germany}>} Sales )`

Uppsättningar av element

En elementuppsättning kan definieras med följande:

- En lista med värden
- En sökning
- En referens till ett annat fält
- En uppsättningsfunktion

Om elementuppsättningsdefinitionen utesluts kommer set-modifieraren att rensa alla urval i det här fältet. Exempel:

```
Sum( {$<Year = >} Sales )
```

Exempel: Diagramuttryck för set-modifierare som baseras på elementuppsättningar

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
MyTable: Load * Inline [ Country, Year, Sales Argentina, 2014, 66295.03 Argentina, 2015, 140037.89 Austria, 2014, 54166.09 Austria, 2015, 182739.87 Belgium, 2014, 182766.87 Belgium, 2015, 178042.33 Brazil, 2014, 174492.67 Brazil, 2015, 2104.22 Canada, 2014, 101801.33 Canada, 2015, 40288.25 Denmark, 2014, 45273.25 Denmark, 2015, 106938.41 Finland, 2014, 107565.55 Finland, 2015, 30583.44 France, 2014, 115644.26 France, 2015, 30696.98 Germany, 2014, 8775.18 Germany, 2015, 77185.68 ];
```

Diagramuttryck

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

Tabell – set-modifierare baserade på elementuppsättningar

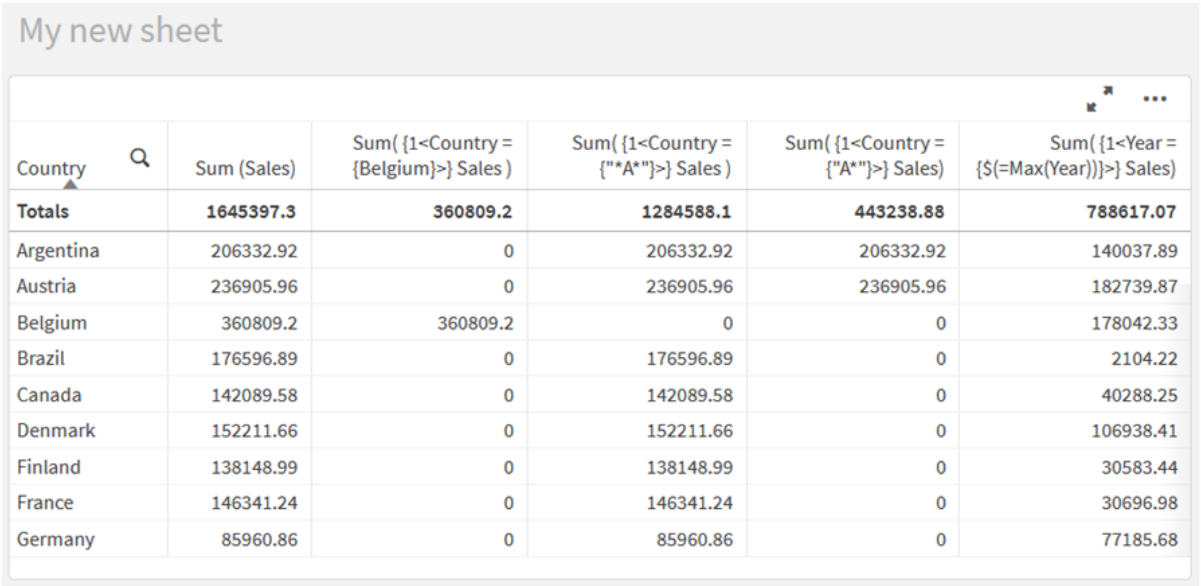
Land	Sum(Sales)	Sum ({1<Country= {Belgium}>} Sales)	Sum ({1<Country= {"*A*"}>} Sales)	Sum ({1<Country= {"A*"}>} Sales)	Sum ({1<Year= {$(=Max (Year))}>} Sales)
Totalvärden	1645397.3	360809.2	1284588.1	443238.88	788617.07
Argentina	206332.92	0	206332.92	206332.92	140037.89
Österrike	236905.96	0	236905.96	236905.96	182739.87
Belgien	360809.2	360809.2	0	0	178042.33
Brasilien	176596.89	0	176596.89	0	2104.22
Kanada	142089.58	0	142089.58	0	40288.25
Danmark	152211.66	0	152211.66	0	106938.41

Land	Sum(Sales)	Sum ({1<Country= {Belgium}>} Sales)	Sum ({1<Country= {"*A*"}>} Sales)	Sum ({1<Country= {"A*"}>} Sales)	Sum ({1<Year= {$(=Max (Year))}>} Sales)
Finland	138148.99	0	138148.99	0	30583.44
Frankrike	146341.24	0	146341.24	0	30696.98
Germany	85960.86	0	85960.86	0	77185.68

Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Sales)`
    Summa `Sales` utan set-uttryck.
  - `Sum({1<Country={Belgium}>}Sales)`
    Välj `Belgium` och sedan summan som motsvarar `Sales`.
  - `Sum({1<Country={"*A*"}>}Sales)`
    Välj alla länder som har ett `A` och sedan summan som motsvarar `Sales`.
  - `Sum({1<Country={"A*"}>}Sales)`
    Välj alla länder som börjar på `A` och sedan summan som motsvarar `Sales`.
  - `Sum({1<Year={$(=Max(Year))}>}Sales)`
    Beräkna `Max(Year)`, vilket är `2015`, och sedan summan som motsvarar `Sales`.

Set-modifierare baserade på elementuppsättningar



Värden i listan

Det vanligaste exemplet på en elementuppsättning är en uppsättning som baseras på en lista med fältvärden som omsluts av klammerparenteser. Exempel:

- `{$<Country = {Canada, Germany, Singapore}>}`
- `{$<Year = {2015, 2016}>}`

De inre klammerparenteserna definierar uppsättningen av element. De individuella värdena separeras med komman.

Citattecken och skiftlägeskänslighet

Om värdena innehåller blanksteg eller specialtecken behöver värdena citeras. Enkla citattecken ger en exakt, skiftlägeskänslig matchning med ett enda fältvärde. Dubbla citattecken innebär en skiftlägesokänslig matchning med ett eller flera fältvärden. Exempel:

- `<Country = {'New Zealand'}>`
  Matchar endast `New Zealand`.
- `<Country = {"New Zealand"}>`
  Matchar `New Zealand`, `NEW ZEALAND` och `new zealand`.

Datum måste stå inom citattecken och använda datumformatet från fältet i fråga. Exempel:

- `<ISO_Date = {'2021-12-31'}>`
- `<US_Date = {'12/31/2021'}>`
- `<UK_Date = {'31/12/2021'}>`

Dubbla citattecken kan bytas ut av hakparenteser eller grava accenter.

Sökningar

Uppsättningar av element kan även skapas genom sökningar. Exempel:

- `<Country = {"C*"}>`
- `<Ingredient = {"*garlic*"}>`
- `<Year = {">2015"}>`
- `<Date = {">12/31/2015"}>`

Jokertecken kan användas i textsökningar: En asterisk (`*`) motsvarar ett obegränsat antal tecken och ett frågetecken (`?`) motsvarar ett enda tecken. Relationsoperatörer kan användas till att definiera numeriska sökningar.

Du bör alltid använda dubbla citattecken för sökningar. Sökningar är inte skiftlägeskänsliga.

Dollarexpansioner

Dollarexpansioner behövs om du vill använda en beräkning i din elementuppsättning. Om du till exempel endast vill titta på det sista möjliga året använder du:

```
<Year = {$(=Max(Year))}>
```

Valda värden i andra fält

Modifierare kan basera sig på de valda värdena i ett annat fält. Exempel:

```
<OrderDate = DeliveryDate>
```

Denna modifierare tar de valda värdena från `DeliveryDate` och applicerar dem som ett val på `OrderDate`. Om det finns många distinkta värden, fler än några hundra, blir denna åtgärd processorkrävande och bör därför undvikas.

Elementuppsättningsfunktioner

Elementuppsättningen kan även baseras på set-funktionerna `P()` (möjliga värden) och `E()` (uteslutna värden).

Om du till exempel vill välja länder där produkten `Cap` har sålts kan du använda:

```
<Country = P({1<Product={Cap}>} Country)>
```

På liknande sätt, om du vill välja ut de länder där produkten `Cap` inte har sålts, kan du använda:

```
<Country = E({1<Product={Cap}>} Country)>
```

Set-modifierare med sökningar

Du kan skapa uppsättningar av element genom sökningar med set-modifierare.

Exempel:

- `<Country = {"C*"}>`
- `<Year = {">2015"}>`
- `<Ingredient = {"*garlic*"}>`

Sökningar bör alltid vara inom dubbla citattecken, hakparenteser eller grava accenter. Du kan använda en blandning av stränglitteraler (enkla citattecken) och sökningar (dubbla citattecken). Exempel:

```
<Product = {'Nut', "*Bolt", Washer}>
```

Textsökningar

Jokertecken och andra symboler kan användas i textsökningar:

- En asterisk (*) kommer att motsvara valfritt antal tecken.
- Ett frågetecken (?) kommer att motsvara ett enda tecken.
- En grav accent (^) kommer att markera början på ett ord.

Exempel:

- `<Country = {"C*", "*land"}>`
  Matcha alla länder som börjar med ett `C` eller slutar med `land`.
- `<Country = {"*^z*"}>`
  Det kommer att matcha alla länder som har ett ord som börjar på `z`, som `New Zealand`.

Numeriska sökningar

Du kan göra numeriska sökningar med hjälp av dessa relationsoperatorer: >, >=, <, <=

En numerisk sökning börjar alltid med en av dessa operatorer. Exempel:

- `<Year = {">2015"}>`
  Matcha 2016 och efterföljande år.
- `<Date = {">=1/1/2015<1/1/2016"}>`
  Matcha alla datum under 2015. Notera syntaxen för att beskriva ett tidsintervall mellan de två datumen. Datumformatet behöver matcha datumformatet för fältet i fråga.

Sökningar med hjälp av uttryck

Du kan använda sökuttryck till att göra mer avancerade sökningar. En aggregering utvärderas sedan för varje fältvärde i sökfältet. Alla värden för vilka sökuttrycket returnerar sant väljs.

En uttryckssökning börjar alltid med ett likhetstecken: =

Exempel:

```
<Customer = {"=Sum(Sales)>1000"}>
```

Det returnerar alla kunder med ett försäljningsvärde som är större än 1 000. `Sum(Sales)` beräknas utifrån det aktuella urvalet. Det betyder att om du har ett urval i ett annat fält, till exempel fältet `Product`, får du endast kunder som uppfyllt försäljningsvillkoret för de valda produkterna som resultat.

Om du vill att villkoret ska vara oberoende av ditt urval behöver du använda set-analysen inne i söksträngen. Exempel:

```
<Customer = {"=Sum({1} Sales)>1000"}>
```

Uttrycken efter likhetstecknet kommer att tolkas som ett booleskt värde. Det betyder att om det utvärderas till något annat kommer alla tal utom noll att tolkas som sanna, samtidigt som noll och strängar tolkas som falska.

Quotes

Använd citattecken när söksträngar innehåller mellanslag eller specialtecken. Enkla citattecken implicerar en exakt, skiftlägeskänslig matchning med ett enda fältvärde. Dubbla citattecken implicerar en sökning som inte är skiftlägeskänslig och som potentiellt matchar flera fältvärden.

Exempel:

- `<Country = {'New Zealand'}>`
  Matcha endast `New Zealand`.
- `<Country = {"New Zealand"}>`
  Matcha `New Zealand`, `NEW ZEALAND` och `new zealand`.

Dubbla citattecken kan bytas ut av hakparenteser eller grava accenter.

I tidigare versioner av Qlik Sense hanterades enkla och dubbla citattecken på samma sätt, och alla strängar inom citattecken behandlades som sökningar. För att säkerställa bakåtkompatibiliteten kommer appar som skapats med äldre versioner av Qlik Sense att fungera på samma sätt som i tidigare versioner. Appar som skapats med Qlik Sense från november 2017 och framåt kommer att skilja mellan enkla och dubbla citattecken.

Exempel: Diagramuttryck för set-modifierare med sökningar

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
MyTable: Load  Year(Date) as Year, Date#(Date,'YYYY-MM-DD') as ISO_Date,  Date(Date#
(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,  Country, Product, Amount  Inline [Date, Country,
Product, Amount 2018-02-20, Canada, Washer, 6 2018-07-08, Germany, Anchor bolt, 10 2018-07-14,
Germany, Anchor bolt, 3 2018-08-31, France, Nut, 2 2018-09-02, Czech Republic, Bolt, 1 2019-
02-11, Czech Republic, Bolt, 3 2019-07-31, Czech Republic, Washer, 6 2020-03-13, France,
Anchor bolt, 1 2020-07-12, Canada, Anchor bolt, 8 2020-09-16, France, Washer, 1];
```

Exempel 1: Diagramuttryck med textsökningar

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

Tabell – set-modifierare med textsökningar

Land	Sum (Amount)	Sum({<Country={"C*"}>} Amount)	Sum({<Country={"*^R*"}>} Amount)	Sum({<Product={"*bolt*"}>} Amount)
Totalvärden	**41**	**24**	**10**	**26**
Kanada	14	14	0	8
Tjeckien	10	10	10	4
Frankrike	4	0	0	1
Germany	13	0	0	13

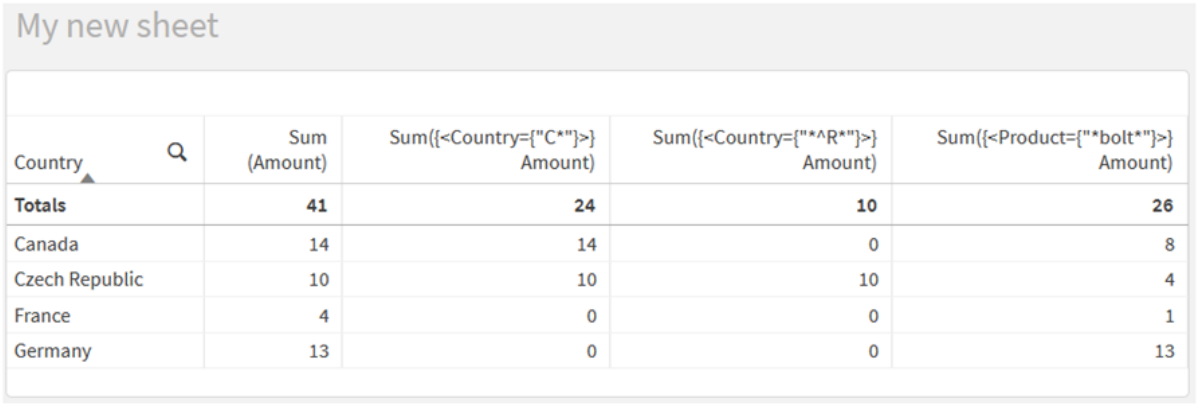
Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Amount)`
    Summa `Amount` utan set-uttryck.
  - `Sum({<Country={"C*"}>}Amount)`

Summan `Amount` för alla länder som börjar på `C`, till exempel `Canada` och `Czech Republic`.

- `Sum({<Country={"*^R*"}>}Amount)`
  Summan `Amount` för alla länder som har ett ord som börjar på `R`, som `Czech Republic`.
- `Sum({<Product={"*bolt*"}>}Amount)`
  Summan `Amount` för alla produkter som innehåller strängen `bolt`, som `Bolt` och `Anchor bolt`.

Set-modifierare med textsökningar



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

Exempel 2: Diagramuttryck med numeriska sökningar

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

Tabell - set-modifierare med numeriska sökningar

Land	Sum (Amount)	Sum({<Year={">2019"}>} Amount)	Sum({<ISO_Date={">=2019-07-01"}>} Amount)	Sum({<US_Date={">=4/1/2018<=12/31/2018"}>} Amount)
Totalvärden	**41**	**10**	**16**	**16**
Kanada	14	8	8	0
Tjeckien	10	0	6	1
Frankrike	4	2	2	2
Germany	13	0	0	13

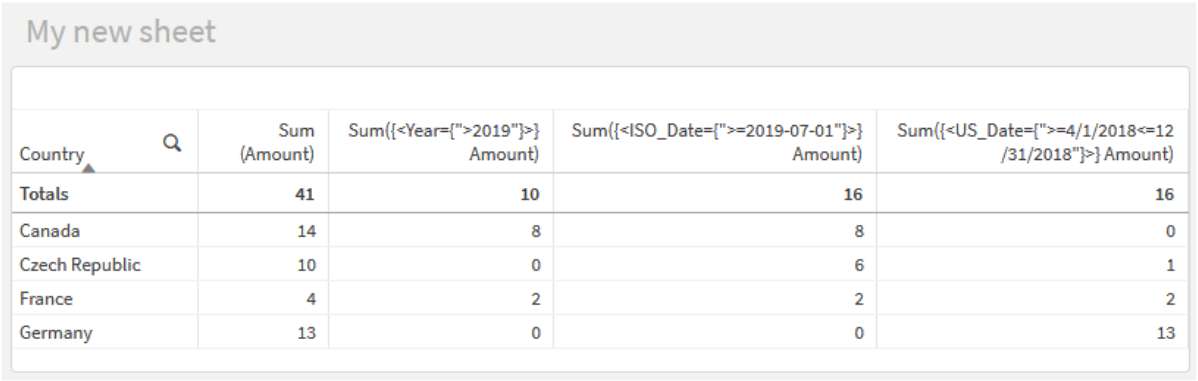
Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Amount)`
    Summa `Amount` utan set-uttryck.
  - `Sum({<Year={">2019"}>}Amount)`

Summan `Amount` för alla år efter `2019`.

- `Sum({<ISO_Date={">=2019-07-01"}>}Amount)`
  Summan `Amount` för alla datum från och med `2019-07-01`. Formatet på datumet i sökningen måste matcha formatet i fältet.
- `Sum({<US_Date={">=4/1/2018<=12/31/2018"}>}Amount)`
  Summan `Amount` för alla datum från `4/1/2018` till `12/31/2018`, inklusive start- och slutdatum. Formatet på datumen i sökningen måste matcha formatet i fältet.

Set-modifierare med numeriska sökningar



Exempel 3: Diagramuttryck med uttryckssökningar

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

Table - Set modifiers with expression searches

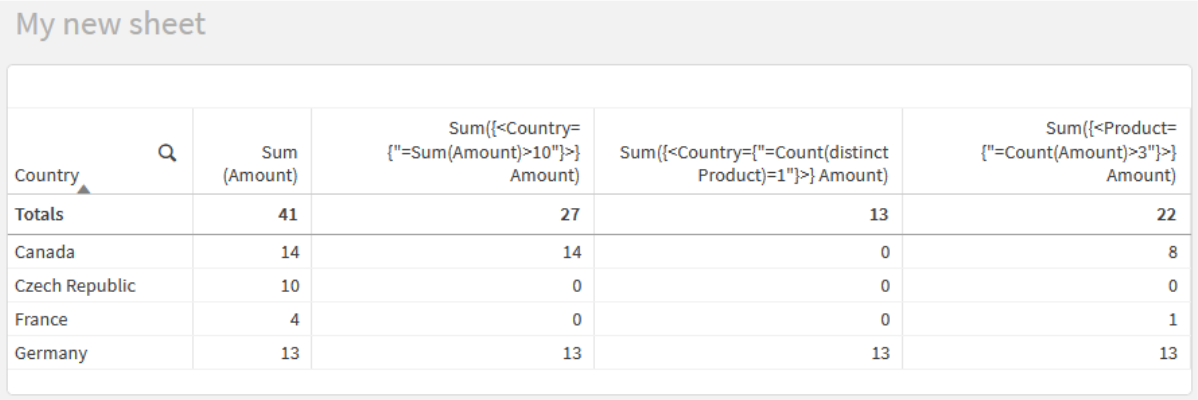
Country	Sum (Amount)	Sum({<Country={"=Sum (Amount)>10"}>} Amount)	Sum({<Country={"=Count(distinct Product)=1"}>} Amount)	Sum({<Product={"=Count (Amount)>3"}>} Amount)
Totals	**41**	**27**	**13**	**22**
Canada	14	14	0	8
Czech Republic	10	0	0	0
France	4	0	0	1
Germany	13	13	13	13

Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Amount)`
    Summa `Amount` utan set-uttryck.

- `Sum({<Country={"=Sum(Amount)>10"}>}Amount)`
  Summan `Amount` för alla länder som har en aggregerad summa på `Amount` över `10`.
- `Sum({<Country={"=Count(distinct Product)=1"}>}Amount)`
  Summan `Amount` för alla länder som är kopplade till exakt en distinkt produkt.
- `Sum({<Product={"=Count(Amount)>3"}>}Amount)`
  Summan `Amount` för alla länder som har över tre transaktioner i data.

Set-modifierare med uttryckssökningar



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

Set-modifierare med dollarteckenexpansioner

Dollarteckenexpansioner är konstruktioner som beräknas innan uttrycket tolkats och utvärderats. Resultatet matas sedan in i uttrycket istället för `$(…)`. Beräkningen av uttrycket görs sedan med hjälp av resultatet av dollarexpansionen.

Uttrycksredigeraren visar en förhandsgranskning av dollarexpansionen så att du kan bekräfta vad din dollarteckenexpansion utvärderas till.

Förhandsgranskning med dollarteckensexpansion i uttrycksredigeraren

Använd dollarteckensexpansioner när du vill använda en beräkning i din elementuppsättning.

Om du till exempel endast vill titta på det sista möjliga året kan du använda följande konstruktion:

```
<Year = {$(=Max(Year))}>
```

`Max(Year)` beräknas först och resultatet matas in i uttrycket istället för `$(…)`.

Resultatet efter dollarexpansionen kommer att vara ett uttryck som detta:

```
<Year = {2021}>
```

Uttrycket i dollarteckenexpansionen beräknas baserat på det aktuella urvalet. Det betyder att om du har ett urval i ett annat fält kommer resultatet av uttrycket att påverkas.

Om du vill att beräkningen ska vara fristående från urvalet kan du använda set-analysen i dollarexpansionen. Exempel:

```
<Year = {$(=Max({1} Year))}>
```

Strings

När du vill att dollarexpansionen ska resultera i en sträng gäller normala citatregler. Exempel:

```
<Country = {'$(=FirstSortedValue(Country,Date)'}>
```

Resultatet efter dollarexpansionen kommer att vara ett uttryck som detta:

```
<Country = {'New Zealand'}>
```

Du kommer att få ett syntaxfel om du inte använder citattecken.

Numbers

När du vill att dollarexpansionen ska resultera i en siffra ska du se till att expansionen får samma formatering som fältet. Det betyder att du ibland behöver radbryta uttrycket i en formateringsfunktion.

Exempel:

```
<Amount = {$(=Num(Max(Amount), '###0.00'))}>
```

Resultatet efter dollarexpansionen kommer att vara ett uttryck som detta:

```
<Amount = {12362.00}>
```

Använd en hash för att tvinga expansionen att alltid använda decimalkomma och ingen tusentalsavgränsare. Exempel:

```
<Amount = {$(#=Max(Amount))}>
```

Datum

Om du vill att dollarexpansionen ska resultera i ett datum ska du se till att expansionen har korrekt formatering. Det betyder att du ibland behöver radbryta uttrycket i en formateringsfunktion.

Exempel:

```
<Date = {'$(=Date(Max(Date)))'}>
```

Resultatet efter dollarexpansionen kommer att vara ett uttryck som detta:

```
<Date = {'12/31/2015'}>
```

Precis som för strängar behöver du använda korrekta citattecken.

Ett vanligt exempel kan vara när du vill att beräkningen ska vara begränsad till den senaste månaden (eller året). Då kan du använda en numerisk sökning i kombination med funktionen `AddMonths()`.

Exempel:

```
<Date = {">=$(=AddMonths(Today(),-1))"}>
```

Resultatet efter dollarexpansionen kommer att vara ett uttryck som detta:

```
<Date = {">=9/31/2021"}>
```

Det kommer att välja ut alla händelser som inträffat under den senaste månaden.

Exempel: Diagramuttryck för set-modifierare med dollarteckenexpansioner

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
Let vToday = Today(); MyTable: Load  Year(Date) as Year, Date#(Date,'YYYY-MM-DD') as ISO_Date,
 Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,  Country, Product, Amount  Inline
[Date, Country, Product, Amount 2018-02-20, Canada, Washer, 6 2018-07-08, Germany, Anchor
bolt, 10 2018-07-14, Germany, Anchor bolt, 3 2018-08-31, France, Nut, 2 2018-09-02, Czech
Republic, Bolt, 1 2019-02-11, Czech Republic, Bolt, 3 2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1 2020-07-12, Canada, Anchor bolt, 8 2021-10-15, France,
Washer, 1];
```

Diagramuttryck med dollarteckenexpansioner

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

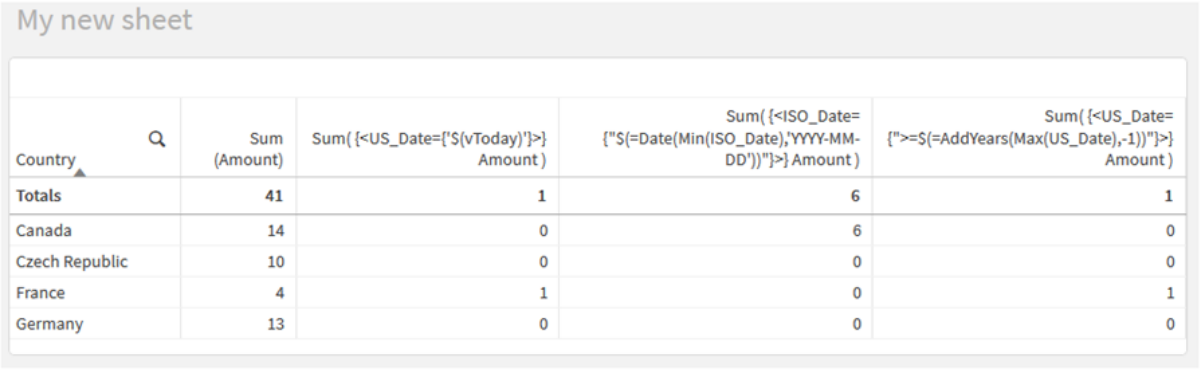
Tabell – set-modifierare med dollarteckenexpansioner

Land	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)
Totalvärden	**41**	**1**	**6**	**1**
Kanada	14	0	6	0
Tjeckien	10	0	0	0
Frankrike	4	1	0	1
Germany	13	0	0	0

Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Amount)`
    Summa `Amount` utan set-uttryck.
  - `Sum({<US_Date={'$(vToday)'}>}Amount)`
    Summan `Amount` för alla poster där `US_Date` är samma som i variabeln `vToday`.
  - `Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>}Amount)`
    Summan `Amount` för alla poster där `ISO_Date` är samma som den första (minsta) möjliga `ISO_Date`. Funktionen `Date()` behövs för att se till att formatet för datumet matchar fältets.
  - `Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>}Amount)`
    Summan `Amount` för alla poster som har ett `US_Date` som är lika med eller kommer efter datumet ett år före det senaste (högsta) möjliga `US_Date`. Funktionen `AddYears()` kommer att returnera ett datum i formatet som anges av variabeln `DateFormat` och det behöver matcha formatet för fältet `US_Date`.

Set-modifierare med dollarteckenexpansioner


My new sheet

Country	Sum (Amount)	Sum({<US_Date={'$(vToday)'}>} Amount)	Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>} Amount)	Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>} Amount)
Totals	**41**	**1**	**6**	**1**
Canada	14	0	6	0
Czech Republic	10	0	0	0
France	4	1	0	1
Germany	13	0	0	0

Set-modifierare med set-operatorer

Set-operatorer används till att inkludera, utesluta eller genomskära olika elementuppsättningar. De kombinerar de olika metoderna för att definiera elementuppsättningar.

Operatorer är samma som de som används för set-identifierare.

Operatorer

Operator	Beskrivning
+	Union. Denna binära operation returnerar en uppsättning som består av de poster eller element som tillhör någon av de två set-operanderna.

Operator	Beskrivning
-	Exklusion. Denna binära operation returnerar en uppsättning som består av de poster eller element som enbart tillhör den första men inte den andra av de två set-operanderna. Den returnerar dessutom det komplementära setet när det används som en unär operator.
*	Intersektion. Denna binära operation returnerar en uppsättning som består av de poster eller element som tillhör båda set-operanderna.
/	Symmetrisk skillnad (XOR). Denna binära operation returnerar en uppsättning som består av de poster eller element som tillhör endera men inte båda set-operanderna.

Till exempel kan de följande två modifierarna definiera samma uppsättning med fältvärden:

- `<Year = {1997, "20*"}>`
- `<Year = {1997} + {"20*"}>`

Båda uttrycken väljer `1997` och åren som börjar med `20`. Med andra ord är det här unionen av de två villkoren.

Set-operatorer möjliggör även mer komplicerade definitioner. Exempel:

```
<Year = {1997, "20*"} - {2000}>
```

Det här uttrycket kommer att välja samma år som de ovan, men dessutom utesluta år `2000`.

.

Exempel: Diagramuttryck för set-modifierare med set-operatorer

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
MyTable: Load  Year(Date) as Year, Date#(Date,'YYYY-MM-DD') as ISO_Date,  Date(Date#
(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,  Country, Product, Amount  Inline [Date, Country,
Product, Amount 2018-02-20, Canada, Washer, 6 2018-07-08, Germany, Anchor bolt, 10 2018-07-14,
Germany, Anchor bolt, 3 2018-08-31, France, Nut, 2 2018-09-02, Czech Republic, Bolt, 1 2019-
02-11, Czech Republic, Bolt, 3 2019-07-31, Czech Republic, Washer, 6 2020-03-13, France,
Anchor bolt, 1 2020-07-12, Canada, Anchor bolt, 8 2020-09-16, France, Washer, 1];
```

Diagramuttryck

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

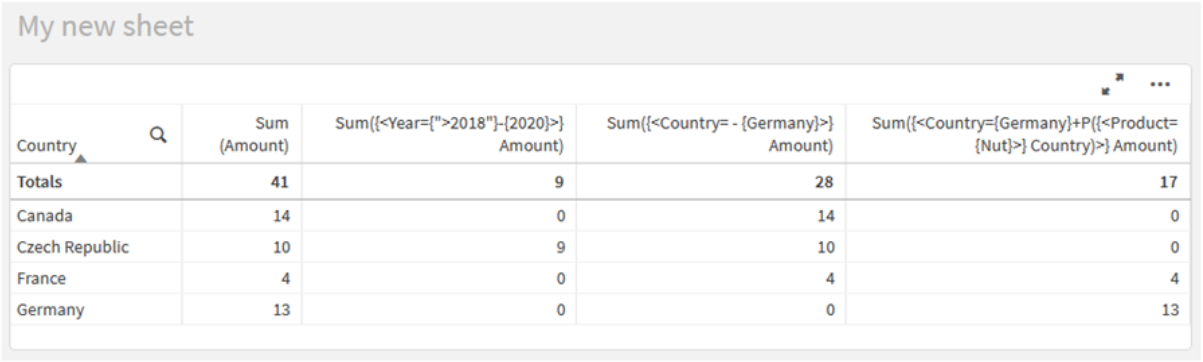
Tabell - set-modifierare med set-operatorer

Land	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum ({<Country=-{Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>}Country)>} Amount)
Totalvärden	**41**	**9**	**28**	**17**
Kanada	14	0	14	0
Tjeckien	10	9	10	0
Frankrike	4	0	4	4
Germany	13	0	0	13

Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Amount)`
    Summa `Amount` utan set-uttryck.
  - `Sum({<Year={">2018"}-{2020}>}Amount)`
    Summan `Amount` för alla år efter `2018`, utom `2020`.
  - `Sum({<Country=-{Germany}>}Amount)`
    Summan `Amount` för alla länder utom `Germany`. Observera den unära exklusionsoperatorn.
  - `Sum({<Country={Germany}+P({<Product={Nut}>}Country)>}Amount)`
    Summan `Amount` för `Germany` och alla länder som är kopplade till produkten `Nut`.

Set-modifierare med set-operatorer



Set-modifierare med implicita set-operatorer

Standardmetoden för att skriva urval i en set-modifierare är att använda likhetstecken. Exempel:

```
Year = {">2015"}
```

Uttrycket till höger om likhetstecknet i set-modifieraren kallas elementuppsättningen. Det definierar en uppsättning med distinkta fältvärden, med andra ord ett urval.

Denna notation definierar ett nytt urval, utan hänsyn till det aktuella urvalet i fältet. Så om set-identifieraren innehåller ett urval i det här fältet kommer det gamla urvalet att ersättas av det i elementuppsättningen.

När du vill basera ditt urval på det aktuella urvalet i fältet ska du använda ett annat uttryck

Om du till exempel vill fortsätta att använda det gamla urvalet men lägga till kravet att året ska vara efter 2015, kan du skriva följande:

```
Year = Year * {">2015"}
```

Asterisken är en set-operator som definierar en intersektion, så du kommer att få intersektionen mellan det aktuella urvalet i `Year` och det ytterligare kravet att året är efter `2015`. Ett alternativt sätt att skriva det här på är följande:

```
Year *= {">2015"}
```

Det innebär att tilldelningsoperatorn (`*=`) implicit definierar en intersektion.

På ett liknande sätt kan implicita unioner, uteslutningar och symmetriska skillnader definieras med följande: `+=`, `-=`, `/=`

Exempel: Diagramuttryck för set-modifierare med implicita set-operatorer

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
 MyTable: Load  Year(Date) as Year, Date#(Date,'YYYY-MM-DD') as ISO_Date,  Date(Date#
(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,  Country, Product, Amount  Inline [Date, Country,
Product, Amount 2018-02-20, Canada, Washer, 6 2018-07-08, Germany, Anchor bolt, 10 2018-07-14,
Germany, Anchor bolt, 3 2018-08-31, France, Nut, 2 2018-09-02, Czech Republic, Bolt, 1 2019-
02-11, Czech Republic, Bolt, 3 2019-07-31, Czech Republic, Washer, 6 2020-03-13, France,
Anchor bolt, 1 2020-07-12, Canada, Anchor bolt, 8 2020-09-16, France, Washer, 1];
```

Diagramuttryck med implicita set-operatorer

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

Välj `Canada` och `Czech Republic` från en lista över länder.

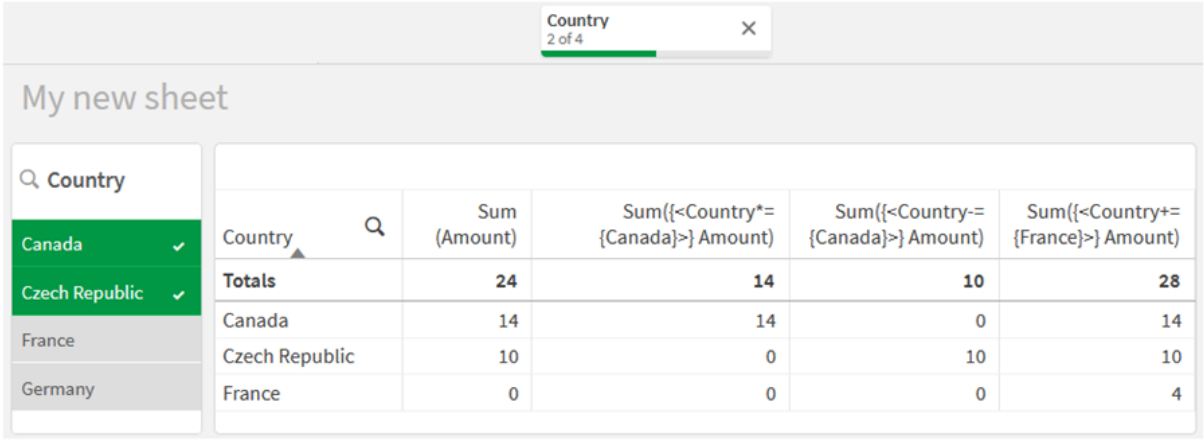
Tabell - Diagramuttryck med implicita set-operatorer

Land	Sum (Amount)	Sum({<Country*= {Canada}>} Amount)	Sum({<Country-= {Canada}>} Amount)	Sum({<Country+= {France}>} Amount)
Totalvärden	**24**	**14**	**10**	**28**
Kanada	14	14	0	14
Tjeckien	10	0	10	10
Frankrike	0	0	0	4

Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Amount)`
    Summan `Amount` för det aktuella urvalet. Observera att endast `Canada` och `Czech Republic` har värden som inte är noll.
  - `Sum({<Country*={Canada}>}Amount)`
    Summan `Amount` för det aktuella urvalet, som genomskärs av kravet att `Country` ska vara `Canada`. Om `Canada` inte ingår i användarens urval returnerar set-uttrycket en tom uppsättning och kolumnen kommer att ha `0` på alla rader.
  - `Sum({<Country-={Canada}>}Amount)`
    Summan `Amount` för det aktuella urvalet, men uteslut först `Canada` från urvalet `Country`. Om `Canada` inte ingår i användarens urval kommer set-uttrycket inte att ändra några siffror.
  - `Sum({<Country+={France}>}Amount)`
    Summan `Amount` för det aktuella urvalet, men lägg först till `France` i urvalet `Country`. Om `France` redan ingår i användarens urval kommer set-uttrycket inte att ändra några siffror.

Set-modifierare med implicta set-operatorer



Set-modifierare som använder set-funktioner

Ibland behöver du definiera en uppsättning fältvärden med hjälp av en nästlad set-definition. Till exempel kanske du vill välja alla kunder som har köpt en viss produkt, utan att välja produkten.

I sådana fall använder du elementuppsättningsfunktionerna `P()` och `E()`. Dessa returnerar elementuppsättningar med möjliga värden respektive uteslutna värden för ett fält. Inom parenteserna kan du ange fältet i fråga och ett set-uttryck som definierar omfattningen. Exempel:

```
P({1<Year = {2021}>} Customer)
```

Det returnerar uppsättningen med kunder som har haft transaktioner under 2021. Du kan använda det här i en set-modifierare. Exempel:

```
Sum({<Customer = P({1<Year = {2021}>} Customer)>} Amount)
```

Det här set-uttrycket kommer att välja dessa kunder men kommer inte att begränsa urvalet till 2021.

Dessa funktioner kan inte användas i andra uttryck.

Dessutom kan endast naturliga uppsättningar användas i elementuppsättningsfunktionerna. Det vill säga, en uppsättning med poster som definieras genom ett enkelt urval.

Till exempel kan den uppsättning som anges med `{1-$}` inte alltid definieras genom urval och det är därför inte en naturlig uppsättning. Användning av dessa funktioner med icke-naturliga uppsättningar returnerar oväntade resultat.

Exempel: Diagramuttryck för set-modifierare som använder set-funktioner

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
MyTable: Load  Year(Date) as Year, Date#(Date,'YYYY-MM-DD') as ISO_Date,  Date(Date#
(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,  Country, Product, Amount  Inline [Date, Country,
Product, Amount 2018-02-20, Canada, Washer, 6 2018-07-08, Germany, Anchor bolt, 10 2018-07-14,
Germany, Anchor bolt, 3 2018-08-31, France, Nut, 2 2018-09-02, Czech Republic, Bolt, 1 2019-
02-11, Czech Republic, Bolt, 3 2019-07-31, Czech Republic, Washer, 6 2020-03-13, France,
Anchor bolt, 1 2020-07-12, Canada, Anchor bolt, 8 2020-09-16, France, Washer, 1];
```

Diagramuttryck

Skapa en tabell i ett Qlik Sense-ark med följande diagramuttryck.

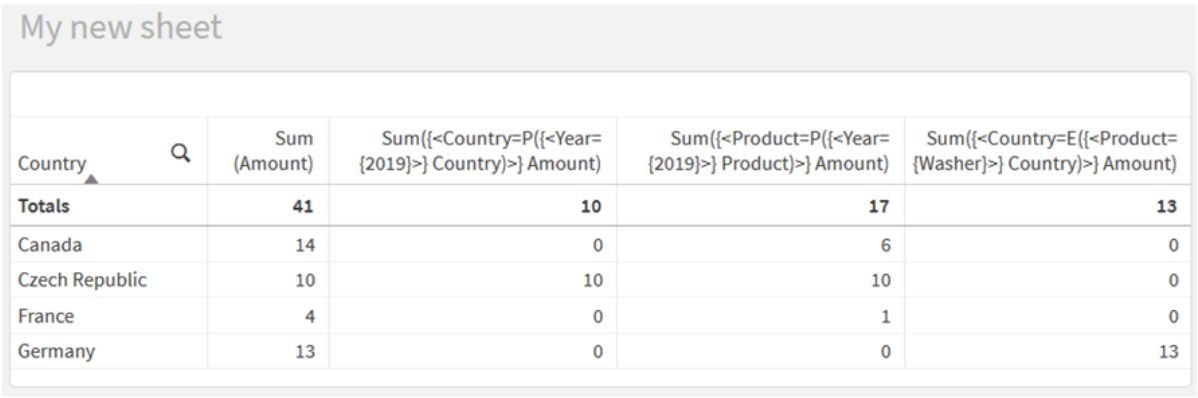
Tabell - Set-modifierare som använder set-funktioner

Land	Sum (Amount)	Sum({<Country=P({<Year={2019}>}Country)>} Amount)	Sum({<Product=P({<Year={2019}>} Product)>} Amount)	Sum({<Country=E({<Product={Washer}>}Country)>} Amount)
Totalvärden	**41**	**10**	**17**	**13**
Kanada	14	0	6	0
Tjeckien	10	10	10	0
Frankrike	4	0	1	0
Germany	13	0	0	13

Förklaring

- Dimensioner:
  - `Country`
- Mått:
  - `Sum(Amount)`
    Summa `Amount` utan set-uttryck.
  - `Sum({<Country=P({<Year={2019}>} Country)>} Amount)`
    Summan `Amount` för länderna som är kopplade till året `2019`. Det kommer dock inte att begränsa beräkningen till `2019`.
  - `Sum({<Product=P({<Year={2019}>} Product)>} Amount)`
    Summan `Amount` för produkterna som är kopplade till året `2019`. Det kommer dock inte att begränsa beräkningen till `2019`.
  - `Sum({<Country=E({<Product={Washer}>} Country)>} Amount)`
    Summan `Amount` för länderna som inte är kopplade till produkten `Washer`.

Set-modifierare som använder set-funktioner



Introduktion – Skapa ett set-uttryck

Du kan bygga set-uttryck som ger stöd till dataanalys. I det här sammanhanget kallas analysen ofta för set-analys. Set-analys gör det möjligt att definiera en omfattning som är olik den uppsättning med poster som definieras av det aktuella urvalet i en app.

Det här får du lära dig

Denna introduktion ger data och diagramuttryck för att bygga set-uttryck med hjälp av set-modifierare, identifierare och operatorer.

Vem är den här introduktionskursen avsedd för?

Den här introduktionskursen är avsedd för apputvecklare som är vana att arbeta med Skriptredigeraren och diagramuttryck.

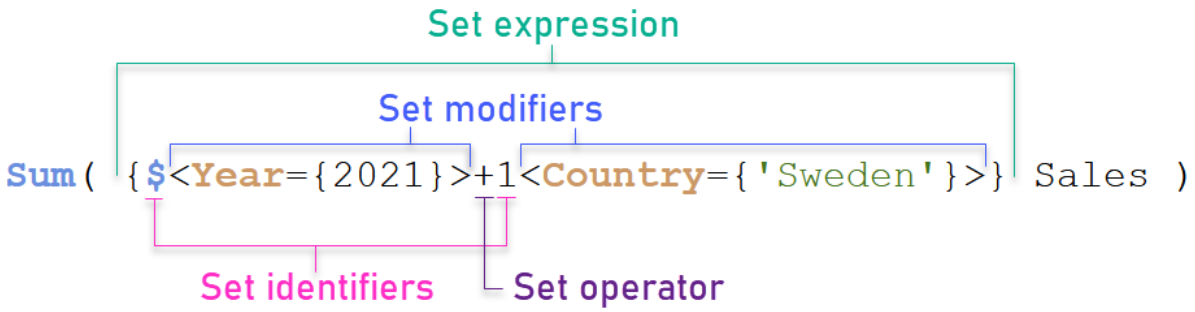
Det här behövs innan du kan börja

En allokerad Qlik Sense Enterprise Professional-åtkomst, som tillåter att du kan ladda data och skapa appar.

Element i ett set-uttryck

Set-uttryck omsluts av en aggregeringsfunktion, som `Sum()`, `Max()`, `Min()`, `Avg()` eller `Count()`. Set-uttryck består av byggstenar som kallas för element. Dessa element är set-modifierare, -identifierare och -operatorer.

Element i ett set-uttryck



Set-uttrycket ovan är till exempel byggt från aggregeringen `Sum(Sales)`. Set-uttrycket är omslutet av de yttre klammerparenteserna: `{ }`

Den första operanden i uttrycket är: `$<Year={2021}>`

Denna operand returnerar försäljningen under året `2021` för det aktuella urvalet. Modifieraren, `<Year={2021}>`, innehåller urvalet av år, d.v.s. 2021. Set-identifieraren `$` indikerar att set-uttrycket är baserat på det aktuella urvalet.

Den andra operanden i uttrycket är: `1<Country={'Sweden'}>`

Operanden returnerar Sales för Sweden. Modifieraren, `<Country={'Sweden'}>`, innehåller urvalet av land, d.v.s. Sweden. Set-identifieraren `1` indikerar att urval som görs i appen kommer att ignoreras.

Slutligen indikerar set-operatorn `+` att uttrycket returnerar en uppsättning som består av posterna som tillhör någon av de två set-operanderna.

Introduktionskursen Skapa ett set-uttryck

Slutför följande procedurer för att skapa set-uttrycken som visas i den här introduktionskursen.

Skapa en ny app och ladda data

Gör följande:

1. Skapa en ny app.
2. Klicka på **Skriptredigeraren**. Du kan även klicka på **Förbered** > **Skriptredigeraren** i navigeringsfältet.
3. Skapa ett nytt delavsnitt i **Skriptredigeraren**.
4. Kopiera följande data och klistra in dem i det nya delavsnittet: *Introduktionsdata för set-uttryck (page 198)*
5. Klicka på **Ladda data**. Data laddas som en inline load.

Skapa set-uttryck med modifierare

Set-modifieraren består av ett eller flera fältnamn, som vart och ett följs av ett urval som ska göras på just det fältet. Modifieraren omsluts av vinkelparenteser. Ett exempel är detta set-uttryck:

```
Sum ( {<Year = {2015}>} Sales )
```

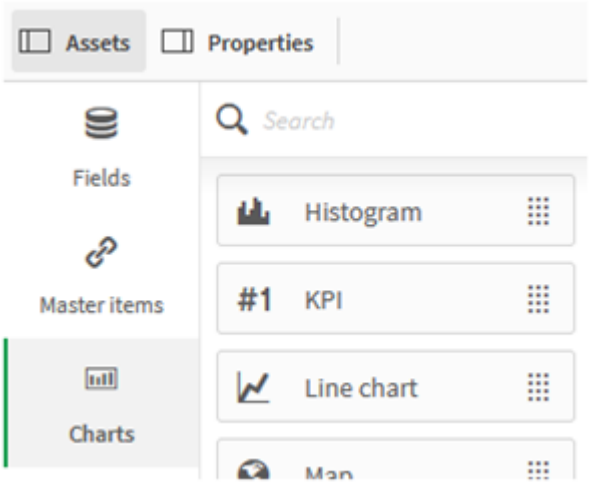
Modifieraren är:

```
<Year = {2015}>
```

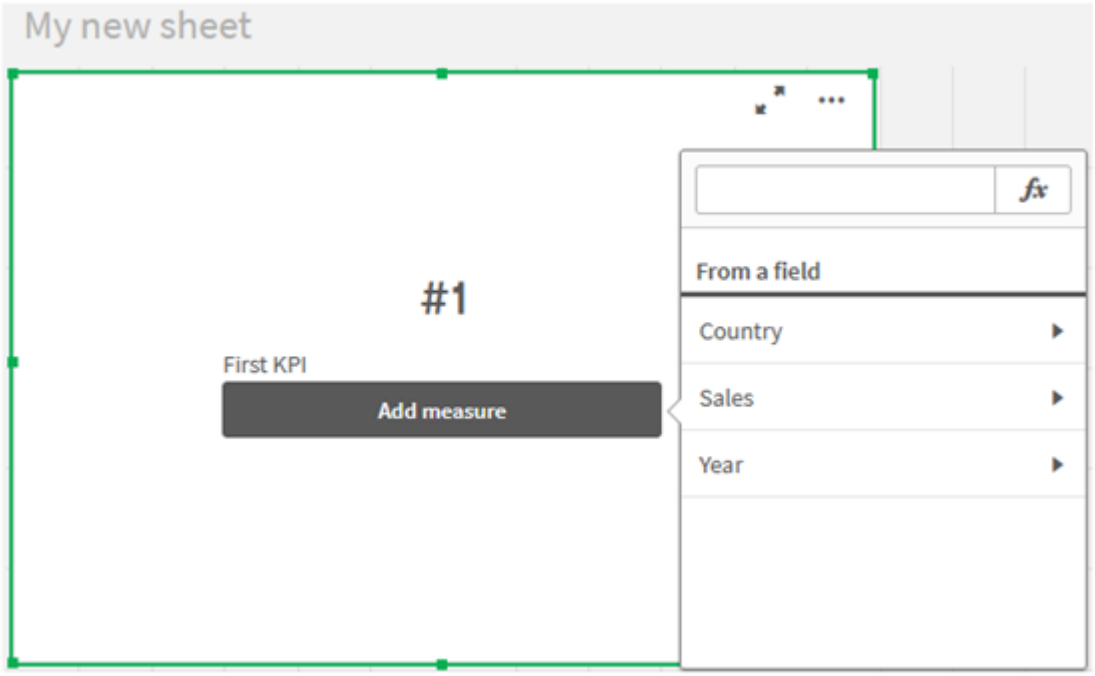
Den här modifieraren anger att data från år 2015 kommer att väljas. Klammerparenteserna som omsluter modifieraren indikerar att det är ett set-uttryck.

Gör följande:

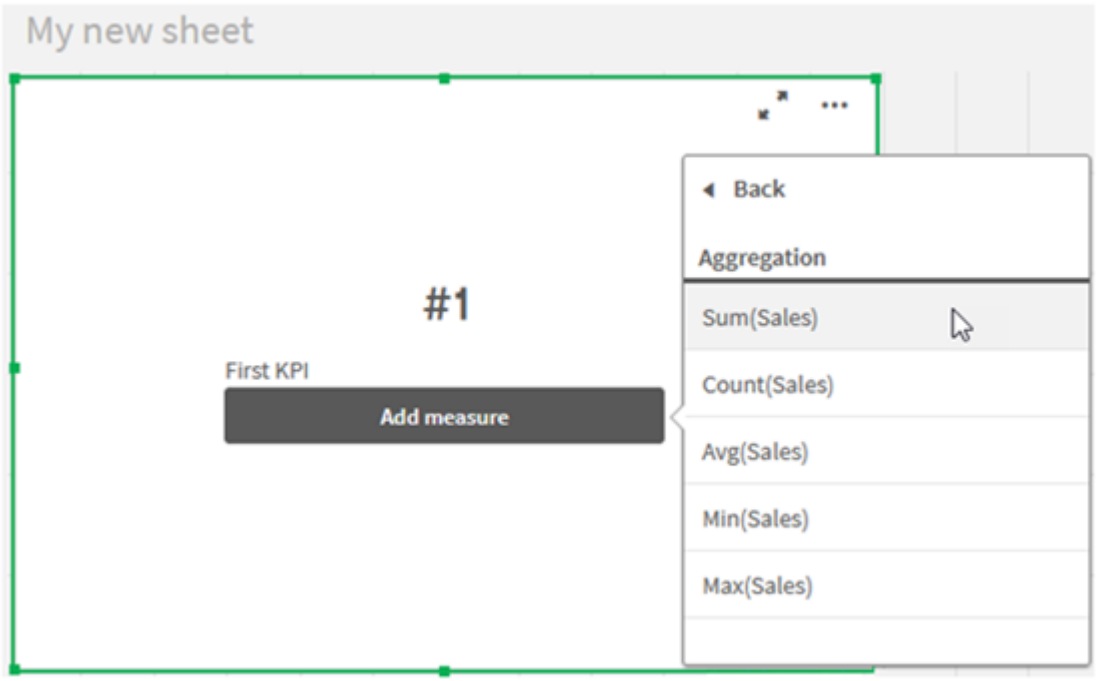
1. I ett ark öppnar du panelen **Resurser** från navigeringsfältet och klickar sedan på **Diagram**.



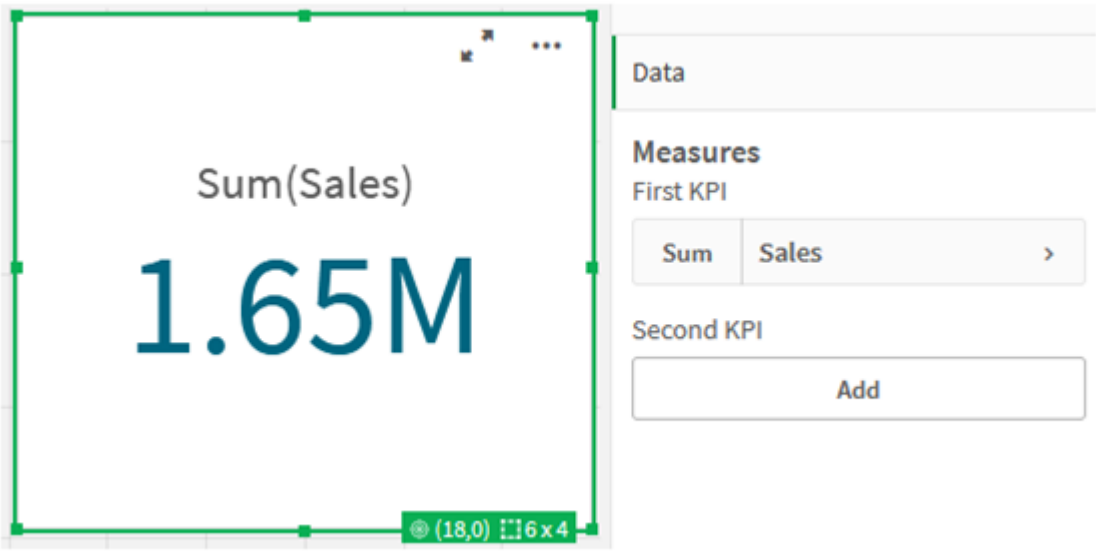
2. Dra ett **KPI** till arket och klicka sedan på **Lägg till mått**.



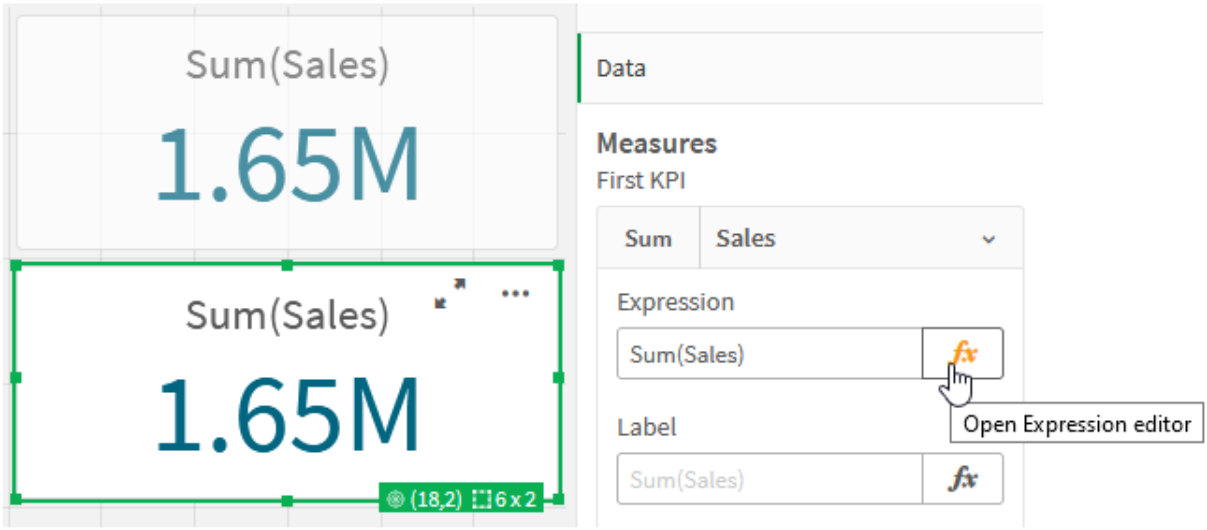
3. Klicka på `Sales` och välj sedan `Sum(Sales)` för aggregeringen.



KPI visar summan av försäljningen för alla år.



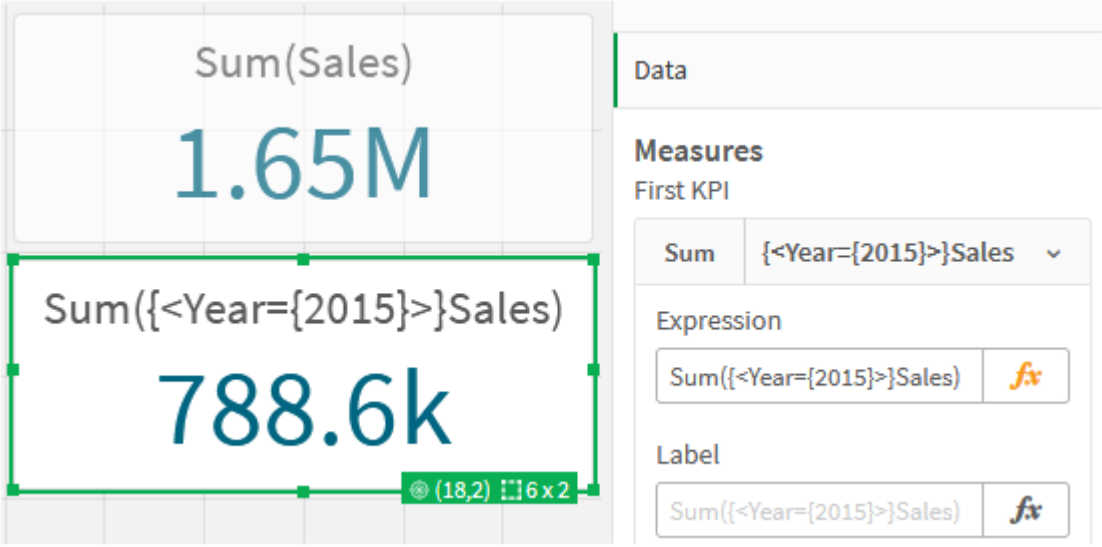
4. Kopiera och klistra in KPI för att skapa en ny KPI.
5. Klicka på det nya KPI, klicka på **Sales** under **Mått** och sedan på **Öppna uttrycksredigeraren**.



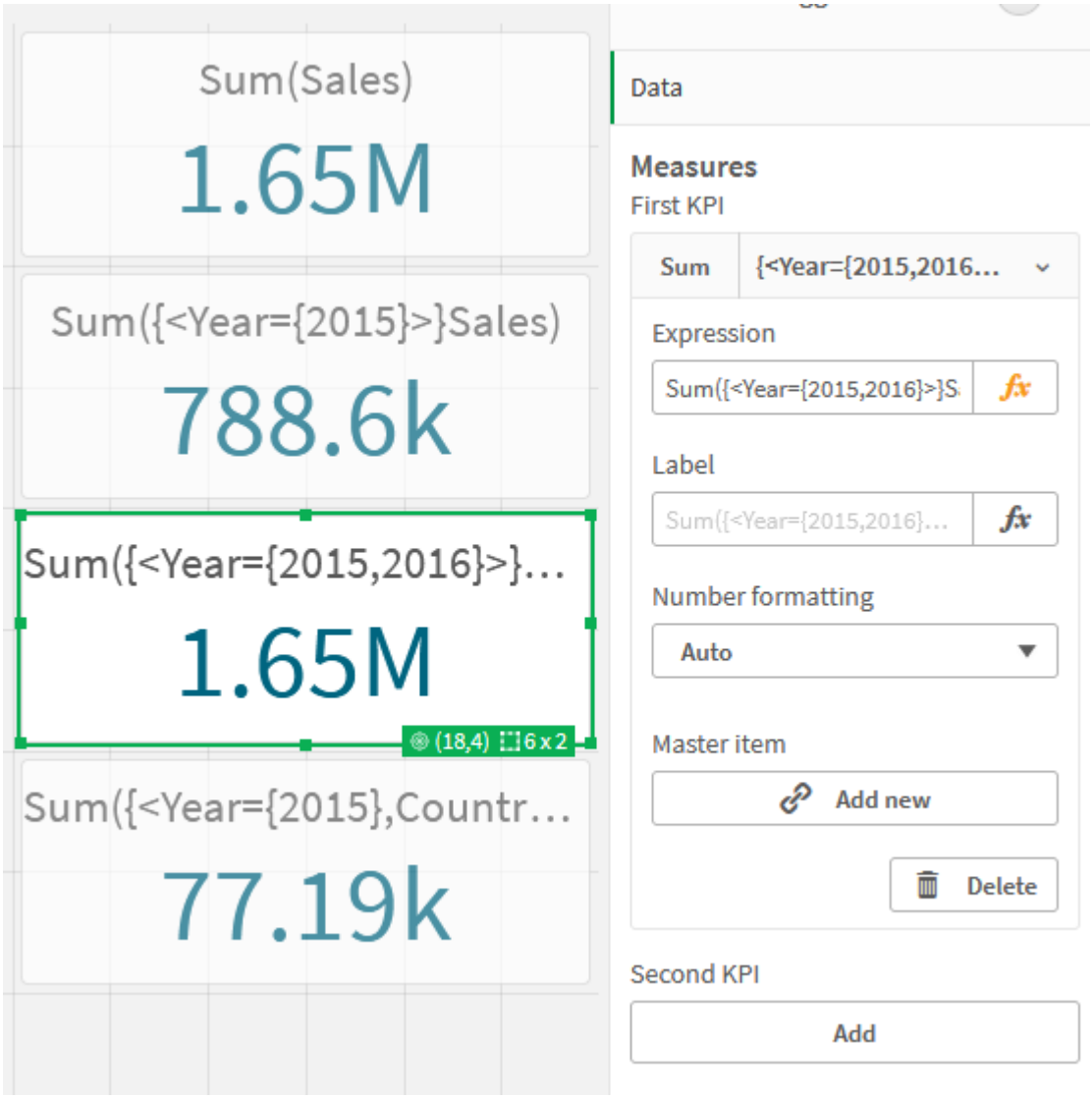
Uttrycksredigeraren öppnas med aggregeringen `Sum(Sales)`.

6. Skapa ett uttryck i uttrycksredigeraren för att få summan av Sales för endast 2015:
   i. Lägg till klammerparenteser för att indikera ett set-uttryck: `Sum({}Sales)`
   i. Lägg till vinkelparenteser för att indikera en set-modifierare: `Sum({<>}Sales)`
   ii. I vinkelparentesen lägger du till fältet som ska väljas, i det här fallet är fältet `Year`, följt av ett likhetstecken. Sedan ska du omge 2015 med nya klammerparenteser. Det ger följande set-modifierare som resultat: `{<Year={2015}>}`.
   Hela uttrycket är:
   `Sum({<Year={2015}>}Sales)`

iii. Klicka på **Använd** för att spara uttrycket och avsluta uttrycksredigeraren. Summan av Sales för 2015 visas i KPI.



7. Skapa två KPI:er till med följande uttryck:
   `Sum({<Year={2015,2016}>}Sales)`
   Modifieraren ovan är `<Year={2015,2016}>`. Uttrycket kommer att returnera summan av Sales för 2015 och 2016.
   `Sum({<Year={2015},Country={'Germany'}>} Sales)`
   Modifieraren ovan är `<Year={2015}, Country={'Germany'}>`. Uttrycket kommer att returnera summan Sales för 2015, där 2015 skär Germany.
   KPI:er som använder set-modifierare



Lägga till set-identifierare

Set-uttrycken ovan använder de aktuella urvalen som bas, eftersom en identifierare inte användes. Därefter lägger du till identifierare för att specificera beteendet när urval görs.

Gör följande:

På ditt ark ska du bygga eller kopiera följande set-uttryck:

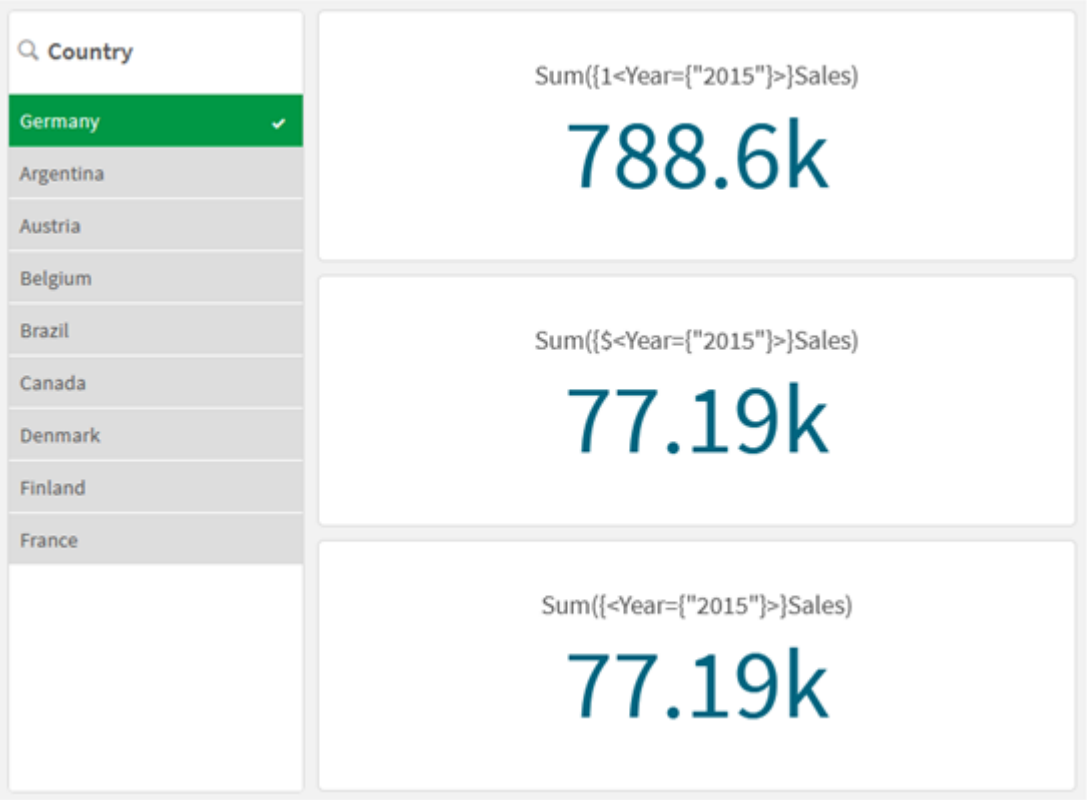
```
Sum({$<Year={"2015"}>}Sales
```

`$`-identifieraren baserar set-uttrycket på de aktuella urvalen som har gjorts i data. Det här är även standardbeteendet när en identifierare inte används.

```
Sum({1<Year={"2015"}>}Sales)
```

Identifieraren `1` kommer att orsaka att aggregeringen `Sum(Sales)` på 2015 ignorerar det aktuella urvalet. Värdet för aggregeringen kommer inte att förändras när användaren gör andra urval. Till exempel när Germany väljs nedan ändras inte värdet för den aggregerade summan för 2015.

KPI:er som använder set-modifierare och set-identifierare



Lägga till operatorer

Set-operatorer används till att inkludera, utesluta eller genomskära datauppsättningar. Alla operatorer använder uppsättningar som operander och returnerar en uppsättning som resultat.

Du kan använda set-operatorer i två olika situationer:

- För att utföra en set-åtgärd på set-identifierare, som motsvarar uppsättningar med poster i data.
- För att utföra en set-åtgärd på elementuppsättningar, på fältvärden eller i en set-modifierare.

Gör följande:

På ditt ark ska du bygga eller kopiera följande set-uttryck:

```
Sum({$<Year={2015}>+1<Country={'Germany'}>}Sales)
```

Operatorn med plustecknet `(+)` ger en union av datauppsättningarna för 2015 och Germany. Som vi förklarade för set-identifierarna ovan, innebär identifieraren med dollartecknet `($)` att de aktuella urvalen kommer att användas för den första operanden, `<Year={2015}>`, som respekteras. Identifieraren `1` innebär att urvalet kommer att ignoreras för den andra operanden, `<Country={'Germany'}>`.

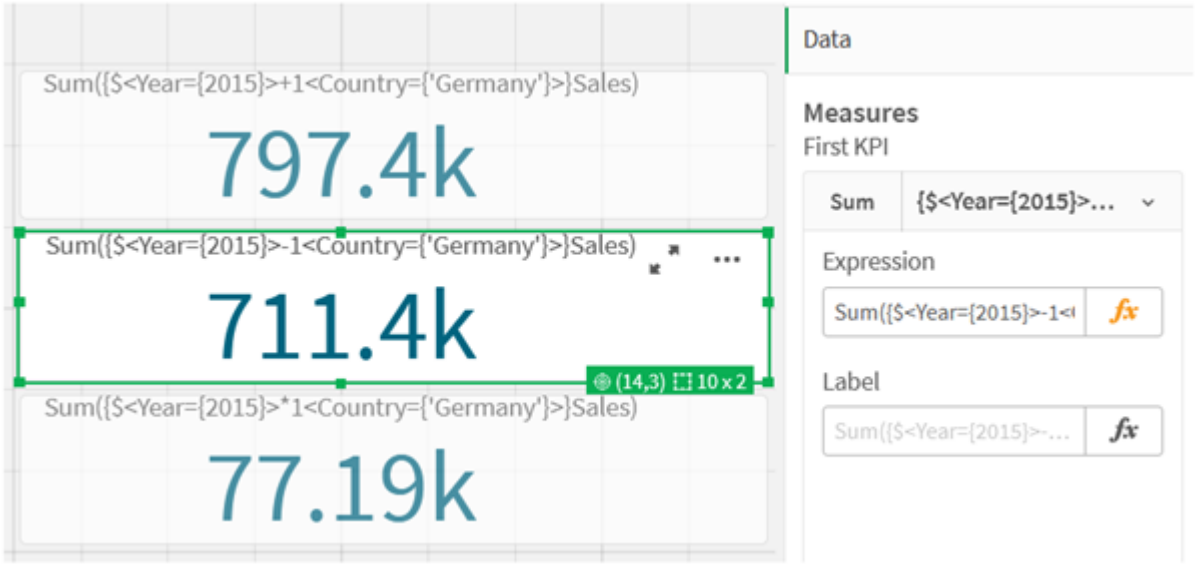
KPI som använder operatorn (+)

Alternativt kan du använda ett minustecken `(-)` för att returnera en datauppsättning som består av posterna som tillhör 2015 men inte Germany. Eller så kan du använda en asterisk `(*)` till att returnera en uppsättning som består av poster som tillhör båda uppsättningarna.

```
Sum({$<Year={2015}>-1<Country={'Germany'}>}Sales)
```

```
Sum({$<Year={2015}>*1<Country={'Germany'}>}Sales)
```

KPI:er som använder operatorer



Introduktionsdata för set-uttryck

Laddningsskript

Ladda följande data som en inline-laddning och skapa sedan diagramuttrycken i introduktionen.

```
//Create table SalesByCountry SalesByCountry: Load * Inline [ Country, Year, Sales Argentina,
2016, 66295.03 Argentina, 2015, 140037.89 Austria, 2016, 54166.09 Austria, 2015, 182739.87
Belgium, 2016, 182766.87 Belgium, 2015, 178042.33 Brazil, 2016, 174492.67 Brazil, 2015,
2104.22 Canada, 2016, 101801.33 Canada, 2015, 40288.25 Denmark, 2016, 45273.25 Denmark, 2015,
106938.41 Finland, 2016, 107565.55 Finland, 2015, 30583.44 France, 2016, 115644.26 France,
2015, 30696.98 Germany, 2016, 8775.18 Germany, 2015, 77185.68 ];
```

Syntax för set-uttryck

Hela syntaxen (utan det möjliga alternativet med vanliga parenteser för att definiera företräde) beskrivs med Backus-Naur-formalism:

```
set_expression ::= { set_entity { set_operator set_entity } }
set_entity ::= set_identifier [ set_modifier ] | set_modifier
set_identifier ::= 1 | $ | $N | $_N | bookmark_id | bookmark_name
set_operator ::= + | - | * | /
set_modifier ::= < field_selection {, field_selection } >
field_selection ::= field_name [ = | += | -= | *= | /= ] element_set_
expression
element_set_expression ::= [ - ] element_set { set_operator element_set }
element_set ::= [ field_name ] | { element_list } | element_function
element_list ::= element { , element }
element_function ::= ( P | E ) ( [set_expression] [field_name] )
element ::= field_value | " search_mask "
```

3.3 Allmän syntax för diagramuttryck

Den här allmänna syntaxstrukturen kan användas för diagramuttryck, med många valfria parametrar:

```
expression ::= ( constant | expressionname | operator1 expression | expression operator2
expression | function | aggregation function | (expression ) )
```
där:

constant är en sträng (en text, ett datum, en tid) inom enkla, raka citationstecken, eller ett tal. Konstanter skrivs utan tusentalsavgränsare och med decimalkomma som decimalavgränsare.

expressionname är namnet (etiketten) på ett annat uttryck i samma diagram.

operator1 är en unär operator (som arbetar med ett uttryck, det till höger).

operator2 är en binär operator (som arbetar med två uttryck, ett på varje sida).

```
function ::= functionname ( parameters )
parameters ::= expression { , expression }
```
Varken typen av parameter eller antalet parametrar är godtyckligt. utan beror på funktionen som används.

```
aggregationfunction ::= aggregationfunctionname ( parameters2 )
parameters2 ::= aggrexpression { , aggrexpression }
```
Varken typen av parameter eller antalet parametrar är godtyckligt. utan beror på funktionen som används.

3.4 Allmän syntax för aggregeringar

Den här allmänna syntaxstrukturen kan användas för aggregeringar, med många valfria parametrar:

```
aggrexpression ::= ( fieldref | operator1 aggrexpression | aggrexpression operator2
aggrexpression | functioninaggr | ( aggrexpression ) )
```
fieldref är ett fältnamn.

```
functionaggr ::= functionname ( parameters2 )
```

Uttryck och funktioner kan således nästlas efter önskemål. Så länge **fieldref** alltid är omslutet av exakt en aggregeringsfunktion, och under förutsättning att uttrycket returnerar ett värde som kan tolkas, ger Qlik Sense inte några felmeddelanden.

4 Operatorer

Det här avsnittet beskriver operatorerna som kan användas i Qlik Sense. Det finns två sorters operatorer:

- Unära operatorer (tar endast en operand)
- Binära operatorer (tar två operander)

De flesta operatorer är binära.

Följande operatorer kan definieras:

- Bit-operatorer
- Logiska operatorer
- Numeriska operatorer
- Relationsoperatorer
- Strängoperatorer

4.1 Bit-operatorer

Alla bit-operatorer konverterar (trunkerar) operanderna till signerade (32-bitars-) heltal och returnerar resultatet som signerade heltal. Alla operationer sker per bit på bitnivå. Om en operand inte kan tolkas som ett tal, returnerar operationen NULL.

Bit-operatorer

Operator	Fullständigt namn	Beskrivning
bitnot	Bit invers.	Unär operator. Operationen returnerar operandens logiska motsats på bitnivå. **Exempel:** `bitnot 17` returnerar -18.
bitand	Bit och.	Operationen returnerar logiskt och mellan operanderna bit för bit. **Exempel:** `17 bitand 7` returnerar 1.
bitor	Bit eller.	Operationen returnerar logiskt eller mellan operanderna bit för bit. **Exempel:** `17 bitor 7` returnerar 23.

Operator	Fullständigt namn	Beskrivning
bitxor	Bit exklusivt eller.	Operationen returnerar logiskt bitvist exklusivt eller mellan operanderna. **Exempel:** `17 bitxor 7` returnerar 22.
>>	Bit högerskift.	Operationen returnerar den första operanden skiftat till höger. Antalet steg anges i den andra operanden. **Exempel:** `8 >> 2` returnerar 2.
<<	Bit vänsterskift.	Operationen returnerar den första operanden skiftat till vänster. Antalet steg anges i den andra operanden. **Exempel:** `8 << 2` returnerar 32.

4.2 Logiska operatorer

Alla logiska operatorer tolkar operanderna logiskt och returnerar True (-1) eller False (0) som resultat.

Logiska operatorer

Operator	Beskrivning
not	Logisk motsats. En av de få unära operatorerna. Operationen returnerar operandens logiska motsats.
and	Logiskt och. Operationen returnerar operandernas logiska och.
or	Logiskt eller. Operationen returnerar operandernas logiska eller.
Xor	Logiskt exklusivt eller. Operationen returnerar operandernas logiska exklusiva eller. Detta liknar logiskt eller, men med den skillnaden att resultatet är False om båda operanderna är True.

4.3 Numeriska operatorer

Alla numeriska operatorer använder sig av operandernas numeriska värden och resulterar i ett numeriskt värde.

Numeriska operatorer

Operator	Beskrivning
+	Tecken för positivt tal (unär operator) eller aritmetisk addition. Den binära operatorn resulterar i summan av de två operanderna.
-	Tecken för negativt tal (unär operator) eller aritmetisk subtraktion. Den unära operationen resulterar i operanden multiplicerad med -1, den binära i skillnaden mellan de två operanderna.
*	Aritmetisk multiplikation. Operationen resulterar i produkten av de två operanderna.
/	Aritmetisk division. Operationen resulterar i kvoten av de två operanderna.

4.4 Relationsoperatorer

Alla relationsoperatorer jämför operandernas värden och returnerar True (-1) eller False (0) som resultat. Alla relationsoperatorer är binära.

Relationsoperatorer

Operator	Beskrivning
<	Mindre än. En numerisk jämförelse görs om båda operatorerna kan tolkas numeriskt. Operationen resulterar i jämförelsens logiska värde.
<=	Mindre än eller lika med. En numerisk jämförelse görs om båda operatorerna kan tolkas numeriskt. Operationen resulterar i jämförelsens logiska värde.
>	Större än. En numerisk jämförelse görs om båda operatorerna kan tolkas numeriskt. Operationen resulterar i jämförelsens logiska värde.
>=	Större än eller lika med. En numerisk jämförelse görs om båda operatorerna kan tolkas numeriskt. Operationen resulterar i jämförelsens logiska värde.
=	Lika med. En numerisk jämförelse görs om båda operatorerna kan tolkas numeriskt. Operationen resulterar i jämförelsens logiska värde.
<>	Inte lika med. En numerisk jämförelse görs om båda operatorerna kan tolkas numeriskt. Operationen resulterar i jämförelsens logiska värde.

Operator	Beskrivning
precedes	Till skillnad från med operatorn **<** görs inget försök att tolka argumentens värden numeriskt före jämförelsen. Åtgärden returnerar sant om värdet till vänster om operatorn består av text som, enligt strängjämförelsen, kommer före det högra värdets text. **Exempel:** `'1 ' precedes ' 2'` returnerar FALSE `' 1' precedes ' 2'` returnerar TRUE som ASCII-värdet för ett blanksteg (” ”) har ett mindre värde än ASCII-värdet för ett tal. Jämför detta med: `'1 ' < ' 2'` returnerar TRUE `' 1' < ' 2'` returnerar TRUE
follows	Till skillnad från med operatorn **>** görs inget försök att tolka argumentens värden numeriskt före jämförelsen. Åtgärden returnerar sant om värdet till vänster om operatorn består av text som, enligt strängjämförelsen, kommer efter det högra värdets text. **Exempel:** `' 2' follows '1'` returnerar FALSE ' 2' follows ' 1' returnerar TRUE som ASCII-värdet för ett blanksteg (” ”) har ett mindre värde än ASCII-värdet för ett tal. Jämför detta med: `' 2' > ' 1'` returnerar TRUE `' 2' > '1 '` returnerar TRUE

4.5 Strängoperatorer

Det finns två textsträngsoperatorer. Den ena använder sig av operandernas textsträngsvärden och resulterar i en textsträng. Den andra jämför operanderna och returnerar ett booleskt värde som markerar matchning.

&

Konkatenering av strängar. Operationen resulterar i en textsträng som består av de två operandernas textsträngar, den ena efter den andra.

Exempel:

`'abc' & 'xyz'` returnerar 'abcxyz'

like

Jämför strängar med jokertecken. Åtgärden returnerar ett booleskt True (-1) om strängen som föregår operatorn matchas av strängen som följer efter operatorn. Den andra strängen kan innehålla jokertecknen * (valfritt antal av godtyckliga tecken) eller ? (ett godtyckligt tecken).

Exempel:

`'abc' like 'a*'` returnerar True (-1)

`'abcd' like 'a?c*'` returnerar True (-1)

`'abc' like 'a??bc'` returnerar False (0)

5 Skript- och diagramfunktioner

Omvandla och aggregera data med hjälp av funktioner i dataladdningsskript och diagramuttryck.

Många funktioner kan användas på samma sätt både i dataladdningsskript och diagramuttryck, men det finns ett antal undantag:

- Vissa funktioner kan endast användas i dataladdningsskript. Dessa markeras med – skriptfunktion.
- Vissa funktioner kan endast användas i diagramuttryck. Dessa markeras med – diagramfunktion.
- Vissa funktioner kan användas i både dataladdningsskript och diagramuttryck, men med olikheter i parametrar och användning. Dessa beskrivs i separata avsnitt markerade med – skriptfunktion eller – diagramfunktion.

5.1 Analytiska kopplingar för komplement på serversidan (SSE)

Funktioner som aktiveras genom analytiska kopplingar visas bara om du har konfigurerat de analytiska kopplingarna och Qlik Sense har startats.

Du konfigurerar de analytiska kopplingarna i QMC, se avsnittet "Skapa en analytisk koppling" i guiden Hantera Qlik Sense-webbplatser.

I Qlik Sense Desktop konfigurerar du de analytiska kopplingarna genom att redigera filen *Settings.ini*, se avsnittet ”Konfigurera analytiska kopplingar i Qlik Sense Desktop” i guiden Qlik Sense Desktop.

5.2 Aggregeringsfunktioner

Den grupp av funktioner som kallas aggregeringsfunktioner utgörs av funktioner som tar flera fältvärden som indata och returnerar ett enda resultat per grupp, där grupperingen definieras av en diagramdimension eller en **group by**-sats i skriptsatsen.

Aggregeringsfunktionerna omfattar **Sum()**, **Count()**, **Min()**, **Max()** med flera.

De flesta aggregeringsfunktioner kan användas i både dataladdningsskriptet och diagramuttryck, men syntaxen är olika.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Använda aggregeringsfunktioner i ett dataladdningsskript

Aggregeringsfunktioner kan enbart användas inuti **LOAD** - och **SELECT**-satser.

Använda aggregeringsfunktioner i diagramuttryck

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

En aggregeringsfunktion aggregerar över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan dock definieras med ett s.k. set-uttryck i set-analys.

Så beräknas aggregeringar

En aggregering körs i en slinga över posterna för en specifik tabell och aggregerar posterna i den. Till exempel räknar **Count**(<Field>) antalet poster i tabellen där <Field> finns. Om du bara vill aggregera distinkta fältvärden ska du använda **distinct**-satsen, som **Count**(**distinct** <Field>).

Om aggregeringsfunktionen innehåller fält från olika tabeller kör aggregeringsfunktionen en slinga över posterna från de olika tabellerna för de befintliga fälten. Detta påverkar prestandan och därför bör sådana aggregeringar undvikas, speciellt när du har stora datamängder.

Aggregering av nyckelfält

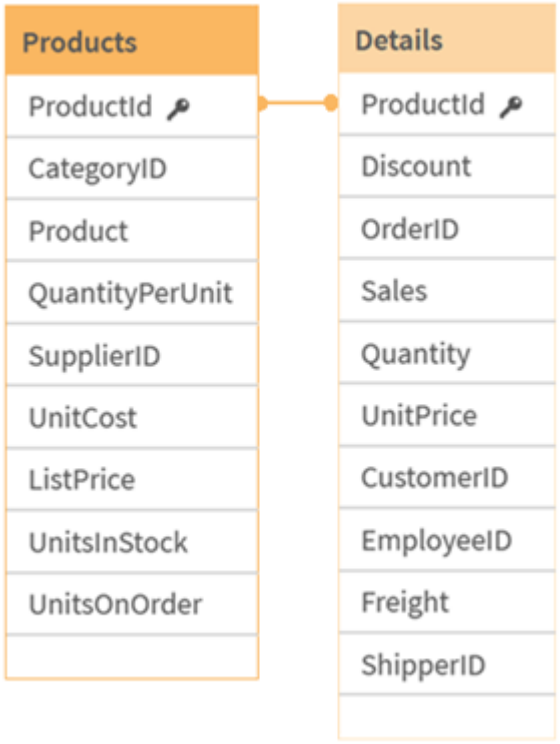
Sättet aggregeringar beräknas på gör att du inte kan aggregera nyckelfält, eftersom det inte framgår vilken tabell som ska användas för aggregeringen. Om fältet <Key> till exempel är länken mellan två tabeller, så framgår det inte om **Count**(<Key>) ska returnera antalet poster från den första eller andra tabellen.

Om du använder **distinct**-satsen är emellertid aggregeringen väldefinierad och kan beräknas.

Så om du har ett nyckelfält inuti en aggregeringsfunktion utan **distinct**-satsen returnerar Qlik Sense ett tal som kan vara meningslöst. Lösningen är att antingen använda **distinct**-satsen eller använda en kopia av nyckeln – en kopia som bara finns i en tabell.

I följande tabeller är ProductID nyckeln mellan tabellerna.

Nyckeln ProductID mellan tabellerna Products och Details



Count(ProductID) kan räknas antingen i tabellen Products (som bara har en post per produkt – ProductID är primärnyckeln) eller i tabellen Details (som antagligen har flera poster per produkt). Om du vill räkna antalet distinkta produkter ska du använda Count(distinct ProductID). Om du vill räkna antalet rader i en specifik tabell ska du inte använda nyckeln.

Grundläggande aggregeringsfunktioner

Grundläggande aggregeringsfunktioner - en översikt

Grundläggande aggregeringsfunktioner är en grupp av de vanligaste aggregeringsfunktionerna.

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

Grundläggande aggregeringsfunktioner i dataladdningsskriptet

FirstSortedValue

FirstSortedValue() returnerar värdet från det uttryck som har angetts i **value** som motsvarar resultatet av sorteringen av **sort_weight**-argumentet, exempelvis namnet på produkten med det lägsta enhetspriset. Det n:te värdet i sorteringsordningen kan anges i **rank**. Om fler än ett resultatvärde delar samma **sort_weight** för den angivna **rank** returnerar funktionen NULL. De sorterade värdena itereras över ett antal poster, enligt vad som definieras i en **group by**-sats, eller aggregerat över den fullständiga datauppsättningen om ingen **group by**-sats har definierats.

```
FirstSortedValue ([ distinct ] expression, sort_weight [, rank ])
```

Max

Max() hittar det högsta numeriska värdet för aggregerade data i uttrycket, som definieras av en **group by**-sats. Genom att ange en **rank** n, återfinns det n:te högsta värdet.

```
Max ( expression[, rank])
```

Min

Min() returnerar det lägsta numeriska värdet för aggregerade data i uttrycket, som definieras av en **group by**-sats. Genom att ange en **rank** n återfinns det n:te lägsta värdet.

```
Min ( expression[, rank])
```

Mode

Mode() returnerar det vanligaste värdet, lägesvärdet, för aggregerade data i uttrycket, som definieras av en **group by**-sats. Funktionen **Mode()** kan returnera numeriska värden såväl som textvärden.

```
Mode (expression )
```

Only

Only() returnerar ett värde om det finns ett, och endast ett, möjligt resultat från aggregerade data. Om poster endast innehåller ett värde returneras detta värde. Annars returneras NULL. Använd **group by**-satsen för att utvärdera över flera poster. Funktionen **Only()** kan returnera numeriska värden och textvärden.

```
Only (expression )
```

Sum

Sum() beräknar summan av värden aggregerade i uttrycket, som definieras av en **group by**-sats.

```
Sum ([distinct]expression)
```

Grundläggande aggregeringsfunktioner i diagramuttryck

Aggregeringsfunktioner för diagram kan endast användas på fält i diagramuttryck. Argumentuttrycket för en aggregeringsfunktion får inte innehålla en annan aggregeringsfunktion.

FirstSortedValue

FirstSortedValue() returnerar värdet från det uttryck som har angetts i **value** som motsvarar resultatet av sorteringen av **sort_weight**-argumentet, exempelvis namnet på produkten med det lägsta enhetspriset. Det n:te värdet i sorteringsordningen kan anges i **rank**. Om fler än ett resultatvärde delar samma **sort_weight** för den angivna **rank** returnerar funktionen NULL.

```
FirstSortedValue - diagramfunktion([{SetExpression}] [DISTINCT] [TOTAL [<fld
{,fld}>]] value, sort_weight [,rank])
```

Max

Max() finner det högsta värdet för aggregerade data. Genom att ange en **rank** n, återfinns det n:te högsta värdet.

```
Max - diagramfunktionMax() finner det högsta värdet för aggregerade data.
Genom att ange en rank n, återfinns det n:te högsta värdet. Titta gärna på
FirstSortedValue och rangemax, som har liknande funktionalitet som Max-
funktionen. Max([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank])
```

```
numeriska ArgumentArgumentBeskrivningexprDet uttryck eller fält som
innehåller de data som ska mätas.rankStandardvärdet för rank är 1, vilket
motsvarar det högsta värdet. Om du anger rank som 2 returneras det näst
högsta värdet. Om rank är 3 returneras det tredje högsta
värdet.SetExpressionSom standard kommer aggregeringsfunktionen att aggregera
över den uppsättning möjliga poster som definierats av urvalet. En alternativ
uppsättning poster kan definieras med ett uttryck för set-analys. TOTALOm
ordet TOTAL står före funktionsargumenten görs beräkningen över alla valbara
värden givet de aktuella valen, och inte bara sådana som rör det aktuella
dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att
använda TOTAL [<fld {.fld}>], där kvalificeraren TOTAL följs av en lista med
ett eller flera fältnamn som en delmängd av diagramdimensionens variabler,
skapar du en delmängd av de totala möjliga
värdena. DataCustomerProductUnitSalesUnitPrice
AstridaAA416AstridaAA1015AstridaBB99BetacabBB510BetacabCC220BetacabDD-
25CanutilityAA815CanutilityCC-19Exempel och resultatExempelResultatMax
(UnitSales)10, eftersom detta är det högsta värdet i UnitSales.Värdet för en
order beräknas utifrån antal sålda enheter i (UnitSales) multiplicerat med
priset per enhet.Max(UnitSales*UnitPrice)150, eftersom detta är det högsta
värdet som blir resultatet av att beräkna alla möjliga värden av (UnitSales)*
(UnitPrice).Max(UnitSales, 2)9, som är det näst högsta värdet.Max
(TOTAL UnitSales)10, eftersom TOTAL-kvalificeraren innebär att det högsta
möjliga värdet hittas, oavsett diagramdimensionerna. För ett diagram med
Customer som dimension säkerställer kvalificeraren TOTAL att det maximala
värdet över den fullständiga datauppsättningen returneras, i stället för
maximal UnitSales för varje kund.Välj Customer B.Max({1} TOTAL UnitSales)10,
oberoende av urval, eftersom Set Analysis-uttrycket {1} definierar den
uppsättning poster som ska utvärderas som ALL, oavsett vilket urval som har
gjorts.Data som används i exempel: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() finner det lägsta värdet för aggregerade data. Genom att ange en **rank** n återfinns det n:te lägsta värdet.

```
Min - diagramfunktion([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr [,rank])
```

Mode

Mode() finner det vanligast förekommande värdet, lägesvärdet, i aggregerade data. Funktionen **Mode()** kan behandla textvärden samt numeriska värden.

```
Mode - diagramfunktion ({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Only

Only() returnerar ett värde om det finns ett, och endast ett, möjligt resultat från aggregerade data. Till exempel returnerar en sökning efter den enda produkten med styckpriset =9 NULL om fler än en produkt har styckpriset 9.

```
Only - diagramfunktion([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr)
```

Sum

Sum() beräknar summan av de värden som ges av uttrycket eller fältet över aggregerade data.

```
Sum - diagramfunktion([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr])
```

FirstSortedValue

FirstSortedValue() returnerar värdet från det uttryck som har angetts i **value** som motsvarar resultatet av sorteringen av **sort_weight**-argumentet, exempelvis namnet på produkten med det lägsta enhetspriset. Det n:te värdet i sorteringsordningen kan anges i **rank**. Om fler än ett resultatvärde delar samma **sort_weight** för den angivna **rank** returnerar funktionen NULL. De sorterade värdena itereras över ett antal poster, enligt vad som definieras i en **group by**-sats, eller aggregerat över den fullständiga datauppsättningen om ingen **group by**-sats har definierats.

Syntax:

```
FirstSortedValue ([ distinct ] value, sort-weight [, rank ])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
value Expression	Funktionen finner värdet av uttrycket **value** som motsvarar resultatet vid sortering av **sort_weight**.
sort-weight Expression	Uttrycket som innehåller data som ska sorteras. Det första (lägsta) värdet i **sort_weight** hittas, varifrån motsvarande värde i uttrycket **value** fastställs. Om du sätter ett minustecken framför **sort_weight** returnerar funktionen det sista (högsta) sorterade värdet i stället.
rank Expression	Genom att ange **rank** "n" som är större än 1 får du det n:te sorterade värdet.
distinct	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår app för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Skriptexempel

Exempel	Resultat

`Temp:`
`LOAD * inline [`
`Customer|Product|OrderNumber|UnitSales|CustomerID`
`Astrida|AA|1|10|1`
`Astrida|AA|7|18|1`
`Astrida|BB|4|9|1`
`Astrida|CC|6|2|1`
`Betacab|AA|5|4|2`
`Betacab|BB|2|5|2`
`Betacab|DD|12|25|2`
`Canutility|AA|3|8|3`
`Canutility|CC|13|19|3`
`Divadip|AA|9|16|4`
`Divadip|AA|10|16|4`
`Divadip|DD|11|10|4`
`] (delimiter is '|');`
 	
`FirstSortedValue:`
`LOAD Customer,FirstSortedValue(Product, UnitSales)`
`as MyProductWithSmallestOrderByCustomer Resident`
`Temp Group By Customer;`
	`Customer`
	`MyProductWithSmallestOrderByCustomer`
	`Astrida CC`
	`Betacab AA`
	`Canutility AA`
	`Divadip DD`
	Funktionen sorterar UnitSales från den minsta till den största, och söker efter värdet Customer med det minsta värdet för UnitSales, den minsta ordern.
	
	Eftersom CC motsvarar den minsta ordern (värdet UnitSales = 2) för kunden Astrida. AA motsvarar den minsta ordern (4) för kunden Betacab, AA motsvarar den minsta ordern (8) för kunden Canutility, och DD motsvarar den minsta ordern (10) för kunden Divadip..

Givet att **Temp**-tabellen är laddad som i föregående exempel:
 	
`LOAD Customer,FirstSortedValue(Product, -UnitSales)`
`as MyProductWithLargestOrderByCustomer Resident`
`Temp Group By Customer;`
	`Customer`
	`MyProductWithLargestOrderByCustomer`
	`Astrida AA`
	`Betacab DD`
	`Canutility CC`
	`Divadip –`
	Argumentet sort_weight inleds med ett minustecken så att funktionen sorterar den största först.
	
	Eftersom AA motsvarar den största ordern (värdet av UnitSales:18) för kunden Astrida, DD motsvarar den största ordern (12) för kunden Betacab och CC motsvarar den största ordern (13) för kunden Canutility. Det finns två identiska värden för den största ordern (16) för kunden Divadip, därför producerar detta ett null-resultat.

Exempel	Resultat

Givet att **Temp**-tabellen är laddad som i föregående exempel:
`LOAD Customer,FirstSortedValue(distinct Product, -UnitSales) as MyProductWithSmallestOrderByCustomer Resident Temp Group By Customer;`
	`Customer MyProductWithLargestOrderByCustomer`
	`Astrida AA`
	`Betacab DD`
	`Canutility CC`
	`Divadip AA`
	Detta är samma som i föregående exempel, förutom att kvalificeraren distinct används. Detta gör att dubblettresultatet för Divadip ignoreras, så att ett icke-null-värde kan returneras.

FirstSortedValue - diagramfunktion

FirstSortedValue() returnerar värdet från det uttryck som har angetts i **value** som motsvarar resultatet av sorteringen av **sort_weight**-argumentet, exempelvis namnet på produkten med det lägsta enhetspriset. Det n:te värdet i sorteringsordningen kan anges i **rank**. Om fler än ett resultatvärde delar samma **sort_weight** för den angivna **rank** returnerar funktionen NULL.

Syntax:

```
FirstSortedValue([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] value, sort_weight [,rank])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
value	Utdatafält. Funktionen finner värdet av uttrycket **value** som motsvarar resultatet vid sortering av **sort_weight**.
sort_weight	Indatafält. Uttrycket som innehåller data som ska sorteras. Det första (lägsta) värdet i **sort_weight** hittas, varifrån motsvarande värde i uttrycket **value** fastställs. Om du sätter ett minustecken framför **sort_weight** returnerar funktionen det sista (högsta) sorterade värdet i stället.
rank	Genom att ange **rank** "n" som är större än 1 får du det n:te sorterade värdet.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.

Argument	Beskrivning
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel och resultat:

Data

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

Exempel och resultat

Exempel	Resultat
`firstsortedvalue (Product, UnitPrice)`	BB, som är en `Product` med det lägsta `UnitPrice`(9).
`firstsortedvalue (Product, UnitPrice, 2)`	BB, som är en `Product` med det näst lägsta `UnitPrice`(10).
`firstsortedvalue (Customer, -UnitPrice, 2)`	Betacab, som är en `Customer`med den `Product` som har det nästa högsta `UnitPrice`(20).
`firstsortedvalue (Customer, UnitPrice, 3)`	NULL, eftersom det finns två värden för `Customer` (Astrida och Canutility) med sammarank (tredje lägsta) `UnitPrice`(15). Använd kvalificeraren distinct för att se till att det inte uppstår oväntade null-resultat.
`firstsortedvalue (Customer, -UnitPrice*UnitSales, 2)`	Canutility, vilket är `Customer` med det näst högsta säljordervärdet `UnitPrice` multiplicerat med `UnitSales` (120).

Data som används i exempel:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Max

Max() hittar det högsta numeriska värdet för aggregerade data i uttrycket, som definieras av en **group by**-sats. Genom att ange en **rank** n, återfinns det n:te högsta värdet.

Syntax:

```
Max ( expr [, rank])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.
rank Expression	Standardvärdet för **rank** är 1, vilket motsvarar det högsta värdet. Om du anger **rank** som 2 returneras det näst högsta värdet. Om **rank** är 3 returneras det tredje högsta värdet.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår app för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Exempel:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
```

```
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');

Max:
LOAD Customer, Max(UnitSales) as MyMax Resident Temp Group By Customer;
```

Resultattabell

Customer	MyMax
Astrida	18
Betacab	5
Canutility	8

Exempel:

Givet att **Temp**-tabellen är laddad som i föregående exempel:

```
LOAD Customer, Max(UnitSales,2) as MyMaxRank2 Resident Temp Group By Customer;
```

Resultattabell

Customer	MyMaxRank2
Astrida	10
Betacab	4
Canutility	-

Max - diagramfunktion

Max() finner det högsta värdet för aggregerade data. Genom att ange en **rank** n, återfinns det n:te högsta värdet.

*Titta gärna på **FirstSortedValue** och **rangemax**, som har liknande funktionalitet som **Max**-funktionen.*

Syntax:

```
Max([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Argument	Beskrivning
rank	Standardvärdet för **rank** är 1, vilket motsvarar det högsta värdet. Om du anger **rank** som 2 returneras det näst högsta värdet. Om **rank** är 3 returneras det tredje högsta värdet.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel och resultat:

Data

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

Exempel och resultat

Exempel	Resultat
`Max(UnitSales)`	10, eftersom detta är det högsta värdet i `UnitSales`.

Exempel	Resultat
Värdet för en order beräknas utifrån antal sålda enheter i (`UnitSales`) multiplicerat med priset per enhet. `Max (UnitSales*UnitPrice)`	150, eftersom detta är det högsta värdet som blir resultatet av att beräkna alla möjliga värden av (`UnitSales`)*(`UnitPrice`).
`Max(UnitSales, 2)`	9, som är det näst högsta värdet.
`Max(TOTAL UnitSales)`	10, eftersom TOTAL-kvalificeraren innebär att det högsta möjliga värdet hittas, oavsett diagramdimensionerna. För ett diagram med Customer som dimension säkerställer kvalificeraren TOTAL att det maximala värdet över den fullständiga datauppsättningen returneras, i stället för maximal UnitSales för varje kund.
Välj `Customer B`. `Max({1} TOTAL UnitSales)`	10, oberoende av urval, eftersom Set Analysis-uttrycket {1} definierar den uppsättning poster som ska utvärderas som ALL, oavsett vilket urval som har gjorts.

Data som används i exempel:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Se även:

- *FirstSortedValue - diagramfunktion (page 213)*
- *RangeMax (page 706)*

Min

Min() returnerar det lägsta numeriska värdet för aggregerade data i uttrycket, som definieras av en **group by**-sats. Genom att ange en **rank** n återfinns det n:te lägsta värdet.

Syntax:

```
Min ( expr [, rank])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.
rank Expression	Standardvärdet för **rank** är 1, vilket motsvarar det lägsta värdet. Om du anger **rank** som 2, returneras det näst lägsta värdet. Om **rank** är 3, returneras det tredje lägsta värdet och så vidare.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår app för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Exempel:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');
Min:
LOAD Customer, Min(UnitSales) as MyMin Resident Temp Group By Customer;
```

Resultattabell

Customer	MyMin
Astrida	2
Betacab	4
Canutility	8

Exempel:

Givet att **Temp**-tabellen är laddad som i föregående exempel:

```
LOAD Customer, Min(UnitSales,2) as MyMinRank2 Resident Temp Group By Customer;
```

Resultattabell

Customer	MyMinRank2
Astrida	9
Betacab	5
Canutility	-

Min - diagramfunktion

Min() finner det lägsta värdet för aggregerade data. Genom att ange en **rank** n återfinns det n:te lägsta värdet.

*Titta gärna på **FirstSortedValue** och **rangemin**, som har liknande funktionalitet som **Min**-funktionen.*

Syntax:

```
Min({[SetExpression] [TOTAL [<fld {,fld}>]]} expr [,rank])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
rank	Standardvärdet för **rank** är 1, vilket motsvarar det lägsta värdet. Om du anger **rank** som 2, returneras det näst lägsta värdet. Om **rank** är 3, returneras det tredje lägsta värdet och så vidare.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel och resultat:

Data

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

Funktionen Min() måste returnera ett värde som inte får vara NULL från den uppsättning värden som uttrycket ger, om ett sådant finns. Eftersom det finns NULL-värden i de data som anges i exemplen, returnerar funktionen de första värden som inte är NULL som utvärderas från värdeuttrycken.

Exempel och resultat

Exempel	Resultat
`Min(UnitSales)`	2, eftersom detta är det lägsta värde som inte är NULL i `UnitSales`.
Värdet för en order beräknas utifrån antal sålda enheter i (`UnitSales`) multiplicerat med priset per enhet. `Min (UnitSales*UnitPrice)`	40, eftersom detta är det lägsta värde som inte är NULL som blir resultatet av att beräkna alla tänkbara värden av (`UnitSales`)*(`UnitPrice`).
`Min(UnitSales, 2)`	4, vilket är det näst lägsta värdet (efter NULL-värdena).
`Min(TOTAL UnitSales)`	2, eftersom TOTAL-kvalificeraren innebär att det lägsta möjliga värdet hittas, oavsett diagramdimensionerna. För en tabell med Customer som dimension säkerställer TOTAL-kvalificeraren att det minsta värdet över den fullständiga datauppsättningen returneras, istället för det minsta UnitSales-värdet för varje kund.

Exempel	Resultat

Välj Customer B.
 	
`Min({1}`
`TOTAL UnitSales)`
	2, vilket är oberoende av urvalet i Customer B.
	
	Set Analysis-uttrycket {1} definierar den uppsättning poster som ska utvärderas som ALL, oavsett vilket urval som har gjorts.

Data som används i exempel:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Se även:

- *FirstSortedValue - diagramfunktion (page 213)*
- *RangeMin (page 710)*

Mode

Mode() returnerar det vanligaste värdet, lägesvärdet, för aggregerade data i uttrycket, som definieras av en **group by**-sats. Funktionen **Mode()** kan returnera numeriska värden såväl som textvärden.

Syntax:

```
Mode ( expr)
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.

Begränsningar:

Om fler än ett värde är lika vanligt förekommande returneras NULL.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår app för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Skriptexempel

Exempel	Resultat

`Temp:`
`LOAD * inline [`
`Customer|Product|OrderNumber|UnitSales|CustomerID`
`Astrida|AA|1|10|1`
`Astrida|AA|7|18|1`
`Astrida|BB|4|9|1`
`Astrida|CC|6|2|1`
`Betacab|AA|5|4|2`
`Betacab|BB|2|5|2`
`Betacab|DD`
`Canutility|DD|3|8`
`Canutility|CC`
`] (delimiter is '|');`
 	
`Mode:`
`LOAD Customer, Mode(Product) as MyMostOftenSoldProduct`
`Resident Temp Group By Customer;`
	MyMostOftenSoldProduct
	
	AA
	
	eftersom AA är den enda produkt som har sålts mer än en gång.

Mode - diagramfunktion

Mode() finner det vanligast förekommande värdet, lägesvärdet, i aggregerade data. Funktionen **Mode()** kan behandla textvärden samt numeriska värden.

Syntax:

```
Mode({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {,fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel och resultat:

Data

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

Exempel och resultat

Exempel	Resultat
`Mode(UnitPrice)` Välj `Customer A`.	15, eftersom detta är det vanligaste värdet i `UnitSales`. Returnerar NULL (-). Inget enskilt värde uppträder oftare än ett annat.
`Mode(Product)` Gör valet `Customer A`.	AA, eftersom detta är det vanligaste värdet i `Product`. Returnerar NULL (-). Inget enskilt värde uppträder oftare än ett annat.
`Mode` `(TOTAL UnitPrice)`	15, eftersom TOTAL-kvalificeraren innebär att det vanligaste värdet fortfarande är 15, oavsett diagramdimensioner.

Välj `Customer B`.
 	
`Mode({1}`
`TOTAL UnitPrice)`
	15, oberoende av urval, eftersom Set Analysis-uttrycket {1} definierar den uppsättning poster som ska utvärderas som ALL, oavsett vilket urval som har gjorts.

Data som används i exempel:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Se även:

- *Avg - diagramfunktion (page 267)*
- *Median - diagramfunktion (page 303)*

Only

Only() returnerar ett värde om det finns ett, och endast ett, möjligt resultat från aggregerade data. Om poster endast innehåller ett värde returneras detta värde. Annars returneras NULL. Använd **group by**-satsen för att utvärdera över flera poster. Funktionen **Only()** kan returnera numeriska värden och textvärden.

Syntax:

```
Only ( expr )
```

Returnerad datatyp: dual

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår app för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');
Only:
LOAD Customer, Only(CustomerID) as MyUniqIDCheck Resident Temp Group By Customer;
```

Resultattabell

Customer	MyUniqIDCheck
Astrida	1 eftersom enbart kund Astrida har fullständiga register som omfattar CustomerID.

Only - diagramfunktion

Only() returnerar ett värde om det finns ett, och endast ett, möjligt resultat från aggregerade data. Till exempel returnerar en sökning efter den enda produkten med styckpriset =9 NULL om fler än en produkt har styckpriset 9.

Syntax:

```
Only([{SetExpression}] [TOTAL [<fld {,fld}>]] expr)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Använd Only() när du vill ha ett NULL-resultat om det finns flera möjliga värden i exempeldata.

Exempel och resultat:

Data

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15

Customer	Product	UnitSales	UnitPrice
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Exempel och resultat

Exempel	Resultat
`Only({<UnitPrice={9}>} Product)`	BB, eftersom detta är den enda `Product` som har ett `UnitPrice` på "9".
`Only({<Product={DD}>} Customer)`	Betacab, eftersom det är den enda `Customer` som säljer en `Product` som heter ”DD”.
`Only({<UnitPrice={20}>} UnitSales)`	Det finns 2 `UnitSales` där `UnitPrice` är 20, eftersom det enbart finns ett värde för `UnitSales` där `UnitPrice` = 20.
`Only({<UnitPrice={15}>} UnitSales)`	NULL, eftersom det finns två värden för `UnitSales` där (`UnitPrice`) = 15.

Data som används i exempel:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Sum

Sum() beräknar summan av värden aggregerade i uttrycket, som definieras av en **group by**-sats.

Syntax:

```
sum ( [ distinct] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
distinct	Om predikatet **distinct** förekommer framför uttrycket kommer alla dubbletter att ignoreras.
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår app för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');
Sum:
LOAD Customer, Sum(UnitSales) as MySum Resident Temp Group By Customer;
```

Resultattabell

Customer	MySum
Astrida	39
Betacab	9
Canutility	8

Sum - diagramfunktion

Sum() beräknar summan av de värden som ges av uttrycket eller fältet över aggregerade data.

Syntax:

```
Sum([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten. *Även om DISTINCT-kvalificeraren stöds bör du vara mycket försiktig med att använda den eftersom den kan ge läsaren en felaktig bild av att ett totalt värde visas om vissa data har utelämnats.*
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel och resultat:

Data

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

Exempel och resultat

Exempel	Resultat
`Sum(UnitSales)`	38. Summan av värdena i `UnitSales`.
`Sum(UnitSales*UnitPrice)`	505. Summan av `UnitPrice` multiplicerat med `UnitSales` aggregerat.
`Sum (TOTAL UnitSales*UnitPrice)`	505 för alla rader i tabellen samt totalen, eftersom kvalificeraren TOTAL innebär att summan fortfarande är 505, och diagramdimensionerna ignoreras.
Välj `Customer B`. `Sum({1} TOTAL UnitSales*UnitPrice)`	505, oberoende av urval, eftersom Set Analysis-uttrycket {1} definierar den uppsättning poster som ska utvärderas som ALL, oavsett vilket urval som har gjorts.

Data som används i exempel:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Räkneaggregeringsfunktioner

Räkneaggregeringsfunktioner returnerar olika slags beräkningar av ett uttryck över ett antal poster i ett dataladdningsskript eller ett antal värden i en dimension i ett diagram.

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

Räkneaggregeringsfunktioner i dataladdningsskriptet

Count

Count() returnerar summan av-värden aggregerade i uttrycket, som definieras av en **group by**-sats.

```
Count ([distinct ] expression | * )
```

MissingCount

MissingCount() returnerar summan av-saknade värden aggregerade i uttrycket, som definieras av en **group by**-sats.

```
MissingCount ([ distinct ] expression)
```

NullCount

NullCount() returnerar summan av NULL-värden aggregerade i uttrycket, som definieras av en **group by**-sats.

```
NullCount ([ distinct ] expression)
```

NumericCount

NumericCount() returnerar summan av-numeriska värden aggregerade i uttrycket, som definieras av en **group by**-sats.

```
NumericCount ([ distinct ] expression)
```

TextCount

TextCount() returnerar summan av fältvärden som är icke-numeriska aggregerade i uttrycket, som definieras av en **group by**-sats.

```
TextCount ([ distinct ] expression)
```

Räkneaggregeringsfunktioner i diagramuttryck

Följande räkneaggregeringsfunktioner kan användas i diagram:

Count

Count() används för att aggregera antalet värden, textvärden och numeriska värden, i varje diagramdimension.

```
Count - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]}
expr)
```

MissingCount

MissingCount() används för att aggregera antalet saknade värden i varje diagramdimension. Saknade värden är alltid icke-numeriska.

```
MissingCount - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]] expr)
```

NullCount

NullCount() används för att aggregera antalet NULL-värden i varje diagramdimension.

```
NullCount - diagramfunktion({[SetExpression][DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

NumericCount

NumericCount() aggregerar antalet numeriska värden i varje diagramdimension.

```
NumericCount - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

TextCount

TextCount() används för att aggregera antalet fältvärden som är icke-numeriska i varje diagramdimension.

```
TextCount - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

Count

Count() returnerar summan av-värden aggregerade i uttrycket, som definieras av en **group by**-sats.

Syntax:

```
Count( [distinct ] expr)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket ignoreras alla dubbletter.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår app för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Skriptexempel

Exempel	Resultat
Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|UnitPrice Astrida|AA|1|4|16 Astrida|AA|7|10|15 Astrida|BB|4|9|9 Betacab|CC|6|5|10 Betacab|AA|5|2|20 Betacab|BB|1|25| 25 Canutility|AA|3|8|15 Canutility|CC|||19 Divadip|CC|2|4|16 Divadip|DD|3|1|25] (delimiter is '|'); Count1: LOAD Customer,Count(OrderNumber) as OrdersByCustomer Resident Temp Group By Customer;	Customer OrdersByCustomer Astrida 3 Betacab 3 Canutility 2 Divadip 2 Så länge dimensionen Customer tas med i tabellen på arket, annars blir resultatet för OrdersByCustomer 3, 2.

Exempel	Resultat
Givet att **Temp**-tabellen är laddad som i föregående exempel: `LOAD Count(OrderNumber) as TotalOrderNumber Resident Temp;`	`TotalOrderNumber` `10`

Givet att **Temp**-tabellen är laddad som i det första exemplet:
`LOAD Count(distinct OrderNumber) as TotalOrderNumber Resident Temp;`
	`TotalOrderNumber`
	`8`
	Eftersom det finns två värden för OrderNumber med samma värde, 1, och ett nollvärde.

Count - diagramfunktion

Count() används för att aggregera antalet värden, textvärden och numeriska värden, i varje diagramdimension.

Syntax:

```
Count({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel och resultat:

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

I följande exempel antas att alla kunder är valda, utom där det särskilt anges.

Exempel och resultat

Exempel	Resultat
`Count(OrderNumber)`	10, eftersom det finns 10 fält som kan ha ett värde för OrderNumber, och alla poster, även tomma, räknas. *"0" räknas som ett värde och inte som en tom cell. Om ett värde aggregeras till 0 för en dimension kommer den dimensionen dock inte att tas med i diagram.*
`Count(Customer)`	10, eftersom Count utvärderar antalet förekomster i alla fält.
`Count(DISTINCT [Customer])`	4, eftersom användning av Distinct-kvalificeraren innebär att Count endast utvärderar unika förekomster.
Anta att kunden Canutility är vald `Count (OrderNumber)/Count ({1} TOTAL OrderNumber)`	0.2 eftersom uttrycket returnerar antal order från den valda kunden i form av en procentsats av order från alla kunder. I detta fall 2/10.

Exempel	Resultat
Anta att kunderna Astrida och Canutility är valda `Count(TOTAL <Product> OrderNumber)`	5 eftersom detta är antalet order lagda för produkter för enbart de valda kunderna och tomma celler räknas.

Data som används i exempel:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB|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() returnerar summan av-saknade värden aggregerade i uttrycket, som definieras av en **group by**-sats.

Syntax:

```
MissingCount ( [ distinct ] expr)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket ignoreras alla dubbletter.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår app för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Skriptexempel

Exempel	Resultat

`Temp:`
`LOAD * inline [`
`Customer|Product|OrderNumber|UnitSales|UnitPrice`
`Astrida|AA|1|4|16`
`Astrida|AA|7|10|15`
`Astrida|BB|4|9|9`
`Betacab|CC|6|5|10`
`Betacab|AA|5|2|20`
`Betacab|BB||| 25`
`Canutility|AA|||15`
`Canutility|CC| ||19`
`Divadip|CC|2|4|16`
`Divadip|DD|3|1|25`
`] (delimiter is '|');`
`MissCount1:`
`LOAD Customer,MissingCount(OrderNumber) as`
`MissingOrdersByCustomer Resident Temp Group By Customer;`
 	
`Load MissingCount(OrderNumber) as TotalMissingCount`
`Resident Temp;`
	`Customer`
	`MissingOrdersByCustomer`
	`Astrida 0`
	`Betacab 1`
	`Canutility 2`
	`Divadip 0`
	
	Den andra satsen ger:
	
	`TotalMissingCount`
	`3`
	i en tabell med den dimensionen.

Givet att **Temp**-tabellen är laddad som i föregående exempel:
 	
`LOAD MissingCount(distinct OrderNumber) as`
`TotalMissingCountDistinct Resident Temp;`
	`TotalMissingCountDistinct`
	`1`
	Eftersom det finns enbart ett saknat OrderNumber-värde.

MissingCount - diagramfunktion

MissingCount() används för att aggregera antalet saknade värden i varje diagramdimension. Saknade värden är alltid icke-numeriska.

Syntax:

```
MissingCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.

Argument	Beskrivning
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel och resultat:

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			25
Canutility	AA			15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

Exempel och resultat

Exempel	Resultat
`MissingCount([OrderNumber])`	3, eftersom 3 av de 10 OrderNumber-fälten är tomma *"0" räknas som ett värde och inte som en tom cell. Om ett värde aggregeras till 0 för en dimension kommer den dimensionen dock inte att tas med i diagram.*

Exempel	Resultat
`MissingCount ([OrderNumber])/MissingCount ({1} Total [OrderNumber])`	Uttrycket returnerar antalet ofullständiga order från den valda kunden i form av en decimaldel av ofullständiga order från alla kunder. Det finns totalt 3 saknade värden för OrderNumber för alla kunder. Så för varje Customer som har ett saknat värde för Product är resultatet 1/3.

Data som används i exemplet:

```
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() returnerar summan av NULL-värden aggregerade i uttrycket, som definieras av en **group by**-sats.

Syntax:

```
NullCount ( [ distinct ] expr)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket ignoreras alla dubbletter.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår app för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Skriptexempel

Exempel	Resultat
Set NULLINTERPRET = NULL; Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|CustomerID Astrida|AA|1|10|1 Astrida|AA|7|18|1 Astrida|BB|4|9|1 Astrida|CC|6|2|1 Betacab|AA|5|4|2 Betacab|BB|2|5|2 Betacab|DD||| Canutility|AA|3|8| Canutility|CC|NULL||] (delimiter is '|'); Set NULLINTERPRET=; NullCount1: LOAD Customer,NullCount(OrderNumber) as NullOrdersByCustomer Resident Temp Group By Customer; LOAD NullCount(OrderNumber) as TotalNullCount Resident Temp;	Customer NullOrdersByCustomer Astrida 0 Betacab 0 Canutility 1 Den andra satsen ger: TotalNullCount 1 i en tabell med den dimensionen, eftersom endast en post innehåller ett null-värde.

NullCount - diagramfunktion

NullCount() används för att aggregera antalet NULL-värden i varje diagramdimension.

Syntax:

```
NullCount({[SetExpression][DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
set_ expression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.

Argument	Beskrivning
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
NullCount ([OrderNumber])	1 eftersom vi har introducerat ett null-värde med hjälp av NullInterpret i inline-satsen **LOAD**.

Data som används i exemplet:

```
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() returnerar summan av-numeriska värden aggregerade i uttrycket, som definieras av en **group by**-sats.

Syntax:

```
NumericCount ( [ distinct ] expr)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket ignoreras alla dubbletter.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår app för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Skriptexempel

Exempel	Resultat
`LOAD NumericCount(OrderNumber) as TotalNumericCount Resident Temp;`	Den andra satsen ger: TotalNumericCount 7 i en tabell med den dimensionen.
Givet att **Temp**-tabellen är laddad som i föregående exempel: `LOAD NumericCount(distinct OrderNumber) as TotalNumericCountDistinct Resident Temp;`	TotalNumericCountDistinct 6 Eftersom det finns ett OrderNumber som duplicerar ett annat, innebär det att resultatet är 6 som inte är dubbletter.

Exempel:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|7|1|25
] (delimiter is '|');
NumCount1:
LOAD Customer,NumericCount(OrderNumber) as NumericCountByCustomer Resident Temp Group By
Customer;
```

Resultattabell

Customer	NumericCountByCustomer
Astrida	3
Betacab	2
Canutility	0
Divadip	2

NumericCount - diagramfunktion

NumericCount() aggregerar antalet numeriska värden i varje diagramdimension.

Syntax:

```
NumericCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
set_ expression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel och resultat:

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	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

I följande exempel antas att alla kunder är valda, utom där det särskilt anges.

Exempel och resultat

Exempel	Resultat
`NumericCount ([OrderNumber])`	7 eftersom tre av de 10 fälten i OrderNumber är tomma. *"0" räknas som ett värde och inte som en tom cell. Om ett värde aggregeras till 0 för en dimension kommer den dimensionen dock inte att tas med i diagram.*
`NumericCount ([Product])`	0 eftersom alla produktnamn skrivs med bokstäver. Du kan använda detta för att kontrollera att inga textfält har fått numeriskt innehåll.
`NumericCount (DISTINCT [OrderNumber])/Count (DISTINCT [OrderNumber)]`	Räknar det fullständiga antalet distinkta numeriska ordernummer och delar dem med antalet numeriska och icke-numeriska ordernummer. Detta blir 1 om alla fältvärden är numeriska. Du kan använda detta för att kontrollera att alla fältvärden är numeriska. I exemplet finns det 7 distinkta numeriska värden för OrderNumber av 8 distinkta numeriska och icke numeriska, så att uttrycket returnerar 0,875.

Data som används i exemplet:

```
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() returnerar summan av fältvärden som är icke-numeriska aggregerade i uttrycket, som definieras av en **group by**-sats.

Syntax:

```
TextCount ( [ distinct ] expr)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
expr Expression	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket ignoreras alla dubbletter.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår app för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Exempel:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
TextCount1:
LOAD Customer,TextCount(Product) as ProductTextCount Resident Temp Group By Customer;
```

Resultattabell

Customer	ProductTextCount
Astrida	3
Betacab	3
Canutility	2
Divadip	2

Exempel:

```
LOAD Customer,TextCount(OrderNumber) as OrderNumberTextCount Resident Temp Group By Customer;
```

Resultattabell

Customer	OrderNumberTextCount
Astrida	0
Betacab	1
Canutility	2
Divadip	0

TextCount - diagramfunktion

TextCount() används för att aggregera antalet fältvärden som är icke-numeriska i varje diagramdimension.

Syntax:

```
TextCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.

Argument	Beskrivning
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel och resultat:

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

Exempel och resultat

Exempel	Resultat
`TextCount ([Product])`	10, eftersom alla 10 fält i Product är text. *"0" räknas som ett värde och inte som en tom cell. Om ett värde aggregeras till 0 för en dimension kommer den dimensionen dock inte att tas med i diagram. Tomma celler utvärderas som icke-text och räknas inte av TextCount.*
`TextCount ([OrderNumber])`	3 eftersom tomma celler räknas. Du kan använda detta för att kontrollera att inga numeriska fält har fått textinnehåll eller är icke-noll.

Exempel	Resultat
`TextCount (DISTINCT [Product])/Count ([Product)]`	Beräknar hela antalet distinkta textvärden för Product (4) och delar det med det totala antalet värden i Product (10). Resultatet är 0,4.

Data som används i exemplet:

```
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 '|');
```

Finansiella aggregeringsfunktioner

Den här delen beskriver aggregeringsfunktioner för finansiella operationer vad gäller betalning och kassaflöde.

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

Finansiella aggregeringsfunktioner i dataladdningsskriptet

IRR

IRR() returnerar den aggregerade internräntan för en serie kassaflöden, representerade av talen i uttryck itererade över ett antal poster enligt vad som definierats i en group by-sats.

```
IRR (expression)
```

XIRR

XIRR() returnerar den aggregerade internräntan för en tablå av kassaflöden (som inte nödvändigtvis är periodisk), representerad av parvisa tal i **pmt** och **date** itererade över ett antal poster enligt vad som definierats i en group by-sats. Alla betalningar diskonteras utifrån ett 365-dagarsår.

```
XIRR (valueexpression, dateexpression )
```

NPV

NPV returnerar det aggregerade avkastningsvärdet för en investering baserad på en **discount_rate**och en serie framtida betalningar (negativa värden) och inkomster (positiva värden), representerade av talen i **value**, itererade över ett antal poster enligt vad som definierats i en group by-sats. Betalningar och

inkomster förväntas i slutet av varje period.

```
NPV (rate, expression)
```

XNPV

XNPV() returnerar det aggregerade aktuella nettovärdet för en tablå av kassaflöden (som inte nödvändigtvis är periodisk), representerade av parvisa tal i **pmt** och **date**, itererade över ett antal poster enligt vad som definierats i en group by-sats. Ränta är räntesatsen per period. Alla betalningar diskonteras utifrån ett 365-dagarsår.

```
XNPV (rate, valueexpression, dateexpression)
```

Finansiella aggregeringsfunktioner i diagramuttryck

Dessa finansiella aggregeringsfunktioner kan användas i diagram.

IRR

IRR() returnerar den aggregerade avkastningsgraden för en serie kassaflöden som motsvaras av talen i det uttryck som ges av **value** itererat över diagrammets dimensioner.

```
 IRR - diagramfunktion[TOTAL [<fld {,fld}>]] value)
```

NPV

NPV() returnerar det aggregerade aktuella nettovärdet för en investering baserat på **discount_rate** per period och en serie framtida betalningar (negativa värden) och inkomster (positiva värden). Dessa motsvaras av talen i **value** itererat över diagrammets dimensioner. Betalningar och inkomster förväntas i slutet av varje period.

```
NPV - diagramfunktion([TOTAL [<fld {,fld}>]] discount_rate, value)
```

XIRR

XIRR() returnerar den aggregerade avkastningsgraden för ett kassaflödesschema (inte nödvändigtvis periodiskt) motsvarat av parvisa tal i de uttryck som ges av **pmt** och **date** itererat över diagrammets dimensioner. Alla betalningar diskonteras utifrån ett 365-dagarsår.

```
XIRR - diagramfunktion (page 255)([TOTAL [<fld {,fld}>]] pmt, date)
```

XNPV

XNPV() returnerar det aktuella aggregerade nettovärdet för ett kassaflödesschema (inte nödvändigtvis periodiskt) motsvarat av parvisa tal i de uttryck som ges av **pmt** och **date** itererat över diagrammets dimensioner. Alla betalningar diskonteras utifrån ett 365-dagarsår.

```
XNPV - diagramfunktion([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

IRR

IRR() returnerar den aggregerade internräntan för en serie kassaflöden, representerade av talen i uttryck itererade över ett antal poster enligt vad som definierats i en group by-sats.

Dessa kassaflöden måste inte vara jämna vilket krävs för annuiteter. Kassaflödena måste dock infalla med jämna intervall (som månatligen eller årligen). Internräntan är räntesatsen som mottagits för en investering som består av betalningar (negativa värden) och inkomster (positiva värden) som infaller regelbundet. Funktionen kräver minst ett positivt och ett negativt värde för att beräknas.

Syntax:

```
IRR(value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Det uttryck eller fält som innehåller de data som ska mätas.

Begränsningar:

Textvärden, NULL-värden samt saknade värden ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Exempel och resultat:

Exempel och resultat

Exempel	År	IRR2013
Cashflow: LOAD 2013 as Year, * inline [Date|Discount|Payments 2013-01-01|0.1|-10000 2013-03-01|0.1|3000 2013-10-30|0.1|4200 2014-02-01|0.2|6800] (delimiter is '|'); Cashflow1: LOAD Year,IRR(Payments) as IRR2013 Resident Cashflow Group By Year;	2013	0.1634

IRR - diagramfunktion

IRR() returnerar den aggregerade avkastningsgraden för en serie kassaflöden som motsvaras av talen i det uttryck som ges av **value** itererat över diagrammets dimensioner.

Dessa kassaflöden måste inte vara jämna vilket krävs för annuiteter. Kassaflödena måste dock infalla med jämna intervall (som månatligen eller årligen). Internränta är den räntesats som ges vid en investering i form av betalning (negativa värden) och inkomst (positiva värden) som infaller regelbundet. För att funktionen ska kunna beräknas krävs minst ett negativt och ett positivt värde.

Syntax:

```
IRR([TOTAL [<fld {,fld}>]] value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Det uttryck eller fält som innehåller de data som ska mätas.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden samt saknade värden ignoreras.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`IRR (Payments)`	0.1634 Betalningarna antas vara periodiska till sin natur, exempelvis månatliga. *Datum-fältet används i XIRR-exemplet där betalningarna kan vara icke-periodiska så länge du uppger de datum då betalningarna gjordes.*

Data som används i exempel:

```
Cashflow:
LOAD 2013 as Year, * inline [
```

```
Date|Discount|Payments
2013-01-01|0.1|-10000
2013-03-01|0.1|3000
2013-10-30|0.1|4200
2014-02-01|0.2|6800
] (delimiter is '|');
```

Se även:

- *XIRR - diagramfunktion (page 255)*
- *Aggr - diagramfunktion (page 404)*

NPV

NPV returnerar det aggregerade avkastningsvärdet för en investering baserad på en **discount_rate**och en serie framtida betalningar (negativa värden) och inkomster (positiva värden), representerade av talen i **value**, itererade över ett antal poster enligt vad som definierats i en group by-sats. Betalningar och inkomster förväntas i slutet av varje period.

Syntax:

```
NPV(discount_rate, value)
```

Returnerad datatyp: numeriska. Resultatet anges i ett fördefinierat talformat (valuta).

Argument:

Argument

Argument	Beskrivning
discount_rate	**discount_rate** är räntan över periodens längd.
value	Det uttryck eller fält som innehåller de data som ska mätas.

Begränsningar:

Textvärden, NULL-värden samt saknade värden ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Exempel och resultat

Exempel	År	NPV1_2013
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,NPV(0.2, Payments) as NPV1_2013 Resident Cashflow Group By Year;	2013	-$540.12

Exempel och resultat

Exempel	År	Rabatt	NPV2_2013
Givet att **Cashflow**-tabellen är laddad som i föregående exempel: LOAD Year,NPV(Discount, Payments) as NPV2_2013 Resident Cashflow Group By Year, Discount; Observera att satsen Group By sorterar resultaten efter Year ochDiscount. Det första argumentet, discount_rate, visas som ett fält (Discount), snarare än ett specifikt tal och därför krävs ett andra sorteringsvillkor. Ett fält kan innehålla olika värden. Därför måste de aggregerade posterna sorteras för att möjliggöra för olika värden för Year och Discount. ;	2013 2013	0.1 0.2	-$3456.05 $5666.67

NPV - diagramfunktion

NPV() returnerar det aggregerade aktuella nettovärdet för en investering baserat på **discount_rate** per period och en serie framtida betalningar (negativa värden) och inkomster (positiva värden). Dessa motsvaras av talen i **value** itererat över diagrammets dimensioner. Betalningar och inkomster förväntas i slutet av varje period.

Syntax:

```
NPV([TOTAL [<fld {,fld}>]] discount_rate, value)
```

Returnerad datatyp: numeriska Resultatet anges i ett fördefinierat talformat (valuta).

Argument:

Argument

Argument	Beskrivning
discount_rate	**discount_rate** är räntan över periodens längd.
value	Det uttryck eller fält som innehåller de data som ska mätas.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena. Bestämningen **TOTAL** kan följas av en lista med ett eller flera fältnamn inom vinkelparenteser. Dessa fältnamn bör ingå i en underuppsättning av diagrammets dimensionsvariabler. Vid uträkningen beaktas i detta fall alla diagrammets dimensionsvariabler utom de som ingår i listan, d.v.s. ett värde returneras för varje kombination av fältvärden i dimensionsfälten som finns i listan. Även fält som för närvarande inte utgör en dimension i ett diagram kan ingå i listan. Detta är användbart när man arbetar med grupper av dimensioner där dimensionsfälten inte är fasta. Genom att lista alla variabler i gruppen säkerställs att funktionen fungerar när den hierarkiska nivån ändras.

Begränsningar:

discount_rate och **value** får inte innehålla aggregeringsfunktioner, såvida inte dessa inre aggregeringar innehåller kvalificeraren **TOTAL**. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden samt saknade värden ignoreras.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`NPV(Discount, Payments)`	-$540.12

Data som används i exempel:

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Discount|Payments
```

```
2013-01-01|0.1|-10000
2013-03-01|0.1|3000
2013-10-30|0.1|4200
2014-02-01|0.2|6800
] (delimiter is '|');
```

Se även:

- *XNPV - diagramfunktion (page 257)*
- *Aggr - diagramfunktion (page 404)*

XIRR

XIRR() returnerar den aggregerade internräntan för en tablå av kassaflöden (som inte nödvändigtvis är periodisk), representerad av parvisa tal i **pmt** och **date** itererade över ett antal poster enligt vad som definierats i en group by-sats. Alla betalningar diskonteras utifrån ett 365-dagarsår.

Syntax:

```
XIRR(pmt, date )
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
pmt	Betalningar. Uttrycket eller fältet som innehåller de kassaflöden som motsvarar det betalningsschema som anges i **date**.
date	Uttrycket eller fältet som innehåller de datascheman som motsvarar kassaflödesbetalningar som ges i **pmt**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Exempel och resultat

Exempel	År	XIRR2013
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,XIRR(Payments, Date) as XIRR2013 Resident Cashflow Group By Year;	2013	0.5385

XIRR - diagramfunktion

XIRR() returnerar den aggregerade avkastningsgraden för ett kassaflödesschema (inte nödvändigtvis periodiskt) motsvarat av parvisa tal i de uttryck som ges av **pmt** och **date** itererat över diagrammets dimensioner. Alla betalningar diskonteras utifrån ett 365-dagarsår.

Syntax:

```
XIRR([TOTAL [<fld {,fld}>]] pmt, date)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
pmt	Betalningar. Uttrycket eller fältet som innehåller de kassaflöden som motsvarar det betalningsschema som anges i **date**.
date	Uttrycket eller fältet som innehåller de datascheman som motsvarar kassaflödesbetalningar som ges i **pmt**.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

pmt och **date** får inte innehålla aggregeringsfunktioner, såvida inte dessa inre aggregeringar innehåller kvalificeraren **TOTAL**. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`XIRR(Payments, Date)`	0.5385

Data som används i exempel:

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Discount|Payments
2013-01-01|0.1|-10000
2013-03-01|0.1|3000
2013-10-30|0.1|4200
2014-02-01|0.2|6800
] (delimiter is '|');
```

Se även:

IRR - diagramfunktion (page 249)
Aggr - diagramfunktion (page 404)

XNPV

XNPV() returnerar det aggregerade aktuella nettovärdet för en tablå av kassaflöden (som inte nödvändigtvis är periodisk), representerade av parvisa tal i **pmt** och **date**, itererade över ett antal poster enligt vad som definierats i en group by-sats. Ränta är räntesatsen per period. Alla betalningar diskonteras utifrån ett 365-dagarsår.

Syntax:

```
XNPV(discount_rate, pmt, date)
```

Returnerad datatyp: numeriska. Resultatet anges i ett fördefinierat talformat (valuta). .

Argument:

Argument

Argument	Beskrivning
discount_rate	**discount_rate** är räntan över periodens längd.
pmt	Det uttryck eller fält som innehåller de data som ska mätas.
date	Uttrycket eller fältet som innehåller de datascheman som motsvarar kassaflödesbetalningar som ges i **pmt**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Exempel och resultat

Exempel	År	XNPV1_2013

`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,XNPV(0.2, Payments, Date) as XNPV1_2013 Resident`
`Cashflow Group By Year;`
	2013	$2104.37

Exempel och resultat

Exempel	År	Rabatt	XNPV2_2013
Givet att **Cashflow**-tabellen är laddad som i föregående exempel: `LOAD Year,XNPV(Discount, Payments, Date) as XNPV2_2013 Resident Cashflow Group By Year, Discount;` Observera att satsen Group By sorterar resultaten efter Year ochDiscount. Det första argumentet, discount_rate, visas som ett fält (Discount), snarare än ett specifikt tal och därför krävs ett andra sorteringsvillkor. Ett fält kan innehålla olika värden. Därför måste de aggregerade posterna sorteras för att möjliggöra för olika värden för Year och Discount.	2013 2013	0.1 0.2	-$3164.35 $6800.00

XNPV - diagramfunktion

XNPV() returnerar det aktuella aggregerade nettovärdet för ett kassaflödesschema (inte nödvändigtvis periodiskt) motsvarat av parvisa tal i de uttryck som ges av **pmt** och **date** itererat över diagrammets dimensioner. Alla betalningar diskonteras utifrån ett 365-dagarsår.

Syntax:

```
XNPV([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

Returnerad datatyp: numeriska Resultatet anges i ett fördefinierat talformat (valuta).

Argument:

Argument

Argument	Beskrivning
discount_rate	**discount_rate** är räntan över periodens längd.
pmt	Betalningar. Uttrycket eller fältet som innehåller de kassaflöden som motsvarar det betalningsschema som anges i **date**.
date	Uttrycket eller fältet som innehåller de datascheman som motsvarar kassaflödesbetalningar som ges i **pmt**.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

discount_rate,**pmt** och **date** får inte innehålla aggregeringsfunktioner, såvida inte dessa inre aggregeringar innehåller kvalificerarna **TOTAL** eller **ALL**. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`XNPV(Discount, Payments, Date)`	-$3164.35

Data som används i exempel:

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Discount|Payments
2013-01-01|0.1|-10000
2013-03-01|0.1|3000
2013-10-30|0.1|4200
2014-02-01|0.2|6800
] (delimiter is '|');
```

Se även:

- *NPV - diagramfunktion (page 252)*
- *Aggr - diagramfunktion (page 404)*

Statistiska aggregeringsfunktioner

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

Statistiska aggregeringsfunktioner i dataladdningsskriptet

Följande statistiska aggregeringsfunktioner kan användas i skript.

Avg

Avg() hittar medelvärdet för aggregerade data i uttrycket över ett antal poster som definieras av en **group by**-sats.

```
Avg  ([distinct] expression)
```

Correl

Correl() returnerar den aggregerade korrelationskoefficienten för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definierats i en **group by**-sats.

```
Correl (x-expression, y-expression)
```

Fractile

Fractile() hittar värdet som motsvarar den inklusiva fraktilen (kvantilen) för aggregerade data i uttrycket över ett antal poster som definieras av en **group by**-sats.

```
Fractile (expression, fractile)
```

FractileExc

FractileExc() hittar värdet som motsvarar den exklusiva fraktilen (kvantilen) för aggregerade data i uttrycket över ett antal poster som definieras av en **group by**-sats.

```
FractileExc (expression, fractile)
```

Kurtosis

Kurtosis() returnerar graden av kurtosis av data i uttrycket över ett antal poster som definieras av en **group by**-sats.

```
Kurtosis  ([distinct ] expression )
```

LINEST_B

LINEST_B() returnerar det aggregerade b-värdet (y-intercept) hos en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras i en **group by**-sats.

```
LINEST_B (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_df

LINEST_DF() returnerar den aggregerade frihetsgraden hos en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras i en**group by**-sats.

```
LINEST_DF  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_f

Denna skriptfunktion returnerar det aggregerade F-värdet ($r^2/(1-r^2)$) hos en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras av en **group by**-sats.

```
LINEST_F (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_m

LINEST_M() returnerar det aggregerade m-värdet (lutning) hos en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras i en **group by**-sats.

```
LINEST_M  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_r2

LINEST_R2() returnerar det aggregerade r^2-värdet (bestämningskoefficienten) av en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som representeras av parvisa tal i x-expression och y-expression itererat över ett antal poster som definieras av en **group by**-sats.

```
LINEST_R2 (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_seb

LINEST_SEB() returnerar det aggregerade standardfelet hos b-värdet i en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt definitionen av en **group by**-sats.

```
LINEST_SEB  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sem

LINEST_SEM() returnerar det aggregerade standardfelet hos m-värdet i en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt definitionen av en **group by**-sats.

```
LINEST_SEM  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sey

LINEST_SEY() returnerar det aggregerade standardfelet hos y-uppskattningen i en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt definitionen av en **group by**-sats.

```
LINEST_SEY  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_ssreg

LINEST_SSREG returnerar den aggregerade restsumman av en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som representeras av parvisa tal i x-expression och y-expression itererat över ett antal poster som definieras av en **group by**-sats.

```
LINEST_SSREG  (y-expression, x-expression [, y0 [, x0 ]])
```

Linest_ssresid

LINEST_SSRESID() returnerar den aggregerade restsumman av en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som representeras av parvisa tal i x-expression och y-expression itererat över ett antal poster som definieras av en **group by**-sats.

```
LINEST_SSRESID  (y-expression, x-expression [, y0 [, x0 ]])
```

Median

Median() returnerar den aggregerade medianen av värdena i uttrycket över ett antal poster som definieras av en **group by**-sats.

```
Median (expression)
```

Skew

Skew() returnerar skevheten i uttryck över ett antal poster enligt vad som definierats i en **group by**-sats.

```
Skew ([ distinct] expression)
```

Stdev

Stdev() returnerar värdenas standardavvikelse i uttrycket över ett antal poster som definieras av en **group by**-sats.

```
Stdev ([distinct] expression)
```

Sterr

Sterr() returnerar det aggregerade standardfelet (stdev/sqrt(n)) för en serie värden, representerade av uttryck över ett antal poster enligt vad som definierats i en **group by**-sats.

```
Sterr  ([distinct] expression)
```

STEYX

STEYX() returnerar det aggregerade standardfelet hos det predikterade y-värdet för varje x-värde i regressionen för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definierats i en **group by**-sats.

```
STEYX  (y-expression, x-expression)
```

Statistiska aggregeringsfunktioner i diagramuttryck

Följande statistiska aggregeringsfunktioner kan användas i diagram.

Avg

Avg() returnerar det aggregerade medelvärdet av uttrycket eller fältet itererat över diagrammets dimensioner.

```
Avg - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} expr)
```

Correl

Correl() returnerar den aggregerade korrelationskoefficienten för två datauppsättningar. Korrelationsfunktionen är ett mått på förhållandet mellan datauppsättningarna och aggregeras för (x,y)-värdeparen itererat över diagrammets dimensioner.

```
Correl - diagramfunktion({[SetExpression] [TOTAL [<fld {, fld}>]]} value1, value2 )
```

Fractile

Fractile() finner det värde som motsvarar den inklusiva fraktilen (kvantilen) av aggregerade data i det intervall som ges av uttrycket itererat över diagrammets dimensioner.

```
Fractile - diagramfunktion({[SetExpression] [TOTAL [<fld {, fld}>]]} expr, fraction)
```

FractileExc

FractileExc() finner det värde som motsvarar den exklusiva fraktilen (kvantilen) av aggregerade data i det intervall som ges av uttrycket itererat över diagrammets dimensioner.

```
FractileExc - diagramfunktion({[SetExpression] [TOTAL [<fld {, fld}>]]} expr, fraction)
```

Kurtosis

Kurtosis() finner kurtosis av dataintervallet aggregerat i uttrycket eller fältet itererat över diagrammets dimensioner.

```
Kurtosis - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} expr)
```

LINEST_b

LINEST_B() returnerar det aggregerade b-värdet (y-intercept) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_R2 - diagramfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_value, x_value[, y0_const[, x0_const]])
```

LINEST_df

LINEST_DF() returnerar de aggregerade frihetsgraderna för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_DF - diagramfunktion({[SetExpression] [TOTAL [<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_f

LINEST_F() returnerar det aggregerade F-värdet (r2/(1-r2)) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_F - diagramfunktion({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_m

LINEST_M() returnerar det aggregerade m-värdet (lutningen) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_M - diagramfunktion({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_r2

LINEST_R2() returnerar det aggregerade r2-värdet (bestämningskoefficienten) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_R2 - diagramfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_value,
x_value[, y0_const[, x0_const]])
```

LINEST_seb

LINEST_SEB() returnerar det aggregerade standardfelet för b-värdet av en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_SEB - diagramfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_sem

LINEST_SEM() returnerar det aggregerade standardfelet för m-värdet av en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_SEM - diagramfunktion([{set_expression}][ distinct ] [total [<fld
{,fld}>] ] y-expression, x-expression [, y0 [, x0 ]] )
```

LINEST_sey

LINEST_SEY() returnerar det aggregerade standardfelet för y-estimatet av en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_SEY - diagramfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssreg

LINEST_SSREG() returnerar den aggregerade regressionskvadratsumman av en linjär regression definierad genom ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_SSREG - diagramfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssresid

LINEST_SSRESID() returnerar den aggregerade residualkvadratsumman hos en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **x_value** och **y_value**, itererat över diagrammets dimensioner.

```
LINEST_SSRESID - diagramfunktionLINEST_SSRESID() returnerar den aggregerade
residualkvadratsumman hos en linjär regression som den definieras av
ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de
uttryck som ges av x_value och y_value, itererat över diagrammets
dimensioner.  LINEST_SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{,
fld}>]] y_value, x_value[, y0_const[, x0_const]])
numeriska ArgumentArgumentBeskrivningy_valueUttrycket eller fältet som
innehåller y-värdeintervallet som ska mätas.x_valueUttrycket eller fältet som
innehåller x-värdeintervallet som ska mätas.y0, x0Ett valfritt värde y0 kan
anges för att forcera regressionslinjen att passera y-axeln vid en bestämd
punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en
viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst
två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits,
räcker det med ett enda datapar. SetExpressionSom standard kommer
aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som
definierats av urvalet. En alternativ uppsättning poster kan definieras med
ett uttryck för set-analys. DISTINCTOm predikatet DISTINCT förekommer framför
funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån
funktionsargumenten. TOTALOm ordet TOTAL står före funktionsargumenten görs
beräkningen över alla valbara värden givet de aktuella valen, och inte bara
sådana som rör det aktuella dimensionsvärdet, det vill säga att
diagramdimensionerna ignoreras. Genom att använda TOTAL [<fld {.fld}>], där
kvalificeraren TOTAL följs av en lista med ett eller flera fältnamn som en
delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala
möjliga värdena.Ett valfritt värde y0 kan anges för att forcera
regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som
x0 anges kan regressionlinjen forceras att passera en viss koordinat.
```

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller TOTAL-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade Aggr-funktionen i kombination med en specificerad dimension. Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras. An example of how to use linest functionsavg({[SetExpression] [**TOTAL** [<fld{ ,fld}>]] }y_value, x_value [, y0_const[, x0_const]]**)**

Median

Median() returnerar medianvärdet för värdeintervallet som aggregeras i uttrycket, itererat över diagrammets dimensioner.

Median - diagramfunktion({[SetExpression] [**TOTAL** [<fld{, fld}>]]} expr**)**

MutualInfo

MutualInfo beräknar ömsesidig information (MI) mellan två fält eller mellan aggregerade värden i **Aggr()**.

MutualInfo - diagramfunktion (page 305){[SetExpression] [**DISTINCT**] [**TOTAL** target, driver [, datatype [, breakdownbyvalue [, samplesize]]]**)**

Skew

Skew() returnerar den aggregerade skevheten av uttrycket eller fältet itererat över diagrammets dimensioner.

Skew - diagramfunktion{[SetExpression] [**DISTINCT**] [**TOTAL** [<fld{ ,fld}>]]} expr**)**

Stdev

Stdev() finner standardavvikelsen för dataintervallet som aggregerats i uttrycket eller fältet itererat över diagrammets dimensioner.

Stdev - diagramfunktion({[SetExpression] [**DISTINCT**] [**TOTAL** [<fld{, fld}>]]} expr**)**

Sterr

Sterr() finner värdet av standardfelet av medelvärdet (stdev/sqrt(n)) för värdeserien som aggregerats i uttrycket itererat över diagrammets dimensioner.

Sterr - diagramfunktion({[SetExpression] [**DISTINCT**] [**TOTAL**[<fld{, fld}>]]} expr**)**

STEYX

STEYX() returnerar det aggregerade standardfelet vid prediktering av y-värden för varje x-värde i en linjär regression som ges av en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **y_value** och **x_value**.

STEYX - diagramfunktion{[SetExpression] [**TOTAL** [<fld{, fld}>]]} y_value, x_value**)**

Avg

Avg() hittar medelvärdet för aggregerade data i uttrycket över ett antal poster som definieras av en **group by**-sats.

Syntax:

```
Avg([DISTINCT] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
DISTINCT	Om predikatet **distinct** förekommer framför uttrycket kommer alla dubbletter att ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Resulterande data

Exempel	Resultat

`Temp:`
`crosstable (Month, Sales) load * inline [`
`Customer|Jan|Feb|Mar||Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec`
`Astrida|46|60|70|13|78|20|45|65|78|12|78|22`
`Betacab|65|56|22|79|12|56|45|24|32|78|55|15`
`Canutility|77|68|34|91|24|68|57|36|44|90|67|27`
`Divadip|36|44|90|67|27|57|68|47|90|80|94`
`] (delimiter is '|');`
 	
`Avg1:`
`LOAD Customer, Avg(Sales) as MyAverageSalesByCustomer`
`Resident Temp Group By Customer;`
	`Customer`
	`MyAverageSalesByCustomer`
	`Astrida 48.916667`
	`Betacab 44.916667`
	`Canutility 56.916667`
	`Divadip 63.083333`
	Detta kan kontrolleras på arket genom att skapa en tabell som innehåller måttet:
	`Sum(Sales)/12`

Givet att **Temp**-tabellen är laddad som i föregående exempel:
 	
`LOAD Customer,Avg(DISTINCT Sales) as MyAvgSalesDistinct`
`Resident Temp Group By Customer;`
	`Customer`
	`MyAverageSalesByCustomer`
	`Astrida 43.1`
	`Betacab 43.909091`
	`Canutility 55.909091`
	`Divadip 61`
	Enbart distinkta värden räknas. Dela summan med antalet värden som inte är dubbletter.

Avg - diagramfunktion

Avg() returnerar det aggregerade medelvärdet av uttrycket eller fältet itererat över diagrammets dimensioner.

Syntax:

```
Avg([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Exempel och resultat:

Example table

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Astrida	46	60	70	13	78	20	45	65	78	12	78	22
Betacab	65	56	22	79	12	56	45	24	32	78	55	15
Canutility	77	68	34	91	24	68	57	36	44	90	67	27
Divadip	57	36	44	90	67	27	57	68	47	90	80	94

Exempel på funktioner

Exempel	Resultat
`Avg(Sales)`	För en tabell som innehåller dimensionen `Customer` och måttet `Avg([Sales])`, är resultatet 2566 om **Totalvärden** visas.
`Avg([TOTAL (Sales))`	53.458333 för alla värden för `Customer`, eftersom TOTAL-kvalificeraren innebär att dimensionerna ignoreras.
`Avg(DISTINCT (Sales))`	51.862069 som totalvärde, eftersom Distinct-kvalificeraren innebär att endast unika värden i `Sales` för varje `Customer` utvärderas.

Data som används i exempel:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Se även:

Aggr - diagramfunktion (page 404)

Correl

Correl() returnerar den aggregerade korrelationskoefficienten för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definierats i en **group by**-sats.

Syntax:

```
Correl(value1, value2)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value1, value2	Uttrycken eller fälten som innehåller de två stickprovsuppsättningarna för vilka korrelationskoefficienten ska mätas.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Resulterande data

Exempel	Resultat

`Salary:`
`Load *, 1 as Grp;`
`LOAD * inline [`
`"Employee name"|Gender|Age|Salary`
`Aiden Charles|Male|20|25000`
`Brenda Davies|Male|25|32000`
`Charlotte Edberg|Female|45|56000`
`Daroush Ferrara|Male|31|29000`
`Eunice Goldblum|Female|31|32000`
`Freddy Halvorsen|Male|25|26000`
`Gauri Indu|Female|36|46000`
`Harry Jones|Male|38|40000`
`Ian Underwood|Male|40|45000`
`Jackie Kingsley|Female|23|28000`
`] (delimiter is '|');`
 	
`Correl1:`
`LOAD Grp,`
`Correl(Age,Salary) as Correl_`
`Salary Resident Salary Group By`
`Grp;`
	I en tabell med dimensionen `Correl_Salary` visas resultatet av Correl()-beräkningen i dataladdningsskriptet: 0.9270611

Correl - diagramfunktion

Correl() returnerar den aggregerade korrelationskoefficienten för två datauppsättningar. Korrelationsfunktionen är ett mått på förhållandet mellan datauppsättningarna och aggregeras för (x,y)-värdeparen itererat över diagrammets dimensioner.

Syntax:

```
Correl([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] value1, value2 )
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value1, value2	Uttrycken eller fälten som innehåller de två stickprovsuppsättningarna för vilka korrelationskoefficienten ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`Correl(Age, Salary)`	För en tabell som innehåller dimensionen `Employee name` och måttet `Correl(Age, Salary)` är resultatet 0,9270611. Resultaten visas endast för totalcellen.

Exempel	Resultat
`Correl(TOTAL Age, Salary))`	0.927. Detta och följande resultat visas med tre decimaler för bättre läsbarhet. Om du skapar en filterruta med dimensionen Gender och gör ett val från den, ser du resultatet 0,951 när Female väljs och 0,939 om Male väljs. Detta beror på att valet utesluter alla resultat som inte hör till det andra värdet för Gender.
`Correl({1} TOTAL Age, Salary))`	0.927. Oberoende av val. Detta beror på att set-uttrycket {1} bortser från alla urval och dimensioner.
`Correl(TOTAL <Gender> Age, Salary))`	0,927 i totalcellen, 0,939 för alla värden för Male och 0,951 för alla värden för Female. Det här motsvarar resultaten om du gör valen i en filterruta baserad på Gender.

Data som används i exempel:

```
Salary:
LOAD * inline [
"Employee name"|Gender|Age|Salary
Aiden Charles|Male|20|25000
Brenda Davies|Male|25|32000
Charlotte Edberg|Female|45|56000
Daroush Ferrara|Male|31|29000
Eunice Goldblum|Female|31|32000
Freddy Halvorsen|Male|25|26000
Gauri Indu|Female|36|46000
Harry Jones|Male|38|40000
Ian Underwood|Male|40|45000
Jackie Kingsley|Female|23|28000
] (delimiter is '|');
```

Se även:

- *Aggr - diagramfunktion (page 404)*
- *Avg - diagramfunktion (page 267)*
- *RangeCorrel (page 698)*

Fractile

Fractile() hittar värdet som motsvarar den inklusiva fraktilen (kvantilen) för aggregerade data i uttrycket över ett antal poster som definieras av en **group by**-sats.

Du kan använda FractileExc (page 275) för att beräkna den exklusiva fraktilen.

Syntax:

```
Fractile(expr, fraction)
```

Returnerad datatyp: numeriska

Funktionen returnerar värdet som motsvarar rangordningen enligt definitionen `rangordning = fraktion * (N-1) + 1` där `N` är antalet värden i `expr`. Om `rangordningen` inte är ett heltal görs en interpolering mellan de två närmaste värdena.

Argument:

Argument

Argument	Beskrivning
expr	Uttrycket eller fältet som innehåller de data som ska användas för beräkning av fraktilen.
fraction	Ett tal mellan 0 och 1 som motsvarar den fraktil (kvantil uttryckt som bråkdel) som ska beräknas.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Resulterande data

Exempel	Resultat

`Table1:`
`crosstable LOAD recno() as ID, *`
`inline [`
`Observation|Comparison`
`35|2`
`40|27`
`12|38`
`15|31`
`21|1`
`14|19`
`46|1`
`10|34`
`28|3`
`48|1`
`16|2`
`30|3`
`32|2`
`48|1`
`31|2`
`22|1`
`12|3`
`39|29`
`19|37`
`25|2 ] (delimiter is '|');`
 	
`Fractile1:`
`LOAD Type,`
`Fractile(Value,0.75) as MyFractile`
`Resident Table1 Group By Type;`
	I en tabell med dimensionerna `Type` och `MyFractile` blir resultaten av Fractile()-beräkningarna i dataladdningsskriptet:
	
	`Type MyFractile`
	`Comparison 27.5`
	`Observation 36`

Fractile - diagramfunktion

Fractile() finner det värde som motsvarar den inklusiva fraktilen (kvantilen) av aggregerade data i det intervall som ges av uttrycket itererat över diagrammets dimensioner.

Du kan använda FractileExc - diagramfunktion (page 276) för att beräkna den exklusiva fraktilen.

Syntax:

```
Fractile([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr, fraction)
```

Returnerad datatyp: numeriska

Funktionen returnerar värdet som motsvarar rangordningen enligt definitionen `rangordning = fraktion * (N-1) + 1` där `N` är antalet värden i `expr`. Om `rangordningen` inte är ett heltal görs en interpolering mellan de två närmaste värdena.

Argument:

Argument

Argument	Beskrivning
expr	Uttrycket eller fältet som innehåller de data som ska användas för beräkning av fraktilen.
fraction	Ett tal mellan 0 och 1 som motsvarar den fraktil (kvantil uttryckt som bråkdel) som ska beräknas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {,fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Exempel och resultat:

Example table

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Astrida	46	60	70	13	78	20	45	65	78	12	78	22
Betacab	65	56	22	79	12	56	45	24	32	78	55	15
Canutility	77	68	34	91	24	68	57	36	44	90	67	27
Divadip	57	36	44	90	67	27	57	68	47	90	80	94

Exempel på funktioner

Exempel	Resultat
`Fractile (Sales, 0.75)`	För en tabell som innehåller dimensionen `Customer` och måttet `Fractile([Sales])`, är resultatet 71,75 om **Totalvärden** visas. Detta är punkten i värdefördelningen för `Sales` som 75 % av värdena faller under.
`Fractile (TOTAL Sales, 0.75))`	71.75 för alla värden för `Customer`, eftersom TOTAL-kvalificeraren innebär att dimensionerna ignoreras.
`Fractile (DISTINCT Sales, 0.75)`	70 som totalvärde, eftersom DISTINCT-kvalificeraren innebär att endast unika värden i `Sales` för varje `Customer` utvärderas.

Data som används i exempel:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
```

```
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Se även:

- *Aggr - diagramfunktion (page 404)*

FractileExc

FractileExc() hittar värdet som motsvarar den exklusiva fraktilen (kvantilen) för aggregerade data i uttrycket över ett antal poster som definieras av en **group by**-sats.

Du kan använda Fractile (page 271) för att beräkna den inklusiva fraktilen.

Syntax:

```
FractileExc(expr, fraction)
```

Returnerad datatyp: numeriska

Funktionen returnerar värdet som motsvarar rangordningen enligt definitionen `rangordning = fraktion * (N+1)` där `N` är antalet värden i `expr`. Om `rangordningen` inte är ett heltal görs en interpolering mellan de två närmaste värdena.

Argument:

Argument

Argument	Beskrivning
expr	Uttrycket eller fältet som innehåller de data som ska användas för beräkning av fraktilen.
fraction	Ett tal mellan 0 och 1 som motsvarar den fraktil (kvantil uttryckt som bråkdel) som ska beräknas.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Resulterande data

Exempel	Resultat
Table1: crosstable LOAD recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|'); Fractile1: LOAD Type, FractileExc(Value,0.75) as MyFractile Resident Table1 Group By Type;	I en tabell med dimensionerna `Type` och `MyFractile` blir resultaten av FractileExc()-beräkningarna i dataladdningsskriptet: Type MyFractile Comparison 28.5 Observation 38

FractileExc - diagramfunktion

FractileExc() finner det värde som motsvarar den exklusiva fraktilen (kvantilen) av aggregerade data i det intervall som ges av uttrycket itererat över diagrammets dimensioner.

Du kan använda Fractile - diagramfunktion (page 273) för att beräkna den inklusiva fraktilen.

Syntax:

```
FractileExc([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr,
fraction)
```

Returnerad datatyp: numeriska

Funktionen returnerar värdet som motsvarar rangordningen enligt definitionen `rangordning = fraktion * (N+1)` där `N` är antalet värden i `expr`. Om `rangordningen` inte är ett heltal görs en interpolering mellan de två närmaste värdena.

Argument:

Argument

Argument	Beskrivning
expr	Uttrycket eller fältet som innehåller de data som ska användas för beräkning av fraktilen.
fraction	Ett tal mellan 0 och 1 som motsvarar den fraktil (kvantil uttryckt som bråkdel) som ska beräknas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Exempel och resultat:

Example table

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Astrida	46	60	70	13	78	20	45	65	78	12	78	22
Betacab	65	56	22	79	12	56	45	24	32	78	55	15
Canutility	77	68	34	91	24	68	57	36	44	90	67	27
Divadip	57	36	44	90	67	27	57	68	47	90	80	94

Exempel på funktioner

Exempel	Resultat
`FractileExc (Sales, 0.75)`	För en tabell som innehåller dimensionen `Customer` och måttet `FractileExc([Sales])`, är resultatet 75.25 om **Totalvärden** visas. Detta är punkten i värdefördelningen för `Sales` som 75 % av värdena faller under.
`FractileExc (TOTAL Sales, 0.75))`	75.25 för alla värden för `Customer`, eftersom TOTAL-kvalificeraren innebär att dimensionerna ignoreras.
`FractileExc (DISTINCT Sales, 0.75)`	73.50 som totalvärde, eftersom DISTINCT-kvalificeraren innebär att endast unika värden i `Sales` för varje `Customer` utvärderas.

Data som används i exempel:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Se även:

Aggr - diagramfunktion (page 404)

Kurtosis

Kurtosis() returnerar graden av kurtosis av data i uttrycket över ett antal poster som definieras av en **group by**-sats.

Syntax:

```
Kurtosis([distinct ] expr )
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket kommer alla dubbletter att ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Resulterande data

Exempel	Resultat
Table1: crosstable LOAD recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|'); Kurtosis1: LOAD Type, Kurtosis(Value) as MyKurtosis1, Kurtosis(DISTINCT Value) as MyKurtosis2 Resident Table1 Group By Type;	I en tabell med dimensionerna `Type`, `MyKurtosis1` och `MyKurtosis2` blir resultaten av Kurtosis()-beräkningarna i dataladdningsskriptet: Type MyKurtosis1 MyKurtosis2 Comparison -1.1612957 -1.4982366 Observation -1.1148768 -0.93540144

Kurtosis - diagramfunktion

Kurtosis() finner kurtosis av dataintervallet aggregerat i uttrycket eller fältet itererat över diagrammets dimensioner.

Syntax:

```
Kurtosis([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Exempel och resultat:

Example table

Type	Value																			
Comparison	2	27	38	31	1	19	1	34	3	1	2	3	2	1	2	1	3	29	37	2
Observation	35	40	12	15	21	14	46	10	28	48	16	30	32	48	31	22	12	39	19	25

Exempel på funktioner

Exempel	Resultat
`Kurtosis(Value)`	För en tabell som innehåller dimensionen `Type` och måttet `Kurtosis(Value)` och om **Totalvärden** visas för tabellen och talformatet är inställt på 3 signifikanta siffror, är resultatet 1,252. För `Comparison` är det 1,161 och för `Observation` är det 1,115.

Exempel	Resultat
`Kurtosis (TOTAL Value))`	1.252 för alla värden för `Type`, eftersom TOTAL-kvalificeraren innebär att dimensionerna ignoreras.

Data som används i exempel:

```
Table1:
crosstable LOAD recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2 ] (delimiter is '|');
```

Se även:

Avg - diagramfunktion (page 267)

LINEST_B

LINEST_B() returnerar det aggregerade b-värdet (y-intercept) hos en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras i en **group by**-sats.

Syntax:

```
LINEST_B (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)

LINEST_B - diagramfunktion

LINEST_B() returnerar det aggregerade b-värdet (y-intercept) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_B([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [ , x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0_const, x0_const	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)
Avg - diagramfunktion (page 267)

LINEST_DF

LINEST_DF() returnerar den aggregerade frihetsgraden hos en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras i en**group by**-sats.

Syntax:

```
LINEST_DF (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)

LINEST_DF - diagramfunktion

LINEST_DF() returnerar de aggregerade frihetsgraderna för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_DF([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value [, y0_const [, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)
Avg - diagramfunktion (page 267)

LINEST_F

Denna skriptfunktion returnerar det aggregerade F-värdet ($r^2/(1-r^2)$) hos en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras av en **group by**-sats.

Syntax:

```
LINEST_F (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)

LINEST_F - diagramfunktion

LINEST_F() returnerar det aggregerade F-värdet (r2/(1-r2)) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_F([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)
Avg - diagramfunktion (page 267)

LINEST_M

LINEST_M() returnerar det aggregerade m-värdet (lutning) hos en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definieras i en **group by**-sats.

Syntax:

```
LINEST_M (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)

LINEST_M - diagramfunktion

LINEST_M() returnerar det aggregerade m-värdet (lutningen) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_M([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)
Avg - diagramfunktion (page 267)

LINEST_R2

LINEST_R2() returnerar det aggregerade r^2-värdet (bestämningskoefficienten) av en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som representeras av parvisa tal i x-expression och y-expression itererat över ett antal poster som definieras av en **group by**-sats.

Syntax:

```
LINEST_R2 (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)

LINEST_R2 - diagramfunktion

LINEST_R2() returnerar det aggregerade r2-värdet (bestämningskoefficienten) för en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_R2([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)
Avg - diagramfunktion (page 267)

LINEST_SEB

LINEST_SEB() returnerar det aggregerade standardfelet hos b-värdet i en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt definitionen av en **group by**-sats.

Syntax:

```
LINEST_SEB (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)

LINEST_SEB - diagramfunktion

LINEST_SEB() returnerar det aggregerade standardfelet för b-värdet av en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_SEB([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)
Avg - diagramfunktion (page 267)

LINEST_SEM

LINEST_SEM() returnerar det aggregerade standardfelet hos m-värdet i en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt definitionen av en **group by**-sats.

Syntax:

```
LINEST_SEM (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)

LINEST_SEM - diagramfunktion

LINEST_SEM() returnerar det aggregerade standardfelet för m-värdet av en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_SEM([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)
Avg - diagramfunktion (page 267)

LINEST_SEY

LINEST_SEY() returnerar det aggregerade standardfelet hos y-uppskattningen i en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt definitionen av en **group by**-sats.

Syntax:

```
LINEST_SEY (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)

LINEST_SEY - diagramfunktion

LINEST_SEY() returnerar det aggregerade standardfelet för y-estimatet av en linjär regression som definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_SEY([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)
Avg - diagramfunktion (page 267)

LINEST_SSREG

LINEST_SSREG returnerar den aggregerade restsumman av en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som representeras av parvisa tal i x-expression och y-expression itererat över ett antal poster som definieras av en **group by**-sats.

Syntax:

```
LINEST_SSREG (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)

LINEST_SSREG - diagramfunktion

LINEST_SSREG() returnerar den aggregerade regressionskvadratsumman av en linjär regression definierad genom ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal som ges av uttrycken **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_SSREG([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)
Avg - diagramfunktion (page 267)

LINEST_SSRESID

LINEST_SSRESID() returnerar den aggregerade restsumman av en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som representeras av parvisa tal i x-expression och y-expression itererat över ett antal poster som definieras av en **group by**-sats.

Syntax:

```
LINEST_SSRESID (y_value, x_value[, y0 [, x0 ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.
y(0), x(0)	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)

LINEST_SSRESID - diagramfunktion

LINEST_SSRESID() returnerar den aggregerade residualkvadratsumman hos en linjär regression som den definieras av ekvationen y=mx+b för en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **x_value** och **y_value**, itererat över diagrammets dimensioner.

Syntax:

```
LINEST_SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value,
x_value[, y0_const[, x0_const]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Argument	Beskrivning
y0, x0	Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat. *Om inte både y0 och x0 har angetts kräver funktionen minst två giltiga datapar för att kunna beräknas. Om både y0 och x0 har angivits, räcker det med ett enda datapar.*
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Ett valfritt värde y0 kan anges för att forcera regressionslinjen att passera y-axeln vid en bestämd punkt. Om såväl y0 som x0 anges kan regressionlinjen forceras att passera en viss koordinat.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Se även:

Exempel på användning av linest-funktioner (page 323)
Avg - diagramfunktion (page 267)

Median

Median() returnerar den aggregerade medianen av värdena i uttrycket över ett antal poster som definieras av en **group by**-sats.

Syntax:

```
Median (expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Exempel: Skriptuttryck som använder medianen

Exempel - skriptuttryck

Laddningsskript

Ladda följande inline-data och skriptuttryck i skriptredigeraren för det här exemplet.

```
Table 1: Load RecNo() as RowNo, Letter, Number Inline [Letter, Number A,1 A,3 A,4 A,9 B,2 B,8
B,9];                                    Median: LOAD Letter, Median(Number) as MyMedian Resident Table1 Grou
```

Skapa en visualisering

Skapa en tabellvisualisering i ett Qlik Sense-ark med **Letter** och **MyMedian** som dimensioner.

Resultat

Letter	MyMedian
A	3.5
B	8

Förklaring

Medianen betraktas som värdet ”i mitten” när de har sorterats från lägsta till högsta. Om datauppsättningen har ett jämnt antal värden kommer funktionen att returnera genomsnittet för de två mittenvärdena. I det här exemplet beräknas medianen för varje uppsättning med värden för **A** och **B**, som är 3,5 respektive 8.

Median - diagramfunktion

Median() returnerar medianvärdet för värdeintervallet som aggregeras i uttrycket, itererat över diagrammets dimensioner.

Syntax:

```
Median([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Exempel: Diagramuttryck som använder medianen

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplet nedan.

```
Load RecNo() as RowNo, Letter, Number Inline [Letter, Number A,1 A,3 A,4 A,9 B,2 B,8 B,9];
```

Skapa en visualisering

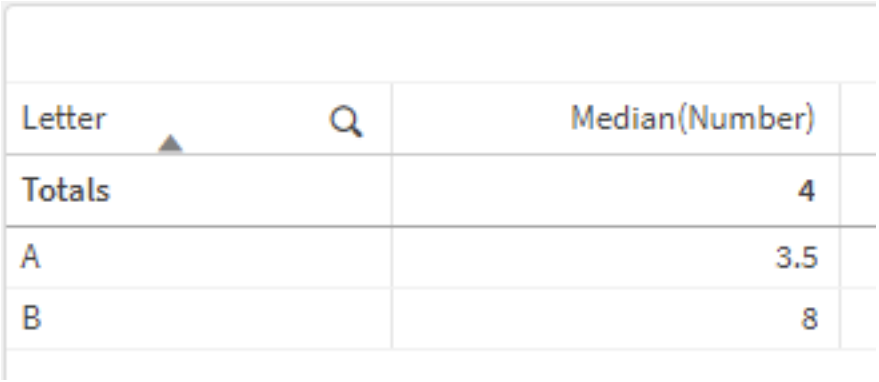
Skapa en tabellvisualisering i ett Qlik Sense-ark med **Letter** som en dimension.

Diagramuttryck

Lägg till följande uttryck i tabellen som ett mått:

```
Median(Number)
```

Resultat



Letter	Median(Number)
Totals	4
A	3.5
B	8

Förklaring

Medianen betraktas som värdet ”i mitten” när de har sorterats från lägsta till högsta. Om datauppsättningen har ett jämnt antal värden kommer funktionen att returnera genomsnittet för de två mittenvärdena. I det här exemplet beräknas medianen för varje uppsättning med värden för **A** och **B**, som är 3,5 respektive 8.

Medianen för **Totalvärden** beräknas för alla värden, vilket är lika med 4.

Se även:

Avg - diagramfunktion (page 267)

MutualInfo - diagramfunktion

MutualInfo beräknar ömsesidig information (MI) mellan två fält eller mellan aggregerade värden i **Aggr()**.

MutualInfo returnerar den aggregerade gemensamma informationen för två datauppsättningar. Detta möjliggör analys av nyckeldrivare för ett fält och en potentiell drivare. Gemensam information mäter förhållandet mellan datauppsättningarna och aggregeras för (x,y)-parvärden, vilket upprepas för diagrammets dimensioner. Gemensam information har ett mått mellan 0 och 1 och kan formateras som ett percentilvärde. **MutualInfo** definieras av antingen urval eller ett set-uttryck.

MutualInfo tillåter olika typer av MI-analys:

- Gemensam information (MI) för par: Beräkna MI i ett drivande fält och ett målfält.
- Uppdelning av drivare efter värde: MI beräknas med individuella fältvärden i drivande fält och målfält.
- Funktionsurval: Använd **MutualInfo** i ett rutnätsdiagram om du vill skapa en matris där alla fält jämförs med varandra baserat på MI.

MutualInfo visar inte nödvändigtvis på kausalitet mellan fält med gemensam information. Även om information är gemensam för två fält behöver de inte driva varandra lika mycket. Om du till exempel jämför glassförsäljning och utomhustemperatur, visar **MutualInfo** att de har gemensam information. Detta visar inte om det är utomhustemperaturen som driver glassförsäljningen, vilket är troligt, eller om det är glassförsäljningen som driver utomhustemperaturen, vilket är osannolikt.

Vid beräkning av ömsesidig information påverkar associationer överensstämmelsen mellan och frekvensen av värden från fält som kommer från olika tabeller.

Värdena som returneras för samma fält eller urval kan variera något. Detta beror på att varje **MutualInfo**-anrop verkar på slumpmässigt utvalda exempeldata och på **MutualInfo**-algoritmens inbyggda slumpmässighet.

MutualInfo kan tillämpas på funktionen **Aggr()**.

Syntax:

```
MutualInfo({SetExpression}] [DISTINCT] [TOTAL] field1, field2 , datatype [,
breakdownbyvalue [, samplesize ]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
field1, field2	Uttrycken eller fälten som innehåller de två exempeluppsättningarna den gemensamma informationen ska mätas för.
datatype	Datatyperna som finns i målet och drivaren, 1 eller 'dd' för diskret:diskret 2 eller 'cc' för kontinuerlig:kontinuerlig 3 eller 'cd' för kontinuerlig:diskret 4 eller 'dc' för diskret:kontinuerlig Datatyper är inte skiftlägeskänsliga.
breakdownbyvalue	Ett statiskt värde som motsvarar ett värde i drivaren. Om det tillhandahålls beräknas MI-bidraget för det värdet. Du kan använda **ValueList()** eller **ValueLoop()**. Om **Null()** läggs till beräknas total MI för alla värden i drivaren. För uppdelning efter värde måste drivaren innehålla diskreta data.
samplesize	Antal värden som ska ingå i exempeldata från mål och drivare. Val av exempeldata är slumpmässigt. **MutualInfo** kräver en exempelstorlek på minst 80. Som standard använder **MutualInfo** endast upp till 10 000 datapar som exempeldata eftersom **MutualInfo** kan kräva mycket resurser. Du kan specificera fler datapar i exempeldatastorleken. Om **MutualInfo** når tidsgränsen minskar du mängden exempeldata.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.

Argument	Beskrivning
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Exempel på funktioner

Exempel	Resultat
`mutualinfo(Age, Salary, 1)`	Om en tabell innehåller dimensionen `Employee name` och måttet `mutualinfo(Age, Salary, 1)` är resultatet 0.99820986. Resultaten visas endast för totalcellen.
`mutualinfo (TOTAL Age, Salary, 1, null (), 81)`	Om du skapar en filterruta med dimensionen Gender och gör urval från den, ser du resultatet 0.99805677 när Female är valt och 0.99847373 om Male är valt. Detta beror på att urvalet utesluter alla resultat som inte hör till det andra värdet för Gender.
`mutualinfo (TOTAL Age, Gender, 1, ValueLoop (25,35))`	0.68196996. Alla värden som väljs från Gender ändrar detta till 0.
`mutualinfo({1} TOTAL Age, Salary, 1, null ())`	0.99820986. Detta är oberoende av urval. Set-uttrycket {1} bortser från alla urval och dimensioner.

Data som används i exempel:

```
Salary:

LOAD * inline [

"Employee name"|Age|Gender|Salary
```

```
Aiden Charles|20|Male|25000

Ann Lindquist|69|Female|58000

Anna Johansen|37|Female|36000

Anna Karlsson|42|Female|23000

Antonio Garcia|20|Male|61000

Benjamin Smith|42|Male|27000

Bill Yang|49|Male|50000

Binh Protzmann|69|Male|21000

Bob Park|51|Male|54000

Brenda Davies|25|Male|32000

Celine Gagnon|48|Female|38000

Cezar Sandu|50|Male|46000

Charles Ingvar Jönsson|27|Male|58000

Charlotte Edberg|45|Female|56000

Cindy Lynn|69|Female|28000

Clark Wayne|63|Male|31000

Daroush Ferrara|31|Male|29000

David Cooper|37|Male|64000

David Leg|58|Male|57000

Eunice Goldblum|31|Female|32000

Freddy Halvorsen|25|Male|26000

Gauri Indu|36|Female|46000

George van Zaant|59|Male|47000

Glenn Brown|58|Male|40000

Harry Jones|38|Male|40000

Helen Brolin|52|Female|66000

Hiroshi Ito|24|Male|42000
```

```
Ian Underwood|40|Male|45000

Ingrid Hendrix|63|Female|27000

Ira Baumel|39|Female|39000

Jackie Kingsley|23|Female|28000

Jennica Williams|36|Female|48000

Jerry Tessel|31|Male|57000

Jim Bond|50|Male|58000

Joan Callins|60|Female|65000

Joan Cleaves|25|Female|61000

Joe Cheng|61|Male|41000

John Doe|36|Male|59000

John Lemon|43|Male|21000

Karen Helmkey|54|Female|25000

Karl Berger|38|Male|68000

Karl Straubaum|30|Male|40000

Kaya Alpan|32|Female|60000

Kenneth Finley|21|Male|25000

Leif Shine|63|Male|70000

Lennart Skoglund|63|Male|24000

Leona Korhonen|46|Female|50000

Lina André|50|Female|65000

Louis Presley|29|Male|36000

Luke Langston|50|Male|63000

Marcus Salvatori|31|Male|46000

Marie Simon|57|Female|23000

Mario Rossi|39|Male|62000

Markus Danzig|26|Male|48000
```

```
Michael Carlen|21|Male|45000

Michelle Tyson|44|Female|69000

Mike Ashkenaz|45|Male|68000

Miro Ito|40|Male|39000

Nina Mihn|62|Female|57000

Olivia Nguyen|35|Female|51000

Olivier Simenon|44|Male|31000

Östen Ärlig|68|Male|57000

Pamala Garcia|69|Female|29000

Paolo Romano|34|Male|45000

Pat Taylor|67|Female|69000

Paul Dupont|34|Male|38000

Peter Smith|56|Male|53000

Pierre Clouseau|21|Male|37000

Preben Jørgensen|35|Male|38000

Rey Jones|65|Female|20000

Ricardo Gucci|55|Male|65000

Richard Ranieri|30|Male|64000

Rob Carsson|46|Male|54000

Rolf Wesenlund|25|Male|51000

Ronaldo Costa|64|Male|39000

Sabrina Richards|57|Female|40000

Sato Hiromu|35|Male|21000

Sehoon Daw|57|Male|24000

Stefan Lind|67|Male|35000

Steve Cioazzi|58|Male|23000

Sunil Gupta|45|Male|40000
```

```
Sven Svensson|45|Male|55000

Tom Lindwall|46|Male|24000

Tomas Nilsson|27|Male|22000

Trinity Rizzo|52|Female|48000

Vanessa Lambert|54|Female|27000

] (delimiter is '|');
```

Skew

Skew() returnerar skevheten i uttryck över ett antal poster enligt vad som definierats i en **group by**-sats.

Syntax:

```
Skew([ distinct] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
DISTINCT	Om predikatet **distinct** förekommer framför uttrycket kommer alla dubbletter att ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Bygg sedan en rak tabell med `Type` och *MySkew* som dimensioner.

Resulterande data

Exempel	Resultat
Table1: crosstable LOAD recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|'); Skew1: LOAD Type, Skew(Value) as MySkew Resident Table1 Group By Type;	Resultaten av Skew()-beräkningen är: • Type är MySkew • Comparison är 0.86414768 • Observation är 0.32625351

Skew - diagramfunktion

Skew() returnerar den aggregerade skevheten av uttrycket eller fältet itererat över diagrammets dimensioner.

Syntax:

```
Skew([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.

Argument	Beskrivning
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Bygg sedan en rak tabell med `Type` som dimension och `Skew (Value)` som mått.

`Totals` ska vara aktiverat i tabellegenskaperna.

Exempel	Resultat

`Table1:`
`crosstable LOAD recno()`
`as ID, * inline [`
`Observation|Comparison`
`35|2`
`40|27`
`12|38`
`15|31`
`21|1`
`14|19`
`46|1`
`10|34`
`28|3`
`48|1`
`16|2`
`30|3`
`32|2`
`48|1`
`31|2`
`22|1`
`12|3`
`39|29`
`19|37`
`25|2 ] (delimiter is`
`'|');`
	Resultaten av Skew(Value)-beräkningen är:
	
	• `Total`är 0.23522195
	• `Comparison` är 0.86414768
	• `Observation` är 0.32625351

Se även:

- *Avg - diagramfunktion (page 267)*

Stdev

Stdev() returnerar värdenas standardavvikelse i uttrycket över ett antal poster som definieras av en **group by**-sats.

Syntax:

```
Stdev([distinct] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket kommer alla dubbletter att ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Bygg sedan en rak tabell med `Type` och `MyStdev` som dimensioner.

Resulterande data

Exempel	Resultat
Table1: crosstable LOAD recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|'); Stdev1: LOAD Type, Stdev(Value) as MyStdev Resident Table1 Group By Type;	Resultaten av Stdev()-beräkningen är: • Type är MyStdev • Comparison är 14.61245 • Observation är 12.507997

Stdev - diagramfunktion

Stdev() finner standardavvikelsen för dataintervallet som aggregerats i uttrycket eller fältet itererat över diagrammets dimensioner.

Syntax:

```
Stdev([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.

Argument	Beskrivning
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Bygg sedan en rak tabell med `Type` som dimension och `Stdev (Value)` som mått.

`Totals` ska vara aktiverat i tabellegenskaperna.

Exempel	Resultat

`Stdev(Value)`
`Table1:`
`crosstable LOAD recno()`
`as ID, * inline [`
`Observation|Comparison`
`35|2`
`40|27`
`12|38`
`15|31`
`21|1`
`14|19`
`46|1`
`10|34`
`28|3`
`48|1`
`16|2`
`30|3`
`32|2`
`48|1`
`31|2`
`22|1`
`12|3`
`39|29`
`19|37`
`25|2 ] (delimiter is`
`'|');`
	Resultaten av Stdev(Value)-beräkningen är:
	
	• `Total`är 15.47529
	• `Comparison` är 14.61245
	• `Observation` är 12.507997

Se även:

- *Avg - diagramfunktion (page 267)*
- *STEYX - diagramfunktion (page 321)*

Sterr

Sterr() returnerar det aggregerade standardfelet (stdev/sqrt(n)) för en serie värden, representerade av uttryck över ett antal poster enligt vad som definierats i en **group by**-sats.

Syntax:

```
Sterr ([distinct] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
distinct	Om predikatet **distinct** förekommer framför uttrycket kommer alla dubbletter att ignoreras.

Begränsningar:

Textvärden, NULL-värden samt saknade värden ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Resulterande data

Exempel	Resultat
Table1: crosstable LOAD recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|'); Sterr1: LOAD Type, Sterr(Value) as MySterr Resident Table1 Group By Type;	I en tabell med dimensionerna `Type` och `MySterr` blir resultaten av Sterr ()-beräkningen i dataladdningsskriptet: Type MySterr Comparison 3.2674431 Observation 2.7968733

Sterr - diagramfunktion

Sterr() finner värdet av standardfelet av medelvärdet (stdev/sqrt(n)) för värdeserien som aggregerats i uttrycket itererat över diagrammets dimensioner.

Syntax:

```
Sterr([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Argument	Beskrivning
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {,fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden samt saknade värden ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Bygg sedan en rak tabell med `Type` som dimension och `Sterr (Value)` som mått.

`Totals` ska vara aktiverat i tabellegenskaperna.

Exempel	Resultat
Table1: crosstable LOAD recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|');	Resultaten av Sterr(Value)-beräkningen är: • Totalär 2.4468583 • Comparison är 3.2674431 • Observation är 2.7968733

Se även:

- *Avg - diagramfunktion (page 267)*
- *STEYX - diagramfunktion (page 321)*

STEYX

STEYX() returnerar det aggregerade standardfelet hos det predikterade y-värdet för varje x-värde i regressionen för en serie koordinater, representerade av parvisa tal i x-expression och y-expression itererade över ett antal poster enligt vad som definierats i en **group by**-sats.

Syntax:

```
STEYX (y_value, x_value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller y-värdeintervallet som ska mätas.
x_value	Uttrycket eller fältet som innehåller x-värdeintervallet som ska mätas.

Begränsningar:

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Resulterande data

Exempel	Resultat

`Trend:`
`Load *, 1 as Grp;`
`LOAD * inline [`
`Month|KnownY|KnownX`
`Jan|2|6`
`Feb|3|5`
`Mar|9|11`
`Apr|6|7`
`May|8|5`
`Jun|7|4`
`Jul|5|5`
`Aug|10|8`
`Sep|9|10`
`Oct|12|14`
`Nov|15|17`
`Dec|14|16`
`] (delimiter is '|');`
 	
`STEYX1:`
`LOAD Grp,`
`STEYX(KnownY, KnownX)`
`as MySTEYX`
`Resident Trend Group`
`By Grp;`
	I en tabell med storleken `MySTEYX` är resultatet i beräkningen STEYX() i dataladdningsskriptet 2.0714764.

STEYX - diagramfunktion

STEYX() returnerar det aggregerade standardfelet vid prediktering av y-värden för varje x-värde i en linjär regression som ges av en serie koordinater som motsvaras av parvisa tal i de uttryck som ges av **y_value** och **x_value**.

Syntax:

```
STEYX([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
y_value	Uttrycket eller fältet som innehåller intervallet av kända y-värden som ska mätas.
x_value	Uttrycket eller fältet som innehåller intervallet av kända x-värden som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {,fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Aggregeringsfunktionens parameter får inte innehålla andra aggregeringsfunktioner, om inte dessa inre aggregeringar innehåller **TOTAL**-kvalificeraren. För mer avancerade nästlade aggregeringar använder du den avancerade **Aggr**-funktionen i kombination med en specificerad dimension.

Textvärden, NULL-värden och saknade värden i någon eller båda delarna av ett datapar resulterar i att hela dataparet ignoreras.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Bygg sedan en rak tabell med `KnownY` och `KnownX` som dimension och `Steyx(KnownY,KnownX)` som mått.

`Totals` ska vara aktiverat i tabellegenskaperna.

Exempel	Resultat
Trend: LOAD * inline [Month|KnownY|KnownX Jan|2|6 Feb|3|5 Mar|9|11 Apr|6|7 May|8|5 Jun|7|4 Jul|5|5 Aug|10|8 Sep|9|10 Oct|12|14 Nov|15|17 Dec|14|16] (delimiter is '|');	Resultatet av STEYX(KnownY,KnownX)-beräkningen är 2,071 (om nummerformatering är inställt på tre decimaler.)

Se även:

- *Avg - diagramfunktion (page 267)*
- *Sterr - diagramfunktion (page 318)*

Exempel på användning av linest-funktioner

Funktioner av typen linest används för att hitta värden associerade med analys av linjär regression. I det här avsnittet beskrivs hur du kan bygga visualiseringar med hjälp av sticksprovsdata för att hitta värdena för linest-funktionerna som finns i Qlik Sense. linest-funktionerna kan användas i dataladdningsskriptet och diagramuttryck.

Se de individuella avsnitten för linest-diagram- och skriptfunktionerna för beskrivningar av syntax och argument.

Uttryck för data och skript används i exemplen

Ladda följande inline-data och skriptuttryck i skriptredigeraren för linest()-exemplen nedan.

```
T1: LOAD *, 1 as Grp; LOAD * inline [ X|Y 1|0 2|1 3|3 4|8 5|14 6|20 7|0 8|50 9|25 10|60 11|38 12|19 13|26 14|143 15|98 16|27 17|59 18|78 19|158 20|279 ] (delimiter is '|');                    R1: LOAD Grp, linest_B(Y,X) as Linest_B, linest_DF(Y,X) as Linest_DF, linest_F(Y,X) as Linest_F, linest_M(Y,X) as Linest_M, linest_R2(Y,X) as Linest_R2, linest_SEB(Y,X,1,1) as Linest_SEB, linest_SEM(Y,X) as Linest_SEM, linest_SEY(Y,X) as Linest_SEY, linest_SSREG(Y,X) as Linest_SSREG, linest_SSRESID(Y,X) as Linest_SSRESID resident T1 group by Grp;
```

Exempel 1: Skriptuttryck som använder linest

Exempel: Skriptuttryck

Skapa en visualisering från beräkningar av dataladdningsskript

Skapa en tabellvisualisering i ett Qlik Sense-ark med följande fält som kolumner:

- Linest_B
- Linest_DF

- Linest_F
- Linest_M
- Linest_R2
- Linest_SEB
- Linest_SEM
- Linest_SEY
- Linest_SSREG
- Linest_SSRESID

Resultat

Tabellen med resultat från linest-beräkningarna som gjordes för dataladdningsskriptet ska se ut så här:

Resultattabell

Linest_B	Linest_DF	Linest_F	Linest_M	Linest_R2	Linest_SEB
-35.047	18	20.788	8.605	0.536	22.607

Resultattabell

Linest_SEM	Linest_SEY	Linest_SSREG	Linest_SSRESID
1.887	48.666	49235.014	42631.186

Exempel 2: Diagramuttryck som använder linest

Exempel: Diagramuttryck

Skapa en tabellvisualisering i ett Qlik Sense-ark med följande fält som dimensioner:

```
ValueList('Linest_b', 'Linest_df','Linest_f', 'Linest_m','Linest_r2','Linest_SEB','Linest_
SEM','Linest_SEY','Linest_SSREG','Linest_SSRESID')
```

Nu använder uttrycket funktionen för syntetiska dimensioner för att skapa etiketter för dimensionerna med namnen på linest-funktionerna. Du kan ändra etiketten till **Linest functions** för att spara plats.

Lägg till följande uttryck i tabellen som ett mått:

```
Pick(Match(ValueList('Linest_b', 'Linest_df','Linest_f', 'Linest_m','Linest_r2','Linest_
SEB','Linest_SEM','Linest_SEY','Linest_SSREG','Linest_SSRESID'),'Linest_b', 'Linest_
df','Linest_f', 'Linest_m','Linest_r2','Linest_SEB','Linest_SEM','Linest_SEY','Linest_
SSREG','Linest_SSRESID'),Linest_b(Y,X),Linest_df(Y,X),Linest_f(Y,X),Linest_m(Y,X),Linest_r2
(Y,X),Linest_SEB(Y,X,1,1),Linest_SEM(Y,X),Linest_SEY(Y,X),Linest_SSREG(Y,X),Linest_SSRESID
(Y,X) )
```

Detta uttryck visar värdet av resultatet för varje linest-funktion mot motsvarande namn i den syntetiska dimensionen. Resultatet för `Linest_b(Y,X)` visas jämte **linest_b** och så vidare.

Resultat

Resultattabell

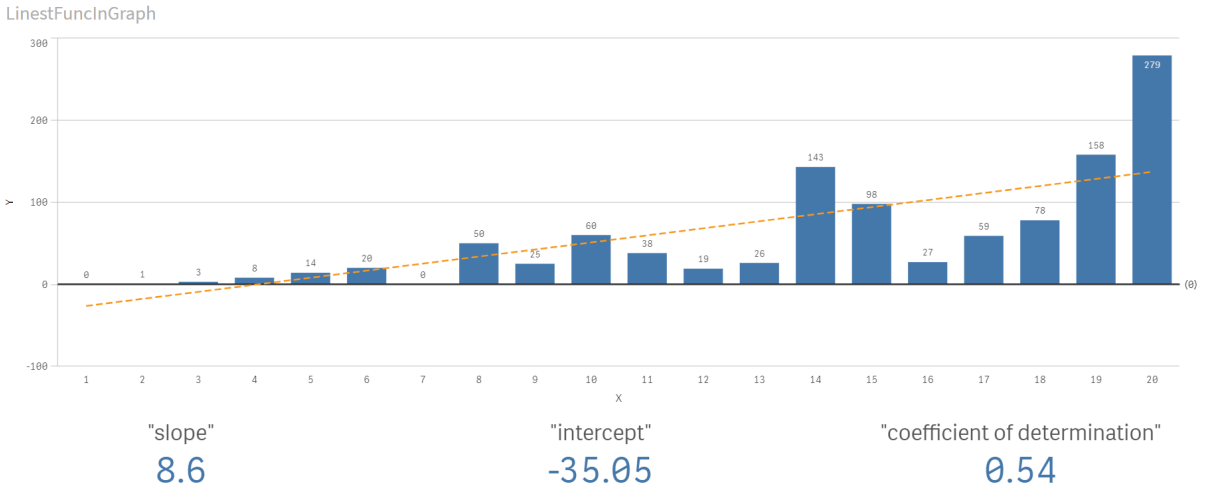
Linest functions	Linest function results
Linest_b	-35.047
Linest_df	18
Linest_f	20.788
Linest_m	8.605
Linest_r2	0.536
Linest_SEB	22.607
Linest_SEM	1.887
Linest_SEY	48.666
Linest_SSREG	49235.014
Linest_SSRESID	42631.186

Exempel 3: Diagramuttryck som använder linest

Exempel: Diagramuttryck

1. Skapa en visualisering med stapeldiagram i ett Qlik Sense-ark med **X** som dimension och **Y** som mått.
2. Lägg till en linjär trendlinje i Y-måttet.
3. Lägg till en KPI-visualisering på arket.
    1. Lägg till *lutning* som en etikett för KPI.
    2. Lägg till `sum(Linest_M)` som ett uttryck för KPI.
4. Lägg till en andra KPI-visualisering på arket.
    1. Lägg till *skärningspunkt* som en etikett för KPI.
    2. Lägg till `Sum(Linest_B)` som ett uttryck för KPI.
5. Lägg till en tredje KPI-visualisering på arket.
    1. Lägg till *bestämningskoefficient* som en etikett för KPI.
    2. Lägg till `Sum(Linest_R2)` som ett uttryck för KPI.

Resultat



Förklaring

Stapeldiagrammet visar hur X- och Y-data ritas ut. Relevanta linest()-funktioner ger värden till den linjära regressionsekvationen som trendlinjen baseras på, nämligen y = m * x + b. Ekvationen använder minsta kvadratmetoden för att beräkna en rak linje (trendlinje) genom att returnera en matris som beskriver en linje som passar bäst för data.

KPI:er visar resultaten för linest()-funktionerna **sum(Linest_M)** för lutning och **sum(Linest_B)** för Y-skärningspunkten, som är variabler i den linjära regressionsekvationen och motsvarande aggregerade R2-värde för bestämningskoefficienten.

Statistiska testfunktioner

Statistiska testfunktioner kan användas i både dataladdningsskriptet och diagramuttryck, men syntaxen är olika.

Chi-2-testfunktioner

Används allmänt i studien av kvalitativa variabler. Man kan jämföra observerade frekvenser i en enkelriktad frekvenstabell med förväntade frekvenser, eller studera sambandet mellan två variabler i en reservtabell.

T-testfunktioner

T-testfunktioner används för statistisk undersökning av två populationsmedelvärden. Ett t-test med två stickprov undersöker om två stickprov skiljer sig åt och används vanligtvis när två normalfördelningar har okända varianser och när en liten urvalsstorlek används vid ett experiment.

Z-testfunktioner

En statistisk undersökning av två populationsmedelvärden. Ett z-test med två stickprov som undersöker om två stickprov skiljer sig åt och används vanligtvis när två normalfördelningar har kända varianser och när ett experiment använder en stor urvalsstorlek.

Chi2-testfunktioner

Används allmänt i studien av kvalitativa variabler. Man kan jämföra observerade frekvenser i en enkelriktad frekvenstabell med förväntade frekvenser, eller studera sambandet mellan två variabler i en reservtabell.Chi-squared test functions are used to determine whether there is a statistically significant difference between the expected frequencies and the observed frequencies in one or more groups. Often a histogram is used, and the different bins are compared to an expected distribution.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Chi2Test_chi2
Chi2Test_chi2() returnerar det aggregerade värdet av ett chi^2-test för en eller två värdeserier.

```
Chi2Test_chi2() returnerar det aggregerade värdet av ett chi2-test för en
eller två värdeserier.(col, row, actual_value[, expected_value])
```

Chi2Test_df
Chi2Test_df() returnerar det aggregerade df-värdet (frihetsgrader) av ett chi^2-test för en eller två värdeserier.

```
Chi2Test_df() returnerar det aggregerade df-värdet (frihetsgrader) av ett
chi2-test för en eller två värdeserier.(col, row, actual_value[, expected_
value])
```

Chi2Test_p
Chi2Test_p() returnerar det aggregerade p-värdet (signifikansen) av ett chi^2-test för en eller två värdeserier.

```
Chi2Test_p - diagramfunktion(col, row, actual_value[, expected_value])
```

Se även:

- *T-testfunktioner (page 330)*
- *Z-testfunktioner (page 363)*

Chi2Test_chi2

Chi2Test_chi2() returnerar det aggregerade värdet av ett chi^2-test för en eller två värdeserier.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Alla Qlik Sense chi²-testfunktioner har samma argument.

Syntax:

```
Chi2Test_chi2(col, row, actual_value[, expected_value])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
col, row	Den angivna kolumnen och raden i den matris av värden som ska testas.
actual_value	Det observerade värdet av data vid angiven **col** och **row**.
expected_value	Det väntade värdet för fördelningen vid angiven **col** och **row**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
Chi2Test_chi2( Grp, Grade, Count )
Chi2Test_chi2( Gender, Description, Observed, Expected )
```

Se även:

- *Exempel på användning av chi2-test-funktioner i diagram (page 378)*
- *Exempel på användning av chi2-test-funktioner i dataladdningsskriptet (page 381)*

Chi2Test_df

Chi2Test_df() returnerar det aggregerade df-värdet (frihetsgrader) av ett chi²-test för en eller två värdeserier.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Alla Qlik Sense chi²-testfunktioner har samma argument.

Syntax:

```
Chi2Test_df(col, row, actual_value[, expected_value])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
col, row	Den angivna kolumnen och raden i den matris av värden som ska testas.
actual_value	Det observerade värdet av data vid angiven **col** och **row**.
expected_value	Det väntade värdet för fördelningen vid angiven **col** och **row**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
Chi2Test_df( Grp, Grade, Count )
Chi2Test_df( Gender, Description, Observed, Expected )
```

Se även:

Exempel på användning av chi2-test-funktioner i diagram (page 378)
Exempel på användning av chi2-test-funktioner i dataladdningsskriptet (page 381)

Chi2Test_p - diagramfunktion

Chi2Test_p() returnerar det aggregerade p-värdet (signifikansen) av ett chi^2-test för en eller två värdeserier. Testet kan genomföras antingen för värdena i **actual_value** när man testar för variationer inom den angivna **col**- och **row**-matrisen eller genom att jämföra värden i **actual_value** med motsvarande värden i **expected_value**, om sådana finns angivna.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Alla Qlik Sense chi^2-testfunktioner har samma argument.

Syntax:

```
Chi2Test_p(col, row, actual_value[, expected_value])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
col, row	Den angivna kolumnen och raden i den matris av värden som ska testas.
actual_value	Det observerade värdet av data vid angiven **col** och **row**.
expected_value	Det väntade värdet för fördelningen vid angiven **col** och **row**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
Chi2Test_p( Grp, Grade, Count )
Chi2Test_p( Gender, Description, Observed, Expected )
```

Se även:

Exempel på användning av chi2-test-funktioner i diagram (page 378)
Exempel på användning av chi2-test-funktioner i dataladdningsskriptet (page 381)

T-testfunktioner

T-testfunktioner används för statistisk undersökning av två populationsmedelvärden. Ett t-test med två stickprov undersöker om två stickprov skiljer sig åt och används vanligtvis när två normalfördelningar har okända varianser och när en liten urvalsstorlek används vid ett experiment.

I följande avsnitt grupperas de statistiska t-testfunktionerna enligt exempeltestet som gäller för funktionen.

Skapa en typisk t-test-rapport (page 383)

T-test med två oberoende stickprov

Följande funktioner gäller student-t-test med två oberoende stickprov.

ttest_conf
TTest_conf returnerar det aggregerade värdet av konfidensintervallet för ett t-test för två oberoende stickprov.

```
TTest_conf returnerar det aggregerade värdet av konfidensintervallet för ett
t-test för två oberoende stickprov. ( grp, value [, sig[, eq_var]])
```

ttest_df

TTest_df() returnerar det aggregerade värdet (frihetsgrader) av ett t-test för två oberoende värdeserier.

```
TTest_df() returnerar det aggregerade värdet (frihetsgrader) av ett t-test
för två oberoende värdeserier. (grp, value [, eq_var)
```

ttest_dif

TTest_dif() är en numerisk funktion som returnerar den aggregerade skillnaden i medelvärde för ett t-test med två oberoende värdeserier.

```
TTest_dif() är en numerisk funktion som returnerar den aggregerade skillnaden
i medelvärde för ett t-test med två oberoende värdeserier. (grp, value)
```

ttest_lower

TTest_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

```
TTest_lower() returnerar det aggregerade värdet för konfidensintervallets
nedre gräns för två oberoende värdeserier. (grp, value [, sig[, eq_var]])
```

ttest_sig

TTest_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för två oberoende värdeserier.

```
TTest_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-
test för två oberoende värdeserier. (grp, value [, eq_var])
```

ttest_sterr

TTest_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för två oberoende värdeserier.

```
TTest_sterr() returnerar det aggregerade standardfelet för skillnaden i
medelvärde för ett t-test för två oberoende värdeserier. (grp, value [, eq_
var])
```

ttest_t

TTest_t() returnerar det aggregerade t-värdet för två oberoende värdeserier.

```
TTest_t() returnerar det aggregerade t-värdet för två oberoende värdeserier.
(grp, value [, eq_var])
```

ttest_upper

TTest_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

```
TTest_upper() returnerar det aggregerade värdet för konfidensintervallets
övre gräns för två oberoende värdeserier. (grp, value [, sig [, eq_var]])
```

T-test med två oberoende viktade stickprov

Följande funktioner gäller student-t-test med två oberoende stickprov där indataserien anges i viktat tvåkolumnsformat.

ttestw_conf

TTestw_conf() returnerar det aggregerade t-värdet för två oberoende värdeserier.

```
TTestw_conf() returnerar det aggregerade t-värdet för två oberoende
värdeserier. (weight, grp, value [, sig[, eq_var]])
```

ttestw_df

TTestw_df() returnerar det aggregerade df-värdet (frihetsgrader) av ett t-test för två oberoende värdeserier.

```
TTestw_df() returnerar det aggregerade df-värdet (frihetsgrader) av ett t-
test för två oberoende värdeserier. (weight, grp, value [, eq_var])
```

ttestw_dif

TTestw_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-test med två oberoende värdeserier.

```
TTestw_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-
test med två oberoende värdeserier. ( weight, grp, value)
```

ttestw_lower

TTestw_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

```
TTestw_lower() returnerar det aggregerade värdet för konfidensintervallets
nedre gräns för två oberoende värdeserier. (weight, grp, value [, sig[, eq_
var]])
```

ttestw_sig

TTestw_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för två oberoende värdeserier.

```
TTestw_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-
test för två oberoende värdeserier. ( weight, grp, value [, eq_var])
```

ttestw_sterr

TTestw_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för två oberoende värdeserier.

```
TTestw_sterr() returnerar det aggregerade standardfelet för skillnaden i
medelvärde för ett t-test för två oberoende värdeserier. (weight, grp, value
[, eq_var])
```

ttestw_t

TTestw_t() returnerar det aggregerade t-värdet för två oberoende värdeserier.

```
TTestw_t() returnerar det aggregerade t-värdet för två oberoende värdeserier.
(weight, grp, value [, eq_var])
```

ttestw_upper

TTestw_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

```
TTestw_upper() returnerar det aggregerade värdet för konfidensintervallets
övre gräns för två oberoende värdeserier. (weight, grp, value [, sig [, eq_
var]])
```

T-test med ett stickprov

Följande funktioner gäller student-t-test med ett stickprov.

ttest1_conf

TTest1_conf() returnerar det aggregerade konfidensintervallvärdet för en värdeserie.

```
TTest1_conf() returnerar det aggregerade konfidensintervallvärdet för en
värdeserie. (value [, sig])
```

ttest1_df

TTest1_df() returnerar det aggregerade df-värdet (frihetsgrader) för ett t-test för en värdeserie.

```
TTest1_df() returnerar det aggregerade df-värdet (frihetsgrader) för ett t-
test för en värdeserie. (value)
```

ttest1_dif

TTest1_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-test för en värdeserie.

```
TTest1_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-
test för en värdeserie. (value)
```

ttest1_lower

TTest1_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för en värdeserie.

```
TTest1_lower() returnerar det aggregerade värdet för konfidensintervallets
nedre gräns för en värdeserie. (value [, sig])
```

ttest1_sig

TTest1_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för en värdeserie.

```
TTest1_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-
test för en värdeserie. (value)
```

ttest1_sterr

TTest1_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för en värdeserie.

```
TTest1_sterr() returnerar det aggregerade standardfelet för skillnaden i
medelvärde för ett t-test för en värdeserie. (value)
```

ttest1_t

TTest1_t() returnerar det aggregerade t-värdet för en värdeserie.

```
TTest1_t() returnerar det aggregerade t-värdet för en värdeserie. (value)
```

ttest1_upper

TTest1_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för en värdeserie.

```
TTest1_upper() returnerar det aggregerade värdet för konfidensintervallets
övre gräns för en värdeserie. (value [, sig])
```

T-test med ett viktat stickprov

Följande funktioner gäller student-t-test med ett stickprov där indataserien anges i viktat tvåkolumnsformat.

ttest1w_conf

TTest1w_conf() är en **numerisk** funktion som returnerar det aggregerade konfidensintervallvärdet för en värdeserie.

```
TTest1w_conf() är en numerisk funktion som returnerar det aggregerade
konfidensintervallvärdet för en värdeserie. (weight, value [, sig])
```

ttest1w_df

TTest1w_df() returnerar det aggregerade df-värdet (frihetsgrader) för ett t-test för en värdeserie.

```
TTest1w_df() returnerar det aggregerade df-värdet (frihetsgrader) för ett t-
test för en värdeserie. (weight, value)
```

ttest1w_dif

TTest1w_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-test för en värdeserie.

```
TTest1w_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-
test för en värdeserie. (weight, value)
```

ttest1w_lower

TTest1w_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för en värdeserie.

```
TTest1w_lower() returnerar det aggregerade värdet för konfidensintervallets
nedre gräns för en värdeserie. (weight, value [, sig])
```

ttest1w_sig

TTest1w_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för en värdeserie.

```
TTest1w_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett
t-test för en värdeserie. (weight, value)
```

ttest1w_sterr

TTest1w_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för en värdeserie.

```
TTest1w_sterr() returnerar det aggregerade standardfelet för skillnaden i
medelvärde för ett t-test för en värdeserie. (weight, value)
```

ttest1w_t

TTest1w_t() returnerar det aggregerade t-värdet för en värdeserie.

```
TTest1w_t() returnerar det aggregerade t-värdet för en värdeserie. ( weight, value)
```

ttest1w_upper

TTest1w_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för en värdeserie.

```
TTest1w_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för en värdeserie. (weight, value [, sig])
```

TTest_conf

TTest_conf returnerar det aggregerade värdet av konfidensintervallet för ett t-test för två oberoende stickprov.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_conf ( grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_conf( Group, Value )
TTest_conf( Group, Value, Sig, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest_df

TTest_df() returnerar det aggregerade värdet (frihetsgrader) av ett t-test för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_df (grp, value [, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_df( Group, Value )
TTest_df( Group, Value, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest_dif

TTest_dif() är en numerisk funktion som returnerar den aggregerade skillnaden i medelvärde för ett t-test med två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_dif (grp, value [, eq_var] )
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_dif( Group, Value )
TTest_dif( Group, Value, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest_lower

TTest_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_lower (grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_lower( Group, Value )
TTest_lower( Group, Value, Sig, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest_sig

TTest_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_sig (grp, value [, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_sig( Group, Value )
TTest_sig( Group, Value, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest_sterr

TTest_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_sterr (grp, value [, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_sterr( Group, Value )
TTest_sterr( Group, Value, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest_t

TTest_t() returnerar det aggregerade t-värdet för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_t(grp, value[, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_t( Group, Value, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest_upper

TTest_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest_upper (grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest_upper( Group, Value )
TTest_upper( Group, Value, sig, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTestw_conf

TTestw_conf() returnerar det aggregerade t-värdet för två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTestw_conf (weight, grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_conf( Weight, Group, Value )
TTestw_conf( Weight, Group, Value, sig, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTestw_df

TTestw_df() returnerar det aggregerade df-värdet (frihetsgrader) av ett t-test för två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTestw_df (weight, grp, value [, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_df( Weight, Group, Value )
TTestw_df( Weight, Group, Value, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTestw_dif

TTestw_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-test med två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTestw_dif (weight, grp, value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_dif( Weight, Group, Value )
TTestw_dif( Weight, Group, Value, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTestw_lower

TTestw_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTestw_lower (weight, grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_lower( Weight, Group, Value )
TTestw_lower( Weight, Group, Value, sig, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTestw_sig

TTestw_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTestw_sig ( weight, grp, value [, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_sig( Weight, Group, Value )
TTestw_sig( Weight, Group, Value, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTestw_sterr

TTestw_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTestw_sterr (weight, grp, value [, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_sterr( Weight, Group, Value )
TTestw_sterr( Weight, Group, Value, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTestw_t

TTestw_t() returnerar det aggregerade t-värdet för två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ttestw_t (weight, grp, value [, eq_var])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_t( Weight, Group, Value )
TTestw_t( Weight, Group, Value, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTestw_upper

TTestw_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

Denna funktion gäller t-test med två oberoende stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTestw_upper (weight, grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTestw_upper( Weight, Group, Value )
TTestw_upper( Weight, Group, Value, sig, false )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1_conf

TTest1_conf() returnerar det aggregerade konfidensintervallvärdet för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_conf (value [, sig ])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_conf( Value )
TTest1_conf( Value, 0.005 )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1_df

TTest1_df() returnerar det aggregerade df-värdet (frihetsgrader) för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_df (value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_df( Value )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1_dif

TTest1_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_dif (value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_dif( Value )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1_lower

TTest1_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_lower (value [, sig])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_lower( Value )
TTest1_lower( Value, 0.005 )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1_sig

TTest1_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_sig (value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_sig( Value )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1_sterr

TTest1_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_sterr (value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_sterr( Value )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1_t

TTest1_t() returnerar det aggregerade t-värdet för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_t (value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_t( Value )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1_upper

TTest1_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för en värdeserie.

Funktionen gäller t-test med ett stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1_upper (value [, sig])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1_upper( Value )
TTest1_upper( Value, 0.005 )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1w_conf

TTest1w_conf() är en **numerisk** funktion som returnerar det aggregerade konfidensintervallvärdet för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_conf (weight, value [, sig ])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_conf( Weight, Value )
TTest1w_conf( Weight, Value, 0.005 )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1w_df

TTest1w_df() returnerar det aggregerade df-värdet (frihetsgrader) för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_df (weight, value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_df( Weight, Value )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1w_dif

TTest1w_dif() returnerar den aggregerade skillnaden i medelvärde för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_dif (weight, value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_dif( Weight, Value )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1w_lower

TTest1w_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_lower (weight, value [, sig ])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_lower( Weight, Value )
TTest1w_lower( Weight, Value, 0.005 )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1w_sig

TTest1w_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_sig (weight, value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_sig( Weight, Value )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1w_sterr

TTest1w_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett t-test för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_sterr (weight, value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_sterr( Weight, Value )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1w_t

TTest1w_t() returnerar det aggregerade t-värdet för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_t ( weight, value)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_t( Weight, Value )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

TTest1w_upper

TTest1w_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för en värdeserie.

Funktionen gäller t-test med ett stickprov där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
TTest1w_upper (weight, value [, sig])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickproven som ska evalueras. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.

Argument	Beskrivning
weight	Varje värde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
TTest1w_upper( Weight, Value )
TTest1w_upper( Weight, Value, 0.005 )
```

Se även:

Skapa en typisk t-test-rapport (page 383)

Z-testfunktioner

En statistisk undersökning av två populationsmedelvärden. Ett z-test med två stickprov som undersöker om två stickprov skiljer sig åt och används vanligtvis när två normalfördelningar har kända varianser och när ett experiment använder en stor urvalsstorlek.

Z-testens statistiska testfunktioner grupperas enligt typ av indataserier som gäller för funktionen.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Exempel på användning av z-test-funktioner (page 386)

Funktioner i enkolumnsformat

Följande funktioner gäller för z-test med enkla indataserier.

ztest_conf

ZTest_conf() returnerar det aggregerade z-värdet för en värdeserie.

```
ZTest_conf() returnerar det aggregerade z-värdet för en värdeserie. (value [, sigma [, sig ])
```

ztest_dif

ZTest_dif() returnerar den aggregerade skillnaden i medelvärde för ett z-test för en värdeserie.

```
ZTest_dif() returnerar den aggregerade skillnaden i medelvärde för ett z-test för en värdeserie. (value [, sigma])
```

ztest_sig

ZTest_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett z-test för en värdeserie.

```
ZTest_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett z-test för en värdeserie. (value [, sigma])
```

ztest_sterr

ZTest_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett z-test för en värdeserie.

```
ZTest_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett z-test för en värdeserie. (value [, sigma])
```

ztest_z

ZTest_z() returnerar det aggregerade z-värdet för en värdeserie.

```
ZTest_z() returnerar det aggregerade z-värdet för en värdeserie. (value [, sigma])
```

ztest_lower

ZTest_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

```
ZTest_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier. (grp, value [, sig [, eq_var]])
```

ztest_upper

ZTest_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

```
ZTest_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier. (grp, value [, sig [, eq_var]])
```

Funktioner i viktat tvåkolumnsformat

Följande funktioner gäller z-test där indataserien anges i viktat tvåkolumnsformat.

ztestw_conf

ZTestw_conf() returnerar det aggregerade z-konfidensintervallvärdet för en värdeserie.

```
ZTestw_conf() returnerar det aggregerade z-konfidensintervallvärdet för en värdeserie. (weight, value [, sigma [, sig]])
```

ztestw_dif

ZTestw_dif() returnerar den aggregerade skillnaden i medelvärde för ett z-test för en värdeserie.

```
ZTestw_dif() returnerar den aggregerade skillnaden i medelvärde för ett z-test för en värdeserie.  (weight, value [, sigma])
```

ztestw_lower

ZTestw_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

```
ZTestw_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier. (weight, value [, sigma])
```

ztestw_sig

ZTestw_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett z-test för en värdeserie.

```
ZTestw_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett z-test för en värdeserie. (weight, value [, sigma])
```

ztestw_sterr

ZTestw_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett z-test för en värdeserie.

```
ZTestw_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett z-test för en värdeserie. (weight, value [, sigma])
```

ztestw_upper

ZTestw_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

```
ZTestw_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier. (weight, value [, sigma])
```

ztestw_z

ZTestw_z() returnerar det aggregerade z-värdet för en värdeserie.

```
ZTestw_z() returnerar det aggregerade z-värdet för en värdeserie. (weight, value [, sigma])
```

ZTest_z

ZTest_z() returnerar det aggregerade z-värdet för en värdeserie.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_z(value[, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Populationsmedelvärdet 0 förutsätts. Om testet ska utföras kring ett annat medelvärde dras det medelvärdet av från stickprovsvärdena.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTest_z( Value-TestValue )
```

Se även:

Exempel på användning av z-test-funktioner (page 386)

ZTest_sig

ZTest_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett z-test för en värdeserie.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_sig(value[, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Populationsmedelvärdet 0 förutsätts. Om testet ska utföras kring ett annat medelvärde dras det medelvärdet av från stickprovsvärdena.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTest_sig(Value-TestValue)
```

Se även:

Exempel på användning av z-test-funktioner (page 386)

ZTest_dif

ZTest_dif() returnerar den aggregerade skillnaden i medelvärde för ett z-test för en värdeserie.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_dif(value[, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Populationsmedelvärdet 0 förutsätts. Om testet ska utföras kring ett annat medelvärde dras det medelvärdet av från stickprovsvärdena.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTest_dif(Value-TestValue)
```

Se även:

Exempel på användning av z-test-funktioner (page 386)

ZTest_sterr

ZTest_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett z-test för en värdeserie.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_sterr(value[, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Populationsmedelvärdet 0 förutsätts. Om testet ska utföras kring ett annat medelvärde dras det medelvärdet av från stickprovsvärdena.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTest_sterr(Value-TestValue)
```

Se även:

Exempel på användning av z-test-funktioner (page 386)

ZTest_conf

ZTest_conf() returnerar det aggregerade z-värdet för en värdeserie.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_conf(value[, sigma[, sig]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Populationsmedelvärdet 0 förutsätts. Om testet ska utföras kring ett annat medelvärde dras det medelvärdet av från stickprovsvärdena.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTest_conf(Value-TestValue)
```

Se även:

Exempel på användning av z-test-funktioner (page 386)

ZTest_lower

ZTest_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_lower (grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTest_lower( Group, Value )
ZTest_lower( Group, Value, sig, false )
```

Se även:

Exempel på användning av z-test-funktioner (page 386)

ZTest_upper

ZTest_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_upper (grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTest_upper( Group, Value )
ZTest_upper( Group, Value, sig, false )
```

Se även:

Exempel på användning av z-test-funktioner (page 386)

ZTestw_z

ZTestw_z() returnerar det aggregerade z-värdet för en värdeserie.

Funktionen gäller z-test där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTestw_z (weight, value [, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Värdena bör returneras per **value**. För stickproven förutsätts medelvärdet 0. Om testet ska utföras kring ett annat medelvärde dras det värdet av från stickprovsvärdena.
weight	Varje stickprovsvärde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTestw_z( Weight, Value-TestValue)
```

Se även:

Exempel på användning av z-test-funktioner (page 386)

ZTestw_sig

ZTestw_sig() returnerar den aggregerade tvåsidiga signifikansnivån för ett z-test för en värdeserie.

Funktionen gäller z-test där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTestw_sig (weight, value [, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Värdena bör returneras per **value**. För stickproven förutsätts medelvärdet 0. Om testet ska utföras kring ett annat medelvärde dras det värdet av från stickprovsvärdena.
weight	Varje stickprovsvärde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTestw_sig( Weight, Value-TestValue)
```

Se även:

Exempel på användning av z-test-funktioner (page 386)

ZTestw_dif

ZTestw_dif() returnerar den aggregerade skillnaden i medelvärde för ett z-test för en värdeserie.

Funktionen gäller z-test där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTestw_dif ( weight, value [, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Värdena bör returneras per **value**. För stickproven förutsätts medelvärdet 0. Om testet ska utföras kring ett annat medelvärde dras det värdet av från stickprovsvärdena.
weight	Varje stickprovsvärde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTestw_dif( Weight, Value-TestValue)
```

Se även:

Exempel på användning av z-test-funktioner (page 386)

ZTestw_sterr

ZTestw_sterr() returnerar det aggregerade standardfelet för skillnaden i medelvärde för ett z-test för en värdeserie.

Funktionen gäller z-test där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTestw_sterr (weight, value [, sigma])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Värdena bör returneras per **value**. För stickproven förutsätts medelvärdet 0. Om testet ska utföras kring ett annat medelvärde dras det värdet av från stickprovsvärdena.
weight	Varje stickprovsvärde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTestw_sterr( Weight, Value-TestValue)
```

Se även:

Exempel på användning av z-test-funktioner (page 386)

ZTestw_conf

ZTestw_conf() returnerar det aggregerade z-konfidensintervallvärdet för en värdeserie.

Funktionen gäller z-test där serien för indata anges i viktat tvåkolumnsformat.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTest_conf(weight, value[, sigma[, sig]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Populationsmedelvärdet 0 förutsätts. Om testet ska utföras kring ett annat medelvärde dras det medelvärdet av från stickprovsvärdena.
weight	Varje stickprovsvärde i **value** kan räknas en eller flera gånger beroende på ett motsvarande viktvärde i **weight**.
sigma	Om standardavvikelsen är känd kan den anges i **sigma**. Om **sigma** inte är angivet används den aktuella standardavvikelsen för stickproven.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTestw_conf( Weight, Value-TestValue)
```

Se även:

Exempel på användning av z-test-funktioner (page 386)

ZTestw_lower

ZTestw_lower() returnerar det aggregerade värdet för konfidensintervallets nedre gräns för två oberoende värdeserier.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTestw_lower (grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTestw_lower( Group, Value )
ZTestw_lower( Group, Value, sig, false )
```

Se även:

Exempel på användning av z-test-funktioner (page 386)

ZTestw_upper

ZTestw_upper() returnerar det aggregerade värdet för konfidensintervallets övre gräns för två oberoende värdeserier.

Funktionen gäller test med två oberoende stickprov.

Om funktionen används i dataladdningsskriptet itereras värdena över ett antal poster enligt vad som definierats i group by-satsen.

Om funktionen används i ett diagramuttryck itereras värdena över diagramdimensionerna.

Syntax:

```
ZTestw_upper (grp, value [, sig [, eq_var]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Stickprovsvärdena som ska evalueras. Stickprovsvärdena måste grupperas logiskt enligt vad som anges av exakt två värden i **group**. Om ett fältnamn för stickprovsvärdena inte finns med i laddningsskriptet får fältet automatiskt namnet **Value**.
grp	Detta fält innehåller namnen på var och en av de två stickprovsgrupperna. Om ett fältnamn för gruppen inte finns i laddningsskriptet får fältet automatiskt namnet **Type**.
sig	Den tvåsidiga signifikansen (p-värdet) kan anges i **sig**. Om inget annat anges beräknas **sig** som 0.025, vilket resulterar i konfidensgraden 95%.
eq_var	Om **eq_var** anges som False (0) antas separata varianser av de två stickproven. Om **eq_var** anges som True (1) antas lika varianser mellan stickproven.

Begränsningar:

Textvärden, NULL-värden och saknade värden i uttrycksvärdet resulterar i att funktionen returnerar NULL.

Exempel:

```
ZTestw_upper( Group, Value )
ZTestw_upper( Group, Value, sig, false )
```

Se även:

Exempel på användning av z-test-funktioner (page 386)

Exempel på statistiska testfunktioner

Detta avsnitt visar exempel på statistiska testfunktioner när de tillämpas på diagram och dataladdningsskript.

Exempel på användning av chi2-test-funktioner i diagram

Funktioner av typen chi2-test används för att hitta värden associerade med statistisk analys med chi2.

I det här avsnittet beskrivs hur du kan bygga visualiseringar med hjälp av stickprovsdata för att hitta värdena för fördelningstestfunktionerna med chi2 som finns i Qlik Sense. Se de individuella avsnitten för chi2-test-diagramfunktionerna för beskrivningar av syntax och argument.

Ladda data för exemplen

Det finns tre uppsättningar med stickprovsdata som beskriver de tre olika statistiska stickproven som laddas till skriptet.

Gör följande:

1. Skapa en ny app.
2. I laddningen anger du följande:

```
// Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the
top of the script.
Sample_1:
LOAD * inline [
Grp,Grade,Count
I,A,15
I,B,7
I,C,9
I,D,20
I,E,26
I,F,19
II,A,10
II,B,11
II,C,7
II,D,15
II,E,21
II,F,16
];
// Sample_2 data is pre-aggregated: If raw data is used, it must be aggregated using
count()...
Sample_2:
LOAD * inline [
Sex,Opinion,OpCount
1,2,58
1,1,11
1,0,10
2,2,35
2,1,25
2,0,23 ] (delimiter is ',');
// Sample_3a data is transformed using the crosstable statement...
Sample_3a:
crosstable(Gender, Actual) LOAD
Description,
[Men (Actual)] as Men,
[Women (Actual)] as Women;
LOAD * inline [
Men (Actual),Women (Actual),Description
58,35,Agree
11,25,Neutral
10,23,Disagree ] (delimiter is ',');
// Sample_3b data is transformed using the crosstable statement...
Sample_3b:
crosstable(Gender, Expected) LOAD
Description,
[Men (Expected)] as Men,
[Women (Expected)] as Women;
LOAD * inline [
Men (Expected),Women (Expected),Description
45.35,47.65,Agree
17.56,18.44,Neutral
16.09,16.91,Disagree ] (delimiter is ',');
// Sample_3a and Sample_3b will result in a (fairly harmless) Synthetic Key...
```

3. Klicka på  om du vill ladda data.

Skapa visualiseringar för chi2-test-diagramfunktioner

Exempel: Exempel 1

Gör följande:

1. Gå till Skriptredigeraren och klicka på för att komma till appvyn, och klicka på arket du skapade tidigare.
   Arkvyn öppnas.
2. Klicka på **Redigera ark** för att redigera arket.
3. Från **Diagram** lägger du till en tabell och från **Fält** lägger du till Grp, Grade och Count som dimensioner.
   Den här tabellen visar dina stickprovsdata.
4. Lägg till ytterligare en tabell med följande uttryck som en dimension:
   `ValueList('p','df','Chi2')`
   Nu används funktionen för syntetiska dimensioner för att skapa etiketter för dimensionerna med namnen på de tre chi2-test-funktionerna.
5. Lägg till följande uttryck i tabellen som ett mått:
   ```
   IF(ValueList('p','df','Chi2')='p',Chi2Test_p(Grp,Grade,Count),
   IF(ValueList('p','df','Chi2')='df',Chi2Test_df(Grp,Grade,Count),
   Chi2Test_Chi2(Grp,Grade,Count)))
   ```
   Detta gör att resultatvärdet för varje chi2-test-funktion sätts in i tabellen jämte dess associerade syntetiska dimension.
6. Ställ in **Talformat** för måttet på **Tal** och **3Gällande siffror**.

> *I uttrycket för måttet kan du använda följande uttryck istället:* `Pick(Match(ValueList('p','df','Chi2'),'p','df','Chi2'),Chi2Test_p(Grp,Grade,Count),Chi2Test_df(Grp,Grade,Count),Chi2Test_Chi2(Grp,Grade,Count))`

Resultat:

Resultattabellen för chi2-test-funktionerna för Exempel 1-data kommer att innehålla följande värden:

Resultattabell

p	df	Chi2
0.820	5	2.21

Exempel: Exempel 2

Gör följande:

1. I arket som du redigerade med Exempel 1 lägger du till en tabell från **Diagram** och från **Fält** lägger du till Sex, Opinion och OpCount som dimensioner.
2. Gör en kopia av resultattabellen från Exempel 1 genom att använda kommandona **Kopiera** och **Klistra in**. Redigera uttrycket i måttet och ersätt argumenten i alla tre chi2-test-funktioner med

namnen på fälten som användes i data för Exempel 2, till exempel: `Chi2Test_p (Sex,Opinion,OpCount)`

Resultat:

Resultattabellen för chi2-test-funktionerna för Exempel 2-data kommer att innehålla följande värden:

Resultattabell

p	df	Chi2
0.000309	2	16.2

Exempel: Exempel 3

Gör följande:

1. Skapa två ytterligare tabeller på samma sätt som i exemplen för data i Exempel 1 och 2. Använd följande fält som dimensioner i dimensionstabellen: Gender, Description, Actual och Expected.
2. I resultattabellen använder du namnen på fälten som användes i data för Exempel 3, till exempel: `Chi2Test_p(Gender,Description,Actual,Expected)`

Resultat:

Resultattabellen för chi2-test-funktionerna för Exempel 3-data kommer att innehålla följande värden:

Resultattabell

p	df	Chi2
0.000308	2	16.2

Exempel på användning av chi2-test-funktioner i dataladdningsskriptet

Funktioner av typen chi2-test används för att hitta värden associerade med statistisk analys med chi2. I det här avsnittet finns en beskrivning av hur man använder de chi-kvadratiska distributionstestfunktioner som finns tillgängliga i Qlik Sense-dataladdningsskriptet. Se de individuella avsnitten om chi2-test-skriptfunktionerna för beskrivningar av syntax och argument.

I det här exemplet används en tabell som innehåller antalet studenter som har fått ett betyg (A-F) för två grupper med studenter (I och II).

Data table

Group	A	B	C	D	E	F
I	15	7	9	20	26	19
II	10	11	7	15	21	16

Ladda exempeldata

Gör följande:

1. Skapa en ny app.
2. I Skriptredigeraren anger du följande:
```
// Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the top of the script.
Sample_1:
LOAD * inline [
Grp,Grade,Count
I,A,15
I,B,7
I,C,9
I,D,20
I,E,26
I,F,19
II,A,10
II,B,11
II,C,7
II,D,15
II,E,21
II,F,16
];
```
3. Klicka på  om du vill ladda data.

Du har nu laddat exempeldata.

Ladda chi2-test-funktionsvärdena

Nu laddar vi chi2-test-värdena utifrån exempeldata i en ny tabell, grupperade efter Grp.

Gör följande:

1. Lägg till följande i slutet av skriptet i Skriptredigeraren:
```
// Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the top of the script.
Chi2_table:
LOAD Grp,
Chi2Test_chi2(Grp, Grade, Count) as chi2,
Chi2Test_df(Grp, Grade, Count) as df,
Chi2Test_p(Grp, Grade, Count) as p
resident Sample_1 group by Grp;
```
2. Klicka på  om du vill ladda data.

Du har nu laddat chi2-test-värdena i en tabell med namnet Chi2_table.

Resultat

Du kan visa de resulterande chi2-test-värdena i datamodellvyn under **Förhandsgranskning**. De bör se ut så här:

Results

Grp	chi2	df	p
I	16.00	5	0.007
II	9.40	5	0.094

Skapa en typisk t-test-rapport

En typisk t-test-rapport kan innehålla tabeller med **Group Statistics** och **Independent Samples Test**-resultat.

I följande avsnitt kommer vi att bygga upp de här tabellerna med hjälp av Qlik Senset-test-funktioner som tillämpas på två fristående grupper av stickprov, Observation och Comparison. Motsvarande tabeller för dessa stickprov skulle se ut så här:

Statistik för gruppen

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Independent Sample Test

Fristående stickprov

Type	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval of the Difference (Lower)	95% Confidence Interval of the Difference (Upper)
Equal Variance not Assumed	3.534	37.116717335823	0.001	15.2	4.30101	6.48625	23.9137
Equal Variance Assumed	3.534	38	0.001	15.2	4.30101	6.49306	23.9069

Ladda exempeldata

Gör följande:

1. Skapa en ny app med ett nytt ark och öppna arket.
2. Ange följande i Skriptredigeraren:

```
Table1:
crosstable LOAD recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
```

```
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2 ] (delimiter is '|');
```

I detta laddningsskript är **recno()** inkluderat eftersom **crosstable** kräver tre argument. Det innebär att **recno()** helt enkelt ger ett extra argument, i det här fallet ett ID för varje rad. Utan det skulle **Comparison**-stickprovsvärdena inte läsas in.

3. Klicka på om du vill ladda data.

Skapa tabellen Group Statistics

Gör följande:

1. Gå till Skriptredigeraren och klicka på för att komma till appvyn, och klicka på arket du skapade tidigare.
   Arkvyn öppnas.
2. Klicka på **Redigera ark** för att redigera arket.
3. Från **Diagram** lägger du till en tabell och från **Fields** lägger du till följande uttryck som mått:

   Exempel på uttryck

Etikett	Uttryck
N	Count(Value)
Mean	Avg(Value)
Standard Deviation	Stdev(Value)
Standard Error Mean	Sterr(Value)

4. Lägg till Type som dimension i tabellen.
5. Klicka på **Sortering** och flytta Type till toppen av sorteringslistan.

   Resultat:

   En Group Statistics-tabell för dessa stickprov skulle se ut så här:

   Statistik för gruppen

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Skapa tabellen Two Independent Sample Student's T-test

Gör följande:

1. Klicka på **Redigera ark** för att redigera arket.
2. Lägg till följande uttryck som en dimension i tabellen. `=ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1))`
3. Från **Diagram** lägger du till en tabell med följande uttryck som mått:

Exempel på uttryck

Etikett	Uttryck
conf	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_conf(Type, Value),TTest_conf(Type, Value, 0))
t	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_t(Type, Value),TTest_t(Type, Value, 0))
df	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_df(Type, Value),TTest_df(Type, Value, 0))
Sig. (2-tailed)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_sig(Type, Value),TTest_sig(Type, Value, 0))
Mean Difference	TTest_dif(Type, Value)
Standard Error Difference	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_sterr(Type, Value),TTest_sterr(Type, Value, 0))
95% Confidence Interval of the Difference (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 of the Difference (Upper)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_upper(Type, Value,(1-(95)/100)/2),TTest_upper (Type, Value,(1-(95)/100)/2, 0))

Resultat:

Fristående stickprov

Type	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval of the Difference (Lower)	95% Confidence Interval of the Difference (Upper)
Equal Variance not Assumed	3.534	37.116717335823	0.001	15.2	4.30101	6.48625	23.9137

Type	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval of the Difference (Lower)	95% Confidence Interval of the Difference (Upper)
Equal Variance Assumed	3.534	38	0.001	15.2	4.30101	6.49306	23.9069

Exempel på användning av z-test-funktioner

z-test-funktionerna används för att hitta värden associerade med statistiska analyser av z-test-typ för stora uppsättningar av stickprovsdata, vanligtvis större än 30, och där du vet att varians förekommer.

I det här avsnittet beskrivs hur du kan bygga visualiseringar med hjälp av sticksprovsdata för att hitta värdena för z-test-funktionerna som finns i Qlik Sense. Se de individuella avsnitten för z-test-diagramfunktionerna för beskrivningar av syntax och argument.

Ladda exempeldata

Stickprovsdata som används här är samma som används i exemplen för t-test-funktionen. Datauppsättningen skulle normalt sett anses för liten för att använda vid z-test-analyser, men den är tillräcklig för att illustrera användningen av de olika z-test-funktionerna i Qlik Sense.

Gör följande:

1. Skapa en ny app med ett nytt ark och öppna arket.

   Om du skapade en app för t-test -funktionerna, kunde du använda den och skapa ett nytt ark för de här funktionerna.

2. I Skriptredigeraren anger du följande:

```
Table1:
crosstable LOAD recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
```

```
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2 ] (delimiter is '|');
```

I detta laddningsskript är **recno()** inkluderat eftersom **crosstable** kräver tre argument. Det innebär att **recno()** helt enkelt ger ett extra argument, i det här fallet ett ID för varje rad. Utan det skulle **Comparison**-stickprovsvärdena inte läsas in.

3. Klicka på om du vill ladda data.

Skapa visualiseringar för z-test-diagramfunktioner

Gör följande:

1. Gå till Skriptredigeraren och klicka på för att komma till appvyn. Klicka sedan på det ark som du skapade när du laddade data.
   Arkvyn öppnas.
2. Klicka på **Redigera ark** för att redigera arket.
3. Från **Diagram** lägger du till en tabell och från **Fält** lägger du till Type som en dimension.
4. Lägg till följande uttryck i tabellen som ett mått.

Exempel på uttryck

Etikett	**Uttryck**
ZTest Conf	ZTest_conf(Value)
ZTest Dif	ZTest_dif(Value)
ZTest Sig	ZTest_sig(Value)
ZTest Sterr	ZTest_sterr(Value)
ZTest Z	ZTest_z(Value)

*Du kanske måste justera talformatet för måtten för att se meningsfulla värden. Tabellen blir enklare att läsa om du ställer in talformat på de flesta av måtten till **Tal > Enkel**, i stället för **Auto**. Men för exempelvis ZTest Sig använder du talformatet: **Anpassa** och justerar sedan formatsträngen till **# ##**.*

Resultat:

Den resulterande tabellen för z-test-funktionerna för exempeldata kommer att innehålla följande värden:

Resultattabell

Type	ZTest Conf	ZTest Dif	ZTest Sig	ZTest Sterr	ZTest Z
Comparison	6.40	11.95	0.000123	3.27	3.66
Value	5.48	27.15	0.001	2.80	9.71

Skapa visualiseringar för z-testw-diagramfunktioner

z-testw-funktionerna används när indataserien visas i viktat tvåkolumnsformat. Uttrycken kräver ett värde för weight. I exemplen här används värdet 2 genomgående, men du kan använda ett uttryck som ska definiera ett värde för weight för varje observation.

Exempel och resultat:

Med samma exempeldata och talformat som för z-test-funktionerna, kommer den resulterande tabellen för z-testw-funktionerna att innehålla följande värden:

Resultattabell

Type	ZTestw Conf	ZTestw Dif	ZTestw Sig	ZTestw Sterr	ZTestw Z
Comparison	3.53	2.95	5.27e-005	1.80	3.88
Value	2.97	34.25	0	4.52	20.49

Strängaggregeringsfunktioner

Den här delen beskriver strängrelaterade aggregeringsfunktioner.

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

Strängaggregeringsfunktioner i dataladdningsskriptet

Concat

Concat() används för att kombinera strängvärden. Denna skriptfunktion returnerar den aggregerade sträng-konkateneringen av alla värden i uttryck itererat över ett antal poster enligt vad som definierats i en **group by**-sats.

```
Concat ([ distinct ] expression [, delimiter [, sort-weight]])
```

FirstValue

FirstValue() returnerar värdet som laddades först från posterna som definierats av uttrycket, sorterade efter en **group by**-sats.

Funktionen finns endast som skriptfunktion.

```
FirstValue  (expression)
```

LastValue

LastValue() returnerar värdet som laddades sist från posterna som definierats av uttrycket, sorterade efter en **group by**-sats.

Funktionen finns endast som skriptfunktion.

```
LastValue  (expression)
```

MaxString

MaxString() hittar strängvärden i uttrycket och returnerar det senaste textvärdet alfabetiskt sorterat under ett antal poster, som definieras av en **group by**-sats.

```
MaxString (expression )
```

MinString

MinString() hittar strängvärden i uttrycket och returnerar det första textvärdet alfabetiskt sorterat under ett antal poster, som definieras av en **group by**-sats.

```
MinString (expression )
```

Strängaggregeringsfunktioner i diagram

Följande diagramfunktioner kan användas för att aggregera strängar i diagram.

Concat

Concat() används för att kombinera strängvärden. Funktionen returnerar den aggregerade strängkonkateneringen av alla värden av uttrycket utvärderat över varje dimension.

```
Concat - diagramfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]
string[, delimiter[, sort_weight]])
```

MaxString

MaxString() finner strängvärden i uttrycket eller fältet och returnerar det senaste textvärdet i alfabetisk sorteringsordning.

```
MaxString - diagramfunktion({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MinString

MinString() finner strängvärden i uttrycket eller fältet och returnerar det första textvärdet i alfabetisk sorteringsordning.

```
MinString - diagramfunktion({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Concat

Concat() används för att kombinera strängvärden. Denna skriptfunktion returnerar den aggregerade sträng-konkateneringen av alla värden i uttryck itererat över ett antal poster enligt vad som definierats i en **group by**-sats.

Syntax:

```
Concat ([ distinct ] string [, delimiter [, sort-weight]])
```

Returnerad datatyp: sträng

Argument:

Det uttryck eller fält som innehåller den sträng som ska behandlas.

Argument

Argument	Beskrivning
string	Det uttryck eller fält som innehåller den sträng som ska behandlas.
delimiter	Varje värde kan avgränsas med den sträng som finns i delimiter.
sort-weight	Konkanteneringens ordning kan avgöras av värdet på dimensionen **sort-weight**, om sådant finns, där strängen motsvarar det lägsta värde som visas först i konkateneringen.
distinct	Om predikatet **distinct** förekommer framför uttrycket ignoreras alla dubbletter.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Exempel och resultat

Exempel	Resultat	Resultat som har lagts till på ett ark

`TeamData:`
`LOAD * inline [`
`SalesGroup|Team|Date|Amount`
`East|Gamma|01/05/2013|20000`
`East|Gamma|02/05/2013|20000`
`West|Zeta|01/06/2013|19000`
`East|Alpha|01/07/2013|25000`
`East|Delta|01/08/2013|14000`
`West|Epsilon|01/09/2013|17000`
`West|Eta|01/10/2013|14000`
`East|Beta|01/11/2013|20000`
`West|Theta|01/12/2013|23000`
`] (delimiter is '|');`
 	
`Concat1:`
`LOAD SalesGroup,Concat(Team) as TeamConcat1`
`Resident TeamData Group By SalesGroup;`
	SalesGroup
	East
	West
		TeamConcat1
		AlphaBetaDeltaGammaGamma
		EpsilonEtaThetaZeta

Givet att **TeamData**-tabellen är laddad som i föregående exempel:
 	
`LOAD SalesGroup,Concat(distinct Team,'-')`
`as TeamConcat2 Resident TeamData Group By`
`SalesGroup;`
	SalesGroup
	East
	West
		TeamConcat2
		Alpha-Beta-Delta-Gamma
		Epsilon-Eta-Theta-Zeta

Exempel	Resultat	Resultat som har lagts till på ett ark
Givet att **TeamData**-tabellen är laddad som i föregående exempel. Eftersom argumentet för **sort-weight** läggs till ordnas resultaten efter värdet på dimensionen Amount: `LOAD SalesGroup,Concat(distinct Team,'-',Amount) as TeamConcat2 Resident TeamData Group By SalesGroup;`	SalesGroup East West	TeamConcat2 Delta-Beta-Gamma-Alpha Eta-Epsilon-Zeta-Theta

Concat - diagramfunktion

Concat() används för att kombinera strängvärden. Funktionen returnerar den aggregerade strängkonkateneringen av alla värden av uttrycket utvärderat över varje dimension.

Syntax:

```
Concat({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} string[, delimiter [, sort_weight]])
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
string	Det uttryck eller fält som innehåller den sträng som ska behandlas.
delimiter	Varje värde kan avgränsas med den sträng som finns i delimiter.
sort-weight	Konkanteneringens ordning kan avgöras av värdet på dimensionen **sort-weight**, om sådant finns, där strängen motsvarar det lägsta värde som visas först i konkateneringen.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om predikatet **DISTINCT** förekommer framför funktionsargumenten ignoreras alla dubbletter som evaluerats utifrån funktionsargumenten.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {,fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel och resultat:

Results table

SalesGroup	Amount	Concat(Team)	Concat(TOTAL <SalesGroup> Team)
East	25000	Alpha	AlphaBetaDeltaGammaGamma
East	20000	BetaGammaGamma	AlphaBetaDeltaGammaGamma
East	14000	Delta	AlphaBetaDeltaGammaGamma
West	17000	Epsilon	EpsilonEtaThetaZeta
West	14000	Eta	EpsilonEtaThetaZeta
West	23000	Theta	EpsilonEtaThetaZeta
West	19000	Zeta	EpsilonEtaThetaZeta

Exempel på funktioner

Exempel	Resultat
`Concat(Team)`	Tabellen är konstruerad av dimensionerna SalesGroup och Amount, samt variationer av måttet Concat(Team). Bortse från totalvärderesultatet och observera att trots att det finns data för åtta värden av Team spridda över två värden av SalesGroup, är det enda resultatet av måttet Concat(Team) som konkatenerar fler än ett Team-strängvärde i tabellen raden med dimensionen Amount 20 000, vilket ger resultatet BetaGammaGamma. Detta beror på att det finns tre värden för Amount 20 000 i indata. Alla andra resultat förblir okonkatenerade när måttet omfattar dimensionerna eftersom det bara finns ett värde av Team för varje kombination av SalesGroup och Amount.
`Concat (DISTINCT Team,', ')`	Beta, Gamma. eftersom DISTINCT-kvalificeraren betyder att dublettresultatet för Gamma ignoreras. Dessutom definieras avgränsarargumentet som ett komma följt av mellanslag.
`Concat (TOTAL <SalesGroup> Team)`	Alla strängvärden av Team konkateneras om TOTAL-kvalificeraren används. Med fältvalet <SalesGroup> angivet delar detta resultatet i två värden av dimensionen SalesGroup. För SalesGroupEast är resultaten AlphaBetaDeltaGammaGamma. För SalesGroupWest är resultaten EpsilonEtaThetaZeta.
`Concat (TOTAL <SalesGroup> Team,';', Amount)`	Om du lägger till argumentet för **sort-weight**: Amount ordnas resultaten efter värdet på dimensionen Amount. Resultaten blir DeltaBetaGammaGammaAlpha och EtaEpsilonZEtaTheta.

Data som används i exemplet:

```
TeamData:
LOAD * inline [
SalesGroup|Team|Date|Amount
East|Gamma|01/05/2013|20000
East|Gamma|02/05/2013|20000
West|Zeta|01/06/2013|19000
```

```
East|Alpha|01/07/2013|25000
East|Delta|01/08/2013|14000
West|Epsilon|01/09/2013|17000
West|Eta|01/10/2013|14000
East|Beta|01/11/2013|20000
West|Theta|01/12/2013|23000
] (delimiter is '|');
```

FirstValue

FirstValue() returnerar värdet som laddades först från posterna som definierats av uttrycket, sorterade efter en **group by**-sats.

Funktionen finns endast som skriptfunktion.

Syntax:

```
FirstValue ( expr)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Begränsningar:

Om inget textvärde hittas returneras NULL.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Resulterande data

Exempel	Resultat	Resultat på ett ark
TeamData: LOAD * inline [SalesGroup|Team|Date|Amount East|Gamma|01/05/2013|20000 East|Gamma|02/05/2013|20000 West|Zeta|01/06/2013|19000 East|Alpha|01/07/2013|25000 East|Delta|01/08/2013|14000 West|Epsilon|01/09/2013|17000 West|Eta|01/10/2013|14000 East|Beta|01/11/2013|20000 West|Theta|01/12/2013|23000] (delimiter is '|'); FirstValue1: LOAD SalesGroup,FirstValue(Team) as FirstTeamLoaded Resident TeamData Group By SalesGroup;	SalesGroup East West	FirstTeamLoaded Gamma Zeta

LastValue

LastValue() returnerar värdet som laddades sist från posterna som definierats av uttrycket, sorterade efter en **group by**-sats.

Funktionen finns endast som skriptfunktion.

Syntax:

```
LastValue ( expr )
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Begränsningar:

Om inget textvärde hittas returneras NULL.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i vår app för att se resultatet.

Om du vill få samma utseende som i resultatkolumnen nedan växlar du från automatisk till egen visning i egenskapspanelen under Sortering. Avmarkera sedan numerisk och alfabetisk sortering.

Exempel	Resultat	Resultat med anpassad sortering
TeamData: LOAD * inline [SalesGroup|Team|Date|Amount East|Gamma|01/05/2013|20000 East|Gamma|02/05/2013|20000 West|Zeta|01/06/2013|19000 East|Alpha|01/07/2013|25000 East|Delta|01/08/2013|14000 West|Epsilon|01/09/2013|17000 West|Eta|01/10/2013|14000 East|Beta|01/11/2013|20000 West|Theta|01/12/2013|23000] (delimiter is '|'); LastValue1: LOAD SalesGroup,LastValue(Team) as LastTeamLoaded Resident TeamData Group By SalesGroup;	SalesGroup East West	LastTeamLoaded Beta Theta

MaxString

MaxString() hittar strängvärden i uttrycket och returnerar det senaste textvärdet alfabetiskt sorterat under ett antal poster, som definieras av en **group by**-sats.

Syntax:

```
MaxString ( expr )
```

Returnerad datatyp: dual

Argument:

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Begränsningar:

Om inget textvärde hittas returneras NULL.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Exempel	Resultat	
TeamData: LOAD * inline [SalesGroup|Team|Date|Amount East|Gamma|01/05/2013|20000 East|Gamma|02/05/2013|20000 West|Zeta|01/06/2013|19000 East|Alpha|01/07/2013|25000 East|Delta|01/08/2013|14000 West|Epsilon|01/09/2013|17000 West|Eta|01/10/2013|14000 East|Beta|01/11/2013|20000 West|Theta|01/12/2013|23000] (delimiter is '|'); Concat1: LOAD SalesGroup,MaxString(Team) as MaxString1 Resident TeamData Group By SalesGroup;	SalesGroup East West	MaxString1 Gamma Zeta
Givet att **TeamData**-tabellen är laddad som i föregående exempel och ditt dataladdningsskript har SET-satsen: SET DateFormat='DD/MM/YYYY';: LOAD SalesGroup,MaxString(Date) as MaxString2 Resident TeamData Group By SalesGroup;	SalesGroup East West	MaxString2 01/11/2013 01/12/2013

MaxString - diagramfunktion

MaxString() finner strängvärden i uttrycket eller fältet och returnerar det senaste textvärdet i alfabetisk sorteringsordning.

Syntax:

```
MaxString({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.

Argument	Beskrivning
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Begränsningar:

Om uttrycket inte innehåller några värden med strängrepresentation, returneras NULL.

Exempel och resultat:

Resultattabell

SalesGroup	Amount	MaxString(Team)	MaxString(Date)
East	14000	Delta	2013/08/01
East	20000	Gamma	2013/11/01
East	25000	Alpha	2013/07/01
West	14000	Eta	2013/10/01
West	17000	Epsilon	2013/09/01
West	19000	Zeta	2013/06/01
West	23000	Theta	2013/12/01

Exempel på funktioner

Exempel	Resultat
`MaxString (Team)`	Det finns tre värden för 20000 för dimensionen Amount: två av Gamma (på olika datum) och en av Beta. Resultatet av måttet MaxString (Team) är därför Gamma eftersom detta är det högsta värdet i den sorterade strängen.
`MaxString (Date)`	2013/11/01 är det största Date-värdet av de tre som är associerade med dimensionen Amount. Detta förutsätter att ditt skript har SET-satsen `SET DateFormat='YYYY-MM-DD';'`

Data som används i exemplet:

```
TeamData:
LOAD * inline [
SalesGroup|Team|Date|Amount
East|Gamma|01/05/2013|20000
East|Gamma|02/05/2013|20000
West|Zeta|01/06/2013|19000
East|Alpha|01/07/2013|25000
East|Delta|01/08/2013|14000
```

```
West|Epsilon|01/09/2013|17000
West|Eta|01/10/2013|14000
East|Beta|01/11/2013|20000
West|Theta|01/12/2013|23000
] (delimiter is '|');
```

MinString

MinString() hittar strängvärden i uttrycket och returnerar det första textvärdet alfabetiskt sorterat under ett antal poster, som definieras av en **group by**-sats.

Syntax:

```
MinString ( expr )
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.

Begränsningar:

Om inget textvärde hittas returneras NULL.

Exempel och resultat:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Resulterande data

Exempel	Resultat	
TeamData: LOAD * inline [SalesGroup|Team|Date|Amount East|Gamma|01/05/2013|20000 East|Gamma|02/05/2013|20000 West|Zeta|01/06/2013|19000 East|Alpha|01/07/2013|25000 East|Delta|01/08/2013|14000 West|Epsilon|01/09/2013|17000 West|Eta|01/10/2013|14000 East|Beta|01/11/2013|20000 West|Theta|01/12/2013|23000] (delimiter is '|'); Concat1: LOAD SalesGroup,MinString(Team) as MinString1 Resident TeamData Group By SalesGroup;	SalesGroup East West	MinString1 Alpha Epsilon

Exempel	Resultat	
Givet att **TeamData**-tabellen är laddad som i föregående exempel och ditt dataladdningsskript har SET-satsen: `SET DateFormat='DD/MM/YYYY';`: `LOAD SalesGroup,MinString(Date) as MinString2 Resident TeamData Group By SalesGroup;`	SalesGroup East West	MinString2 01/05/2013 01/06/2013

MinString - diagramfunktion

MinString() finner strängvärden i uttrycket eller fältet och returnerar det första textvärdet i alfabetisk sorteringsordning.

Syntax:

```
MinString({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
TOTAL	Om ordet **TOTAL** står före funktionsargumenten görs beräkningen över alla valbara värden givet de aktuella valen, och inte bara sådana som rör det aktuella dimensionsvärdet, det vill säga att diagramdimensionerna ignoreras. Genom att använda **TOTAL [<fld {.fld}>]**, där kvalificeraren **TOTAL** följs av en lista med ett eller flera fältnamn som en delmängd av diagramdimensionens variabler, skapar du en delmängd av de totala möjliga värdena.

Exempel och resultat:

Exempeldata

SalesGroup	Amount	MinString(Team)	MinString(Date)
East	14000	Delta	2013/08/01
East	20000	Beta	2013/05/01
East	25000	Alpha	2013/07/01
West	14000	Eta	2013/10/01

SalesGroup	Amount	MinString(Team)	MinString(Date)
West	17000	Epsilon	2013/09/01
West	19000	Zeta	2013/06/01
West	23000	Theta	2013/12/01

Exempel på funktioner

Exempel	Resultat
`MinString (Team)`	Det finns tre värden för 20000 för dimensionen Amount: två av Gamma (på olika datum) och en av Beta. Resultatet av måttet MinString (Team) är därför Beta eftersom detta är det första värdet i den sorterade strängen.
`MinString (Date)`	2013/11/01 är det tidigaste Date-värdet av de tre som är associerade med dimensionen Amount. Detta förutsätter att ditt skript har SET-satsen `SET DateFormat='YYYY-MM-DD';'`

Data som används i exemplet:

```
TeamData:
LOAD * inline [
SalesGroup|Team|Date|Amount
East|Gamma|01/05/2013|20000
East|Gamma|02/05/2013|20000
West|Zeta|01/06/2013|19000
East|Alpha|01/07/2013|25000
East|Delta|01/08/2013|14000
West|Epsilon|01/09/2013|17000
West|Eta|01/10/2013|14000
East|Beta|01/11/2013|20000
West|Theta|01/12/2013|23000
] (delimiter is '|');
```

Syntetiska dimensionsfunktioner

En syntetisk dimension skapas i appen av värden som genererats från de syntetiska dimensionsfunktionerna och inte direkt från fält i datamodellen. När värden som genererats från en syntetisk dimensionsfunktion används i ett diagram som en beräknad dimension skapar detta en syntetisk dimension. Syntetiska dimensioner gör att du kan skapa till exempel diagram med dimensioner med värden som kommer från dina data, det vill säga, dynamiska dimensioner.

Syntetiska dimensioner påverkas inte av urval.

Följande syntetiska dimensionsfunktioner kan användas i diagram.

ValueList

ValueList() returnerar en uppsättning listade värden, som när de används i en beräknad dimension bildar en syntetisk dimension.

```
ValueList - diagramfunktion (v1 {, Expression})
```

ValueLoop

ValueLoop() returnerar en uppsättning itererade värden, som när de används i en beräknad dimension bildar en syntetisk dimension.

```
ValueLoop - diagramfunktion(from [, to [, step ]])
```

ValueList - diagramfunktion

ValueList() returnerar en uppsättning listade värden, som när de används i en beräknad dimension bildar en syntetisk dimension.

*I diagram som har en syntetisk dimension skapad med **ValueList**-funktionen är det möjligt att referera till det dimensionsvärde som motsvarar en specifik uttryckscell genom att åberopa **ValueList**-funktionen med samma parametrar i diagramuttrycket. Funktionen kan självfallet användas överallt i layouten. Utöver användningen i syntetiska dimensioner är den endast meningsfull inuti en aggregeringsfunktion.*

Syntetiska dimensioner påverkas inte av urval.

Syntax:

```
ValueList(v1 {,...})
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
v1	Statiskt värde (vanligtvis en sträng, men kan vara ett tal).
{,...}	Valbar lista över statiska värden.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`ValueList ('Number of Orders', 'Average Order Size', 'Total Amount')`	När det används för att skapa en dimension i en tabell till exempel, resulterar det här i tre strängvärden som radetiketter i tabellen. Dessa kan refereras i ett uttryck.

Exempel	Resultat
`=IF( ValueList ('Number of Orders', 'Average Order Size', 'Total Amount') = 'Number of Orders', count (SaleID), IF( ValueList ('Number of Orders', 'Average Order Size', 'Total Amount') = 'Average Order Size', avg (Amount), sum (Amount) ))`	Det här uttrycket tar värdena från den skapade dimensionen och refererar dem i en nästlad IF-sats som indata för tre aggregeringsfunktioner: **ValueList()** Created dimension | Year | Added expression **522.00** Number of Orders | 2012 | 5.00 Number of Orders | 2013 | 7.00 Average Order Size | 2012 | 13.20 Average Order Size | 2013 | 15.43 Total Amount | 2012 | 66.00 Total Amount | 2013 | 108.00

Data som används i exempel:

```
SalesPeople:
LOAD * INLINE [
SaleID|SalesPerson|Amount|Year
1|1|12|2013
2|1|23|2013
3|1|17|2013
4|2|9|2013
5|2|14|2013
6|2|29|2013
7|2|4|2013
8|1|15|2012
9|1|16|2012
10|2|11|2012
11|2|17|2012
12|2|7|2012
] (delimiter is '|');
```

ValueLoop - diagramfunktion

ValueLoop() returnerar en uppsättning itererade värden, som när de används i en beräknad dimension bildar en syntetisk dimension.
De genererade värdena börjar med **from**-värdet och avslutas med **to**-värdet inklusive de mellanliggande värdena i stegvis ordning.

*I diagram som har en syntetisk dimension skapad med **ValueLoop**-funktionen är det möjligt att referera till det dimensionsvärde som motsvarar en specifik uttryckscell genom att åberopa **ValueLoop**-funktionen med samma parametrar i diagramuttrycket. Funktionen kan självfallet användas överallt i layouten. Utöver användningen i syntetiska dimensioner är den endast meningsfull inuti en aggregeringsfunktion.*

Syntetiska dimensioner påverkas inte av urval.

Syntax:

```
ValueLoop(from [, to [, step ]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
from	Startvärdet i den värdeuppsättning som ska genereras.
to	Slutvärdet i den värdeuppsättning som ska genereras.
step	Storleken på stegen mellan värdena.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`ValueLoop (1, 10)`	Detta skapar en dimension i en tabell till exempel, som kan användas för syften som numrerade etiketter. Exemplet här resulterar i värden mellan 1 och 10. Dessa värden kan sedan refereras i ett uttryck.
`ValueLoop (2, 10,2)`	Detta exempel resulterar i värdena 2, 4, 6, 8 och 10 eftersom argumentet step har värdet 2.

Nästlade aggregeringar

Ibland kan man behöva applicera en aggregering på resultatet av en annan aggregering. Detta kallas nästlade aggregeringar.

Det går inte att nästla aggregeringar i de flesta diagramuttrycken. Det går dock att nästla aggregeringar om du använder kvalificeraren **TOTAL** i den inre aggregeringsfunktionen.

Det går inte att nästla mer än 100 nivåer.

Nästlade aggregeringar med TOTAL-kvalificeraren

Exempel:

Säg att du till exempel vill beräkna summan för fältet **Sales** men bara vill inkludera transaktioner med ett **OrderDate** som motsvarar föregående år. Föregående år kan nås via aggregeringsfunktionen **Max(TOTAL** `Year(OrderDate)`**)**.

Följande aggregering skulle returnera det önskade resultatet:

```
Sum(If(Year(OrderDate)=Max(TOTAL Year(OrderDate)), Sales))
```

Qlik Sense kräver att kvalificeraren **TOTAL** inkluderas i den här typen av nästling. Det krävs för den önskade jämförelsen. Den här sortens nästling är ganska vanlig och utgör en god vana.

Se även:

Aggr - diagramfunktion (page 404)

5.3 Aggr - diagramfunktion

Aggr() returnerar en uppsättning värden för uttrycket beräknade över den angivna dimensionen eller de angivna dimensionerna. Exempelvis det maximala värdet av försäljningen, per kund, per region.

Aggr-funktionen används för nästlade aggregeringar där den första parametern (den inre aggregeringen) beäknas en gång per dimensionsvärde. Dimensionerna anges i den andra parametern (och efterföljande parametrar).

Dessutom ska **Aggr**-funktionen omslutas av en yttre aggregeringsfunktion där matrisen med resultat från **Aggr**-funktionen används som indata till den aggregering som den är nästlad i.

Syntax:

```
Aggr({SetExpression}[DISTINCT] [NODISTINCT ] expr, StructuredParameter{,
StructuredParameter})
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Ett uttryck som består av en aggregeringsfunktion. Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet.

Argument	Beskrivning
StructuredParameter	StructuredParameter består av en dimension och ett (valfritt) sorteringskriterium i formatet: `(Dimension(Sort-type, Ordering))` Dimensionen är ett enda fält och kan inte vara ett uttryck. Dimensionen används för att bedöma vilken uppsättning med värden som Aggr-uttrycket ska beräknas på. Om sorteringskriterier används, sorteras den uppsättning med värden som skapas av Aggr-funktionen och som beräknats för dimensionen. Detta är viktigt när sorteringsordningen påverkar resultatet av det uttryck som Aggr-funktionen är innesluten i. Mer information om hur du använder sorteringskriterier finns i Lägga till sorteringskriterier till dimensionen i den strukturerade parametern.
SetExpression	Som standard kommer aggregeringsfunktionen att aggregera över den uppsättning möjliga poster som definierats av urvalet. En alternativ uppsättning poster kan definieras med ett uttryck för set-analys.
DISTINCT	Om uttrycksargumentet föregås av kvalificeraren **distinct**, eller om ingen kvalificerare alls används, kommer varje distinkt kombination av dimensionsvärden endast att generera ett returvärde. Det är normalt så här aggregeringar skapas - varje distinkt kombination av dimensionsvärden ger en rad i diagrammet.
NODISTINCT	Om uttrycksargumentet föregås av kvalificeraren **nodistinct** kan varje kombination av dimensionsvärden generera mer än ett returvärde, beroende på den underliggande datastrukturen. Om det bara finns en dimension, returnerar funktionen **aggr** en uppsättning med samma antal element som det finns rader i källdata.

Grundläggande aggregeringsfunktioner, såsom **Sum**, **Min** och **Avg**, returnerar ett enda numeriskt värde, medan funktionen Aggr() kan jämföras med att skapa en temporär uppsättning med resultat (en virtuell tabell), över vilken en annan aggregering kan göras. Detta kan exempelvis göras genom att beräkna ett genomsnittligt försäljningsvärde genom att summera försäljningen efter kund i en **Aggr()**-sats och sedan beräkna genomsnittet av de summerade resultaten: **Avg(TOTAL Aggr(Sum(Sales),Customer))**.

Använd Aggr()-funktionen i beräknade dimensioner om du vill skapa kapslade diagramaggregeringar på flera nivåer.

Begränsningar:

Varje dimension i en Aggr()-funktion måste bestå av ett enda fält. Den får inte vara ett uttryck (beräknad dimension).

Lägga till sorteringskriterier till dimensionen i den strukturerade parametern

I sin grundläggande form är argumentet StructuredParameter i syntaxen för Aggr-funktionen en enda dimension. Uttrycket: Aggr(Sum(Sales, Month)) hittar det totala värdet för försäljning per månad. Om det innesluts i en annan aggregeringsfunktion kan det dock ge oväntade resultat om inte sorteringskriterier används. Det beror på att en del dimensioner kan sorteras numeriskt eller alfabetiskt, och så vidare.

I StructuredParameter-argumentet i Aggr-funktionen kan du ange sorteringskriterier för dimensionen i dina uttryck. På så sätt fastställer du en sorteringsordning för den virtuella tabell som produceras av Aggr-funktionen.

Argumentet StructuredParameter har följande syntax:

```
(FieldName, (Sort-type, Ordering))
```

Strukturerade parametrar kan kapslas:

```
(FieldName, (FieldName2, (Sort-type, Ordering)))
```

Sorteringstypen kan vara: NUMERIC, TEXT, FREQUENCY eller LOAD_ORDER.

Ordningstyperna som är associerade med varje sorteringstyp är som följer:

Tillåtna ordningstyper

Sorteringstyp	Tillåtna ordningstyper
NUMERIC	ASCENDING, DESCENDING eller REVERSE
TEXT	ASCENDING, A2Z, DESCENDING, REVERSE eller Z2A
FREQUENCY	DESCENDING, REVERSE eller ASCENDING
LOAD_ORDER	ASCENDING, ORIGINAL, DESCENDING eller REVERSE

Ordningstyperna REVERSE och DESCENDING är motsvarigheter.

För sorteringstypen TEXT är ordningstyperna ASCENDING och A2Z motsvarigheter, och DESCENDING, REVERSE och Z2A är motsvarigheter.

För sorteringstypen LOAD_ORDER är ordningstyperna ASCENDING och ORIGINAL motsvarigheter.

Exempel: Diagramuttryck som använder Aggr

Exempel – diagramuttryck

Diagramuttryck exempel 1

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplet nedan.

```
ProductData:  LOAD * inline [ Customer|Product|UnitSales|UnitPrice Astrida|AA|4|16
Astrida|AA|10|15 Astrida|BB|9|9 Betacab|BB|5|10 Betacab|CC|2|20 Betacab|DD|25|25
Canutility|AA|8|15 Canutility|CC|0|19 ] (delimiter is '|');
```

Diagramuttryck

Skapa en KPI-visualisering i ett Qlik Sense-ark. Lägg till följande uttryck i KPI-visualiseringen som ett mått:

```
Avg(Aggr(Sum(UnitSales*UnitPrice), Customer))
```

Resultat

376.7

Förklaring

Uttrycket `Aggr(Sum(UnitSales*UnitPrice), Customer)` hittar det totala värdet för försäljning utifrån **Customer**, och returnerar en uppsättning med värden: 295, 715 och 120 för de tre **Customer**-värdena.

Vi har i praktiken skapat en tillfällig lista med värden utan att ha behövt skapa en explicit tabell eller kolumn som innehåller dessa värden.

De här tre värdena används som indata för funktionen **Avg()** som räknar ut medelvärdet för försäljning, 376.7.

Diagramuttryck exempel 2

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplet nedan.

```
ProductData:  LOAD * inline [ Customer|Product|UnitSales|UnitPrice Astrida|AA|4|16
Astrida|AA|10|15 Astrida|BB|10|15 Astrida|BB|9|9 Betacab|BB|5|10 Betacab|BB|7|12
Betacab|CC|2|22 Betacab|CC|4|20 Betacab|DD|25|25 Canutility|AA|8|15 Canutility|AA|5|11
Canutility|CC|0|19 ] (delimiter is '|');
```

Diagramuttryck

Skapa en tabellvisualisering i ett Qlik Sense-ark med **Customer**, **Product**, **UnitPrice** och **UnitSales** som dimensioner. Lägg till följande uttryck i tabellen som ett mått:

```
Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
```

Resultat

Customer	Product	UnitPrice	UnitSales	Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
Astrida	AA	15	10	16
Astrida	AA	16	4	16
Astrida	BB	9	9	15

Customer	Product	UnitPrice	UnitSales	Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
Astrida	BB	15	10	15
Betacab	BB	10	5	12
Betacab	BB	12	7	12
Betacab	CC	20	4	22
Betacab	CC	22	2	22
Betacab	DD	25	25	25
Canutility	AA	11	5	15
Canutility	AA	15	8	15
Canutility	CC	19	0	19

Förklaring

En uppsättning med värden: 16, 16, 15, 15, 12, 12, 22, 22, 25, 15, 15 och 19. Kvalificeraren **nodistinct** innebär att uppsättningen innehåller ett element för varje rad i källdata: varje är maximalt **UnitPrice** för varje **Customer** och **Product**.

Diagramuttryck exempel 3

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplet nedan.

```
Set vNumberOfOrders = 1000;  OrderLines: Load   RowNo() as OrderLineID,         OrderID,        OrderDate,
Round((Year(OrderDate)-2005)*1000*Rand()*Rand()*Rand1) as Sales         While Rand()<=0.5 or IterNo
()=1; Load * Where OrderDate<=Today(); Load     Rand() as Rand1,        Date(MakeDate(2013)+Floor
((365*4+1)*Rand())) as OrderDate,       RecNo() as OrderID      Autogenerate vNumberOfOrders;
Calendar: Load distinct         Year(OrderDate) as Year,        Month(OrderDate) as Month,      OrderDate
Resident OrderLines;
```

Diagramuttryck

Skapa en tabellvisualisering i ett Qlik Sense-ark med **Year** och **Month** som dimensioner. Lägg till följande uttryck i tabellen som mått:

- `Sum(Sales)`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))` har etiketten Structured Aggr() i tabellen.

Resultat

Year	Month	Sum(Sales)	Structured Aggr()
2013	Jan	53495	53495
2013	Feb	48580	102075
2013	Mar	25651	127726
2013	Apr	36585	164311
2013	May	61211	225522
2013	Jun	23689	249211
2013	Jul	42311	291522
2013	Aug	41913	333435
2013	Sep	28886	362361
2013	Oct	25977	388298
2013	Nov	44455	432753
2013	Dec	64144	496897
2014	Jan	67775	67775

Förklaring

Det här exemplet visar aggregerade värden under en tolvmånadersperiod för varje år i kronologiskt stigande ordning, vilket förklarar de strukturerade parametrarna (Numeric, Ascending) i uttrycket **Aggr()**. Två specifika dimensioner krävs som strukturerade parametrar: **Year** och **Month**, sorterade (1) **Year** (numeric) och (2) **Month** (numeric). Dessa två dimensioner måste användas i tabellen eller diagramvisualiseringen. Det krävs för att dimensionslistan i funktionen **Aggr()** ska motsvara dimensionerna i objektet som används i visualiseringen.

Du kan jämföra skillnaden mellan dessa mått i en tabell eller i separata linjediagram:

- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year), (Month) ))`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))`

Det bör vara tydligt att endast det senare uttrycket utför den önskade ackumuleringen av aggregerade värden.

Se även:

Grundläggande aggregeringsfunktioner (page 208)

5.4 Färgfunktioner

Dessa funktioner kan användas i uttryck som används för att ange och utvärdera färgegenskaper för diagramobjekt, samt i dataladdningsskript.

*Qlik Sense har stöd för färgfunktionerna **Color()**, **qliktechblue** och **qliktechgray** för att säkerställa bakåtkompatibilitet, men dessa funktioner bör helst inte användas.*

ARGB

ARGB() används i uttryck för att ange eller utvärdera färgegenskaperna hos ett diagramobjekt där färgen definieras med en röd komponent **r**, en grön komponent **g** och en blå komponent **b**, med en alfafaktor (opacitet) på **alpha**.

```
ARGB (alpha, r, g, b)
```

HSL

HSL() används i uttryck för att ange eller utvärdera färgegenskaperna för ett diagramobjekt där färgen definieras av värden för **hue**, **saturation** och **luminosity**, mellan 0 och 1.

```
HSL  (hue, saturation, luminosity)
```

RGB

RGB() returnerar ett heltal som motsvarar färgkoden för den färg som definieras av tre parametrar: den röda komponenten r, den gröna komponenten g och den blå komponenten b. Komponenterna måste ha heltalsvärden mellan 0 och 255. Funktionen kan användas i uttryck för att ställa in eller utvärdera färgegenskaper för ett diagramobjekt.

```
RGB  (r, g, b)
```

Colormix1

Colormix1() används i uttryck för att returnera en ARGB-färgrepresentation från en tvåfärgad toning baserad på ett värde mellan 0 och 1.

```
Colormix1 (Value , ColorZero , ColorOne)
```

Value är ett reellt tal mellan 0 och 1.

- Om Value = 0 returneras ColorZero .
- Om Value = 1 returneras ColorOne .
- Om 0 < Value < 1 returneras motsvarande färgblandning.

ColorZero är en giltig RGB-färgrepresentation för den färg som ska associeras med intervallets lägsta värde.

ColorOne är en giltig RGB-färgrepresentation för den färg som ska associeras med intervallets högsta värde.

Exempel:

```
Colormix1(0.5, red(), blue())
```
returnerar:

```
ARGB(255,64,0,64) (purple)
```

Colormix2

Colormix2() används i uttryck för att returnera en ARGB-färgrepresentation från en tvåfärgad toning baserat på ett värde mellan -1 och 1, med möjlighet att ange en mellanliggande färg för intervallets mitt (0)

```
Colormix2 (Value ,ColorMinusOne , ColorOne[ , ColorZero])
```
Value är ett reellt tal mellan -1 och 1.

- Om Value = -1 returneras den första färgen.
- Om Value = 1 returneras den andra färgen.
- Om -1 < Value< 1 returneras den lämpliga färgblandningen.

ColorMinusOne är en giltig RGB-färgrepresentation för den färg som ska associeras med intervallets lägsta värde.

ColorOne är en giltig RGB-färgrepresentation för den färg som ska associeras med intervallets högsta värde.

ColorZero är en alternativ giltig RGB-färgrepresentation för den färg som ska associeras med intervallets mittersta värde.

SysColor

SysColor() returnerar ARGB-färgåtergivningen för Windows systemfärg nr, där nr motsvarar parametern till Windows API-funktion **GetSysColor(nr)**.

```
SysColor (nr)
```

ColorMapHue

ColorMapHue() returnerar ett ARGB-värde av en färg från en färgkarta som varierar nyanskomponenten för HSV-färgmodellen. Färgkartan börjar med röd, övergår sedan till gul, grön, cyan, blå, magenta och återgår till röd. x måste vara ett värde mellan 0 och 1.

```
ColorMapHue (x)
```

ColorMapJet

ColorMapJet() returnerar ett ARGB-värde för en färg från en färgkarta som börjar med blå, övergår till cyan, gul och orange och återgår till röd. x måste vara ett värde mellan 0 och 1.

```
ColorMapJet (x)
```

Fördefinierade färgfunktioner

Följande funktioner kan användas i uttryck för fördefinierade färger. Varje funktion returnerar en RGB-färgrepresentation.

En parameter för alfafaktorn kan också ges, vilket leder till att en ARGB-färgrepresentation returneras. Alfafaktorn 0 motsvarar full genomskinlighet. Alfafaktorn 255 motsvarar full opacitet. Om inget värde anges för alfa antas det vara 255.

Fördefinierade färgfunktioner

Färgfunktion	RGB -värde
black ([alpha])	(0,0,0)
blue([alpha])	(0,0,128)
brown([alpha])	(128,128,0)
cyan([alpha])	(0,128,128)
darkgray([alpha])	(128,128,128)
green([alpha])	(0,128,0)
lightblue([alpha])	(0,0,255)
lightcyan([alpha])	(0,255,255)
lightgray([alpha])	(192,192,192)
lightgreen([alpha])	(0,255,0)
lightmagenta([alpha])	(255,0,255)
lightred([alpha])	(255,0,0)
magenta([alpha])	(128,0,128)
red([alpha])	(128,0,0)
white([alpha])	(255,255,255)
yellow([alpha])	(255,255,0)

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Blue()`	RGB(0,0,128)
`Blue(128)`	ARGB(128,0,0,128)

ARGB

ARGB() används i uttryck för att ange eller utvärdera färgegenskaperna hos ett diagramobjekt där färgen definieras med en röd komponent **r**, en grön komponent **g** och en blå komponent **b**, med en alfafaktor (opacitet) på **alpha**.

Syntax:

```
ARGB(alpha, r, g, b)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
alpha	Genomskinlighetsvärde i intervallet 0–255. 0 är helt genomskinligt. 255 är helt ogenomskinligt.
r, g, b	Röda, gröna och blå komponentvärden. En färgkomponent med värdet 0 motsvarar ingen påverkan och värdet 255 motsvarar full påverkan.

Alla argument måste vara uttryck som resulterar i heltal i intervallet 0 till 255.

Om den numeriska komponenten tolkas och formateras i hexadecimal notation blir det lättare att se värdena på färgkomponenterna. Exempelvis har ljusgrön numret 4 278 255 360, som i hexadecimal notation är FF00FF00. De första två positionerna 'FF' (255) anger **alpha**-kanalen. De följande två positionerna "00" visar mängden för **röd**, de därpå följande två positionerna "FF" visar mängden för **grön** och de två sista positionerna "00" visar mängden för **blå**.

RGB

RGB() returnerar ett heltal som motsvarar färgkoden för den färg som definieras av tre parametrar: den röda komponenten r, den gröna komponenten g och den blå komponenten b. Komponenterna måste ha heltalsvärden mellan 0 och 255. Funktionen kan användas i uttryck för att ställa in eller utvärdera färgegenskaper för ett diagramobjekt.

Syntax:

```
RGB (r, g, b)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
r, g, b	Röda, gröna och blå komponentvärden. En färgkomponent med värdet 0 motsvarar ingen påverkan och värdet 255 motsvarar full påverkan.

Alla argument måste vara uttryck som resulterar i heltal i intervallet 0 till 255.

Om den numeriska komponenten tolkas och formateras i hexadecimal notation blir det lättare att se värdena på färgkomponenterna. Exempelvis har ljusgrön numret 4 278 255 360, som i hexadecimal notation är FF00FF00. De första två positionerna 'FF' (255) anger **alpha**-kanalen. I funktionerna **RGB** och **HSL** är denna alltid "FF" (ogenomskinlig). De följande två positionerna "00" visar mängden för **röd**, de därpå följande två positionerna "FF" visar mängden för **grön** och de två sista positionerna "00" visar mängden för **blå**.

Exempel: Diagramuttryck
I det här exemplet tilldelas ett diagram en anpassad färg:

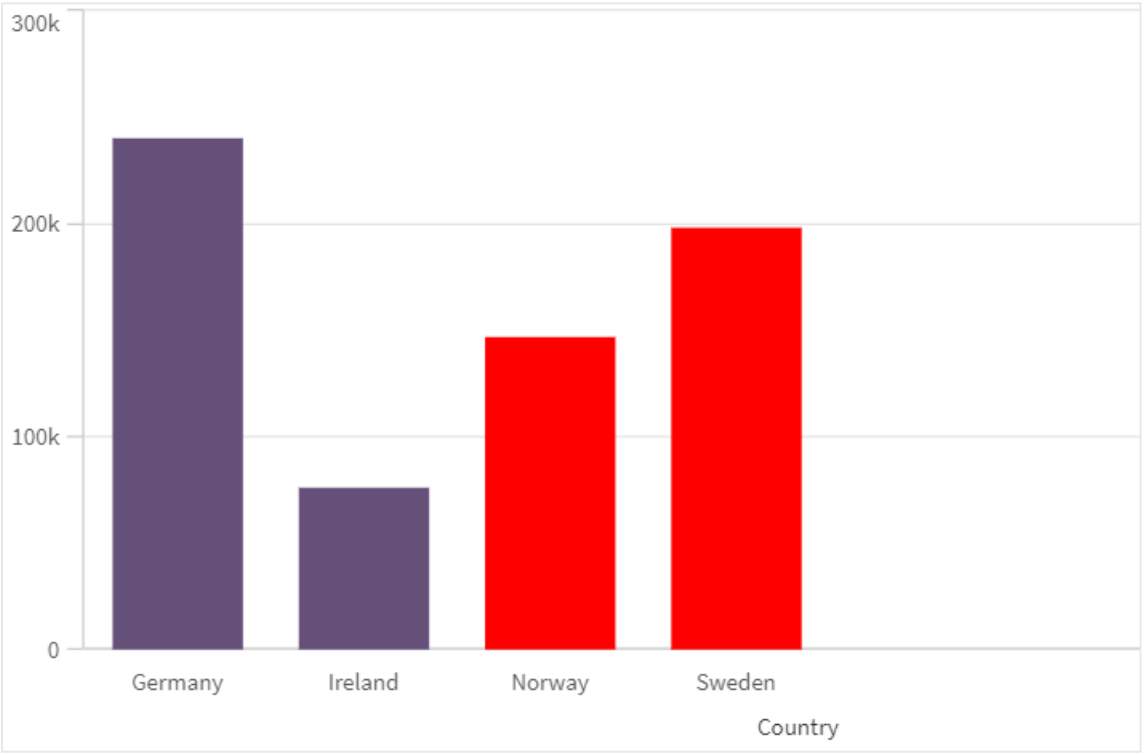
Data som används i exemplet:

```
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 ',');
```

Ange följande uttryck i egenskapspanelen **Färger och teckenförklaring**:

```
If (Sum(Sales)>Sum(Budget),RGB(255,0,0),RGB(100,80,120))
```

Resultat:



Exempel: Laddningsskript
Följande exempel visar motsvarande RGB-värden för värden i hex-format:

```
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];
```

Resultat:

Text	Färg
010203	RGB(1,2,3)
AABBCC	RGB(170,187,204)

HSL

HSL() används i uttryck för att ange eller utvärdera färgegenskaperna för ett diagramobjekt där färgen definieras av värden för **hue**, **saturation** och **luminosity**, mellan 0 och 1.

Syntax:

```
HSL (hue, saturation, luminosity)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
hue, saturation, luminosity	Komponentvärden för hue, saturation och luminosity mellan 0 och 1.

Alla argument måste vara uttryck som resulterar i heltal i intervallet 0 till 1.

Om den numeriska komponenten tolkas och formateras i hexadecimal notation blir det lättare att se RGB - värdena på färgkomponenterna. Exempelvis har ljusgrön numret 4 278 255 360, som i hexadecimal notation är FF00FF00 och RGB (0,255,0). Detta motsvarar HSL (80/240, 240/240, 120/240) , ett HSL - värde på (0.33, 1, 0.5).

5.5 Villkorsfunktioner

Alla villkorsfunktioner utvärderar ett villkor. Sedan returnerar de olika svar beroende på villkorets värde. Funktionerna kan användas i dataladdningsskriptet och diagramuttryck.

Villkorsfunktioner – en översikt

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

alt

alt-funktionen returnerar den första av parametrarna som har ett giltigt numeriskt värde. Om ingen av parametrarna har ett giltigt numeriskt värde, returneras den sista parametern. Det finns inga begränsningar vad gäller antalet parametrar.

```
alt (expr1[ , expr2 , expr3 , ...] , else)
```

class

class-funktionen kopplar den första parametern till ett klassintervall. Resultatet är ett dualt värde med a<=x<b som textvärde, där a och b är de övre och nedre gränserna av bin, och det nedre gränsvärdet ett numeriskt värde.

```
class (expression, interval [ , label [ , offset ]])
```

coalesce

Coalesce-funktionen returnerar den första av parametrarna som har en giltig non-NULL-representation. Det finns inga begränsningar vad gäller antalet parametrar.

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

if

if-funktionen returnerar ett värde beroende på om villkoret som hör samman med funktionen utvärderas som True eller False.

```
if (condition , then , else)
```

match

Funktionen **match** jämför den första parametern med alla de följande och returnerar den numeriska platsen för uttryck som matchar. Jämförelsen är skiftlägeskänslig.

```
match ( str, expr1 [ , expr2,...exprN ])
```

mixmatch

Funktionen **mixmatch** jämför den första parametern med alla de följande och returnerar den numeriska platsen för uttryck som matchar. Jämförelsen är inte skiftlägeskänslig.

```
mixmatch ( str, expr1 [ , expr2,...exprN ])
```

pick

Funktionen pick returnerar det *n*:te uttrycket i listan.

```
pick (n, expr1[ , expr2,...exprN])
```

wildmatch

Funktionen **wildmatch** jämför den första parametern med alla de följande och returnerar talet för det uttryck som matchar. Jokertecken (***** och **?**) kan användas i jämförelsesträngarna. ***** motsvarar vilken teckensekvens som helst. **?** motsvarar vilket enstaka tecken som helst. Jämförelsen är inte skiftlägeskänslig.

```
wildmatch ( str, expr1 [ , expr2,...exprN ])
```

alt

alt-funktionen returnerar den första av parametrarna som har ett giltigt numeriskt värde. Om ingen av parametrarna har ett giltigt numeriskt värde, returneras den sista parametern. Det finns inga begränsningar vad gäller antalet parametrar.

Syntax:

```
alt(expr1[ , expr2 , expr3 , ...] , else)
```

Argument:

Argument

Argument	Beskrivning
expr1	Det första uttrycket som ska kontrolleras om det har ett giltigt numeriskt värde.
expr2	Det andra uttrycket som ska kontrolleras om det har ett giltigt numeriskt värde.
expr3	Det tredje uttrycket som ska kontrolleras om det har ett giltigt numeriskt värde.
else	Värde som ska returneras om ingen av de föregående parametrarna har ett giltigt numeriskt värde.

Funktionen alt används ofta tillsammans med funktioner för tal- eller datumtolkning. På så vis kan Qlik Sense testa olika datumformat i en prioriterad ordningsföljd. Den kan även användas för att hantera NULL-värden i numeriska uttryck.

Exempel:

Exempel

Exempel	Resultat

`alt( date#( dat , 'YYYY/MM/DD' ),`
 	
`date#( dat , 'MM/DD/YYYY' ),`
 	
`date#( dat , 'MM/DD/YY' ),`
 	
`'No valid date' )`
	Uttrycket testar om fältet date innehåller ett datum enligt något av de tre angivna datumformaten. Om så är fallet, returneras ett dualt värde som innehåller den ursprungliga strängen och ett giltigt numeriskt värde som motsvarar ett datum. Om ingen matchning hittas returneras texten 'No valid date' (utan giltig numerisk representation).

`alt(Sales,0) + alt(Margin,0)`	Det här uttrycket lägger till fälten Sales och Margin, och ersätter eventuella saknade värden (NULL) med en nolla.

class

class-funktionen kopplar den första parametern till ett klassintervall. Resultatet är ett dualt värde med a<=x<b som textvärde, där a och b är de övre och nedre gränserna av bin, och det nedre gränsvärdet ett numeriskt värde.

Syntax:

```
class(expression, interval [ , label [ , offset ]])
```

Argument:

Argument

Argument	Beskrivning
interval	Ett tal som anger bin-bredden.
label	Valfri sträng som kan ersätta 'x' i den resulterande texten.
offset	Ett tal som kan användas som förskjutning från standardstartpunkten i klassificeringen. Startpunkten är normalt 0.

Exempel:

Exempel

Exempel	Resultat
`class( var,10 )` med `var = 23`	returnerar '20<=x<30'
`class( var,5,'value' )` med `var = 23`	returnerar '20<= value <25'
`class( var,10,'x',5 )` med `var = 23`	returnerar '15<=x<25'

Exempel – Laddningsskript som använder class

Exempel: laddningsskript

Laddningsskript

I det här exemplet laddar vi en tabell som innehåller namn och ålder på olika personer. Vi vill lägga till ett fält som klassificerar varje person utifrån en åldersgrupp med ett intervall på tio år. Den ursprungliga källtabellen ser ut enligt följande.

Resultat

Name	Age
John	25
Karen	42
Yoshi	53

Om du vill lägga till klassificeringsfältet för åldersgrupp kan du lägga till en föregående load-sats med hjälp av **class**-funktionen.

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. Skapa tabellen nedan i Qlik Sense för att se resultatet.

```
 LOAD *, class(Age, 10, 'age') As Agegroup;  LOAD * INLINE [ Age, Name 25, John 42, Karen 53, Yoshi];
```

Resultat

Resultat

Name	Age	Agegroup
John	25	20 <= age < 30
Karen	42	40 <= age < 50
Yoshi	53	50 <= age < 60

coalesce

Coalesce-funktionen returnerar den första av parametrarna som har en giltig non-NULL-representation. Det finns inga begränsningar vad gäller antalet parametrar.

Syntax:

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

Argument:

Argument

Argument	Beskrivning
expr1	Det första uttrycket som ska kontrolleras om det har något giltigt icke-NULL-värde.
expr2	Det andra uttrycket som ska kontrolleras om det har något giltigt icke-NULL-värde.
expr3	Det tredje uttrycket som ska kontrolleras om det har något giltigt icke-NULL-värde.

Exempel:

Exempel

Exempel	Resultat
	Det här uttrycker ändrar alla NULL-värden för ett fält till ”N/A”.
`Coalesce(ProductDescription, ProductName, ProductCode, 'no description available')`	Det här uttrycket väljer mellan tre olika produktbeskrivningsfält och används om vissa fält kanske inte har värden för produkten. Det första av fälten, i angiven ordning, som har ett icke-NULL-värde kommer att returneras. Om inte något av fälten innehåller ett värde blir resultatet ”no description available”.
`Coalesce(TextBetween(FileName, '"', '"'), FileName)`	Det här uttrycket tar bort eventuella citattecken runt fältet *FileName*. Om det *FileName* som används omges av citattecken tas de bort, och *FileName* returneras utan citattecknen. Om funktionen *TextBetween* inte hittar avgränsarna returnerar den NULL, vilket **Coalesce** avvisar, och returnerar *FileName* i råformat.

if

if-funktionen returnerar ett värde beroende på om villkoret som hör samman med funktionen utvärderas som True eller False.

Syntax:

```
if(condition , then [, else])
```

if-funktionen har tre parametrar: *condition*,*then* och *else*. Alla dessa är uttryck. De två andra, *then* och *else*, kan vara av vilken typ som helst.

Argument

Argument	Beskrivning
condition	Uttryck som tolkas logiskt.
then	Uttrycket kan vara av vilken typ som helst. Om *condition* är True, returnerar funktionen if värdet för uttrycket *then*.
else	Uttrycket kan vara av vilken typ som helst. Om *condition* är False, returnerar funktionen if värdet för uttrycket *else*. Den här parametern är valfri. Om *condition* är False returneras NULL om du inte har specificerat else.

Exempel

Exempel	Resultat
`if( Amount>= 0, 'OK', 'Alarm' )`	Uttrycket testar om beloppet är ett positivt tal (0 eller större) och returnerar 'OK' om det är det. Om beloppet är mindre än 0 returneras 'Alarm'.

Exempel – Laddningsskript som använder if

Exempel: Laddningsskript

Laddningsskript

If kan användas i laddningsskript med andra metoder och objekt, inklusive variabler. Om du till exempel ställer in variabeln *threshold* och vill inkludera ett fält i datamodellen baserat på den tröskeln, ska du göra följande.

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. Skapa tabellen nedan i Qlik Sense för att se resultatet.

```
Transactions: Load * 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  Comp
```

```
amount field from Transaction table to threshold of 100. Output results into a new field
called Compared to Threshold                                                    */
If(transaction_amount > $(threshold),'Greater than $(threshold)','Less than $(threshold)') as
[Compared to Threshold] Resident Transactions;
```

Resultat

Qlik Sense-tabellen visar utdata vid användning av *if*-funktionen i laddningsskriptet.

transaction_id	Jämfört med tröskel
3750	Mindre än 100
3751	Större än 100
3752	Mindre än 100
3753	Större än 100
3754	Större än 100
3756	Mindre än 100
3757	Större än 100

Exempel – Diagramuttryck som använder if

Exempel: Diagramuttryck

Diagramuttryck 1

Laddningsskript

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. När data har laddats skapar du diagramuttrycksexemplen nedan i en Qlik Sense-tabell.

```
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];
```

Qlik Sense-tabell som visar exempel på *if*-funktionen i ett diagramuttryck.

Datum	Plats	Incidenter	if(Incidents>=10, 'Critical', 'Ok')	if(Incidents>=10, 'Critical', If(Incidents>=1 and Incidents<10, 'Warning', 'Ok'))
1/3/2016	Peking	0	OK	OK
1/3/2016	Boston	12	Kritiskt	Kritiskt
1/3/2016	Stockholm	3	OK	Varning
1/3/2016	Toronto	0	OK	OK

Datum	Plats	Incidenter	if(Incidents>=10, 'Critical', 'Ok')	if(Incidents>=10, 'Critical', If(Incidents>=1 and Incidents<10, 'Warning', 'Ok'))
1/4/2016	Peking	0	OK	OK
1/4/2016	Boston	8	OK	Varning

Diagramuttryck 2

I en ny app lägger du till följande skript i en ny flik i Skriptredigeraren och laddar sedan data. Sedan kan du skapa tabellen med diagramuttrycken nedan.

```
SET FirstWeekDay=0; Load  Date(MakeDate(2022)+RecNo()-1) as Date Autogenerate 14;
```

Qlik Sense-tabell som visar ett exempel på *if*-funktionen i ett diagramuttryck.

Datum	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/1/2022	lör	Helg
1/2/2022	sön	Helg
1/3/2022	mån	Vardag
1/4/2022	tis	Vardag
1/5/2022	ons	Vardag
1/6/2022	tors	Vardag
1/7/2022	fre	Vardag
1/8/2022	lör	Helg
1/9/2022	sön	Helg
1/10/2022	mån	Vardag
1/11/2022	tis	Vardag
1/12/2022	ons	Vardag
1/13/2022	tors	Vardag
1/14/2022	fre	Vardag

match

Funktionen **match** jämför den första parametern med alla de följande och returnerar den numeriska platsen för uttryck som matchar. Jämförelsen är skiftlägeskänslig.

Syntax:

```
match( str, expr1 [ , expr2,...exprN ])
```

*Om du vill använda ej skiftlägeskänslig jämförelse använder du funktionen **mixmatch**. Om du vill använda ej skiftlägeskänslig jämförelse och jokertecken använder du funktionen **wildmatch**.*

Exempel: Laddningsskript som använder match

Exempel: Laddningsskript

Laddningsskript

Du kan använda match för att ladda en delmängd av dina data. Du kan till exempel returnera ett numeriskt värde för ett uttryck i funktionen. Sedan kan du begränsa laddade data utifrån det numeriska värdet. Match returnerar 0 om det inte finns någon matchning. Alla uttryck som inte har någon matchning i det här exemplet returnerar därför 0 och WHERE-satsen utesluter dem från dataladdningen.

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. Skapa tabellen nedan i Qlik Sense för att se resultatet.

```
Transactions: Load * 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;
```

Resultat

Qlik Sense-tabell som visar utdata vid användning av match-funktionen i laddningsskriptet

Color Code Blue and Black	Customer
Black	203521
Black	3036491
Blå	2038593

Exempel – Diagramuttryck som använder match

Exempel: Diagramuttryck

Diagramuttryck 1

Laddningsskript

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. När data har laddats skapar du diagramuttrycksexemplen nedan i en Qlik Sense-tabell.

```
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];
```

Det första uttrycket i tabellen nedan returnerar 0 för Stockholm, eftersom 'Stockholm' inte finns med i listan med uttryck i **match**-funktionen. Det returnerar även 0 för 'Zurich', eftersom **match**-jämförelsen är skiftlägeskänslig.

Qlik Sense-tabell som visar exempel på *match*-funktionen i ett diagramuttryck

Cities	match(Cities,'Toronto','Boston','Beijing','Zurich')	match(Cities,'Toronto','Boston','Beijing','Stockholm','zurich')
Beijing	3	3
Boston	2	2
Stockholm	0	4
Toronto	1	1
zurich	0	5

Diagramuttryck 2

Du kan använda match för att utföra en anpassad sortering för ett uttryck.

Som standard sorteras kolumner numeriskt eller alfabetiskt, beroende på datatyp.

Qlik Sense-tabell som visar ett exempel på standardsorteringsordningen

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Gör så här om du vill ändra ordningen:

1. Öppna delavsnittet **Sortering** för ditt diagram i **egenskapspanelen**.
2. Inaktivera automatisk sortering för den kolumn du vill använda anpassad sortering för.
3. Avmarkera **Sortera numeriskt** och **Sortera alfabetiskt**.
4. Välj **Sortera efter uttryck** och ange sedan ett uttryck som liknar följande:
   `=match( Cities, 'Toronto','Boston','Beijing','Stockholm','zurich')`
   Sorteringsordningen för kolumnen Cities ändras.

Qlik Sense-tabell som visar ett exempel på hur sorteringsordningen ändras med *match*-funktionen

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Du kan även visa det numeriska värde som returneras.

Qlik Sense-tabell som visar ett exempel på de numeriska värden som returneras av *match*-funktionen

Cities	Cities & ' - ' & match (Cities, 'Toronto','Boston', 'Beijing','Stockholm','zurich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

mixmatch

Funktionen **mixmatch** jämför den första parametern med alla de följande och returnerar den numeriska platsen för uttryck som matchar. Jämförelsen är inte skiftlägeskänslig.

Syntax:

```
mixmatch( str, expr1 [ , expr2,...exprN ])
```

Om du istället vill använda skiftlägeskänslig jämförelse använder du funktionen **match**. Om du vill använda ej skiftlägeskänslig jämförelse och jokertecken använder du funktionen **wildmatch**.

Exempel – Laddningsskript som använder mixmatch

Exempel: Laddningsskript

Laddningsskript

Du kan använda mixmatch för att ladda en delmängd av dina data. Du kan till exempel returnera ett numeriskt värde för ett uttryck i funktionen. Sedan kan du begränsa laddade data utifrån det numeriska värdet. Mixmatch returnerar 0 om det inte finns någon matchning. Alla uttryck som inte har någon matchning i det här exemplet returnerar därför 0 och WHERE-satsen utesluter dem från dataladdningen.

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. Skapa tabellen nedan i Qlik Sense för att se resultatet.

```
Load * Inline [ transaction_id, transaction_date, transaction_amount, transaction_quantity,
customer_id, size, color_code 3750, 20180830, 23.56, 2, 2038593, L, Red 3751, 20180907,
556.31, 6, 203521, m, orange 3752, 20180916, 5.75, 1, 5646471, S, blue 3753, 20180922, 125.00,
7, 3036491, l, Black 3754, 20180922, 484.21, 13, 049681, xs, Red 3756, 20180922, 59.18, 2,
2038593, M, Blue 3757, 20180923, 177.42, 21, 203521, XL, Black ];  /* Create new table called
Transaction_Buckets Create new fields called Customer, and Color code - Black, Blue, blue Load
Transactions table. Mixmatch returns 1 for 'Black', 2 for 'Blue'.  Also returns 3 for 'blue'
because mixmatch is not case sensitive. Only values that returned numeric value greater than 0
 are loaded by WHERE statement into Transactions_Buckets table. */  Transaction_Buckets: Load
  customer_id,     customer_id as [Customer],    color_code as [Color Code - Black, Blue,
blue] Resident Transactions Where mixmatch(color_code,'Black','Blue') > 0;
```

Resultat

Qlik Sense-tabellen visar utdata vid användning av mixmatch-funktionen i laddningsskriptet.

Color Code Black, Blue, blue	Customer
Black	203521
Black	3036491
Blå	2038593
blue	5646471

Exempel – Diagramuttryck som använder mixmatch

Exempel: Diagramuttryck

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. När data har laddats skapar du diagramuttrycksexemplen nedan i en Qlik Sense-tabell.

Diagramuttryck 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];
```

Det första uttrycket i tabellen nedan returnerar 0 för Stockholm, eftersom 'Stockholm' inte finns med i listan med uttryck i **mixmatch**-funktionen. Det returnerar 4 för 'Zurich', eftersom **mixmatch**-jämförelsen inte är skiftlägeskänslig.

Qlik Sense-tabell som visar exempel på *mixmatch*-funktionen i ett diagramuttryck

Cities	mixmatch(Cities,'Toronto','Boston','Beijing','Zurich')	mixmatch(Cities,'Toronto','Boston','Beijing','Stockholm','Zurich')
Beijing	3	3
Boston	2	2
Stockholm	0	4
Toronto	1	1
zurich	4	5

Diagramuttryck 2

Du kan använda mixmatch för att utföra en anpassad sortering för ett uttryck.

Som standard sorteras kolumner alfabetiskt eller numeriskt, beroende på datatyp.

Qlik Sense-tabell som visar ett exempel på standardsorteringsordningen

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Gör så här om du vill ändra ordningen:

1. Öppna delavsnittet **Sortering** för ditt diagram i **egenskapspanelen**.
2. Inaktivera automatisk sortering för den kolumn du vill använda anpassad sortering för.
3. Avmarkera **Sortera numeriskt** och **Sortera alfabetiskt**.
4. Välj **Sortera efter uttryck** och ange sedan följande uttryck:
   `=mixmatch( Cities, 'Toronto','Boston','Beijing','Stockholm','Zurich')`
   Sorteringsordningen för kolumnen Cities ändras.

Qlik Sense-tabell som visar ett exempel på hur sorteringsordningen har ändrats med *mixmatch*-funktionen.

Cities
Toronto

Cities
Boston
Beijing
Stockholm
zurich

Du kan även visa det numeriska värde som returneras.

Qlik Sense-tabell som visar ett exempel på de numeriska värden som returneras av *mixmatch*-funktionen.

Cities	Cities & ' - ' & mixmatch (Cities, 'Toronto','Boston', 'Beijing','Stockholm','Zurich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

pick

Funktionen pick returnerar det *n*:te uttrycket i listan.

Syntax:

```
pick(n, expr1[ , expr2,...exprN])
```

Argument:

Argument

Argument	Beskrivning
n	*n* är ett heltal mellan 1 och N.

Exempel:

Exempel

Exempel	Resultat
`pick( N, 'A','B',4, 6 )`	returnerar 'B' om N = 2 returnerar 4 om N = 3

wildmatch

Funktionen **wildmatch** jämför den första parametern med alla de följande och returnerar talet för det uttryck som matchar. Jokertecken (***** och **?**) kan användas i jämförelsesträngarna. ***** motsvarar vilken teckensekvens som helst. **?** motsvarar vilket enstaka tecken som helst. Jämförelsen är inte skiftlägeskänslig.

Syntax:

```
wildmatch( str, expr1 [ , expr2,...exprN ])
```

Om du vill använda en jämförelse utan jokertecken, ska du använda **match** eller **mixmatch**-funktionerna.

Exempel: Laddningsskript som använder wildmatch

Exempel: Laddningsskript

Laddningsskript

Du kan använda wildmatch för att ladda en delmängd av dina data. Du kan till exempel returnera ett numeriskt värde för ett uttryck i funktionen. Sedan kan du begränsa laddade data utifrån det numeriska värdet. Wildmatch returnerar 0 om det inte finns någon matchning. Alla uttryck som inte har någon matchning i det här exemplet returnerar därför 0 och WHERE-satsen utesluter dem från dataladdningen.

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. Skapa tabellen nedan i Qlik Sense för att se resultatet.

```
Transactions: Load * Inline [ transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size, color_code 3750, 20180830, 23.56, 2, 2038593, L, Red 3751, 20180907, 556.31, 6, 203521, m, orange 3752, 20180916, 5.75, 1, 5646471, S, blue 3753, 20180922, 125.00, 7, 3036491, l, Black 3754, 20180922, 484.21, 13, 049681, xs, Red 3756, 20180922, 59.18, 2, 2038593, M, Blue 3757, 20180923, 177.42, 21, 203521, XL, Black ];   /* Create new table called Transaction_Buckets Create new fields called Customer, and Color code - Black, Blue, blue, red Load Transactions table. Wildmatch returns 1 for 'Black', 'Blue', and 'blue', and 2 for 'Red'.  Only values that returned numeric value greater than 0  are loaded by WHERE statement into Transactions_Buckets table. */   Transaction_Buckets: Load customer_id,    customer_id as [Customer],    color_code as [Color Code Black, Blue, blue, Red] Resident Transactions Where wildmatch(color_code,'Bl*','R??') > 0;
```

Resultat

Qlik Sense-tabell som visar utdata vid användning av *wildmatch*-funktionen i laddningsskriptet

Color Code Black, Blue, blue, Red	Customer
Black	203521
Black	3036491
Blå	2038593

Color Code Black, Blue, blue, Red	Customer
blue	5646471
Red	049681
Red	2038593

Exempel: Diagramuttryck som använder wildmatch

Exempel: Diagramuttryck

Diagramuttryck 1

Skapa en ny flik i Skriptredigeraren och ladda sedan följande data som en inline load. När data har laddats skapar du diagramuttrycksexemplen nedan i en Qlik Sense-tabell.

```
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];
```

Det första uttrycket i tabellen nedan returnerar 0 för Stockholm, eftersom 'Stockholm' inte finns med i listan med uttryck i **wildmatch**-funktionen. Det returnerar även 0 för 'Boston', eftersom ? bara utför matchningar på enstaka tecken.

Qlik Sense-tabell som visar exempel på *wildmatch*-funktionen i ett diagramuttryck

Cities	wildmatch(Cities,'Tor*','?ton','Beijing','*urich')	wildmatch(Cities,'Tor*','???ton','Beijing','Stockholm','*urich')
Beijing	3	3
Boston	0	2
Stockholm	0	4
Toronto	1	1
zurich	4	5

Diagramuttryck 2

Du kan använda wildmatch för att utföra en anpassad sortering för ett uttryck.

Som standard sorteras kolumner numeriskt eller alfabetiskt, beroende på datatyp.

Qlik Sense-tabell som visar ett exempel på standardsorteringsordningen

Cities
Beijing
Boston

Cities
Stockholm
Toronto
zurich

Gör så här om du vill ändra ordningen:

1. Öppna delavsnittet **Sortering** för ditt diagram i **egenskapspanelen**.
2. Inaktivera automatisk sortering för den kolumn du vill använda anpassad sortering för.
3. Avmarkera **Sortera numeriskt** och **Sortera alfabetiskt**.
4. Välj **Sortera efter uttryck** och ange sedan ett uttryck som liknar följande:
   `=wildmatch( Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')`
   Sorteringsordningen för kolumnen Cities ändras.

Qlik Sense-tabell som visar ett exempel på hur sorteringsordningen har ändrats med *wildmatch*-funktionen.

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Du kan även visa det numeriska värde som returneras.

Qlik Sense-tabell som visar ett exempel på de numeriska värden som returneras av *wildmatch*-funktionen

Cities	Cities & ' - ' & wildmatch (Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

5.6 Räknefunktioner

I det här avsnittet beskrivs funktioner relaterade till posträknare under utvärdering av **LOAD**-satser i dataladdningsskriptet. Den enda funktion som kan användas i diagramuttryck är **RowNo()**.

En del räknefunktioner har inga parametrar, men de avslutande parenteserna är ändå obligatoriska.

Räknefunktioner - en översikt

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

autonumber

Denna skriptfunktion returnerar ett unikt heltal för varje distinkt utvärderat värde av *expression* som påträffas under skriptexekveringen. Funktionen kan exempelvis användas för att skapa en kompakt minnesrepresentation av en komplex nyckel.

```
autonumber (expression[ , AutoID])
```

autonumberhash128

Denna skriptfunktion beräknar en 128-bitars hashning av de kombinerade indatauttrycksvärdena och returnerar ett unikt heltalsvärde för varje distinkt hash-värde som påträffas under skriptexekveringen. Funktionen kan exempelvis användas för att skapa en kompakt minnesrepresentation av en komplex nyckel.

```
autonumberhash128 (expression {, expression})
```

autonumberhash256

Denna skriptfunktion beräknar en 256-bitars hashning av de kombinerade indatauttrycksvärdena och returnerar ett unikt heltalsvärde för varje distinkt hash-värde som påträffas under skriptexekveringen. Funktionen kan exempelvis användas för att skapa en kompakt minnesrepresentation av en komplex nyckel.

```
autonumberhash256 (expression {, expression})
```

IterNo

Denna skriptfunktion returnerar ett heltal som anger för vilken gång i ordningen en post utvärderas i en **LOAD**-sats med ett **while**-tillägg. Den första upprepningen får nummer 1. Funktionen **IterNo** är endast meningsfull om den används tillsammans med en **while**-sats.

```
IterNo ( )
```

RecNo

Denna skriptfunktion returnerar ett heltal som motsvarar numret på den rad i den aktuella tabellen som läses vid det givna tillfället. Första posten får nummer 1.

```
RecNo ( )
```

RowNo - script function

Denna funktion returnerar ett heltal som anger den aktuella radens placering i den resulterande interna Qlik Sense-tabellen. Första raden får nummer 1.

```
RowNo ( )
```

RowNo - chart function

RowNo() returnerar numret på den aktuella raden i det aktuella kolumnsegmentet i en tabell. För bitmappsdiagram returnerar **RowNo()** numret på den aktuella raden i diagrammets raka tabellmotsvarighet.

```
RowNo - diagramfunktion([TOTAL])
```

autonumber

Denna skriptfunktion returnerar ett unikt heltal för varje distinkt utvärderat värde av *expression* som påträffas under skriptexekveringen. Funktionen kan exempelvis användas för att skapa en kompakt minnesrepresentation av en komplex nyckel.

> *Du kan enbart koppla **autonumber**-nycklar som har skapats i samma dataladdning, eftersom heltalet genereras i enlighet med den ordning som tabellen laddas i. Om du behöver använda nycklar som är varaktiga mellan dataladdningar, fristående från källdatasortering, bör du använda funktionerna **hash128**, **hash160** eller **hash256**.*

Syntax:

```
autonumber(expression[ , AutoID])
```

Argument:

Argument	Beskrivning
AutoID	För att skapa flera räkneinstanser i de fall där funktionen **autonumber** används för flera olika nycklar inom ett skript, kan man använda den frivilliga parametern *AutoID* för att namnge instanserna.

Exempel: Skapa en sammansatt nyckel

I det här exemplet skapar vi en sammansatt nyckel med hjälp av funktionen **autonumber** för att spara minne. Exemplet är kortfattat eftersom det är avsett som en illustration, men blir meningsfullt med en tabell som innehåller ett stort antal rader.

Exempeldata

Region	Year	Month	Sales
North	2014	May	245
North	2014	May	347
North	2014	June	127
South	2014	June	645
South	2013	May	367
South	2013	May	221

Källdata laddas med inline-data. Sedan lägger vi till en föregående load som skapar en sammansatt nyckel från fälten Region, Year och Month.

```
RegionSales:
LOAD *,
AutoNumber(Region&Year&Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May, 367
South, 2013,   May,    221
];
```

Den resulterande tabellen ser ut så här:

Resultattabell

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1
North	2014	June	127	2
South	2014	June	645	3
South	2013	May	367	4
South	2013	May	221	4

I det här exemplet kan du hänvisa till RYMkey, exempelvis 1, i stället för strängen "North2014May" om du behöver länka till en annan tabell.

Nu laddar vi en källtabell med kostnader på liknande sätt. Fälten Region, Year och Month undantas i föregående laddning för att undvika att skapa en syntetisk nyckel. Vi skapar redan en sammansatt nyckel med funktionen **autonumber** som länkar tabellerna.

```
RegionCosts:
LOAD Costs,
AutoNumber(Region&Year&Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Costs
South, 2013,   May,    167
North, 2014,   May,    56
North, 2014,   June,   199
South, 2014,   June,   64
South, 2013,   May, 172
South, 2013,   May,    126
];
```

Nu kan vi lägga till en tabellvisualisering på ett ark och lägga till fälten Region, Year och Month samt summamått för försäljning och kostnader. Tabellen ser då ut så här:

Resultattabell

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

autonumberhash128

Denna skriptfunktion beräknar en 128-bitars hashning av de kombinerade indatauttrycksvärdena och returnerar ett unikt heltalsvärde för varje distinkt hash-värde som påträffas under skriptexekveringen. Funktionen kan exempelvis användas för att skapa en kompakt minnesrepresentation av en komplex nyckel.

*Du kan enbart koppla **autonumberhash128**-nycklar som har skapats i samma dataladdning, eftersom heltalet genereras i enlighet med den ordning som tabellen laddas i. Om du behöver använda nycklar som är varaktiga mellan dataladdningar, fristående från källdatasortering, bör du använda funktionerna **hash128**, **hash160** eller **hash256**.*

Syntax:

```
autonumberhash128(expression {, expression})
```

Exempel: Skapa en sammansatt nyckel

I det här exemplet skapar vi en sammansatt nyckel med hjälp av funktionen **autonumberhash128** för att spara minne. Exemplet är kortfattat eftersom det är avsett som en illustration, men blir meningsfullt med en tabell som innehåller ett stort antal rader.

Exempeldata

Region	Year	Month	Sales
North	2014	May	245
North	2014	May	347
North	2014	June	127
South	2014	June	645
South	2013	May	367
South	2013	May	221

Källdata laddas med inline-data. Sedan lägger vi till en föregående load som skapar en sammansatt nyckel från fälten Region, Year och Month.

```
RegionSales:
LOAD *,
AutoNumberHash128(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May, 367
South, 2013,   May,    221
];
```

Den resulterande tabellen ser ut så här:

Resultattabell

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1
North	2014	June	127	2
South	2014	June	645	3
South	2013	May	367	4
South	2013	May	221	4

I det här exemplet kan du hänvisa till RYMkey, exempelvis 1, i stället för strängen "North2014May" om du behöver länka till en annan tabell.

Nu laddar vi en källtabell med kostnader på liknande sätt. Fälten Region, Year och Month undantas i föregående laddning för att undvika att skapa en syntetisk nyckel. Vi skapar redan en sammansatt nyckel med funktionen **autonumberhash128** som länkar tabellerna.

```
RegionCosts:
LOAD Costs,
AutoNumberHash128(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Costs
South, 2013,   May,    167
North, 2014,   May,    56
North, 2014,   June,   199
South, 2014,   June,   64
South, 2013,   May, 172
South, 2013,   May,    126
];
```

Nu kan vi lägga till en tabellvisualisering på ett ark och lägga till fälten Region, Year och Month samt summamått för försäljning och kostnader. Tabellen ser då ut så här:

Resultattabell

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

autonumberhash256

Denna skriptfunktion beräknar en 256-bitars hashning av de kombinerade indatauttrycksvärdena och returnerar ett unikt heltalsvärde för varje distinkt hash-värde som påträffas under skriptexekveringen. Funktionen kan exempelvis användas för att skapa en kompakt minnesrepresentation av en komplex nyckel.

*Du kan enbart koppla **autonumberhash256**-nycklar som har skapats i samma dataladdning, eftersom heltalet genereras i enlighet med den ordning som tabellen laddas i. Om du behöver använda nycklar som är varaktiga mellan dataladdningar, fristående från källdatasortering, bör du använda funktionerna **hash128**, **hash160** eller **hash256**.*

Syntax:

```
autonumberhash256(expression {, expression})
```

Exempel: Skapa en sammansatt nyckel

I det här exemplet skapar vi en sammansatt nyckel med hjälp av funktionen **autonumberhash256** för att spara minne. Exemplet är kortfattat eftersom det är avsett som en illustration, men blir meningsfullt med en tabell som innehåller ett stort antal rader.

Exempeltabell

Region	Year	Month	Sales
North	2014	May	245
North	2014	May	347
North	2014	June	127
South	2014	June	645
South	2013	May	367
South	2013	May	221

Källdata laddas med inline-data. Sedan lägger vi till en föregående load som skapar en sammansatt nyckel från fälten Region, Year och Month.

```
RegionSales:
LOAD *,
AutoNumberHash256(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May, 367
South, 2013,   May,    221
];
```

Den resulterande tabellen ser ut så här:

Resultattabell

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1
North	2014	June	127	2
South	2014	June	645	3
South	2013	May	367	4
South	2013	May	221	4

I det här exemplet kan du hänvisa till RYMkey, exempelvis 1, i stället för strängen "North2014May" om du behöver länka till en annan tabell.

Nu laddar vi en källtabell med kostnader på liknande sätt. Fälten Region, Year och Month undantas i föregående laddning för att undvika att skapa en syntetisk nyckel. Vi skapar redan en sammansatt nyckel med funktionen **autonumberhash256** som länkar tabellerna.

```
RegionCosts:
LOAD Costs,
AutoNumberHash256(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Costs
South, 2013,   May,    167
North, 2014,   May,    56
North, 2014,   June,   199
South, 2014,   June,   64
South, 2013,   May, 172
South, 2013,   May,    126
];
```

Nu kan vi lägga till en tabellvisualisering på ett ark och lägga till fälten Region, Year och Month samt summamått för försäljning och kostnader. Tabellen ser då ut så här:

Resultattabell

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

IterNo

Denna skriptfunktion returnerar ett heltal som anger för vilken gång i ordningen en post utvärderas i en **LOAD**-sats med ett **while**-tillägg. Den första upprepningen får nummer 1. Funktionen **IterNo** är endast meningsfull om den används tillsammans med en **while**-sats.

Syntax:

```
IterNo( )
```

Exempel och resultat:

Exempel:

```
LOAD
   IterNo() as Day,
   Date( StartDate + IterNo() - 1 ) as Date
   While StartDate + IterNo() - 1 <= EndDate;

LOAD * INLINE
[StartDate, EndDate
2014-01-22, 2014-01-26
];
```

Denna **LOAD**-sats genererar en post per datum inom det intervall som är angivet med **StartDate** och **EndDate**.

I det här exemplet ser den resulterande tabellen ut så här:

Resultattabell

Day	Date
1	2014-01-22

Day	Date
2	2014-01-23
3	2014-01-24
4	2014-01-25
5	2014-01-26

RecNo

Denna skriptfunktion returnerar ett heltal som motsvarar numret på den rad i den aktuella tabellen som läses vid det givna tillfället. Första posten får nummer 1.

Syntax:

```
RecNo( )
```

I kontrast till **RowNo( )**, som räknar rader i den resulterande Qlik Sense tabellen, räknar **RecNo( )** posterna i rådatatabellen och återställer dem när en rådatatabell sammanfogas till en annan.

Exempel: Dataladdningsskript

Laddning av rådatatabell

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Laddar post och radnummer för markerade rader:

```
QTab:
LOAD *,
RecNo( ),
RowNo( )
resident Tab1 where A<>2;

LOAD
C as A,
D as B,
RecNo( ),
RowNo( )
resident Tab2 where A<>5;
```

```
//We don't need the source tables anymore, so we drop them
Drop tables Tab1, Tab2;
```

Resultatet blir en intern tabell i Qlik Sense:

Resultattabell

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo

Denna funktion returnerar ett heltal som anger den aktuella radens placering i den resulterande interna Qlik Sense-tabellen. Första raden får nummer 1.

Syntax:

```
RowNo( [TOTAL])
```

I motsats till **RecNo( )**, som räknar posterna i rådatatabellen, räknar **RowNo( )**-funktionen inte poster som uteslutits av **where**-tillägg och börjar inte om på 1 när rådatatabeller konkateneras med varandra.

> *Om du använder föregående load, det vill säga ett antal **LOAD** -flervärdessatser som läses från samma tabell kan du bara använda **RowNo( )** i den översta **LOAD** -satsen. Om du använder **RowNo( )** i de följande **LOAD** -satserna returneras 0.*

Exempel: Dataladdningsskript

Laddning av rådatatabell

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Laddar post och radnummer för markerade rader:

```
QTab:
```

```
LOAD *,
RecNo( ),
RowNo( )
resident Tab1 where A<>2;

LOAD
C as A,
D as B,
RecNo( ),
RowNo( )
resident Tab2 where A<>5;

//We don't need the source tables anymore, so we drop them
Drop tables Tab1, Tab2;
```

Resultatet blir en intern tabell i Qlik Sense:

Resultattabell

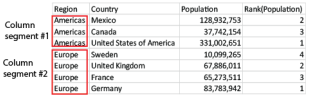
A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo - diagramfunktion

RowNo() returnerar numret på den aktuella raden i det aktuella kolumnsegmentet i en tabell. För bitmappsdiagram returnerar **RowNo()** numret på den aktuella raden i diagrammets raka tabellmotsvarighet.

Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner, utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

Kolumnsegment



*Sortering efter y-värde i diagram eller sortering efter uttryckskolumner i tabeller är inte tillåtet när **RowNo()** används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade.*

Syntax:

```
RowNo([TOTAL])
```

Returnerad datatyp: heltal

Argument:

Argument	Beskrivning
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

Exempel: Diagramuttryck med RowNo

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
Temp:
 LOAD * inline [ Customer|Product|OrderNumber|UnitSales|UnitPrice Astrida|AA|1|4|16
Astrida|AA|7|10|15 Astrida|BB|4|9|9 Betacab|CC|6|5|10 Betacab|AA|5|2|20 Betacab|BB|1|25| 25
Canutility|AA|3|8|15 Canutility|CC|5|4|19 Divadip|CC|2|4|16 Divadip|DD|3|1|25 ] (delimiter is
'|');
```

Diagramuttryck

Skapa en tabellvisualisering i ett Qlik Sense-ark med **Customer** och **UnitSales** som dimensioner. Lägg till `RowNo( )` och `RowNo(TOTAL)` som mått med etiketten **Row in Segment** respektive **Row Number**. Lägg till följande uttryck i tabellen som ett mått:

```
If( RowNo( )=1, 0, UnitSales / Above( UnitSales ))
```

Resultat

Customer	UnitSales	Row in Segment	Row Number	If(RowNo( )=1, 0, UnitSales / Above(UnitSales))
Astrida	4	1	1	0
Astrida	9	2	2	2.25
Astrida	10	3	3	1.1111111111111
Betacab	2	1	4	0
Betacab	5	2	5	2.5
Betacab	25	3	6	5

Customer	UnitSales	Row in Segment	Row Number	If(RowNo( )=1, 0, UnitSales / Above(UnitSales))
Canutility	4	1	7	0
Canutility	8	2	8	2
Divadip	1	1	9	0
Divadip	4	2	10	4

Förklaring

Kolumnen **Row in Segment** visar resultatet 1,2,3 för kolumnsegmentet som innehåller värdena i UnitSales för kunden Astrida. Radnumreringen börjar sedan på 1 igen för nästa kolumnsegment, som är Betacab.

Kolumnen **Row Number** bortser från dimensionerna på grund av argumentet `TOTAL` för `RowNo()` och räknar raderna i tabellen.

Det här uttrycket returnerar 0 för den första raden i varje kolumnsegment, så kolumnen visar:

0, 2,25, 1,1111111, 0, 2,5, 5, 0, 2, 0 och 4.

Se även:

Above - diagramfunktion (page 642)

5.7 Datum- och tidsfunktioner

Datum- och tidsfunktionerna i Qlik Sense används för att konvertera datum- och tidsvärden. Alla funktioner kan användas både i dataladdningsskriptet och diagramuttryck.

Funktionerna bygger på ett serienummer för datum och tid som motsvarar antalet dagar sedan den 30 december 1899. Heltalet motsvarar dagen och fraktionen motsvarar tiden på dagen.

Qlik Sense använder parameterns numeriska värde, så ett tal är giltigt som parameter även när det inte är formaterat som ett datum eller en tidspunkt. Om parametern inte motsvarar ett numeriskt värde, exempelvis eftersom den är en sträng, försöker Qlik Sense tolka strängen enligt datum- och tidsmiljövariablerna.

Om du använder ett tidformat i parametern som inte motsvarar operativsystemets inställningar, kan Qlik Sense inte göra en korrekt tolkning. För att lösa detta kan du antingen ändra inställningarna eller använda en tolkningsfunktion.

I exemplen för varje funktion antas standardformaten för tid och datum hh:mm:ss och YYYY-MM-DD (ISO 8601).

Qlik Sense ignorerar alla parametrar för sommartid när en tidsmarkör bearbetas med en datum- eller tidsfunktion, såvida inte datum- eller tidsfunktionen innehåller en geografisk position.

Exempelvis skulle `ConvertToLocalTime( filetime('Time.qvd'), 'Paris')` *använda parametrar för sommartid, medan* `ConvertToLocalTime(filetime('Time.qvd'), 'GMT-01:00')` *inte skulle använda parametrar för sommartid.*

Datum- och tidsfunktioner – en översikt

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

Tidsuttryck

second

Denna funktion returnerar ett heltal som motsvarar sekunden om decimaldelen av **expression** tolkas som en tidpunkt enligt standardtolkningen av tal.

```
second (expression)
```

minute

Denna funktion returnerar ett heltal som motsvarar minuten om decimaldelen av **expression** tolkas som tid enligt standardtolkningen av tal.

```
minute (expression)
```

hour

Denna funktion returnerar ett heltal som motsvarar timmen om decimaldelen av **expression** tolkas som en tidpunkt enligt standardtolkningen av tal.

```
hour (expression)
```

day

Denna funktion returnerar ett heltal som motsvarar dagen om decimaldelen av **expression** tolkas som datum enligt standardtolkningen av tal.

```
day (expression)
```

week

Denna funktion returnerar ett heltal som motsvarar veckonumret enligt ISO 8601. Veckonumret beräknas utifrån datumtolkningen av uttrycket, enligt standardtalformatet.

```
week (expression)
```

month

Denna funktion returnerar ett dualt värde: månadsnamnet som det är definierat i miljövariabeln **MonthNames** och ett heltal mellan 1 och 12. Månaden beräknas utifrån datumtolkningen av uttrycket, enligt standardtalformatet.

```
month (expression)
```

year

Denna funktion returnerar ett heltal som motsvarar året om **expression** tolkas som ett datum enligt standardtolkningen av tal.

```
year (expression)
```

weekyear

Denna funktion returnerar det år som veckonumret hör till enligt ISO 8601. Veckonummer går från 1 till cirka 52.

```
weekyear (expression)
```

weekday

Denna funktion returnerar ett dualt värde med: Ett namn på en dag som definierat i miljövariabeln **DayNames**. Ett heltal mellan 0 och 6 som motsvarar den nominella veckodagen (0-6).

```
weekday (date)
```

Tidsmarkörer

now

Denna funktion returnerar en tidsmarkör för aktuell tid från systemklockan. Standardvärdet är 1.

```
now ([ timer_mode])
```

today

Denna funktion returnerar aktuellt datum från systemklockan.

```
today ([timer_mode])
```

LocalTime

Denna funktion returnerar en tidsmarkör för aktuell tid från systemklockan för en angiven tidszon.

```
localtime ([timezone [, ignoreDST ]])
```

Make

makedate

Denna funktion returnerar ett datum beräknat utifrån året **YYYY**, månaden **MM** och dagen **DD**.

```
makedate (YYYY [ , MM [ , DD ] ])
```

makeweekdate

Denna funktion returnerar ett datum beräknat utifrån året **YYYY**, veckan **WW** och veckodagen **D**.

```
makeweekdate (YYYY [ , WW [ , D ] ])
```

maketime

Denna funktion returnerar en tid beräknat utifrån timmar **hh**, minuter **mm** och sekunder **ss**.

```
maketime (hh [ , mm [ , ss [ .fff ] ] ])
```

Övriga datumfunktioner

AddMonths

Denna funktion returnerar det datum som infaller **n** månader efter **startdate** eller, om **n** är negativt, det datum som infaller **n** månader före **startdate**.

```
addmonths (startdate, n , [ , mode])
```

AddYears

Denna funktion returnerar det datum som infaller **n** år efter **startdate** eller, om **n** är negativt, det datum som infaller**n** år före **startdate**.

```
addyears (startdate, n)
```

yeartodate

Denna funktion räknar ut om indatatidsmarkören hamnar inom året för datumet då skriptet senast laddades och returnerar True om så är fallet, False om så inte är fallet.

```
yeartodate (date [ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Tidszoner

timezone

Denna funktion returnerar namnet på den aktuella tidszonen så som den definierats i Windows.

```
timezone ( )
```

GMT

Denna funktion returnerar aktuell Greenwich Mean Time utifrån systemklockan och Windows-tidsinställningarna.

```
GMT ( )
```

UTC

Returnerar aktuell Coordinated Universal Time.

```
UTC ( )
```

daylightsaving

Returnerar den aktuella anpassningen till sommartid/vintertid enligt Windows definition.

```
daylightsaving ( )
```

converttolocaltime

Konverterar en UTC- eller GMT-tidsmarkör till lokal tid i form av ett dualt värde. Ett antal platser (städer, platser och tidszoner) i hela världen kan användas.

```
converttolocaltime (timestamp [, place [, ignore_dst=false]])
```

Ange tid

setdateyear

Den här funktionen tar som indata en **timestamp** och ett **year** och uppdaterar **timestamp** med det **year** som har angetts i indata.

```
setdateyear (timestamp, year)
```

setdateyearmonth

Den här funktionen tar som indata en **timestamp**, en **month** och ett **year** och uppdaterar **timestamp** med det **year** och den **month** som har angetts i indata.

```
setdateyearmonth (timestamp, year, month)
```

In...

inyear

Denna funktion returnerar True om **timestamp** ligger inom det år som innehåller **base_date**.

```
inyear (date, basedate , shift [, first_month_of_year = 1])
```

inyeartodate

Denna funktion returnerar True om **timestamp** ligger inom den del av året som innehåller **base_date** fram till och inklusive den sista millisekunden av **base_date**.

```
inyeartodate (date, basedate , shift [, first_month_of_year = 1])
```

inquarter

Denna funktion returnerar True om **timestamp** ligger inom det kvartal som innehåller **base_date**.

```
inquarter (date, basedate , shift [, first_month_of_year = 1])
```

inquartertodate

Denna funktion returnerar True om **timestamp** ligger inom den del av det kvartal som innehåller **base_date** fram till och inklusive den sista millisekunden av **base_date**.

```
inquartertodate (date, basedate , shift [, first_month_of_year = 1])
```

inmonth

Denna funktion returnerar True om **timestamp** ligger inom den månad som innehåller **base_date**.

```
inmonth (date, basedate , shift)
```

inmonthtodate

Returnerar True om **date** ligger inom den del av månaden som innehåller **basedate** fram till och inklusive den sista millisekunden av **basedate**.

```
inmonthtodate (date, basedate , shift)
```

inmonths

Med den här funktionen får vi reda på om en tidsmarkör finns inom samma månad, tvåmånadersperiod, kvartal, tredjedels år eller halvår som ett basdatum. Det går även att se om tidsmarkören finns inom en föregående eller senare tidsperiod.

```
inmonths (n, date, basedate , shift [, first_month_of_year = 1])
```

inmonthstodate

Med den här funktionen får vi reda på om en tidsmarkör finns inom delen av månaden, tvåmånadersperioden, kvartalet, tredjedelens år eller halvåret fram till och inklusive den sista millisekunden av **base_date**. Det går även att se om tidsmarkören finns inom en föregående eller senare tidsperiod.

```
inmonthstodate  (n, date, basedate , shift [, first_month_of_year = 1])
```

inweek

Denna funktion returnerar True om **timestamp** ligger inom den vecka som innehåller **base_date**.

```
inweek (date, basedate , shift [, weekstart])
```

inweektodate

Denna funktion returnerar True om **timestamp** ligger inom den del av veckan som innehåller **base_date** fram till och inklusive den sista millisekunden av **base_date**.

```
inweektodate  (date, basedate , shift [, weekstart])
```

inlunarweek

Med denna funktion får vi reda på om **timestamp** ligger inom den sjudagarsperiod som innehåller **base_date**. Sjudagarsperioder i Qlik Sense definieras genom att 1 januari räknas som den första dagen på veckan.

```
inlunarweek  (date, basedate , shift [, weekstart])
```

inlunarweektodate

Denna funktion tar reda på om **timestamp** ligger inom delen av sjudagarsperioden fram till och inklusive den sista millisekunden av **base_date**. Sjudagarsperioder i Qlik Sense definieras genom att 1 januari räknas som den första dagen på veckan.

```
inlunarweektodate  (date, basedate , shift [, weekstart])
```

inday

Denna funktion returnerar True om **timestamp** ligger inom den dag som innehåller **base_timestamp**.

```
inday (timestamp, basetimestamp , shift [, daystart])
```

indaytotime

Denna funktion returnerar True om **timestamp** ligger inom den del av dagen som innehåller **base_timestamp** fram till och inklusive den exakta millisekunden för **base_timestamp**.

```
indaytotime (timestamp, basetimestamp , shift [, daystart])
```

Start... end

yearstart

Denna funktion returnerar en tidsmarkör som motsvarar starten av den första dagen i det år som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

```
yearstart ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör med den sista millisekunden av den sista dagen i det år som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

```
yearend ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearname

Denna funktion returnerar ett fyrsiffrigt år som visningsvärde med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden av den första dagen på det år som innehåller **date**.

```
yearname (date [, shift = 0 [, first_month_of_year = 1]] )
```

quarterstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden i det kvartal som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

```
quarterstart (date [, shift = 0 [, first_month_of_year = 1]])
```

quarterend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden i det kvartal som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

```
quarterend (date [, shift = 0 [, first_month_of_year = 1]])
```

quartername

Denna funktion returnerar ett visningsvärde med kvartalets månader (formaterat enligt skriptvariabeln **MonthNames**) och år med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden av kvartalets första dag.

```
quartername (date [, shift = 0 [, first_month_of_year = 1]])
```

monthstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden av den första dagen i den månad som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

```
monthstart (date [, shift = 0])
```

monthend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör med den sista millisekunden av den sista dagen i den månad som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

```
monthend (date [, shift = 0])
```

monthname

Denna funktion returnerar ett visningsvärde med månaden (formaterat enligt skriptvariabeln **MonthNames**) och året med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden av månadens första dag.

```
monthname (date [, shift = 0])
```

monthsstart

Denna funktion returnerar ett värde som motsvarar tidsmarkören för den första millisekunden i månaden, tvåmånadersperioden, kvartalet, tredjedelens år eller halvåret som innehåller ett basdatum. Det går även att hitta tidsmarkören för en föregående eller senare tidsperiod.

```
monthsstart (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden i månaden, tvåmånadersperioden, kvartalet, tredjedelens år eller halvåret som innehåller ett basdatum. Det går även att hitta tidsmarkören för en föregående eller senare tidsperiod.

```
monthsend (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsname

Denna funktion returnerar ett visningsvärde som representerar månadsintervallet i perioden (formaterat enligt skriptvariabeln **MonthNames**) liksom året. Det underliggande numeriska värdet motsvarar en tidsmarkör för den första millisekunden i månaden, tvåmånadersperioden, kvartalet, tredjedelen av året eller halvåret som innehåller ett basdatum.

```
monthsname (n, date [, shift = 0 [, first_month_of_year = 1]])
```

weekstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden på den första dagen (måndag) i den kalendervecka som innehåller **date**. Det förvalda utdataformatet är det **DateFormat** som har definierats i skriptet.

```
weekstart (date [, shift = 0 [,weekoffset = 0]])
```

weekend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden på den sista dagen (söndag) i kalenderveckan som innehåller **date**. Standardformatet för utdata är det **DateFormat** som är angivet i skriptet.

```
weekend (date [, shift = 0 [,weekoffset = 0]])
```

weekname

Denna funktion returnerar ett värde som visar år och veckonummer med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden på den första dagen i veckan som innehåller **date**.

```
weekname (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden i den sjudagarsperiod som innehåller **date**. Sjudagarsperioder i Qlik Sense definieras genom att 1 januari räknas som den första dagen på veckan.

```
lunarweekstart (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden i den sjudagarsperiod som innehåller **date**. Sjudagarsperioder i Qlik Sense definieras genom att 1 januari räknas som den första dagen på veckan.

```
lunarweekend (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekname

Denna funktion returnerar ett visningsvärde som visar året och sjudagarsperiodsnumret som motsvarar en tidsmarkör för den första millisekunden på den första dagen i sjudagarsperioden som innehåller **date**. Sjudagarsperioder i Qlik Sense definieras genom att 1 januari räknas som den första dagen på veckan.

```
lunarweekname (date [, shift = 0 [,weekoffset = 0]])
```

daystart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör med den första millisekunden av den dag som ingår i argumentet **time**. Det förvalda utdataformatet blir det **TimestampFormat** som har definierats i skriptet.

```
daystart (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden av den dag som ingår i **time**. Det förvalda utdataformatet blir det **TimestampFormat** som har definierats i skriptet.

```
dayend (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayname

Denna funktion returnerar ett värde som visar datumet med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden på dagen som innehåller **time**.

```
dayname (timestamp [, shift = 0 [, dayoffset = 0]])
```

Dagsnumrering

age

Funktionen **age** returnerar åldern vid tidpunkten för **timestamp** (i fyllda år) för någon som är född på **date_of_birth**.

```
age (timestamp, date_of_birth)
```

networkdays

Funktionen **networkdays** returnerar antalet arbetsdagar (måndag-fredag) mellan och inklusive **start_date** och **end_date** med hänsyn till eventuella **holiday**.

```
networkdays  (start:date, end_date {, holiday})
```

firstworkdate

Funktionen **firstworkdate** returnerar senast möjliga startdatum för att uppnå **no_of_workdays** (måndag-fredag) som tar slut senast **end_date** med hänsyn till alla eventuella helgdagar. **end_date** och **holiday** ska vara giltiga datum eller tidsmarkörer.

```
firstworkdate (end_date, no_of_workdays {, holiday} )
```

lastworkdate

Funktionen **lastworkdate** returnerar det tidigaste slutdatumet för att uppnå **no_of_workdays** (måndag-fredag) med början vid **start_date** med hänsyn till alla eventuella **holiday**. **start_date** och **holiday** ska vara giltiga datum eller tidsmarkörer.

```
lastworkdate (start_date, no_of_workdays {, holiday})
```

daynumberofyear

Beräknar dagens nummer på året där tidsmarkören finns. Beräkningen är gjord från den första millisekunden av den första dagen på året, men startpunkten för den första månaden kan flyttas.

```
daynumberofyear (date[,firstmonth])
```

daynumberofquarter

Beräknar dagens nummer i kvartalet där tidsmarkören finns.

```
daynumberofquarter (date[,firstmonth])
```

addmonths

Denna funktion returnerar det datum som infaller **n** månader efter **startdate** eller, om **n** är negativt, det datum som infaller **n** månader före **startdate**.

Syntax:

```
AddMonths(startdate, n , [ , mode])
```

Returnerad datatyp: dual

AddMonths-funktionen returnerar ett dualt värde med både sträng- och talvärdet. Funktionen tar det numeriska värdet i indatauttrycket och genererar en sträng som representerar talet. Strängen visas och det numeriska värdet används för alla numeriska beräkningar och sortering.

Argument:

Argument

Argument	Beskrivning
startdate	Startdatumet som en tidsmarkör, exempelvis '2012-10-12'.
n	Antal månader som ett positivt eller negativt heltal.
mode	Anger om månaden läggs till relativt månadens början eller månadens slut. Standardläget är 0 när den läggs till relativt månadens början. Ange läget 1 när den läggs till relativt månadens slut. När läget är inställt på 1 och indatadatumet är 28 eller högre, kontrollerar funktionen hur många dagar som återstår före månadens slut på startdatumet. Samma antal datar som krävs för att nå månadens slut anges på det returnerade datumet.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`addmonths ('2003-01-29',3)`	returnerar '2003-04-29'
`addmonths ('2003-01-29',3,0)`	returnerar '2003-04-29'
`addmonths ('2003-01-29',3,1)`	returnerar '2003-04-28'
`addmonths ('2003-01-29',1,0)`	returnerar '2003-02-28'
`addmonths ('2003-01-29',1,1)`	returnerar '2003-02-26'
`addmonths ('2003-02-28',1,0)`	returnerar '2003-03-28'
`addmonths ('2003-02-28',1,1)`	returnerar '2003-03-31'
`addmonths ('2003-01-29',-3)`	returnerar '2002-10-29'

addyears

Denna funktion returnerar det datum som infaller **n** år efter **startdate** eller, om **n** är negativt, det datum som infaller**n** år före **startdate**.

Syntax:

```
AddYears(startdate, n)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
startdate	Startdatumet som en tidsmarkör, exempelvis '2012-10-12'.
n	Antal år som ett positivt eller negativt heltal.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`addyears ('2010-01-29',3)`	returnerar '2013-01-29'
`addyears ('2010-01-29',-1)`	returnerar '2009-01-29'

age

Funktionen **age** returnerar åldern vid tidpunkten för **timestamp** (i fyllda år) för någon som är född på **date_of_birth**.

Syntax:

```
age(timestamp, date_of_birth)
```

Kan vara ett uttryck.

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
timestamp	Tidsmarkören, eller uttrycksresultatet som tidsmarkör, som antalet fyllda år ska beräknas fram till.
date_of_birth	Födelsedatum för personen vars ålder beräknas. Kan vara ett uttryck.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`age('25/01/2014', '29/10/2012')`	Returnerar 1.
`age('29/10/2014', '29/10/2012')`	Returnerar 2.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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;
```

Den resulterande tabellen visar de returnerade värdena för age för varje post i tabellen.

Resultattabell

Member	DateOfBirth	Age
John	28/03/1989	26
Linda	10/12/1990	24
Steve	5/2/1992	23
Birg	31/3/1993	22
Raj	19/5/1994	21
Prita	15/9/1994	20
Su	11/12/1994	20
Goran	2/3/1995	20

Member	DateOfBirth	Age
Sunny	14/5/1996	19
Ajoa	13/6/1996	19
Daphne	7/7/1998	17
Biffy	4/8/2000	15

converttolocaltime

Konverterar en UTC- eller GMT-tidsmarkör till lokal tid i form av ett dualt värde. Ett antal platser (städer, platser och tidszoner) i hela världen kan användas.

Syntax:

```
ConvertToLocalTime(timestamp [, place [, ignore_dst=false]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
timestamp	Tidsmarkören, eller uttrycksresultatet som tidsmarkör som ska konverteras.
place	En plats eller tidszon från tabellen över giltiga platser och tidszoner nedan. Du kan också använda GMT ellerUTC för att definiera lokal tid. Följande värden och tidsförskjutningsvärden är giltiga: • GMT • GMT-12:00 - GMT-01:00 • GMT+01:00 - GMT+14:00 • UTC • UTC-12:00 - UTC-01:00 • UTC+01:00 - UTC+14:00 *Du kan bara använda standardinställda tidsförskjutningar. Det går inte att använda en godtycklig tidsförskjutning, till exempelGMT-04:27.*
ignore_dst	Ställ in på True om du vill ignorera DST (sommartid).

Resultatet justeras för sommartid, såvida inte **ignore_dst** har ställts in på True.

Giltiga platser och tidszoner

A-C	D-K	L-R	S-Z
Abu Dhabi	Darwin	La Paz	Samoa
Adelaide	Dhaka	Lima	Santiago
Alaska	Eastern Time (US & Canada)	Lisbon	Sapporo
Amsterdam	Edinburgh	Ljubljana	Sarajevo
Arizona	Ekaterinburg	London	Saskatchewan
Astana	Fiji	Madrid	Seoul
Athens	Georgetown	Magadan	Singapore
Atlantic Time (Canada)	Greenland	Mazatlan	Skopje
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

A-C	D-K	L-R	S-Z
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	-
Chongqing	-	Rome	-
Copenhagen	-	-	-

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`ConvertToLocalTime('2007-11-10 23:59:00','Paris')`	Returnerar '2007-11-11 00:59:00' och motsvarande interna tidsmarkör.
`ConvertToLocalTime(UTC(), 'GMT-05:00')`	Returnerar tidsangivelsen för Nordamerikas ostkust, t.ex. New York.
`ConvertToLocalTime(UTC(), 'GMT-05:00', True)`	Returnerar tidsangivelsen för Nordamerikas ostkust, t.ex. New York, utan justering för sommartid.

day

Denna funktion returnerar ett heltal som motsvarar dagen om decimaldelen av **expression** tolkas som datum enligt standardtolkningen av tal.

Funktionen returnerar dagen i månaden för ett visst datum. Den används vanligtvis för att härleda ett dagfält som en del av en kalenderdimension.

Syntax:

```
day(expression)
```

Returnerad datatyp: heltal

Exempel på funktioner

Exempel	Resultat
`day( '1971-10-12' )`	returnerar 12
`day( '35648' )`	returnerar 6, eftersom 35648 = 1997-08-06

dayend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden av den dag som ingår i **time**. Det förvalda utdataformatet blir det **TimestampFormat** som har definierats i skriptet.

Syntax:

```
DayEnd(time[, [period_no[, day_start]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
time	Tidsmarkören som ska evalueras.
period_no	**period_no** är ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den dag som innehåller **time**. Negativa värden i **period_no** anger föregående dagar och positiva värden anger efterföljande dagar.
day_start	Om du vill ange att dagar inte startar vid midnatt anger du en startpunkt som delar av en dag i **day_start**. Till exempel betecknar 0,125 03.00.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`dayend('25/01/2013 16:45:00')`	Returnerar 25/01/2013 23:59:59.
`dayend('25/01/2013 16:45:00', -1)`	Returnerar 24/01/2013 23:59:59.
`dayend('25/01/2013 16:45:00', 0, 0.5)`	Returnerar 26/01/2013 11:59:59.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet hittas tidsmarkören som markerar slutet på dagen efter varje faktureringsdatum i tabellen.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
DayEnd(InvDate, 1) AS DEnd
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för dayend ()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	DEnd
28/03/2012	29/03/2012 23:59:59
10/12/2012	11/12/2012 23:59:59
5/2/2013	07/02/2013 23:59:59
31/3/2013	01/04/2013 23:59:59
19/5/2013	20/05/2013 23:59:59
15/9/2013	16/09/2013 23:59:59
11/12/2013	12/12/2013 23:59:59
2/3/2014	03/03/2014 23:59:59
14/5/2014	15/05/2014 23:59:59
13/6/2014	14/06/2014 23:59:59

InvDate	DEnd
7/7/2014	08/07/2014 23:59:59
4/8/2014	05/08/2014 23:59:59

daylightsaving

Returnerar den aktuella anpassningen till sommartid/vintertid enligt Windows definition.

Syntax:

```
DaylightSaving( )
```

Returnerad datatyp: dual

Exempel:

```
daylightsaving( )
```

dayname

Denna funktion returnerar ett värde som visar datumet med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden på dagen som innehåller **time**.

Syntax:

```
DayName(time[, period_no [, day_start]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
time	Tidsmarkören som ska evalueras.
period_no	**period_no** är ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den dag som innehåller **time**. Negativa värden i **period_no** anger föregående dagar och positiva värden anger efterföljande dagar.
day_start	Om du vill ange att dagar inte startar vid midnatt anger du en startpunkt som delar av en dag i **day_start**. Till exempel betecknar 0,125 03.00.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`dayname('25/01/2013 16:45:00')`	Returnerar 25/01/2013.
`dayname('25/01/2013 16:45:00', -1)`	Returnerar 24/01/2013.
`dayname('25/01/2013 16:45:00', 0, 0.5 )`	Returnerar 25/01/2013. När du visar hela tidsmarkören kan du se att den motsvarar25/01/2013 12:00:00.000.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet skapas dagsnamnet utifrån tidsmarkören som markerar början på dagen efter varje faktureringsdatum i tabellen.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
DayName(InvDate, 1) AS DName
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för dayname()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	DName
28/03/2012	29/03/2012 00:00:00
10/12/2012	11/12/2012 00:00:00

InvDate	DName
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

Beräknar dagens nummer i kvartalet där tidsmarkören finns.

Syntax:

```
DayNumberOfQuarter(timestamp[,start_month])
```

Returnerad datatyp: heltal

Funktionen utgår alltid från år med 366 dagar.

Argument:

Argument

Argument	Beskrivning
timestamp	Det datum som ska evalueras.
start_month	Genom att ange en **start_month** mellan 2 och 12 (1 om parametern utelämnas), kan årets början flyttas fram till första dagen på valfri månad. Om du exempelvis vill börja ett budgetår den 1 mars anger du **start_month** som 3.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`DayNumberOfQuarter('12/09/2014')`	Returnerar 74, dagens nummer för innevarande kvartal.

Exempel	Resultat
DayNumberOfQuarter ('12/09/2014',3)	Returnerar 12, dagens nummer för innevarande kvartal. I det här fallet börjar det första kvartalet med mars (eftersom start_month har angetts som 3). Det innebär att det innevarande kvartalet är det tredje kvartalet, som började den 1 september.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

```
ProjectTable:
LOAD recno() as InvID, * INLINE [
StartDate
28/03/2014
10/12/2014
5/2/2015
31/3/2015
19/5/2015
15/9/2015
] ;
NrDays:
Load *,
DayNumberOfQuarter(StartDate,4) As DayNrQtr
Resident ProjectTable;
Drop table ProjectTable;
```

Den resulterande tabellen visar de returnerade värdena för DayNumberOfQuarter för varje post i tabellen.

Resultattabell

InvID	StartDate	DayNrQtr
1	28/03/2014	88
2	10/12/2014	71
3	5/2/2015	36
4	31/3/2015	91
5	19/5/2015	49
6	15/9/2015	77

daynumberofyear

Beräknar dagens nummer på året där tidsmarkören finns. Beräkningen är gjord från den första millisekunden av den första dagen på året, men startpunkten för den första månaden kan flyttas.

Syntax:

```
DayNumberOfYear(timestamp[,start_month])
```

Returnerad datatyp: heltal

Funktionen utgår alltid från år med 366 dagar.

Argument:

Argument

Argument	Beskrivning
timestamp	Det datum som ska evalueras.
start_month	Genom att ange en **start_month** mellan 2 och 12 (1 om parametern utelämnas), kan årets början flyttas fram till första dagen på valfri månad. Om du exempelvis vill börja ett budgetår den 1 mars anger du **start_month** som 3.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`DayNumberOfYear('12/09/2014')`	Returnerar 256, dagens nummer räknat från den första dagen på året.
`DayNumberOfYear ('12/09/2014',3)`	Returnerar 196, dagens nummer räknat från 1 mars.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

```
ProjectTable:
LOAD recno() as InvID, * INLINE [
StartDate
28/03/2014
10/12/2014
5/2/2015
31/3/2015
19/5/2015
15/9/2015
] ;
NrDays:
Load *,
DayNumberOfYear(StartDate,4) As DayNrYear
Resident ProjectTable;
Drop table ProjectTable;
```

Den resulterande tabellen visar de returnerade värdena för DayNumberOfYear för varje post i tabellen.

Resultattabell

InvID	StartDate	DayNrYear
1	28/03/2014	363
2	10/12/2014	254
3	5/2/2015	311
4	31/3/2015	366
5	19/5/2015	49
6	15/9/2015	168

daystart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör med den första millisekunden av den dag som ingår i argumentet **time**. Det förvalda utdataformatet blir det **TimestampFormat** som har definierats i skriptet.

Syntax:

```
DayStart(time[, [period_no[, day_start]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
time	Tidsmarkören som ska evalueras.
period_no	**period_no** är ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den dag som innehåller **time**. Negativa värden i **period_no** anger föregående dagar och positiva värden anger efterföljande dagar.
day_start	Om du vill ange att dagar inte startar vid midnatt anger du en startpunkt som delar av en dag i **day_start**. Till exempel betecknar 0,125 03.00.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`daystart('25/01/2013 16:45:00')`	Returnerar 25/01/2013 00:00:00.
`daystart('25/01/2013 16:45:00', -1)`	Returnerar 24/01/2013 00:00:00.
`daystart('25/01/2013 16:45:00', 0, 0.5)`	Returnerar 25/01/2013 12:00:00.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet hittas tidsmarkören som markerar början på dagen efter varje faktureringsdatum i tabellen.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
DayStart(InvDate, 1) AS DStart
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för daystart ()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	DStart
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

InvDate	DStart
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

firstworkdate

Funktionen **firstworkdate** returnerar senast möjliga startdatum för att uppnå **no_of_workdays** (måndag-fredag) som tar slut senast **end_date** med hänsyn till alla eventuella helgdagar. **end_date** och **holiday** ska vara giltiga datum eller tidsmarkörer.

Syntax:

```
firstworkdate(end_date, no_of_workdays {, holiday} )
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
end_date	Tidsmarkör för det slutdatum som ska utvärderas.
no_of_ workdays	Antalet arbetsdagar som ska uppnås.
holiday	Ledighetsperioder som ska undantas från arbetsdagar. En ledighetsperiod anges som ett startdatum och ett slutdatum, avgränsade med kommatecken. **Exempel:** `'25/12/2013', '26/12/2013'` Du kan ange fler än en ledighetsperiod, avgränsade med kommatecken. **Exempel:** `'25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014'`

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`firstworkdate ('29/12/2014', 9)`	Returnerar 17/12/2014.
`firstworkdate ('29/12/2014', 9, '25/12/2014', '26/12/2014')`	Returnerar 15/12/2014 eftersom en ledighetsperiod på två dagar räknas in.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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;
```

Den resulterande tabellen visar de returnerade värdena för FirstWorkDate för varje post i tabellen.

Resultattabell

InvID	EndDate	StartDate
1	28/03/2015	13/10/2014
2	10/12/2015	26/06/2015
3	5/2/2016	24/08/2015
4	31/3/2016	16/10/2015
5	19/5/2016	04/12/2015
6	15/9/2016	01/04/2016

GMT

Denna funktion returnerar aktuell Greenwich Mean Time utifrån systemklockan och Windows-tidsinställningarna.

Syntax:

```
GMT( )
```

Returnerad datatyp: dual

Exempel:

```
gmt( )
```

hour

Denna funktion returnerar ett heltal som motsvarar timmen om decimaldelen av **expression** tolkas som en tidpunkt enligt standardtolkningen av tal.

Syntax:

```
hour(expression)
```

Returnerad datatyp: heltal

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`hour( '09:14:36' )`	returnerar 9
`hour( '0.5555' )`	returnerar 13 (eftersom 0.5555 = 13:19:55)

inday

Denna funktion returnerar True om **timestamp** ligger inom den dag som innehåller **base_timestamp**.

Syntax:

```
InDay (timestamp, base_timestamp, period_no[, day_start])
```

Returnerad datatyp: Boolesk

Argument:

Argument

Argument	Beskrivning
timestamp	Det datum och den tid du vill jämföra mot **base_timestamp**.
base_timestamp	Datum och tid som används för att utvärdera tidsmarkören.
period_no	Dagens startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger den dag som innehåller **base_timestamp**. Negativa värden i **period_no** anger föregående dagar och positiva värden anger efterföljande dagar.

Argument	Beskrivning
day_start	Om du vill använda dagar som inte startar vid midnatt kan du ange en startpunkt i delar av en dag med **day_start**, till exempel 0,125 för att beteckna 03.00.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`inday ('12/01/2006 12:23:00', '12/01/2006 00:00:00', 0)`	Returnerar True
`inday ('12/01/2006 12:23:00', '13/01/2006 00:00', 0)`	Returnerar False
`inday ('12/01/2006 12:23:00', '12/01/2006 00:00:00', -1)`	Returnerar False
`inday ('11/01/2006 12:23:00', '12/01/2006 00:00:00', -1)`	Returnerar True
`inday ('12/01/2006 12:23:00', '12/01/2006 00:00:00', 0, 0.5)`	Returnerar False
`inday ('12/01/2006 11:23:00', '12/01/2006 00:00:00', 0, 0.5)`	Returnerar True

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet undersöker vi om ett faktureringsdatum infaller under dagen som startar med base_ timestamp.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvTime
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 *,
InDay(InvTime, '28/03/2012 00:00:00', 0) AS InDayEx
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för inday()-funktionen.

Resultattabell

InvTime	InDayEx
28/03/2012	-1 (True)
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

indaytotime

Denna funktion returnerar True om **timestamp** ligger inom den del av dagen som innehåller **base_timestamp** fram till och inklusive den exakta millisekunden för **base_timestamp**.

Syntax:

```
InDayToTime (timestamp, base_timestamp, period_no[, day_start])
```

Returnerad datatyp: Boolesk

Argument:

Argument

Argument	Beskrivning
timestamp	Det datum och den tid du vill jämföra mot **base_timestamp**.
base_ timestamp	Datum och tid som används för att utvärdera tidsmarkören.
period_no	Dagens startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger den dag som innehåller **base_timestamp**. Negativa värden i **period_no** anger föregående dagar och positiva värden anger efterföljande dagar.
day_start	(valbar) Om du vill använda dagar som inte startar vid midnatt kan du ange en startpunkt i delar av en dag med **day_start**, till exempel 0,125 för att beteckna 03.00.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`indaytotime ('12/01/2006 12:23:00', '12/01/2006 23:59:00', 0)`	Returnerar True
`indaytotime ('12/01/2006 12:23:00', '12/01/2006 00:00:00', 0)`	Returnerar False
`indaytotime ('11/01/2006 12:23:00', '12/01/2006 23:59:00', -1)`	Returnerar True

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet undersöker vi om en fakturas tidsmarkör infaller innan 17:00:00 under dagen som startar med base_timestamp.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvTime
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 *,
InDayToTime(InvTime, '28/03/2012 17:00:00', 0) AS InDayExTT
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för indaytotime()-funktionen.

Resultattabell

InvTime	InDayExTT
28/03/2012	-1 (True)
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	0 (False)

InvTime	InDayExTT
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inlunarweek

Med denna funktion får vi reda på om **timestamp** ligger inom den sjudagarsperiod som innehåller **base_date**. Sjudagarsperioder i Qlik Sense definieras genom att 1 januari räknas som den första dagen på veckan.

Syntax:

```
InLunarWeek (timestamp, base_date, period_no[, first_week_day])
```

Returnerad datatyp: Boolesk

Argument:

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Data som används för att utvärdera sjudagarsperioden.
period_no	Sjudagarsperiodens startpunkt kan flyttas med **period_no**. period_no är ett heltal där värdet 0 anger den sjudagarsperiod som innehåller **base_date**. Negativa värden i **period_no** anger föregående sjudagarsperioder och positiva värden anger efterföljande sjudagarsperioder.
first_week_day	En förflyttning av startpunkten som kan vara större eller mindre än noll. Detta flyttar början på året med det angivna antalet dagar och/eller delar av en dag.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`inlunarweek ('12/01/2013', '14/01/2013', 0)`	Returnerar True. Eftersom värdet förtimestamp, 12/01/2013 infaller under veckan 08/01/2013 till 14/01/2013.
`inlunarweek ('12/01/2013', '07/01/2013', 0)`	Returnerar False. Eftersom base_date 07/01/2013 är sjudagarsperioden som är definierad som 01/01/2013 till 07/01/2013.
`inlunarweek ('12/01/2013', '14/01/2013', -1)`	Returnerar False. Eftersom att ange ett värde för period_no som -1 flyttar veckan till den föregående veckan, 01/01/2013 till 07/01/2013.
`inlunarweek ('07/01/2013', '14/01/2013', -1)`	Returnerar True. Jämfört med föregående exempel är tidsmarkören i veckan, efter att ha tagit med flytten bakåt i beräkningen.
`inlunarweek ('11/01/2006', '08/01/2006', 0, 3)`	Returnerar False. Eftersom att angefirst_week_day som 3 innebär att årets start beräknas från 04/01/2013, så värdet för base_date infaller i den första veckan, och värdet för timestamp infaller i veckan 11/01/2013 till 17/01/2013.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet undersöker vi om ett faktureringsdatum infaller inom veckan som flyttats från värdet för base_date med fyra veckor.

```
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 *,
InLunarWeek(InvDate, '11/01/2013', 4) AS InLWeekPlus4
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för inlunarweek()-funktionen.

Funktionen returnerar True för värdet för InvDate5/2/2013 eftersom värdet för base_date, 11/01/2013, är flyttat med fyra veckor och således infaller under veckan 5/02/2013 till 11/02/2013.

Resultattabell

InvDate	InLWeekPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inlunarweektodate

Denna funktion tar reda på om **timestamp** ligger inom delen av sjudagarsperioden fram till och inklusive den sista millisekunden av **base_date**. Sjudagarsperioder i Qlik Sense definieras genom att 1 januari räknas som den första dagen på veckan.

Syntax:

```
InLunarWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Returnerad datatyp: Boolesk

Argument:

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Data som används för att utvärdera sjudagarsperioden.

Argument	Beskrivning
period_no	Sjudagarsperiodens startpunkt kan flyttas med **period_no**. period_no är ett heltal där värdet 0 anger den sjudagarsperiod som innehåller **base_date**. Negativa värden i **period_no** anger föregående sjudagarsperioder och positiva värden anger efterföljande sjudagarsperioder.
first_week_day	En förflyttning av startpunkten som kan vara större eller mindre än noll. Detta flyttar början på året med det angivna antalet dagar och/eller delar av en dag.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`inlunarweektodate ('12/01/2013', '13/01/2013', 0)`	Returnerar True. Eftersom värdet förtimestamp, 12/01/2013 infaller under veckodelen 08/01/2013 till 13/01/2013.
`inlunarweektodate ('12/01/2013', '11/01/2013', 0)`	Returnerar False. Eftersom värdet för timestamp är senare än värdet för base_date trots att båda datumen infaller under samma sjudagarsperiod innan 12/01/2012.
`inlunarweektodate ('12/01/2006', '05/01/2006', 1)`	Returnerar True. Att ange värdet 1 för period_no flyttar base_date framåt en vecka, så att värdet för timestamp Infaller under en del av sjudagarsperioden.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet undersöker vi om ett faktureringsdatum infaller inom veckodelen som flyttats från värdet för base_date med fyra veckor.

```
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 *,
```

```
InLunarWeekToDate(InvDate, '07/01/2013', 4) AS InLWeek2DPlus4
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för inlunarweek()-funktionen.

Funktionen returnerar True för värdet för InvDate5/2/2013 eftersom värdet för base_date, 11/01/2013, är flyttat med fyra veckor och således infaller under veckodelen 5/02/2013 till 07/02/2013.

Resultattabell

InvDate	InLWeek2DPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inmonth

Denna funktion returnerar True om **timestamp** ligger inom den månad som innehåller **base_date**.

Syntax:

```
InMonth (timestamp, base_date, period_no[, first_month_of_year])
```

Returnerad datatyp: Boolesk

Argument:

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera månaden.

Argument	Beskrivning
period_no	Månadens startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger den månad som innehåller **base_date**. Negativa värden i **period_no** anger föregående månader och positiva värden anger efterföljande månader.
first_month_of_year	Parametern **first_month_of_year** är inaktiverad och reserverad för framtida användning.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`inmonth ('25/01/2013', '01/01/2013', 0 )`	Returnerar True
`inmonth('25/01/2013', '01/04/2013', 0)`	Returnerar False
`inmonth ('25/01/2013', '01/01/2013', -1)`	Returnerar False
`inmonth ('25/12/2012', '01/01/2013', -1)`	Returnerar True

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet undersöker vi om ett faktureringsdatum infaller under den fjärde månaden efter månaden i base_date, genom att ange period_no som 4.

```
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 *,
InMonth(InvDate, '31/01/2013', 4) AS InMthPlus4
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för inmonth ()-funktionen.

Resultattabell

InvDate	InMthPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	0 (False)
19/5/2013	-1 (True)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inmonths

Med den här funktionen får vi reda på om en tidsmarkör finns inom samma månad, tvåmånadersperiod, kvartal, tredjedels år eller halvår som ett basdatum. Det går även att se om tidsmarkören finns inom en föregående eller senare tidsperiod.

Syntax:

```
InMonths(n_months, timestamp, base_date, period_no [, first_month_of_year])
```

Returnerad datatyp: Boolesk

Argument:

Argument

Argument	Beskrivning
n_months	Antalet månader som definierar perioden. 1 (motsvaras av inmonth()-funktionen), 2 (tvåmånadersperiod), 3 (motsvaras av inquarter()-funktionen), 4 (tredjedels år), eller 6 (halvår).
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera perioden.

Argument	Beskrivning
period_no	Periodens startpunkt kan flyttas med **period_no**, ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den period som innehåller **base_date**. Negativa värden i **period_no** anger föregående perioder och positiva värden anger efterföljande perioder.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`inmonths(4, '25/01/2013', '25/04/2013', 0)`	Returnerar True. Eftersom värdet för timestamp, 25/01/2013, befinner sig inom fyramånadersperioden 01/01/2013 till 30/04/2013, där värdet för base_date, 25/04/2013 finns.
`inmonths(4, '25/05/2013', '25/04/2013', 0)`	Returnerar False. Eftersom 25/05/2013 är utanför samma period som i förra exemplet.
`inmonths(4, '25/11/2012', '01/02/2013', -1 )`	Returnerar True. Eftersom värdet för period_no, -1, flyttar sökperioden bakåt en period på fyra månader (värdet för n-months), vilket gör sökperioden till 01/09/2012 till 31/12/2012.
`inmonths( 4, '25/05/2006', '01/03/2006', 0, 3)`	Returnerar True. Eftersom värdet för first_month_of_year är angett som 3, vilket gör sökperioden till 01/03/2006 till 30/07/2006 istället för 01/01/2006 till 30/04/2006.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet undersöker vi om faktureringsdatumet i tabellen infaller under tvåmånadersperioden som innefattar base_date framflyttad en tvåmånadersperiod (genom att ange period_no som 1).

```
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 *,
InMonths(2, InvDate, '11/02/2013', 1) AS InMthsPlus1
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för InMonths()-funktionen.

Sökperioden är 01/03/2013 till 30/04/2013, eftersom värdet för base_date flyttas framåt två månader från värdet i funktionen (11/02/2013).

Resultattabell

InvDate	InMthsPlus1
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	-1 (True)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inmonthstodate

Med den här funktionen får vi reda på om en tidsmarkör finns inom delen av månaden, tvåmånadersperioden, kvartalet, tredjedelens år eller halvåret fram till och inklusive den sista millisekunden av **base_date**. Det går även att se om tidsmarkören finns inom en föregående eller senare tidsperiod.

Syntax:

```
InMonths (n_months, timestamp, base_date, period_no[, first_month_of_year ])
```

Returnerad datatyp: Boolesk

Argument:

Argument

Argument	Beskrivning
n_months	Antalet månader som definierar perioden. 1 (motsvaras av inmonth()-funktionen), 2 (tvåmånadersperiod), 3 (motsvaras av inquarter()-funktionen), 4 (tredjedels år), eller 6 (halvår).
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera perioden.
period_no	Periodens startpunkt kan flyttas med **period_no**, ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den period som innehåller **base_date**. Negativa värden i **period_no** anger föregående perioder och positiva värden anger efterföljande perioder.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`inmonthstodate(4, '25/01/2013', '25/04/2013', 0)`	Returnerar True. Eftersom värdet för timestamp, 25/01/2013, befinner sig inom fyramånadersperioden 01/01/2013 till slutet på 25/04/2013, där värdet för base_date, 25/04/2013 finns.
`inmonthstodate(4, '26/04/2013', '25/04/2006', 0)`	Returnerar False. Eftersom 26/04/2013 är utanför samma period som i förra exemplet.

Exempel	Resultat
`inmonthstodate(4, '25/09/2005', '01/02/2006', -1)`	Returnerar True. Eftersom värdet för period_no, -1, flyttar sökperioden bakåt en period på fyra månader (värdet för n-months), vilket gör sökperioden till 01/09/2005 till 01/02/2006.
`inmonthstodate(4, '25/04/2006', '01/06/2006', 0, 3)`	Returnerar True. Eftersom värdet för first_month_of_year är angett som 3, vilket gör sökperioden till 01/03/2006 till 01/06/2006 istället för 01/05/2006 till 01/06/2006.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet undersöker vi om faktureringsdatumet i tabellen infaller under delen av tvåmånadersperioden som sträcker sig till och innefattar base_date framflyttad fyra tvåmånadersperioder (genom att ange period_no som 4).

```
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 *,
InMonthsToDate(2, InvDate, '15/02/2013', 4) AS InMths2DPlus4
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för InMonths()-funktionen.

Sökperioden är 01/09/2013 till 15/10/2013, eftersom värdet för base_date flyttas framåt åtta månader från värdet i funktionen (15/02/2013).

Resultattabell

InvDate	InMths2DPlus4
28/03/2012	0 (False)

InvDate	InMths2DPlus4
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	-1 (True)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inmonthtodate

Returnerar True om **date** ligger inom den del av månaden som innehåller **basedate** fram till och inklusive den sista millisekunden av **basedate**.

Syntax:

```
InMonthToDate (timestamp, base_date, period_no)
```

Returnerad datatyp: Boolesk

Argument:

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera månaden.
period_no	Månadens startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger den månad som innehåller **base_date**. Negativa värden i **period_no** anger föregående månader och positiva värden anger efterföljande månader.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`inmonthtodate ('25/01/2013', '25/01/2013', 0)`	Returnerar True

Exempel	Resultat
`inmonthtodate ('25/01/2013', '24/01/2013', 0)`	Returnerar False
`inmonthtodate ('25/01/2013', '28/02/2013', -1)`	Returnerar True

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

Genom att du anger period_no som 4 kontrollerar vi i det här exemplet om ett faktureringsdatum infaller i den fjärde månaden efter månaden i base_date men innan slutet på dagen som angetts i base_date.

```
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 *,
InMonthToDate(InvDate, '31/01/2013', 4) AS InMthPlus42D
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för inmonthtodate()-funktionen.

Resultattabell

InvDate	InMthPlus42D
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	0 (False)
31/3/2013	0 (False)
19/5/2013	-1 (True)
15/9/2013	0 (False)

InvDate	InMthPlus42D
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inquarter

Denna funktion returnerar True om **timestamp** ligger inom det kvartal som innehåller **base_date**.

Syntax:

```
InQuarter (timestamp, base_date, period_no[, first_month_of_year])
```

Returnerad datatyp: Boolesk

Argument:

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera kvartalet.
period_no	Kvartalets startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger det kvartal som innehåller **base_date**. Negativa värden i **period_no** anger föregående kvartal och positiva värden anger efterföljande kvartal.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`inquarter ('25/01/2013', '01/01/2013', 0)`	Returnerar True
`inquarter ('25/01/2013', '01/04/2013', 0)`	Returnerar False
`inquarter ('25/01/2013', '01/01/2013', -1)`	Returnerar False
`inquarter ('25/12/2012', '01/01/2013', -1)`	Returnerar True

Exempel	Resultat
`inquarter ('25/01/2013', '01/03/2013', 0, 3)`	Returnerar False
`inquarter ('25/03/2013', '01/03/2013', 0, 3)`	Returnerar True

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet kontrollerar vi om ett faktureringsdatum infaller under det fjärde kvartalet av budgetåret genom att ange värdet förfirst_month_of_year som 4, och ha base_date 31/01/2013.

```
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 *,
InQuarter(InvDate, '31/01/2013', 0, 4) AS Qtr4FinYr1213
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för inquarter ()-funktionen.

Resultattabell

InvDate	Qtr4Fin1213
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)
31/3/2013	-1 (True)
19/5/2013	0 (False)
15/9/2013	0 (False)

InvDate	Qtr4Fin1213
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inquartertodate

Denna funktion returnerar True om **timestamp** ligger inom den del av det kvartal som innehåller **base_date** fram till och inklusive den sista millisekunden av **base_date**.

Syntax:

```
InQuarterToDate (timestamp, base_date, period_no [, first_month_of_year])
```

Returnerad datatyp: Boolesk

Argument:

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera kvartalet.
period_no	Kvartalets startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger det kvartal som innehåller **base_date**. Negativa värden i **period_no** anger föregående kvartal och positiva värden anger efterföljande kvartal.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`inquartertodate ('25/01/2013', '25/01/2013', 0)`	Returnerar True
`inquartertodate ( 25/01/2013', '24/01/2013', 0)`	Returnerar False
`inquartertodate ('25/01/2012', '01/02/2013', -1)`	Returnerar True

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet kontrollerar vi om ett faktureringsdatum infaller under ett budgetår som angetts genom att ange värdet för first_month_of_year som 4, och i det fjärde kvartalet, innan slutet på 28/02/2013.

```
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 *,
InQuarterToDate(InvDate, '28/02/2013', 0, 4) AS Qtr42Date
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för inquartertodate()-funktionen.

Resultattabell

InvDate	Qtr42Date
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)

InvDate	Qtr42Date
7/7/2014	0 (False)
4/8/2014	0 (False)

inweek

Denna funktion returnerar True om **timestamp** ligger inom den vecka som innehåller **base_date**.

Syntax:

```
InWeek (timestamp, base_date, period_no[, first_week_day])
```

Returnerad datatyp: Boolesk

Argument:

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera veckan.
period_no	Veckans startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger den vecka som innehåller **base_date**. Negativa värden i **period_no** anger föregående veckor och positiva värden anger efterföljande veckor.
first_week_day	Som standardinställning är veckans första dag måndag, och börjar vid midnatt mellan söndag och måndag. Om du vill att veckan ska starta en annan dag anger du den flyttade startpunkten med **first_week_day**. Denna kan ges som ett helt antal dagar och/eller delar av en dag.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`inweek ('12/01/2006', '14/01/2006', 0)`	Returnerar True
`inweek ('12/01/2006', '20/01/2006', 0 )`	Returnerar False
`inweek ('12/01/2006', '14/01/2006', -1 )`	Returnerar False
`inweek ('07/01/2006', '14/01/2006', -1)`	Returnerar True
`inweek ('12/01/2006', '09/01/2006', 0, 3)`	Returnerar False Eftersom first_week_day är angett som 3 (torsdag), vilket gör 12/01/2006 till den första dagen i veckan efter veckan som innehåller 09/01/2006.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet undersöker vi om ett faktureringsdatum infaller under den fjärde veckan efter veckan i base_date, genom att ange period_no som 4.

```
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 *,
InWeek(InvDate, '11/01/2013', 4) AS InWeekPlus4
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för inweek ()-funktionen.

InvDate5/2/2013 infaller under veckan som ligger fyra veckor efter base_date: 11/1/2013.

Resultattabell

InvDate	InWeekPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)

InvDate	InWeekPlus4
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inweektodate

Denna funktion returnerar True om **timestamp** ligger inom den del av veckan som innehåller **base_date** fram till och inklusive den sista millisekunden av **base_date**.

Syntax:

```
InWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Returnerad datatyp: Boolesk

Argument:

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera veckan.
period_no	Veckans startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger den vecka som innehåller **base_date**. Negativa värden i **period_no** anger föregående veckor och positiva värden anger efterföljande veckor.
first_week_day	Som standardinställning är veckans första dag måndag, och börjar vid midnatt mellan söndag och måndag. Om du vill att veckan ska starta en annan dag anger du den flyttade startpunkten med **first_week_day**. Denna kan ges som ett helt antal dagar och/eller delar av en dag.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`inweektodate ('12/01/2006', '12/01/2006', 0)`	Returnerar True
`inweektodate ('12/01/2006', '11/01/2006', 0)`	Returnerar False

Exempel	Resultat
`inweektodate ('12/01/2006', '18/01/2006', -1)`	Returnerar False Eftersom period_no är angett som -1, är de faktiska data som timestamp jämförs med 11/01/2006.
`inweektodate ( '11/01/2006', '12/01/2006', 0, 3 )`	Returnerar False Eftersom first_week_day är angett som 3 (torsdag), vilket gör 12/01/2006 till den första dagen i veckan efter veckan som innehåller 12/01/2006.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet undersöker vi om ett faktureringsdatum infaller under den fjärde veckan efter veckan i base_date, genom att ange period_no som 4, men innan värdet för base_date.

```
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 *,
InWeekToDate(InvDate, '11/01/2013', 4) AS InWeek2DPlus4
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för inweek ()-funktionen.

Resultattabell

InvDate	InWeek2DPlus4
28/03/2012	0 (False)
10/12/2012	0 (False)
5/2/2013	-1 (True)

InvDate	InWeek2DPlus4
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inyear

Denna funktion returnerar True om **timestamp** ligger inom det år som innehåller **base_date**.

Syntax:

```
InYear (timestamp, base_date, period_no [, first_month_of_year])
```

Returnerad datatyp: Boolesk

Argument:

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera året.
period_no	Årets startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger året som innehåller **base_date**. Negativa värden i **period_no** anger föregående år och positiva värden anger efterföljande år.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`inyear ('25/01/2013', '01/01/2013', 0 )`	Returnerar True
`inyear ('25/01/2012', '01/01/2013', 0)`	Returnerar False
`inyear ('25/01/2013', '01/01/2013', -1)`	Returnerar False
`inyear ('25/01/2012', '01/01/2013', -1 )`	Returnerar True
`inyear ('25/01/2013', '01/01/2013', 0, 3)`	Returnerar True Värdet för base_date och first_month_of_year anger att timestamp måste infalla inom 01/03/2012 och 28/02/2013
`inyear ('25/03/2013', '01/07/2013', 0, 3 )`	Returnerar True

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet kontrollerar vi om ett faktureringsdatum infaller under budgetåret genom att ange värdet för first_month_of_year som 4, och ha base_date mellan 1/4/2012 och 31/03/2013.

```
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
];
```

Testa om InvDate är inom budgetåret 1/04/2012 till 31/03/2013:

```
InvoiceData:
LOAD *,
InYear(InvDate, '31/01/2013', 0, 4) AS FinYr1213
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för inyear()-funktionen.

Resultattabell

InvDate	FinYr1213
28/03/2012	0 (False)
10/12/2012	-1 (True)
5/2/2013	-1 (True)
31/3/2013	-1 (True)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

inyeartodate

Denna funktion returnerar True om **timestamp** ligger inom den del av året som innehåller **base_date** fram till och inklusive den sista millisekunden av **base_date**.

Syntax:

```
InYearToDate (timestamp, base_date, period_no[, first_month_of_year])
```

Returnerad datatyp: Boolesk

Argument:

Argument

Argument	Beskrivning
timestamp	Det datum du vill jämföra mot **base_date**.
base_date	Datum som används för att utvärdera året.
period_no	Årets startpunkt kan flyttas med **period_no**. **period_no** är ett heltal där värdet 0 anger året som innehåller **base_date**. Negativa värden i **period_no** anger föregående år och positiva värden anger efterföljande år.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`inyeartodate ('2013/01/25', '2013/02/01', 0)`	Returnerar True
`inyeartodate ('2012/01/25', '2013/01/01', 0)`	Returnerar False
`inyeartodate ('2012/01/25', '2013/02/01', -1)`	Returnerar True
`inyeartodate ('2012/11/25', '2013/01/31', 0, 4)`	Returnerar True Värdet för timestamp infaller under budgetåret som börjar i den fjärde månaden och innan värdet för base_date.
`inyeartodate ( '2013/3/31', '2013/01/31', 0, 4 )`	Returnerar False Jämfört med det förra exemplet är värdet för timestamp fortfarande inom budgetåret, men efter datumet för base_date, så det infaller utanför delen av året.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet kontrollerar vi om ett faktureringsdatum infaller under ett budgetår som angetts genom att ange värdet förfirst_month_of_year som 4, och under delen av året innan slutet på 31/01/2013.

```
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 *,
InYearToDate(InvDate, '31/01/2013', 0, 4) AS FinYr2Date
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för inyeartodate()-funktionen.

Resultattabell

InvDate	FinYr2Date
28/03/2012	0 (False)
10/12/2012	-1 (True)
5/2/2013	0 (False)
31/3/2013	0 (False)
19/5/2013	0 (False)
15/9/2013	0 (False)
11/12/2013	0 (False)
2/3/2014	0 (False)
14/5/2014	0 (False)
13/6/2014	0 (False)
7/7/2014	0 (False)
4/8/2014	0 (False)

lastworkdate

Funktionen **lastworkdate** returnerar det tidigaste slutdatumet för att uppnå **no_of_workdays** (måndag-fredag) med början vid **start_date** med hänsyn till alla eventuella **holiday**. **start_date** och **holiday** ska vara giltiga datum eller tidsmarkörer.

Syntax:

```
lastworkdate(start_date, no_of_workdays {, holiday})
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
start_date	Startdatum som ska utvärderas.
no_of_workdays	Antalet arbetsdagar som ska uppnås.

Argument	Beskrivning
holiday	Ledighetsperioder som ska undantas från arbetsdagar. En ledighetsperiod anges som ett startdatum och ett slutdatum, avgränsade med kommatecken. **Exempel:** `'25/12/2013', '26/12/2013'` Du kan ange fler än en ledighetsperiod, avgränsade med kommatecken. **Exempel:** `'25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014'`

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`lastworkdate ('19/12/2014', 9)`	Returnerar '31/12/2014'
`lastworkdate ('19/12/2014', 9, '2014-12-25', '2014-12-26')`	Returnerar "02/01/2015" då en ledighetsperiod på två dagar räknas in.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

```
ProjectTable:
LOAD *, recno() as InvID, INLINE [
StartDate
28/03/2014
10/12/2014
5/2/2015
31/3/2015
19/5/2015
15/9/2015
] ;
NrDays:
Load *,
LastWorkDate(StartDate,120) As EndDate
Resident ProjectTable;
Drop table ProjectTable;
```

Den resulterande tabellen visar de returnerade värdena för LastWorkDate för varje post i tabellen.

Resultattabell

InvID	StartDate	EndDate
1	28/03/2014	11/09/2014
2	10/12/2014	26/05/2015
3	5/2/2015	27/07/2015
4	31/3/2015	14/09/2015
5	19/5/2015	02/11/2015
6	15/9/2015	29/02/2016

localtime

Denna funktion returnerar en tidsmarkör för aktuell tid från systemklockan för en angiven tidszon.

Syntax:

```
LocalTime([timezone [, ignoreDST ]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
timezone	**timezone**anges som en sträng som innehåller någon av de geografiska platser som listas under **Time Zone** i **Windows Control Panel** för **Date and Time** eller som en sträng i formatet "GMT+tt:mm". Om ingen tidszon anges returneras den lokala tiden.
ignoreDST	Om **ignoreDST** är -1 (True) ignoreras sommartid.

Exempel och resultat:

Exemplen nedan bygger på den funktion som anropas den 2014-10-22 12:54:47 lokal tid, där den lokala tidszonen är GMT+01:00.

Skriptexempel

Exempel	Resultat
`localtime ()`	Returnerar lokal tid 2014-10-22 12:54:47.
`localtime ('London')`	Returnerar lokal tid i London, 2014-10-22 11:54:47.
`localtime ('GMT+02:00')`	Returnerar lokal tid i tidszonen GMT+02:00, 2014-10-22 13:54:47.
`localtime ('Paris','-1')`	Returnerar lokal tid i Paris utan hänsyn till sommartid, 2014-10-22 11:54:47.

lunarweekend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden i den sjudagarsperiod som innehåller **date**. Sjudagarsperioder i Qlik Sense definieras genom att 1 januari räknas som den första dagen på veckan.

Syntax:

```
LunarweekEnd(date[, period_no[, first_week_day]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
date	Det datum som ska evalueras.
period_no	**period_no** är ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den sjudagarsperiod som innehåller **date**. Negativa värden i **period_no** anger föregående sjudagarsperioder och positiva värden anger efterföljande sjudagarsperioder.
first_week_day	En förflyttning av startpunkten som kan vara större eller mindre än noll. Detta flyttar början på året med det angivna antalet dagar och/eller delar av en dag.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`lunarweekend('12/01/2013')`	Returnerar 14/01/2013 23:59:59.
`lunarweekend('12/01/2013', -1)`	Returnerar 7/01/2013 23:59:59.
`lunarweekend('12/01/2013', 0, 1)`	Returnerar 15/01/2013 23:59:59.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet hittas den sista dagen i sjudagarsperioden för varje faktureringsdatum i tabellen, där date flyttas en vecka genom att ange period_no som 1.

```
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 *,
LunarWeekEnd(InvDate, 1) AS LWkEnd
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för lunarweekend()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	LWkEnd
28/03/2012	07/04/2012
10/12/2012	22/12/2012
5/2/2013	18/02/2013
31/3/2013	08/04/2013
19/5/2013	27/05/2013
15/9/2013	23/09/2013
11/12/2013	23/12/2013
2/3/2014	11/03/2014
14/5/2014	27/05/2014
13/6/2014	24/06/2014
7/7/2014	15/07/2014
4/8/2014	12/08/2014

lunarweekname

Denna funktion returnerar ett visningsvärde som visar året och sjudagarsperiodsnumret som motsvarar en tidsmarkör för den första millisekunden på den första dagen i sjudagarsperioden som innehåller **date**. Sjudagarsperioder i Qlik Sense definieras genom att 1 januari räknas som den första dagen på veckan.

Syntax:

```
LunarWeekName(date [, period_no[, first_week_day]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
date	Det datum som ska evalueras.
period_no	**period_no** är ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den sjudagarsperiod som innehåller **date**. Negativa värden i **period_no** anger föregående sjudagarsperioder och positiva värden anger efterföljande sjudagarsperioder.
first_week_day	En förflyttning av startpunkten som kan vara större eller mindre än noll. Detta flyttar början på året med det angivna antalet dagar och/eller delar av en dag.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`lunarweekname('12/01/2013')`	Returnerar 2006/02.
`lunarweekname('12/01/2013', -1)`	Returnerar 2006/01.
`lunarweekname('12/01/2013', 0, 1)`	Returnerar 2006/02.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

För varje faktureringsdatum tabellen i det här exemplet skapas sjudagarsperiodens namn utifrån året där veckan finns och dess associerade sjudagarsperiodsnummer, med en veckas förskjutning genom att ange period_no som 1.

```
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 *,
LunarWeekName(InvDate, 1) AS LWkName
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för lunarweekname()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	LWkName
28/03/2012	2012/14
10/12/2012	2012/51
5/2/2013	2013/07
31/3/2013	2013/14
19/5/2013	2013/21
15/9/2013	2013/38
11/12/2013	2013/51
2/3/2014	2014/10
14/5/2014	2014/21
13/6/2014	2014/25
7/7/2014	2014/28
4/8/2014	2014/32

lunarweekstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden i den sjudagarsperiod som innehåller **date**. Sjudagarsperioder i Qlik Sense definieras genom att 1 januari räknas som den första dagen på veckan.

Syntax:

```
LunarweekStart(date[, period_no[, first_week_day]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
date	Det datum som ska evalueras.
period_no	**period_no** är ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den sjudagarsperiod som innehåller **date**. Negativa värden i **period_no** anger föregående sjudagarsperioder och positiva värden anger efterföljande sjudagarsperioder.
first_week_day	En förflyttning av startpunkten som kan vara större eller mindre än noll. Detta flyttar början på året med det angivna antalet dagar och/eller delar av en dag.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`lunarweekstart ('12/01/2013')`	Returnerar 08/01/2013.
`lunarweekstart ('12/01/2013', -1)`	Returnerar 01/01/2013.
`lunarweekstart ('12/01/2013', 0, 1 )`	Returnerar 09/01/2013. Eftersom startpunkten som anges genom att ange first_week_day till 1 innebär att början av året ändras till 02/01/2013.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet hittas den första dagen i sjudagarsperioden för varje faktureringsdatum i tabellen, där date flyttas en vecka genom att ange period_no som 1.

```
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 *,
LunarWeekStart(InvDate, 1) AS LWkStart
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för lunarweekstart()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	LWkStart
28/03/2012	01/04/2012
10/12/2012	16/12/2012
5/2/2013	12/02/2013
31/3/2013	02/04/2013
19/5/2013	21/05/2013
15/9/2013	17/09/2013
11/12/2013	17/12/2013
2/3/2014	05/03/2014
14/5/2014	21/05/2014
13/6/2014	18/06/2014
7/7/2014	09/07/2014
4/8/2014	06/08/2014

makedate

Denna funktion returnerar ett datum beräknat utifrån året **YYYY**, månaden **MM** och dagen **DD**.

Syntax:

```
MakeDate(YYYY [ , MM [ , DD ] ])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
YYYY	Året i form av ett heltal.
MM	Månaden i form av ett heltal. Om inget månadsnummer anges, antas 1 (januari).
DD	Dagen i form av ett heltal. Om inget dagsnummer anges, antas 1 (den första).

Exempel: Diagramuttryck

Exempel på diagramuttryck

Exempel	Resultat
`makedate(2012)`	returnerar 2012-01-01
`makedate(12)`	returnerar 0012-01-01
`makedate(2012,12)`	returnerar 2012-12-01
`makedate(2012,2,14)`	returnerar 2012-02-14

Exempel: Laddningsskript
makedate kan användas i laddningsskript för att kombinera datumdata från olika fält i ett nytt datumfält. I exemplet nedan kombineras year-, month- och day-data från fälten transaction_year, transaction_month och transaction_day i ett nytt fält med namnet Transaction Date.

Gå till **Skriptredigeraren** och skapa ett nytt delavsnitt. Lägg sedan till exempelskriptet och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i din app för att se resultatet.

Laddningsskript

```
SET DateFormat='DD/MM/YYYY';
SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT';
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';

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, discount, customer_id, size, color_code
```

```
3750, 2018, 08, 30, 12423.56, 23, 0,2038593, L, Red
3751, 2018, 09, 07, 5356.31, 6, 0.1, 203521, m, orange
3752, 2018, 09, 16, 15.75, 1, 0.22, 5646471, S, blue
3753, 2018, 09, 22, 1251, 7, 0, 3036491, l, Black
3754, 2018, 09, 22, 21484.21, 1356, 75, 049681, xs, Red
3756, 2018, 09, 22, -59.18, 2, 0.333333333333333, 2038593, M, Blue
3757, 2018, 09, 23, 3177.4, 21, .14, 203521, XL, Black
];
```

Resultat

Qlik Sense table showing results of the makedate function being used in the load script.

transaction_id	Transaction Date
3750	30/08/2018
3751	07/09/2018
3752	16/09/2018
3753	22/09/2018
3754	22/09/2018
3756	22/09/2018
3757	23/09/2018

maketime

Denna funktion returnerar en tid beräknat utifrån timmar **hh**, minuter **mm** och sekunder **ss**.

Syntax:

```
MakeTime(hh [ , mm [ , ss ] ])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
hh	Timmen i form av ett heltal.
mm	Minuten i form av ett heltal. Om ingen minut angivits, antas 00.
ss	Sekunden i form av ett heltal. Om ingen sekund angivits, antas 00.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`maketime( 22 )`	returnerar 22:00:00
`maketime( 22, 17 )`	returnerar 22:17:00
`maketime( 22, 17, 52 )`	returnerar 22:17:52

makeweekdate

Denna funktion returnerar ett datum beräknat utifrån året **YYYY**, veckan **WW** och veckodagen **D**.

Syntax:

```
MakeWeekDate(YYYY [ , WW [ , D ] ])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
YYYY	Året i form av ett heltal.
WW	Veckan i form av ett heltal.
D	Veckodagen i form av ett heltal. Om inget veckodagsnummer anges, antas 0 (måndag).

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`makeweekdate(2014,6,6)`	returnerar 2014-02-09
`makeweekdate(2014,6,1)`	returnerar 2014-02-04
`makeweekdate(2014,6)`	returnerar 2014-02-03 (veckodag 0 antas)

minute

Denna funktion returnerar ett heltal som motsvarar minuten om decimaldelen av **expression** tolkas som tid enligt standardtolkningen av tal.

Syntax:

```
minute(expression)
```

Returnerad datatyp: heltal

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`minute ( '09:14:36' )`	returnerar 14
`minute ( '0.5555' )`	returnerar 19 (eftersom 0.5555 = 13:19:55)

month

Denna funktion returnerar ett dualt värde: månadsnamnet som det är definierat i miljövariabeln **MonthNames** och ett heltal mellan 1 och 12. Månaden beräknas utifrån datumtolkningen av uttrycket, enligt standardtalformatet.

Funktionen returnerar månadens namn i formatet för systemvariabeln `MonthName` för ett visst datum. Den används vanligtvis för att skapa ett dagfält som en dimension i en Master Calendar.

Syntax:

```
month(expression)
```

Returnerad datatyp: heltal

Exempel på funktioner

Exempel	Resultat
`month( '2012-10-12' )`	returnerar Oct
`month( '35648' )`	returnerar Aug, eftersom 35648 = 1997-08-06

monthend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör med den sista millisekunden av den sista dagen i den månad som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

Syntax:

```
MonthEnd(date[, period_no])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
date	Det datum som ska evalueras.
period_no	**period_no** är ett heltal som om det är 0 eller utelämnas anger månaden som innehåller **date**. Negativa värden i **period_no** anger föregående månader och positiva värden anger efterföljande månader.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`monthend('19/02/2012')`	Returnerar 29/02/2012 23:59:59.
`monthend('19/02/2001', -1)`	Returnerar 31/01/2001 23:59:59.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet hittas den sista dagen i månaden för varje faktureringsdatum i tabellen, där basdatumet flyttas fyra månader genom att du anger *period_no* som 4.

```
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 *,
MonthEnd(InvDate, 4) AS MthEnd
```

```
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för monthend()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	MthEnd
28/03/2012	31/07/2012
10/12/2012	30/04/2013
5/2/2013	30/06/2013
31/3/2013	31/07/2013
19/5/2013	30/09/2013
15/9/2013	31/01//2014
11/12/2013	30/04//2014
2/3/2014	31/07//2014
14/5/2014	30/09/2014
13/6/2014	31/10/2014
7/7/2014	30/11/2014
4/8/2014	31/12/2014

monthname

Denna funktion returnerar ett visningsvärde med månaden (formaterat enligt skriptvariabeln **MonthNames**) och året med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden av månadens första dag.

Syntax:

```
MonthName(date[, period_no])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
date	Det datum som ska evalueras.
period_no	**period_no** är ett heltal som om det är 0 eller utelämnas anger månaden som innehåller **date**. Negativa värden i **period_no** anger föregående månader och positiva värden anger efterföljande månader.

Exempel: Diagramuttryck
I det här exemplet används datumformatet **DD/MM/YYYY** som anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav. I **SET Monthnames**-satsen anges Jan;Feb;Mar och så vidare.

Exempel på diagramuttryck

Exempel	Resultat
`monthname('19/10/2013')`	Returnerar Oct 2013
`monthname('19/10/2013', -1)`	Returnerar Sep 2013

Exempel: Laddningsskript
I det här exemplet skapas månadens namn för varje faktureringsdatum i tabellen av månadens namn framflyttat fyra månader från base_date samt av året.

Gå till **Skriptredigeraren** och skapa ett nytt delavsnitt. Lägg sedan till exempelskriptet och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i din app för att se resultatet.

Laddningsskript

```
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 *,
MonthName(InvDate, 4) AS MthName
Resident TempTable;
Drop table TempTable;
```

Resultat

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med värdet som returneras av monthname()-funktionen.

InvDate	MthName
28/03/2012	Jul 2012
10/12/2012	Apr 2013
5/2/2013	Jun 2013
31/3/2013	Jul 2013
19/5/2013	Sep 2013
15/9/2013	Jan 2014
11/12/2013	Apr 2014
2/3/2014	Jul 2014
14/5/2014	Sep 2014
13/6/2014	Oct 2014
7/7/2014	Nov 2014
4/8/2014	Dec 2014

Exempel: Laddningsskript
I det här exemplet skapas ett Returnable_Until-värde för varje transaction_date i tabellen. Returnable_Until-värdet beräknas genom att månaden i transaction_date ändras till en månad senare.

Gå till **Skriptredigeraren** och skapa ett nytt delavsnitt. Lägg sedan till exempelskriptet och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i din app för att se resultatet.

Laddningsskript

```
SET DateFormat='YYYYMMDD';
SET TimestampFormat='YYYYMMDD h:mm:ss[.fff] TT';
SET FirstMonthOfYear=1;
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';
SET
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';

Transactions:
Load
*,
MonthName(Date#(transaction_date,'YYYYMMDD'), 1) as Returnable_Until,
```

```
;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 20180830, 12423.56, 23, 0,2038593, L, Red
3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue
3753, 20180922, 1251, 7, 0, 3036491, l, Black
3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red
3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue
3757, 20180923, 3177.4, 21, .14, 203521, XL, Black
];
```

Resultat

Qlik Sense table showing results of the monthname function being used in the load script.

transaction_id	transaction_date	Returnable_Until
3750	20180830	Sep 2018
3751	20180907	Oct 2018
3752	20180916	Oct 2018
3753	20180922	Oct 2018
3754	20180922	Oct 2018
3756	20180922	Oct 2018
3757	20180923	Oct 2018

monthsend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden i månaden, tvåmånadersperioden, kvartalet, tredjedelens år eller halvåret som innehåller ett basdatum. Det går även att hitta tidsmarkören för en föregående eller senare tidsperiod.

Syntax:

```
MonthsEnd(n_months, date[, period_no [, first_month_of_year]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
n_months	Antalet månader som definierar perioden. 1 (motsvaras av inmonth()-funktionen), 2 (tvåmånadersperiod), 3 (motsvaras av inquarter()-funktionen), 4 (tredjedels år), eller 6 (halvår).
date	Det datum som ska evalueras.
period_no	Periodens startpunkt kan flyttas med **period_no**, ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den period som innehåller **base_date**. Negativa värden i **period_no** anger föregående perioder och positiva värden anger efterföljande perioder.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`monthsend(4, '19/07/2013')`	Returnerar 31/08/2013.
`monthsend(4, '19/10/2013', -1)`	Returnerar 31/08/2013.
`monthsend(4, '19/10/2013', 0, 2)`	Returnerar 31/01/2014. Eftersom årets start blir månad 2.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet hittar vi slutet på den sista dagen i en tvåmånadersperiod för varje faktureringsdatum, framflyttad med en tvåmånadersperiod.

```
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 *,
MonthsEnd(2, InvDate, 1) AS BiMthsEnd
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för MonthsEnd()-funktionen.

Resultattabell

InvDate	BiMthsEnd
28/03/2012	30/06/2012
10/12/2012	28/02/2013
5/2/2013	30/04/2013
31/3/2013	30/04/2013
19/5/2013	31/08/2013
15/9/2013	31/12/2013
11/12/2013	28/02/2014
2/3/2014	30/06/2014
14/5/2014	31/08/2014
13/6/2014	31/08/2014
7/7/2014	31/10/2014
4/8/2014	31/10/2014

monthsname

Denna funktion returnerar ett visningsvärde som representerar månadsintervallet i perioden (formaterat enligt skriptvariabeln **MonthNames**) liksom året. Det underliggande numeriska värdet motsvarar en tidsmarkör för den första millisekunden i månaden, tvåmånadersperioden, kvartalet, tredjedelen av året eller halvåret som innehåller ett basdatum.

Syntax:

```
MonthsName(n_months, date[, period_no[, first_month_of_year]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
n_months	Antalet månader som definierar perioden. 1 (motsvaras av inmonth()-funktionen), 2 (tvåmånadersperiod), 3 (motsvaras av inquarter()-funktionen), 4 (tredjedels år), eller 6 (halvår).
date	Det datum som ska evalueras.
period_no	Periodens startpunkt kan flyttas med **period_no**, ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den period som innehåller **base_date**. Negativa värden i **period_no** anger föregående perioder och positiva värden anger efterföljande perioder.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`monthsname(4, '19/10/2013')`	Returnerar Sep-Dec 2013. Eftersom **SET Monthnames**-satsen i det här och de andra exemplen har angetts som Jan;Feb;Mar och så vidare.
`monthsname(4, '19/10/2013', -1)`	Returnerar May-Aug 2013.
`monthsname(4, '19/10/2013', 0, 2)`	Returnerar Oct-Jan 2014. Eftersom året har angetts att börja med månad 2 och därför slutar fyramånadersperioden med den första månaden av nästföljande år.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet skapas månadens namn för varje faktureringsdatum i tabellen av månadsintervallet i tvåmånadersperioden samt året. Intervallet flyttas med 4x2 månader genom att du anger period_no som 4.

```
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 *,
MonthsName(2, InvDate, 4) AS MthsName
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för monthsname()-funktionen.

Resultattabell

InvDate	MthsName
28/03/2012	Nov-Dec 2012
10/12/2012	Jul-Aug 2013
5/2/2013	Sep-Oct 2013
31/3/2013	Nov-Dec2013
19/5/2013	Jan-Feb 2014
15/9/2013	May-Jun 2014
11/12/2013	Jul-Aug 2014
2/3/2014	Nov-Dec 2014
14/5/2014	Jan-Feb 2015
13/6/2014	Jan-Feb 2015
7/7/2014	Mar-Apr 2015
4/8/2014	Mar-Apr 2015

monthsstart

Denna funktion returnerar ett värde som motsvarar tidsmarkören för den första millisekunden i månaden, tvåmånadersperioden, kvartalet, tredjedelens år eller halvåret som innehåller ett basdatum. Det går även att hitta tidsmarkören för en föregående eller senare tidsperiod.

Syntax:

```
MonthsStart(n_months, date[, period_no [, first_month_of_year]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
n_months	Antalet månader som definierar perioden. 1 (motsvaras av inmonth()-funktionen), 2 (tvåmånadersperiod), 3 (motsvaras av inquarter()-funktionen), 4 (tredjedels år), eller 6 (halvår).
date	Det datum som ska evalueras.
period_no	Periodens startpunkt kan flyttas med **period_no**, ett heltal, eller ett uttryck som resulterar i ett heltal, där värdet 0 anger den period som innehåller **base_date**. Negativa värden i **period_no** anger föregående perioder och positiva värden anger efterföljande perioder.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`monthsstart(4, '19/10/2013')`	Returnerar 1/09/2013.
`monthsstart(4, '19/10/2013, -1)`	Returnerar 01/05/2013.
`monthsstart(4, '19/10/2013', 0, 2 )`	Returnerar 01/10/2013. Eftersom årets start blir månad 2.

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet hittar vi den första dagen i en tvåmånadersperiod för varje faktureringsdatum, framflyttad med en tvåmånadersperiod.

```
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 *,
MonthsStart(2, InvDate, 1) AS BiMthsStart
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för MonthsStart()-funktionen.

Resultattabell

InvDate	BiMthsStart
28/03/2012	01/05/2012
10/12/2012	01/01/2013
5/2/2013	01/03/2013
31/3/2013	01/05/2013
19/5/2013	01/07/2013
15/9/2013	01/11/2013
11/12/2013	01/01/2014
2/3/2014	01/05/2014
14/5/2014	01/07/2014
13/6/2014	01/07/2014
7/7/2014	01/09/2014
4/8/2014	01/09/2014

monthstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden av den första dagen i den månad som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

Syntax:

```
MonthStart(date[, period_no])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
date	Det datum som ska evalueras.
period_no	**period_no** är ett heltal som om det är 0 eller utelämnas anger månaden som innehåller **date**. Negativa värden i **period_no** anger föregående månader och positiva värden anger efterföljande månader.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`monthstart('19/10/2001')`	Returnerar 01/10/2001.
`monthstart('19/10/2001', -1)`	Returnerar 01/09/2001.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet hittas den första dagen i månaden för varje faktureringsdatum i tabellen, där base_date flyttas fyra månader genom att du anger period_no som 4.

```
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 *,
MonthStart(InvDate, 4) AS MthStart
```

```
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för monthstart()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	MthStart
28/03/2012	01/07/2012
10/12/2012	01/04/2013
5/2/2013	01/06/2013
31/3/2013	01/07/2013
19/5/2013	01/09/2013
15/9/2013	01/01/2014
11/12/2013	01/04/2014
2/3/2014	01/07/2014
14/5/2014	01/09/2014
13/6/2014	01/10/2014
7/7/2014	01/11/2014
4/8/2014	01/12/2014

networkdays

Funktionen **networkdays** returnerar antalet arbetsdagar (måndag-fredag) mellan och inklusive **start_date** och **end_date** med hänsyn till eventuella **holiday**.

Syntax:

```
networkdays (start_date, end_date [, holiday])
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
start_date	Startdatum som ska utvärderas.
end_date	Slutdatumn som ska utvärderas.

Argument	Beskrivning
holiday	Ledighetsperioder som ska undantas från arbetsdagar. En ledighetsperiod anges som ett startdatum och ett slutdatum, avgränsade med kommatecken. **Exempel:** `'25/12/2013', '26/12/2013'` Du kan ange fler än en ledighetsperiod, avgränsade med kommatecken. **Exempel:** `'25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014'`

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`networkdays ('19/12/2013', '07/01/2014')`	Returnerar 14. Detta exempel tar inte hänsyn till semester.
`networkdays ('19/12/2013', '07/01/2014', '25/12/2013', '26/12/2013')`	Returnerar 12. Detta exempel tar hänsyn till semester 25/12/2013 till 26/12/2013.
`networkdays ('19/12/2013', '07/01/2014', '25/12/2013', '26/12/2013', '31/12/2013', '01/01/2014')`	Returnerar 10. I detta exempel beaktas två ledighetsperioder.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

```
PayTable:
LOAD recno() as InvID, * INLINE [
InvRec|InvPaid
28/03/2012|28/04/2012
10/12/2012|01/01/2013
5/2/2013|5/3/2013
31/3/2013|01/5/2013
19/5/2013|12/6/2013
15/9/2013|6/10/2013
11/12/2013|12/01/2014
2/3/2014|2/4/2014
14/5/2014|14/6/2014
13/6/2014|14/7/2014
7/7/2014|14/8/2014
4/8/2014|4/9/2014
] (delimiter is '|');
NrDays:
Load *,
NetWorkDays(InvRec,InvPaid) As PaidDays
Resident PayTable;
```

```
Drop table PayTable;
```

Den resulterande tabellen visar de returnerade värdena för NetworkDays för varje post i tabellen.

Resultattabell

InvID	InvRec	InvPaid	PaidDays
1	28/03/2012	28/04/2012	23
2	10/12/2012	01/01/2013	17
3	5/2/2013	5/3/2013	21
4	31/3/2013	01/5/2013	23
5	19/5/2013	12/6/2013	18
6	15/9/2013	6/10/2013	15
7	11/12/2013	12/01/2014	23
8	2/3/2014	2/4/2014	23
9	14/5/2014	14/6/2014	23
10	13/6/2014	14/7/2014	22
11	7/7/2014	14/8/2014	29
12	4/8/2014	4/9/2014	24

now

Denna funktion returnerar en tidsmarkör för aktuell tid från systemklockan. Standardvärdet är 1.

Syntax:

```
now([ timer_mode])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
timer_mode	Kan ha följande värden: 0 (tiden för senast avslutade dataladdning) 1 (tiden för funktionsanrop) 2 (tiden då appen öppnades) *Om du använder funktionen i ett dataladdningsskript, resulterar **timer_mode=0** i tiden för senast slutförda dataladdning, medan **timer_mode=1** visar tiden för funktionsanropet i den aktuella dataladdningen.*

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`now( 0)`	Returnerar den tidpunkt då den sista dataläsningen slutfördes.
`now( 1)`	När den används i ett visualiseringsuttryck, returnerar detta tidpunkten för funktionsanropet. Vid användning i ett dataladdningsskript, returnerar detta tidpunkten för funktionsanropet i den aktuella dataladdningen.
`now( 2)`	Returnerar tiden då appen öppnades.

quarterend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden i det kvartal som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

Syntax:

```
QuarterEnd(date[, period_no[, first_month_of_year]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
date	Det datum som ska evalueras.
period_no	**period_no** är ett heltal, där värdet 0 anger det kvartal som innehåller **date**. Negativa värden i **period_no** anger föregående kvartal och positiva värden anger efterföljande kvartal.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`quarterend('29/10/2005')`	Returnerar 31/12/2005 23:59:59.
`quarterend('29/10/2005', -1)`	Returnerar 30/09/2005 23:59:59.
`quarterend('29/10/2005', 0, 3)`	Returnerar 30/11/2005 23:59:59.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet hittas den sista dagen i kvartalet för varje faktureringsdatum i tabellen, där den första månaden på året har angetts som månad 3.

```
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 *,
QuarterEnd(InvDate, 0, 3) AS QtrEnd
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för quarterend()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	QtrEnd
28/03/2012	31/05/2012
10/12/2012	28/02/2013
5/2/2013	28/02/2013
31/3/2013	31/05/2013
19/5/2013	31/05/2013
15/9/2013	30/11/2013
11/12/2013	28/02/2014
2/3/2014	31/05/2014
14/5/2014	31/05/2014
13/6/2014	31/08/2014
7/7/2014	31/08/2014
4/8/2014	31/08/2014

quartername

Denna funktion returnerar ett visningsvärde med kvartalets månader (formaterat enligt skriptvariabeln **MonthNames**) och år med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden av kvartalets första dag.

Syntax:

```
QuarterName(date[, period_no[, first_month_of_year]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
date	Det datum som ska evalueras.
period_no	**period_no** är ett heltal, där värdet 0 anger det kvartal som innehåller **date**. Negativa värden i **period_no** anger föregående kvartal och positiva värden anger efterföljande kvartal.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`quartername('29/10/2013')`	Returnerar Oct-Dec 2013.
`quartername('29/10/2013', -1)`	Returnerar Jul-Sep 2013.
`quartername('29/10/2013', 0, 3)`	Returnerar Sep-Nov 2013.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet skapas kvartalsnamnet baserat på det kvartal som innehåller *InvID* för varje faktureringsdatum i tabellen. Den första månaden på året angetts som månad 4.

```
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 *,
QuarterName(InvDate, 0, 4) AS QtrName
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för quartername()-funktionen.

Resultattabell

InvDate	QtrName
28/03/2012	Jan-Mar 2011
10/12/2012	Oct-Dec 2012
5/2/2013	Jan-Mar 2012
31/3/2013	Jan-Mar 2012
19/5/2013	Apr-Jun 2013
15/9/2013	Jul-Sep 2013
11/12/2013	Oct-Dec 2013
2/3/2014	Jan-Mar 2013
14/5/2014	Apr-Jun 2014
13/6/2014	Apr-Jun 2014
7/7/2014	Jul-Sep 2014
4/8/2014	Jul-Sep 2014

quarterstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden i det kvartal som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

Syntax:

```
QuarterStart(date[, period_no[, first_month_of_year]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
date	Det datum som ska evalueras.

Argument	Beskrivning
period_no	**period_no** är ett heltal, där värdet 0 anger det kvartal som innehåller **date**. Negativa värden i **period_no** anger föregående kvartal och positiva värden anger efterföljande kvartal.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`quarterstart('29/10/2005')`	Returnerar 01/10/2005.
`quarterstart('29/10/2005', -1 )`	Returnerar 01/07/2005.
`quarterstart('29/10/2005', 0, 3)`	Returnerar 01/09/2005.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet hittas den första dagen i kvartalet för varje faktureringsdatum i tabellen, där den första månaden på året har angetts som månad 3.

```
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 *,
QuarterStart(InvDate, 0, 3) AS QtrStart
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för quarterstart()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	QtrStart
28/03/2012	01/03/2012
10/12/2012	01/12/2012
5/2/2013	01/12/2012
31/3/2013	01/03/2013
19/5/2013	01/03/2013
15/9/2013	01/09/2013
11/12/2013	01/12/2013
2/3/2014	01/03/2014
14/5/2014	01/03/2014
13/6/2014	01/06/2014
7/7/2014	01/06/2014
4/8/2014	01/06/2014

second

Denna funktion returnerar ett heltal som motsvarar sekunden om decimaldelen av **expression** tolkas som en tidpunkt enligt standardtolkningen av tal.

Syntax:

```
second (expression)
```

Returnerad datatyp: heltal

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`second( '09:14:36' )`	returnerar 36
`second( '0.5555' )`	returnerar 55 (eftersom 0.5555 = 13:19:55)

setdateyear

Den här funktionen tar som indata en **timestamp** och ett **year** och uppdaterar **timestamp** med det **year** som har angetts i indata.

Syntax:

```
setdateyear (timestamp, year)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
timestamp	En tidsangivelse i standard-Qlik Sense-format (ofta endast ett datum).
year	Ett fyrsiffrigt årtal.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`setdateyear ('29/10/2005', 2013)`	Returnerar29/10/2013
`setdateyear ('29/10/2005 04:26:14', 2013)`	Returnerar '29/10/2013 04:26:14' För att se tidsdelen av tidsmarkören i en visualisering måste du ange talformatet som Datum och välja ett värde för Formatering som visar tidsvärden.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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
];
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn där året har angetts som 2013.

Resultattabell

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
2/3/2014	2/3/2013
14/5/2014	14/5/2013
13/6/2014	13/6/2013
7/7/2014	7/7/2013
4/8/2014	4/8/2013

setdateyearmonth

Den här funktionen tar som indata en **timestamp**, en **month** och ett **year** och uppdaterar **timestamp** med det **year** och den **month** som har angetts i indata. .

Syntax:

```
SetDateYearMonth (timestamp, year, month)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
timestamp	En tidsangivelse i standard-Qlik Sense-format (ofta endast ett datum).
year	Ett fyrsiffrigt årtal.
month	En en- eller tvåsiffrig månad.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`setdateyearmonth ('29/10/2005', 2013, 3)`	Returnerar29/03/2013
`setdateyearmonth ('29/10/2005 04:26:14', 2013, 3)`	Returnerar '29/03/2013 04:26:14' För att se tidsdelen av tidsmarkören i en visualisering måste du ange talformatet som Datum och välja ett värde för Formatering som visar tidsvärden.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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
];
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn där året har angetts som 2013.

Resultattabell

testdates	NewYearMonth
1/11/2012	1/3/2013
10/12/2012	10/3/2013
2/1/2012	2/3/2013
19/5/2013	19/3/2013
15/9/2013	15/3/2013

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

Denna funktion returnerar namnet på den aktuella tidszonen så som den definierats i Windows.

Syntax:

```
TimeZone( )
```

Returnerad datatyp: sträng

Exempel:

```
timezone( )
```

today

Denna funktion returnerar aktuellt datum från systemklockan.

Syntax:

```
today([ timer_mode])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
timer_mode	Kan ha följande värden: 0 (dagen för senast avslutade dataladdning) 1 (dagen för funktionsanrop) 2 (dagen då appen öppnades) *Om du använder funktionen i ett dataladdningsskript, resulterar **timer_mode=0** i dagen för senast slutförda dataladdning, medan **timer_mode=1** visar dagen för den aktuella dataladdningen.*

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`Today( 0)`	Returnerar dagen i den senaste slutförda dataladdningen.
`Today( 1)`	När den används i ett visualiseringsuttryck, returnerar detta dagen för funktionsanropet. Vid användning i ett dataladdningsskript, returneras den dag då den aktuella dataladdningen började.
`Today( 2)`	Returnerar dagen då appen öppnades.

UTC

Returnerar aktuell Coordinated Universal Time.

Syntax:

```
UTC( )
```

Returnerad datatyp: dual

Exempel:

```
utc( )
```

week

Denna funktion returnerar ett heltal som motsvarar veckonumret enligt ISO 8601. Veckonumret beräknas utifrån datumtolkningen av uttrycket, enligt standardtalformatet.

Syntax:

```
week(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

Returnerad datatyp: heltal

Argument

Argument	Beskrivning
timestamp	Datum som ska utvärderas som en tidsmarkör eller ett uttryck som resulterar i en tidsmarkör, och som ska konverteras, till exempel '2012-10-12'.

Argument	Beskrivning
first_week_day	Om du inte specificerar **first_week_day**, kommer värdet för variabeln **FirstWeekDay** att användas som den första dagen i veckan. Om du vill använda en annan dag som den första dagen i veckan ska du ställa in **first_week_day** till: • 0 för måndag • 1 för tisdag • 2 för onsdag • 3 för torsdag • 4 för fredag • 5 för lördag • 6 för söndag Den siffra som returneras av funktionen kommer nu att använda den första dag i veckan som du har ställt in med **first_week_day**.
broken_weeks	Om du inte anger **broken_weeks** används värdet för variabeln **BrokenWeeks** till att definiera om veckor är brutna eller inte. Som standard använder Qlik Sense-funktionerna obrutna veckor. Det betyder att: • Vissa år börjar vecka 1 i december, och andra år fortsätter vecka 52 eller 53 in i januari. • Vecka 1 har alltid minst 4 dagar i januari. Alternativet är att använda brutna veckor. • Vecka 52 eller 53 fortsätter inte in i januari. • Vecka 1 börjar den 1 januari och är i de flesta fall inte en hel vecka. Följande värden kan användas: • 0 (= använd obrutna veckor) • 1 (= använd brutna veckor)

Argument	Beskrivning
reference_day	Om du inte specificerar **reference_day** används värdet för variabeln **ReferenceDay** till att definiera vilken dag i januari som ska ställas in som referensdag för att definiera vecka 1. Qlik Sense-funktioner använder som standard 4 som referensdag. Det betyder att vecka 1 måste innehålla 4 januari, eller med andra ord, vecka 1 måste alltid bestå av minst 4 dagar i januari. Följande värden kan användas för att ställa in en annan referensdag. • 1 (= 1 januari) • 2 (= 2 januari) • 3 (= 3 januari) • 4 (= 4 januari) • 5 (= 5 januari) • 6 (= 6 januari) • 7 (= 7 januari)

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`week( '2012-10-12' )`	returnerar 41.
`week( '35648' )`	returnerar 32, eftersom 35648 = 1997-08-06
`week('2012-10-12', 0, 1)`	returnerar 42

weekday

Denna funktion returnerar ett dualt värde med:

- Ett namn på en dag som definierat i miljövariabeln **DayNames**.
- Ett heltal mellan 0 och 6 som motsvarar den nominella veckodagen (0-6).

Syntax:

```
weekday(date [,first_week_day=0])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
date	Det datum som ska evalueras.

Argument	Beskrivning
first_week_day	Om du inte specificerar **first_week_day** kommer värdet för variabeln **FirstWeekDay** att användas som den första dagen i veckan. Om du vill använda en annan dag som den första dagen i veckan ska du ställa in **first_week_day** till: • 0 för måndag • 1 för tisdag • 2 för onsdag • 3 för torsdag • 4 för fredag • 5 för lördag • 6 för söndag Den siffra som returneras av funktionen kommer nu att använda den första dag i veckan som du har ställt in med **first_week_day** som bas (0). *FirstWeekDay (page 150)*

Exempel: Diagramuttryck
Om inget annat sägs, är **FirstWeekDay** inställd till 0 i dessa exempel.

Skriptexempel

Exempel	Resultat
`weekday( '1971-10-12' )`	returnerar ' Tue' och 1
`weekday( '1971-10-12' , 6)`	returnerar 'Tue' och 2 I det här exemplet har vi använt söndag (6) som den första dagen i veckan.

`SET FirstWeekDay = 6;`
 	
`...`
`weekday( '1971-10-12')`
	returnerar 'Tue' och 2

Exempel: Laddningsskript

Laddningsskript

weekday kan användas i ett laddningsskript för att returnera en sträng och ett tal som representerar en veckodag, även om *FirstWeekDay* och *ReferenceDay* redan är inställda i skriptet. Laddningsskriptet nedan innehåller specifika värden för *FirstWeekDay* och *ReferenceDay*, och använder sedan *weekday* för att returnera både strängar och tal som representerar veckodagar från data i kolumnen *transaction_date*.

I resultaten som visas innehåller kolumnen *Day* de returnerade strängarna. *Numeric value of Day* och *Numeric value of week starting from Sunday* innehåller de returnerade numeriska värdena. I laddningsskriptet multipliceras *weekday* med 1, som ett enkelt sätt att se till att en numerisk datatyp returneras.

Gå till **Skriptredigeraren** och skapa ett nytt delavsnitt. Lägg sedan till exempelskriptet och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i din app för att se resultatet.

```
SET DateFormat='DD/MM/YYYY';
SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT';
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';

Transactions:
Load
*,
WeekDay(transaction_date) as [Day],
1*WeekDay(transaction_date) as [Numeric value of Day]
1*WeekDay(transaction_date, 6) as [Numeric value of a week starting from Sunday],
;
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 20180830, 12423.56, 23, 0,2038593, L, Red
3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue
3753, 20180922, 1251, 7, 0, 3036491, l, Black
3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red
3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue
3757, 20180923, 3177.4, 21, .14, 203521, XL, Black
];
```

Resultat

Qlik Sense-tabell som visar resultat från hur weekday-funktionen används i laddningsskriptet.

transaction_id	transaction_ date	Dag	Numeriskt värde för dag	Numeriskt värde för en vecka som börjar på söndag
3750	20180830	tors	3	4
3751	20180907	tors	3	4
3752	20180916	lör	5	6
3753	20180922	fre	4	5
3754	20180922	fre	4	5
3756	20180922	fre	4	5
3757	20180923	lör	5	6

weekend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den sista millisekunden på den sista dagen (söndag) i kalenderveckan som innehåller **date**. Standardformatet för utdata är det **DateFormat** som är angivet i skriptet.

Syntax:

```
WeekEnd(date [, period_no[, first_week_day]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
date	Det datum som ska evalueras.
period_no	**shift** är ett heltal, där värdet 0 anger den vecka som innehåller **date**. Negativa värden i skift anger föregående veckor och positiva värden anger efterföljande veckor.
first_week_day	Anger dagen då veckan startar. Om utelämnat används värdet för variabeln **FirstWeekDay**. De valbara värdena för **first_week_day** är: • 0 för måndag • 1 för tisdag • 2 för onsdag • 3 för torsdag • 4 för fredag • 5 för lördag • 6 för söndag *FirstWeekDay (page 150)*

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Exempel	Resultat
`weekend('10/01/2013')`	Returnerar 12/01/2013 23:59:59.
`weekend('10/01/2013', -1)`	Returnerar 06/01/2013 23:59:59.
`weekend('10/01/2013', 0, 1)`	Returnerar 14/01/2013 23:59:59.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet hittas den sista dagen i veckan som följer på veckan för varje faktureringsdatum i tabellen.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
WeekEnd(InvDate, 1) AS WkEnd
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för weekend()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	WkEnd
28/03/2012	08/04/2012
10/12/2012	23/12/2012
5/2/2013	17/02/2013
31/3/2013	07/04/2013
19/5/2013	26/05/2013
15/9/2013	22/09/2013
11/12/2013	22/12/2013
2/3/2014	09/03/2014
14/5/2014	25/05/2014

13/6/2014	22/06/2014
7/7/2014	20/07/2014
4/8/2014	17/08/2014

weekname

Denna funktion returnerar ett värde som visar år och veckonummer med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden på den första dagen i veckan som innehåller **date**.

Syntax:

```
WeekName(date[, period_no[,first_week_day]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
date	Det datum som ska evalueras.
period_no	**shift** är ett heltal, där värdet 0 anger den vecka som innehåller **date**. Negativa värden i skift anger föregående veckor och positiva värden anger efterföljande veckor.
first_week_day	Anger dagen då veckan startar. Om utelämnat används värdet för variabeln **FirstWeekDay**. De valbara värdena för **first_week_day** är: • 0 för måndag • 1 för tisdag • 2 för onsdag • 3 för torsdag • 4 för fredag • 5 för lördag • 6 för söndag *FirstWeekDay (page 150)*

Exempel och resultat:

Exempel	Resultat
`weekname('12/01/2013')`	Returnerar 2013/02.
`weekname('12/01/2013', -1)`	Returnerar 2013/01.
`weekname('12/01/2013', 0, 1)`	Returnerar 2013/02.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

För varje faktureringsdatum i tabellen i det här exemplet skapas veckans namn utifrån året där veckan finns och dess associerade veckonummer, med en veckas förskjutning genom att ange period_no som 1.

```
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 *,
WeekName(InvDate, 1) AS WkName
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för weekname()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	WkName
28/03/2012	2012/14
10/12/2012	2012/51
5/2/2013	2013/07
31/3/2013	2013/14

InvDate	WkName
19/5/2013	2013/21
15/9/2013	2013/38
11/12/2013	2013/51
2/3/2014	2014/10
14/5/2014	2014/21
13/6/2014	2014/25
7/7/2014	2014/29
4/8/2014	2014/33

weekstart

Denna funktion returnerar ett värde som motsvarar en tidsmarkör för den första millisekunden på den första dagen (måndag) i den kalendervecka som innehåller **date**. Det förvalda utdataformatet är det **DateFormat** som har definierats i skriptet.

Syntax:

```
WeekStart(date [, period_no[, first_week_day]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
date	Det datum som ska evalueras.
period_no	**shift** är ett heltal, där värdet 0 anger den vecka som innehåller **date**. Negativa värden i skift anger föregående veckor och positiva värden anger efterföljande veckor.
first_week_day	Anger dagen då veckan startar. Om utelämnat används värdet för variabeln **FirstWeekDay**. De valbara värdena för **first_week_day** är: • 0 för måndag • 1 för tisdag • 2 för onsdag • 3 för torsdag • 4 för fredag • 5 för lördag • 6 för söndag *FirstWeekDay (page 150)*

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`weekstart('12/01/2013')`	Returnerar 07/01/2013.
`weekstart('12/01/2013', -1 )`	Returnerar 31/11/2012.
`weekstart('12/01/2013', 0, 1)`	Returnerar 08/01/2013.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet hittas den första dagen i veckan som följer på veckan för varje faktureringsdatum i tabellen.

```
TempTable:
LOAD RecNo() as InvID, * Inline [
InvDate
28/03/2012
10/12/2012
5/2/2013
31/3/2013
19/5/2013
15/9/2013
11/12/2013
2/3/2014
14/5/2014
13/6/2014
7/7/2014
4/8/2014
];

InvoiceData:
LOAD *,
WeekStart(InvDate, 1) AS WkStart
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för weekstart()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	WkStart
28/03/2012	02/04/2012
10/12/2012	17/12/2012

InvDate	WkStart
5/2/2013	11/02/2013
31/3/2013	01/04/2013
19/5/2013	20/05/2013
15/9/2013	16/09/2013
11/12/2013	16/12/2013
2/3/2014	03/03/2014
14/5/2014	19/05/2014
13/6/2014	16/06/2014
7/7/2014	14/07/2014
4/8/2014	11/08/2014

weekyear

Denna funktion returnerar det år som veckonumret hör till enligt ISO 8601. Veckonummer går från 1 till cirka 52.

Syntax:

```
weekyear(expression)
```

Returnerad datatyp: heltal

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`weekyear( '1996-12-30' )`	returnerar 1997, eftersom vecka 1 i 1997 börjar 1996-12-30
`weekyear( '1997-01-02' )`	returnerar 1997
`weekyear( '1997-12-28' )`	returnerar 1997
`weekyear( '1997-12-30' )`	returnerar 1998, eftersom vecka 1 i 1998 börjar den 1997-12-29
`weekyear( '1999-01-02' )`	returnerar 1998, eftersom vecka 53 i 1998 slutar den 1999-01-03

Begränsningar:

Vissa år börjar vecka #1 i december, till exempel december 1997. Andra år kan börja med vecka #53 från föregående år, till exempel januari 1999. Under dessa få dagar då veckonumret hör till ett annat år returnerar funktionerna **year** och **weekyear** andra värden.

year

Denna funktion returnerar ett heltal som motsvarar året om **expression** tolkas som ett datum enligt standardtolkningen av tal.

Syntax:

```
year(expression)
```

Returnerad datatyp: heltal

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`year( '2012-10-12' )`	returnerar 2012
`year( '35648' )`	returnerar 1997, eftersom 35648 = 1997-08-06

yearend

Denna funktion returnerar ett värde som motsvarar en tidsmarkör med den sista millisekunden av den sista dagen i det år som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

Syntax:

```
YearEnd( date[, period_no[, first_month_of_year = 1]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
date	Det datum som ska evalueras.
period_no	**period_no** är ett heltal, där värdet 0 anger det år som innehåller **date**. Negativa värden i **period_no** anger föregående år och positiva värden anger efterföljande år.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`yearend ( '19/10/2001' )`	Returnerar 31/12/2001 23:59:59.
`yearend ( '19/10/2001', -1 )`	Returnerar 31/12/2000 23:59:59.
`yearend ( '19/10/2001', 0, 4)`	Returnerar 31/03/2002 23:59:59.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet hittas den sista dagen på året för varje faktureringsdatum i tabellen, där den första månaden på året har angetts som månad 4.

```
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 *,
YearEnd(InvDate, 0, 4) AS YrEnd
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för yearend ()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	YrEnd
28/03/2012	31/03/2011
10/12/2012	31/03/2012
5/2/2013	31/03/2013
31/3/2013	31/03/2013

19/5/2013	31/03/2014
15/9/2013	31/03/2014
11/12/2013	31/03/2014
2/3/2014	31/03/2014
14/5/2014	31/03/2015
13/6/2014	31/03/2015
7/7/2014	31/03/2015
4/8/2014	31/03/2015

yearname

Denna funktion returnerar ett fyrsiffrigt år som visningsvärde med ett underliggande numeriskt värde som motsvarar en tidsmarkör för den första millisekunden av den första dagen på det år som innehåller **date**.

Syntax:

```
YearName(date[, period_no[, first_month_of_year]] )
```

Returnerad datatyp: dual

Argument:

Argument	Beskrivning
date	Det datum som ska evalueras.
period_no	**period_no** är ett heltal, där värdet 0 anger det år som innehåller **date**. Negativa värden i **period_no** anger föregående år och positiva värden anger efterföljande år.
first_ month_of_ year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**. Visningsvärdet blir då en sträng som visar två år.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`yearname ( '19/10/2001')`	Returnerar 2001.

Exempel	Resultat
`yearname ( '19/10/2001', -1 )`	Returnerar 2000.
`yearname ( '19/10/2001', 0, 4)`	Returnerar 2001-2002.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet hittas den första dagen på året för varje faktureringsdatum i tabellen, där den första månaden på året har angetts som månad 4.

I det här exemplet skapar vi ett fyra-plus-fyra-siffror-namn för åren som varje faktureringsdatum i tabellen finns på. Detta beror på att den första månaden på året angetts som månad 4.

```
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 *,
YearName(InvDate, 0, 4) AS YrName
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för yearname()-funktionen.

Resultattabell

InvDate	YrName
28/03/2012	2011-2012
10/12/2012	2012-2013
5/2/2013	2012-2013
31/3/2013	2012-2013

InvDate	YrName
19/5/2013	2013-2014
15/9/2013	2013-2014
11/12/2013	2013-2014
2/3/2014	2013-2014
14/5/2014	2014-2015
13/6/2014	2014-2015
7/7/2014	2014-2015
4/8/2014	2014-2015

yearstart

Denna funktion returnerar en tidsmarkör som motsvarar starten av den första dagen i det år som innehåller **date**. Det förvalda utdataformatet blir det **DateFormat** som har definierats i skriptet.

Syntax:

```
YearStart(date[, period_no[, first_month_of_year]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
date	Det datum som ska evalueras.
period_no	**period_no** är ett heltal, där värdet 0 anger det år som innehåller **date**. Negativa värden i **period_no** anger föregående år och positiva värden anger efterföljande år.
first_month_of_year	Om du vill arbeta med (budget)år som inte börjar i januari kan du ange ett värde mellan 2 och 12 i **first_month_of_year**.

Exempel och resultat:
I de här exemplen används datumformatet **DD/MM/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet. Ändra formatet i exemplen så att det passar dina krav.

Skriptexempel

Exempel	Resultat
`yearstart ('19/10/2001')`	Returnerar 01/01/2001.

Exempel	Resultat
`yearstart ('19/10/2001', -1)`	Returnerar 01/01/2000.
`yearstart ('19/10/2001', 0, 4)`	Returnerar 01/04/2001.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.

I det här exemplet hittas den första dagen på året för varje faktureringsdatum i tabellen, där den första månaden på året har angetts som månad 4.

```
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 *,
YearStart(InvDate, 0, 4) AS YrStart
Resident TempTable;
Drop table TempTable;
```

Den resulterande tabellen innehåller de ursprungliga datumen och en kolumn med returvärdet för yearstart ()-funktionen. Du kan visa hela tidsmarkören genom att ange formateringen i egenskapspanelen.

Resultattabell

InvDate	YrStart
28/03/2012	01/04/2011
10/12/2012	01/04/2012
5/2/2013	01/04/2012
31/3/2013	01/04/2012
19/5/2013	01/04/2013
15/9/2013	01/04/2013

InvDate	YrStart
11/12/2013	01/04/2013
2/3/2014	01/04/2013
14/5/2014	01/04/2014
13/6/2014	01/04/2014
7/7/2014	01/04/2014
4/8/2014	01/04/2014

yeartodate

Denna funktion räknar ut om indatatidsmarkören hamnar inom året för datumet då skriptet senast laddades och returnerar True om så är fallet, False om så inte är fallet.

Syntax:

```
YearToDate(timestamp[ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Returnerad datatyp: Boolesk

Om ingen av de valfria parametrarna används, omfattar definitionen alla datum inom ett kalenderår från den 1 januari t.o.m. datumet för senaste skriptexekvering.

Argument:

Argument

Argument	Beskrivning
timestamp	Den tidsmarkör som ska utvärderas, exempelvis 2012-10-12.
yearoffset	Genom att ange en **yearoffset**, **yeartodate** returneras True för samma period ett annat år. Ett negativt värde på **yearoffset** indikerar ett tidigare år, ett positivt värde ett framtida år. Det senaste året uppnås genom att ange yearoffset = -1. Om det utelämnas, antas 0.
firstmonth	Genom att ange en **firstmonth** mellan 1 och 12 (1 om parametern utelämnas), kan årets början flyttas till första dagen på valfri månad. Om du exempelvis vill börja ett budgetår den 1 maj, sätter du **firstmonth** till 5.
todaydate	Genom att ange **todaydate** (tidsmarkör för senaste skriptexekveringen om parametern utelämnas), kan du flytta dagen som används som periodens övre gräns.

Exempel och resultat:
I följande exempel antas att senaste laddningstillfället är 2011-11-18.

Skriptexempel

Exempel	Resultat
`yeartodate( '2010-11-18')`	returnerar False
`yeartodate( '2011-02-01')`	returnerar True
`yeartodate( '2011-11-18')`	returnerar True
`yeartodate( '2011-11-19')`	returnerar False
`yeartodate( '2011-11-19', 0, 1, '2011-12-31')`	returnerar True
`yeartodate( '2010-11-18', -1)`	returnerar True
`yeartodate( '2011-11-18', -1)`	returnerar False
`yeartodate( '2011-04-30', 0, 5)`	returnerar False
`yeartodate( '2011-05-01', 0, 5)`	returnerar True

5.8 Exponentiella och logaritmiska funktioner

Den här delen beskriver funktioner som är relaterade till exponential- och logaritmberäkningar. Alla funktioner kan användas både i dataladdningsskriptet och diagramuttryck.

I funktionerna nedan är parametrarna uttryck där **x** och **y** ska tolkas som reella tal.

exp

Den naturliga exponentiella funktionen, e^x, med den naturliga logaritmen **e** som bas. Resultatet blir ett positivt tal.

exp(*x* **)**

Exempel och resultat:

`exp(3)` returnerar 20,085.

log

Den naturliga logaritmen av **x**. Funktionen är bara definierad om **x**> 0. Resulterar i ett tal.

log(*x* **)**

Exempel och resultat:

`log(3)` returnerar 1,0986.

log10

Den vanliga logaritmen (bas 10) av **x**. Funktionen är bara definierad om **x**> 0. Resulterar i ett tal.

log10(*x* **)**

Exempel och resultat:

`log10(3)` returnerar 0.4771

pow
Returnerar **x** upphöjt till **y**. Resulterar i ett tal.

```
pow(x,y )
```

Exempel och resultat:

`pow(3, 3)` returnerar 27

sqr
x i kvadrat (**x** upphöjt till 2). Resulterar i ett tal.

```
sqr (x )
```

Exempel och resultat:

`sqr(3)` returnerar 9

sqrt
Kvadratroten av **x**. Funktionen är bara definierad om **x** >= 0. Resultatet blir ett positivt tal.

```
sqrt(x )
```

Exempel och resultat:

`sqrt(3)` returnerar 1.732

5.9 Fältfunktioner

Dessa funktioner kan endast användas i diagramuttryck.

Fältfunktioner returnerar antingen heltal eller strängar som identifierar olika aspekter av fälturval.

Räknefunktioner

GetAlternativeCount
GetAlternativeCount() används för att hitta antalet alternativa (ljusgrå) värden i det identifierade fältet.

```
GetAlternativeCount - diagramfunktion (field_name)
```

GetExcludedCount
GetExcludedCount() finner antalet uteslutna distinkta värden i det identifierade fältet. Uteslutna värden omfattar alternativa (ljusgrå), uteslutna (mörkgrå) och valda uteslutna (mörkgrå med markering) fält.

```
GetExcludedCount - diagramfunktion (page 563)(field_name)
```

GetNotSelectedCount
Denna diagramfunktion returnerar antalet icke-valda värden i fältet **fieldname**. Fältet måste vara i and-läge för att funktionen ska vara relevant.

```
GetNotSelectedCount - diagramfunktion(fieldname [, includeexcluded=false])
```

GetPossibleCount
GetPossibleCount() används för att hitta antalet möjliga värden i det identifierade fältet. Om det identifierade fältet innehåller urval räknas de markerade (gröna) fälten. Värden som är associerade på annat sätt (vita värden) räknas.

```
GetPossibleCount - diagramfunktion(field_name)
```

GetSelectedCount
GetSelectedCount() finner antalet valda (gröna) värden i ett fält.

```
GetSelectedCount - diagramfunktion (field_name [, include_excluded])
```

Fält- och urvalsfunktioner

GetCurrentSelections
GetCurrentSelections() returnerar en lista med de aktuella urvalen i appen. Om urvalen görs med en söksträng i en sökruta returnerar **GetCurrentSelections()** söksträngen.

```
GetCurrentSelections - diagramfunktion([record_sep [,tag_sep [,value_sep
[,max_values]]]])
```

GetFieldSelections
GetFieldSelections() returnerar en **sträng** med de aktuella urvalen i ett fält.

```
GetFieldSelections - diagramfunktion ( field_name [, value_sep [, max_
values]])
```

GetObjectDimension
GetObjectDimension() returnerar namnet på dimensionen. **Index** är ett valfritt heltal som anger dimensionen som ska returneras.

```
GetObjectDimension - diagramfunktion ([index])
```

GetObjectField
GetObjectField() returnerar namnet på dimensionen. **Index** är ett valfritt heltal som anger dimensionen som ska returneras.

```
GetObjectField - diagramfunktion ([index])
```

GetObjectMeasure
GetObjectMeasure() returnerar namnet på måttet. **Index** är ett valfritt heltal som anger måttet som ska returneras.

```
GetObjectMeasure - diagramfunktion ([index])
```

GetAlternativeCount - diagramfunktion

GetAlternativeCount() används för att hitta antalet alternativa (ljusgrå) värden i det identifierade fältet.

Syntax:

```
GetAlternativeCount (field_name)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
field_name	Fältet som innehåller dataintervallet som ska mätas.

Exempel och resultat:

I följande exempel används fältet **First name** som laddas i en filterruta.

Exempel och resultat

Exempel	Resultat
Anta att **John** har valts under **First name**. `GetAlternativeCount ([First name])`	4 eftersom det finns 4 unika och uteslutna (grå) värden under **First name**.
Anta att **John** och **Peter** är valda. `GetAlternativeCount ([First name])`	3 eftersom det finns 3 unika och uteslutna (grå) värden under **First name**.
Anta att inga värden är valda under **First name**. `GetAlternativeCount ([First name])`	0 eftersom det inte finns några val.

Data som används i exemplet:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

GetCurrentSelections - diagramfunktion

GetCurrentSelections() returnerar en lista med de aktuella urvalen i appen. Om urvalen görs med en sökning i en sökruta returnerar **GetCurrentSelections()** söksträngen.

Om alternativ används måste du ange record_sep. Om du vill ange en ny rad ställer du in **record_sep** som **chr(13)&chr(10)**.

Om alla utom två värden, eller alla utom ett värde, är valda används formatet "NOT x,y" respektive "NOT y". Om du väljer alla värden och antalet värden blir högre än max_values, returneras texten ALL.

Syntax:

```
GetCurrentSelections ([record_sep [, tag_sep [, value_sep [, max_values [, state_name]]]]])
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
record_sep	Avgränsaren som avgränsar posterna. Standardinställningen är <CR><LF> vilket innebär en ny rad.
tag_sep	Avgränsaren som ska finnas mellan fältnamnets tagg och fältvärdena. Standard är': '.
value_sep	Avgränsaren som avgränsar fältvärdena. ',' är standard.
max_values	Det maximala antalet fältvärden som ska visas individuellt i en lista. När ett större antal värden är valt används i stället formatet 'x av y värden'. 6 är standardvärde.
state_name	Namnet på ett alternativt tillstånd som har valts till den specifika visualiseringen. Om argumentet **state_name** används tas endast hänsyn till de urval som är förknippade med det angivna tillståndsnamnet.

Exempel och resultat:

I exemplet som följer har två fält lästs in i olika filterrutor, en för **First name**, och en för **Initials**.

Exempel och resultat

Exempel	Resultat
Anta att **John** har valts under **First name**. `GetCurrentSelections ()`	'First name: John'
Anta att **John** och **Peter** är valda under **First name**. `GetCurrentSelections ()`	'First name: John, Peter'

Exempel	Resultat
Anta att **John** och **Peter** väljs under **First name** och **JA** är valt under**Initials**. `GetCurrentSelections ()`	'First name: John, Peter Initials: JA'
Anta att **John** är vald under **First name** och **JA** är valt under **Initials**. `GetCurrentSelections ( chr(13)&chr(10) , ' = ' )`	'First name = John Initials = JA'
Anta att du har valt alla namn utom Sue under **First name** och inga val under **Initials**. `GetCurrentSelections (chr(13)&chr(10), '=', ',' ,3)`	'First name=NOT Sue'

Data som används i exemplet:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

GetExcludedCount - diagramfunktion

GetExcludedCount() finner antalet uteslutna distinkta värden i det identifierade fältet. Uteslutna värden omfattar alternativa (ljusgrå), uteslutna (mörkgrå) och valda uteslutna (mörkgrå med markering) fält.

Syntax:

```
GetExcludedCount (field_name)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
field_name	Fältet som innehåller dataintervallet som ska mätas.

Exempel och resultat:

I exemplet som följer har tre fält lästs in i olika filterrutor, ett för **First name**, ett för **Last name** och ett för **Initials**.

Exempel och resultat

Exempel	Resultat
Om inga värden är valda i **First name**.	`GetExcludedCount (Initials) = 0` Inga urval finns.
Om **John** är valt i **First name**.	`GetExcludedCount (Initials) = 5` Det finns 5 uteslutna värden i **Initials** (mörkgrå färg). Den sjätte cellen (JA) är vit eftersom den är associerad med urvalet John i **First name**.
Om **John** och **Peter** är valda.	`GetExcludedCount (Initials) = 3` John är associerat med 1 värde och Peter är associerat med 2 värden i **Initials**.
Om **John** och **Peter** är valda i **First name**, och **Franc** är valt i **Last name**.	`GetExcludedCount ([First name]) = 4` Det finns 4 uteslutna värden i **First name** (mörkgrå färg). **GetExcludedCount()** utvärderas för fält med uteslutna värden, inklusive alternativa och valda uteslutna fält.
Om **John** och **Peter** är valda i **First name**, och **Franc** och **Anderson** är valda i **Last name**.	`GetExcludedCount (Initials) = 4` Det finns 4 uteslutna värden i **Initials** (mörkgrå färg). De andra två cellerna (JA och PF) är vita eftersom de är associerade med valen John och Peter i **First name**.
Om **John** och **Peter** är valda i **First name**, och **Franc** och **Anderson** är valda i **Last name**.	`GetExcludedCount ([Last name]) = 4` Det finns 4 uteslutna värden i **Initials**. Devonshire är ljusgrått; Brown, Carr och Elliot är mörkgrå.

Data som används i exemplet:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

GetFieldSelections - diagramfunktion

GetFieldSelections() returnerar en **sträng** med de aktuella urvalen i ett fält.

Om alla utom två värden, eller alla utom ett värde, är valda används formatet ”NOT x,y” respektive ”NOT y”. Om du väljer alla värden och antalet värden blir högre än max_values, returneras texten ALL.

Syntax:

```
GetFieldSelections ( field_name [, value_sep [, max_values [, state_name]]])
```

Returnerad datatyp: sträng

Retursträngformat

Format	Beskrivning
'a, b, c'	Om antalet valda värden är max_values eller färre, returneras en sträng med en lista över de valda värdena. Värdena separeras med value_sep som avgränsare.
'NOT a, b, c'	Om antalet ej valda värden är max_values eller färre, returneras en sträng med en lista över de ej valda värdena med NOT som prefix. Värdena separeras med value_sep som avgränsare.
'x of y'	x = antalet valda värden y = det totala antalet värden Det här returneras när max_values < x < (y - max_values).
'ALL'	Returneras om alla värden har valts.
'-'	Returneras om inget värde har valts.
<search string>	Om du har valt med sökning returneras söksträngen.

Argument:

Argument

Argument	Beskrivning
field_name	Fältet som innehåller dataintervallet som ska mätas.
value_sep	Avgränsaren som avgränsar fältvärdena. ',' är standard.
max_values	Det maximala antalet fältvärden som ska visas individuellt i en lista. När ett större antal värden är valt används i stället formatet 'x av y värden'. 6 är standardvärde.
state_name	Namnet på ett alternativt tillstånd som har valts till den specifika visualiseringen. Om argumentet **state_name** används tas endast hänsyn till de urval som är förknippade med det angivna tillståndsnamnet.

Exempel och resultat:

I följande exempel används fältet **First name** som laddas i en filterruta.

Exempel och resultat

Exempel	Resultat
Anta att **John** har valts under **First name**. `GetFieldSelections ([First name])`	'John'
Anta att **John** och **Peter** är valda. `GetFieldSelections ([First name])`	'John,Peter'
Anta att **John** och **Peter** är valda. `GetFieldSelections ([First name],'; ')`	'John; Peter'
Anta att **John**, **Sue**, **Mark** är valda under **First name**. `GetFieldSelections ([First name],';',2)`	"NOT Jane;Peter", eftersom värdet 2 anges som värdet för max_values-argumentet. Annars skulle resultatet ha blivit John; Sue; Mark.

Data som används i exemplet:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

GetNotSelectedCount - diagramfunktion

Denna diagramfunktion returnerar antalet icke-valda värden i fältet **fieldname**. Fältet måste vara i and-läge för att funktionen ska vara relevant.

Syntax:

```
GetNotSelectedCount(fieldname [, includeexcluded=false])
```

Argument:

Argument

Argument	Beskrivning
fieldname	Namnet på det fält som ska utvärderas.

Argument	Beskrivning
includeexcluded	Om **includeexcluded** anges som True inkluderas valda värden som är exkluderade av urval i andra fält.

Exempel:

```
GetNotSelectedCount( Country )
GetNotSelectedCount( Country, true )
```

GetObjectDimension - diagramfunktion

GetObjectDimension() returnerar namnet på dimensionen. **Index** är ett valfritt heltal som anger dimensionen som ska returneras.

Du kan inte använda den här funktionen i ett diagram på följande platser: titel, underrubrik, fotnot, referenslinjeuttryck.

Du kan inte referera till namnet på en dimension eller ett mått i ett annat objekt med Object ID.

Syntax:

```
GetObjectDimension ([index])
```

Exempel:

```
GetObjectDimension(1)
```

Exempel: Diagramuttryck

Qlik Sense-tabell som visar exempel på GetObjectDimension-funktionen i ett diagramuttryck

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

Om du vill returnera namnet på ett mått använder du funktionen **GetObjectMeasure** istället.

GetObjectField - diagramfunktion

GetObjectField() returnerar namnet på dimensionen. **Index** är ett valfritt heltal som anger dimensionen som ska returneras.

Du kan inte använda den här funktionen i ett diagram på följande platser: titel, underrubrik, fotnot, referenslinjeuttryck.

Du kan inte referera till namnet på en dimension eller ett mått i ett annat objekt med Object ID.

Syntax:

```
GetObjectField ([index])
```

Exempel:

```
GetObjectField(1)
```

Exempel: Diagramuttryck

Qlik Sense-tabell som visar exempel på GetObjectField-funktionen i ett diagramuttryck.

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

Om du vill returnera namnet på ett mått använder du funktionen **GetObjectMeasure** istället.

GetObjectMeasure - diagramfunktion

GetObjectMeasure() returnerar namnet på måttet. **Index** är ett valfritt heltal som anger måttet som ska returneras.

Du kan inte använda den här funktionen i ett diagram på följande platser: titel, underrubrik, fotnot, referenslinjeuttryck.

Du kan inte referera till namnet på en dimension eller ett mått i ett annat objekt med Object ID.

Syntax:

```
GetObjectMeasure ([index])
```

Exempel:

```
GetObjectMeasure(1)
```

Exempel: Diagramuttryck

Qlik Sense-tabell som visar exempel på GetObjectMeasure-funktionen i ett diagramuttryck

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)

Om du vill returnera namnet på en dimension använder du funktionen **GetObjectField** istället.

GetPossibleCount - diagramfunktion

GetPossibleCount() används för att hitta antalet möjliga värden i det identifierade fältet. Om det identifierade fältet innehåller urval räknas de markerade (gröna) fälten. Värden som är associerade på annat sätt (vita värden) räknas. .

För fält med urval returnerar **GetPossibleCount()** antalet valda (gröna) fält.

Returnerad datatyp: heltal

Syntax:

```
GetPossibleCount (field_name)
```

Argument:

Argument

Argument	Beskrivning
field_name	Fältet som innehåller dataintervallet som ska mätas.

Exempel och resultat:

I exemplet som följer har två fält lästs in i olika filterrutor, en för **First name**, och en för **Initials**.

Exempel och resultat

Exempel	Resultat
Anta att **John** har valts under **First name**. `GetPossibleCount ([Initials])`	1 eftersom det finns 1 värde bland initialerna kopplade till valet, **John**, i **First name**.
Anta att **John** har valts under **First name**. `GetPossibleCount ([First name])`	1 då det finns 1 val, **John**, i **First name**.

Exempel	Resultat
Anta att **Peter** har valts under **First name**. `GetPossibleCount ([Initials])`	2 eftersom Peter är associerad med 2 värden under **Initials**.
Anta att inga värden är valda under **First name**. `GetPossibleCount ([First name])`	5 då det inte finns några val och det finns fem unika värden under **First name**.
Anta att inga värden är valda under **First name**. `GetPossibleCount ([Initials])`	6 då det inte finns några val och det finns sex unika värden under **Initials**.

Data som används i exemplet:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

GetSelectedCount - diagramfunktion

GetSelectedCount() finner antalet valda (gröna) värden i ett fält.

Syntax:

```
GetSelectedCount (field_name [, include_excluded [, state_name]])
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
field_name	Fältet som innehåller dataintervallet som ska mätas.
include_ excluded	Om det anges som **True()**, inkluderar räkningen valda värden som just nu är exkluderade av urval i andra fält. Om de är False eller utelämnade inkluderas dessa värden inte.
state_name	Namnet på ett alternativt tillstånd som har valts till den specifika visualiseringen. Om argumentet **state_name** används tas endast hänsyn till de urval som är förknippade med det angivna tillståndsnamnet.

Exempel och resultat:

I exemplet som följer har tre fält lästs in i olika filterrutor, en för förnamn, **First name**, en för initialer, **Initials**, och en för mobiltelefoni, **Has cellphone**.

Exempel och resultat

Exempel	Resultat
Anta att **John** har valts under **First name**. `GetSelectedCount ([First name])`	1 eftersom ett värde är valt under **First name**.
Anta att **John** har valts under **First name**. `GetSelectedCount ([Initials])`	0 då inga värden är valda under **Initials**.
Om det inte finns några val i **First name** markerar du alla värden i **Initials** och efter det markerar du värdet **Yes** under **Has cellphone**. `GetSelectedCount ([Initials], True())`	6. Även om val med **Initials**MC och PD har **Has cellphone** inställd på **No**, är resultatet fortfarande 6, eftersom argumentet include_excluded är inställt på True().

Data som används i exemplet:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

5.10 Filfunktioner

Filfunktionerna (finns endast i skriptuttryck) returnerar information om tabellfilen som för närvarande läses. Funktionerna returnerar NULL för alla datakällor utom tabellfiler (undantag: **ConnectString( )**).

Filfunktioner – en översikt

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

Attribute

Denna skriptfunktion returnerar metataggarnas värde från olika filformat som text. Följande filformat stöds: MP3, WMA, WMV, PNG och JPG. Om filen **filename** inte finns, inte är en filtyp som stöds eller inte innehåller en metatagg som heter **attributename**, returneras NULL.

```
Attribute (filename, attributename)
```

ConnectString

ConnectString()-funktionen returnerar namnet på den aktiva datakopplingen för ODBC- eller OLE DB -kopplingar. Funktionen returnerar en tom sträng om ingen **connect**-sats har exekverats eller efter en **disconnect**-sats.

```
ConnectString ()
```

FileBaseName

FileBaseName-funktionen returnerar en textsträng som anger namnet på den tabellfil som håller på att läsas in, dock utan sökväg och filtillägg.

```
FileBaseName ()
```

FileDir

FileDir-funktionen returnerar en textsträng som anger sökvägen till katalogen som innehåller den tabellfil som håller på att läsas in.

```
FileDir ()
```

FileExtension

FileExtension-funktionen returnerar en textsträng som anger filtillägget till den tabellfil som håller på att läsas in.

```
FileExtension ()
```

FileName

FileName-funktionen returnerar en textsträng som anger namnet på den tabellfil som håller på att läsas in, utan sökväg men med filtillägget.

```
FileName ()
```

FilePath

FilePath-funktionen returnerar en textsträng som anger den kompletta sökvägen till den tabellfil som håller på att läsas in.

```
FilePath ()
```

FileSize

FileSize-funktionen returnerar ett heltal som anger storleken i byte för filen filename eller, om inget filename angetts, för den tabellfil som håller på att läsas in.

```
FileSize ()
```

FileTime

FileTime-funktionen returnerar en tidsmarkör för datum och tid för den senaste ändringen av filen filename. Om inget filename har angetts, använder funktionen den senast inlästa tabellfilen.

```
FileTime ([ filename ])
```

GetFolderPath

Funktionen **GetFolderPath** returnerar värdet av funktionen Microsoft Windows *SHGetFolderPath*. Den här funktionen tar som indata namnet på en Microsoft Windows -mapp och returnerar den fullständiga sökvägen till mappen.

```
GetFolderPath ()
```

QvdCreateTime

Denna skriptfunktion returnerar XML-huvudets tidsstämpel från en QVD-fil, om någon sådan finns, annars returneras NULL.

```
QvdCreateTime (filename)
```

QvdFieldName

Denna skriptfunktion returnerar namnet på fält nummer **fieldno** i en QVD-fil. Om fältet inte finns returneras NULL.

```
QvdFieldName (filename , fieldno)
```

QvdNoOfFields

Denna skriptfunktion returnerar antalet fält i en QVD-fil.

```
QvdNoOfFields (filename)
```

QvdNoOfRecords

Denna skriptfunktion returnerar aktuellt antal poster i en QVD-fil.

```
QvdNoOfRecords (filename)
```

QvdTableName

Denna skriptfunktion returnerar namnet på tabellen i en QVD-fil.

```
QvdTableName (filename)
```

Attribute

Denna skriptfunktion returnerar metataggarnas värde från olika filformat som text. Följande filformat stöds: MP3, WMA, WMV, PNG och JPG. Om filen **filename** inte finns, inte är en filtyp som stöds eller inte innehåller en metatagg som heter **attributename**, returneras NULL.

Syntax:

```
Attribute(filename, attributename)
```

Ett stort antal metataggar kan läsas. Exemplen i det här avsnittet visar vilka taggar som kan läsas för respektive filtyp som stöds.

> *Du kan bara läsa metataggar som har sparats i filen enligt den relevanta specifikationen, exempelvis ID2v3 för MP3-filer eller EXIF för JPG-filer, inte metainformation som har sparats i* ***Utforskaren*** *i Windows.*

Argument:

Argument

Argument	Beskrivning
filename	Namnet på en mediafil, inklusive sökväg om så krävs, som en mappdatakoppling. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense-appens arbetskatalog. **Exempel:** *data*
attributename	Namnet på en metatagg.

I exemplen används **GetFolderPath**-funktionen för att hitta sökvägarna till mediefiler. Eftersom **GetFolderPath** endast stöds i bakåtkompatibelt läge måste du byta ut referenserna till **GetFolderPath** mot en lib://-datakopplingssökväg när du använder funktionen i standardläge eller i Qlik Sense SaaS.

Behörighetskontroll för filsystem (page 814)

Example 1: MP3-filer

Det här skriptet läser alla möjliga MP3-metataggar i mappen *MyMusic*.

```
// Script to read MP3 meta tags for each vExt in 'mp3' for each vFoundFile in filelist(
GetFolderPath('MyMusic') & '\*.'& vExt ) FileList: LOAD FileLongName,     subfield
(FileLongName,'\',-1) as FileShortName,     num(FileSize(FileLongName),'# ### ### ###',',','
') as FileSize,     FileTime(FileLongName) as FileTime,      // ID3v1.0 and ID3v1.1 tags
Attribute(FileLongName, 'Title') as Title,     Attribute(FileLongName, 'Artist') as Artist,
 Attribute(FileLongName, 'Album') as Album,     Attribute(FileLongName, 'Year') as Year,
Attribute(FileLongName, 'Comment') as Comment,     Attribute(FileLongName, 'Track') as Track,
   Attribute(FileLongName, 'Genre') as Genre,

     // ID3v2.3 tags     Attribute(FileLongName, 'AENC') as AENC, // Audio encryption
Attribute(FileLongName, 'APIC') as APIC, // Attached picture     Attribute(FileLongName,
'COMM') as COMM, // Comments     Attribute(FileLongName, 'COMR') as COMR, // Commercial frame
   Attribute(FileLongName, 'ENCR') as ENCR, // Encryption method registration     Attribute
```

```
(FileLongName, 'EQUA') as EQUA, // Equalization     Attribute(FileLongName, 'ETCO') as ETCO,
// Event timing codes     Attribute(FileLongName, 'GEOB') as GEOB, // General encapsulated
object     Attribute(FileLongName, 'GRID') as GRID, // Group identification registration
Attribute(FileLongName, 'IPLS') as IPLS, // Involved people list     Attribute(FileLongName,
'LINK') as LINK, // Linked information     Attribute(FileLongName, 'MCDI') as MCDI, // Music
CD identifier     Attribute(FileLongName, 'MLLT') as MLLT, // MPEG location lookup table
Attribute(FileLongName, 'OWNE') as OWNE, // Ownership frame     Attribute(FileLongName,
'PRIV') as PRIV, // Private frame     Attribute(FileLongName, 'PCNT') as PCNT, // Play counter
    Attribute(FileLongName, 'POPM') as POPM, // Popularimeter

     Attribute(FileLongName, 'POSS') as POSS, // Position synchronisation frame     Attribute
(FileLongName, 'RBUF') as RBUF, // Recommended buffer size     Attribute(FileLongName, 'RVAD')
as RVAD, // Relative volume adjustment     Attribute(FileLongName, 'RVRB') as RVRB, // Reverb
   Attribute(FileLongName, 'SYLT') as SYLT, // Synchronized lyric/text     Attribute
(FileLongName, 'SYTC') as SYTC, // Synchronized tempo codes     Attribute(FileLongName,
'TALB') as TALB, // Album/Movie/Show title     Attribute(FileLongName, 'TBPM') as TBPM, // BPM
(beats per minute)     Attribute(FileLongName, 'TCOM') as TCOM, // Composer     Attribute
(FileLongName, 'TCON') as TCON, // Content type     Attribute(FileLongName, 'TCOP') as TCOP,
// Copyright message     Attribute(FileLongName, 'TDAT') as TDAT, // Date     Attribute
(FileLongName, 'TDLY') as TDLY, // Playlist delay

     Attribute(FileLongName, 'TENC') as TENC, // Encoded by     Attribute(FileLongName,
'TEXT') as TEXT, // Lyricist/Text writer     Attribute(FileLongName, 'TFLT') as TFLT, // File
type     Attribute(FileLongName, 'TIME') as TIME, // Time     Attribute(FileLongName, 'TIT1')
as TIT1, // Content group description     Attribute(FileLongName, 'TIT2') as TIT2, //
Title/songname/content description     Attribute(FileLongName, 'TIT3') as TIT3, //
Subtitle/Description refinement     Attribute(FileLongName, 'TKEY') as TKEY, // Initial key
 Attribute(FileLongName, 'TLAN') as TLAN, // Language(s)     Attribute(FileLongName, 'TLEN')
as TLEN, // Length     Attribute(FileLongName, 'TMED') as TMED, // Media type

     Attribute(FileLongName, 'TOAL') as TOAL, // Original album/movie/show title     Attribute
(FileLongName, 'TOFN') as TOFN, // Original filename     Attribute(FileLongName, 'TOLY') as
TOLY, // Original lyricist(s)/text writer(s)     Attribute(FileLongName, 'TOPE') as TOPE, //
Original artist(s)/performer(s)     Attribute(FileLongName, 'TORY') as TORY, // Original
release year     Attribute(FileLongName, 'TOWN') as TOWN, // File owner/licensee     Attribute
(FileLongName, 'TPE1') as TPE1, // Lead performer(s)/Soloist(s)     Attribute(FileLongName,
'TPE2') as TPE2, // Band/orchestra/accompaniment

     Attribute(FileLongName, 'TPE3') as TPE3, // Conductor/performer refinement     Attribute
(FileLongName, 'TPE4') as TPE4, // Interpreted, remixed, or otherwise modified by
Attribute(FileLongName, 'TPOS') as TPOS, // Part of a set     Attribute(FileLongName, 'TPUB')
as TPUB, // Publisher     Attribute(FileLongName, 'TRCK') as TRCK, // Track number/Position in
set     Attribute(FileLongName, 'TRDA') as TRDA, // Recording dates     Attribute
(FileLongName, 'TRSN') as TRSN, // Internet radio station name     Attribute(FileLongName,
'TRSO') as TRSO, // Internet radio station owner

     Attribute(FileLongName, 'TSIZ') as TSIZ, // Size     Attribute(FileLongName, 'TSRC') as
TSRC, // ISRC (international standard recording code)     Attribute(FileLongName, 'TSSE') as
TSSE, // Software/Hardware and settings used for encoding     Attribute(FileLongName, 'TYER')
as TYER, // Year     Attribute(FileLongName, 'TXXX') as TXXX, // User defined text information
frame     Attribute(FileLongName, 'UFID') as UFID, // Unique file identifier     Attribute
(FileLongName, 'USER') as USER, // Terms of use     Attribute(FileLongName, 'USLT') as USLT,
// Unsychronized lyric/text transcription     Attribute(FileLongName, 'WCOM') as WCOM, //
Commercial information     Attribute(FileLongName, 'WCOP') as WCOP, // Copyright/Legal
information
```

```
    Attribute(FileLongName, 'WOAF') as WOAF, // Official audio file webpage     Attribute
(FileLongName, 'WOAR') as WOAR, // Official artist/performer webpage     Attribute
(FileLongName, 'WOAS') as WOAS, // Official audio source webpage     Attribute(FileLongName,
'WORS') as WORS, // Official internet radio station homepage     Attribute(FileLongName,
'WPAY') as WPAY, // Payment     Attribute(FileLongName, 'WPUB') as WPUB, // Publishers
official webpage     Attribute(FileLongName, 'WXXX') as WXXX; // User defined URL link frame
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels); Next vFoundFile Next vExt
```

Example 2: JPEG

Det här skriptet läser alla möjliga EXIF-metataggar från JPG-filer i mappen *MyPictures*.

```
// Script to read Jpeg Exif meta tags for each vExt in 'jpg', 'jpeg', 'jpe', 'jfif', 'jif',
'jfi' for each vFoundFile in filelist( GetFolderPath('MyPictures') & '\*.'& vExt )

 FileList: LOAD FileLongName,     subfield(FileLongName,'\',-1) as FileShortName,     num
(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,     FileTime(FileLongName) as
FileTime,      // ************   Exif Main (IFD0) Attributes   ************     Attribute
(FileLongName, 'ImageWidth') as ImageWidth,     Attribute(FileLongName, 'ImageLength') as
ImageLength,     Attribute(FileLongName, 'BitsPerSample') as BitsPerSample,     Attribute
(FileLongName, 'Compression') as Compression,

    //  examples: 1=uncompressed, 2=CCITT, 3=CCITT 3, 4=CCITT 4,

    //5=LZW, 6=JPEG (old style), 7=JPEG, 8=Deflate, 32773=PackBits RLE,      Attribute
(FileLongName, 'PhotometricInterpretation') as PhotometricInterpretation,

    //  examples: 0=WhiteIsZero, 1=BlackIsZero, 2=RGB, 3=Palette, 5=CMYK, 6=YCbCr,
Attribute(FileLongName, 'ImageDescription') as ImageDescription,     Attribute(FileLongName,
'Make') as Make,     Attribute(FileLongName, 'Model') as Model,     Attribute(FileLongName,
'StripOffsets') as StripOffsets,     Attribute(FileLongName, 'Orientation') as Orientation,

    //  examples: 1=TopLeft, 2=TopRight, 3=BottomRight, 4=BottomLeft,

    // 5=LeftTop, 6=RightTop, 7=RightBottom, 8=LeftBottom,      Attribute(FileLongName,
'SamplesPerPixel') as SamplesPerPixel,     Attribute(FileLongName, 'RowsPerStrip') as
RowsPerStrip,     Attribute(FileLongName, 'StripByteCounts') as StripByteCounts,     Attribute
(FileLongName, 'XResolution') as XResolution,     Attribute(FileLongName, 'YResolution') as
YResolution,     Attribute(FileLongName, 'PlanarConfiguration') as PlanarConfiguration,

    //  examples: 1=chunky format, 2=planar format,      Attribute(FileLongName,
'ResolutionUnit') as ResolutionUnit,

    //  examples: 1=none, 2=inches, 3=centimeters,      Attribute(FileLongName,
'TransferFunction') as TransferFunction,     Attribute(FileLongName, 'Software') as Software,
   Attribute(FileLongName, 'DateTime') as DateTime,     Attribute(FileLongName, 'Artist') as
Artist,     Attribute(FileLongName, 'HostComputer') as HostComputer,     Attribute
(FileLongName, 'WhitePoint') as WhitePoint,     Attribute(FileLongName,
'PrimaryChromaticities') as PrimaryChromaticities,     Attribute(FileLongName,
'YCbCrCoefficients') as YCbCrCoefficients,     Attribute(FileLongName, 'YCbCrSubSampling') as
YCbCrSubSampling,     Attribute(FileLongName, 'YCbCrPositioning') as YCbCrPositioning,

    //  examples: 1=centered, 2=co-sited,      Attribute(FileLongName, 'ReferenceBlackWhite')
as ReferenceBlackWhite,     Attribute(FileLongName, 'Rating') as Rating,     Attribute
```

```
(FileLongName, 'RatingPercent') as RatingPercent,     Attribute(FileLongName,
'ThumbnailFormat') as ThumbnailFormat,

    //  examples: 0=Raw Rgb, 1=Jpeg,      Attribute(FileLongName, 'Copyright') as Copyright,
  Attribute(FileLongName, 'ExposureTime') as ExposureTime,     Attribute(FileLongName,
'FNumber') as FNumber,     Attribute(FileLongName, 'ExposureProgram') as ExposureProgram,

    //  examples: 0=Not defined, 1=Manual, 2=Normal program, 3=Aperture priority, 4=Shutter
priority,

    // 5=Creative program, 6=Action program, 7=Portrait mode, 8=Landscape mode, 9=Bulb,
Attribute(FileLongName, 'ISOSpeedRatings') as ISOSpeedRatings,     Attribute(FileLongName,
'TimeZoneOffset') as TimeZoneOffset,     Attribute(FileLongName, 'SensitivityType') as
SensitivityType,

    //  examples: 0=Unknown, 1=Standard output sensitivity (SOS), 2=Recommended exposure index
(REI),

    // 3=ISO speed, 4=Standard output sensitivity (SOS) and Recommended exposure index (REI),

    //5=Standard output sensitivity (SOS) and ISO Speed, 6=Recommended exposure index (REI)
and ISO Speed,

    // 7=Standard output sensitivity (SOS) and Recommended exposure index (REI) and ISO speed,
     Attribute(FileLongName, 'ExifVersion') as ExifVersion,     Attribute(FileLongName,
'DateTimeOriginal') as DateTimeOriginal,     Attribute(FileLongName, 'DateTimeDigitized') as
DateTimeDigitized,     Attribute(FileLongName, 'ComponentsConfiguration') as
ComponentsConfiguration,

    //  examples: 1=Y, 2=Cb, 3=Cr, 4=R, 5=G, 6=B,      Attribute(FileLongName,
'CompressedBitsPerPixel') as CompressedBitsPerPixel,     Attribute(FileLongName,
'ShutterSpeedValue') as ShutterSpeedValue,     Attribute(FileLongName, 'ApertureValue') as
ApertureValue,     Attribute(FileLongName, 'BrightnessValue') as BrightnessValue, //
examples: -1=Unknown,      Attribute(FileLongName, 'ExposureBiasValue') as ExposureBiasValue,
   Attribute(FileLongName, 'MaxApertureValue') as MaxApertureValue,     Attribute
(FileLongName, 'SubjectDistance') as SubjectDistance,

    //  examples: 0=Unknown, -1=Infinity,      Attribute(FileLongName, 'MeteringMode') as
MeteringMode,

    //  examples: 0=Unknown, 1=Average, 2=CenterWeightedAverage, 3=Spot,

    // 4=MultiSpot, 5=Pattern, 6=Partial, 255=Other,      Attribute(FileLongName,
'LightSource') as LightSource,

    //  examples: 0=Unknown, 1=Daylight, 2=Fluorescent, 3=Tungsten, 4=Flash, 9=Fine weather,

    // 10=Cloudy weather, 11=Shade, 12=Daylight fluorescent,

    // 13=Day white fluorescent, 14=Cool white fluorescent,

    // 15=White fluorescent, 17=Standard light A, 18=Standard light B, 19=Standard light C,

    // 20=D55, 21=D65, 22=D75, 23=D50, 24=ISO studio tungsten, 255=other light source,
Attribute(FileLongName, 'Flash') as Flash,     Attribute(FileLongName, 'FocalLength') as
```

```
FocalLength,     Attribute(FileLongName, 'SubjectArea') as SubjectArea,     Attribute
(FileLongName, 'MakerNote') as MakerNote,     Attribute(FileLongName, 'UserComment') as
UserComment,     Attribute(FileLongName, 'SubSecTime') as SubSecTime,

     Attribute(FileLongName, 'SubsecTimeOriginal') as SubsecTimeOriginal,     Attribute
(FileLongName, 'SubsecTimeDigitized') as SubsecTimeDigitized,     Attribute(FileLongName,
'XPTitle') as XPTitle,     Attribute(FileLongName, 'XPComment') as XPComment,

     Attribute(FileLongName, 'XPAuthor') as XPAuthor,     Attribute(FileLongName,
'XPKeywords') as XPKeywords,     Attribute(FileLongName, 'XPSubject') as XPSubject,
Attribute(FileLongName, 'FlashpixVersion') as FlashpixVersion,     Attribute(FileLongName,
'ColorSpace') as ColorSpace, //  examples: 1=sRGB, 65535=Uncalibrated,      Attribute
(FileLongName, 'PixelXDimension') as PixelXDimension,     Attribute(FileLongName,
'PixelYDimension') as PixelYDimension,     Attribute(FileLongName, 'RelatedSoundFile') as
RelatedSoundFile,

     Attribute(FileLongName, 'FocalPlaneXResolution') as FocalPlaneXResolution,     Attribute
(FileLongName, 'FocalPlaneYResolution') as FocalPlaneYResolution,     Attribute(FileLongName,
'FocalPlaneResolutionUnit') as FocalPlaneResolutionUnit,

    //  examples: 1=None, 2=Inch, 3=Centimeter,      Attribute(FileLongName, 'ExposureIndex')
as ExposureIndex,     Attribute(FileLongName, 'SensingMethod') as SensingMethod,

    //  examples: 1=Not defined, 2=One-chip color area sensor, 3=Two-chip color area sensor,

    // 4=Three-chip color area sensor, 5=Color sequential area sensor,

    // 7=Trilinear sensor, 8=Color sequential linear sensor,      Attribute(FileLongName,
'FileSource') as FileSource,

    //  examples: 0=Other, 1=Scanner of transparent type,

    // 2=Scanner of reflex type, 3=Digital still camera,      Attribute(FileLongName,
'SceneType') as SceneType,

    //  examples: 1=A directly photographed image,      Attribute(FileLongName, 'CFAPattern')
as CFAPattern,     Attribute(FileLongName, 'CustomRendered') as CustomRendered,

    //  examples: 0=Normal process, 1=Custom process,      Attribute(FileLongName,
'ExposureMode') as ExposureMode,

    //  examples: 0=Auto exposure, 1=Manual exposure, 2=Auto bracket,      Attribute
(FileLongName, 'WhiteBalance') as WhiteBalance,

    //  examples: 0=Auto white balance, 1=Manual white balance,      Attribute(FileLongName,
'DigitalZoomRatio') as DigitalZoomRatio,     Attribute(FileLongName, 'FocalLengthIn35mmFilm')
as FocalLengthIn35mmFilm,     Attribute(FileLongName, 'SceneCaptureType') as SceneCaptureType,

    //  examples: 0=Standard, 1=Landscape, 2=Portrait, 3=Night scene,      Attribute
(FileLongName, 'GainControl') as GainControl,

    //  examples: 0=None, 1=Low gain up, 2=High gain up, 3=Low gain down, 4=High gain down,
  Attribute(FileLongName, 'Contrast') as Contrast,
```

```
    //  examples: 0=Normal, 1=Soft, 2=Hard,      Attribute(FileLongName, 'Saturation') as
Saturation,

    //  examples: 0=Normal, 1=Low saturation, 2=High saturation,      Attribute(FileLongName,
'Sharpness') as Sharpness,

    //  examples: 0=Normal, 1=Soft, 2=Hard,      Attribute(FileLongName,
'SubjectDistanceRange') as SubjectDistanceRange,

    //  examples: 0=Unknown, 1=Macro, 2=Close view, 3=Distant view,      Attribute
(FileLongName, 'ImageUniqueID') as ImageUniqueID,     Attribute(FileLongName,
'BodySerialNumber') as BodySerialNumber,     Attribute(FileLongName, 'CMNT_GAMMA') as CMNT_
GAMMA,     Attribute(FileLongName, 'PrintImageMatching') as PrintImageMatching,     Attribute
(FileLongName, 'OffsetSchema') as OffsetSchema,

     // ************   Interoperability Attributes   ************     Attribute(FileLongName,
'InteroperabilityIndex') as InteroperabilityIndex,     Attribute(FileLongName,
'InteroperabilityVersion') as InteroperabilityVersion,     Attribute(FileLongName,
'InteroperabilityRelatedImageFileFormat') as InteroperabilityRelatedImageFileFormat,
Attribute(FileLongName, 'InteroperabilityRelatedImageWidth') as
InteroperabilityRelatedImageWidth,     Attribute(FileLongName,
'InteroperabilityRelatedImageLength') as InteroperabilityRelatedImageLength,     Attribute
(FileLongName, 'InteroperabilityColorSpace') as InteroperabilityColorSpace,

    //  examples: 1=sRGB, 65535=Uncalibrated,      Attribute(FileLongName,
'InteroperabilityPrintImageMatching') as InteroperabilityPrintImageMatching,     //
************   GPS Attributes   ************     Attribute(FileLongName, 'GPSVersionID') as
GPSVersionID,     Attribute(FileLongName, 'GPSLatitudeRef') as GPSLatitudeRef,     Attribute
(FileLongName, 'GPSLatitude') as GPSLatitude,     Attribute(FileLongName, 'GPSLongitudeRef')
as GPSLongitudeRef,     Attribute(FileLongName, 'GPSLongitude') as GPSLongitude,     Attribute
(FileLongName, 'GPSAltitudeRef') as GPSAltitudeRef,

    //  examples: 0=Above sea level, 1=Below sea level,      Attribute(FileLongName,
'GPSAltitude') as GPSAltitude,     Attribute(FileLongName, 'GPSTimeStamp') as GPSTimeStamp,
 Attribute(FileLongName, 'GPSSatellites') as GPSSatellites,     Attribute(FileLongName,
'GPSStatus') as GPSStatus,     Attribute(FileLongName, 'GPSMeasureMode') as GPSMeasureMode,
 Attribute(FileLongName, 'GPSDOP') as GPSDOP,     Attribute(FileLongName, 'GPSSpeedRef') as
GPSSpeedRef,

     Attribute(FileLongName, 'GPSSpeed') as GPSSpeed,     Attribute(FileLongName,
'GPSTrackRef') as GPSTrackRef,     Attribute(FileLongName, 'GPSTrack') as GPSTrack,
Attribute(FileLongName, 'GPSImgDirectionRef') as GPSImgDirectionRef,     Attribute
(FileLongName, 'GPSImgDirection') as GPSImgDirection,     Attribute(FileLongName,
'GPSMapDatum') as GPSMapDatum,     Attribute(FileLongName, 'GPSDestLatitudeRef') as
GPSDestLatitudeRef,

     Attribute(FileLongName, 'GPSDestLatitude') as GPSDestLatitude,     Attribute
(FileLongName, 'GPSDestLongitudeRef') as GPSDestLongitudeRef,     Attribute(FileLongName,
'GPSDestLongitude') as GPSDestLongitude,     Attribute(FileLongName, 'GPSDestBearingRef') as
GPSDestBearingRef,     Attribute(FileLongName, 'GPSDestBearing') as GPSDestBearing,
Attribute(FileLongName, 'GPSDestDistanceRef') as GPSDestDistanceRef,

     Attribute(FileLongName, 'GPSDestDistance') as GPSDestDistance,     Attribute
(FileLongName, 'GPSProcessingMethod') as GPSProcessingMethod,     Attribute(FileLongName,
```

```
'GPSAreaInformation') as GPSAreaInformation,     Attribute(FileLongName, 'GPSDateStamp') as
GPSDateStamp,     Attribute(FileLongName, 'GPSDifferential') as GPSDifferential;

    //  examples: 0=No correction, 1=Differential correction,  LOAD @1:n as FileLongName
Inline "$(vFoundFile)" (fix, no labels); Next vFoundFile Next vExt
```

Example 3: Windows-mediafiler

Det här skriptet läser alla möjliga WMA/WMV ASF-metataggar i mappen *MyMusic*.

```
// Script to read WMA/WMV ASF meta tags for each vExt in 'asf', 'wma', 'wmv' for each
vFoundFile in filelist( GetFolderPath('MyMusic') & '\*.'& vExt )

 FileList: LOAD FileLongName,     subfield(FileLongName,'\',-1) as FileShortName,     num
(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,     FileTime(FileLongName) as
FileTime,      Attribute(FileLongName, 'Title') as Title,     Attribute(FileLongName,
'Author') as Author,     Attribute(FileLongName, 'Copyright') as Copyright,     Attribute
(FileLongName, 'Description') as Description,

     Attribute(FileLongName, 'Rating') as Rating,     Attribute(FileLongName, 'PlayDuration')
as PlayDuration,     Attribute(FileLongName, 'MaximumBitrate') as MaximumBitrate,
Attribute(FileLongName, 'WMFSDKVersion') as WMFSDKVersion,     Attribute(FileLongName,
'WMFSDKNeeded') as WMFSDKNeeded,     Attribute(FileLongName, 'IsVBR') as IsVBR,     Attribute
(FileLongName, 'ASFLeakyBucketPairs') as ASFLeakyBucketPairs,

     Attribute(FileLongName, 'PeakValue') as PeakValue,     Attribute(FileLongName,
'AverageLevel') as AverageLevel; LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no
labels); Next vFoundFile Next vExt
```

Example 4: PNG

Det här skriptet läser alla möjliga PNG-metataggar i mappen *MyPictures*.

```
// Script to read PNG meta tags for each vExt in 'png' for each vFoundFile in filelist(
GetFolderPath('MyPictures') & '\*.'& vExt )

 FileList: LOAD FileLongName,     subfield(FileLongName,'\',-1) as FileShortName,     num
(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,     FileTime(FileLongName) as
FileTime,      Attribute(FileLongName, 'Comment') as Comment,

     Attribute(FileLongName, 'Creation Time') as Creation_Time,     Attribute(FileLongName,
'Source') as Source,     Attribute(FileLongName, 'Title') as Title,     Attribute
(FileLongName, 'Software') as Software,     Attribute(FileLongName, 'Author') as Author,
Attribute(FileLongName, 'Description') as Description,

     Attribute(FileLongName, 'Copyright') as Copyright; LOAD @1:n as FileLongName Inline
"$(vFoundFile)" (fix, no labels); Next vFoundFile Next vExt
```

ConnectString

ConnectString()-funktionen returnerar namnet på den aktiva datakopplingen för ODBC- eller OLE DB -kopplingar. Funktionen returnerar en tom sträng om ingen **connect**-sats har exekverats eller efter en **disconnect**-sats.

Syntax:

```
ConnectString()
```

Exempel och resultat:

Skriptexempel

Exempel	Resultat

`LIB CONNECT TO 'Tutorial ODBC';`
`ConnectString:`
`Load ConnectString() as ConnectString`
`AutoGenerate 1;`
	Returnerar Tutorial ODBC i fältet ConnectString.
	
	I det här exemplet antas att du har en tillgänglig datakälla som heter Tutorial ODBC.

FileBaseName

FileBaseName-funktionen returnerar en textsträng som anger namnet på den tabellfil som håller på att läsas in, dock utan sökväg och filtillägg.

Syntax:

```
FileBaseName()
```

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`LOAD *, filebasename( ) as X from C:\UserFiles\abc.txt`	Returnerar 'abc' i fält X i varje post som läses in.

FileDir

FileDir-funktionen returnerar en textsträng som anger sökvägen till katalogen som innehåller den tabellfil som håller på att läsas in.

Syntax:

```
FileDir()
```

Funktionen har enbart stöd för mappdatakopplingar i standardläget.

Exempel och resultat:

Skriptexempel

Exempel	Resultat

`Load *, filedir( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Returnerar 'C:\UserFiles' i fält X i varje post som läses in.

FileExtension

FileExtension-funktionen returnerar en textsträng som anger filtillägget till den tabellfil som håller på att läsas in.

Syntax:

```
FileExtension()
```

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`LOAD *, FileExtension( ) as X from C:\UserFiles\abc.txt`	Returnerar 'txt' i fält X i varje post som läses in.

FileName

FileName-funktionen returnerar en textsträng som anger namnet på den tabellfil som håller på att läsas in, utan sökväg men med filtillägget.

Syntax:

```
FileName()
```

Exempel och resultat:

Skriptexempel

Exempel	Resultat

`LOAD *, FileName( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Returnerar 'abc.txt' i fält X i varje post som läses in.

FilePath

FilePath-funktionen returnerar en textsträng som anger den kompletta sökvägen till den tabellfil som håller på att läsas in.

Syntax:

```
FilePath()
```

Funktionen har enbart stöd för mappdatakopplingar i standardläget.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`Load *, FilePath( ) as X from C:\UserFiles\abc.txt`	Returnerar 'C:\UserFiles\abc.txt' i fält X i varje post som läses in.

FileSize

FileSize-funktionen returnerar ett heltal som anger storleken i byte för filen filename eller, om inget filename angetts, för den tabellfil som håller på att läsas in.

Syntax:
FileSize([filename]**)**

Argument:

Argument

Argument	Beskrivning
filename	Namnet på en fil, vid behov inklusive sökväg, som en mapp eller webbfildatakoppling. Om du inte anger ett filnamn används den tabellfil som för närvarande läses. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense-appens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`LOAD *, FileSize( ) as X from abc.txt;`	Returnerar den angivna filens (abc.txt) storlek som ett heltal i fält X i varje post som läses in.
`FileSize( 'lib://DataFiles/xyz.xls' )`	Returnerar storleken på filen xyz.xls.

FileTime

FileTime-funktionen returnerar en tidsmarkör för datum och tid för den senaste ändringen av filen filename. Om inget filename har angetts, använder funktionen den senast inlästa tabellfilen.

Syntax:

```
FileTime([ filename ])
```

Argument:

Argument

Argument	Beskrivning
filename	Namnet på en fil, vid behov inklusive sökväg, som en mapp eller webbfildatakoppling. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense-appens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`LOAD *, FileTime( ) as X from abc.txt;`	Returnerar datum och tid för den senaste ändringen av filen (abc.txt) som tidsmarkering i fält X i varje post som läses in.
FileTime('xyz.xls')	Returnerar datum och tid för den senaste ändringen av filen xyz.xls.

GetFolderPath

Funktionen **GetFolderPath** returnerar värdet av funktionen Microsoft Windows *SHGetFolderPath*. Den här funktionen tar som indata namnet på en Microsoft Windows -mapp och returnerar den fullständiga sökvägen till mappen.

Funktionen stöds inte i standardläget.

Syntax:

```
GetFolderPath(foldername)
```

Argument:

Argument

Argument	Beskrivning
foldername	Namnet på mappen Microsoft Windows. Mappnamnet får inte innehålla några blanksteg. Eventuellt blanksteg i mappnamnet som syns i Windows Explorer ska tas bort från mappnamnet. Exempel: *MyMusic* *MyDocuments*

Exempel och resultat:

Målet med det här exemplet är att få fram sökvägarna till Microsoft Windows följande *MyMusic*, *MyPictures* och *Windows*-mappar: Lägg till exempelskriptet i appen och ladda om den.

```
LOAD   GetFolderPath('MyMusic') as MyMusic,  GetFolderPath('MyPictures') as MyPictures,
GetFolderPath('Windows') as Windows AutoGenerate 1;
```

När appen laddas om läggs fälten *MyMusic*, *MyPictures* och *Windows* till i datamodellen. Varje fält innehåller sökvägen till den mapp som har definierats i indata. Exempel:

- *C:\Users\smu\Music* for the folder *MyMusic*
- *C:\Users\smu\Pictures* for the folder *MyPictures*
- *C:\Windows* for the folder *Windows*

QvdCreateTime

Denna skriptfunktion returnerar XML-huvudets tidsstämpel från en QVD-fil, om någon sådan finns, annars returneras NULL.

Syntax:

```
QvdCreateTime(filename)
```

Argument:

Argument

Argument	Beskrivning
filename	Namnet på en QVD-fil, vid behov inklusive sökväg, exempelvis en mapp eller webbdatakoppling. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense-appens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*

Exempel:

```
QvdCreateTime('MyFile.qvd')

QvdCreateTime('C:\MyDir\MyFile.qvd')

QvdCreateTime('lib://DataFiles/MyFile.qvd')
```

QvdFieldName

Denna skriptfunktion returnerar namnet på fält nummer **fieldno** i en QVD-fil. Om fältet inte finns returneras NULL.

Syntax:

```
QvdFieldName(filename , fieldno)
```

Argument:

Argument

Argument	Beskrivning
filename	Namnet på en QVD-fil, vid behov inklusive sökväg, exempelvis en mapp eller webbdatakoppling. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense-appens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*
fieldno	Fältnumret i den tabell som finns i QVD-filen.

Exempel:

```
QvdFieldName ('MyFile.qvd', 5)

QvdFieldName ('C:\MyDir\MyFile.qvd', 5)

QvdFieldName ('lib://DataFiles/MyFile.qvd', 5)
```

Alla tre exemplen returnerar namnet på det femte fältet i tabellen som finns i QVD-filen.

QvdNoOfFields

Denna skriptfunktion returnerar antalet fält i en QVD-fil.

Syntax:

```
QvdNoOfFields(filename)
```

Argument:

Argument

Argument	Beskrivning
filename	Namnet på en QVD-fil, vid behov inklusive sökväg, exempelvis en mapp eller webbdatakoppling. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense-appens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*

Exempel:

```
QvdNoOfFields ('MyFile.qvd')

QvdNoOfFields ('C:\MyDir\MyFile.qvd')

QvdNoOfFields ('lib://DataFiles/MyFile.qvd')
```

QvdNoOfRecords

Exempel: Denna skriptfunktion returnerar aktuellt antal poster i en QVD-fil.

Syntax:

```
QvdNoOfRecords(filename)
```

Argument:

Argument

Argument	Beskrivning
filename	Namnet på en QVD-fil, vid behov inklusive sökväg, exempelvis en mapp eller webbdatakoppling. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense-appens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*

Exempel:

```
QvdNoOfRecords ('MyFile.qvd')

QvdNoOfRecords ('C:\MyDir\MyFile.qvd')

QvdNoOfRecords ('lib://DataFiles/MyFile.qvd')
```

QvdTableName

Denna skriptfunktion returnerar namnet på tabellen i en QVD-fil.

Syntax:

```
QvdTableName(filename)
```

Argument:

Argument

Argument	Beskrivning
filename	Namnet på en QVD-fil, vid behov inklusive sökväg, exempelvis en mapp eller webbdatakoppling. **Exempel:** *'lib://Table Files/'* I det bakåtkompatibla skriptläget stöds även följande sökvägsformat: • absolut **Exempel:** *c:\data* • relativ till Qlik Sense-appens arbetskatalog. **Exempel:** *data* • URL-adress (HTTP eller FTP), som pekar till en plats på internet eller ett intranät. **Exempel:** *http://www.qlik.com*

Exempel:

```
QvdTableName ('MyFile.qvd')
QvdTableName ('C:\MyDir\MyFile.qvd')
QvdTableName ('lib://data\MyFile.qvd')
```

5.11 Finansiella funktioner

Finansiella funktioner kan användas i dataladdningsskriptet och diagramuttryck för att beräkna betalningar och räntesatser.
I alla parametrar betecknar negativa tal pengar som man betalar ut. Positiva tal betecknar pengar som erhålls.
I listan visas de parametrar som används i de finansiella funktionerna (förutom dem som börjar med **range**-).

*För alla finansiella funktioner är det mycket viktigt att vara konsekvent vid angivande av enheter för **rate** och **nper**. Om man gör en månatlig avbetalning på ett femårigt lån med en årlig räntesats på 6%, bör man använda 0.005 (6%/12) för **rate** och 60 (5*12) för **nper**. Om man gör en årlig avbetalning på samma lån, bör man använda 6% för **rate** och 5 för **nper**.*

Finansiella funktioner – en översikt

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

FV

Denna funktion returnerar det framtida värdet av en investering baserat på periodiska, återkommande inbetalningar och en enkel årsränta.

```
FV (rate, nper, pmt [ ,pv [ , type ] ])
```

nPer

Denna funktion returnerar antalet perioder för en investering baserad på periodiska, konstanta inbetalningar och en konstant räntesats.

```
nPer (rate, pmt, pv [ ,fv [ , type ] ])
```

Pmt

Denna funktion returnerar inbetalningsbeloppet för ett lån baserat på periodiska, konstanta inbetalningar och en konstant räntesats. Den kan inte ändras under annuitetens löptid. En betalning anges som ett negativt tal, till exempel -20.

```
Pmt (rate, nper, pv [ ,fv [ , type ] ] )
```

PV

Denna funktion returnerar det aktuella värdet av en investering.

```
PV (rate, nper, pmt [ ,fv [ , type ] ])
```

Rate

Denna funktion returnerar räntesatsen per period i en annuitet. Resultatet anges i ett fördefinierat talformat: **Fix**, två decimaler och %.

```
Rate (nper, pmt , pv [ ,fv [ , type ] ])
```

BlackAndSchole

Black and Scholes-modellen är en matematisk modell för finansmarknadsanalyser. Formeln beräknar det teoretiska värdet av en option. I Qlik Sense returnerar **BlackAndSchole**-funktionen värdet enligt den omodifierade Black and Scholes-formeln (europeisk optionsmodell).

```
BlackAndSchole(strike , time_left , underlying_price , vol , risk_free_rate ,
type)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
strike	Aktiens framtida inköpspris.
time_left	Antalet återstående perioder.
underlying_ price	Aktiens aktuella värde.
vol	Volatiliteten (för aktiekurser) uttryckt som procentandel i decimalform, per tidsperiod.
risk_free_rate	Den riskfria räntan uttryckt som procentandel i decimalform, per tidsperiod.
call_or_put	Typ av option: 'c', 'call' eller vilket numeriskt värde som helst, utom noll, för köpoptioner 'p', 'put' eller 0 för säljoptioner.

Begränsningar:

Värdet för strike, time_left och underlying_price måste vara >0.

Värdet för vol och risk_free_rate måste vara: <0 or >0.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`BlackAndSchole(130, 4, 68.5, 0.4, 0.04, 'call')` Detta beräknar det teoretiska priset på en option som köps på 4 år till ett värde av 130 per aktie, vilket idag motsvarar 68,5. Formeln använder volatiliteten 0,4 (40 %) per år och en riskfri ränta på 0,04 (4 %).	Returnerar 11,245

FV

Denna funktion returnerar det framtida värdet av en investering baserat på periodiska, återkommande inbetalningar och en enkel årsränta.

Syntax:

```
FV(rate, nper, pmt [ ,pv [ , type ] ])
```

Returnerad datatyp: numeriska. Resultatet anges i ett fördefinierat talformat (valuta). .

Argument:

Argument

Argument	Beskrivning
rate	Räntesatsen per period.
nper	Det totala antalet inbetalningsperioder i en annuitet.
pmt	Den inbetalning som görs varje period. Den kan inte ändras under annuitetens löptid. En betalning anges som ett negativt tal, till exempel -20.
pv	Det aktuella värdet (present value), eller den klumpsumma som en serie framtida inbetalningar för närvarande är värda. Om **pv** utelämnas, förutsätts 0 (noll).
type	Ska sättas till 0 om inbetalningarna ska göras i slutet av perioden och 1 om inbetalningarna ska göras i början av perioden. Om **type** utelämnas, förutsätts 0.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
Du betalar av en ny hushållsmaskin med 36 månadsavbetalningar på 20 $. Räntesatsen är 6 % per år. Räkningen kommer i slutet av månaden. Hur mycket har du investerat totalt när den sista räkningen betalats? `FV(0.005,36,-20)`	Returnerar $786.72

nPer

Denna funktion returnerar antalet perioder för en investering baserad på periodiska, konstanta inbetalningar och en konstant räntesats.

Syntax:

```
nPer(rate, pmt, pv [ ,fv [ , type ] ])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
rate	Räntesatsen per period.
nper	Det totala antalet inbetalningsperioder i en annuitet.

Argument	Beskrivning
pmt	Den inbetalning som görs varje period. Den kan inte ändras under annuitetens löptid. En betalning anges som ett negativt tal, till exempel -20.
pv	Det aktuella värdet (present value), eller den klumpsumma som en serie framtida inbetalningar för närvarande är värda. Om **pv** utelämnas, förutsätts 0 (noll).
fv	Det framtida värdet, eller den kassabehållning man vill uppnå efter att den sista räkningen betalats. Om **fv** utelämnas, förutsätts 0.
type	Ska sättas till 0 om inbetalningarna ska göras i slutet av perioden och 1 om inbetalningarna ska göras i början av perioden. Om **type** utelämnas, förutsätts 0.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
Du vill sälja en hushållsmaskin med månadsavbetalningar på 200 kronor. Räntesatsen är 6 % per år. Räkningen kommer i slutet av månaden. Hur många perioder krävs för att värdet på pengarna du erhållit efter att den sista räkningen betalats ska vara 800 $? `nPer(0.005,-20,0,800)`	Returnerar 36,56

Pmt

Denna funktion returnerar inbetalningsbeloppet för ett lån baserat på periodiska, konstanta inbetalningar och en konstant räntesats. Den kan inte ändras under annuitetens löptid. En betalning anges som ett negativt tal, till exempel -20.

```
Pmt(rate, nper, pv [ ,fv [ , type ] ] )
```

Returnerad datatyp: numeriska. Resultatet anges i ett fördefinierat talformat (valuta). .

För att räkna ut det totala beloppet som betalats in under lånets löptid multiplicerar du det returnerade **pmt**-värdet med **nper**.

Argument:

Argument

Argument	Beskrivning
rate	Räntesatsen per period.
nper	Det totala antalet inbetalningsperioder i en annuitet.
pv	Det aktuella värdet (present value), eller den klumpsumma som en serie framtida inbetalningar för närvarande är värda. Om **pv** utelämnas, förutsätts 0 (noll).

Argument	Beskrivning
fv	Det framtida värdet, eller den kassabehållning man vill uppnå efter att den sista räkningen betalats. Om **fv** utelämnas, förutsätts 0.
type	Ska sättas till 0 om inbetalningarna ska göras i slutet av perioden och 1 om inbetalningarna ska göras i början av perioden. Om **type** utelämnas, förutsätts 0.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
Följande formel returnerar det månatliga inbetalningsbeloppet för ett lån på 20 000 kronor som ska vara avbetalat på 8 månader och där räntesatsen är 10%: `Pmt(0.1/12,8,20000)`	Returnerar -$2,594.66
Om inbetalningarna för samma lån ska göras i början av perioden blir inbetalningsbeloppet följande: `Pmt(0.1/12,8,20000,0,1)`	Returnerar -$2,573.21

PV

Denna funktion returnerar det aktuella värdet av en investering.

```
PV(rate, nper, pmt [ ,fv [ , type ] ])
```

Returnerad datatyp: numeriska. Resultatet anges i ett fördefinierat talformat (valuta). .

Det aktuella värdet är det totala belopp som en serie framtida inbetalningar för närvarande är värda. Om man exempelvis lånar pengar, är lånebeloppet det aktuella värdet för den som lånar ut.

Argument:

Argument

Argument	Beskrivning
rate	Räntesatsen per period.
nper	Det totala antalet inbetalningsperioder i en annuitet.
pmt	Den inbetalning som görs varje period. Den kan inte ändras under annuitetens löptid. En betalning anges som ett negativt tal, till exempel -20.
fv	Det framtida värdet, eller den kassabehållning man vill uppnå efter att den sista räkningen betalats. Om **fv** utelämnas, förutsätts 0.
type	Ska sättas till 0 om inbetalningarna ska göras i slutet av perioden och 1 om inbetalningarna ska göras i början av perioden. Om **type** utelämnas, förutsätts 0.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
Vilket är det aktuella värdet av en skuld som du betalar av med 100 $ i slutet av varje månad under en femårsperiod, om räntan är sju procent? `PV(0.07/12,12*5,-100,0,0)`	Returnerar $5,050.20

Rate

Denna funktion returnerar räntesatsen per period i en annuitet. Resultatet anges i ett fördefinierat talformat: **Fix**, två decimaler och %.

Syntax:

```
Rate(nper, pmt , pv [ ,fv [ , type ] ])
```

Returnerad datatyp: numeriska.

rate beräknas ut med hjälp av iteration och kan ha noll eller fler lösningar. Om de successiva resultaten av **rate** inte konvergerar, returneras NULL.

Argument:

Argument

Argument	Beskrivning
nper	Det totala antalet inbetalningsperioder i en annuitet.
pmt	Den inbetalning som görs varje period. Den kan inte ändras under annuitetens löptid. En betalning anges som ett negativt tal, till exempel -20.
pv	Det aktuella värdet (present value), eller den klumpsumma som en serie framtida inbetalningar för närvarande är värda. Om **pv** utelämnas, förutsätts 0 (noll).
fv	Det framtida värdet, eller den kassabehållning man vill uppnå efter att den sista räkningen betalats. Om **fv** utelämnas, förutsätts 0.
type	Ska sättas till 0 om inbetalningarna ska göras i slutet av perioden och 1 om inbetalningarna ska göras i början av perioden. Om **type** utelämnas, förutsätts 0.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
Vilken är räntesatsen på ett femårigt annuitetslån på 10 000 SEK med månatliga inbetalningar på 300 SEK? `Rate(60,-300,10000)`	Returnerar 2.00%

5.12 Formateringsfunktioner

Formateringsfunktionerna bestämmer visningsformat för indata i form av numeriska fält eller uttryck. Beroende på datatyp kan du ange tecken för decimalavgränsare, tusendelsavgränsare, o.s.v.

Funktionerna returnerar ett dualt värde med både strängen och det numeriska värdet, men det kan också ses som att de gör en konvertering från tal till sträng. **Dual()** är ett specialfall, men de andra formateringsfunktionerna tar det numeriska värdet av indatauttrycken och genererar en sträng som representerar numret.

Tolkningsfunktionerna däremot gör det motsatta: de tar stränguttryck och utvärderar dem som tal och specificerar sedan format för det resulterande talet.

Funktionerna kan användas både i dataladdningsskript och diagramuttryck.

Alla numeriska värden anges med en decimalpunkt som decimalavgränsare.

Formateringsfunktioner - en översikt

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

ApplyCodepage

ApplyCodepage() tillämpar en annan teckenuppsättning för kodsidan på fältet eller texten som angetts i uttrycket. Argumentet **codepage** måste skrivas i talformat.

```
ApplyCodepage (text, codepage)
```

Date

Date() formaterar ett uttryck som ett datum med hjälp av formatet som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, eller i en formatsträng, om så är tillämpligt.

```
Date (number[, format])
```

Dual

Dual() kombinerar ett tal och en sträng till en enda post, så att talåtergivningen av posten kan användas för sortering och beräkning, samtidigt som strängvärdet kan användas för visning.

```
Dual (text, number)
```

Interval

Interval() formaterar ett tal som ett tidsintervall med hjälp av formatet som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, eller i en formatsträng, om så är tillämpligt.

```
Interval (number[, format])
```

Money

Money() formaterar ett uttryck numeriskt som ett valutavärde i det format som har ställts in i systemvariablerna i dataladdningsskriptet, eller i operativsystemet, om inte en formatsträng finns. Den formaterar även valbara decimal- och tusentalsavgränsare.

```
Money (number[, format[, dec_sep [, thou_sep]]])
```

Num

Num() formaterar ett tal, vilket innebär att det numeriska värdet i indata konverteras och visas med det format som specificerats i den andra parametern. Om den andra parametern utelämnas använder funktionen de decimal- och tusentalsavgränsare som anges i dataladdningsskriptet. Egna symboler för decimaler och tusentalsseparatorer är valbara parametrar.

```
Num (number[, format[, dec_sep [, thou_sep]]])
```

Time

Time() formaterar ett uttryck som ett tidsvärde, i det tidsformat som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns.

```
Time (number[, format])
```

Timestamp

TimeStamp() formaterar ett uttryck som ett datum och tid-värde, i det format för tidsmarkörer som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns.

```
Timestamp (number[, format])
```

Se även:

Tolkningsfunktioner (page 631)

ApplyCodepage

ApplyCodepage() tillämpar en annan teckenuppsättning för kodsidan på fältet eller texten som angetts i uttrycket. Argumentet **codepage** måste skrivas i talformat.

ApplyCodepage kan användas i diagramuttryck men är oftast en skriptfunktion i Skriptredigeraren. När du exempelvis läser in filer som kan ha sparats med andra teckenuppsättningar kan du tillämpa kodsidan med den teckenuppsättning som du behöver.

Syntax:

```
ApplyCodepage(text, codepage)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
text	Fält eller text som du vill använda en annan kodsida för, som fås av argumentet **codepage**.
codepage	Tal som representerar kodsidan som ska tillämpas på fältet eller uttrycket som fås med **text**.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`LOAD ApplyCodepage(ROWX,1253) as GreekProduct, ApplyCodepage (ROWY, 1255) as HebrewProduct, ApplyCodepage (ROWZ, 65001) as EnglishProduct; SQL SELECT ROWX, ROWY, ROWZ From Products;`	Vid inläsning från SQL kan källan ha en blandning av olika teckenuppsättningar: Kyrilliska, hebreiska med mera från UTF-8-formatet. Dessa måste läsas in rad för rad och olika kodsidor måste tillämpas för varje rad. **codepage**-värdet 1253 motsvarar Windows grekiska teckenuppsättning, värdet 1255 representerar hebreiska och värdet 65001 representerar vanliga latinska UTF-8-tecken.

Se även: *Teckenuppsättning (page 107)*

Date

Date() formaterar ett uttryck som ett datum med hjälp av formatet som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, eller i en formatsträng, om så är tillämpligt.

Syntax:

```
Date(number[, format])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
number	Talet som ska formateras.

Argument	Beskrivning
format	En sträng som beskriver formatet på den sträng som blir resultatet. Om ingen formatsträng anges används det datumformat som har definierats i systemvariablerna i dataladdningsskriptet eller operativsystemet.

Exempel och resultat:
Exemplen nedan förutsätter följande standardinställningar:

- Datuminställning 1: YY-MM-DD
- Datuminställning 2: M/D/YY

Exempel:

```
Date( A )
```
where A=35648

Resultattabell

Resultat	Inställning 1	Inställning 2
Sträng:	97-08-06	8/6/97
Tal:	35648	35648

Exempel:

```
Date( A, 'YY.MM.DD' )
```
where A=35648

Resultattabell

Resultat	Inställning 1	Inställning 2
Sträng:	97.08.06	97.08.06
Tal:	35648	35648

Exempel:

```
Date( A, 'DD.MM.YYYY' )
```
where A=35648.375

Resultattabell

Resultat	Inställning 1	Inställning 2
Sträng:	06.08.1997	06.08.1997
Tal:	35648.375	35648.375

Exempel:

```
Date( A, 'YY.MM.DD' )
```
where A=8/6/97

Resultattabell

Resultat	Inställning 1	Inställning 2
Sträng:	NULL (inget)	97.08.06
Tal:	NULL	35648

Dual

Dual() kombinerar ett tal och en sträng till en enda post, så att talåtergivningen av posten kan användas för sortering och beräkning, samtidigt som strängvärdet kan användas för visning.

Syntax:

```
Dual(text, number)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
text	Strängvärdet som ska användas i kombination med talargumentet.
number	Talet som ska användas i kombination med strängen i strängargumentet.

I Qlik Sense är alla fältvärden potentiellt duala värden. Detta innebär att fältvärdena både kan ha ett numeriskt värde och ett textvärde. Exempelvis kan ett datum ha det numeriska värdet 40908 och textrepresentationen '2011-12-31'.

När flera dataelement som laddas i ett fält har olika textsträngar men samma numeriska värde, delar de alla den första påträffade textsträngen.

***dual**-funktionen används vanligen tidigt i skriptet, innan andra data lästs in i det berörda fältet, för att skapa denna första strängrepresentation som kommer att visas i filterrutor.*

Exempel och resultat:

Skriptexempel

Exempel	Beskrivning
Lägg till följande exempel i skriptet och kör det. `Load dual ( NameDay,NumDay ) as DayOfWeek inline [ NameDay,NumDay Monday,0 Tuesday,1 Wednesday,2 Thursday,3 Friday,4 Saturday,5 Sunday,6 ];`	Fältet DayOfWeek kan användas i en visualisering, som en dimension till exempel. I en tabell med veckodagar sorteras de automatiskt i rätt nummerordning, istället för i alfabetisk ordning.
`Load Dual('Q' & Ceil (Month(Now())/3), Ceil (Month(Now())/3)) as Quarter AutoGenerate 1;`	Det här exemplet letar rätt på aktuellt kvartal. Det visas som Q1 när funktionen **Now()** körs under de första tre månaderna under året, Q2 för följande tre månader och så vidare. Om du använder sortering kommer dock fältet Quarter att fungera som sitt numeriska värde 1 till 4:
`Dual('Q' & Ceil(Month (Date)/3), Ceil(Month (Date)/3)) as Quarter`	Som i tidigare exempel skapas fältet Quarter med textvärdena 'Q1' till 'Q4', och tilldelas de numeriska värdena 1 till 4. För att kunna använda detta i skriptet måste värdena för Date laddas.
`Dual(WeekYear(Date) & '-W' & Week(Date), WeekStart(Date)) as YearWeek`	Det här exemplet skapar ett fält YearWeek med textvärden i formen '2012-W22' och tilldelar samtidigt ett numeriskt värde som motsvarar datumnumret för veckans första dag, exempelvis: 41057. För att kunna använda detta i skriptet måste värdena för Date laddas.

Interval

Interval() formaterar ett tal som ett tidsintervall med hjälp av formatet som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, eller i en formatsträng, om så är tillämpligt.

Intervall kan formateras som tid, dagar eller som en kombination av dagar, timmar, minuter, sekunder och bråkdels sekunder.

Syntax:

```
Interval(number[, format])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
number	Talet som ska formateras.
format	Sträng som beskriver hur den resulterande intervallsträngen ska formateras. Om den utelämnas används det korta datumformat, tidsformat och decimalavgränsare som har ställts in för operativsystemet.

Exempel och resultat:
Exemplen nedan förutsätter följande standardinställningar:

- Inställning för datumformat 1: YY-MM-DD
- Inställning för datumformat 2: hh:mm:ss
- Decimalavgränsare för tal: .

Resultattabell

Exempel	Sträng	Tal
`Interval( A )` där A=0,375	09:00:00	0.375
`Interval( A )` där A=1,375	33:00:00	1.375
`Interval( A, 'D hh:mm' )` där A=1,375	1 09:00	1.375
`Interval( A-B, 'D hh:mm' )` där A=97-08-06 09:00:00 och B=96-08-06 00:00:00	365 09:00	365.375

Money

Money() formaterar ett uttryck numeriskt som ett valutavärde i det format som har ställts in i systemvariablerna i dataladdningsskriptet, eller i operativsystemet, om inte en formatsträng finns. Den formaterar även valbara decimal- och tusentalsavgränsare.

Syntax:

```
Money(number[, format[, dec_sep[, thou_sep]]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
number	Talet som ska formateras.
format	Sträng som beskriver hur den resulterande valutasträngen ska formateras.
dec_sep	Sträng som specificerar decimalavgränsare.
thou_sep	Sträng som specificerar tusendelsavgränsare.

Om argumenten 2-4 utelämnas, används det valutaformat som definierats i operativsystemet.

Exempel och resultat:
Exemplen nedan förutsätter följande standardinställningar:

- MoneyFormat-inställning 1: kr ##0,00, MoneyThousandSep' '
- MoneyFormat-inställning 2: $ #,##0.00, MoneyThousandSep','

Exempel:

```
Money( A )
```
där A=35648

Resultattabell

Resultat	Inställning 1	Inställning 2
Sträng:	35 648,00 kr	$ 35,648.00
Tal:	35648.00	35648.00

Exempel:

```
Money( A, '#,##0 ¥', '.' , ',' )
```
där A=3564800

Resultattabell

Resultat	Inställning 1	Inställning 2
Sträng:	3,564,800 ¥	3,564,800 ¥
Tal:	3564800	3564800

Num

Num() formaterar ett tal, vilket innebär att det numeriska värdet i indata konverteras och visas med det format som specificerats i den andra parametern. Om den andra parametern utelämnas använder funktionen de decimal- och tusentalsavgränsare som anges i dataladdningsskriptet. Egna symboler för decimaler och tusentalsseparatorer är valbara parametrar.

Syntax:

```
Num(number[, format[, dec_sep [, thou_sep]]])
```

Returnerad datatyp: dual

Num-funktionen returnerar ett dualt värde med både strängvärdet och det numeriska värdet. Funktionen tar det numeriska värdet i indatauttrycket och genererar en sträng som representerar talet.

Argument:

Argument

Argument	Beskrivning
number	Talet som ska formateras.
format	Sträng som anger hur den resulterande strängen ska formateras. Om den utelämnas används de decimal- och tusentalsavgränsare som anges i dataladdningsskriptet.
dec_sep	Sträng som specificerar decimalavgränsare. Om den utelämnas används värdet för variabeln DecimalSep som ställts in i dataladdningsskriptet.
thou_sep	Sträng som specificerar tusendelsavgränsare. Om den utelämnas används värdet för variabeln ThousandSep som ställts in i dataladdningsskriptet.

Exempel: Diagramuttryck

Exempel:

Följande tabell visar resultaten när fältet A är lika med 35648,312.

Resultat

En	Resultat
Num(A)	35648,312 (beror på miljövariabler i skriptet)
Num(A, '0.0', '.')	35648.3
Num(A, '0,00', ',')	35648,31
Num(A, '#,##0.0', '.', ',')	35,648.3
Num(A, '# ##0', ',', ' ')	35 648

Exempel: Laddningsskript

Laddningsskript

Num kan användas i laddningsskriptet för att formatera ett tal, även om tusentals- och decimalavgränsare redan har ställts in i skriptet. I laddningsskriptet nedan ingår specifika tusentals- och decimalavgränsare, men *Num* används för att formatera data på olika sätt.

Gå till **Skriptredigeraren** och skapa ett nytt delavsnitt. Lägg sedan till exempelskriptet och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i din app för att se resultatet.

```
SET ThousandSep=','; SET DecimalSep='.'; Transactions: Load *, Num(transaction_amount) as [No
formatting], Num(transaction_amount,'0') as [0], Num(transaction_amount,'#,##0') as [#,##0],
Num(transaction_amount,'# ###,00') as [# ###,00], Num(transaction_amount,'# ###,00',',',' ')
as [# ###,00 , ',' , ' '], Num(transaction_amount,'#,###.00','.',',') as [#,###.00 , '.' ,
',], Num(transaction_amount,'$#,###.00') as [$#,###.00], ; Load * Inline [ transaction_id,
transaction_date, transaction_amount, transaction_quantity, discount, customer_id, size,
color_code 3750, 20180830, 12423.56, 23, 0,2038593, L, Red 3751, 20180907, 5356.31, 6, 0.1,
203521, m, orange 3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue 3753, 20180922, 1251, 7, 0,
3036491, l, Black 3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red 3756, 20180922, -59.18,
2, 0.333333333333333, 2038593, M, Blue 3757, 20180923, 3177.4, 21, .14, 203521, XL, Black ];
```

Qlik Sense-tabell som visar resultaten man får när *Num*-funktionen används på olika sätt i laddningsskriptet. Tabellens fjärde kolumn innehåller exempel på när formatering inte används korrekt.

Ingen formatering	0	#,##0	# ###,00	# ###,00 , '' , ''	#,###.00 , '.' , ''	$#,###.00
-59.18	-59	-59	-59###,00	-59,18	-59.18	$-59,18
15.75	16	16	16###,00	15,75	15.75	$15,75
1251	1251	1,251	1251###,00	1 251,00	1,251.00	$1,251.00
3177.4	3177	3,177	3177###,00	3 177,40	3,177.40	$3,177.40
5356.31	5356	5,356	5356###,00	5 356,31	5,356.31	$5,356.31
12423.56	12424	12,424	12424###,00	12 423,56	12,423.56	$12,423.56
21484.21	21484	21,484	21484###,00	21 484,21	21,484.21	$21,484.21

Exempel: Laddningsskript

Laddningsskript

Num kan användas i laddningsskript för att formatera ett tal som en procentandel.

Gå till **Skriptredigeraren** och skapa ett nytt delavsnitt. Lägg sedan till exempelskriptet och kör det. Lägg sedan till åtminstone de fält som listas i resultatkolumnen till ett ark i din app för att se resultatet.

```
SET ThousandSep=','; SET DecimalSep='.'; Transactions: Load *, Num(discount,'#,##0%') as
[Discount #,##0%] ; Load * Inline [ transaction_id, transaction_date, transaction_amount,
transaction_quantity, discount, customer_id, size, color_code 3750, 20180830, 12423.56, 23,
0,2038593, L, Red 3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange 3752, 20180916, 15.75, 1,
0.22, 5646471, S, blue 3753, 20180922, 1251, 7, 0, 3036491, l, Black 3754, 20180922, 21484.21,
```

```
1356, 75, 049681, xs, Red 3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue 3757,
20180923, 3177.4, 21, .14, 203521, XL, Black ];
```

Qlik Sense-tabell som visar resultaten man får när *Num*-funktionen används i laddningsskriptet för att formatera procentandelar.

Discount	Discount #,##0%
0.333333333333333	33%
0.22	22%
0	0%
.14	14%
0.1	10%
0	0%
75	7,500%

Time

Time() formaterar ett uttryck som ett tidsvärde, i det tidsformat som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns.

Syntax:

```
Time(number[, format])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
number	Talet som ska formateras.
format	Sträng som beskriver hur den resulterande tidsträngen ska formateras. Om den utelämnas används det korta datumformat, tidsformat och decimalavgränsare som har ställts in för operativsystemet.

Exempel och resultat:
Exemplen nedan förutsätter följande standardinställningar:

- Inställning för tidsformat 1: hh:mm:ss
- Inställning för tidsformat 2: hh.mm.ss

Exempel:

```
Time( A )
```
där A=0,375

Resultattabell

Resultat	Inställning 1	Inställning 2
Sträng:	09:00:00	09.00.00
Tal:	0.375	0.375

Exempel:

```
Time( A )
```
där A=35648,375

Resultattabell

Resultat	Inställning 1	Inställning 2
Sträng:	09:00:00	09.00.00
Tal:	35648.375	35648.375

Exempel:

```
Time( A, 'hh-mm' )
```
där A=0,99999

Resultattabell

Resultat	Inställning 1	Inställning 2
Sträng:	23-59	23-59
Tal:	0.99999	0.99999

Timestamp

TimeStamp() formaterar ett uttryck som ett datum och tid-värde, i det format för tidsmarkörer som har ställts in i systemvariablerna i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns.

Syntax:

```
Timestamp(number[, format])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
number	Talet som ska formateras.
format	Sträng som beskriver hur den resulterande tidsmarkörsträngen ska formateras. Om den utelämnas används det korta datumformat, tidsformat och decimalavgränsare som har ställts in för operativsystemet.

Exempel och resultat:
Exemplen nedan förutsätter följande standardinställningar:

- Inställning för TimeStampFormat 1: YY-MM-DD hh:mm:ss
- Inställning för TimeStampFormat 2: M/D/YY hh:mm:ss

Exempel:

```
Timestamp( A )
```
där A=35648,375

Resultattabell

Resultat	Inställning 1	Inställning 2
Sträng:	97-08-06 09:00:00	8/6/97 09:00:00
Tal:	35648.375	35648.375

Exempel:

```
Timestamp( A,'YYYY-MM-DD hh.mm')
```
där A=35648

Resultattabell

Resultat	Inställning 1	Inställning 2
Sträng:	1997-08-06 00.00	1997-08-06 00.00
Tal:	35648	35648

5.13 Allmänna numeriska funktioner

I dessa allmänna numeriska funktioner är argumenten uttryck där **x** ska tolkas som ett reellt tal. Alla funktioner kan användas både i dataladdningsskript och diagramuttryck.

Allmänna numeriska funktioner – en översikt

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

bitcount

BitCount() returnerar hur många bitar i den binära motsvarigheten till ett decimaltal som är inställda på 1. Denna funktion returnerar alltså antalet inställda bitar i **integer_number**, där **integer_number** tolkas som ett signerat 32-bitars heltal.

```
BitCount(integer_number)
```

div

Div() returnerar heltalsdelen av den aritmetiska divisionen av det första argumentet med det andra argumentet. Båda parametrarna tolkas som reella tal, de behöver alltså inte nödvändigtvis vara heltal.

```
Div (integer_number1, integer_number2)
```

fabs

Fabs() returnerar absolutvärdet för **x**. Resultatet blir ett positivt tal.

```
Fabs (x)
```

fact

Fact() returnerar faktorn av ett positivt heltal **x**.

```
Fact (x)
```

frac

Frac() returnerar decimaldelen av **x**.

```
Frac (x)
```

sign

Sign() returnerar 1, 0 eller -1 beroende på om **x** är ett positivt tal, 0 eller ett negativt tal.

```
Sign (x)
```

Kombinations- och permutationsfunktioner

combin

Combin() returnerar antalet kombinationer av **q**-element som kan väljas från en uppsättning **p**-element. Motsvaras av formeln: `Combin(p,q) = p! / q!(p-q)!` Ordningen som komponenterna väljs i är inte signifikant.

```
Combin (p, q)
```

permut

Permut() returnerar antalet permutationer av **q** element som kan väljas från en uppsättning med **p** element. Motsvaras av formeln: `Permut(p,q) = (p)! / (p - q)!` Ordningen som komponenterna väljs i är signifikant.

```
Permut (p, q)
```

Modulofunktioner

fmod

fmod() är en generaliserad modulofunktion som returnerar den återstående delen av heltalsdivisionen av det första argumentet (dividenden) med det andra argumentet (divisorn). Resultatet blir ett reellt tal. Båda argumenten tolkas som reella tal, de behöver alltså inte nödvändigtvis vara heltal.

```
Fmod (a, b)
```

mod

Mod() är en matematisk modulofunktion som returnerar den icke-negativa återstoden av en heltalsdivison. Det första argumentet är dividenden, det andra argumentet är divisorn. Båda argumenten måste vara heltalsvärden.

```
Mod (integer_number1, integer_number2)
```

Paritetsfunktioner

even

Even() returnerar True (-1) om **integer_number** är ett jämnt heltal eller noll. Den returnerar False (0) om **integer_number** är ett udda heltal och NULL om **integer_number** inte är ett heltal.

```
Even (integer_number)
```

odd

Odd() returnerar True (-1) om **integer_number** är ett udda heltal eller noll. Det returnerar False (0) om **integer_number** är ett jämnt heltal och NULL om **integer_number** inte är ett heltal.

```
Odd (integer_number)
```

Avrundningsfunktioner

ceil

Ceil() rundar av ett tal uppåt till närmaste multipel av det **step** som förskjuts av **offset** -talet.

```
Ceil (x[, step[, offset]])
```

floor

Floor() rundar av ett tal nedåt till närmaste multipel av det **step** som förskjuts av **offset** -talet.

```
Floor (x[, step[, offset]])
```

round

Round() returnerar resultatet av avrundning av ett tal uppåt eller nedåt till närmaste multipel av **step** som förskjuts av **offset** -talet.

Round (x [, steg [, förskjutning]])

BitCount

BitCount() returnerar hur många bitar i den binära motsvarigheten till ett decimaltal som är inställda på 1. Denna funktion returnerar alltså antalet inställda bitar i **integer_number**, där **integer_number** tolkas som ett signerat 32-bitars heltal.

Syntax:

```
BitCount(integer_number)
```

Returnerad datatyp: heltal

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
BitCount (3)	3 är 11 binärt, detta returnerar därför 2
BitCount (-1)	-1 är 64 ettor binärt, detta returnerar därför 64

Ceil

Ceil() rundar av ett tal uppåt till närmaste multipel av det **step** som förskjuts av **offset** -talet.

Jämför med funktionen **floor** som rundar av indata nedåt.

Syntax:

```
Ceil(x[, step[, offset]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
x	Indatatal.
step	Intervallsteg. Standardvärdet är 1.
offset	Definierar basen för stegintervallet. Standardvärdet är 0.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Ceil(2.4 )`	Returnerar 3 I det här exemplet är storleken på steget 1 och basen på stegintervallet 0. Intervallen är ...0 < x <=1, 1 < x <= 2, **2< x <=3**, 3< x <=4...
`Ceil(4.2 )`	Returnerar 5
`Ceil(3.88 ,0.1)`	Returnerar 3,9 I det här exemplet är storleken på intervallet 0,1 och basen på intervallet 0. Intervallen är ... 3.7 < x <= 3.8, **3.8 < x <= 3.9**, 3.9 < x <= 4.0...
`Ceil(3.88 ,5)`	Returnerar 5
`Ceil(1.1 ,1)`	Returnerar 2
`Ceil(1.1 ,1,0.5)`	Returnerar 1,5 I det här exemplet är storleken på steget 1 och förskjutningen 0,5. Det innebär att basen på stegintervallet är 0,5 och inte 0. Intervallen är ...**0.5 < x <=1.5**, 1.5 < x <= 2.5, 2.5< x <=3.5, 3.5< x <=4.5...
`Ceil(1.1 ,1,-0.01)`	Returnerar 1,99 Intervallen är ...-0.01< x <= 0.99, **0.99< x <= 1.99**, 1.99 < x <=2.99...

Combin

Combin() returnerar antalet kombinationer av **q**-element som kan väljas från en uppsättning **p**-element. Motsvaras av formeln: `Combin(p,q) = p! / q!(p-q)!` Ordningen som komponenterna väljs i är inte signifikant.

Syntax:

```
Combin(p, q)
```

Returnerad datatyp: heltal

Begränsningar:

Icke-heltal kommer att trunkeras.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
Hur många kombinationer av 7 nummer kan göras bland totalt 35 lotterinummer? `Combin( 35,7 )`	Returnerar 6 724 520

Div

Div() returnerar heltalsdelen av den aritmetiska divisionen av det första argumentet med det andra argumentet. Båda parametrarna tolkas som reella tal, de behöver alltså inte nödvändigtvis vara heltal.

Syntax:

```
Div(integer_number1, integer_number2)
```

Returnerad datatyp: heltal

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Div( 7,2 )`	Returnerar 3
`Div( 7.1,2.3 )`	Returnerar 3
`Div( 9,3 )`	Returnerar 3
`Div( -4,3 )`	Returnerar -1
`Div( 4,-3 )`	Returnerar -1
`Div( -4,-3 )`	Returnerar 1

Even

Even() returnerar True (-1) om **integer_number** är ett jämnt heltal eller noll. Den returnerar False (0) om **integer_number** är ett udda heltal och NULL om **integer_number** inte är ett heltal.

Syntax:

```
Even(integer_number)
```

Returnerad datatyp: Boolesk

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Even( 3 )`	Returnerar 0, False
`Even( 2 * 10 )`	Returnerar -1, True
`Even( 3.14 )`	Returnerar NULL

Fabs

Fabs() returnerar absolutvärdet för **x**. Resultatet blir ett positivt tal.

Syntax:

```
fabs(x)
```

Returnerad datatyp: numeriska

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`fabs( 2.4 )`	Returnerar 2,4
`fabs( -3.8 )`	Returnerar 3,8

Fact

Fact() returnerar faktorn av ett positivt heltal **x**.

Syntax:

```
Fact(x)
```

Returnerad datatyp: heltal

Begränsningar:

Om talet **x** inte är ett heltal trunkeras det. Icke-positiva nummer returnerar NULL.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Fact( 1 )`	Returnerar 1
`Fact( 5 )`	Returnerar 120 (1 * 2 * 3 * 4 * 5 = 120)
`Fact( -5 )`	Returnerar NULL

Floor

Floor() rundar av ett tal nedåt till närmaste multipel av det **step** som förskjuts av **offset** -talet.

Jämför med funktionen **ceil**, som rundar av indatatal uppåt.

Syntax:

```
Floor(x[, step[, offset]])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
x	Indatatal.
step	Intervallsteg. Standardvärdet är 1.
offset	Definierar basen för stegintervallet. Standardvärdet är 0.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Floor(2.4)`	Returnerar 2 In this example, the size of the step is 1 and the base of the step interval is 0. The intervals are ...0 <= x <1, 1 <= x < 2, **2<= x <3**, 3<= x <4....
`Floor(4.2)`	Returnerar 4
`Floor(3.88 ,0.1)`	Returnerar 3,8 I det här exemplet är storleken på intervallet 0,1 och basen på intervallet 0. Intervallen är ... 3.7 <= x < 3.8, **3.8 <= x < 3.9**, 3.9 <= x < 4.0...

Exempel	Resultat
`Floor(3.88 ,5)`	Returnerar 0
`Floor(1.1 ,1)`	Returnerar 1
`Floor(1.1 ,1,0.5)`	Returnerar 0,5 I det här exemplet är storleken på steget 1 och förskjutningen 0,5. Det innebär att basen på stegintervallet är 0,5 och inte 0. Intervallen är ...**0.5 <= x <1.5**, 1.5 <= x < 2.5, 2.5<= x <3.5,...

Fmod

fmod() är en generaliserad modulofunktion som returnerar den återstående delen av heltalsdivisionen av det första argumentet (dividenden) med det andra argumentet (divisorn). Resultatet blir ett reellt tal. Båda argumenten tolkas som reella tal, de behöver alltså inte nödvändigtvis vara heltal.

Syntax:

```
fmod(a, b)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
a	Dividend
b	Divisor

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`fmod( 7,2 )`	Returnerar 1
`fmod( 7.5,2 )`	Returnerar 1,5
`fmod( 9,3 )`	Returnerar 0
`fmod( -4,3 )`	Returnerar -1
`fmod( 4,-3 )`	Returnerar 1
`fmod( -4,-3 )`	Returnerar -1

Frac

Frac() returnerar decimaldelen av **x**.

Fraktionens definieras så att `Frac(x ) + Floor(x ) = x`. Förenklat innebär detta att fraktionen av ett positivt tal är skillnaden mellan talet (x) och heltalet som föregår decimaldelen.

Exempel: Fraktionen av 11.43 = 11.43 - 11 = 0.43

För ett negativt tal, säg -1,4, Floor(-1.4) = -2, som ger följande resultat:

Fraktionen av -1,4 = -1,4 - (-2) = -1,4 + 2 = 0,6

Syntax:

```
Frac(x)
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
x	Tal som decimaldel ska returneras för.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Frac( 11.43 )`	Returnerar 0,43
`Frac( -1.4 )`	Returnerar 0,6
Extrahera tidskomponenten från den numeriska representationen av en tidsmarkör, vilket därmed utelämnar datumet. `Time(Frac(44518.663888889))`	Returnerar 15.56.00

Mod

Mod() är en matematisk modulofunktion som returnerar den icke-negativa återstoden av en heltalsdivison. Det första argumentet är dividenden, det andra argumentet är divisorn. Båda argumenten måste vara heltalsvärden.

Syntax:

```
Mod(integer_number1, integer_number2)
```

Returnerad datatyp: heltal

Begränsningar:

integer_number2 måste vara större än 0.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Mod( 7,2 )`	Returnerar 1
`Mod( 7.5,2 )`	Returnerar NULL
`Mod( 9,3 )`	Returnerar 0
`Mod( -4,3 )`	Returnerar 2
`Mod( 4,-3 )`	Returnerar NULL
`Mod( -4,-3 )`	Returnerar NULL

Odd

Odd() returnerar True (-1) om **integer_number** är ett udda heltal eller noll. Det returnerar False (0) om **integer_number** är ett jämnt heltal och NULL om **integer_number** inte är ett heltal.

Syntax:

```
Odd(integer_number)
```

Returnerad datatyp: Boolesk

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Odd( 3 )`	Returnerar -1, True
`Odd( 2 * 10 )`	Returnerar 0, False
`Odd( 3.14 )`	Returnerar NULL

Permut

Permut() returnerar antalet permutationer av **q** element som kan väljas från en uppsättning med **p** element. Motsvaras av formeln: `Permut(p,q) = (p)! / (p - q)!` Ordningen som komponenterna väljs i är signifikant.

Syntax:

```
Permut(p, q)
```

Returnerad datatyp: heltal

Begränsningar:

Om parametrarna inte är heltal kommer de att trunkeras.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
På hur många sätt kan medaljerna guld, silver och brons fördelas vid en 100-meters final med 8 deltagare? `Permut( 8,3 )`	Returnerar 336

Round

Round() returnerar resultatet av avrundning av ett tal uppåt eller nedåt till närmaste multipel av **step** som förskjuts av **offset** -talet.

Om det tal som ska rundas av ligger exakt mitt i ett intervall sker avrundningen uppåt.

Syntax:

```
Round(x[, step[, offset]])
```

Returnerad datatyp: numeriska

> *Om du rundar av ett flyttalsnummer kan felaktiga resultat visas. De här avrundningsfelen uppstår eftersom flyttalsnummer återges med ett ändligt antal binära siffror. Resultaten beräknas därför med hjälp av ett tal som redan är avrundat. Om dessa avrundningsfel påverkar ditt arbete multiplicerar du talen för att omvandla dem till heltal innan du avrundar.*

Argument:

Argument

Argument	Beskrivning
x	Indatatal.
step	Intervallsteg. Standardvärdet är 1.
offset	Definierar basen för stegintervallet. Standardvärdet är 0.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Round(3.8 )`	Returnerar 4 I det här exemplet är storleken på steget 1 och basen på stegintervallet 0. Intervallen är ...0 <= x <1, 1 <= x < 2, 2<= x <3, **3<= x <4**...
`Round(3.8,4 )`	Returnerar 4
`Round(2.5 )`	Returnerar 3. I det här exemplet är storleken på steget 1 och basen på stegintervallet 0. Intervallen är ...0 <= x <1, 1 <= x <2, **2<= x <3**...
`Round(2,4 )`	Returnerar 4. Avrundat uppåt eftersom 2 är exakt hälften av stegintervallet 4. I det här exemplet är storleken på steget 4 och basen på stegintervallet 0. Intervallen är ...**0 <= x <4**, 4 <= x <8, 8<= x <12...
`Round(2,6 )`	Returnerar 0. Avrundat nedåt eftersom 2 är mindre än hälften av stegintervallet 6. I det här exemplet är storleken på steget 6 och basen på stegintervallet 0. Intervallen är ...**0 <= x <6**, 6 <= x <12, 12<= x <18...
`Round(3.88 ,0.1)`	Returnerar 3,9 I det här exemplet är storleken på steget 0,1 och basen på stegintervallet 0. Intervallen är ... 3.7 <= x <3.8, **3.8 <= x <3.9**, 3.9 <= x < 4.0...
`Round (3.88875,1/1000)`	Returnerar 3,889 I det här exemplet är storleken på steget 0,001, vilket avrundar talet uppåt och begränsar det till tre decimalpunkter.
`Round(3.88 ,5)`	Returnerar 5
`Round(1.1 ,1,0.5)`	Returnerar 1,5 I det här exemplet är storleken på steget 1 och basen på stegintervallet 0,5. Intervallen är ...**0.5 <= x <1.5**, 1.5 <= x <2.5, 2.5<= x <3.5...

Sign

Sign() returnerar 1, 0 eller -1 beroende på om **x** är ett positivt tal, 0 eller ett negativt tal.

Syntax:

```
Sign(x)
```

Returnerad datatyp: numeriska

Begränsningar:

Om inget numeriskt värde påträffas returneras NULL.

Exempel och resultat:

Exempel och resultat

Exempel	Resultat
`Sign( 66 )`	Returnerar 1
`Sign( 0 )`	Returnerar 0
`Sign( - 234 )`	Returnerar -1

5.14 Geospatiala funktioner

De här funktionerna används för att hantera geospatiala data i kartvisualiseringar. Qlik Sense följer GeoJSON-specifikationer för geospatiala data och stöder följande:

- Punkt
- Linjesträng
- Polygon
- Multipolygon

För mer information om GeoJSON-specifikationer, se:
GeoJSON.org

Översikt över geospatiala funktioner

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

Det finns två kategorier av geospatiala funktioner: aggregering och icke-aggregering.

Aggregeringsfunktioner tar en geometriuppsättning (områdspunkter) som indata och returnerar en enhetlig geometri. Till exempel kan flera områden slås ihop och en enda gräns för det aggregerade området kan ritas ut på kartan.

Icke-aggregerande funktioner tar en enda geometri och returnerar en geometri. Till exempel för funktionen GeoGetPolygonCenter(), om gränsgeometrin för ett område används som indata så returneras punktgeometri (longitud och latitud) för mitten av det området.

Följande är aggregeringsfunktioner:

GeoAggrGeometry

GeoAggrGeometry() används för att aggregera ett antal områden till ett större område, exempelvis aggregera ett antal underregioner till en region.

```
GeoAggrGeometry (field_name)
```

GeoBoundingBox

GeoBoundingBox() används i skript för att aggregera geometri på en yta och beräkna den minsta omskrivande rektangeln som innehåller alla koordinater.

```
GeoBoundingBox (field_name)
```

GeoCountVertex

GeoCountVertex() används för att hitta antal hörn som en polygongeometri innehåller.

```
GeoCountVertex(field_name)
```

GeoInvProjectGeometry

GeoInvProjectGeometry() används för att aggregera geometri på en yta och tillämpa inversionen av en projektion.

```
GeoInvProjectGeometry(type, field_name)
```

GeoProjectGeometry

GeoProjectGeometry() används för att aggregera geometri på en yta och tillämpa en projektion.

```
GeoProjectGeometry(type, field_name)
```

GeoReduceGeometry

GeoReduceGeometry() används för att minska antalet hörn i en geometri och för att aggregera ett antal områden till ett område, samtidigt som gränslinjerna från de enskilda områdena fortfarande visas.

```
GeoReduceGeometry (geometry)
```

Följande är icke-aggregerande funktioner:

GeoGetBoundingBox

GeoGetBoundingBox() används i skript- och diagramuttryck för att beräkna den minsta geospatiala omskrivande rektangeln som innehåller alla koordinaterna i en geometri.

```
GeoGetBoundingBox (geometry)
```

GeoGetPolygonCenter

GeoGetPolygonCenter() används i skript och diagramuttryck för att beräkna och returnera mittpunkten i en geometri.

```
GeoGetPolygonCenter (geometry)
```

GeoMakePoint

GeoMakePoint() används i skript och diagramuttryck för att skapa och tagga en punkt med latitud och longitud.

```
GeoMakePoint (lat_field_name, lon_field_name)
```

GeoProject

GeoProject() används i skript och diagramuttryck för att tillämpa en projektion på en geometri.

```
GeoProject (type, field_name)
```

GeoAggrGeometry

GeoAggrGeometry() används för att aggregera ett antal områden till ett större område, exempelvis aggregera ett antal underregioner till en region.

Syntax:

```
GeoAggrGeometry(field_name)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

Vanligtvis kan **GeoAggrGeometry()** användas för att kombinera geospatiala gränsdata. Till exempel så kanske du har postnummerområden för förorter i en stad och försäljningsintäkter för varje områden. Om en säljares område täcker flera postnummer kan det vara bra att presentera total försäljning per område, snarare än individuella områden, och visa resultaten på en färgkarta.

GeoAggrGeometry() kan beräkna aggregerade data för de enskilda förortsområdena och generera en sammanslagen områdesgeometri i datamodellen. Om säljområdenas gränser sedan justerar kan de nya sammanslagna områdena och intäkterna visas på kartan när data laddas igen.

Eftersom **GeoAggrGeometry()** är en aggregerande funktion krävs en **LADDA** instruktion med en **Gruppera enligt**-klausul.

*Gränserna på kartan som skapats med **GeoAggrGeometry()** gäller för de sammanslagna områdena. Om du vill visa enskilda gränser för områden som de såg ut innan de aggregerades kan du använda**GeoReduceGeometry()**.*

Exempel:
I detta exempel läses en KML-fil med områdesdata in och sedan läses en tabell med aggregerade områdesdata in.

```
[Kartkälla]: LADDA [world.Name], [world.Point], [world.Area] FRÅN [lib://Downloads/world.kml]
(kml, Tabellen är [World.shp/Features]); Karta: LADDA world.Name, GeoAggrGeometry(world.Area)
som [AggrArea] resident MapSource Group By world.Name;

 Drop Table MapSource;
```

GeoBoundingBox

GeoBoundingBox() används i skript för att aggregera geometri på en yta och beräkna den minsta omskrivande rektangeln som innehåller alla koordinater.

En GeoBoundingBoxvisas som en lista med fyra värden: vänster, höger, överst, nederst.

Syntax:

```
GeoBoundingBox(field_name)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

GeoBoundingBox() aggregerar en uppsättning geometrier och returnerar fyra koordinater för den minsta rektangeln som innehåller alla koordinater för den aggregerade geometrin.

Om du vill visualisera resultaten på en karta ska du överföra resultatsträngen med fyra koordinater i polygonformat, tagga det överförda fältet med ett geopolygonformat och dra och släppa det fältet på kartobjektet. De rektangulära rutorna kommer sedan att visas i kartvisualiseringen.

GeoCountVertex

GeoCountVertex() används för att hitta antal hörn som en polygongeometri innehåller.

Syntax:

```
GeoCountVertex(field_name)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

GeoGetBoundingBox

GeoGetBoundingBox() används i skript- och diagramuttryck för att beräkna den minsta geospatiala omskrivande rektangeln som innehåller alla koordinaterna i en geometri.

En geospatial ormskrivande rektangel, som skapats av funktionen GeoBoundingBox(), visas som en lista med fyra värden: vänster, höger, överst, nederst.

Syntax:

```
GeoGetBoundingBox(field_name)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

*Använd inte satsen **Group by** i Skriptredigeraren med denna och andra icke-aggregerade geospatiala funktioner, eftersom detta kommer att leda till ett fel vid laddning.*

GeoGetPolygonCenter

GeoGetPolygonCenter() används i skript och diagramuttryck för att beräkna och returnera mittpunkten i en geometri.

I vissa fall är det ett krav att plotta små punkter istället för färgifyllning på en karta. Om de befintliga geospatiala data bara är tillgängliga i form av områdesgeometri (till exempel en gräns) ska du använda **GeoGetPolygonCenter()** för att hämta ett par med longitud och latitud för mittenområdet.

Syntax:

```
GeoGetPolygonCenter(field_name)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

*Använd inte satsen **Group by** i Skriptredigeraren med denna och andra icke-aggregerade geospatiala funktioner, eftersom detta kommer att leda till ett fel vid laddning.*

GeoInvProjectGeometry

GeoInvProjectGeometry() används för att aggregera geometri på en yta och tillämpa inversionen av en projektion.

Syntax:

```
GeoInvProjectGeometry(type, field_name)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
type	Projektionstyp som används för att omvandla kartans geometri. Den kan ta ett av två värden: ”enhet” (standard), som ger en 1:1-projektion, eller ”mercator”, som ger standardprojektionen Mercator.
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

Exempel:

Skriptexempel

Exempel	Resultat
I en `Load`-sats: `GeoInvProjectGeometry ('mercator',AreaPolygon) as InvProjectGeometry`	Geometrin laddas som **AreaPolygon** omvandlas med hjälp av inverterad omvandling av Mercator-projektionen och sparas som **InvProjectGeometry** för att användas i visualiseringar.

GeoMakePoint

GeoMakePoint() används i skript och diagramuttryck för att skapa och tagga en punkt med latitud och longitud. GeoMakePoint returnerar punkter i ordningen longitud-latitud.

Syntax:

```
GeoMakePoint(lat_field_name, lon_field_name)
```

Returnerad datatyp: sträng, formaterad [longitud, latitud]

Argument:

Argument

Argument	Beskrivning
lat_field_name	Ett fält eller ett uttryck som hänvisar till ett fält som representerar punktens latitud.
lon_field_name	Ett fält eller ett uttryck som hänvisar till ett fält som representerar punktens longitud.

*Använd inte satsen **Group by** i Skriptredigeraren med denna och andra icke-aggregerade geospatiala funktioner, eftersom detta kommer att leda till ett fel vid laddning.*

GeoProject

GeoProject() används i skript och diagramuttryck för att tillämpa en projektion på en geometri.

Syntax:

```
GeoProject(type, field_name)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
type	Projektionstyp som används för att omvandla kartans geometri. Den kan ta ett av två värden: ”enhet” (standard), som ger en 1:1-projektion, eller ”mercator”, som ger webbprojektionen Mercator.
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

*Använd inte satsen **Group by** i Skriptredigeraren med denna och andra icke-aggregerade geospatiala funktioner, eftersom detta kommer att leda till ett fel vid laddning.*

Exempel:

Skriptexempel

Exempel	Resultat
I en `Load`-sats: `GeoProject ('mercator',Area) as GetProject`	Mercator-projektionen tillämpas på geometrin som laddats som **Area**, och resultatet sparas som **GetProject**.

GeoProjectGeometry

GeoProjectGeometry() används för att aggregera geometri på en yta och tillämpa en projektion.

Syntax:

```
GeoProjectGeometry(type, field_name)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
type	Projektionstyp som används för att omvandla kartans geometri. Den kan ta ett av två värden: ”enhet” (standard), som ger en 1:1-projektion, eller ”mercator”, som ger webbprojektionen Mercator.

Argument	Beskrivning
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.

Exempel:

Exempel	Resultat
I en `Load`-sats: `GeoProjectGeometry ('mercator',AreaPolygon) as ProjectGeometry`	Geometrin som laddats som **AreaPolygon** omvandlas med hjälp Mercator-projektion och sparas som **ProjectGeometry** för att användas i visualiseringar.

GeoReduceGeometry

GeoReduceGeometry() används för att minska antalet hörn i en geometri och för att aggregera ett antal områden till ett område, samtidigt som gränslinjerna från de enskilda områdena fortfarande visas.

Syntax:

```
GeoReduceGeometry(field_name[, value])
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
field_name	Ett fält eller ett uttryck som hänvisar till ett fält som innehåller geometrin som ska representeras. Detta kan antingen vara en punkt (eller en uppsättning punkter) som definierar longitud eller latitud, eller ett område.
value	Mängden reduktion som ska tillämpas på geometrin. Intervallet är från 0 till 1, där 0 representerar ingen reduktion och 1 representerar maximal reduktion av hörn.  *Användning av ett value på 0,9 eller högre med en komplex datauppsättning kan minska antalet hörn till en nivå där den visuella återgivningen är felaktig.*

GeoReduceGeometry() utför även en funktion som liknar **GeoAggrGeometry()** genom att det aggregerar ett antal områden till ett enda. Skillnaden är att de enskilda gränslinjerna från förhandsaggregerade data visas på kartan om du använder **GeoReduceGeometry()**.

Eftersom **GeoReduceGeometry()** är en aggregerande funktion krävs en **LOAD**-sats med en **Group by**-sats.

Exempel:
I detta exempel läses en KML-fil med områdesdata in och sedan läses en tabell in med minskade och aggregerade områdesdata.

```
[Kartkälla]: LOAD [world.Name], [world.Point], [world.Area] FROM [lib://Downloads/world.kml]
(kml, Tabellen är [World.shp/Features]); Karta: LOAD world.Name, GeoReduceGeometry
(world.Area,0.5) as [ReducedArea] resident MapSource Group By world.Name;

 Drop Table MapSource;
```

5.15 Tolkningsfunktioner

Tolkningsfunktionerna utvärderar innehållet i textdatafält eller uttryck och fastställer ett angivet dataformat för det resulterande numeriska värdet. Med de här funktionerna kan du ange format för talet, i enlighet med dess datatyp, vilket kan vara sådana attribut som t.ex. decimalavgränsare, tusendelsavgränsare och datumformat.

Tolkningsfunktionerna returnerar ett dualt värde med både strängen och det numeriska värdet, men det kan också ses som att de gör en konvertering från sträng till tal. Funktionerna tar textvärdet i indatauttrycken och generera ett tal som representerar strängen.

Formateringsfunktionerna däremot gör det motsatta: de tar taluttryck och utvärderar dem som strängar och specificerar sedan format för den resulterande texten.

Om inga tolkningsfunktioner används, tolkar Qlik Sense data som en blandning av tal, datum, tider, tidsmarkörer och strängar och använder standardinställningar för talformat, datumformat och tidsformat, som har definierats av skriptvariabler och av operativsystemet.

Alla tolkningsfunktioner kan användas både i dataladdningsskript och diagramuttryck.

Alla numeriska värden anges med en decimalpunkt som decimalavgränsare.

Tolkningsfunktioner - en översikt

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

Date#

Date# utvärderar ett uttryck som ett datum i det format som har angetts i det andra argumentet, om så är tillämpligt. Om formatkoden utelämnas används det standarddatumformat som definierats i operativsystemet.

```
Date# (page 632)(text[, format])
```

Interval#

Interval#() utvärderar ett textuttryck som ett tidsintervall i formatet som har ställts in för operativsystemet, som standard eller i formatet som anges i det andra argumentet, om så är tillämpligt.

```
Interval# (page 633)(text[, format])
```

Money#

Money#() konverterar en textsträng till ett monetärt värde i det format som har ställts in i laddningsskriptet eller operativsystemet, om inte en formatsträng tillhandahålls. Egna symboler för decimaler och tusentalsseparatorer är valbara parametrar.

```
Money# (page 634)(text[, format[, dec_sep[, thou_sep ] ] ])
```

Num#

Num#() tolkar en textsträng som ett numeriskt värde, vilket innebär att indatasträngen konverteras till ett tal med det format som specificerats i den andra parametern. Om den andra parametern utelämnas använder funktionen de decimal- och tusentalsavgränsare som anges i dataladdningsskriptet. Egna symboler för decimaler och tusentalsseparatorer är valbara parametrar.

```
Num# (page 635)(text[ , format[, dec_sep[ , thou_sep]]])
```

Text

Text() tvingar en tolkning av uttrycket som text, även om en numerisk tolkning är möjlig.

```
Text(expr)
```

Time#

Time#() utvärderar ett uttryck som ett tidsvärde, i det tidsformat som har ställts in i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns..

```
Time# (page 637)(text[, format])
```

Timestamp#

Timestamp#() utvärderar ett uttryck som ett datum- och tidsvärde, i det tidsmarkörformat som har ställts in i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns.

```
Timestamp# (page 638)(text[, format])
```

Se även:

- *Formateringsfunktioner (page 597)*

Date#

Date# utvärderar ett uttryck som ett datum i det format som har angetts i det andra argumentet, om så är tillämpligt.

Syntax:

```
Date#(text[, format])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
text	Textsträngen som ska utvärderas.
format	En sträng som beskriver formatet på den sträng som ska utvärderas. Om datumformatet utelämnas används det datumformat som är inställt i systemvariablerna i dataladdningsskriptet eller operativsystemet.

Exempel och resultat:
I följande exempel används datumformatet **M/D/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet.

Lägg till det här exempelskriptet i appen och kör det.

```
Load *,
Num(Date#(StringDate)) as Date;
LOAD * INLINE [
StringDate
8/7/97
8/6/1997
```

Om du skapar en tabell med **StringDate** och **Date** som dimensioner, blir resultaten så här:

Resultat

StringDate	Datum
8/7/97	35649
8/6/1997	35648

Interval#

Interval#() utvärderar ett textuttryck som ett tidsintervall i formatet som har ställts in för operativsystemet, som standard eller i formatet som anges i det andra argumentet, om så är tillämpligt.

Syntax:

```
Interval#(text[, format])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
text	Textsträngen som ska utvärderas.
format	En sträng som beskriver det förväntade indataformat som ska användas när man omvandlar strängen till ett numeriskt intervall. Om den utelämnas används det korta datumformat, tidsformat och decimalavgränsare som har ställts in för operativsystemet.

Funktionen **interval#** omvandlar ett texttidsintervall till en numerisk motsvarighet.

Exempel och resultat:
Exemplen nedan förutsätter följande operativsystemsinställningar:

- Kort datumformat: YY-MM-DD
- Tidsformat: M/D/YY
- Decimalavgränsare för tal: .

Resultat

Exempel	Resultat
`Interval#( A, 'D hh:mm' )` där A='1 09:00'	1.375

Money#

Money#() konverterar en textsträng till ett monetärt värde i det format som har ställts in i laddningsskriptet eller operativsystemet, om inte en formatsträng tillhandahålls. Egna symboler för decimaler och tusentalsseparatorer är valbara parametrar.

Syntax:

```
Money#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
text	Textsträngen som ska utvärderas.

Argument	Beskrivning
format	En sträng som beskriver det förväntade indataformat som ska användas när man omvandlar strängen till ett numeriskt intervall. Om denna utelämnas används det valutaformat som definierats i operativsystemet.
dec_sep	Sträng som specificerar decimalavgränsare. Om det utelämnas används värdet för MoneyDecimalSep som angetts i dataladdningsskriptet.
thou_sep	Sträng som specificerar tusendelsavgränsare. Om det utelämnas används värdet för MoneyThousandSep som angetts i dataladdningsskriptet.

money#-funktionen beter sig vanligtvis som **num#**-funktionen, men hämtar standardvärdena för decimal- och tusentalsavgränsare från variablerna för valutaformat eller från operativsystemets inställningar för valuta.

Exempel och resultat:
Exemplen nedan förutsätter följande två operativsystemsinställningar:

- Standardinställning för valutaformat 1: kr # ##0,00
- Standardinställning för valutaformat 2: $ #,##0.00

```
Money#(A , '# ##0,00 kr' )
```
där A=35 648,37 kr

Resultat

Resultat	Inställning 1	Inställning 2
Sträng	35 648.37 kr	35 648.37 kr
Tal	35648.37	3564837

```
Money#( A, ' $#', '.', ',' )
```
där A= 35 648,37 $

Resultat

Resultat	Inställning 1	Inställning 2
Sträng	$35,648.37	$35,648.37
Tal	35648.37	35648.37

Num#

Num#() tolkar en textsträng som ett numeriskt värde, vilket innebär att indatasträngen konverteras till ett tal med det format som specificerats i den andra parametern. Om den andra parametern utelämnas använder funktionen de decimal- och tusentalsavgränsare som anges i dataladdningsskriptet. Egna symboler för decimaler och tusentalsseparatorer är valbara parametrar.

Syntax:

```
Num#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Returnerad datatyp: dual

Funktionen **Num#()** returnerar ett dualt värde med både strängvärdet och det numeriska värdet. Funktionen tar indatauttryckets textrepresentation och genererar ett tal. Talets format ändras inte: utdata formateras på samma sätt som indata.

Argument:

Argument

Argument	Beskrivning
text	Textsträngen som ska utvärderas.
format	Sträng som specificerar talformatet som används i den första parametern. Om den utelämnas används de decimal- och tusentalsavgränsare som anges i dataladdningsskriptet.
dec_sep	Sträng som specificerar decimalavgränsare. Om det utelämnas används värdet för variabeln DecimalSep som angetts i dataladdningsskriptet.
thou_sep	Sträng som specificerar tusendelsavgränsare. Om det utelämnas används värdet för variabeln ThousandSep som angetts i dataladdningsskriptet.

Exempel och resultat:
Följande tabell visar resultatet av *Num#(A, '#', '.', ',')* för olika A-värden.

Resultat

En	Strängrepresentation	Numeriskt värde (visas här med decimalkomma)
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() tvingar en tolkning av uttrycket som text, även om en numerisk tolkning är möjlig.

Syntax:

```
Text (expr)
```

Returnerad datatyp: dual

Exempel:

```
Text( A )
```
där A=1234

Resultat

Sträng	Tal
1234	-

Exempel:

```
Text( pi( ) )
```

Resultat

Sträng	Tal
3.1415926535898	-

Time#

Time#() utvärderar ett uttryck som ett tidsvärde, i det tidsformat som har ställts in i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns..

Syntax:

```
time#(text[, format])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
text	Textsträngen som ska utvärderas.
format	En sträng som beskriver formatet på den sträng som ska utvärderas. Om den utelämnas används det korta datumformat, tidsformat och decimalavgränsare som har ställts in för operativsystemet.

Exempel:

- Standardinställning för tidsformat 1: hh:mm:ss
- Standardinställning för tidsformat 2: hh.mm.ss

```
time#( A )
```
där A=09:00:00

Resultat

Resultat	Inställning 1	Inställning 2
Sträng:	09:00:00	09:00:00
Tal:	0.375	-

Exempel:

- Standardinställning för tidsformat 1: hh:mm:ss
- Standardinställning för tidsformat 2: hh.mm.ss

```
time#( A, 'hh.mm' )
```
där A=09,00

Resultat

Resultat	Inställning 1	Inställning 2
Sträng:	09.00	09.00
Tal:	0.375	0.375

Timestamp#

Timestamp#() utvärderar ett uttryck som ett datum- och tidsvärde, i det tidsmarkörformat som har ställts in i dataladdningsskriptet eller operativsystemet, om inte en formatsträng finns.

Syntax:

```
timestamp#(text[, format])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
text	Textsträngen som ska utvärderas.
format	En sträng som beskriver formatet på den sträng som ska utvärderas. Om den utelämnas används det korta datumformat, tidsformat och decimalavgränsare som har ställts in för operativsystemet. ISO 8601 stöds för tidsmarkörer.

Exempel:

I följande exempel används datumformatet **M/D/YYYY**. Datumformatet anges i **SET DateFormat**-satsen längst upp i dataladdningsskriptet.

Lägg till det här exempelskriptet i appen och kör det.

```
Load *,
Timestamp(Timestamp#(String)) as TS;
LOAD * INLINE [
Sträng
2015-09-15T12:13:14
1952-10-16T13:14:00+0200
1109-03-01T14:15
];
```

Om du skapar en tabell med **String** och **TS** som dimensioner, blir resultaten så här:

Resultat

Sträng	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

5.16 Postöverskridande funktioner

Postöverskridande funktioner används i följande fall:

- I dataladdningsskriptet när ett värde från tidigare laddade dataposter behövs för utvärdering av den aktuella posten.
- I ett diagramuttryck, om ett annat värde från datauppsättningen till en visualisering behövs.

Sortering efter y-värde i diagram eller sortering efter uttryckskolumn i raka tabeller är inte tillåtet när postöverskridande funktioner för diagram används i något av diagrammets uttryck. Dessa sorteringsalternativ är därför automatiskt inaktiverade.

Självrefererande uttrycksdefinitioner kan endast göras på ett pålitligt sätt i tabeller med färre än 100 rader, men detta kan variera beroende på maskinvaran som Qlik-motorn körs på.

Radfunktioner

Dessa funktioner kan endast användas i diagramuttryck.

Above

Above() utvärderar ett uttryck på en rad ovanför den aktuella raden inom ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden direkt ovanför. För diagram som inte är tabeller utvärderar **Above()** för raden ovanför den aktuella raden i diagrammets raka tabellmotsvarighet.

```
Above - diagramfunktion([TOTAL [<fld{,fld}>]] expr [ , offset [,count]])
```

Below

Below() utvärderar ett uttryck på en rad under den aktuella raden inom ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden direkt nedanför. För diagram som inte är tabeller utvärderar **Below()** för raden nedanför den aktuella kolumnen i diagrammets raka tabellmotsvarighet.

```
Below - diagramfunktion([TOTAL[<fld{,fld}>]] expression [ , offset [,count
 ]])
```

Bottom

Bottom() utvärderar ett uttryck på den sista (nedersta) raden i ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden nederst. För diagram som inte är tabeller görs utvärderingen på den sista raden i den aktuella kolumnen i diagrammets raka tabellmotsvarighet.

```
Bottom - diagramfunktion([TOTAL[<fld{,fld}>]] expr [ , offset [,count ]])
```

Top

Top() utvärderar ett uttryck på den första (översta) raden i ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden högst upp. För diagram som inte är tabeller görs **Top()**-utvärderingen på den första raden i den aktuella kolumnen i diagrammets raka tabellmotsvarighet.

```
Top - diagramfunktion([TOTAL [<fld{,fld}>]] expr [ , offset [,count ]])
```

NoOfRows

NoOfRows() returnerar antalet rader i det aktuella kolumnsegmentet i en tabell. För bitmappsdiagram returnerar **NoOfRows()** antalet rader i diagrammets raka tabellmotsvarighet.

```
NoOfRows - diagramfunktion([TOTAL])
```

Kolumnfunktioner

Dessa funktioner kan endast användas i diagramuttryck.

Column

Column() returnerar det värde som återfinns i kolumnen som motsvarar **ColumnNo** i en rak tabell om dimensionerna ignoreras. Exempelvis returnerar **Column(2)** värdet för den andra måttkolumnen.

```
Column - diagramfunktion(ColumnNo)
```

Dimensionality

Dimensionality() returnerar antalet dimensioner för den aktuella raden. Vad gäller pivottabeller returnerar funktionen det totala antalet dimensionskolumner som har icke-aggregerat innehåll, d.v.s. inte innehåller delsummor eller dolda aggregeringar.

```
Dimensionality - diagramfunktion ( )
```

Secondarydimensionality

SecondaryDimensionality() returnerar antalet dimensionspivottabellrader som har icke-aggregerat innehåll, d.v.s. inte innehåller delsummor eller dolda aggregeringar. Denna funktion motsvarar funktionen **dimensionality()** för horisontella pivottabelldimensioner.

```
SecondaryDimensionality - diagramfunktion ( )
```

Fältfunktioner

FieldIndex

FieldIndex() returnerar den placering som fältvärdet **value** har i fältet **field_name** (i laddningsordning).

```
FieldIndex (field_name , value)
```

FieldValue

FieldValue() returnerar det värde som påträffas vid placeringen **elem_no** för fältet **field_name** (i laddningsordning).

```
FieldValue (field_name , elem_no)
```

FieldValueCount

FieldValueCount() är en **heltal**sfunktion som returnerar antalet distinkta värden i ett fält.

```
FieldValueCount (field_name)
```

Pivottabellfunktioner

Dessa funktioner kan endast användas i diagramuttryck.

After

After() returnerar värdet för ett uttryck utvärderat med en pivottabells dimensionsvärden som de visas i kolumnen efter den aktuella kolumnen inom ett radsegment i pivottabellen.

```
After - diagramfunktion([TOTAL] expression [ , offset [,n]])
```

Before

Before()returnerar värdet av ett uttryck utvärderat men en pivottabells dimensionsvärden som de visas i kolumnen framför den aktuella kolumnen inom ett radsegment i pivottabellen.

```
Before - diagramfunktion([TOTAL] expression [ , offset [,n]])
```

First

First() returnerar värdet för ett uttryck utvärderat med en pivottabells dimensionsvärden som de visas i den första kolumnen i det aktuella radsegmentet i pivottabellen. Funktionen returnerar NULL i alla diagramtyper utom pivottabeller.

```
First - diagramfunktion([TOTAL] expression [ , offset [,n]])
```

Last

Last() returnerar värdet för ett uttryck utvärderat med en pivottabells dimensionsvärden som de visas i den sista kolumnen i det aktuella radsegmentet i pivottabellen. Funktionen returnerar NULL i alla diagramtyper utom pivottabeller.

```
Last - diagramfunktion([TOTAL] expression [ , offset [,n]])
```

ColumnNo

ColumnNo() returnerar numret på den aktuella kolumnen inom det aktuella radsegmentet i en pivottabell. Första kolumnen är nummer 1.

```
ColumnNo - diagramfunktion([TOTAL])
```

NoOfColumns

NoOfColumns() returnerar antalet kolumner i det aktuella radsegmentet i en pivottabell.

```
NoOfColumns - diagramfunktion([TOTAL])
```

Postöverskridande funktioner i dataladdningsskriptet

Exists

Exists() avgör om ett specifikt fältvärde redan har laddats in i fältet i dataladdningsskriptet. Funktionen returnerar TRUE eller FALSE, så att det kan användas i **where**-satsen för en **LOAD**-sats eller en **IF**-sats.

```
Exists (field_name [, expr])
```

LookUp

Lookup() letar in i en tabell som redan är laddad och returnerar värdet av **field_name** vilket motsvarande den första förekomsten av värdet **match_field_value** i fältet **match_field_name**. Tabellen kan vara den aktuella tabell eller en annan tabell som har laddats.

```
LookUp (field_name, match_field_name, match_field_value [, table_name])
```

Peek

Peek() returnerar värdet för ett fält i en tabell för en rad som redan har laddats. Radnumret kan anges, liksom tabellen. Om inget radnummer anges används posten som laddades senast.

```
Peek (field_name[, row_no[, table_name ] ])
```

Previous

Previous() returnerar värdet av uttrycket **expr** genom att använda data från en tidigare indatapost som inte uteslutits till följd av en **where**-sats. För den första posten i en intern tabell kommer funktionen att returnera NULL.

```
Previous (page 674)(expr)
```

Se även:

- *Intervallfunktioner (page 693)*

Above - diagramfunktion

Above() utvärderar ett uttryck på en rad ovanför den aktuella raden inom ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden direkt ovanför. För diagram som inte är tabeller utvärderar **Above()** för raden ovanför den aktuella raden i diagrammets raka tabellmotsvarighet.

Syntax:

```
Above([TOTAL] expr [ , offset [,count]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Om en offsetn, större än 0 anges, flyttas utvärderingen av uttrycket n antal rader ovanför den aktuella raden. Om startpunkt anges till 0 utvärderas uttrycket på den aktuella raden. Anges ett negativt värde för startpunkten gör det att Above-funktionen fungerar likadant som Below-funktionen med motsvarande positivt värde för startpunkten.
count	Om ett tredje uttryck, **count**, anges till större än 1, returnerar funktionen ett intervall av **count**-värden, ett för varje **count**-tabellrad. Raderna räknas uppåt från den ursprungliga cellen. I denna form kan funktionen användas som argument i någon av de speciella intervallfunktionerna. *Intervallfunktioner (page 693)*
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

På kolumnsegmentets första rad returneras värdet NULL, eftersom det inte finns någon rad ovanför denna.

Ett kolumnsegment definieras som en konsekutiv underuppsättning med celler som har samma värden för dimensionerna i den aktuella sorteringsordningen. Postöverskridande diagramfunktioner beräknas i kolumnsegmentet exklusive dimensionen längst till höger i motsvarande raka tabelldiagram. Om det enbart finns en dimension i diagrammet, eller om kvalificeraren TOTAL anges, utvärderas uttrycket över en hel tabell.

Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner, utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

Begränsningar:

Rekursiva anrop returnerar NULL.

Exempel och resultat:

Example 1:

Tabellvisualisering för exempel 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

I skärmdumpen av tabellen som visas i det här exemplet skapas tabellvisualiseringen från dimensionen **Customer** och måtten: `Sum(Sales)` och `Above(Sum(Sales))`.

Kolumnen Above(Sum(Sales)) returnerar NULL för **Customer**-raden **Astrida**, eftersom det inte finns någon rad ovanför. Resultatet för raden **Betacab** visar värdet för Sum(Sales) för **Astrida**, resultatet för **Canutility** visar värdet för **Sum(Sales)** för **Betacab** och så vidare.

För kolumnen med etiketten Sum(Sales)+Above(Sum(Sales)) visar raden för **Betacab** resultatet av att **Sum(Sales)**-värdena lades till för raderna **Betacab** + **Astrida** (539+587). Resultatet för raden **Betacab** visar resultatet då **Sum(Sales)**-värden har lagts till för **Canutility** + **Canutility** (683+539).

Måttet med etiketten Above offset 3 som skapades med uttrycket `Sum(Sales)+Above(Sum(Sales), 3)` har argumentet **offset**, angivet som 3. Effekten blir att värdet hämtas i raden tre rader ovanför den aktuella raden. Det lägger till **Sum(Sales)**-värdet för aktuell **Customer** på värdet för **Customer** tre rader högre upp. Värdena som returneras för de första tre **Customer**-raderna är null.

Tabellen visar även mer komplexa mått: ett skapat utifrån `Sum(Sales)+Above(Sum(Sales))` och ett med etiketten **Higher?**, som skapas utifrån `IF(Sum(Sales)>Above(Sum(Sales)), 'Higher')`.

Funktionen kan även användas i andra diagram, inte bara tabeller - exempelvis i stapeldiagram.

För andra diagramtyper, ska du omvandla diagrammet till den raka tabellmotsvarigheten så att du enkelt kan tolka vilken rad funktionen är relaterad till.

Example 2:

I tabellerna i skärmdumparna har fler dimensioner lagts till i visualiseringarna: **Month** och **Product**. För diagram med flera dimensioner beror resultatet för uttryck som innehåller funktionerna **Above**, **Below**, **Top** och **Bottom** på den ordning i vilken kolumndimensionerna sorteras av Qlik Sense. Qlik Sense evaluerar

funktionerna baserat på de kolumnsegment som är resultatet från den dimension som kommer sist i sorteringsordningen. Sorteringsordningen för kolumner styrs från egenskapspanelen under **Sortering**. Den motsvarar inte nödvändigtvis den ordning i vilken kolumnerna visas i en tabell.

I följande skärmdump av tabellvisualiseringen för exempel 2 är den sista dimensionen i sorteringsordningen **Month**, så **Above**-funktionen utvärderar utifrån månader. Det finns en serie resultat för varje **Product**-värde för varje månad (**Jan** till **Aug**) – ett kolumnsegment. Detta följs av en serie för nästa kolumnsegment: för varje **Month** för nästa **Product**. Det finns ett kolumnsegment för varje **Customer**-värde för varje **Product**.

Tabellvisualisering för exempel 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:

I skärmdumpen av tabellvisualiseringen för exempel 3 är den senast sorterade dimensionen **Product**. Detta görs genom att flytta dimensionen Product till läge 3 på fliken Sortering i egenskapspanelen. Funktionen **Above** evalueras för varje **Product**, och eftersom det bara finns två produkter, **AA** och **BB**, innehåller varje serie bara ett resultat som inte är null. I rad **BB** för månaden **Jan** är värdet för **Above(Sum (Sales))** 46. För rad **AA** är värdet null. Värdet i varje **AA**-rad, oavsett månad, är alltid null eftersom det inte finns något värde för **Product** över AA. Den andra serien värderas efter **AA** och **BB** för månaden **Feb**, för **Customer**-värdet **Astrida**. När alla månader har utvärderats för **Astrida** upprepas sekvensen för den andra **Customer**Betacab och så vidare.

Tabellvisualisering för exempel 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

Exempel 4

Example 4:	**Resultat**
Funktionen Above kan användas som indata för intervallfunktionerna. Till exempel: `RangeAvg (Above(Sum(Sales),1,3)).`	I argumenten för Above()-funktionen är offset inställt på 1 och count inställt på 3. Funktionen hittar resultaten av uttrycket Sum(Sales) på de tre raderna som är precis ovanför aktuell rad i kolumnsegmentet (där det finns en rad). De här tre värdena används som indata för funktionen RangeAvg() som räknar ut medelvärdet för ett angivet talintervall. En tabell med Customer som dimension ger följande resultat för uttrycket RangeAvg(). Astrida - Betacab 587 Canutility 563 Divadip: 603

Data som används i exempel:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
```

```
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Se även:

- *Below - diagramfunktion (page 647)*
- *Bottom - diagramfunktion (page 650)*
- *Top - diagramfunktion (page 675)*
- *RangeAvg (page 695)*

Below - diagramfunktion

Below() utvärderar ett uttryck på en rad under den aktuella raden inom ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden direkt nedanför. För diagram som inte är tabeller utvärderar **Below()** för raden nedanför den aktuella kolumnen i diagrammets raka tabellmotsvarighet.

Syntax:

```
Below([TOTAL] expr [ , offset [,count ]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Om en **offset**n större än 1 anges, flyttas utvärderingen av uttrycket n antal rader nedanför den aktuella raden. Om startpunkt anges till 0 utvärderas uttrycket på den aktuella raden. Anges ett negativt värde för startpunkten gör det att **Below**-funktionen fungerar likadant som **Above**-funktionen med motsvarande positivt värde för startpunkten.
count	Om en tredje parameter, **count**, anges till större än 1, returnerar funktionen ett intervall av **count**-värden: ett för varje **count**-tabellrad. Raderna räknas nedåt från den ursprungliga cellen. I denna form kan funktionen användas som argument i någon av de speciella intervallfunktionerna. *Intervallfunktioner (page 693)*

Argument	Beskrivning
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

På kolumnsegmentets sista rad returneras värdet NULL, eftersom det inte finns någon rad nedanför denna.

Ett kolumnsegment definieras som en konsekutiv underuppsättning med celler som har samma värden för dimensionerna i den aktuella sorteringsordningen. Postöverskridande diagramfunktioner beräknas i kolumnsegmentet exklusive dimensionen längst till höger i motsvarande raka tabelldiagram. Om det enbart finns en dimension i diagrammet, eller om kvalificeraren TOTAL anges, utvärderas uttrycket över en hel tabell.

Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner, utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

Begränsningar:

Rekursiva anrop returnerar NULL.

Exempel och resultat:

Example 1:

Tabellvisualisering för exempel 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	-	-	-	-

I tabellen som visas på skärmdumpen för exempel 1 skapas tabellvisualiseringen från dimensionen **Customer** och måtten: `Sum(Sales)` och `Below(Sum(Sales))`

Kolumnen **Below(Sum(Sales))** returnerar NULL för **Customer**-raden **Divadip**, eftersom det inte finns någon rad nedanför. Resultatet för raden **Canutility** visar värdet för Sum(Sales) för **Divadip**, resultatet för **Betacab** visar värdet för **Sum(Sales)** för **Canutility** och så vidare.

I tabellen visas även mer komplexa mått, som du kan se i kolumnerna med etiketterna: `Sum(Sales)+Below(Sum(Sales))`, **Below +Offset 3** och **Higher?**. Dessa uttryck fungerar på det sätt som beskrivs i nedanstående paragrafer.

För kolumnen med etiketten **Sum(Sales)+Below(Sum(Sales))** visar raden för **Astrida** resultatet av att **Sum(Sales)**-värdena lades till för raderna **Betacab** + **Astrida** (539+587). Resultatet för raden **Betacab** visar resultatet då **Sum(Sales)**-värden har lagts till för **Canutility** + **Betacab** (539+683).

Måttet med etiketten **Below +Offset 3** som skapades med uttrycket `Sum(Sales)+Below(Sum(Sales), 3)` har argumentet **offset**, angivet som 3. Effekten blir att värdet hämtas i raden tre rader nedanför den aktuella raden. Det lägger till **Sum(Sales)**-värdet för aktuell **Customer** på värdet från **Customer** tre rader längre ner. Värdena för de lägsta tre **Customer**-raderna är null.

Måttet med etiketten **Higher?** skapas av uttrycket: `IF(Sum(Sales)>Below(Sum(Sales)), 'Higher')`. Detta jämför värdena för den aktuella raden i måttet **Sum(Sales)** med raden nedanför den. Om den aktuella raden har ett större värde visas texten ”Higher”.

Funktionen kan även användas i andra diagram, inte bara tabeller – exempelvis i stapeldiagram.

För andra diagramtyper, ska du omvandla diagrammet till den raka tabellmotsvarigheten så att du enkelt kan tolka vilken rad funktionen är relaterad till.

För diagram med flera dimensioner beror resultatet för uttryck som innehåller funktionerna **Above**, **Below**, **Top** och **Bottom** på den ordning i vilken kolumndimensionerna sorteras av Qlik Sense. Qlik Sense evaluerar funktionerna baserat på de kolumnsegment som är resultatet från den dimension som kommer sist i sorteringsordningen. Sorteringsordningen för kolumner styrs från egenskapspanelen under **Sortering**. Den motsvarar inte nödvändigtvis den ordning i vilken kolumnerna visas i en tabell.Se Exempel: 2 i funktionen **Above** för mer information.

Exempel 2

Example 2:	**Resultat**
Funktionen **Below** kan användas som indata för intervallfunktionerna. Till exempel: `RangeAvg (Below(Sum(Sales),1,3))`.	I argumenten för **Below()**-funktionen är offset inställt på 1 och count inställt på 3. Funktionen hittar resultaten av uttrycket **Sum(Sales)** på de tre raderna som är precis nedanför aktuell rad i kolumnsegmentet (där det finns en rad). De här tre värdena används som indata för funktionen RangeAvg() som räknar ut medelvärdet för ett angivet talintervall. En tabell med **Customer** som dimension ger följande resultat för uttrycket RangeAvg().

Example 2:	Resultat	
	Astrida	659.67
	Betacab	720
	Canutility	757
	Divadip:	-

Data som används i exempel:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Se även:

- *Above - diagramfunktion (page 642)*
- *Bottom - diagramfunktion (page 650)*
- *Top - diagramfunktion (page 675)*
- *RangeAvg (page 695)*

Bottom - diagramfunktion

Bottom() utvärderar ett uttryck på den sista (nedersta) raden i ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden nederst. För diagram som inte är tabeller görs utvärderingen på den sista raden i den aktuella kolumnen i diagrammets raka tabellmotsvarighet.

Syntax:

```
Bottom([TOTAL] expr [ , offset [,count ]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Om en **offset**n större än 1 anges flyttas utvärderingen av uttrycket upp n antal rader ovanför den nedersta raden. Anges ett negativt värde för startpunkten gör det att **Bottom**-funktionen fungerar likadant som **Top**-funktionen med motsvarande positivt värde för startpunkten.
count	Om en tredje parameter, **count**, anges till större än 1, returnerar funktionen inte bara ett värde, utan ett intervall av **count**-värden: ett för var och en av de sista **count**-raderna i det aktuella kolumnsegmentet. I denna form kan funktionen användas som argument i någon av de speciella intervallfunktionerna. *Intervallfunktioner (page 693)*
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

Ett kolumnsegment definieras som en konsekutiv underuppsättning med celler som har samma värden för dimensionerna i den aktuella sorteringsordningen. Postöverskridande diagramfunktioner beräknas i kolumnsegmentet exklusive dimensionen längst till höger i motsvarande raka tabelldiagram. Om det enbart finns en dimension i diagrammet, eller om kvalificeraren TOTAL anges, utvärderas uttrycket över en hel tabell.

Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner, utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

Begränsningar:

Rekursiva anrop returnerar NULL.

Exempel och resultat:

Tabellvisualisering för exempel 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

I skärmdumpen av tabellen som visas i det här exemplet skapas tabellvisualiseringen från dimensionen **Customer** och måtten: `Sum(Sales)` och `Bottom(Sum(Sales))`.

Kolumnen **Bottom(Sum(Sales))** returnerar 757 för alla rader eftersom det är värdet för den nedersta raden: **Divadip**.

Tabellen visar även mer komplexa mått: ett skapat av `Sum(Sales)+Bottom(Sum(Sales))` och ett med etiketten **Bottom offset 3**, som skapas med hjälp av uttrycket `Sum(Sales)+Bottom(Sum(Sales), 3)` och vars argument **offset** är angivet som 3. Det lägger till **Sum(Sales)**-värdet för den aktuella raden till värdet från raden tre rader från den nedersta raden, d.v.s. den aktuella raden plus värdet för **Betacab**.

Exempel: 2

I tabellerna i skärmdumparna har fler dimensioner lagts till i visualiseringarna: **Month** och **Product**. För diagram med flera dimensioner beror resultatet för uttryck som innehåller funktionerna **Above**, **Below**, **Top** och **Bottom** på den ordning i vilken kolumndimensionerna sorteras av Qlik Sense. Qlik Sense evaluerar funktionerna baserat på de kolumnsegment som är resultatet från den dimension som kommer sist i sorteringsordningen. Sorteringsordningen för kolumner styrs från egenskapspanelen under **Sortering**. Den motsvarar inte nödvändigtvis den ordning i vilken kolumnerna visas i en tabell.

I den första tabellen utvärderas uttrycket baserat på **Month**. I den andra tabellen utvärderas det baserat på **Product**. Måttet **End value** innehåller uttrycket `Bottom(Sum(Sales))`. Den nedersta raden för **Month** är Dec och värdet för Dec, båda värdena för **Product**, som visas på skärmdumpen är 22. (Vissa rader har redigerats bort från skärmdumpen för att spara utrymme.)

Första tabell för Exempel 2. Värdet för Bottom för måttet End value baserat på 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

Andra tabell för Exempel 2. Värdet för Bottom för måttet End value baserat på Product (BB för 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

Se Exempel: 2 i funktionen **Above** för mer information.

Exempel 3

Exempel: 3	Resultat
Funktionen **Bottom** kan användas som indata för intervallfunktionerna. Till exempel: `RangeAvg (Bottom(Sum(Sales),1,3))`.	I argumenten för funktionen **Bottom()**, offset inställd på 1 och count är inställd på 3. Funktionen hittar resultatet för uttrycket **Sum(Sales)** på de tre raderna som börjar med raden över den nedersta raden i kolumnsegmentet (eftersom offset=1) och de två raderna över den (där det finns en rad). De här tre värdena används som indata för funktionen RangeAvg() som räknar ut medelvärdet för ett angivet talintervall. En tabell med **Customer** som dimension ger följande resultat för uttrycket 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 '|');
```

Se även:

Top - diagramfunktion (page 675)

Column - diagramfunktion

Column() returnerar det värde som återfinns i kolumnen som motsvarar **ColumnNo** i en rak tabell om dimensionerna ignoreras. Exempelvis returnerar **Column(2)** värdet för den andra måttkolumnen.

Syntax:

```
Column(ColumnNo)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
ColumnNo	Kolumnnumret för en kolumn i tabellen som innehåller ett mått. *Funktionen Column() ignorerar dimensionskolumner.*

Begränsningar:

Om **ColumnNo** refererar till en kolumn för vilken det inte finns något mått, returneras ett NULL-värde.

Rekursiva anrop returnerar NULL.

Exempel och resultat:

Exempel: Procent försäljning totalt

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
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

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
C	CC	19	-	0	505	0.00

Exempel: Procent försäljning för en vald kund

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

Exempel och resultat

Exempel	Resultat
Order Value läggs till i tabellen i form av ett mått med uttrycket: `Sum (UnitPrice*UnitSales)`. Total Sales Value läggs till i form av ett mått med uttrycket: `Sum(TOTAL UnitPrice*UnitSales)` % Sales läggs till i form av ett mått med uttrycket: `100*Column(1)/Column(2)`	Resultatet för Column(1) tas från kolumnen Order Value eftersom det är den första måttkolumnen. Resultatet för Column(2) tas från kolumnen Total Sales Value eftersom det är den andra måttkolumnen. Se resultaten i kolumnen % Sales i exemplet *Procent försäljning totalt (page 655)*.
Välj Customer A.	Urvalet förändrar värdet Total Sales Value, och därmed förändras %Sales. Se exemplet *Procent försäljning för en vald kund (page 656)*.

Data som används i exempel:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Dimensionality - diagramfunktion

Dimensionality() returnerar antalet dimensioner för den aktuella raden. Vad gäller pivottabeller returnerar funktionen det totala antalet dimensionskolumner som har icke-aggregerat innehåll, d.v.s. inte innehåller delsummor eller dolda aggregeringar.

Syntax:

```
Dimensionality ( )
```

Returnerad datatyp: heltal

Begränsningar:

Funktionen är endast tillgänglig i diagram. För alla diagramtyper utom pivottabeller returnerar den antalet dimensioner i alla rader utom summan, som blir 0.

Exempel: Diagramuttryck som använder Dimensionality

Exempel: Diagramuttryck
Funktionen Dimensionality() går att använda med en pivottabell som ett diagramuttryck där du vill använda olika cellformatering beroende på antalet dimensioner i en rad som har oaggregerade data. Det här exemplet använder funktionen Dimensionality() för att använda en bakgrundsfärg till tabellceller som motsvarar ett visst tillstånd.

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplet nedan.

```
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 ',');
```

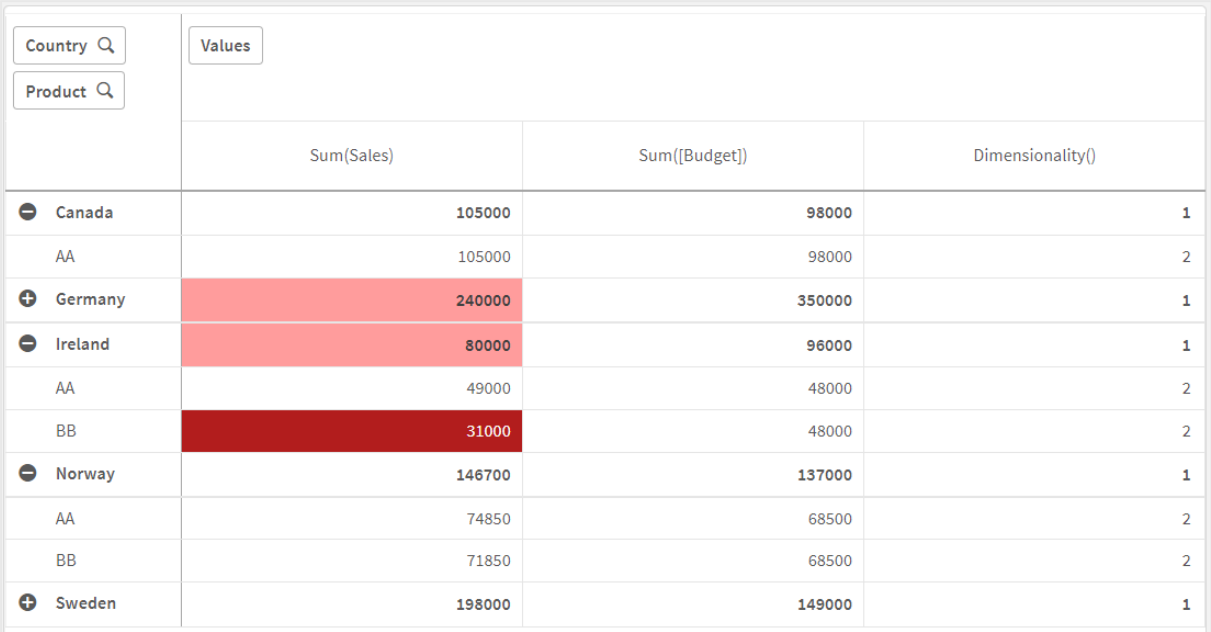
Diagramuttryck

Skapa en pivottabellsvisualisering i ett Qlik Sense-ark med **Country** och **Product** som dimensioner. Lägg till **Sum(Sales)**, **Sum(Budget)** och **Dimensionality()** som mått.

I panelen **Egenskaper** anger du följande uttryck som **Bakgrundsfärguttryck** för måttet **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) ))
```

Resultat:



Country / Product	Values		
	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

Förklaring

Uttrycket `If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156), If(Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)))` innehåller villkorssatser som kontrollerar Dimensionality-värdet och Sum(Sales) samt Sum(Budget) för varje produkt. Om villkoren uppfylls används en bakgrundsfärg på värdet Sum(Sales).

Exists

Exists() avgör om ett specifikt fältvärde redan har laddats in i fältet i dataladdningsskriptet. Funktionen returnerar TRUE eller FALSE, så att det kan användas i **where**-satsen för en **LOAD**-sats eller en **IF**-sats.

> *Du kan även använda **Not Exists()** för att avgöra om ett fältvärde inte har laddats, men du bör vara försiktig om du använder **Not Exists()** i en where-sats. **Exists()**-funktionen testar både tidigare laddade tabeller och tidigare laddade värden i den aktuella tabellen. Så endast den första förekomsten laddas. När den kommer till den andra förekomsten har värdet redan laddats. Se exemplen för mer information.*

Syntax:

```
Exists(field_name [, expr])
```

Returnerad datatyp: Boolesk

Argument:

Argument

Argument	Beskrivning
field_name	Namnet på fältet där du vill söka efter ett värde. Du kan använda explicita fältnamn utan citattecken. Fältet måste redan vara laddat av skriptet. Det vill säga, du kan inte hänvisa till ett fält som laddas i en sats längre ned i skriptet.
expr	Värdet som du vill kontrollera om det finns. Du kan använda ett explicit värde eller ett uttryck som hänvisar till ett eller flera fält i den aktuella load-satsen. *Du kan inte hänvisa till fält som inte ingår i den aktuella load-satsen.* Det här argumentet är valfritt Om du utesluter det kommer funktionen att kontrollera om värdet **field_name** i den aktuella posten redan existerar.

Exempel och resultat:

Exempel 1

```
Exists (Employee)
```

Returnerar -1 (True) om fältets värde **Employee** i den aktuella posten redan finns i en tidigare läst post som innehåller detta fält.

Satserna `Exists (Employee, Employee)` och `Exists (Employee)` är ekvivalenta.

Exempel 2

```
Exists(Employee, 'Bill')
```

Returnerar -1 (True) om fältvärdet **'Bill'** hittas i det aktuella innehållet i fältet **Employee**.

Exempel 3

```
Employees: LOAD * inline [ Employee|ID|Salary Bill|001|20000 John|002|30000 Steve|003|35000 ]
(delimiter is '|');  Citizens: Load * inline [ Employee|Address Bill|New York Mary|London
Steve|Chicago Lucy|Madrid Lucy|Paris John|Miami ] (delimiter is '|') where Exists (Employee);
Drop Tables Employees;
```

Detta resulterar i en tabell som du kan använda i en tabellvisualisering med dimensionerna Employee och Address.

where-satsen, `where Exists (Employee)`, betyder att enbart namnen från tabellen Citizens som också finns i Employees laddas till den nya tabellen. Satsen Drop avlägsnar tabellen Employees för att undvika sammanblandning.

Resultat

Employee	Address
Bill	New York
John	Miami
Steve	Chicago

Exempel 4

```
Employees: Load * inline [  Employee|ID|Salary Bill|001|20000 John|002|30000 Steve|003|35000 ] (delimiter is '|');   Citizens: Load * inline [ Employee|Address Bill|New York Mary|London Steve|Chicago Lucy|Madrid Lucy|Paris John|Miami ] (delimiter is '|') where not Exists (Employee);   Drop Tables Employees;
```

where-satsen innehåller not: `where not Exists (Employee).`

Detta innebär att enbart namnen från tabellen Citizens som inte finns i Employees laddas in i den nya tabellen.

Observera att det finns två värden för Lucy i tabellen Citizens, men att endast ett inkluderas i resultattabellen. När du laddar den första raden med värdet Lucy inkluderas det i fältet Employee. När den andra raden kontrolleras finns alltså redan värdet.

Resultat

Employee	Address
Mary	London
Lucy	Madrid

Exempel 5

Det här exemplet visar hur du laddar alla värden.

```
Employees: Load Employee As Name; LOAD * inline [ Employee|ID|Salary Bill|001|20000 John|002|30000 Steve|003|35000 ] (delimiter is '|');   Citizens: Load * inline [ Employee|Address Bill|New York Mary|London Steve|Chicago Lucy|Madrid Lucy|Paris John|Miami ] (delimiter is '|') where not Exists (Name, Employee);   Drop Tables Employees;
```

För att hämta alla värden för Lucy ändrades två saker:

- En tidigare laddning i tabellen Employees infogades där Employee döptes om till Name.
  `Load Employee As Name;`

- Where-villkoret i Citizens ändrades till:
  `not Exists (Name, Employee).`

  Det skapar fält för Name och Employee. När den andra raden med Lucy kontrolleras finns den fortfarande inte i Name.

Resultat

Employee	Address
Mary	London
Lucy	Madrid
Lucy	Paris

FieldIndex

FieldIndex() returnerar den placering som fältvärdet **value** har i fältet **field_name** (i laddningsordning).

Syntax:

```
FieldIndex(field_name , value)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
field_name	Namnet på det fält för vilket index krävs. Till exempel, kolumnen i en tabell. Måste anges som ett strängvärde. Detta innebär att fältnamnet måste omges av enkla citattecken.
value	Värdet för fältet **field_name**.

Begränsningar:

Om **value** inte påträffas bland fältvärdena i fältet **field_name** returneras 0.

Exempel och resultat:

I följande exempel används fältet: **First name** från tabellen **Names**.

Exempel och resultat

Exempel	Resultat
Lägg till exempeldata i appen och kör den.	Tabellen **Names** är laddad, som i exempeldata,

Exempel	Resultat
Diagramfunktion: I en tabell som innehåller dimensionen First name, lägger du till följande som mått:	
`FieldIndex ('First name','John')`	1, eftersom John visas först i laddningsordningen för fältet **First name**. Observera att i en filterruta skulle **John** visas som nummer 2 från toppen eftersom den sorteras i alfabetisk ordning och inte i laddningsordning.
`FieldIndex ('First name','Peter')`	4, eftersom **FieldIndex()** returnerar endast ett värde, nämligen den första förekomsten i laddningsordningen.
Skriptfunktion: Då tabellen **Names** är laddad, som i exempeldata:	

`John1:`
 	
`Load FieldIndex('First name','John') as MyJohnPos`
 	
`Resident Names;`
	`MyJohnPos=1`, eftersom John visas först i laddningsordningen för fältet **First name**. Observera att i en filterruta skulle **John** visas som nummer 2 från toppen eftersom den sorteras i alfabetisk ordning och inte i laddningsordning.

`Peter1:`
 	
`Load FieldIndex('First name','Peter') as MyPeterPos`
 	
`Resident Names;`
	`MyPeterPos=4`, eftersom **FieldIndex()** returnerar endast ett värde, nämligen den första förekomsten i laddningsordningen.

Data som används i exemplet:

```
Names: LOAD * inline [ First name|Last name|Initials|Has cellphone John|Anderson|JA|Yes
Sue|Brown|SB|Yes Mark|Carr|MC|No Peter|Devonshire|PD|No Jane|Elliot|JE|Yes Peter|Franc|PF|Yes
] (delimiter is '|');                    John1: Load FieldIndex('First name','John') as MyJohnPos Resident
Names;   Peter1: Load FieldIndex('First name','Peter') as MyPeterPos Resident Names;
```

FieldValue

FieldValue() returnerar det värde som påträffas vid placeringen **elem_no** för fältet **field_name** (i laddningsordning).

Syntax:

```
FieldValue(field_name , elem_no)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
field_name	Namnet på det fält för vilket värdet krävs. Till exempel, kolumnen i en tabell. Måste anges som ett strängvärde. Detta innebär att fältnamnet måste omges av enkla citattecken.
elem_no	Lägesnumret (elementnumret) på fältet, efter laddningsordning, som värdet returneras för. Detta kan motsvara en rad i en tabell, men det beror på i vilken ordning elementen (raderna) läses in.

Begränsningar:

Om **elem_no** är större än antalet fältvärden returneras NULL.

Exempel

Laddningsskript

Ladda följande data som en inline load i Skriptredigeraren för att skapa exemplet nedan.

```
Names:                                LOAD * inline [ First name|Last name|Initials|Has cellphone John|And
Sue|Brown|SB|Yes Mark|Carr|MC |No Peter|Devonshire|PD|No Jane|Elliot|JE|Yes Peter|Franc|PF|Yes
] (delimiter is '|');                                    John1:                                                Load
Names;  Peter1:  Load FieldValue('First name',5) as MyPos2 Resident Names;
```

Skapa en visualisering

Skapa en tabellvisualisering i ett Qlik Sense-ark. Lägg till fälten **First name**, **MyPos1** och **MyPos2** i tabellen.

Resultat

First name	MyPos1	MyPos2
Jane	John	Jane
John	John	Jane
Mark	John	Jane
Peter	John	Jane
Sue	John	Jane

Förklaring

FieldValue('First name','1') leder till John som värde för **MyPos1** för alla förnamn eftersom John visas först i laddningsordningen för fältet **First name**. Observera att John skulle visas som nummer 2 från toppen i en filterruta, efter Jane, eftersom den sorteras alfabetiskt och inte efter inläsningsordning.

FieldValue('First name','5') leder till Jane som värde för **MyPos2** för alla förnamn eftersom Jane visas som det femte i laddningsordningen för fältet **First name**.

FieldValueCount

FieldValueCount() är en **heltal**sfunktion som returnerar antalet distinkta värden i ett fält.

En partiell inläsning kan ta bort värden från data, vilka inte kommer att återspeglas i antalet returnerade. Det returnerade numret kommer att motsvara alla distinkta värden som lästes in, antingen i den första inläsningen eller någon efterföljande partiell inläsning.

Syntax:

```
FieldValueCount(field_name)
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
field_name	Namnet på det fält för vilket värdet krävs. Till exempel, kolumnen i en tabell. Måste anges som ett strängvärde. Detta innebär att fältnamnet måste omges av enkla citattecken.

Exempel och resultat:

I följande exempel används fältet **First name** från tabellen **Names**.

Exempel och resultat

Exempel	Resultat
Lägg till exempeldata i appen och kör den.	Tabellen **Names** är laddad, som i exempeldata,
Diagramfunktion: I en tabell som innehåller dimensionen First name, lägger du till följande som mått:	
`FieldValueCount('First name')`	5 eftersom **Peter** visas två gånger.
`FieldValueCount('Initials')`	6 eftersom **Initials** endast innehåller distinkta värden.

Exempel	Resultat
Skriptfunktion: Då tabellen **Names** är laddad, som i exempeldata:	

`FieldCount1:`
 	
`Load FieldValueCount('First name') as MyFieldCount1`
 	
`Resident Names;`
	`MyFieldCount1`=5, eftersom ”Peter” visas två gånger

`FieldCount2:`
 	
`Load FieldValueCount('Initials') as MyInitialsCount1`
 	
`Resident Names;`
	`MyFieldCount1`=6, eftersom ”Initials” endast innehåller distinkta värden.

Data som används i exempel:

```
Names: LOAD * inline [ First name|Last name|Initials|Has cellphone John|Anderson|JA|Yes
Sue|Brown|SB|Yes Mark|Carr|MC|No Peter|Devonshire|PD|No Jane|Elliot|JE|Yes Peter|Franc|PF|Yes
] (delimiter is '|');                    FieldCount1: Load FieldValueCount('First name') as MyFieldCount1
Resident Names;   FieldCount2: Load FieldValueCount('Initials') as MyInitialsCount1 Resident
Names;
```

LookUp

Lookup() letar in i en tabell som redan är laddad och returnerar värdet av **field_name** vilket motsvarande den första förekomsten av värdet **match_field_value** i fältet **match_field_name**. Tabellen kan vara den aktuella tabell eller en annan tabell som har laddats.

Syntax:

```
lookup(field_name, match_field_name, match_field_value [, table_name])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
field_name	Namnet på det fält för vilket returvärdet krävs. Inmatat värde måste anges som en sträng (t ex en litteral inom citationstecken).
match_ field_name	Namnet på fältet som **match_field_value** ska sökas i. Inmatat värde måste anges som en sträng (t ex en litteral inom citationstecken).
match_ field_value	Värdet som ska sökas upp i **match_field_name**-fältet.
table_name	Namnet på tabellen där värdet ska sökas. Inmatat värde måste anges som en sträng (t.ex. en litteral inom citationstecken). Om **table_name** utelämnas, antas aktuell tabell.

Argument utan citattecken syftar på den aktuella tabellen. Om du vill hänvisa till andra tabeller omger du ett argument med enkla citattecken.

Begränsningar:

Sökordningen är densamma som laddningsordningen, såvida inte tabellen är resultatet av komplexa operationer som join-operationer. I så fall är ordningen inte väldefinierad. Både **field_name** och **match_field_name** måste vara fält i samma tabell, som angivet med **table_name**.

Om ingen motsvarighet påträffas, returneras NULL.

Exempel

Laddningsskript

Ladda följande data som en inline load i Skriptredigeraren för att skapa exemplet nedan.

```
ProductList: Load * Inline [ ProductID|Product|Category|Price 1|AA|1|1 2|BB|1|3 3|CC|2|8
4|DD|3|2 ] (delimiter is '|');   OrderData: Load *, Lookup('Category', 'ProductID', ProductID,
'ProductList') as CategoryID Inline [ InvoiceID|CustomerID|ProductID|Units 1|Astrida|1|8
1|Astrida|2|6 2|Betacab|3|10 3|Divadip|3|5 4|Divadip|4|10 ] (delimiter is '|');   Drop Table
ProductList;
```

Skapa en visualisering

Skapa en tabellvisualisering i ett Qlik Sense-ark. Lägg till fälten **ProductID**, **InvoiceID**, **CustomerID**, **Units** och **CategoryID** i tabellen.

Resultat

Resultattabell

ProductID	InvoiceID	CustomerID	Units	CategoryID
1	1	Astrida	8	1
2	1	Astrida	6	1
3	2	Betacab	10	2
3	3	Divadip	5	2
4	4	Divadip	10	3

Förklaring

Exempeldata använder **Lookup()**-funktionen i följande form:

```
Lookup('Category', 'ProductID', ProductID, 'ProductList')
```

ProductList-tabellen laddas föst.

Lookup()-funktionen används för att bygga **OrderData**-tabellen. Den anger det tredje argumentet som **ProductID**. Detta är det fält för vilket en sökning efter värdet ska göras i det andra argumentet **'ProductID'** i **ProductList** enligt vad som anges av de omgivande enkla citattecknen.

Funktionen returnerar värdet för "**Category**" (i tabellen **ProductList**), laddad som **CategoryID**.

Satsen **drop** raderar tabellen **ProductList** från datamodellen eftersom den inte behövs, vilket ger tabellen **OrderData** som resultat.

Lookup()-funktionen är flexibel och kan komma åt alla tabeller som laddats tidigare. Det går dock långsamt jämfört med Applymap()-funktionen.

Se även:

ApplyMap (page 686)

NoOfRows - diagramfunktion

NoOfRows() returnerar antalet rader i det aktuella kolumnsegmentet i en tabell. För bitmappsdiagram returnerar **NoOfRows()** antalet rader i diagrammets raka tabellmotsvarighet.

Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner, utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

Syntax:

```
NoOfRows([TOTAL])
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

Exempel: Diagramuttryck som använder NoOfRows

Exempel – diagramuttryck

Laddningsskript

Ladda följande data som en inline-laddning i Skriptredigeraren för att skapa diagramuttrycksexemplen nedan.

```
Temp: LOAD * inline [ Region|SubRegion|RowNo()|NoOfRows() Africa|Eastern Africa|Western
Americas|Central Americas|Northern Asia|Eastern Europe|Eastern Europe|Northern Europe|Western
Oceania|Australia ] (delimiter is '|');
```

Diagramuttryck

Skapa en tabellvisualisering i ett Qlik Sense-ark med **Region** och **SubRegion** som dimensioner. Lägg till `RowNo( ), NoOfRows()` och `NoOfRows(Total)` som mått.

Resultat

Region	SubRegion	RowNo()	NoOfRows()	NoOfRows (Total)
Africa	Eastern	1	2	9
Africa	Western	2	2	9
Americas	Central	1	2	9
Americas	Northern	2	2	9
Asia	Eastern	1	1	9
Europe	Eastern	1	3	9
Europe	Northern	2	3	9
Eurrope	Western	3	3	9
Oceania	Australia	1	1	9

Förklaring

I det här exemplet är sorteringsordningen efter den första dimensionen, Region. Det leder till att varje kolumnsegment består av en grupp med regioner som har samma värde, till exempel Africa.

Kolumnen **RowNo()** visar radnummer för varje kolumnsegment. Till exempel finns det två rader för regionen Africa. Radnumreringen börjar sedan på 1 igen för nästa kolumnsegment, som är Americas.

Kolumnen **NoOfRows()** räknar antalet rader i varje kolumnsegment. Till exempel har Europe tre rader i kolumnsegmentet.

Kolumnen **NoOfRows(Total)** bortser från dimensionerna på grund av argumentet `TOTAL` för `NoOfRows()` och räknar raderna i tabellen.

Om tabellen sorteras på den andra dimensionen, SubRegion, baseras kolumnsegment på den dimensionen så att radnumreringen ändras för varje SubRegion.

Se även:

RowNo - diagramfunktion (page 442)

Peek

Peek() returnerar värdet för ett fält i en tabell för en rad som redan har laddats. Radnumret kan anges, liksom tabellen. Om inget radnummer anges används posten som laddades senast.

Peek()-funktionen används vanligen för att hitta de relevanta gränserna i en tidigare laddad tabell, det vill säga det första eller sista värdet i ett specifikt fält. I de flesta fall lagras det här värdet i en variabel för att användas senare, till exempel som ett villkor i en do-while-loop.

Syntax:

```
Peek(
field_name
[, row_no[, table_name ] ])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
field_name	Namnet på det fält för vilket returvärdet krävs.Inmatat värde måste anges som en sträng (t ex en litteral inom citationstecken).
row_no	Den rad i tabellen som anger det fält som krävs. Kan vara ett uttryck, men måste lösas till ett heltal. 0 anger första posten, 1 andra posten o.s.v. Negativa tal markerar ordningen från slutet av tabellen. -1 anger den senaste lästa posten. Om ingen **row_no** angivits, antas -1.
table_name	En tabelletikett utan avslutande kolon. Om inget **table_name** har angetts antas den aktuella tabellen. Om det används utanför **LOAD**-satsen, eller refererar till en annan tabell, måste tabellnamnet **table_name** anges.

Begränsningar:

Funktionen kan bara returnera värden från poster som redan laddats. Det betyder att ett anrop som använder -1 som row_no kommer att returnera NULL för den första posten i en tabell.

Exempel och resultat:

Exempel 1

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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;
```

Resultattabell

Employee code	StartDate	EndDate	FirstCode	LastCode
101	02/11/2010	23/06/2012	101	106
102	01/11/2011	30/11/2013	101	106
103	02/01/2012		101	106
104	02/01/2012	31/03/2012	101	106
105	01/04/2012	31/01/2013	101	106
106	02/11/2013		101	106

FirstCode = 101, eftersom `Peek('EmployeeCode',0, 'EmployeeDates')` returnerar det första värdet för EmployeeCode i tabellen EmployeeDates.

LastCode = 106, eftersom `Peek('EmployeeCode',-1, 'EmployeeDates')` returnerar det sista värdet för EmployeeCode i tabellen EmployeeDates.

Att ersätta värdet på argumentet **row_no** returnerar värdena för andra rader i tabellen, enligt följande:

`Peek('EmployeeCode',2, 'EmployeeDates')` returnerar det tredje värdet, 103, i tabellen som FirstCode.

Observera dock att om inte tabellen specificeras som det tredje argumentet **table_name** i dessa exempel, refererar funktionen till den aktuella tabellen (i detta fall, internt).

Exempel 2

Om du vill komma åt data längre ned i en tabell kan du göra det i två steg: först laddar du hela tabellen i en temporär tabell och sedan sorterar du om den med användning av **Peek()**.

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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.

Resultattabell

ID	Lista	Värde
1	3,4	4
1	3,4,6	6
1	3	3
2	1,11	11

ID	Lista	Värde
2	1	1
3	7,8	8
3	7	7
5	2,78	78
5	2,78,13	13
5	2	2

Satsen **IF()** byggs från den tillfälliga tabellen T1.
`Peek('ID')` hänvisar till fältet ID i föregående rad i aktuell tabell T2.
`Peek('List')` hänvisar till fältet List i föregående rad i tabellen T2, som byggs då uttrycket utvärderas.

Satsen utvärderas enligt följande:
Om det aktuella värdet för ID är samma som det tidigare värdet för ID, då ska värdet för Peek('List') skrivas sammanlänkat med det aktuella värdet för Value. Annars skriver du det aktuella värdet av Value endast.

Om Peek('List') redan innehåller ett sammanlänkat resultat, kommer det nya resultatet Peek('List') sammanlänkas till det.

*Observera satsen **Order by**. Detta anger hur tabellen ordnas (genom ID i stigande ordning). Utan detta använder funktionen Peek() godtycklig ordning som den interna tabellen har, vilket kan leda till oförutsägbara resultat.*

Exempel 3

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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];
```

Resultattabell

Amount	AmountMonthBefore	Månad
1	4	2022-06
2	-	2022-01
3	2	2022-02
4	9	2022-05
7	3	2022-03
9	7	2022-04

Fältet AmountMonthBefore kommer att innehålla mängden från föregående månad.

Eftersom parametrarna row_no och table_name är utelämnade, används standardvärdena. I det här exemplet är följande tre funktionsanrop likvärdiga:

- Peek(Amount)
- Peek(Amount,-1)
- Peek(Amount,-1,'Amounts')

När -1 används som row_no kommer värdet från föregående rad att användas. Genom att ersätta det här värdet går det att hämta värden för de andra raderna i tabellen:

Peek(Amount,2) returnerar det tredje värdet i tabellen: 7.

Exempel 4

Data måste vara korrekt sorterade för att få rätt resultat men det är tyvärr inte alltid fallet. Funktionen Peek () kan inte användas för att hänvisa till data som inte har laddats än. Du kan undvika sådana problem genom att använda temporära tabeller och köra flera pass för dina data.

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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;
```

Förklaring

Den inledande tabellen är sorterad på månad. Det betyder att peek()-funktionen i många fall skulle returnera mängden för fel produkt. Det innebär att tabellen måste sorteras om. Detta utförs genom att köra ett andra pass för dina data och skapa den nya tabellen tmp2Amounts. Observera satsen Order By. Först ordnar den posterna efter produkt och därefter efter månad i stigande ordning.

If()-funktionen behövs eftersom AmountMonthBefore endast ska beräknas om den föregående raden innehåller data för samma produkt men bara för den föregående månaden. Genom att jämföra produkten på den aktuella raden med produkten på föregående rad kan det här villkoret valideras.

När den andra tabellen har skapats skippas den första tabellen tmp1Amounts med en Drop Table-sats.

Slutligen körs ett tredje pass för data, men nu med månaderna sorterade i omvänd ordning. Det betyder att även AmountMonthAfter kan beräknas.

Order By-satser anger hur tabellen ska ordnas. Utan detta använder funktionen Peek() en godtycklig ordning som den interna tabellen har, vilket kan leda till oförutsägbara resultat.

Resultat

Resultattabell

Månad	Produkt	Amount	AmountMonthBefore	AmountMonthAfter
2022-01	En	8	-	6
2022-02	B	3	-	4
2022-03	En	6	8	6
2022-04	B	4	3	1
2022-05	En	6	6	5
2022-06	B	1	4	5
2022-01	En	5	6	7
2022-02	B	5	1	6
2022-03	En	7	5	4
2022-04	B	6	5	8
2022-05	En	4	7	-
2022-06	B	8	6	-

Exempel 5

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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;
```

Resultat

Resultattabell

Kvartal	SumVal	AccSumVal
2003q1	65000	65000
2003q2	132450	197450
2003q3	131840	329290
2003q4	9000	338290
2004q1	5250	343540

Kvartal	SumVal	AccSumVal
2004q2	24240	367780
2004q3	88150	455930
2004q4	220650	676580

Förklaring

Load-satsen **Load *, rangesum(SumVal,peek('AccSumVal')) as AccSumVal** har ett rekursivt anrop där de tidigare värdena läggs till i det aktuella värdet. Den här åtgärden används till att beräkna en ackumulering av värden i skriptet.

Se även:

Previous

Previous() returnerar värdet av uttrycket **expr** genom att använda data från en tidigare indatapost som inte uteslutits till följd av en **where**-sats. För den första posten i en intern tabell kommer funktionen att returnera NULL.

Syntax:

```
Previous(expr)
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas. Uttrycket kan innehålla kapslade funktioner **previous()** för att komma åt poster längre bak. Data hämtas direkt från indatakällan och man kan således referera även till fält som inte lästs in i Qlik Sense, dvs. även om de inte har lagrats i dess associativa databas.

Begränsningar:

För den första posten i en intern tabell kommer funktionen att returnera NULL.

Exempel:

Skriv följande i ditt laddningsskript

```
Sales2013:
Load *, (Sales - Previous(Sales) )as Increase Inline [
Month|Sales
1|12
```

```
2|13
3|15
4|17
5|21
6|21
7|22
8|23
9|32
10|35
11|40
12|41
] (delimiter is '|');
```

Genom att använda funktionen **Previous()** i satsen **Load** kan vi jämföra det aktuella värdet av Sales med föregående värde och använda det i ett tredje fält, Increase.

Resultattabell

Månad	Försäljning	Ökning
1	12	-
2	13	1
3	15	2
4	17	2
5	21	4
6	21	0
7	22	1
8	23	1
9	32	9
10	35	3
11	40	5
12	41	1

Top - diagramfunktion

Top() utvärderar ett uttryck på den första (översta) raden i ett kolumnsegment i en tabell. Vilken rad som den beräknas för beror på värdet för **offset**, om sådant finns, standardvärdet är raden högst upp. För diagram som inte är tabeller görs **Top()**-utvärderingen på den första raden i den aktuella kolumnen i diagrammets raka tabellmotsvarighet.

Syntax:

```
Top([TOTAL] expr [ , offset [,count ]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Om en startpunkt, **offset**, på n större än 1 anges flyttas utvärderingen av uttrycket n rader nedanför den översta raden. Anges ett negativt värde för startpunkten gör det att **Top**-funktionen fungerar likadant som **Bottom**-funktionen med motsvarande positivt värde för startpunkten.
count	Om en tredje parameter, **count**, anges till större än 1, returnerar funktionen ett intervall av **count**-värden, ett för var och en av de sista **count**-raderna i det aktuella kolumnsegmentet. I denna form kan funktionen användas som argument i någon av de speciella intervallfunktionerna. *Intervallfunktioner (page 693)*
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

Ett kolumnsegment definieras som en konsekutiv underuppsättning med celler som har samma värden för dimensionerna i den aktuella sorteringsordningen. Postöverskridande diagramfunktioner beräknas i kolumnsegmentet exklusive dimensionen längst till höger i motsvarande raka tabelldiagram. Om det enbart finns en dimension i diagrammet, eller om kvalificeraren TOTAL anges, utvärderas uttrycket över en hel tabell.

Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner, utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

Begränsningar:

Rekursiva anrop returnerar NULL.

Exempel och resultat:

Exempel: 1

I skärmdumpen av tabellen som visas i det här exemplet skapas tabellvisualiseringen från dimensionen **Customer** och måtten: `Sum(Sales)` och `Top(Sum(Sales))`.

Kolumnen **Top(Sum(Sales))** returnerar 587 för alla rader eftersom det är värdet för den översta raden: **Astrida**

Tabellen visar även mer komplexa mått: ett skapat av `Sum(Sales)+Top(Sum(Sales))` och ett med etiketten **Top offset 3**, som skapas med hjälp av uttrycket `Sum(Sales)+Top(Sum(Sales), 3)` och vars argument **offset** är angivet som 3. Det lägger till **Sum(Sales)**-värdet för den aktuella raden till värdet från raden tre rader nedanför den översta raden, d.v.s. den aktuella raden plus värdet för **Canutility**.

Exempel 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

Exempel: 2

I tabellerna i skärmdumparna har fler dimensioner lagts till i visualiseringarna: **Month** och **Product**. För diagram med flera dimensioner beror resultatet för uttryck som innehåller funktionerna **Above**, **Below**, **Top** och **Bottom** på den ordning i vilken kolumndimensionerna sorteras av Qlik Sense. Qlik Sense evaluerar funktionerna baserat på de kolumnsegment som är resultatet från den dimension som kommer sist i sorteringsordningen. Sorteringsordningen för kolumner styrs från egenskapspanelen under **Sortering**. Den motsvarar inte nödvändigtvis den ordning i vilken kolumnerna visas i en tabell.

Första tabell för Exempel 2. Värdet för Top för måttet First value baserat på 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

Andra tabell för Exempel 2. Värdet för Top för måttet First value baserat på Product (AA för 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

Se Exempel: 2 i funktionen **Above** för mer information.

Exempel 3

Exempel: 3	**Resultat**
Funktionen **Top** kan användas som indata för intervallfunktionerna. Till exempel: `RangeAvg (Top (Sum(Sales),1,3))`.	I argumenten för funktionen **Top()**, offset inställd på 1 och count är inställd på 3. Funktionen hittar resultatet för uttrycket**Sum(Sales)** på de tre raderna som börjar med raden under den nedersta raden i kolumnsegmentet (eftersom offset=1) och de två raderna under den (där det finns en rad). De här tre värdena används som indata för funktionen RangeAvg() som räknar ut medelvärdet för ett angivet talintervall. En tabell med **Customer** som dimension ger följande resultat för uttrycket 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 '|');
```

Se även:

- *Bottom - diagramfunktion (page 650)*
- *Above - diagramfunktion (page 642)*
- *Sum - diagramfunktion (page 228)*
- *RangeAvg (page 695)*
- *Intervallfunktioner (page 693)*

SecondaryDimensionality - diagramfunktion

SecondaryDimensionality() returnerar antalet dimensionspivottabellrader som har icke-aggregerat innehåll, d.v.s. inte innehåller delsummor eller dolda aggregeringar. Denna funktion motsvarar funktionen **dimensionality()** för horisontella pivottabelldimensioner.

Syntax:

```
SecondaryDimensionality( )
```

Returnerad datatyp: heltal

Begränsningar:

Såvida funktionen **SecondaryDimensionality** inte används i pivottabeller returnerar den alltid 0.

After - diagramfunktion

After() returnerar värdet för ett uttryck utvärderat med en pivottabells dimensionsvärden som de visas i kolumnen efter den aktuella kolumnen inom ett radsegment i pivottabellen.

Syntax:

```
after([TOTAL] expr [, offset [, count ]])
```

Funktionen returnerar NULL i alla diagramtyper utom pivottabeller.

Argument:

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Om en **offset** n större än 1 anges, flyttas utvärderingen av uttrycket n antal rader till höger om den aktuella raden. Om startpunkt anges till 0 utvärderas uttrycket på den aktuella raden. Anges ett negativt värde för startpunkten gör det att **After**-funktionen fungerar likadant som **Before**-funktionen med motsvarande positivt värde för startpunkten.
count	Om en tredje parameter **count** anges till större än 1 returnerar funktionen ett intervall av värden, ett för var och en av tabellraderna upp till värdet för **count** räknat åt höger från den ursprungliga cellen.
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

I radsegmentets sista kolumn kommer värdet NULL att returneras, eftersom det inte finns någon cell efter denna.

Om pivottabellen har flera horisontella dimensioner, inbegriper det aktuella radsegmentet endast kolumner som har samma värden som den aktuella kolumnen i samtliga dimensionsrader utom den rad som visar den sista dimensionen i fältens inbördes sorteringsordning. Fältens inbördes sorteringsordning för horisontella dimensioner i pivottabeller definieras helt enkelt som dimensionernas ordning uppifrån och ned.

Exempel:

```
after( sum( Sales ))
after( sum( Sales ), 2 )
after( total sum( Sales ))
```

`rangeavg (after(sum(x),1,3))` returnerar medelvärdet för de tre resultaten av **sum(x)**-funktionen utvärderat på de tre kolumnerna omedelbart till höger om den aktuella kolumnen.

Before - diagramfunktion

Before()returnerar värdet av ett uttryck utvärderat men en pivottabells dimensionsvärden som de visas i kolumnen framför den aktuella kolumnen inom ett radsegment i pivottabellen.

Syntax:

```
before([TOTAL] expr [, offset [, count]])
```

Funktionen returnerar NULL i alla diagramtyper utom pivottabeller.

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Om en **offset** n större än 1 anges, flyttas utvärderingen av uttrycket n antal rader till vänster om den aktuella raden. Om startpunkt anges till 0 utvärderas uttrycket på den aktuella raden. Anges ett negativt värde för startpunkten gör det att **Before**-funktionen fungerar likadant som **After**-funktionen med motsvarande positivt värde för startpunkten.
count	Om en tredje parameter **count** anges till större än 1 returnerar funktionen ett intervall av värden, ett för var och en av tabellraderna upp till värdet för **count** räknat åt vänster från den ursprungliga cellen.
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

I radsegmentets första kolumn kommer värdet NULL att returneras eftersom det inte finns någon kolumn före denna.

Om pivottabellen har flera horisontella dimensioner, inbegriper det aktuella radsegmentet endast kolumner som har samma värden som den aktuella kolumnen i samtliga dimensionsrader utom den rad som visar den sista dimensionen i fältens inbördes sorteringsordning.Fältens inbördes sorteringsordning för horisontella dimensioner i pivottabeller definieras helt enkelt som dimensionernas ordning uppifrån och ned.

Exempel:

```
before( sum( Sales ))
before( sum( Sales ), 2 )
before( total sum( Sales ))
```

`rangeavg (before(sum(x),1,3))` returnerar medelvärdet för de tre resultaten av **sum(x)**-funktionen utvärderad på de tre kolumnerna omedelbart till vänster om den aktuella kolumnen.

First - diagramfunktion

First() returnerar värdet för ett uttryck utvärderat med en pivottabells dimensionsvärden som de visas i den första kolumnen i det aktuella radsegmentet i pivottabellen. Funktionen returnerar NULL i alla diagramtyper utom pivottabeller.

Syntax:

```
first([TOTAL] expr [, offset [, count]])
```

Argument:

Argument

Argument	Beskrivning
expression	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Om en **offset** n större än 1 anges, flyttas utvärderingen av uttrycket n antal rader till höger om den aktuella raden. Om startpunkt anges till 0 utvärderas uttrycket på den aktuella raden. Anges ett negativt värde för startpunkten gör det att **First**-funktionen fungerar likadant som **Last**-funktionen med motsvarande positivt värde för startpunkten.
count	Om en tredje parameter **count** anges till större än 1 returnerar funktionen ett intervall av värden, ett för var och en av tabellraderna upp till värdet för **count** räknat åt höger från den ursprungliga cellen.
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

Om pivottabellen har flera horisontella dimensioner, inbegriper det aktuella radsegmentet endast kolumner som har samma värden som den aktuella kolumnen i samtliga dimensionsrader utom den rad som visar den sista dimensionen i fältens inbördes sorteringsordning.Fältens inbördes sorteringsordning för horisontella dimensioner i pivottabeller definieras helt enkelt som dimensionernas ordning uppifrån och ned.

Exempel:

```
first( sum( Sales ))
first( sum( Sales ), 2 )
first( total sum( Sales )
```

`rangeavg (first(sum(x),1,5))` returnerar ett medelvärde av resultaten från funktionen **sum(x)** utvärderat på de fem kolumnerna längst till vänster i det aktuella radsegmentet.

Last - diagramfunktion

Last() returnerar värdet för ett uttryck utvärderat med en pivottabells dimensionsvärden som de visas i den sista kolumnen i det aktuella radsegmentet i pivottabellen. Funktionen returnerar NULL i alla diagramtyper utom pivottabeller.

Syntax:

```
last([TOTAL] expr [, offset [, count]])
```

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
offset	Om en **offset** n större än 1 anges, flyttas utvärderingen av uttrycket n antal rader till vänster om den aktuella raden. Om startpunkt anges till 0 utvärderas uttrycket på den aktuella raden. Anges ett negativt värde för startpunkten gör det att **First**-funktionen fungerar likadant som **Last**-funktionen med motsvarande positivt värde för startpunkten.
count	Om en tredje parameter **count** anges till större än 1 returnerar funktionen ett intervall av värden, ett för var och en av tabellraderna upp till värdet för **count** räknat åt vänster från den ursprungliga cellen.
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

Om pivottabellen har flera horisontella dimensioner, inbegriper det aktuella radsegmentet endast kolumner som har samma värden som den aktuella kolumnen i samtliga dimensionsrader utom den rad som visar den sista dimensionen i fältens inbördes sorteringsordning.Fältens inbördes sorteringsordning för horisontella dimensioner i pivottabeller definieras helt enkelt som dimensionernas ordning uppifrån och ned.

Exempel:

```
last( sum( Sales ))
last( sum( Sales ), 2 )
last( total sum( Sales )
```

`rangeavg (last(sum(x),1,5))` returnerar ett medelvärde av resultaten av funktionen **sum(x)** utvärderad på de fem kolumnerna längst till höger i det aktuella radsegmentet.

ColumnNo - diagramfunktion

ColumnNo() returnerar numret på den aktuella kolumnen inom det aktuella radsegmentet i en pivottabell. Första kolumnen är nummer 1.

Syntax:

```
ColumnNo([total])
```

Argument:

Argument

Argument	Beskrivning
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

Om pivottabellen har flera horisontella dimensioner, inbegriper det aktuella radsegmentet endast kolumner som har samma värden som den aktuella kolumnen i samtliga dimensionsrader utom den rad som visar den sista dimensionen i fältens inbördes sorteringsordning.Fältens inbördes sorteringsordning för horisontella dimensioner i pivottabeller definieras helt enkelt som dimensionernas ordning uppifrån och ned.

Exempel:

```
if( ColumnNo( )=1, 0, sum( Sales ) / before( sum( Sales )))
```

NoOfColumns - diagramfunktion

NoOfColumns() returnerar antalet kolumner i det aktuella radsegmentet i en pivottabell.

Syntax:

```
NoOfColumns([total])
```

Argument:

Argument

Argument	Beskrivning
TOTAL	Om tabellen är endimensionell eller om kvalificeraren i**TOTAL** används som argument, motsvarar det aktuella kolumnsegmentet alltid hela kolumnen.

Om pivottabellen har flera horisontella dimensioner, inbegriper det aktuella radsegmentet endast kolumner som har samma värden som den aktuella kolumnen i samtliga dimensionsrader utom den rad som visar den sista dimensionen i fältens inbördes sorteringsordning.Fältens inbördes sorteringsordning för horisontella dimensioner i pivottabeller definieras helt enkelt som dimensionernas ordning uppifrån och ned.

Exempel:

```
if( ColumnNo( )=NoOfColumns( ), 0, after( sum( Sales )))
```

5.17 Logiska funktioner

Den här delen beskriver funktioner som hanterar logiska operationer. Alla funktioner kan användas både i dataladdningsskriptet och diagramuttryck.

IsNum

Returnerar -1 (True) om uttrycket kan tolkas som ett tal, annars 0 (False).

```
IsNum( expr )
```

IsText

Returnerar -1 (True) om uttrycket består av text, annars 0 (False).

```
IsText( expr )
```

*Både **IsNum** och **IsText** returnerar 0 om uttrycket är NULL.*

Exempel:

I det följande exemplet laddas en inline-tabell med blandade textvärden och numeriska värden, och två fält läggs till för att kontrollera om värdet är ett numeriskt värde eller ett textvärde.

```
Load *, IsNum(Value), IsText(Value)
Inline [
Value
23
Green
Blue
12
33Red];
```

Den resulterande tabellen ser ut så här:

Resulting table

Value	IsNum(Value)	IsText(Value)
23	-1	0
Green	0	-1
Blue	0	-1
12	-1	0
33Red	0	-1

5.18 Mappningsfunktioner

Den här delen beskriver funktioner som hanterar mappningstabeller. En mappningstabell kan användas för att ersätta fältvärden eller fältnamn under skriptexekveringen.

Mappningsfunktioner kan endast användas i dataladdningsskriptet.

Mappningsfunktioner – en översikt

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

ApplyMap

Skriptfunktionen **ApplyMap** används för att mappa utdata för ett uttryck mot en tidigare inläst mappningstabell.

```
ApplyMap ('mapname', expr [ , defaultexpr ] )
```

MapSubstring

Skriptfunktionen **MapSubstring** används för att mappa delar av ett valfritt uttryck mot en tidigare inläst mappningstabell. Mappningen är skiftlägeskänslig och icke-iterativ och delsträngar mappas från vänster till höger.

```
MapSubstring ('mapname', expr)
```

ApplyMap

Skriptfunktionen **ApplyMap** används för att mappa utdata för ett uttryck mot en tidigare inläst mappningstabell.

Syntax:

```
ApplyMap('map_name', expression [ , default_mapping ] )
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
map_name	Namnet på en mappningstabell som har skapats tidigare med hjälp av satserna **mapping load** eller **mapping select**. Dess namn måste stå inom enkla, raka citationstecken. *Om du använder den här funktionen i en makroexpanderad variabel och hänvisar till en kartläggningstabell som inte finns misslyckas funktionen och fältet skapas inte.*
expression	Uttrycket vars resultat ska mappas.
default_ mapping	Om detta anges ska det här värdet användas som ett standardvärde om mappningstabellen inte innehåller något värde som matchar expression. Om inget anges returneras värdet av expression som det är.

Utdatafältet för ApplyMap ska inte ha samma namn som ett av dess indatafält. Det kan leda till oväntade resultat. Exempel på vad du inte ska använda: `ApplyMap('Map', A) as A.`

Exempel:

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

```
// Load mapping table of country codes:
map1:
mapping LOAD *
```

```
Inline [
CCode, Country
Sw, Sweden
Dk, Denmark
No, Norway
] ;

// Load list of salesmen, mapping country code to country
// If the country code is not in the mapping table, put Rest of the world
Salespersons:
LOAD *,
ApplyMap('map1', CCode,'Rest of the world') As Country
Inline [
CCode, Salesperson
Sw, John
Sw, Mary
Sw, Per
Dk, Preben
Dk, Olle
No, Ole
Sf, Risttu
] ;

// We don't need the CCode anymore
Drop Field 'CCode';
```

Den resulterade tabellen (Salespersons) ser ut så här:

Resulting table

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

MapSubstring

Skriptfunktionen **MapSubstring** används för att mappa delar av ett valfritt uttryck mot en tidigare inläst mappningstabell. Mappningen är skiftlägeskänslig och icke-iterativ och delsträngar mappas från vänster till höger.

Syntax:

```
MapSubstring('map_name', expression)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
map_name	Namnet på en mappningstabell som tidigare lästs via en **mapping load**- eller **mapping select**-sats. Namnet måste stå inom enkla, raka citationstecken. *Om du använder den här funktionen i en makroexpanderad variabel och hänvisar till en kartläggningstabell som inte finns misslyckas funktionen och fältet skapas inte.*
expression	Uttrycket vars resultat ska mappas av delsträngar.

Exempel:

I det här exemplet laddar vi en lista med produktmodeller. Varje modell har en uppsättning attribut som beskrivs av en sammansatt kod. Genom att använda mappningstabellen med MapSubstring kan vi utöka attributkoderna till en beskrivning.

```
map2:
mapping LOAD *
Inline [
AttCode, Attribute
R, Red
Y, Yellow
B, Blue
C, Cotton
P, Polyester
S, Small
M, Medium
L, Large
] ;

Productmodels:
LOAD *,
MapSubString('map2', AttCode) as Description
Inline [
Model, AttCode
Twixie, R C S
Boomer, B P L
Raven, Y P M
Seedling, R C L
SeedlingPlus, R C L with hood
Younger, B C with patch
MultiStripe, R Y B C S/M/L
] ;
```

```
// We don't need the AttCode anymore
Drop Field 'AttCode';
```

Den resulterande tabellen ser ut så här:

Resulting table

Model	Description
Twixie	Red Cotton Small
Boomer	Blue Polyester Large
Raven	Yellow Polyester Medium
Seedling	Red Cotton Large
SeedlingPlus	Red Cotton Large with hood
Younger	Blue Cotton with patch
MultiStripe	Red Yellow Blue Cotton Small/Medium/Large

5.19 Matematiska funktioner

Den här delen beskriver funktioner för matematiska konstanter och booleska värden. De här funktionerna har inga parametrar, men parenteserna är ändå obligatoriska.

Alla funktioner kan användas både i dataladdningsskriptet och diagramuttryck.

e

Funktionen returnerar basen för naturliga logaritmer, **e** (2,71828 ...).

```
e( )
```

false

Funktionen returnerar ett dualt värde med textvärde 'False' och numeriskt värde 0. Det returnerade värdet kan användas som logiskt falskt i uttryck.

```
false( )
```

pi

Funktionen returnerar värdet av π (3,14159 ...).

```
pi( )
```

rand

Funktionen returnerar ett slumpmässigt tal mellan 0 och 1. Det kan användas för att skapa exempeldata.

```
rand( )
```

Exempel:

Det här exempelskriptet skapar en tabell med 1 000 poster med slumpvist valda versaler, det vill säga tecken i intervallet 65 till 91 (65+26).

```
Load
     Chr( Floor(rand() * 26) + 65) as UCaseChar,
     RecNo() as ID
     Autogenerate 1000;
```

true

Funktionen returnerar ett dualt värde med textvärde 'True' och numeriskt värde -1. Det returnerade värdet kan användas som logiskt sant i uttryck.

```
true( )
```

5.20 NULL-funktioner

Den här delen beskriver funktioner för att returnera eller hitta NULL-värden.

Alla funktioner kan användas både i dataladdningsskriptet och diagramuttryck.

NULL-funktioner – en översikt

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

EmptyIsNull

EmptyIsNull-funktionen konverterar tomma strängar till NULL. Den returnerar alltså NULL om parametern är en tom sträng. Annars returnerar den parametern.

```
EmptyIsNull (expr )
```

IsNull

Funktionen **IsNull** testar om värdet på ett uttryck är NULL och om det är så, returnerar -1 (True), annars 0 (False).

```
IsNull (expr )
```

Null

Null-funktionen returnerar ett NULL-värde.

```
NULL( )
```

EmptyIsNull

EmptyIsNull-funktionen konverterar tomma strängar till NULL. Den returnerar alltså NULL om parametern är en tom sträng. Annars returnerar den parametern.

Syntax:

```
EmptyIsNull(exp )
```

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`EmptyIsNull(AdditionalComments)`	Istället för tomma strängar returnerar det här uttrycket NULL för eventuella tomma strängvärden för fältet *AdditionalComments*. Strängar som inte är tomma och tal returneras.
`EmptyIsNull(PurgeChar (PhoneNumber, ' -()'))`	Det här uttrycket tar bort alla streck, blanksteg och parenteser från fältet *PhoneNumber*. Om det inte finns några tecken kvar returnerar funktionen EmptyIsNull den tomma strängen som NULL. Ett tomt telefonnummer är ju samma sak som inget telefonnummer.

IsNull

Funktionen **IsNull** testar om värdet på ett uttryck är NULL och om det är så, returnerar -1 (True), annars 0 (False).

Syntax:

```
IsNull(expr )
```

*En sträng med längden noll inte räknas som NULL och gör att **IsNull** returnerar False.*

Exempel: Dataladdningsskript

I detta exempel laddas en inline-tabell med fyra rader, där de tre första raderna innehåller antingen ingenting, - eller 'NULL' i kolumnen Value. Vi omvandlar dessa värden till sanna NULL-värdesåtergivningar med den mellersta som föregår **LOAD** med hjälp av **Null**-funktionen.

Första föregående **LOAD** lägger till ett fält för att kontrollera om värdet är NULL med hjälp av **IsNull**-funktionen.

```
NullsDetectedAndConverted:

LOAD *,
If(IsNull(ValueNullConv), 'T', 'F') as IsItNull;

LOAD *,
If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value ) as ValueNullConv;

LOAD * Inline
[ID, Value
0,
1,NULL
2,-
3,Value];
```

Detta är den resulterande tabellen. I kolumnen ValueNullConv representeras NULL-värdena av -.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

NULL

Null-funktionen returnerar ett NULL-värde.

Syntax:

```
Null( )
```

Exempel: Dataladdningsskript

I detta exempel laddas en inline-tabell med fyra rader, där de tre första raderna innehåller antingen ingenting, - eller 'NULL' i kolumnen Value. Vi vill omvandla dessa värden till äkta NULL-värderepresentationer.

Den mellersta som föregår **LOAD** utför konverteringen med funktionen **Null**.

Den första som föregår **LOAD** lägger till ett fält som kontrollerar om värdet är NULL, endast som illustration i detta exempel.

```
NullsDetectedAndConverted:

LOAD *,
If(IsNull(ValueNullConv), 'T', 'F') as IsItNull;

LOAD *,
If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value ) as ValueNullConv;

LOAD * Inline
[ID, Value
0,
1,NULL
2,-
3,Value];
```

Detta är den resulterande tabellen. I kolumnen ValueNullConv representeras NULL-värdena av -.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

5.21 Intervallfunktioner

Intervallfunktionerna är funktioner som tar en mängd värden och producerar ett enda värde som resultat. Alla intervallfunktioner kan användas i både dataladdningsskriptet och i diagramuttryck.

I en visualisering kan till exempel en intervallfunktion beräkna ett enda värde utifrån en postöverskridande uppsättning. I dataladdningsskriptet kan en intervallfunktion beräkna ett enda värde utifrån en uppsättning värden i en intern tabell.

Intervallfunktionerna ersätter följande allmänna numeriska funktioner: ***numsum****,* ***numavg****,* ***numcount****,* ***nummin*** *och* ***nummax****, som nu bör ses som obsoleta.*

Grundläggande intervallfunktioner

RangeMax

RangeMax() returnerar de högsta numeriska värden som hittas inom uttrycket eller fältet.

```
RangeMax (first_expr[, Expression])
```

RangeMaxString

RangeMaxString() returnerar det sista värdet i textsorteringsordningen som den finner i uttrycket eller fältet.

```
RangeMaxString (first_expr[, Expression])
```

RangeMin

RangeMin() returnerar de lägsta numeriska värden som hittas inom uttrycket eller fältet.

```
RangeMin (first_expr[, Expression])
```

RangeMinString

RangeMinString() returnerar det första värdet i textsorteringsordningen som den finner i uttrycket eller fältet.

```
RangeMinString (first_expr[, Expression])
```

RangeMode

RangeMode() finner det vanligast förekommande värdet (lägesvärdet) i uttrycket eller fältet.

```
RangeMode (first_expr[, Expression])
```

RangeOnly

RangeOnly() är en dual-funktion som returnerar ett värde om uttrycket utvärderas till ett unikt värde. Om så inte är fallet returneras **NULL**.

```
RangeOnly (first_expr[, Expression])
```

RangeSum

RangeSum() returnerar summan av ett värdeintervall. Alla icke-numeriska värden behandlas som 0.

```
RangeSum (first_expr[, Expression])
```

Räkneintervallfunktioner

RangeCount

RangeCount() returnerar antalet värden, både textvärden och talvärden, i uttrycket eller fältet.

```
RangeCount (first_expr[, Expression])
```

RangeMissingCount

RangeMissingCount() returnerar antalet icke-metriska värden (inklusive NULL) i uttrycket eller fältet.

```
RangeMissingCount (first_expr[, Expression])
```

RangeNullCount

RangeNullCount() finner antalet NULL-värden i uttrycket eller fältet.

```
RangeNullCount (first_expr[, Expression])
```

RangeNumericCount

RangeNumericCount() finner antalet numeriska värden i ett uttryck eller fält.

```
RangeNumericCount (first_expr[, Expression])
```

RangeTextCount

RangeTextCount() returnerar antalet textvärden i ett uttryck eller fält.

```
RangeTextCount (first_expr[, Expression])
```

Statistiska intervallfunktioner

RangeAvg

RangeAvg() returnerar medelvärdet av ett intervall. Indata till funktionen kan vara antingen ett intervall med värden eller ett uttryck.

```
RangeAvg (first_expr[, Expression])
```

RangeCorrel

RangeCorrel() returnerar korrelationskoefficienten för två datauppsättningar. Korrelationskoefficienten är ett mått på förhållandet mellan datauppsättningarna.

```
RangeCorrel (x_values , y_values[, Expression])
```

RangeFractile

RangeFractile() returnerar det värde som motsvarar den n:te **fraktilen** (kvantilen) av ett talintervall.

```
RangeFractile (fractile, first_expr[ ,Expression])
```

RangeKurtosis

RangeKurtosis() returnerar det värde som motsvarar kurtosis hos ett talintervall.

```
RangeKurtosis (first_expr[, Expression])
```

RangeSkew

RangeSkew() returnerar det värde som motsvarar skevheten hos ett talintervall.

```
RangeSkew (first_expr[, Expression])
```

RangeStdev

RangeStdev() finner standardavvikelsen hos ett talintervall.

```
RangeStdev (expr1[, Expression])
```

Finansiella intervallfunktioner

RangeIRR

RangeIRR() returnerar internräntan för en serie kassaflöden, representerad av indatavärdena.

```
RangeIRR (value[, value][, Expression])
```

RangeNPV

RangeNPV() returnerar det aktuella nett värdet på en investering baserad på en rabattsats och en serie framtida periodiska betalningar (negativa värden) och inkomster (positiva värden). Resultatet anges i det fördefinierade talformatet **money**.

```
RangeNPV (discount_rate, value[, value][, Expression])
```

RangeXIRR

RangeXIRR() returnerar internräntan för ett schema av kassaflöden som inte nödvändigtvis är periodiska. För att beräkna intern avkastningsgrad för en serie periodiska kassaflöden används **RangeIRR**-funktionen.

```
RangeXIRR (values, dates[, Expression])
```

RangeXNPV

RangeXNPV() returnerar det aktuella nettovärdet för ett schema av kassaflöden som inte nödvändigtvis är periodiskt. Resultatet anges i ett fördefinierat talformat (valuta). För att beräkna det aktuella nettovärdet för en serie periodiska kassaflöden används funktionen rangenpv (se **RangeNPV**).

```
RangeXNPV (discount_rate, values, dates[, Expression])
```

Se även:

Postöverskridande funktioner (page 639)

RangeAvg

RangeAvg() returnerar medelvärdet av ett intervall. Indata till funktionen kan vara antingen ett intervall med värden eller ett uttryck.

Syntax:

```
RangeAvg(first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Om inget numeriskt värde påträffas returneras NULL.

Exempel och resultat:

Skriptexempel

Exempel	Resultat
`RangeAvg (1,2,4)`	Returnerar 2,33333333
`RangeAvg (1,'xyz')`	Returnerar 1
`RangeAvg (null( ), 'abc')`	Returnerar NULL

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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
];
```

Den resulterande tabellen visar de returnerade värdena för MyRangeAvg för varje post i tabellen.

Resultattabell

RangeID	MyRangeAvg
1	7

RangeID	MyRangeAvg
2	4
3	6
4	12.666
5	6.333
6	5

Exempel med uttryck:

```
RangeAvg (Above(MyField),0,3))
```

Returnerar ett glidande genomsnitt för resultatet av intervallet av tre värden för **MyField** beräknat på den aktuella raden och de två raderna ovanför den aktuella raden. Om du anger det tredje argumentet som 3 returnerar **Above()**-funktionen tre värden, där det finns tillräckligt med rader ovanför, vilka används som indata för **RangeAvg()**-funktionen.

Data som används i exempel:

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeAvg (Above (MyField,0,3))	Comments
10	10	Eftersom det här är den översta raden består intervallet endast av ett värde.
2	6	Det finns endast en rad ovanför den här raden, så intervallet är: 10,2.
8	6.6666666667	Motsvarande RangeAvg(10,2,8)
18	9.333333333	-
5	10. 333333333	-
9	10.6666666667	-

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Se även:

- *Avg - diagramfunktion (page 267)*
- *Count - diagramfunktion (page 233)*

RangeCorrel

RangeCorrel() returnerar korrelationskoefficienten för två datauppsättningar. Korrelationskoefficienten är ett mått på förhållandet mellan datauppsättningarna.

Syntax:

```
RangeCorrel(x_value , y_value[, Expression])
```

Returnerad datatyp: numeriska

Dataserien ska anges som (x,y)-par. Ett exempel: för att utvärdera två serier med data array 1 och array 2, där array 1 = 2,6,9 och array 2 = 3,8,4 skulle du skriva `RangeCorrel (2,3,6,8,9,4)`, vilket returnerar 0,269.

Argument:

Argument

Argument	Beskrivning
x-value, y-value	Varje värde motsvarar ett enstaka värde eller ett intervall av värden som de returneras av en postöverskridande funktion med en tredje valbar parameter. Varje värde eller värdeintervall måste svara mot ett **x-value** eller intervall av **y-values**.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

För att funktionen ska kunna beräknas krävs minst två uppsättningar koordinater.

Textvärden, NULL-värden och saknade värden returnerar NULL.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeCorrel (2,3,6,8,9,4,8,5)`	Returnerar 0,2492. Den här funktionen kan läsas in i skriptet eller läggas till i en visualisering i uttrycksredigeraren.

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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 '|');
```

I en tabell med ID1 som dimension och måttet: RangeCorrel(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6)) hittar funktionen **RangeCorrel()** värdet för **Correl** över intervallet för sex x,y-par för vart och ett av ID1-värdena.

Resultattabell

ID1	MyRangeCorrel
01	-0.9517
02	-0.5209
03	-0.5209
04	-0.1599

Exempel:

```
XY:
LOAD recno() as RangeID, * Inline [
X|Y
2|3
6|8
9|4
8|5
](delimiter is '|');
```

I en tabell med RangeID som dimension och måttet: RangeCorrel(Below(X,0,4,BelowY,0,4)) använder funktionen **RangeCorrel()** resultaten av **Below()**-funktionerna, vilka eftersom det tredje argumentet (count) är inställt på 4, producerar ett intervall av x-y-värden från den laddade tabellen XY.

Resultattabell

RangeID	MyRangeCorrel2
01	0.2492
02	-0.9959

RangeID	MyRangeCorrel2
03	-1.0000
04	-

Värdet för RangeID 01 är detsamma som att manuellt ange RangeCorrel(2,3,6,8,9,4,8,5). För de övriga värdena för RangeID är den serie som produceras av Below()-funktionen: (6,8,9,4,8,5), (9,4,8,5) och (8,5), där den sista producerar ett null-resultat.

Se även:

Correl - diagramfunktion (page 269)

RangeCount

RangeCount() returnerar antalet värden, både textvärden och talvärden, i uttrycket eller fältet.

Syntax:

```
RangeCount(first_expr[, Expression])
```

Returnerad datatyp: heltal

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska räknas.
Expression	Valfria uttryck eller fält som innehåller det intervall med data som ska räknas.

Begränsningar:

NULL-värden räknas ej.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeCount (1,2,4)`	Returnerar 3
`RangeCount (2,'xyz')`	Returnerar 2
`RangeCount (null( ))`	Returnerar 0
`RangeCount (2,'xyz', null())`	Returnerar 2

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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
];
```

Den resulterande tabellen visar de returnerade värdena för MyRangeCount för varje post i tabellen.

Resultattabell

RangeID	MyRangeCount
1	3
2	3
3	3
4	3
5	3
6	3

Exempel med uttryck:

```
RangeCount (Above(MyField,1,3))
```

Returnerar antalet värden som finns i de tre resultaten för **MyField**. Om du anger det första argumentet för **Above()**-funktionen som 1 och det andra argumentet som 3, returnerar den värdena från de första tre fälten ovanför den aktuella raden, om det finns tillräckligt med rader. De används som indata för **RangeCount()**-funktionen.

Data som används i exempel:

Exempeldata

MyField	RangeCount(Above(MyField,1,3))
10	0
2	1
8	2
18	3
5	3
9	3

Data som används i exempel:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Se även:

- *Count - diagramfunktion (page 233)*

RangeFractile

RangeFractile() returnerar det värde som motsvarar den n:te **fraktilen** (kvantilen) av ett talintervall.

> *RangeFractile() använder linjär interpolering mellan de närmaste rangordningsnumren när fraktilen beräknas.*

Syntax:

```
RangeFractile(fractile, first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
fractile	Ett tal mellan 0 och 1 som motsvarar den fraktil (kvantil uttryckt som bråkdel) som ska beräknas.
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeFractile (0.24,1,2,4,6)`	Returnerar 1,72
`RangeFractile(0.5,1,2,3,4,6)`	Returnerar 3
`RangeFractile (0.5,1,2,5,6)`	Returnerar 3,5

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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
];
```

Den resulterande tabellen visar de returnerade värdena för MyRangeFrac för varje post i tabellen.

Resultattabell

RangeID	MyRangeFrac
1	6
2	3
3	8
4	11
5	5
6	4

Exempel med uttryck:

```
RangeFractile (0.5, Above(Sum(MyField),0,3))
```

I det här exemplet innehåller den postöverskridande funktionen **Above()** valbara parametrar för offset ochcount. Detta ger ett intervall av resultat som kan användas som indata för vilken som helst av intervallfunktionerna. I det här fallet returnerar `Above(Sum(MyField),0,3)` värdet för `MyField` för aktuell rad och för de två raderna ovanför. De här värdena tillhandahåller indata för funktionen **RangeFractile()**. Så för

bottenraden i tabellen nedan är detta likvärdigt med `RangeFractile(0.5, 3,4,6)`, det vill säga att beräkna 0,5-fraktilen för serien 3, 4 och 6. För de första två raderna i tabellen nedan minskas antalet värden i intervallet utifrån detta när det inte finns några rader ovanför aktuell rad. Liknande resultat fås för andra postöverskridande funktioner.

Exempeldata

MyField	RangeFractile(0.5, Above(Sum(MyField),0,3))
1	1
2	1.5
3	2
4	3
5	4
6	5

Data som används i exempel:

```
RangeTab:
LOAD * INLINE [
MyField
1
2
3
4
5
6
] ;
```

Se även:

- *Above - diagramfunktion (page 642)*
- *Fractile - diagramfunktion (page 273)*

RangeIRR

RangeIRR() returnerar internräntan för en serie kassaflöden, representerad av indatavärdena.

Internränta är den räntesats som ges vid en investering i form av betalning (negativa värden) och inkomst (positiva värden) som infaller regelbundet.

Syntax:

```
RangeIRR(value[, value][, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Ett enstaka värde eller ett intervall av värden som de returneras av en postöverskridande funktion med en tredje valfri parameter. För att funktionen ska kunna beräknas krävs minst ett negativt och ett positivt värde.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Textvärden, NULL-värden samt saknade värden ignoreras.

Exempeltabell

Exempel	Resultat
`RangeIRR(-70000,12000,15000,18000,21000,26000)`	Returnerar 0,0866

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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 '|');`
	Den resulterande tabellen visar de returnerade värdena för RangeIRR för varje post i tabellen.
	RangeID: 1 — RangeIRR: 0.0639
	RangeID: 2 — RangeIRR: 0.8708
	RangeID: 3 — RangeIRR: -
	RangeID: 4 — RangeIRR: 5.8419
	RangeID: 5 — RangeIRR: 0.9318
	RangeID: 6 — RangeIRR: -0.2566

Se även:

Postöverskridande funktioner (page 639)

RangeKurtosis

RangeKurtosis() returnerar det värde som motsvarar kurtosis hos ett talintervall.

Syntax:

```
RangeKurtosis(first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Om inget numeriskt värde påträffas returneras NULL.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeKurtosis (1,2,4,7)`	Returnerar -0,28571428571429

Se även:

Kurtosis - diagramfunktion (page 280)

RangeMax

RangeMax() returnerar de högsta numeriska värden som hittas inom uttrycket eller fältet.

Syntax:

```
RangeMax(first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Om inget numeriskt värde påträffas returneras NULL.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeMax (1,2,4)`	Returnerar 4
`RangeMax (1,'xyz')`	Returnerar 1
`RangeMax (null( ), 'abc')`	Returnerar NULL

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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
];
```

Den resulterande tabellen visar de returnerade värdena för MyRangeMax för varje post i tabellen.

Resultattabell

RangeID	MyRangeMax
1	10
2	7
3	8
4	18
5	9
6	9

Exempel med uttryck:

```
RangeMax (Above(MyField,0,3))
```

Returnerar maxvärdet inom intervallet av tre värden för **MyField**, uträknade på den aktuella raden och de två raderna ovanför den aktuella raden. Om du anger det tredje argumentet som 3 returnerar **Above()**-funktionen tre värden, där det finns tillräckligt med rader ovanför, vilka används som indata för **RangeMax ()**-funktionen.

Data som används i exempel:

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeMax (Above(Sum(MyField),1,3))
10	10
2	10
8	10
18	18
5	18
9	18

Data som används i exempel:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

RangeMaxString

RangeMaxString() returnerar det sista värdet i textsorteringsordningen som den finner i uttrycket eller fältet.

Syntax:

```
RangeMaxString(first_expr[, Expression])
```

Returnerad datatyp: sträng

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeMaxString (1,2,4)`	Returnerar 4
`RangeMaxString ('xyz','abc')`	Returnerar 'xyz'
`RangeMaxString (5,'abc')`	Returnerar 'abc'
`RangeMaxString (null( ))`	Returnerar NULL

Exempel med uttryck:

```
RangeMaxString (Above(MaxString(MyField),0,3))
```

Returnerar det sista (i textsorteringsordning) av de tre resultaten av **MaxString(MyField)**-funktionen som utvärderats på den aktuella raden och de två raderna ovanför den aktuella raden.

Data som används i exempel:

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeMaxString(Above(MaxString(MyField),0,3))
10	10
abc	abc
8	abc
def	def
xyz	xyz
9	xyz

Data som används i exempel:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
```

```
'xyz'
9
] ;
```

Se även:

- *MaxString - diagramfunktion (page 396)*

RangeMin

RangeMin() returnerar de lägsta numeriska värden som hittas inom uttrycket eller fältet.

Syntax:

```
RangeMin(first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Om inget numeriskt värde påträffas returneras NULL.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeMin (1,2,4)`	Returnerar 1
`RangeMin (1,'xyz')`	Returnerar 1
`RangeMin (null( ), 'abc')`	Returnerar NULL

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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
];
```

Den resulterande tabellen visar de returnerade värdena för MyRangeMin för varje post i tabellen.

Resultattabell

RangeID	MyRangeMin
1	5
2	2
3	2
4	9
5	5
6	2

Exempel med uttryck:

```
RangeMin (Above(MyField,0,3)
```

Returnerar minimivärdet inom intervallet av tre värden för **MyField**, uträknade på den aktuella raden och de två raderna ovanför den aktuella raden. Om du anger det tredje argumentet som 3 returnerar **Above()**-funktionen tre värden, där det finns tillräckligt med rader ovanför, vilka används som indata för **RangeMin ()**-funktionen.

Data som används i exempel:

Exempeldata

MyField	RangeMin(Above(MyField,0,3))
10	10
2	2
8	2
18	2
5	5
9	5

Data som används i exempel:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
```

```
] ;
```

Se även:

- *Min - diagramfunktion (page 220)*

RangeMinString

RangeMinString() returnerar det första värdet i textsorteringsordningen som den finner i uttrycket eller fältet.

Syntax:

```
RangeMinString(first_expr[, Expression])
```

Returnerad datatyp: sträng

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeMinString (1,2,4)`	Returnerar 1
`RangeMinString ('xyz','abc')`	Returnerar 'abc'
`RangeMinString (5,'abc')`	Returnerar 5
`RangeMinString (null( ))`	Returnerar NULL

Exempel med uttryck:

```
RangeMinString (Above(MinString(MyField),0,3))
```

Returnerar det första (i textsorteringsordning) av de tre resultaten av **MinString(MyField)**-funktionen som utvärderats på den aktuella raden och de två raderna ovanför den aktuella raden.

Data som används i exempel:

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeMinString(Above(MinString(MyField),0,3))
10	10
abc	10
8	8
def	8
xyz	8
9	9

Data som används i exempel:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

Se även:

- *MinString - diagramfunktion (page 399)*

RangeMissingCount

RangeMissingCount() returnerar antalet icke-metriska värden (inklusive NULL) i uttrycket eller fältet.

Syntax:

```
RangeMissingCount(first_expr[, Expression])
```

Returnerad datatyp: heltal

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska räknas.
Expression	Valfria uttryck eller fält som innehåller det intervall med data som ska räknas.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeMissingCount (1,2,4)`	Returnerar 0
`RangeMissingCount (5,'abc')`	Returnerar 1
`RangeMissingCount (null( ))`	Returnerar 1

Exempel med uttryck:

```
RangeMissingCount (Above(MinString(MyField),0,3))
```

Returnerar antalet icke-numeriska värden i de tre resultaten av **MinString(MyField)**-funktionen som utvärderats på den aktuella raden och de två raderna ovanför den aktuella raden.

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeMissingCount (Above(MinString (MyField),0,3))	Explanation
10	2	Returnerar 2 eftersom det inte finns några rader ovanför den här raden, så två av de tre värdena saknas.
abc	2	Returnerar 2 eftersom det bara finns 1 rad ovanför den aktuella raden och den aktuella raden är icke-numerisk ("abc").
8	1	Returnerar 1 eftersom 1 av de 3 raderna inkluderar ett icke-numeriskt värde ("abc").
def	2	Returnerar 2 eftersom 2 av de 3 raderna inkluderar icke-numeriska värden ("def" och "abc").
xyz	2	Returnerar 2 eftersom 2 av de 3 raderna inkluderar icke-numeriska värden (" xyz" och "def").
9	2	Returnerar 2 eftersom 2 av de 3 raderna inkluderar icke-numeriska värden (" xyz" och "def").

Data som används i exempel:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
```

```
8
'def'
'xyz'
9
] ;
```

Se även:

- *MissingCount - diagramfunktion (page 236)*

RangeMode

RangeMode() finner det vanligast förekommande värdet (lägesvärdet) i uttrycket eller fältet.

Syntax:

```
RangeMode(first_expr {, Expression})
```

Returnerad datatyp: numeriska

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Om mer än ett värde är lika vanligt förekommande, returneras NULL.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeMode (1,2,9,2,4)`	Returnerar 2
`RangeMode ('a',4,'a',4)`	Returnerar NULL
`RangeMode (null( ))`	Returnerar NULL

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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
];
```

Den resulterande tabellen visar de returnerade värdena för **MyRangeMode** för varje post i tabellen.

Resultattabell

RangeID	MyRangMode
1	-
2	-
3	8
4	-
5	5
6	-

Exempel med uttryck:

```
RangeMode (Above(MyField,0,3))
```

Returnerar det vanligast förekommande värdet i de tre resultaten av **MyField** som utvärderats på den aktuella raden och de två raderna ovanför den aktuella raden. Om du anger det tredje argumentet som 3 returnerar **Above()**-funktionen tre värden, där det finns tillräckligt med rader ovanför, vilka används som indata för **RangeMode()**-funktionen.

Data som används i exemplet:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeMode(Above(MyField,0,3))
10	Returnerar 10 eftersom det inte finns några rader ovanför, så att det enskilda värdet är det vanligast förekommande.
2	-
8	-
18	-
5	-
9	-

Se även:

- *Mode - diagramfunktion (page 223)*

RangeNPV

RangeNPV() returnerar det aktuella nett värdet på en investering baserad på en rabattsats och en serie framtida periodiska betalningar (negativa värden) och inkomster (positiva värden). Resultatet anges i det fördefinierade talformatet **money**.

För kassaflöden som inte nödvändigtvis är periodiska, se *RangeXNPV (page 729)*.

Syntax:

```
RangeNPV(discount_rate, value[,value][, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
discount_rate	Räntesatsen per period.
value	En betalning eller inkomst som inträffar i slutet av varje period. Varje värde kan vara ett enstaka värde eller ett intervall av värden som de returneras av en postöverskridande funktion med en tredje valfri parameter.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Textvärden, NULL-värden samt saknade värden ignoreras.

Exempel	Resultat
`RangeNPV(0.1,-10000,3000,4200,6800)`	Returnerar 1188,44

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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 '|');`
	Den resulterande tabellen visar de returnerade värdena för RangeNPV för varje post i tabellen.
	RangeID: 1 — RangeNPV: $-49.13
	2 — $777.78
	3 — $98.77
	4 — $25.51
	5 — $250.83
	6 — $20.40

Se även:

- *Postöverskridande funktioner (page 639)*

RangeNullCount

RangeNullCount() finner antalet NULL-värden i uttrycket eller fältet.

Syntax:

```
RangeNullCount(first_expr [, Expression])
```

Returnerad datatyp: heltal

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeNullCount (1,2,4)`	Returnerar 0
`RangeNullCount (5,'abc')`	Returnerar 0
`RangeNullCount (null( ), null( ))`	Returnerar 2

Exempel med uttryck:

```
RangeNullCount (Above(Sum(MyField),0,3))
```

Returnerar antalet NULL-värden i de tre resultaten av **Sum(MyField)**-funktionen som utvärderats på den aktuella raden och de två raderna ovanför den aktuella raden.

*Att kopiera **MyField** i exemplet nedan resulterar inte i NULL-värde.*

Exempeldata

MyField	RangeNullCount(Above(Sum(MyField),0,3))
10	Returnerar 2 eftersom det inte finns några rader ovanför denna rad så 2 av de 3 värdena saknas (=NULL).
”abc”	Returnerar 1 eftersom det bara finns 1 rad ovanför den aktuella raden så att 1 av 3 värden saknas (=NULL).
8	Returnerar 0 eftersom ingen av de tre värdena är ett NULL-värde.

Data som används i exempel:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
] ;
```

Se även:

NullCount - diagramfunktion (page 239)

RangeNumericCount

RangeNumericCount() finner antalet numeriska värden i ett uttryck eller fält.

Syntax:

```
RangeNumericCount(first_expr[, Expression])
```

Returnerad datatyp: heltal

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeNumericCount (1,2,4)`	Returnerar 3
`RangeNumericCount (5,'abc')`	Returnerar 1
`RangeNumericCount (null( ))`	Returnerar 0

Exempel med uttryck:

```
RangeNumericCount (Above(MaxString(MyField),0,3))
```

Returnerar antal numeriska värden i de tre resultaten av **MaxString(MyField)**-funktionen som utvärderats på den aktuella raden och de två raderna ovanför den aktuella raden.

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeNumericCount(Above(MaxString(MyField),0,3))
10	1
abc	1
8	2
def	1
xyz	1
9	1

Data som används i exempel:

```
RangeTab:
```

```
LOAD * INLINE [
MyField
10
'abc'
8
def
xyz
9
] ;
```

Se även:

- *NumericCount - diagramfunktion (page 242)*

RangeOnly

RangeOnly() är en dual-funktion som returnerar ett värde om uttrycket utvärderas till ett unikt värde. Om så inte är fallet returneras **NULL**.

Syntax:

```
RangeOnly(first_expr[, Expression])
```

Returnerad datatyp: dual

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Exempel och resultat:

Exempel	Resultat
`RangeOnly (1,2,4)`	Returnerar NULL
`RangeOnly (5,'abc')`	Returnerar NULL
`RangeOnly (null( ), 'abc')`	Returnerar 'abc'
`RangeOnly(10,10,10)`	Returnerar 10

Se även:

- *Only - diagramfunktion (page 226)*

RangeSkew

RangeSkew() returnerar det värde som motsvarar skevheten hos ett talintervall.

Syntax:

```
RangeSkew(first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Om inget numeriskt värde påträffas returneras NULL.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`rangeskew (1,2,4)`	Returnerar 0,93521952958283
`rangeskew (above (SalesValue,0,3))`	Returnerar en glidande skevhet av intervallet av tre värden som returnerats från funktionen above() beräknad på den aktuella raden och de två raderna ovanför den aktuella raden.

Data som används i exemplet:

Exempeldata

CustID	RangeSkew(Above(SalesValue,0,3))
1-20	-, -, 0.5676, 0.8455, 1.0127, -0.8741, 1.7243, -1.7186, 1.5518, 1.4332, 0, 1.1066, 1.3458, 1.5636, 1.5439, 0.6952, -0.3766

```
SalesTable:
LOAD recno() as CustID, * inline [
SalesValue
101
163
126
```

```
139
167
86
83
22
32
70
108
124
176
113
95
32
42
92
61
21
] ;
```

Se även:

- *Skew - diagramfunktion (page 312)*

RangeStdev

RangeStdev() finner standardavvikelsen hos ett talintervall.

Syntax:

```
RangeStdev(first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

Om inget numeriskt värde påträffas returneras NULL.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeStdev (1,2,4)`	Returnerar 1,5275252316519
`RangeStdev (null( ))`	Returnerar NULL
`RangeStdev (above (SalesValue),0,3))`	Returnerar en glidande standard av intervallet av tre värden som returnerats från funktionen above() beräknad på den aktuella raden och de två raderna ovanför den aktuella raden.

Data som används i exemplet:

Exempeldata

CustID	RangeStdev(SalesValue, 0,3))
1-20	-,43.841, 34.192, 18.771, 20.953, 41.138, 47.655, 36.116, 32.716, 25.325, 38,000, 27.737, 35.553, 33.650, 42.532, 33.858, 32.146, 25.239, 35.595

```
SalesTable:
LOAD recno() as CustID, * inline [
SalesValue
101
163
126
139
167
86
83
22
32
70
108
124
176
113
95
32
42
92
61
21
] ;
```

Se även:

- *Stdev - diagramfunktion (page 315)*

RangeSum

RangeSum() returnerar summan av ett värdeintervall. Alla icke-numeriska värden behandlas som 0.

Syntax:

```
RangeSum(first_expr[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Begränsningar:

RangeSum-funktionen behandlar alla icke-numeriska värden som 0.

Exempel och resultat:

Exempel

Exempel	Resultat
`RangeSum (1,2,4)`	Returnerar 7
`RangeSum (5,'abc')`	Returnerar 5
`RangeSum (null( ))`	Returnerar 0

Exempel:

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din 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
];
```

Den resulterande tabellen visar de returnerade värdena för MyRangeSum för varje post i tabellen.

Resultattabell

RangeID	MyRangeSum
1	21
2	12
3	18
4	38
5	19
6	15

Exempel med uttryck:

```
RangeSum (Above(MyField,0,3))
```

Returnerar summan av de tre värdena för **MyField)**: från den aktuella raden och de två raderna ovanför den aktuella raden. Om du anger det tredje argumentet som 3 returnerar **Above()**-funktionen tre värden, där det finns tillräckligt med rader ovanför, vilka används som indata för **RangeSum()**-funktionen.

Data som används i exempel:

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	RangeSum(Above(MyField,0,3))
10	10
2	12
8	20
18	28
5	31
9	32

Data som används i exempel:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
```

```
8
18
5
9
] ;
```

Se även:

- *Sum - diagramfunktion (page 228)*
- *Above - diagramfunktion (page 642)*

RangeTextCount

RangeTextCount() returnerar antalet textvärden i ett uttryck eller fält.

Syntax:

```
RangeTextCount(first_expr[, Expression])
```

Returnerad datatyp: heltal

Argument:

Argumenten hos denna funktion kan innehålla postöverskridande funktioner som i sig returnerar en lista över värden.

Argument

Argument	Beskrivning
first_expr	Det uttryck eller fält som innehåller de data som ska mätas.
Expression	Valfria uttryck eller fält som innehåller de intervalldata som ska mätas.

Exempel och resultat:

Exempel på funktioner

Exempel	Resultat
`RangeTextCount (1,2,4)`	Returnerar 0
`RangeTextCount (5,'abc')`	Returnerar 1
`RangeTextCount (null( ))`	Returnerar 0

Exempel med uttryck:

```
RangeTextCount (Above(MaxString(MyField),0,3))
```

Returnerar antal textvärden i de tre resultaten av **MaxString(MyField)**-funktionen som utvärderats på den aktuella raden och de två raderna ovanför den aktuella raden.

Data som används i exempel:

*Inaktivera sortering för **MyField** för att vara säker på att exemplet fungerar.*

Exempeldata

MyField	MaxString(MyField)	RangeTextCount(Above(Sum(MyField),0,3))
10	10	0
abc	abc	1
8	8	1
def	def	2
xyz	xyz	2
9	9	2

Data som används i exempel:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
null()
'xyz'
9
] ;
```

Se även:

TextCount - diagramfunktion (page 245)

RangeXIRR

RangeXIRR() returnerar internräntan för ett schema av kassaflöden som inte nödvändigtvis är periodiska. För att beräkna intern avkastningsgrad för en serie periodiska kassaflöden används **RangeIRR**-funktionen.

Syntax:

```
RangeXIRR(value, date{, value, date})
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
value	Ett kassaflöde eller en serie kassaflöden som motsvarar ett schema över betalningsdatum. Värdeserien måste innehålla minst ett negativt och ett positivt värde.
date	Ett betalningsdatum eller ett schema över betalningsdatum som motsvarar kassaflödesbetalningar.

Begränsningar:

Textvärden, NULL-värden samt saknade värden ignoreras.

Alla betalningar diskonteras utifrån ett 365-dagarsår.

Exempel	Resultat
`RangeXIRR(-2500,'2008-01-01',2750,'2008-09-01')`	Returnerar 0,1532

Se även:

RangeIRR (page 704)

RangeXNPV

RangeXNPV() returnerar det aktuella nettovärdet för ett schema av kassaflöden som inte nödvändigtvis är periodiskt. Resultatet anges i ett fördefinierat talformat (valuta). För att beräkna det aktuella nettovärdet för en serie periodiska kassaflöden används funktionen rangenpv (se **RangeNPV**).

Syntax:

```
RangeXNPV(discount_rate, values, dates[, Expression])
```

Returnerad datatyp: numeriska

Argument:

Argument

Argument	Beskrivning
discount_ rate	Räntesatsen per period.

Argument	Beskrivning
values	Ett kassaflöde ellen en serie kassaflöden som motsvarar ett schema över betalningsdatum. Varje värde kan vara ett enstaka värde eller ett intervall av värden som de returneras av en postöverskridande funktion med en tredje valfri parameter. Värdeserien måste innehålla minst ett negativt och ett positivt värde.
dates	Ett betalningsdatum eller ett schema över betalningsdatum som motsvarar kassaflödesbetalningar.

Begränsningar:

Textvärden, NULL-värden samt saknade värden ignoreras.

Alla betalningar diskonteras utifrån ett 365-dagarsår.

Exempeltabell

Exempel	Resultat
`RangeXNPV(0.1, -2500,'2008-01-01',2750,'2008-09-01')`	Returnerar 80,25

Lägg till exempelskriptet i appen och kör det. När du vill se resultatet lägger du till de fält som anges i resultatkolumnen i ett ark i din app.
`RangeTab3:`
`LOAD *,`
`recno() as RangeID,`
`RangeXNPV(Field1,Field2,Field3) as RangeNPV;`
`LOAD * INLINE [`
`Field1|Field2|Field3`
`10|5|-6000`
`2|NULL|7000`
`8|'abc'|8000`
`18|11|9000`
`5|5|9000`
`9|4|2000`
`] (delimiter is '|');`
	Den resulterande tabellen visar de returnerade värdena för RangeXNPV för varje post i tabellen.
	RangeID — RangeXNPV
	1 — $-49.13
	2 — $777.78
	3 — $98.77
	4 — $25.51
	5 — $250.83
	6 — $20.40

5.22 Rangordnings- och klustringsfunktioner

Dessa funktioner kan endast användas i diagramuttryck.

Rangordningsfunktioner i diagram

Alternativet Visa inte nollvärden inaktiveras automatiskt när dessa funktioner används. NULL-värden ignoreras.

Rank

Rank() utvärderar raderna i diagrammet i uttrycket, och visar för varje rad den relativa placeringen av dimensionsvärdet utvärderat i uttrycket. Funktionen utvärderar uttrycket, jämför resultatet med resultaten på de andra raderna som innehåller det aktuella kolumnsegmentet och returnerar rangordningsnumret för den aktuella raden inom segmentet.

```
Rank - diagramfunktion([TOTAL [<fld {, fld}>]] expr[, mode[, fmt]])
```

HRank

HRank() utvärderar uttrycket och jämför resultatet med resultaten i de andra kolumnerna som innehåller det aktuella radsegmentet i en pivottabell. Funktionen returnerar sedan rangordningsnumret för den aktuella kolumnen inom segmentet.

```
HRank - diagramfunktion([TOTAL] expr[, mode[, fmt]])
```

Klustringsfunktioner i diagram

KMeans2D

Egenskapsgruppen **Platslicens** innehåller egenskaper som är kopplade till licensen för systemet Qlik Sense. Alla fält är obligatoriska och får inte vara tomma.

Egenskaper för platslicens

Egenskapsnamn	Beskrivning
Innehavarens namn	Användarnamnet för Qlik Sense-produktägaren.
Innehavarens företag	Namnet på organisationen som Qlik Sense-produktägaren ingår i.
Serienummer	Serienumret som är tilldelat till Qlik Sense-programvaran.
Kontrollnummer	Kontrollnumret som är tilldelat till Qlik Sense-programvaran.
Åtkomst till **LEF**	License Enabler File (LEF) som är tilldelat till Qlik Sense-programvaran.

KMeans2D() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring, och för varje diagramrad visas kluster-ID för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1 respektive coordinate_2. Dessa är båda aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters. Data normaliseras med normparametern (valfritt).

```
KMeans2D - diagramfunktion(num_clusters, coordinate_1, coordinate_2 [, norm])
```

KMeansND

KMeansND() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring och för varje diagramrad visas kluster-ID för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1 och coordinate_2 osv. upp till n kolumner. Dessa är alla aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters.

```
KMeansND - diagramfunktion(num_clusters, num_iter, coordinate_1, coordinate_2
[,coordinate_3 [, ...]])
```

KMeansCentroid2D

KMeansCentroid2D() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring, och för varje diagramrad visas önskad koordinat för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1 respektive coordinate_2. Dessa är båda aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters. Data normaliseras med normparametern (valfritt).

```
KMeansCentroid2D - diagramfunktion(num_clusters, coordinate_no, coordinate_1, coordinate_2 [, norm])
```

KMeansCentroidND

KMeansCentroidND() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring och för varje diagramrad visas önskad koordinat för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1, coordinate_2 osv. upp till n kolumner. Dessa är alla aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters.

```
KMeansCentroidND - diagramfunktion(num_clusters, num_iter, coordinate_no, coordinate_1, coordinate_2 [,coordinate_3 [, ...]])
```

Rank - diagramfunktion

Rank() utvärderar raderna i diagrammet i uttrycket, och visar för varje rad den relativa placeringen av dimensionsvärdet utvärderat i uttrycket. Funktionen utvärderar uttrycket, jämför resultatet med resultaten på de andra raderna som innehåller det aktuella kolumnsegmentet och returnerar rangordningsnumret för den aktuella raden inom segmentet.

Kolumnsegment

För andra diagram än tabeller definieras det aktuella kolumnsegmentet som det visas i diagrammets raka tabellmotsvarighet.

Syntax:

```
Rank([TOTAL] expr[, mode[, fmt]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
mode	Anger resultatets talrepresentation.

Argument	Beskrivning
fmt	Anger resultatets textrepresentation.
TOTAL	Om diagrammet är endimensionellt eller om uttrycket föregås av kvalificeraren **TOTAL** utvärderas funktionen längs med hela kolumnen. Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

Rangordningen returneras som ett dualt värde, som i det fall där varje rad har en unik rangordning är ett heltal mellan 1 och antalet rader i det aktuella kolumnsegmentet.

Om flera rader har samma rangordningsnummer, kan text- och talrepresentationerna för gruppen kontrolleras med parametrarna **mode** och **fmt**.

mode

Det andra argumentet, **mode**, kan ta följande värden:

Exempel med **mode**

Värde	Beskrivning
0 (standard)	Om alla rangordningsnummer inom gruppen är lägre än det mellersta värdet i den totala rangordningen, får alla rader i gruppen det lägsta rangordningsnumret inom gruppen. Om alla rangordningsnummer inom gruppen är högre än det mellersta värdet i den totala rangordningen, får alla rader i gruppen det högsta rangordningsnumret inom gruppen. Om rangordningsnumren inom gruppen spänner över det mellersta värdet i den totala rangordningen, får alla rader i gruppen ett värde som motsvarar medelvärdet av det högsta och det lägsta rangordningsnumret i hela kolumnsegmentet.
1	Lägsta rangordningstalet ges åt alla rader.
2	Medelvärdet ges som rangordningstal åt alla rader.
3	Högsta rangordningstalet ges åt alla rader.
4	Lägsta rangordningstalet på första raden, därefter ökning med ett för varje rad

fmt

Det tredje argumentet, **fmt**, kan ta följande värden:

Exempel med **fmt**

Värde	Beskrivning
0 (standard)	Lägsta värdet - högsta värdet på alla rader (exempelvis 3 - 4).
1	Lägsta värdet på alla rader.
2	Lägsta värdet på första raden, tom sträng på övriga rader.

Ordningen på rader inom **mode** 4 och **fmt**2 bestäms av sorteringsordningen som angivits för dimensionerna i diagrammet.

Exempel och resultat:

Skapa två visualiseringar från dimensionerna Product och Sales och ytterligare en från Product och UnitSales. Lägg till mått på det sätt som visas i nedanstående tabell.

Rangordningsexempel

Exempel	Resultat
Exempel 1. Skapa en tabell med dimensionerna `Customer` och `Sales` och måttet `Rank(Sales)`	Resultatet beror på dimensionernas sorteringsordning. Om tabellen sorteras på Customer listas i tabellen alla värden för Sales för Astrida, sedan Betacab och så vidare. Resultaten för Rank(Sales) visar 10 för Sales-värde 12, 9 för Sales-värde 13, och så vidare, med rangordningsvärdet 1 returnerat för Sales-värde 78. Nästa kolumnsegment börjar med Betacab, där det första värdet för Sales i segmentet är 12. Rangordningsvärdet för Rank(Sales) för detta ges som 11. Om tabellen sorteras på Sales består kolumnsegmenten av värdena för Sales och motsvarande Customer. Eftersom det finns två Sales-värden 12 (för Astrida och Betacab), är Rank (Sales)-värdet för det kolumnsegmentet 1-2 för varje Customer-värde. Detta beror på att det finns två värden för Customer för Sales-värdet 12. Om det hade funnits 4 värden skulle resultatet blivit 1-4 för alla rader. Detta visar hur resultatet ser ut för standardvärdet (0) för argumentet fmt.
Exempel 2. Ersätt kunddimensionen med produktdimensionen och lägg till måttet `Rank(Sales,1,2)`	Detta returnerar 1 på den första raden för varje kolumnsegment och lämnar alla andra rader tomma eftersom argumenten **mode** och **fmt** är inställda på 1 respektive 2.

Resultat för exempel 1, med tabellen sorterad efter Customer:

Resultattabell

Customer	Sales	Rank(Sales)
Astrida	12	10

Customer	Sales	Rank(Sales)
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

Resultat för exempel 1, med tabellen sorterad efter Sales:

Resultattabell

Customer	Sales	Rank(Sales)
Astrida	12	1-2
Betacab	12	1-2
Astrida	13	1
Betacab	15	1
Astrida	20	1
Astrida	22	1-2
Betacab	22	1-2
Betacab	24	1-2
Canutility	24	1-2

Data som används i exempel:

```
ProductData:

Load * inline [

Customer|Product|UnitSales|UnitPrice

Astrida|AA|4|16

Astrida|AA|10|15
```

```
Astrida|BB|9|9

Betacab|BB|5|10

Betacab|CC|2|20

Betacab|DD|0|25

Canutility|AA|8|15

Canutility|CC|0|19

] (delimiter is '|');


Sales2013:
crosstable (Month, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Se även:

- *Sum - diagramfunktion (page 228)*

HRank - diagramfunktion

HRank() utvärderar uttrycket och jämför resultatet med resultaten i de andra kolumnerna som innehåller det aktuella radsegmentet i en pivottabell. Funktionen returnerar sedan rangordningsnumret för den aktuella kolumnen inom segmentet.

Syntax:

```
HRank([ TOTAL ] expr [ , mode [, fmt ] ])
```

Returnerad datatyp: dual

Den här funktionen fungerar endast i pivottabeller. I alla andra diagramtyper returnerar den NULL.

Argument:

Argument

Argument	Beskrivning
expr	Det uttryck eller fält som innehåller de data som ska mätas.
mode	Anger resultatets talrepresentation.
fmt	Anger resultatets textrepresentation.
TOTAL	Om diagrammet är endimensionellt eller om uttrycket föregås av kvalificeraren **TOTAL** utvärderas funktionen längs med hela kolumnen. Om tabellen eller diagrammets tabellmotsvarighet har flera vertikala dimensioner, inbegriper det aktuella kolumnsegmentet endast rader som har samma värden som den aktuella raden i samtliga dimensionskolumner utom den kolumn som visar den sista dimensionen i fältens inbördes sorteringsordning.

Om pivottabellen är endimensionell eller om uttryck föregås av bestämningen **total**, är det aktuella radsegmentet alltid liktydigt med hela raden. Om pivottabellen har flera horisontella dimensioner, inbegriper det aktuella radsegmentet endast kolumner som har samma värden som den aktuella kolumnen i samtliga dimensionsrader utom den rad som visar den sista dimensionen i fältens inbördes sorteringsordning.

Rangordningsnumret returneras som ett dualt värde med en text- och en talrepresentation, vilket i det fall där varje kolumn har ett unikt rangordningsnummer är ett heltal mellan 1 och antalet kolumner i det aktuella radsegmentet.

Om flera kolumner har samma rangordningsnummer, kan text- och talrepresentationerna för gruppen kontrolleras med argumenten **mode** och **format**.

Det andra argumentet, **mode**, anger funktionsresultatets talrepresentation:

Exempel med **mode**

Värde	Beskrivning
0 (standard)	Om alla rangordningsnummer inom gruppen är lägre än det mellersta värdet i den totala rangordningen, får alla kolumner i gruppen det lägsta rangordningsnumret inom gruppen. Om alla rangordningsnummer inom gruppen är högre än det mellersta värdet i den totala rangordningen, får alla kolumner i gruppen det högsta rangordningsnumret inom gruppen. Om rangordningsnumren inom gruppen spänner över det mellersta värdet i den totala rangordningen, får alla rader i gruppen ett värde som motsvarar medelvärdet av det högsta och det lägsta rangordningsnumret i hela kolumnsegmentet.

Värde	Beskrivning
1	Lägsta rangordningsnumret på alla kolumner i gruppen
2	Medelvärdet ges som rangordningsnummer på alla kolumner i gruppen
3	Högsta rangordningsnumret på alla kolumner i gruppen
4	lägsta rangordningsnumret på första kolumnen, därefter ökning med ett för varje kolumn.

Det tredje argumentet, **format**, anger funktionsresultatets textrepresentation:

Exempel med **format**

Värde	Beskrivning
0 (standard)	Lägsta värdet**&' - '&**högsta värdet på alla kolumner i gruppen (exempelvis 3 - 4).
1	Lägsta värdet på alla kolumner i gruppen
2	Lägsta värdet på första kolumnen, tom sträng på följande kolumner i gruppen

Ordningen på kolumnerna inom **mode** 4 och **format**2 bestäms av sorteringsordningen som angivits för dimensionerna i diagrammet.

Exempel:

```
HRank( sum( Sales ))
HRank( sum( Sales ), 2 )
HRank( sum( Sales ), 0, 1 )
```

Optimering med k-medelvärde: Ett exempel från verkligheten

I det här exemplet beskrivs ett verkligt fall där funktionerna k-medelvärdesklustring och centroid används för en datauppsättning. Funktionen KMeans delar upp datapunkter i kluster med liknande egenskaper. Klustren blir mer kompakta och differentierade efter hand som KMeans-algoritmen används upprepade gånger, där antalet kan konfigureras.

KMeans har många användningsområden, och några exempel på hur kluster kan användas är kundsegmentering, upptäcka bedrägerier, förutsäga förlust av konton, målgruppsstyra kundincitament, identifiera cyberkriminella och optimera leveransrutter. Algoritmen för k-medelvärdesklustring används alltmer när företag vill dra slutsatser om mönster och optimera produkterna som erbjuds.

Qlik Sense Funktionerna KMeans och Centroid

I Qlik Sense finns två KMeans-funktioner som grupperar datapunkter i kluster baserat på likheter. Läs mer i *KMeans2D - diagramfunktion (page 747)* och *KMeansND - diagramfunktion (page 758)*. Funktionen **KMeans2D** accepterar två dimensioner och den fungerar bra om du vill illustrera resultatet med ett **spridningsdiagram**. Med funktionen **KMeansND** kan du använda fler än två dimensioner. Eftersom det är

lätt att visa tvådimensionella resultat i standarddiagram, använder vi KMeans i ett **spridningsdiagram** med två dimensioner i följande demonstration. K-medelvärdesklustring kan visualiseras med färgläggning efter uttryck eller efter dimension så som beskrivs i exemplet.

Qlik Senses centroidfunktioner avgör den aritmetiska medelpositionen för alla datapunkter i klustret och identifierar en mittpunkt, eller centroid, för detta kluster. För varje rad i diagrammet (eller post), visar centroidfunktionen den koordinat i klustret som datapunkten har tilldelats. Läs mer i *KMeansCentroid2D - diagramfunktion (page 769)* och *KMeansCentroidND - diagramfunktion (page 770)*.

Översikt över användning och exempel

Följande exempel visar de olika stegen i ett simulerat verkligt scenario. Ett textilföretag i delstaten New York i USA måste minska sina utgifter genom att minimera leveranskostnaderna. Ett sätt att göra detta är att flytta lagren närmare företagets distributörer. Företaget använder 118 distributörer i delstaten New York. Demonstrationen simulerar hur en driftchef delar in distributörer i fem grupper baserat på geografiska områden med hjälp av funktionen KMeans. Sedan identifieras fem optimala lagerplatser som ligger centralt i dessa kluster med hjälp av funktionen Centroid. Målet är att hitta kartkoordinater som kan användas för att hitta fem centrala lagerplatser.

Datauppsättningen

Datauppsättningen baseras på slumpmässigt genererade namn och adresser i delstaten New York, med verkliga koordinater för latitud och longitud. Datauppsättningen innehåller följande tio kolumner: id, first_name, last_name, telephone, address, city, state, zip, latitude, longitude. Nedan finns datauppsättningen som en fil som du kan hämta till en lokal dator och sedan ladda upp till Qlik Sense eller använda inline i Skriptredigeraren. Appen som skapas får namnet *Distributors KMeans and Centroid* och det första arket i appen heter *Distribution cluster analysis*.

Använd den här länken om du vill hämta filen med exempeldata: DistributorData.csv

Distributor-datauppsättning: inline-laddning för Skriptredigeraren i Qlik Sense (page 745)

Rubrik: DistributorData

Totalt antal poster: 118

Använda funktionen KMeans2D

I det här exemplet visas hur du konfigurerar ett **spridningsdiagram**. Datauppsättningen *DistributorData* används, funktionen **KMeans2D** tillämpas och diagrammet färgläggs per dimension.

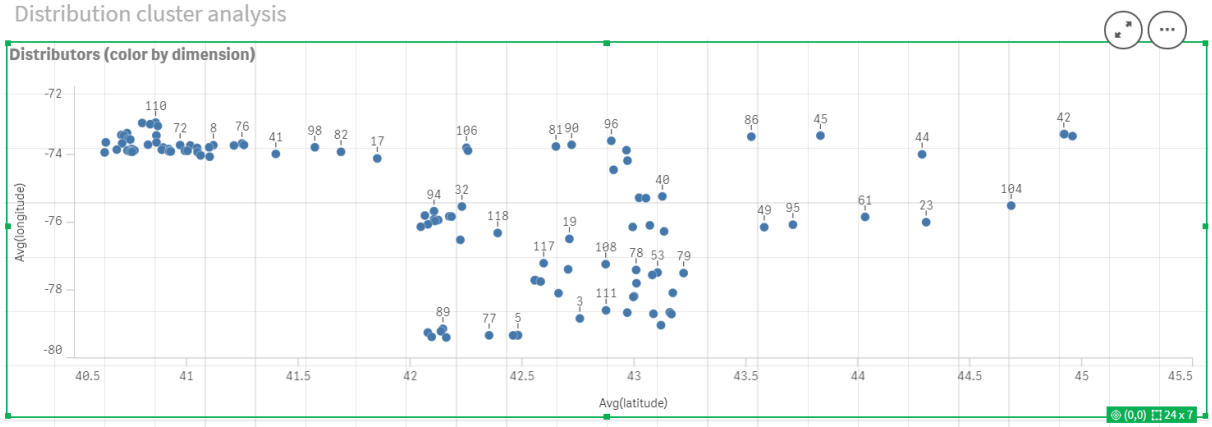
Observera att KMeans-funktionerna i Qlik Sense stöder automatisk klustring med en metod som kallas djupskillnad (depth difference, DeD). När användaren anger 0 som antal kluster bestäms det optimala antalet kluster för den datauppsättningen. I det här exemplet skapas dock en variabel för **num_clusters**-argumentet (syntaxen beskrivs i *KMeans2D - diagramfunktion (page 747)*). Önskat antal kluster (k=5) specificeras därför av en variabel.

1. Ett **spridningsdiagram** dras till arket och får namnet *Distributors (by dimension)*.
2. En **variabel** skapas för att ange antalet kluster. **Variabeln** ges namnet *vDistClusters*. För variabeln **Definition** anges *5*.
3. Konfiguration av **Data** för diagrammet:

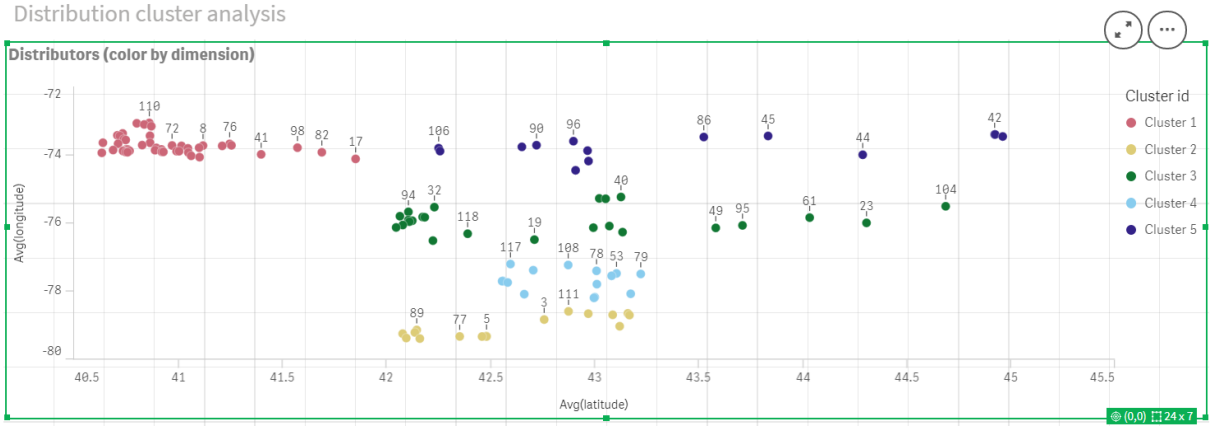
a. Under **Dimensioner** väljs fältet *id* för **Bubbla**. *Cluster id* anges för **Etikett**.
b. Under **Mått** är *Avg([latitude])* uttrycket för **X-axel**.
c. Under **Mått** är *Avg([longitude])* uttrycket för **Y-axel**.

4. Konfiguration av **Utseende**:
   a. Under **Färger och teckenförklaring** väljs **Anpassad** för **Färger**.
   b. **Per dimension** väljs för färgläggning av diagrammet.
   c. Följande uttryck skrivs: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only (longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
   d. Kryssrutan vid **Låsta färger** markeras.

Spridningsdiagram innan funktionen KMeans med färgläggning per dimension används



Spridningsdiagram efter att funktionen KMeans med färgläggning per dimension används



Lägger till en **tabell**: *Distributörer*

Det kan vara praktiskt att ha en tabell där du snabbt kan titta på relevanta data. I **spridningsdiagrammet** visas *ID:n* i en tabell där motsvarande distributörsnamn har lagts till som referens.

1. En **tabell** med namnet *Distributors* dras till arket där följande **Kolumner** (Dimensioner) har lagts till: *id*, *first_name* och *last_name*.

Tabell: Distributörernas namn

Distributors

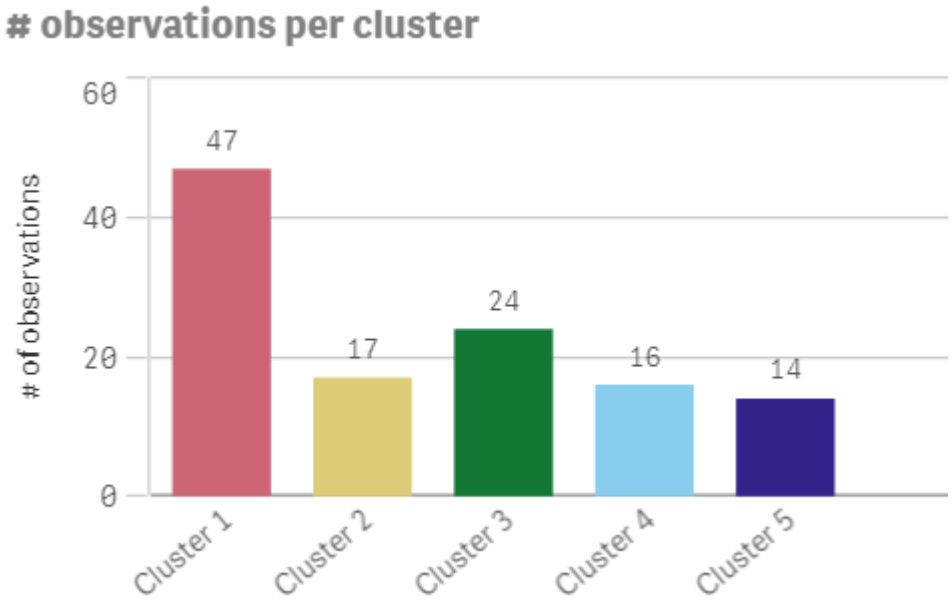
id	first_name	last_name
1	Kaiya	Snow
2	Dean	Roy
3	Eden	Paul
4	Bryanna	Higgins
5	Elisabeth	Lee
6	Skylar	Robinson
7	Cody	Bailey
8	Dario	Sims
9	Deacon	Hood

Lägger till ett **stapeldiagram**: *# observations per cluster*

För exemplet med placering av lager är det praktiskt att veta hur många distributörer som ska använda varje lager. Därför skapas ett **stapeldiagram** som visar hur många distributörer som ingår i varje kluster.

1. Ett **stapeldiagram** dras till arket. Diagrammet ges namnet: *# observations per cluster*.
2. Konfiguration av **Data** för **stapeldiagrammet**:
    a. En **Dimension** med etiketten *Clusters* läggs till (etiketten kan läggas till efter att uttrycket har tillämpats). Följande uttryck skrivs: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    b. Ett **Mått** med etiketten *# of observations* läggs till. Följande uttryck skrivs: *=count(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id))*
3. Konfiguration av **Utseende**:
    a. Under **Färger och teckenförklaring** väljs **Anpassad** för **Färger**.
    b. **Per dimension** väljs för färgläggning av diagrammet.
    c. Följande uttryck skrivs: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    d. Kryssrutan vid **Låsta färger** markeras.
    e. **Visa teckenförklaring** inaktiveras.
    f. Under **Presentation** anges **Auto** för **Värdeetiketter**.
    g. Under **X-axel**: **Kluster**, **Endast etiketter** väljs.

Stapeldiagram: antal observationer per kluster



Använda funktionen **Centroid2D**

En andra tabell läggs till för funktionen **Centroid2D** som kommer att identifiera koordinaterna för potentiella lagerplatser. I tabellen visas den centrala platsen (mittpunktsvärdena) för de fem identifierade distributörsgrupperna.

1. En **Tabell** dras till arket och ges namnet *Cluster centroids*. Följande kolumner läggs till:
    a. En **Dimension** med namnet *Clusters* läggs till. Följande uttryck skrivs: *=pick(aggr (KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1,'Warehouse 1','Warehouse 2','Warehouse 3','Warehouse 4','Warehouse 5'*)
    b. Ett **Mått** med namnet *latitude (D1)* läggs till. Följande uttryck skrivs: *=only(aggr (KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id))*
    Notera att parametern **coordinate_no** motsvarar första dimensionen, dimension(0). I det här fallet ritas dimensionen *latitude* på x-axeln. Om vi arbetade med funktionen **CentroidND** och det fanns upp till sex dimensioner, kunde dessa parametrar bara vara något av sex värden: 0, 1, 2, 3, 4 eller 5.
    c. Ett **Mått** med namnet *longitude (D2)* läggs till. Följande uttryck skrivs: *=only(aggr (KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id))*
    Parametern **coordinate_no** i det här uttrycket motsvarar den andra dimensionen, dimension (1). Dimensionen *longitude* ritas på y-axeln.

Tabell: Beräkningar för klustrens mittpunkter

Cluster centroids

Clusters	latitude (D1)	longitude (D2)
Totals	-	-
Warehouse 1	40.945422240426	-73.719966482979
Warehouse 2	42.590538729412	-79.067889217647
Warehouse 3	42.805089516667	-75.901621883333
Warehouse 4	42.8581692625	-77.6800485875
Warehouse 5	43.436770771429	-73.734622635714

Placering av mittpunkter på karta

Nästa steg är att placera mittpunkterna på kartan. Om så önskas kan apputvecklaren placera visualiseringen på olika ark.

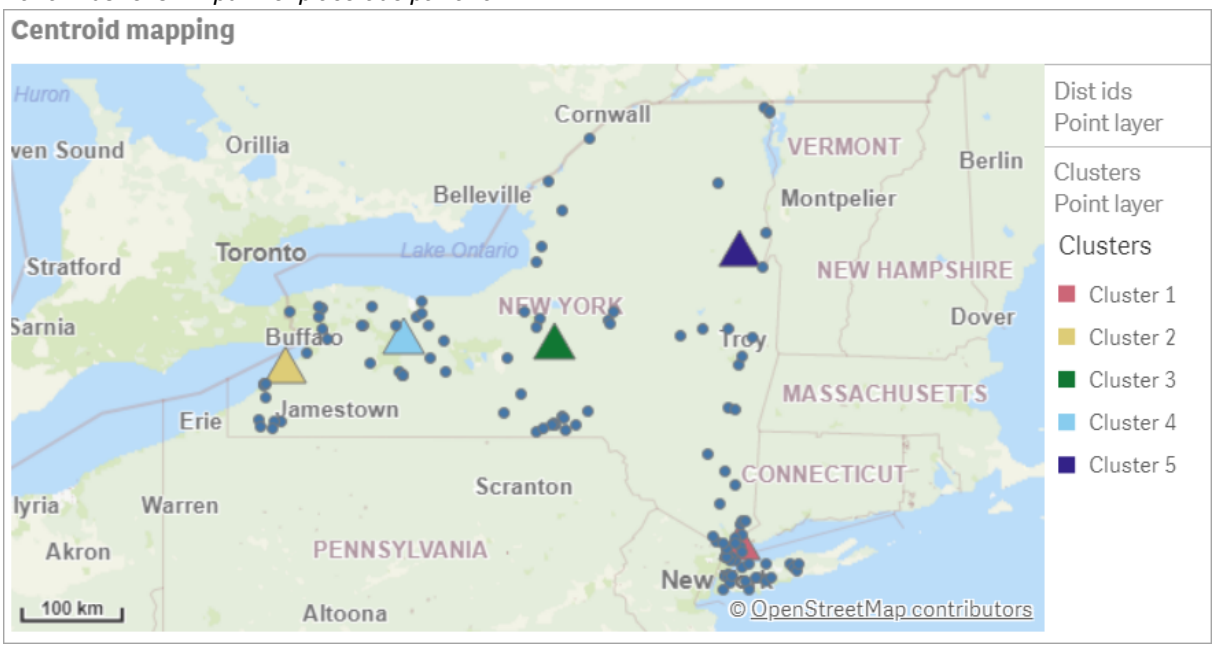
1. En **karta** med namnet *Centroid mapping* dras till arket.
2. I delavsnittet om**skikt** är **lägg till skikt** valt, sedan väljs **punktskikt**.
    a. I **Fält** väljs *id*, och *Dist ids* läggs till vid **Etikett**.
    b. I delavsnittet **Plats** väljs kryssrutan **Latitud- och longitudfält**.
    c. Vid **Latitud** väljs fältet *latitude*.
    d. Vid **Longitud** väljs fältet *longitude*.
    e. I delavsnittet **Storlek & form** väljs **Bubbla** vid **Form**, och **Storlek** minskas till önskad storlek med skjutreglaget.
    f. I delavsnittet **Färger** väljs **En färg**. Blått väljs för **Färg** och grått för **Konturfärg** (dessa val kan ändras enligt önskemål).
3. I delavsnittet **Skikt** läggs ett andra **Punktskikt** till genom att välja först **Lägg till skikt** och sedan **Punktskikt**.
    a. Följande uttryck skrivs: *=aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)*
    b. Vid **Etikett** läggs *Clusters* till.
    c. I delavsnittet **Plats** väljs kryssrutan **Latitud- och longitudfält**.
    d. Vid **Latitud**, som i det här fallet ritas längs x-axeln, läggs följande uttryck till: *=aggr (KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id)*
    e. Vid **Longitud**, som i det här fallet ritas längs y-axeln, läggs följande uttryck till: *=aggr (KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id)*
    f. I delavsnittet **Storlek & form** väljs **Triangel** vid **Form**, och **Storlek** minskas till önskad storlek med skjutreglaget.
    g. Under **Färger och teckenförklaring** väljs **Anpassad** vid **Färger**.

h. **Per dimension** väljs för färgläggning av diagrammet. Följande uttryck skrivs: *=pick(aggr (KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1,'Cluster 1','Cluster 2','Cluster 3','Cluster 4','Cluster 5')*

i. Dimensionen ges etiketten *Clusters*.

4. I **Kartinställningar** väljs **Anpassad** vid **Projektion**. **Metersystemet** väljs vid **Mätenheter**.

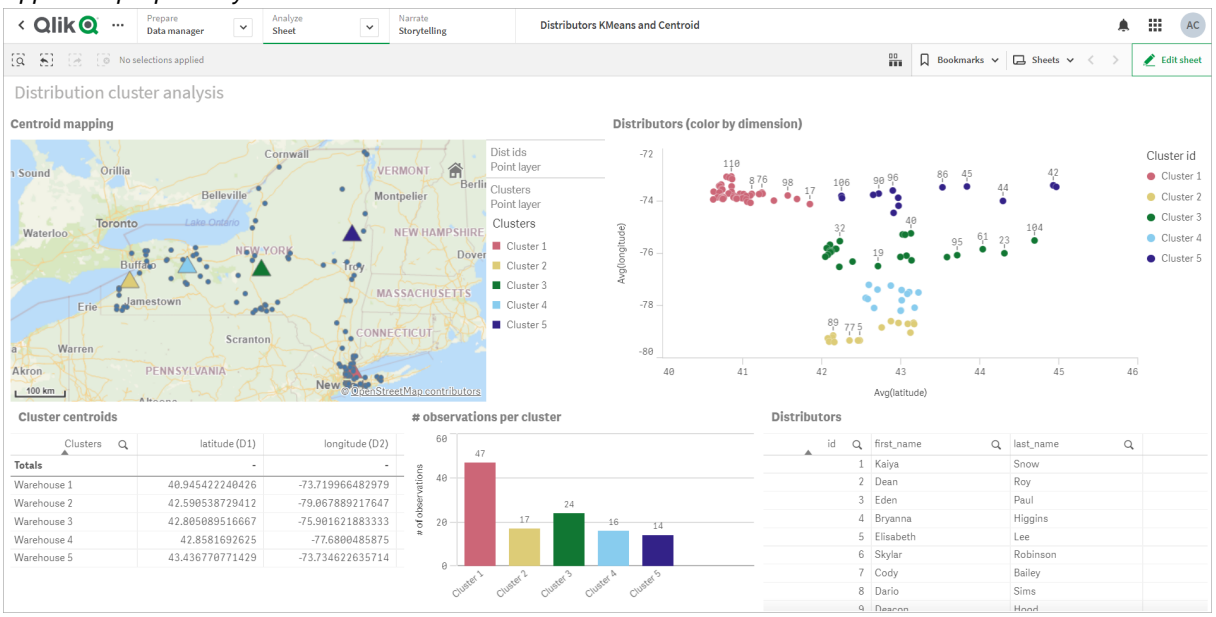
Karta: klustrens mittpunkter placerade på kartan



Slutsats

I det här exemplet från verkligheten har funktionen KMeans använts för att dela in distributörer i grupper eller kluster baserat på likhet; i det här fallet hur nära de ligger varandra geografiskt. Funktionen Centroid tillämpades på klustren för att identifiera fem uppsättningar kartkoordinater. Dessa koordinater ger en första central plats där lager kan byggas eller sökas. Centroid-funktionen tillämpas på **kartdiagrammet** så att appanvändare kan se var mittpunkterna är belägna i relation till datapunkterna i klustret. Resulterande koordinater representerar potentiella platser för lager, som skulle kunna minska leveranskostnader för distributörer i delstaten New York.

App: exempel på analys med KMeans och Centroid



Distributor-datauppsättning: inline-laddning för Skriptredigeraren i Qlik Sense

```
DistributorData: Load * Inline [ id,first_name,last_
name,telephone,address,city,state,zip,latitude,longitude 1,Kaiya,Snow,(716) 201-1212,6231
Tonawanda Creek Rd #APT 308,Lockport,NY,14094,43.08926,-78.69313 2,Dean,Roy,(716) 201-
1588,6884 E High St,Lockport,NY,14094,43.16245,-78.65036 3,Eden,Paul,(716) 202-4596,4647
Southwestern Blvd #APT 350,Hamburg,NY,14075,42.76003,-78.83194 4,Bryanna,Higgins,(716) 203-
7041,418 Park Ave,Dunkirk,NY,14048,42.48279,-79.33088 5,Elisabeth,Lee,(716) 203-7043,36 E
Courtney St,Dunkirk,NY,14048,42.48299,-79.31928 6,Skylar,Robinson,(716) 203-7166,26 Greco
Ln,Dunkirk,NY,14048,42.4612095,-79.3317925 7,Cody,Bailey,(716) 203-7201,114 Lincoln
Ave,Dunkirk,NY,14048,42.4801269,-79.322232 8,Dario,Sims,(408) 927-1606,N Castle
Dr,Armonk,NY,10504,41.11979,-73.714864 9,Deacon,Hood,(410) 244-6221,4856 44th
St,Woodside,NY,11377,40.748372,-73.905445 10,Zackery,Levy,(410) 363-8874,61 Executive
Blvd,Farmingdale,NY,11735,40.7197457,-73.430239 11,Rey,Hawkins,(412) 344-8687,4585 Shimerville
Rd,Clarence,NY,14031,42.972075,-78.6592452 12,Phillip,Howard,(413) 269-4049,464 Main St
#101,Port Washington,NY,11050,40.8273756,-73.7009971 13,Shirley,Tyler,(434) 985-8943,114 Glann
Rd,Apalachin,NY,13732,42.0482515,-76.1229725 14,Aniyah,Jarvis,(440) 244-1808,87 N Middletown
Rd,Pearl River,NY,10965,41.0629,-74.0159 15,Alayna,Woodard,(478) 335-3704,70 W Red Oak Ln,West
Harrison,NY,10604,41.0162722,-73.7234926 16,Jermaine,Lambert,(508) 561-9836,24 Kellogg Rd,New
Hartford,NY,13413,43.0555739,-75.2793197 17,Harper,Gibbs,(239) 466-0238,Po Box
33,Cottekill,NY,12419,41.853392,-74.106082 18,Osvaldo,Graham,(252) 246-0816,6878 Sand Hill
Rd,East Syracuse,NY,13057,43.073215,-76.081448 19,Roberto,Wade,(270) 469-1211,3936 Holley
Rd,Moravia,NY,13118,42.713044,-76.481227 20,Kate,Mcguire,(270) 788-3080,6451 State 64 Rte
#3,Naples,NY,14512,42.707366,-77.380489 21,Dale,Andersen,(281) 480-5690,205 W Service
Rd,Champlain,NY,12919,44.9645392,-73.4470831 22,Lorelai,Burch,(302) 644-2133,1 Brewster
St,Glen Cove,NY,11542,40.865177,-73.633019 23,Amiyah,Flowers,(303) 223-0055,46600 Us
Interstate 81 Rte,Alexandria Bay,NY,13607,44.309626,-75.988365 24,Mckinley,Clements,(303) 918-
3230,200 Summit Lake Dr,Valhalla,NY,10595,41.101145,-73.778298 25,Marc,Gibson,(607) 203-
1233,25 Robinson St,Binghamton,NY,13901,42.107416,-75.901614 26,Kali,Norman,(607) 203-1400,1
Ely Park Blvd #APT 15,Binghamton,NY,13905,42.125866,-75.925026 27,Laci,Cain,(607) 203-1437,16
Zimmer Road,Kirkwood,NY,13795,42.066516,-75.792627 28,Mohammad,Perez,(607) 203-1652,71
```

Endicott Ave #APT 12,Johnson City,NY,13790,42.111894,-75.952187 29,Izabelle,Pham,(607) 204-0392,434 State 369 Rte,Port Crane,NY,13833,42.185838,-75.823074 30,Kiley,Mays,(607) 204-0870,244 Ballyhack Rd #14,Port Crane,NY,13833,42.175612,-75.814917 31,Peter,Trevino,(607) 205-1374,125 Melbourne St.,Vestal,NY,13850,42.080254,-76.051124 32,Ani,Francis,(607) 208-4067,48 Caswell St,Afton,NY,13730,42.232065,-75.525674 33,Jared,Sheppard,(716) 386-3002,4709 430th Rte,Bemus Point,NY,14712,42.162175,-79.39176 34,Dulce,Atkinson,(914) 576-2266,501 Pelham Rd,New Rochelle,NY,10805,40.895449,-73.782602 35,Jayla,Beasley,(716) 526-1054,5010 474th Rte,Ashville,NY,14710,42.096859,-79.375561 36,Dane,Donovan,(718) 545-3732,5014 31st Ave,Woodside,NY,11377,40.756967,-73.909506 37,Brendon,Clay,(585) 322-7780,133 Cummings Ave,Gainesville,NY,14066,42.664309,-78.085651 38,Asia,Nunez,(718) 426-1472,2407 Gilmore ,East Elmhurst,NY,11369,40.766662,-73.869185 39,Dawson,Odonnell,(718) 342-2179,5019 H Ave,Brooklyn,NY,11234,40.633245,-73.927591 40,Kyle,Collins,(315) 733-7078,502 Rockhaven Rd,Utica,NY,13502,43.129184,-75.226726 41,Eliza,Hardin,(315) 331-8072,502 Sladen Place,West Point,NY,10996,41.3993,-73.973003 42,Kasen,Klein,(518) 298-4581,2407 Lake Shore Rd,Chazy,NY,12921,44.925561,-73.387373 43,Reuben,Bradford,(518) 298-4581,33 Lake Flats Dr,Champlain,NY,12919,44.928092,-73.387884 44,Henry,Grimes,(518) 523-3990,2407 Main St,Lake Placid,NY,12946,44.291487,-73.98474 45,Kyan,Livingston,(518) 585-7364,241 Alexandria Ave,Ticonderoga,NY,12883,43.836553,-73.43155 46,Kaitlyn,Short,(516) 678-3189,241 Chance Dr,Oceanside,NY,11572,40.638534,-73.63079 47,Damaris,Jacobs,(914) 664-5331,241 Claremont Ave,Mount Vernon,NY,10552,40.919852,-73.827848 48,Alivia,Schroeder,(315) 469-4473,241 Lafayette Rd,Syracuse,NY,13205,42.996446,-76.12957 49,Bridget,Strong,(315) 298-4355,241 Maltby Rd,Pulaski,NY,13142,43.584966,-76.136317 50,Francis,Lee,(585) 201-7021,166 Ross St,Batavia,NY,14020,43.0031502,-78.17487 51,Makaila,Phelps,(585) 201-7422,58 S Main St,Batavia,NY,14020,42.99941,-78.1939285 52,Jazlynn,Stephens,(585) 203-1087,1 Sinclair Dr,Pittsford,NY,14534,43.084157,-77.545452 53,Ryann,Randolph,(585) 203-1519,331 Eaglehead Rd,East Rochester,NY,14445,43.10785,-77.475552 54,Rosa,Baker,(585) 204-4011,42 Ossian St,Dansville,NY,14437,42.560761,-77.70088 55,Marcel,Barry,(585) 204-4013,42 Jefferson St,Dansville,NY,14437,42.557735,-77.702983 56,Dennis,Schmitt,(585) 204-4061,750 Dansville Mount Morris Rd,Dansville,NY,14437,42.584458,-77.741648 57,Cassandra,Kim,(585) 204-4138,3 Perine Ave APT1,Dansville,NY,14437,42.562865,-77.69661 58,Kolton,Jacobson,(585) 206-5047,4925 Upper Holly Rd,Holley,NY,14470,43.175957,-78.074465 59,Nathanael,Donovan,(718) 393-3501,9604 57th Ave,Corona,NY,11373,40.736077,-73.864858 60,Robert,Frazier,(718) 271-3067,300 56th Ave,Corona,NY,11373,40.735304,-73.873997 61,Jessie,Mora,(315) 405-8991,9607 Forsyth Loop,Watertown,NY,13603,44.036466,-75.833437 62,Martha,Rollins,(347) 242-2642,22 Main St,Corona,NY,11373,40.757727,-73.829331 63,Emely,Townsend,(718) 699-0751,60 Sanford Ave,Corona,NY,11373,40.755466,-73.831029 64,Kylie,Cooley,(347) 561-7149,9608 95th Ave,Ozone Park,NY,11416,40.687564,-73.845715 65,Wendy,Cameron,(585) 571-4185,9608 Union St,Scottsville,NY,14546,43.013327,-77.7907839 66,Kayley,Peterson,(718) 654-5027,961 E 230th St,Bronx,NY,10466,40.889275,-73.850555 67,Camden,Ochoa,(718) 760-8699,59 Vark St,Yonkers,NY,10701,40.929322,-73.89957 68,Priscilla,Castillo,(910) 326-7233,9359 Elm St,Chadwicks,NY,13319,43.024902,-75.26886 69,Dana,Schultz,(913) 322-4580,99 Washington Ave,Hastings on Hudson,NY,10706,40.99265,-73.879748 70,Blaze,Medina,(914) 207-0015,60 Elliott Ave,Yonkers,NY,10705,40.921498,-73.896682 71,Finnegan,Tucker,(914) 207-0015,90 Hillside Drive,Yonkers,NY,10705,40.922514,-73.892911 72,Pranav,Palmer,(914) 214-8376,5 Bruce Ave,Harrison,NY,10528,40.970916,-73.711493 73,Kolten,Wong,(914) 218-8268,70 Barker St,Mount Kisco,NY,10549,41.211993,-73.723202 74,Jasiah,Vazquez,(914) 231-5199,30 Broadway,Dobbs Ferry,NY,10522,41.004629,-73.879825 75,Lamar,Pierce,(914) 232-0380,68 Ridge Rd,Katonah,NY,10536,41.256662,-73.707964 76,Carla,Coffey,(914) 232-0469,197 Beaver Dam Rd,Katonah,NY,10536,41.247934,-73.664363 77,Brooklynn,Harmon,(716) 595-3227,8084 Glasgow Rd,Cassadega,NY,14718,42.353861,-79.329558 78,Raquel,Hodges,(585) 398-8125,809 County Road ,Victor,NY,14564,43.011745,-77.398806 79,Jerimiah,Gardner,(585) 787-9127,809 Houston Rd,Webster,NY,14580,43.224204,-77.491353 80,Clarence,Hammond,(720) 746-1619,809 Pierpont Ave,Piermont,NY,10968,41.0491181,-73.918622 81,Rhys,Gill,(518) 427-7887,81 Columbia St,Albany,NY,12210,42.652824,-73.752096 82,Edith,Parrish,(845) 452-7621,81 Glenwood Ave,Poughkeepsie,NY,12603,41.691058,-73.910829 83,Kobe,Mcintosh,(845) 371-1101,81 Heitman Dr,Spring Valley,NY,10977,41.103227,-74.054396 84,Ayden,Waters,(516) 796-2722,81 Kingfisher

```
Rd,Levittown,NY,11756,40.738939,-73.52826 85,Francis,Rogers,(631) 427-7728,81 Knollwood
Ave,Huntington,NY,11743,40.864905,-73.426107 86,Jaden,Landry,(716) 496-4038,12839 39th
Rte,Chaffee,NY,14030,43.527396,-73.462786 87,Giancarlo,Campos,(518) 885-5717,1284 Saratoga
Rd,Ballston Spa,NY,12020,42.968594,-73.862847 88,Eduardo,Contreras,(716) 285-8987,1285
Saunders Sett Rd,Niagara Falls,NY,14305,43.122963,-79.029274 89,Gabriela,Davidson,(716) 267-
3195,1286 Mee Rd,Falconer,NY,14733,42.147339,-79.137976 90,Evangeline,Case,(518) 272-9435,1287
2nd Ave,Watervliet,NY,12189,42.723132,-73.703818 91,Tyrone,Ellison,(518) 843-4691,1287 Midline
Rd,Amsterdam,NY,12010,42.9730876,-74.1700608 92,Bryce,Bass,(518) 943-9549,1288 Leeds Athens
Rd,Athens,NY,12015,42.259381,-73.876897 93,Londyn,Butler,(518) 922-7095,129 Argersinger
Rd,Fultonville,NY,12072,42.910969,-74.441917 94,Graham,Becker,(607) 655-1318,129 Baker
Rd,Windsor,NY,13865,42.107271,-75.66408 95,Rolando,Fitzgerald,(315) 465-4166,17164 County 90
Rte,Mannsville,NY,13661,43.713443,-76.06232 96,Grant,Hoover,(518) 692-8363,1718 County 113
Rte,Schaghticote,NY,12154,42.900648,-73.585036 97,Mark,Goodwin,(631) 584-6761,172 Cambon
Ave,Saint James,NY,11780,40.871152,-73.146032 98,Deacon,Cantu,(845) 221-7940,172 Carpenter
Rd,Hopewell Junction,NY,12533,41.57388,-73.77609 99,Tristian,Walsh,(516) 997-4750,172 E Cabot
Ln,Westbury,NY,11590,40.7480397,-73.54819 100,Abram,Alexander,(631) 588-3817,172 Lorenzo
Cir,Ronkonkoma,NY,11779,40.837123,-73.09367 101,Lesly,Bush,(516) 489-3791,172 Nassau
Blvd,Garden City,NY,11530,40.71147,-73.660753 102,Pamela,Espinoza,(716) 201-1520,172 Niagara
St ,Lockport,NY,14094,43.169871,-78.70093 103,Bryanna,Newton,(914) 328-4332,172 Warren
Ave,White Plains,NY,10603,41.047207,-73.79572 104,Marcelo,Schmitt,(315) 393-4432,319 Mansion
Ave,Ogdensburg,NY,13669,44.690246,-75.49992 105,Layton,Valenzuela,(631) 676-2113,319
Singingwood Dr,Holbrook,NY,11741,40.801391,-73.058993 106,Roderick,Rocha,(518) 671-6037,319
Warren St,Hudson,NY,12534,42.252527,-73.790629 107,Camryn,Terrell,(315) 635-1680,3192 Olive
Dr,Baldinsville,NY,13027,43.136843,-76.260303 108,Summer,Callahan,(585) 394-4195,3192 Smith
Road,Canandaigua,NY,14424,42.875457,-77.228039 109,Pierre,Novak,(716) 665-2524,3194 Falconer
Kimball Stand Rd,Falconer,NY,14733,42.138439,-79.211091 110,Kennedi,Fry,(315) 543-2301,32
College Rd,Selden,NY,11784,40.861624,-73.04757 111,Wyatt,Pruitt,(716) 681-4042,277 Ransom
Rd,Lancaster ,NY,14086,42.87702,-78.591302 112,Lilly,Jensen,(631) 841-0859,2772 Schliegel
Blvd,Amityville,NY,11701,40.708021,-73.413015 113,Tristin,Hardin,(631) 920-0927,278 Fulton
Street,West Babylon,NY,11704,40.733578,-73.357321 114,Tanya,Stafford,(716) 484-0771,278
Sampson St,Jamestown,NY,14701,42.0797,-79.247805 115,Paris,Cordova,(607) 589-4857,278 Washburn
Rd,Spencer,NY,14883,42.225046,-76.510257 116,Alfonso,Morse,(718) 359-5582,200 Colden
St,Flushing,NY,11355,40.750403,-73.822752 117,Maurice,Hooper,(315) 595-6694,4435 Italy Hill
Rd,Branchport,NY,14418,42.597957,-77.199267 118,Iris,Wolf,(607) 539-7288,444 Harford
Rd,Brooktondale,NY,14817,42.392164,-76.30756 ];
```

KMeans2D - diagramfunktion

KMeans2D() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring, och för varje diagramrad visas kluster-ID för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1 respektive coordinate_2. Dessa är båda aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters. Data normaliseras med normparametern (valfritt).

KMeans2D returnerar ett värde per datapunkt. Det returnerade värdet är en dual och är det heltalsvärde som motsvarar klustret varje datapunkt har tilldelats till.

Syntax:

```
KMeans2D(num_clusters, coordinate_1, coordinate_2 [, norm])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
num_clusters	Heltal som anger antalet kluster.
coordinate_1	Aggregeringen som beräknar den första koordinaten, vanligtvis x-axeln på spridningsdiagrammet som kan skapas från diagrammet. Den ytterligare parametern, coordinate_2, beräknar den andra koordinaten.
norm	Den valfria normaliseringsmetoden tillämpas på datauppsättningen före k-medelvärdesklustring. Möjliga värden: 0 eller ”none” för ingen normalisering 1 eller ”zscore” för z-poängsnormalisering 2 eller ”minmax” för min-max-normalisering Om ingen parameter anges eller om den angivna parametern är felaktig används ingen normalisering. Z-poäng normaliserar data baserat på funktionens median och standardavvikelse. Z-poäng säkerställer inte att varje funktion har samma skala men det fungerar bättre än min-max för behandling av outliers. Min-max-normalisering säkerställer att funktionerna har samma skala genom att ta de minsta och största värdena för varje och räkna om varje datapunkt.

Exempel: Diagramuttryck
I det här exemplet skapar vi ett spridningsdiagram med datauppsättningen *Iris*, och använder sedan KMeans för att färglägga data efter uttryck.

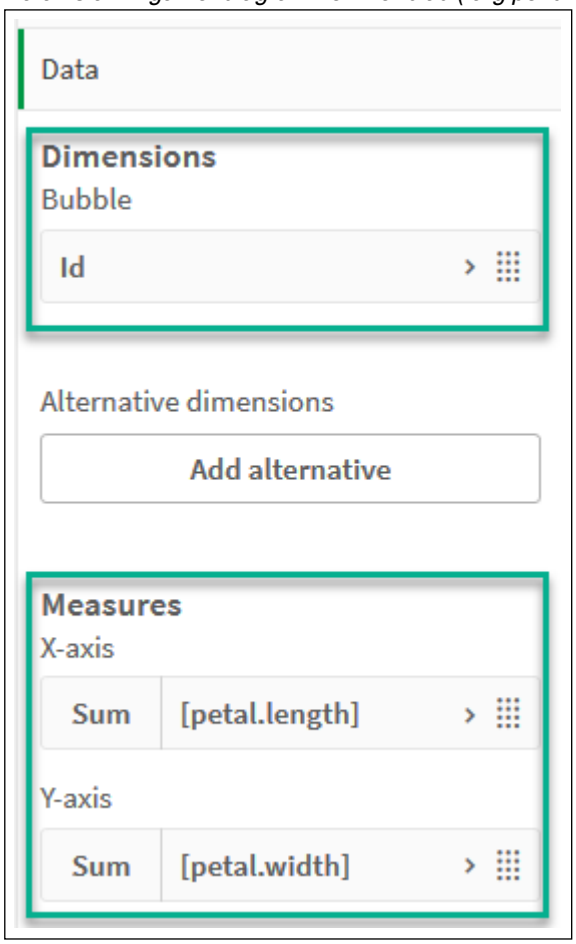
Vi skapar även en variabel för argumentet *num_clusters*, och sedan använder vi en variabelinmatningsruta för att ändra antalet kluster.

Datauppsättningen *Iris* finns offentligt tillgänglig i en mängd format. Vi tillhandahåller dessa data som en inline-tabell som laddas med Skriptredigeraren i Qlik Sense. Observera att vi lagt till en *ID*-kolumn i datatabellen i det här exemplet.

När vi laddat data i Qlik Sense, gör vi följande:

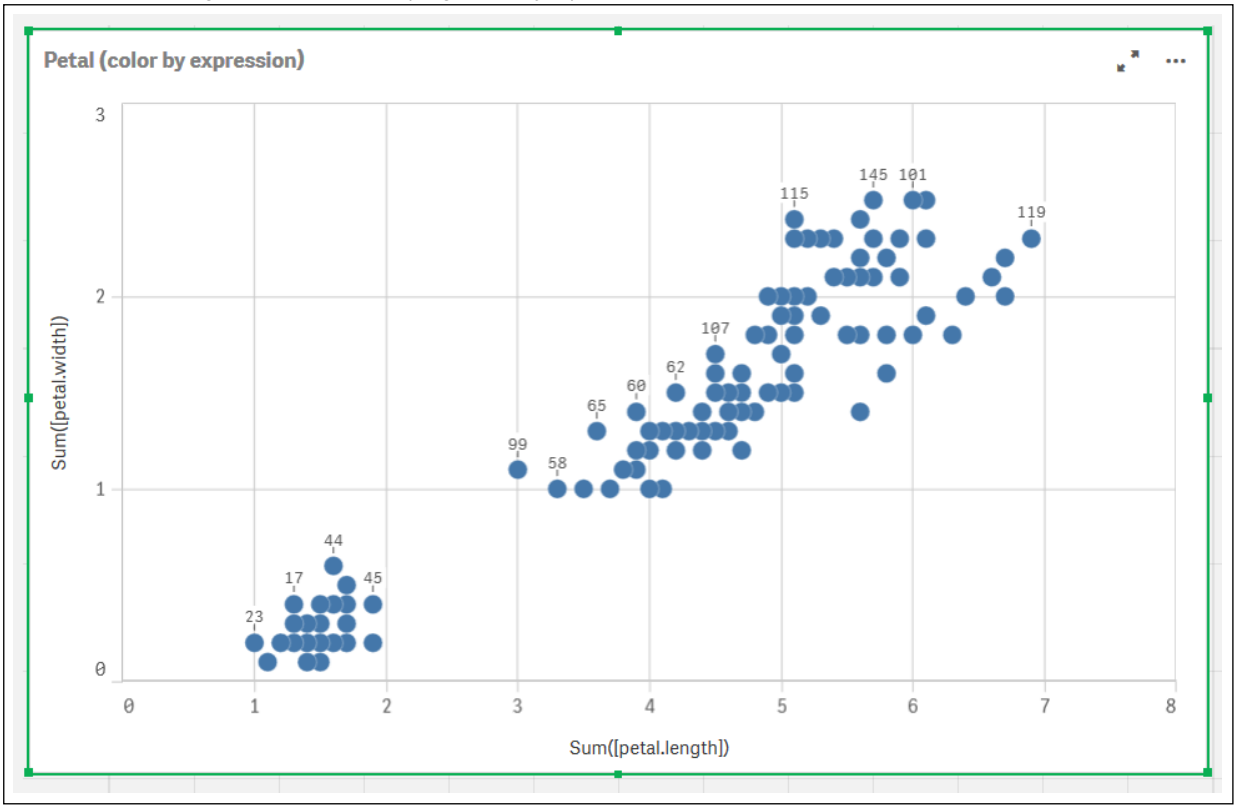
1. Dra ett **spridningsdiagram** till ett nytt ark. Döp diagrammet till *Kronblad (färg per uttryck)*.
2. Skapa en variabel för att ange antalet kluster. För variabeln **Namn** anger du *KmeansPetalClusters*. För variabeln **Definition** anger du *=2*.
3. Konfigurera **Data** för diagrammet:
    i. Under **Dimensioner** väljer du *id* för fältet **Bubbla**. Ange ett kluster-ID för Etikett.
    ii. Under **Mått** väljer du *Sum([petal.length])* för uttrycket för **X-axel**.
    iii. Under **Mått** väljer du *Sum([petal.width])* för uttrycket för **Y-axel**.

    Datainställningar för diagrammet Kronblad (färg per uttryck)



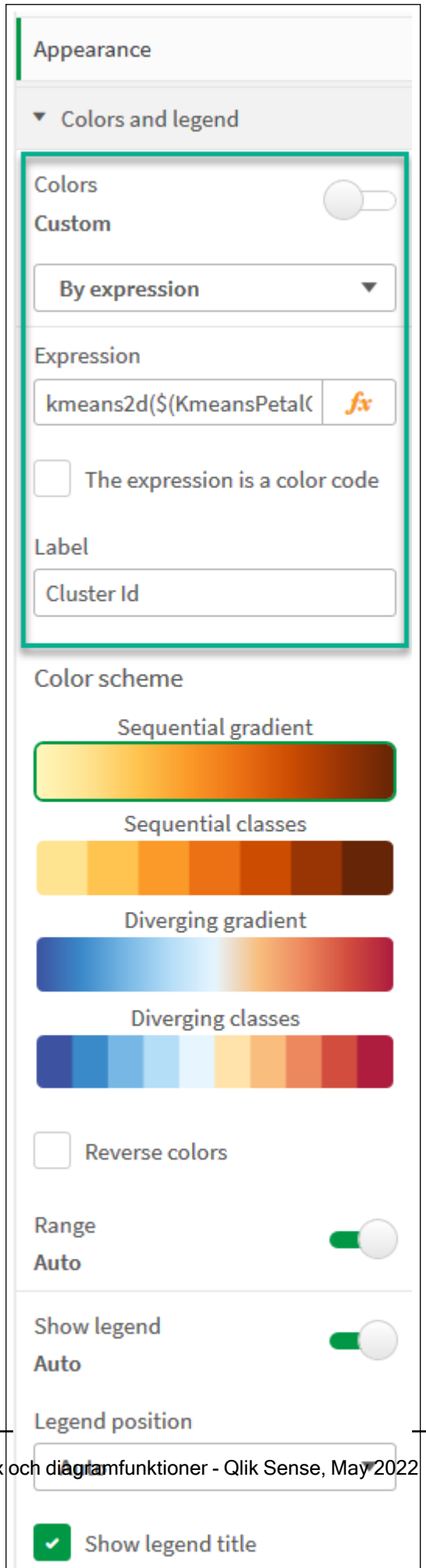
Datapunkterna ritas ut på diagrammet.

Datapunkter på diagrammet Kronblad (färg per uttryck)



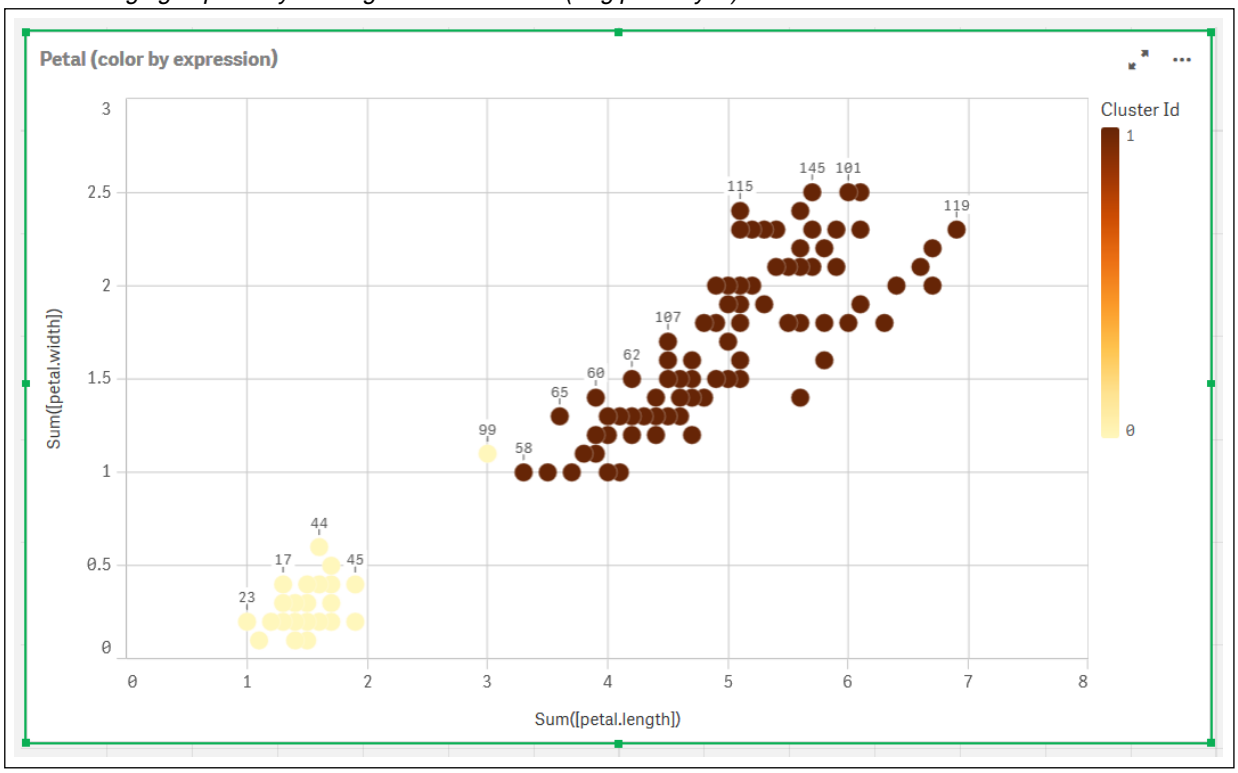
4. Konfigurera **Utseende** för diagrammet:
    i. Under **Färger och teckenförklaring** väljer du **Anpassad** för **Färger**.
    ii. Välj att färglägga diagrammet**Per uttryck**.
    iii. Ange följande för **Uttryck**: *kmeans2d($(KmeansPetalClusters), Sum([petal.length]), Sum ([petal.width]))*
    Observera att *KmeansPetalClusters* är variabeln vi angav som *2*.
    Eller ange följande: *kmeans2d(2, Sum([petal.length]), Sum([petal.width]))*
    iv. Avmarkera kryssrutan **Uttrycket är en färgkod**.
    v. Ange följande för **Etikett**: *Kluster-ID*

Utseendeinställningar för diagrammet Kronblad (färg per uttryck)


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
Show legend title

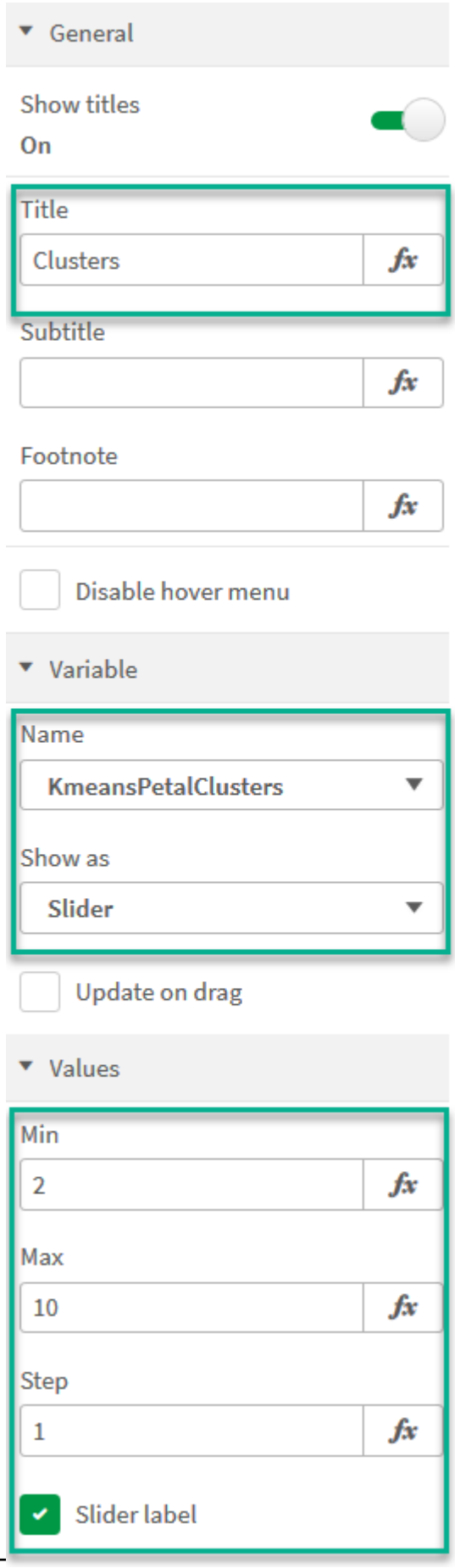
De två klustren i diagrammet färgläggs per KMeans-uttrycket.

Klustren färglagda per uttryck i diagrammet Kronblad (färg per uttryck)



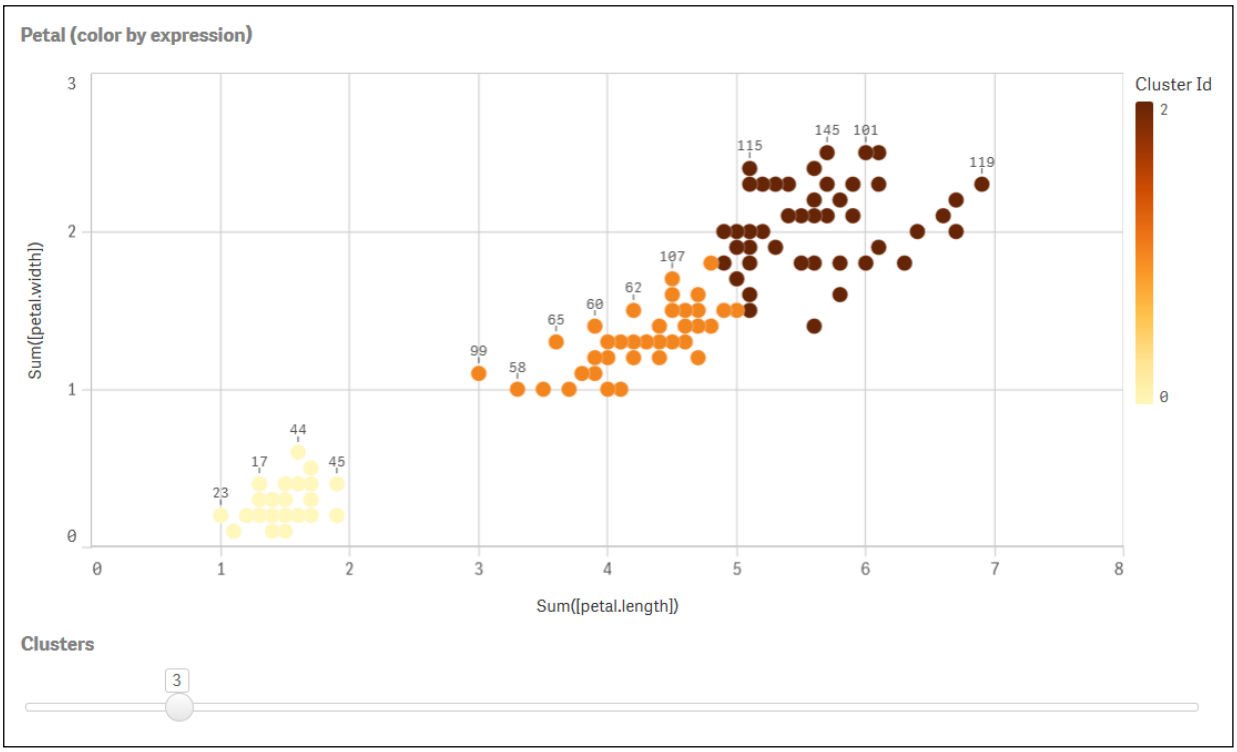
5. Lägg till en ruta för **Variabelinmatning** för antalet kluster.
    i. Under **Anpassade objekt** i panelen **Resurser** väljer du **Qliks instrumentpanelspaket**. Om vi inte hade tillgång till instrumentpanelspaketet kunde vi ändå ändra antalet kluster med variabeln vi skapat, eller direkt som ett heltal i uttrycket.
    ii. Dra en ruta för **Variabelinmatning** till arket.
    iii. Under **Utseende** klickar du på **Allmänt**.
    iv. Ange följande för **Rubrik**: *Kluster*
    v. Klicka på **Variabel**.
    vi. Välj följande variabel för **Namn**: *KmeansPetalClusters*.
    vii. Välj **Skjutreglage** för **Visa som**.
    viii. Välj **Värden** och konfigurera inställningarna efter behov,

Utseende för variabelinmatningsrutan Kluster


General
Show titles
On
Title
Clusters
Subtitle
Footnote
Disable hover menu
Variable
Name
KmeansPetalClusters
Show as
Slider
Update on drag
Values
Min
2
Max
10
Step
1
Slider label

När vi är klara med redigeringen kan vi ändra antalet kluster med hjälp av skjutreglaget i inmatningsrutan för *Kluster*-variabeln.

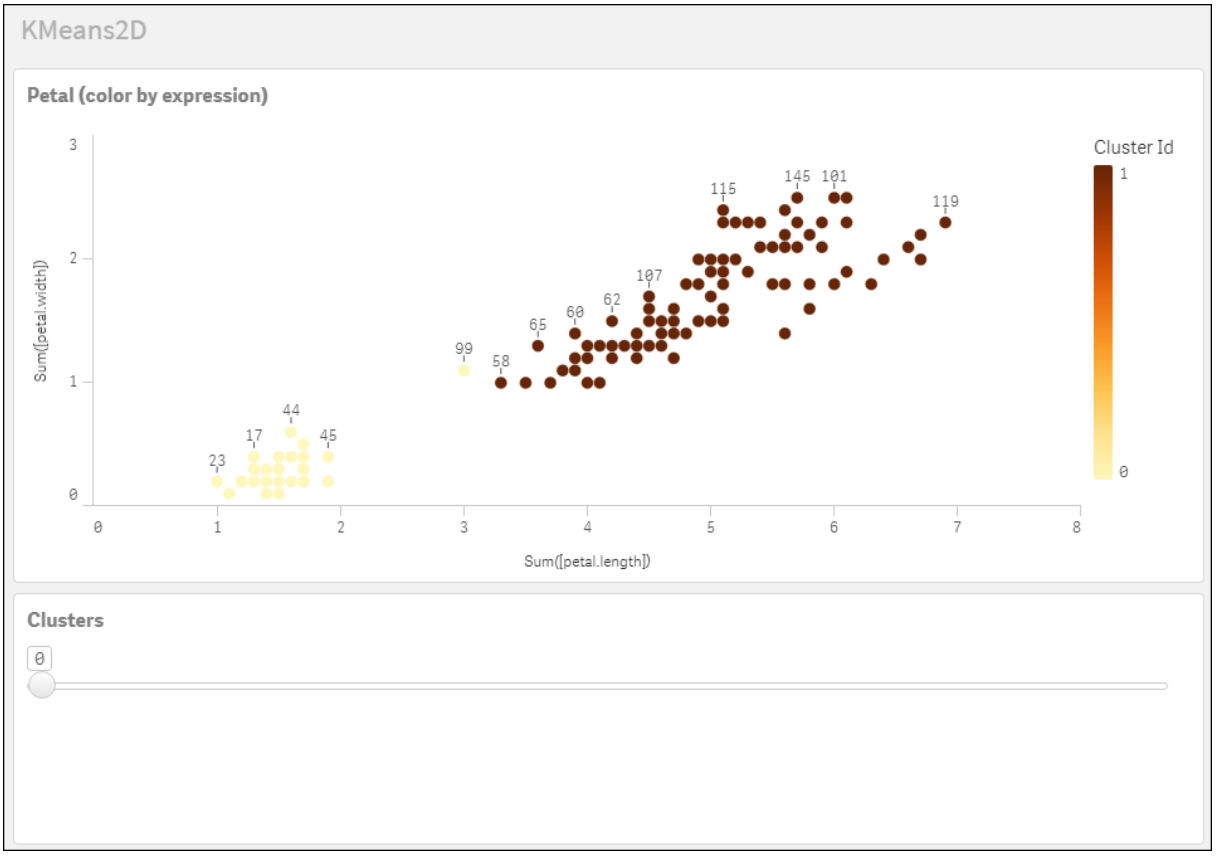
Klustren färglagda per uttryck i diagrammet Kronblad (färg per uttryck)



Automatisk klustring

KMeans-funktioner stöder automatisk klustring med en metod som kallas djupskillnad (depth difference, DeD). När användaren anger 0 som antal kluster bestäms ett optimalt antal kluster för den datauppsättningen. Observera att medan ett heltal som anges för antalet kluster (*k*) inte returneras explicit, beräknas det inom KMeans-algoritmen. Om till exempel 0 specificeras i funktionen för värdet av *KmeansPetalClusters* eller anges via en variabelindataruta, beräknas klustertilldelningar automatiskt för datauppsättningen baserat på ett optimalt antal kluster.

Metoden KMeans-djupskillnad avgör det optimala antalet kluster när (k) är inställt på 0



Iris-datauppsättning: Inline-laddning för Skriptredigeraren i Qlik Sense

```
IrisData: Load * Inline [ sepal.length, sepal.width, petal.length, petal.width, variety, id
5.1, 3.5, 1.4, 0.2, Setosa, 1 4.9, 3, 1.4, 0.2, Setosa, 2 4.7, 3.2, 1.3, 0.2, Setosa, 3 4.6,
3.1, 1.5, 0.2, Setosa, 4 5, 3.6, 1.4, 0.2, Setosa, 5 5.4, 3.9, 1.7, 0.4, Setosa, 6 4.6, 3.4,
1.4, 0.3, Setosa, 7 5, 3.4, 1.5, 0.2, Setosa, 8 4.4, 2.9, 1.4, 0.2, Setosa, 9 4.9, 3.1, 1.5,
0.1, Setosa, 10 5.4, 3.7, 1.5, 0.2, Setosa, 11 4.8, 3.4, 1.6, 0.2, Setosa, 12 4.8, 3, 1.4,
0.1, Setosa, 13 4.3, 3, 1.1, 0.1, Setosa, 14 5.8, 4, 1.2, 0.2, Setosa, 15 5.7, 4.4, 1.5, 0.4,
Setosa, 16 5.4, 3.9, 1.3, 0.4, Setosa, 17 5.1, 3.5, 1.4, 0.3, Setosa, 18 5.7, 3.8, 1.7, 0.3,
Setosa, 19 5.1, 3.8, 1.5, 0.3, Setosa, 20 5.4, 3.4, 1.7, 0.2, Setosa, 21 5.1, 3.7, 1.5, 0.4,
Setosa, 22 4.6, 3.6, 1, 0.2, Setosa, 23 5.1, 3.3, 1.7, 0.5, Setosa, 24 4.8, 3.4, 1.9, 0.2,
Setosa, 25 5, 3, 1.6, 0.2, Setosa, 26 5, 3.4, 1.6, 0.4, Setosa, 27 5.2, 3.5, 1.5, 0.2, Setosa,
28 5.2, 3.4, 1.4, 0.2, Setosa, 29 4.7, 3.2, 1.6, 0.2, Setosa, 30 4.8, 3.1, 1.6, 0.2, Setosa,
31 5.4, 3.4, 1.5, 0.4, Setosa, 32 5.2, 4.1, 1.5, 0.1, Setosa, 33 5.5, 4.2, 1.4, 0.2, Setosa,
34 4.9, 3.1, 1.5, 0.1, Setosa, 35 5, 3.2, 1.2, 0.2, Setosa, 36 5.5, 3.5, 1.3, 0.2, Setosa, 37
4.9, 3.1, 1.5, 0.1, Setosa, 38 4.4, 3, 1.3, 0.2, Setosa, 39 5.1, 3.4, 1.5, 0.2, Setosa, 40 5,
3.5, 1.3, 0.3, Setosa, 41 4.5, 2.3, 1.3, 0.3, Setosa, 42 4.4, 3.2, 1.3, 0.2, Setosa, 43 5,
3.5, 1.6, 0.6, Setosa, 44 5.1, 3.8, 1.9, 0.4, Setosa, 45 4.8, 3, 1.4, 0.3, Setosa, 46 5.1,
3.8, 1.6, 0.2, Setosa, 47 4.6, 3.2, 1.4, 0.2, Setosa, 48 5.3, 3.7, 1.5, 0.2, Setosa, 49 5,
3.3, 1.4, 0.2, Setosa, 50 7, 3.2, 4.7, 1.4, Versicolor, 51 6.4, 3.2, 4.5, 1.5, Versicolor, 52
6.9, 3.1, 4.9, 1.5, Versicolor, 53 5.5, 2.3, 4, 1.3, Versicolor, 54 6.5, 2.8, 4.6, 1.5,
Versicolor, 55 5.7, 2.8, 4.5, 1.3, Versicolor, 56 6.3, 3.3, 4.7, 1.6, Versicolor, 57 4.9, 2.4,
3.3, 1, Versicolor, 58 6.6, 2.9, 4.6, 1.3, Versicolor, 59 5.2, 2.7, 3.9, 1.4, Versicolor, 60
5, 2, 3.5, 1, Versicolor, 61 5.9, 3, 4.2, 1.5, Versicolor, 62 6, 2.2, 4, 1, Versicolor, 63
6.1, 2.9, 4.7, 1.4, Versicolor, 64 5.6, 2.9, 3.6, 1.3, Versicolor, 65 6.7, 3.1, 4.4, 1.4,
Versicolor, 66 5.6, 3, 4.5, 1.5, Versicolor, 67 5.8, 2.7, 4.1, 1, Versicolor, 68 6.2, 2.2,
4.5, 1.5, Versicolor, 69 5.6, 2.5, 3.9, 1.1, Versicolor, 70 5.9, 3.2, 4.8, 1.8, Versicolor, 71
6.1, 2.8, 4, 1.3, Versicolor, 72 6.3, 2.5, 4.9, 1.5, Versicolor, 73 6.1, 2.8, 4.7, 1.2,
```

```
Versicolor, 74 6.4, 2.9, 4.3, 1.3, Versicolor, 75 6.6, 3, 4.4, 1.4, Versicolor, 76 6.8, 2.8,
4.8, 1.4, Versicolor, 77 6.7, 3, 5, 1.7, Versicolor, 78 6, 2.9, 4.5, 1.5, Versicolor, 79 5.7,
2.6, 3.5, 1, Versicolor, 80 5.5, 2.4, 3.8, 1.1, Versicolor, 81 5.5, 2.4, 3.7, 1, Versicolor,
82 5.8, 2.7, 3.9, 1.2, Versicolor, 83 6, 2.7, 5.1, 1.6, Versicolor, 84 5.4, 3, 4.5, 1.5,
Versicolor, 85 6, 3.4, 4.5, 1.6, Versicolor, 86 6.7, 3.1, 4.7, 1.5, Versicolor, 87 6.3, 2.3,
4.4, 1.3, Versicolor, 88 5.6, 3, 4.1, 1.3, Versicolor, 89 5.5, 2.5, 4, 1.3, Versicolor, 90
5.5, 2.6, 4.4, 1.2, Versicolor, 91 6.1, 3, 4.6, 1.4, Versicolor, 92 5.8, 2.6, 4, 1.2,
Versicolor, 93 5, 2.3, 3.3, 1, Versicolor, 94 5.6, 2.7, 4.2, 1.3, Versicolor, 95 5.7, 3, 4.2,
1.2, Versicolor, 96 5.7, 2.9, 4.2, 1.3, Versicolor, 97 6.2, 2.9, 4.3, 1.3, Versicolor, 98 5.1,
2.5, 3, 1.1, Versicolor, 99 5.7, 2.8, 4.1, 1.3, Versicolor, 100 6.3, 3.3, 6, 2.5, Virginica,
101 5.8, 2.7, 5.1, 1.9, Virginica, 102 7.1, 3, 5.9, 2.1, Virginica, 103 6.3, 2.9, 5.6, 1.8,
Virginica, 104 6.5, 3, 5.8, 2.2, Virginica, 105 7.6, 3, 6.6, 2.1, Virginica, 106 4.9, 2.5,
4.5, 1.7, Virginica, 107 7.3, 2.9, 6.3, 1.8, Virginica, 108 6.7, 2.5, 5.8, 1.8, Virginica, 109
7.2, 3.6, 6.1, 2.5, Virginica, 110 6.5, 3.2, 5.1, 2, Virginica, 111 6.4, 2.7, 5.3, 1.9,
Virginica, 112 6.8, 3, 5.5, 2.1, Virginica, 113 5.7, 2.5, 5, 2, Virginica, 114 5.8, 2.8, 5.1,
2.4, Virginica, 115 6.4, 3.2, 5.3, 2.3, Virginica, 116 6.5, 3, 5.5, 1.8, Virginica, 117 7.7,
3.8, 6.7, 2.2, Virginica, 118 7.7, 2.6, 6.9, 2.3, Virginica, 119 6, 2.2, 5, 1.5, Virginica,
120 6.9, 3.2, 5.7, 2.3, Virginica, 121 5.6, 2.8, 4.9, 2, Virginica, 122 7.7, 2.8, 6.7, 2,
Virginica, 123 6.3, 2.7, 4.9, 1.8, Virginica, 124 6.7, 3.3, 5.7, 2.1, Virginica, 125 7.2, 3.2,
6, 1.8, Virginica, 126 6.2, 2.8, 4.8, 1.8, Virginica, 127 6.1, 3, 4.9, 1.8, Virginica, 128
6.4, 2.8, 5.6, 2.1, Virginica, 129 7.2, 3, 5.8, 1.6, Virginica, 130 7.4, 2.8, 6.1, 1.9,
Virginica, 131 7.9, 3.8, 6.4, 2, Virginica, 132 6.4, 2.8, 5.6, 2.2, Virginica, 133 6.3, 2.8,
5.1, 1.5, Virginica, 134 6.1, 2.6, 5.6, 1.4, Virginica, 135 7.7, 3, 6.1, 2.3, Virginica, 136
6.3, 3.4, 5.6, 2.4, Virginica, 137 6.4, 3.1, 5.5, 1.8, Virginica, 138 6, 3, 4.8, 1.8,
Virginica, 139 6.9, 3.1, 5.4, 2.1, Virginica, 140 6.7, 3.1, 5.6, 2.4, Virginica, 141 6.9, 3.1,
5.1, 2.3, Virginica, 142 5.8, 2.7, 5.1, 1.9, Virginica, 143 6.8, 3.2, 5.9, 2.3, Virginica, 144
6.7, 3.3, 5.7, 2.5, Virginica, 145 6.7, 3, 5.2, 2.3, Virginica, 146 6.3, 2.5, 5, 1.9,
Virginica, 147 6.5, 3, 5.2, 2, Virginica, 148 6.2, 3.4, 5.4, 2.3, Virginica, 149 5.9, 3, 5.1,
1.8, Virginica, 150 ];
```

KMeansND - diagramfunktion

KMeansND() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring och för varje diagramrad visas kluster-ID för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1 och coordinate_2 osv. upp till n kolumner. Dessa är alla aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters.

KMeansND returnerar ett värde per datapunkt. Det returnerade värdet är en dual och är det heltalsvärde som motsvarar klustret varje datapunkt har tilldelats till.

Syntax:

```
KMeansND(num_clusters, num_iter, coordinate_1, coordinate_2 [,coordinate_3 [,
...]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
num_clusters	Heltal som anger antalet kluster.
num_iter	Antalet upprepningar av klustring med ominitierade klustercenter.
coordinate_1	Aggregeringen som beräknar den första koordinaten, vanligtvis x-axeln (på ett spridningsdiagram som kan skapas från diagrammet). De ytterligare parametrarna beräknar den andra, tredje och fjärde koordinaten osv.

Exempel: Diagramuttryck
I det här exemplet skapar vi ett spridningsdiagram med datauppsättningen *Iris*, och använder sedan KMeans för att färglägga data efter uttryck.

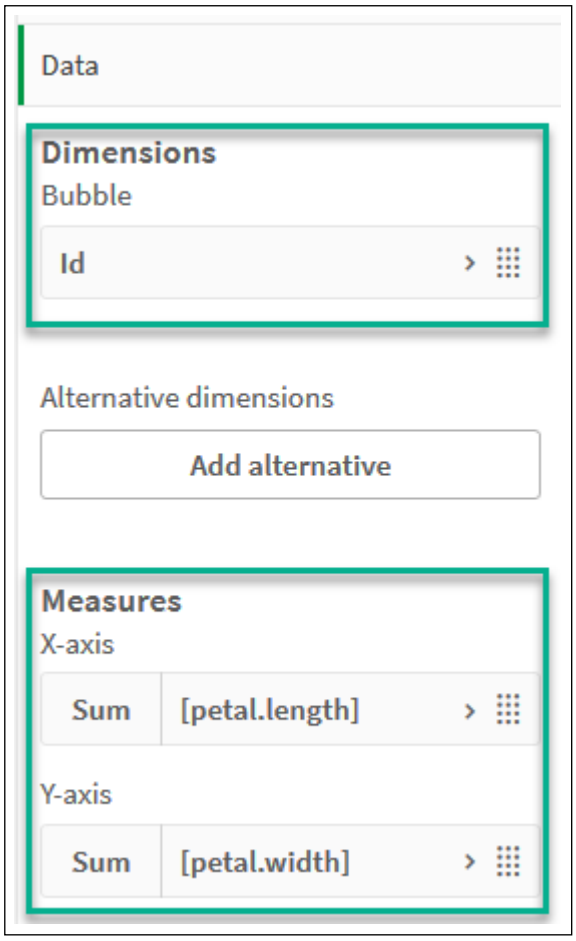
Vi skapar även en variabel för argumentet *num_clusters*, och sedan använder vi en variabelinmatningsruta för att ändra antalet kluster.

Vi skapar även en variabel för argumentet *num_iter*, och sedan använder vi en variabelinmatningsruta för att ändra antalet upprepningar.

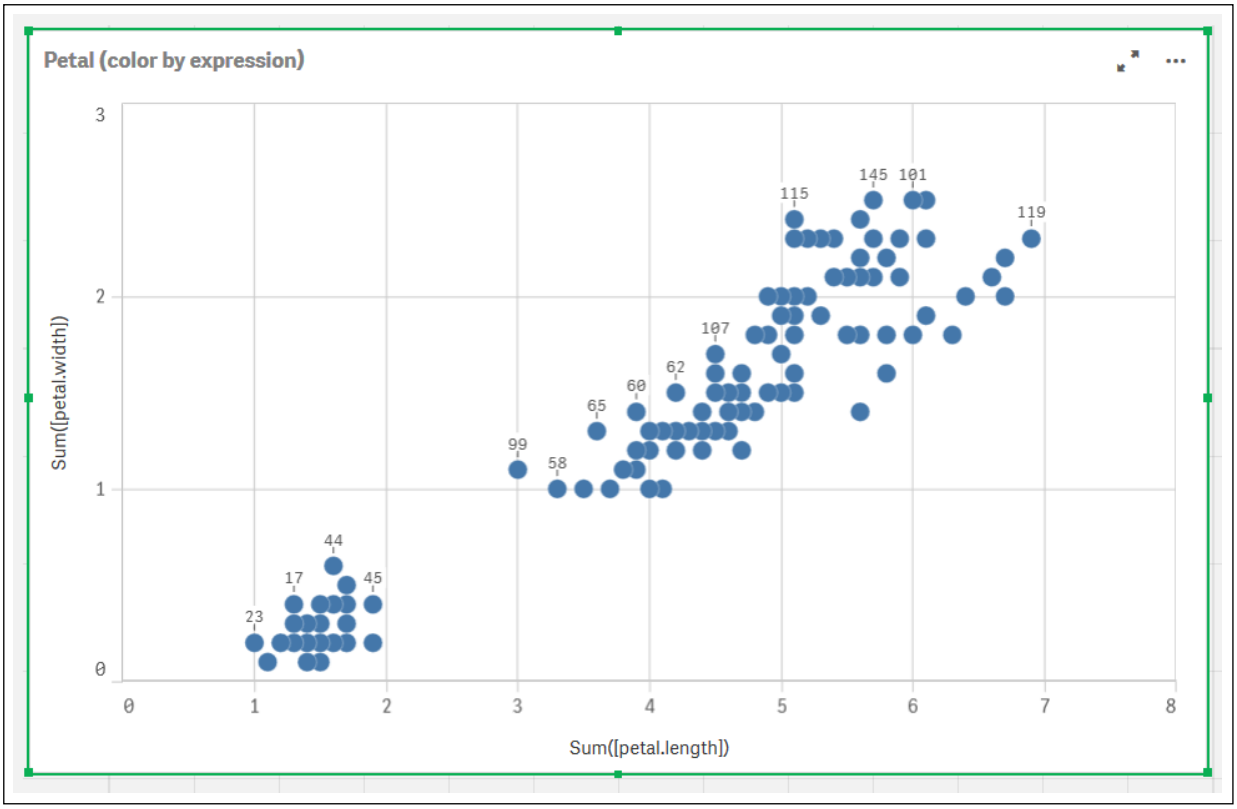
Datauppsättningen *Iris* finns offentligt tillgänglig i en mängd format. Vi tillhandahåller dessa data som en inline-tabell som laddas med Skriptredigeraren i Qlik Sense. Observera att vi lagt till en *ID*-kolumn i datatabellen i det här exemplet.

När vi laddat data i Qlik Sense, gör vi följande:

1. Dra ett **spridningsdiagram** till ett nytt ark. Döp diagrammet till *Kronblad (färg per uttryck)*.
2. Skapa en variabel för att ange antalet kluster. För variabeln **Namn** anger du *KmeansPetalClusters*. För variabeln **Definition** anger du *=2*.
3. Skapa en variabel för att ange antalet upprepningar. För variabeln **Namn** anger du *KmeansNumberIterations*. För variabeln **Definition** anger du *=1*.
4. Konfigurera **Data** för diagrammet:
    i. Under **Dimensioner** väljer du *id* för fältet **Bubbla**. Ange ett kluster-ID för Etikett.
    ii. Under **Mått** väljer du *Sum([petal.length])* för uttrycket för **X-axel**.
    iii. Under **Mått** väljer du *Sum([petal.width])* för uttrycket för **Y-axel**.
    Datainställningar för diagrammet Kronblad (färg per uttryck)

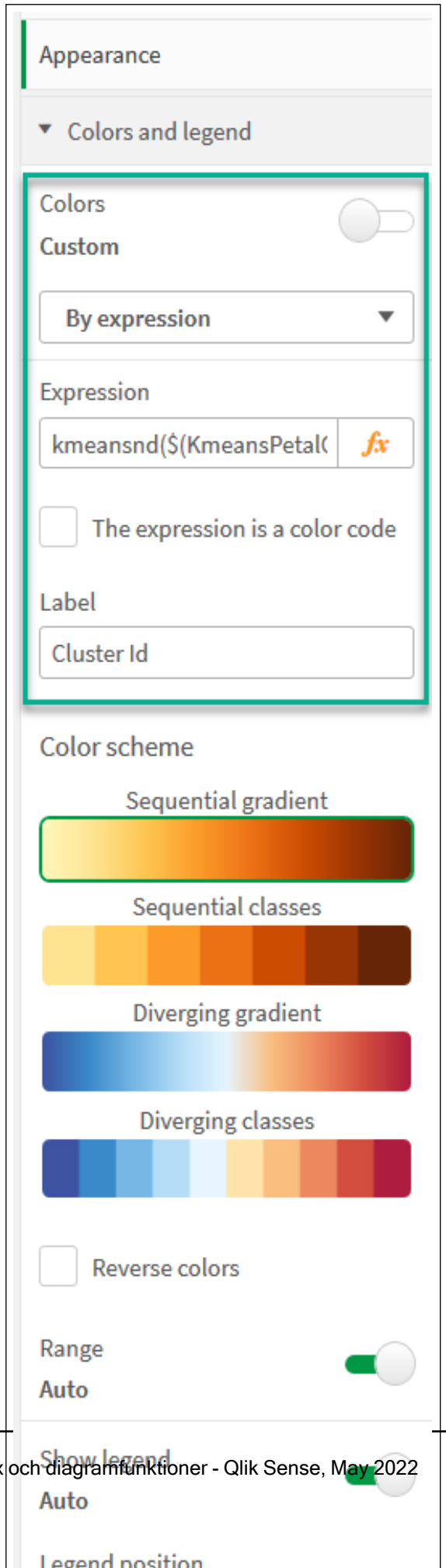


Datapunkterna ritas ut på diagrammet.
Datapunkter på diagrammet Kronblad (färg per uttryck)



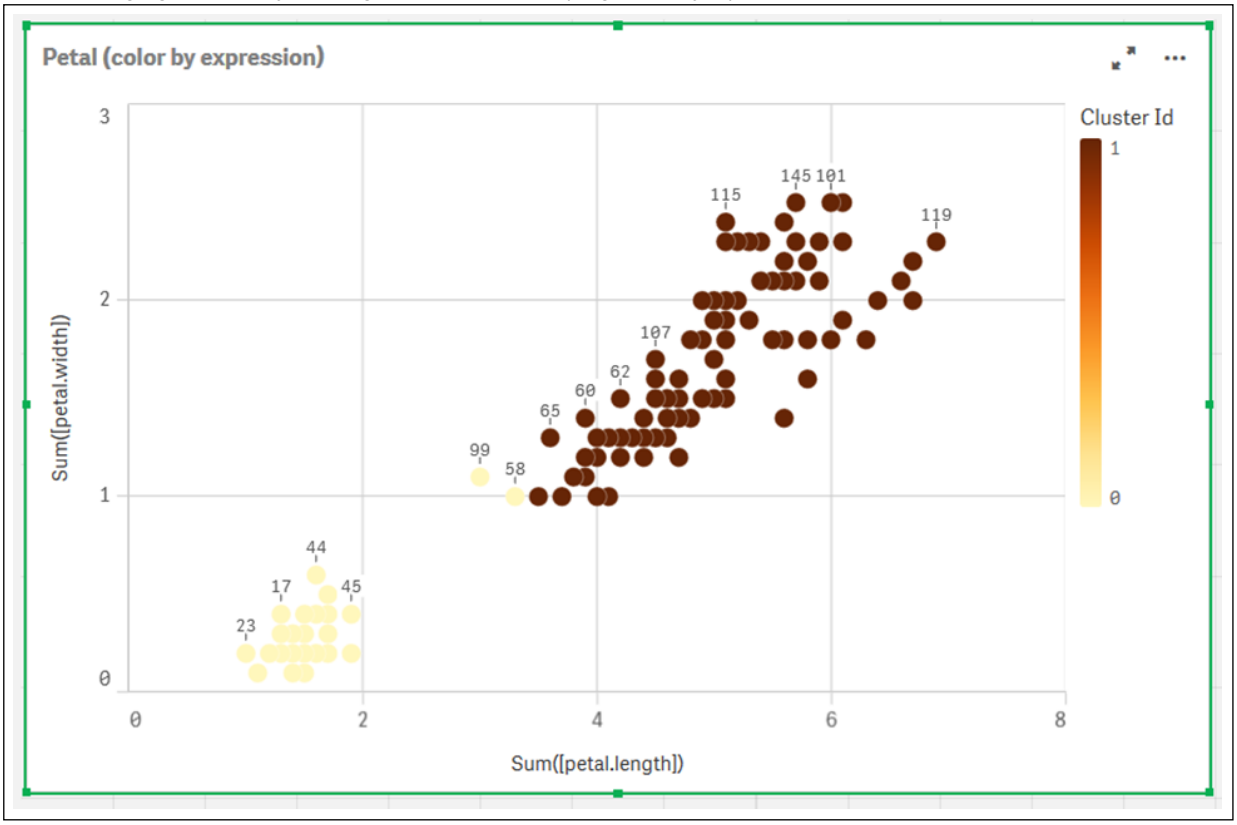
5. Konfigurera **Utseende** för diagrammet:
    i. Under **Färger och teckenförklaring** väljer du **Anpassad** för **Färger**.
    ii. Välj att färglägga diagrammet**Per uttryck**.
    iii. Ange följande för **Uttryck**: *kmeansnd($(KmeansPetalClusters),$(KmeansNumberIterations), Sum([petal.length]), Sum([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
    Observera att *KmeansPetalClusters* är variabeln vi angav som *2*. *KmeansNumberIterations* är variabeln vi angav som *1*.
    Eller ange följande: *kmeansnd(2, 2, Sum([petal.length]), Sum([petal.width]),Sum ([sepal.length]), Sum([sepal.width]))*
    iv. Avmarkera kryssrutan **Uttrycket är en färgkod**.
    v. Ange följande för **Etikett**: *Kluster-ID*

Utseendeinställningar för diagrammet Kronblad (färg per uttryck)


Appearance
Colors and legend
Colors
Custom
By expression
Expression
kmeansnd($(KmeansPetal(
The expression is a color code
Label
Cluster Id
Color scheme
Sequential gradient
Sequential classes
Diverging gradient
Diverging classes
Reverse colors
Range
Auto
Show legend
Auto
Legend position

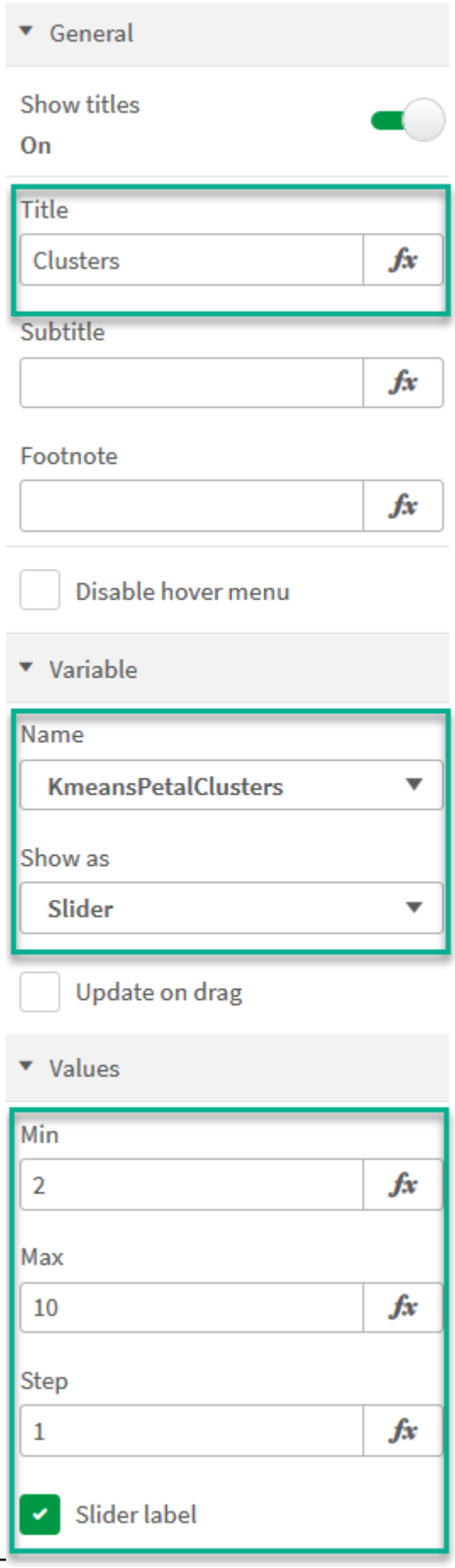
De två klustren i diagrammet färgläggs per KMeans-uttrycket.

Klustren färglagda per uttryck i diagrammet Kronblad (färg per uttryck)



6. Lägg till en ruta för **Variabelinmatning** för antalet kluster.
    i. Under **Anpassade objekt** i panelen **Resurser** väljer du **Qliks instrumentpanelspaket**. Om vi inte hade tillgång till instrumentpanelspaketet kunde vi ändå ändra antalet kluster med variabeln vi skapat, eller direkt som ett heltal i uttrycket.
    ii. Dra en ruta för **Variabelinmatning** till arket.
    iii. Under **Utseende** klickar du på **Allmänt**.
    iv. Ange följande för **Rubrik**: *Kluster*
    v. Klicka på **Variabel**.
    vi. Välj följande variabel för **Namn**: *KmeansPetalClusters*.
    vii. Välj **Skjutreglage** för **Visa som**.
    viii. Välj **Värden** och konfigurera inställningarna efter behov,

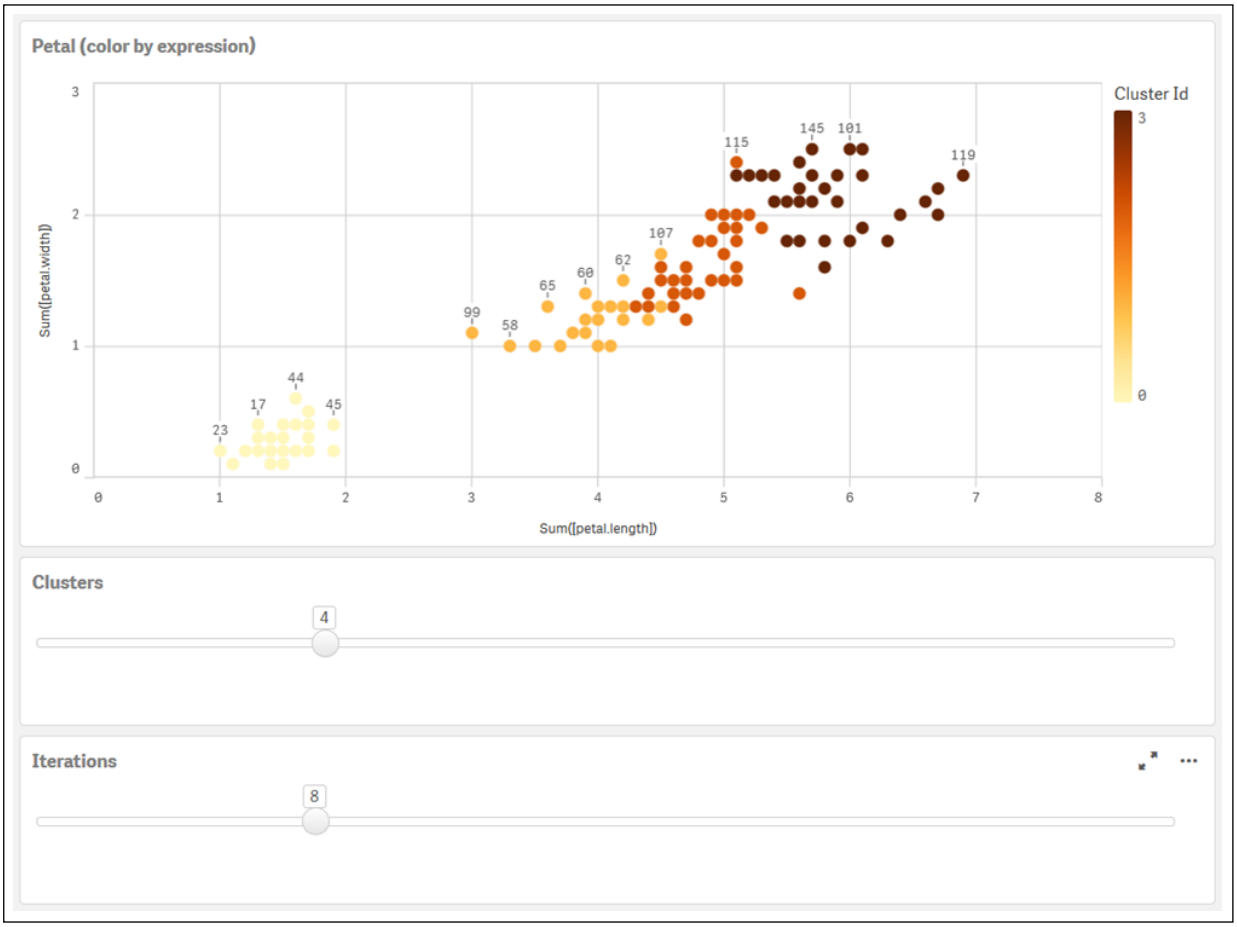
Utseende för variabelinmatningsrutan Kluster


General
Show titles
On
Title
Clusters
Subtitle
Footnote
Disable hover menu
Variable
Name
KmeansPetalClusters
Show as
Slider
Update on drag
Values
Min
2
Max
10
Step
1
Slider label

7. Lägg till en ruta för **Variabelinmatning** för antalet upprepningar.
    i. Dra en ruta för **Variabelinmatning** till arket.
    ii. Under **Utseende** väljer du **Allmänt**.
    iii. Ange följande för **Rubrik**: *Upprepningar*
    iv. Under **Utseende** väljer du **Variabel**.
    v. Välj följande variabel under **Namn**: *KmeansNumberIterations*.
    vi. Konfigurera ytterligare inställningar efter behov,

    Vi kan nu ändra antalet kluster och upprepningar med hjälp av skjutreglagen i inmatningsrutorna för variabel.

    Klustren färglagda per uttryck i diagrammet Kronblad (färg per uttryck)

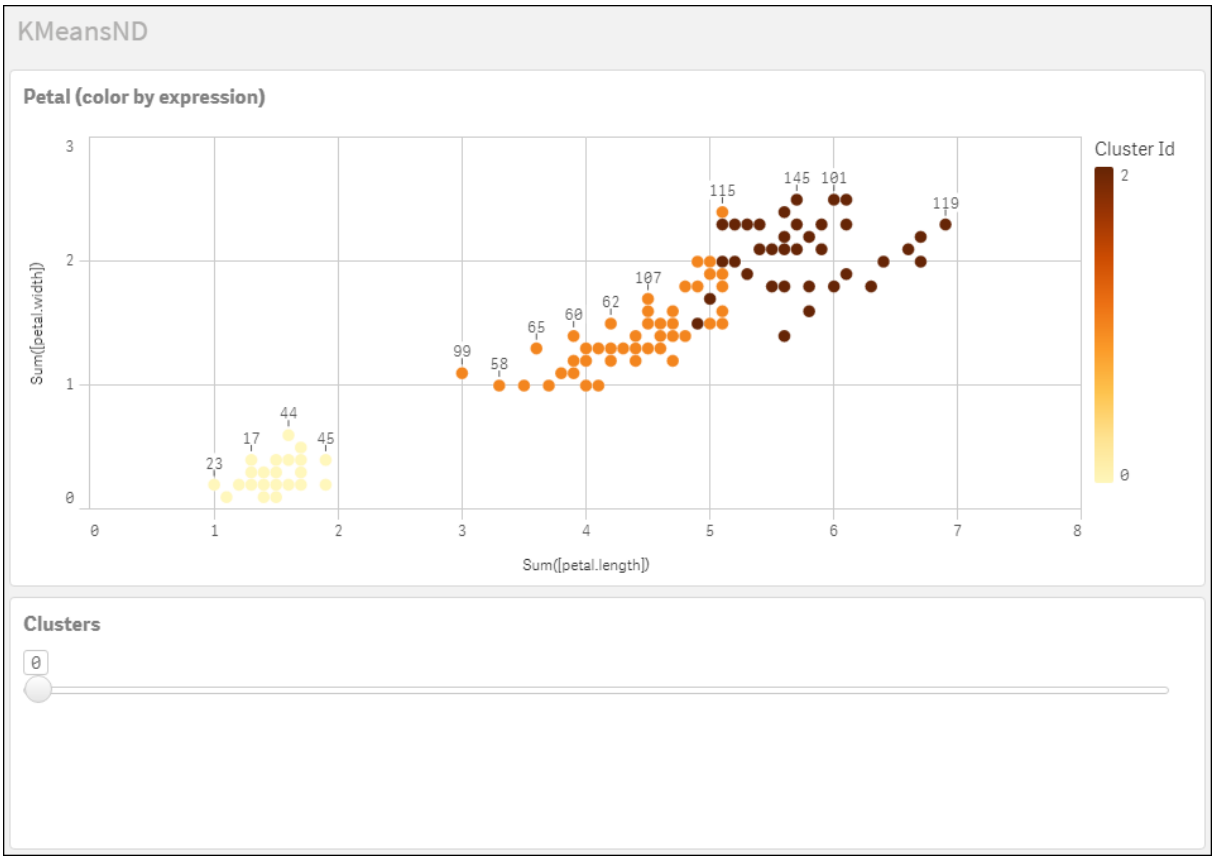


Automatisk klustring

KMeans-funktioner stöder automatisk klustring med en metod som kallas djupskillnad (depth difference, DeD). När användaren anger 0 som antal kluster bestäms ett optimalt antal kluster för den datauppsättningen. Observera att medan ett heltal som anges för antalet kluster (*k*) inte returneras explicit, beräknas det inom KMeans-algoritmen. Om till exempel 0 specificeras i funktionen för värdet av *KmeansPetalClusters* eller anges via en variabelindataruta, beräknas klustertilldelningar automatiskt för

datauppsättningen baserat på ett optimalt antal kluster. Med Iris-datauppsättningen; om 0 väljs för antalet kluster kommer algoritmen att bestämma (automatisk klustring) ett optimalt antal kluster (3) för denna datauppsättning.

Metoden KMeans-djupskillnad avgör det optimala antalet kluster när (k) är inställt på 0.



Iris-datauppsättning: Inline-laddning för Skriptredigeraren i Qlik Sense

```
IrisData: Load * Inline [ sepal.length, sepal.width, petal.length, petal.width, variety, id
5.1, 3.5, 1.4, 0.2, Setosa, 1 4.9, 3, 1.4, 0.2, Setosa, 2 4.7, 3.2, 1.3, 0.2, Setosa, 3 4.6,
3.1, 1.5, 0.2, Setosa, 4 5, 3.6, 1.4, 0.2, Setosa, 5 5.4, 3.9, 1.7, 0.4, Setosa, 6 4.6, 3.4,
1.4, 0.3, Setosa, 7 5, 3.4, 1.5, 0.2, Setosa, 8 4.4, 2.9, 1.4, 0.2, Setosa, 9 4.9, 3.1, 1.5,
0.1, Setosa, 10 5.4, 3.7, 1.5, 0.2, Setosa, 11 4.8, 3.4, 1.6, 0.2, Setosa, 12 4.8, 3, 1.4,
0.1, Setosa, 13 4.3, 3, 1.1, 0.1, Setosa, 14 5.8, 4, 1.2, 0.2, Setosa, 15 5.7, 4.4, 1.5, 0.4,
Setosa, 16 5.4, 3.9, 1.3, 0.4, Setosa, 17 5.1, 3.5, 1.4, 0.3, Setosa, 18 5.7, 3.8, 1.7, 0.3,
Setosa, 19 5.1, 3.8, 1.5, 0.3, Setosa, 20 5.4, 3.4, 1.7, 0.2, Setosa, 21 5.1, 3.7, 1.5, 0.4,
Setosa, 22 4.6, 3.6, 1, 0.2, Setosa, 23 5.1, 3.3, 1.7, 0.5, Setosa, 24 4.8, 3.4, 1.9, 0.2,
Setosa, 25 5, 3, 1.6, 0.2, Setosa, 26 5, 3.4, 1.6, 0.4, Setosa, 27 5.2, 3.5, 1.5, 0.2, Setosa,
28 5.2, 3.4, 1.4, 0.2, Setosa, 29 4.7, 3.2, 1.6, 0.2, Setosa, 30 4.8, 3.1, 1.6, 0.2, Setosa,
31 5.4, 3.4, 1.5, 0.4, Setosa, 32 5.2, 4.1, 1.5, 0.1, Setosa, 33 5.5, 4.2, 1.4, 0.2, Setosa,
34 4.9, 3.1, 1.5, 0.1, Setosa, 35 5, 3.2, 1.2, 0.2, Setosa, 36 5.5, 3.5, 1.3, 0.2, Setosa, 37
4.9, 3.1, 1.5, 0.1, Setosa, 38 4.4, 3, 1.3, 0.2, Setosa, 39 5.1, 3.4, 1.5, 0.2, Setosa, 40 5,
3.5, 1.3, 0.3, Setosa, 41 4.5, 2.3, 1.3, 0.3, Setosa, 42 4.4, 3.2, 1.3, 0.2, Setosa, 43 5,
3.5, 1.6, 0.6, Setosa, 44 5.1, 3.8, 1.9, 0.4, Setosa, 45 4.8, 3, 1.4, 0.3, Setosa, 46 5.1,
3.8, 1.6, 0.2, Setosa, 47 4.6, 3.2, 1.4, 0.2, Setosa, 48 5.3, 3.7, 1.5, 0.2, Setosa, 49 5,
3.3, 1.4, 0.2, Setosa, 50 7, 3.2, 4.7, 1.4, Versicolor, 51 6.4, 3.2, 4.5, 1.5, Versicolor, 52
6.9, 3.1, 4.9, 1.5, Versicolor, 53 5.5, 2.3, 4, 1.3, Versicolor, 54 6.5, 2.8, 4.6, 1.5,
```

```
Versicolor, 55 5.7, 2.8, 4.5, 1.3, Versicolor, 56 6.3, 3.3, 4.7, 1.6, Versicolor, 57 4.9, 2.4,
3.3, 1, Versicolor, 58 6.6, 2.9, 4.6, 1.3, Versicolor, 59 5.2, 2.7, 3.9, 1.4, Versicolor, 60
5, 2, 3.5, 1, Versicolor, 61 5.9, 3, 4.2, 1.5, Versicolor, 62 6, 2.2, 4, 1, Versicolor, 63
6.1, 2.9, 4.7, 1.4, Versicolor, 64 5.6, 2.9, 3.6, 1.3, Versicolor, 65 6.7, 3.1, 4.4, 1.4,
Versicolor, 66 5.6, 3, 4.5, 1.5, Versicolor, 67 5.8, 2.7, 4.1, 1, Versicolor, 68 6.2, 2.2,
4.5, 1.5, Versicolor, 69 5.6, 2.5, 3.9, 1.1, Versicolor, 70 5.9, 3.2, 4.8, 1.8, Versicolor, 71
6.1, 2.8, 4, 1.3, Versicolor, 72 6.3, 2.5, 4.9, 1.5, Versicolor, 73 6.1, 2.8, 4.7, 1.2,
Versicolor, 74 6.4, 2.9, 4.3, 1.3, Versicolor, 75 6.6, 3, 4.4, 1.4, Versicolor, 76 6.8, 2.8,
4.8, 1.4, Versicolor, 77 6.7, 3, 5, 1.7, Versicolor, 78 6, 2.9, 4.5, 1.5, Versicolor, 79 5.7,
2.6, 3.5, 1, Versicolor, 80 5.5, 2.4, 3.8, 1.1, Versicolor, 81 5.5, 2.4, 3.7, 1, Versicolor,
82 5.8, 2.7, 3.9, 1.2, Versicolor, 83 6, 2.7, 5.1, 1.6, Versicolor, 84 5.4, 3, 4.5, 1.5,
Versicolor, 85 6, 3.4, 4.5, 1.6, Versicolor, 86 6.7, 3.1, 4.7, 1.5, Versicolor, 87 6.3, 2.3,
4.4, 1.3, Versicolor, 88 5.6, 3, 4.1, 1.3, Versicolor, 89 5.5, 2.5, 4, 1.3, Versicolor, 90
5.5, 2.6, 4.4, 1.2, Versicolor, 91 6.1, 3, 4.6, 1.4, Versicolor, 92 5.8, 2.6, 4, 1.2,
Versicolor, 93 5, 2.3, 3.3, 1, Versicolor, 94 5.6, 2.7, 4.2, 1.3, Versicolor, 95 5.7, 3, 4.2,
1.2, Versicolor, 96 5.7, 2.9, 4.2, 1.3, Versicolor, 97 6.2, 2.9, 4.3, 1.3, Versicolor, 98 5.1,
2.5, 3, 1.1, Versicolor, 99 5.7, 2.8, 4.1, 1.3, Versicolor, 100 6.3, 3.3, 6, 2.5, Virginica,
101 5.8, 2.7, 5.1, 1.9, Virginica, 102 7.1, 3, 5.9, 2.1, Virginica, 103 6.3, 2.9, 5.6, 1.8,
Virginica, 104 6.5, 3, 5.8, 2.2, Virginica, 105 7.6, 3, 6.6, 2.1, Virginica, 106 4.9, 2.5,
4.5, 1.7, Virginica, 107 7.3, 2.9, 6.3, 1.8, Virginica, 108 6.7, 2.5, 5.8, 1.8, Virginica, 109
7.2, 3.6, 6.1, 2.5, Virginica, 110 6.5, 3.2, 5.1, 2, Virginica, 111 6.4, 2.7, 5.3, 1.9,
Virginica, 112 6.8, 3, 5.5, 2.1, Virginica, 113 5.7, 2.5, 5, 2, Virginica, 114 5.8, 2.8, 5.1,
2.4, Virginica, 115 6.4, 3.2, 5.3, 2.3, Virginica, 116 6.5, 3, 5.5, 1.8, Virginica, 117 7.7,
3.8, 6.7, 2.2, Virginica, 118 7.7, 2.6, 6.9, 2.3, Virginica, 119 6, 2.2, 5, 1.5, Virginica,
120 6.9, 3.2, 5.7, 2.3, Virginica, 121 5.6, 2.8, 4.9, 2, Virginica, 122 7.7, 2.8, 6.7, 2,
Virginica, 123 6.3, 2.7, 4.9, 1.8, Virginica, 124 6.7, 3.3, 5.7, 2.1, Virginica, 125 7.2, 3.2,
6, 1.8, Virginica, 126 6.2, 2.8, 4.8, 1.8, Virginica, 127 6.1, 3, 4.9, 1.8, Virginica, 128
6.4, 2.8, 5.6, 2.1, Virginica, 129 7.2, 3, 5.8, 1.6, Virginica, 130 7.4, 2.8, 6.1, 1.9,
Virginica, 131 7.9, 3.8, 6.4, 2, Virginica, 132 6.4, 2.8, 5.6, 2.2, Virginica, 133 6.3, 2.8,
5.1, 1.5, Virginica, 134 6.1, 2.6, 5.6, 1.4, Virginica, 135 7.7, 3, 6.1, 2.3, Virginica, 136
6.3, 3.4, 5.6, 2.4, Virginica, 137 6.4, 3.1, 5.5, 1.8, Virginica, 138 6, 3, 4.8, 1.8,
Virginica, 139 6.9, 3.1, 5.4, 2.1, Virginica, 140 6.7, 3.1, 5.6, 2.4, Virginica, 141 6.9, 3.1,
5.1, 2.3, Virginica, 142 5.8, 2.7, 5.1, 1.9, Virginica, 143 6.8, 3.2, 5.9, 2.3, Virginica, 144
6.7, 3.3, 5.7, 2.5, Virginica, 145 6.7, 3, 5.2, 2.3, Virginica, 146 6.3, 2.5, 5, 1.9,
Virginica, 147 6.5, 3, 5.2, 2, Virginica, 148 6.2, 3.4, 5.4, 2.3, Virginica, 149 5.9, 3, 5.1,
1.8, Virginica, 150 ];
```

KMeansCentroid2D - diagramfunktion

KMeansCentroid2D() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring, och för varje diagramrad visas önskad koordinat för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1 respektive coordinate_2. Dessa är båda aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters. Data normaliseras med normparametern (valfritt).

KMeansCentroid2D returnerar ett värde per datapunkt. Det returnerade värdet är en dual och är en av koordinaterna för positionen som motsvarar klustercentret datapunkten har tilldelats till.

Syntax:

```
KMeansCentroid2D(num_clusters, coordinate_no, coordinate_1, coordinate_2 [,
norm])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
num_clusters	Heltal som anger antalet kluster.
coordinate_no	Önskat koordinatantal för centroiderna (motsvarande, till exempel, x-, y- eller z-axeln).
coordinate_1	Aggregeringen som beräknar den första koordinaten, vanligtvis x-axeln på spridningsdiagrammet som kan skapas från diagrammet. Den ytterligare parametern, coordinate_2, beräknar den andra koordinaten.
norm	Den valfria normaliseringsmetoden tillämpas på datauppsättningen före k-medelvärdesklustring. Möjliga värden: 0 eller ”none” för ingen normalisering 1 eller ”zscore” för z-poängsnormalisering 2 eller ”minmax” för min-max-normalisering Om ingen parameter anges eller om den angivna parametern är felaktig används ingen normalisering. Z-poäng normaliserar data baserat på funktionens median och standardavvikelse. Z-poäng säkerställer inte att varje funktion har samma skala men det fungerar bättre än min-max för behandling av outliers. Min-max-normalisering säkerställer att funktionerna har samma skala genom att ta de minsta och största värdena för varje och räkna om varje datapunkt.

Automatisk klustring

KMeans-funktioner stöder automatisk klustring med en metod som kallas djupskillnad (depth difference, DeD). När användaren anger 0 som antal kluster bestäms ett optimalt antal kluster för den datauppsättningen. Observera att medan ett heltal som anges för antalet kluster (*k*) inte returneras explicit, beräknas det inom KMeans-algoritmen. Om till exempel 0 specificeras i funktionen för värdet av *KmeansPetalClusters* eller anges via en variabelindataruta, beräknas klustertilldelningar automatiskt för datauppsättningen baserat på ett optimalt antal kluster.

KMeansCentroidND - diagramfunktion

KMeansCentroidND() utvärderar raderna i diagrammet genom att tillämpa k-medelvärdesklustring och för varje diagramrad visas önskad koordinat för klustret som den datapunkten tilldelats till. Kolumnerna som används av klusteralgoritmen avgörs av parametrarna coordinate_1, coordinate_2 osv. upp till n kolumner.

Dessa är alla aggregeringar. Antalet kluster som skapas avgörs av parametern num_clusters.

KMeansCentroidND returnerar ett värde per rad. Det returnerade värdet är en dual och är en av koordinaterna för positionen som motsvarar klustercentret datapunkten har tilldelats till.

Syntax:

```
KMeansCentroidND((num_clusters, num_iter, coordinate_no, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

Returnerad datatyp: dual

Argument:

Argument

Argument	Beskrivning
num_clusters	Heltal som anger antalet kluster.
num_iter	Antalet upprepningar av klustring med ominitierade klustercenter.
coordinate_no	Önskat koordinatantal för centroiderna (motsvarande, till exempel, x-, y- eller z-axeln).
coordinate_1	Aggregeringen som beräknar den första koordinaten, vanligtvis x-axeln (på ett spridningsdiagram som kan skapas från diagrammet). De ytterligare parametrarna beräknar den andra, tredje och fjärde koordinaten osv.

Automatisk klustring

KMeans-funktioner stöder automatisk klustring med en metod som kallas djupskillnad (depth difference, DeD). När användaren anger 0 som antal kluster bestäms ett optimalt antal kluster för den datauppsättningen. Observera att medan ett heltal som anges för antalet kluster (*k*) inte returneras explicit, beräknas det inom KMeans-algoritmen. Om till exempel 0 specificeras i funktionen för värdet av *KmeansPetalClusters* eller anges via en variabelindataruta, beräknas klustertilldelningar automatiskt för datauppsättningen baserat på ett optimalt antal kluster.

5.23 Statistiska fördelningsfunktioner

Funktionen för statistisk fördelning DIST mäter sannolikheten för fördelningsfunktionen vid en punkt i fördelningen som ges av ett värde som tillhandahålls. Funktionerna INV beräknar värdet, i enlighet med sannolikheten för fördelningen. De statistiska aggregeringsfunktionerna däremot beräknar de aggregerade värdena av en serie av statistiska testvärden för olika test av statistiska hypoteser.

Samtliga statistiska fördelningsfunktioner som beskrivs nedan har implementerats i Qlik Sense med Cephes-funktionsbiblioteket. För referenser och mer information om de algoritmer som används, exakthet och så vidare, se: Cephes library. Cephes-funktionsbiblioteket används med tillstånd.

Alla funktioner kan användas både i dataladdningsskriptet och diagramuttryck.

Översikt av statistiska fördelningsfunktioner

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

CHIDIST

CHIDIST() returnerar den ensidiga sannolikheten hos chi^2-fördelningen. chi^2-fördelningen associeras med ett ch^{i2}-test.

```
CHIDIST (value, degrees_freedom)
```

CHIINV

CHIINV() returnerar motsatsen till den ensidiga sannolikheten hos chi^2-fördelningen.

```
CHIINV (prob, degrees_freedom)
```

NORMDIST

NORMDIST() returnerar den kumulativa normalfördelningen hos det angivna medelvärdet och standardavvikelsen. Om mean = 0 och standard_dev = 1, returnerar funktionen standardnormalfördelningen.

```
NORMDIST (value, mean, standard_dev)
```

NORMINV

NORMINV() returnerar inversen av den normala kumulativa fördelningen för det angivna medelvärdet och standardavvikelsen.

```
NORMINV (prob, mean, standard_dev)
```

TDIST

TDIST returnerar sannolikheten hos studentens t-fördelning där ett numeriskt värde är ett beräknat värde av t vars sannolikhet ska kalkyleras.

```
TDIST (value, degrees_freedom, tails)
```

TINV

TINV() returnerar t-värdet hos studentens t-fördelning som en funktion av sannolikhet och frihetsgraden.

```
TINV (prob, degrees_freedom)
```

FDIST

FDIST() returnerar F-sannolikhetsfördelningen.

```
FDIST (value, degrees_freedom1, degrees_freedom2)
```

FINV

FINV() returnerar inversionen av F-sannolikhetsfördelningen.

```
FINV (prob, degrees_freedom1, degrees_freedom2)
```

Se även:

- *Statistiska aggregeringsfunktioner (page 259)*

CHIDIST

CHIDIST() returnerar den ensidiga sannolikheten hos chi^2-fördelningen. chi^2-fördelningen associeras med ett ch^{i2}-test.

Syntax:

```
CHIDIST(value, degrees_freedom)
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
value	Det värde där fördelningen ska utvärderas. Värdet får inte vara negativt.
degrees_freedom	Ett positivt heltal som anger frihetsgrad.

Denna funktion hänger samman med funktionen **CHIINV** på följande sätt:
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Exempel och resultat:

Exempel	Resultat
CHIDIST(8, 15)	Returnerar 0,9238

CHIINV

CHIINV() returnerar motsatsen till den ensidiga sannolikheten hos chi^2-fördelningen.

Syntax:

```
CHIINV(prob, degrees_freedom)
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
prob	En sannolikhet som har samband med chi^2-fördelningen. Måste vara ett tal mellan 0 och 1.
degrees_ freedom	Ett heltal som anger frihetsgrad.

Denna funktion hänger samman med funktionen **CHIDIST** på följande sätt:
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Exempel och resultat:

Exempel	Resultat
CHIINV(0.9237827, 15)	Returnerar 8,0000

FDIST

FDIST() returnerar F-sannolikhetsfördelningen.

Syntax:

```
FDIST(value, degrees_freedom1, degrees_freedom2)
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
value	Det värde där fördelningen ska utvärderas. **Value** får inte vara negativt.
degrees_freedom1	Ett positivt heltal som anger täljarens frihetsgrad.
degrees_freedom2	Ett positivt heltal som anger nämnarens frihetsgrad.

Denna funktion hänger samman med funktionen **FINV** på följande sätt:
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Exempel och resultat:

Exempel	Resultat
FDIST(15, 8, 6)	Returnerar 0,0019

FINV

FINV() returnerar inversionen av F-sannolikhetsfördelningen.

Syntax:

```
FINV(prob, degrees_freedom1, degrees_freedom2)
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
prob	En sannolikhet som har samband med F-fördelningen och måste vara ett tal mellan 0 och 1.
degrees_ freedom	Ett heltal som anger frihetsgrad.

Denna funktion hänger samman med funktionen **FDIST** på följande sätt:
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Exempel och resultat:

Exempel	Resultat
FINV(0.0019369, 8, 6)	Returnerar 15,0000

NORMDIST

NORMDIST() returnerar den kumulativa normalfördelningen hos det angivna medelvärdet och standardavvikelsen. Om mean = 0 och standard_dev = 1, returnerar funktionen standardnormalfördelningen.

Syntax:

```
NORMDIST(value, [mean], [standard_dev], [cumulative])
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
value	Det värde där fördelningen ska utvärderas.
mean	Valfritt värde som anger fördelningens aritmetiska medelvärde. Om du inte anger det här argumentet är standardvärdet 0.
standard_dev	Valfritt positivt värde som anger fördelningens standardavvikelse. Om du inte anger det här argumentet är standardvärdet 1.
cumulative	Du kan alternativt välja att använda en standardnormalfördelning eller en kumulativ fördelning. 0 = standardnormalfördelning 1 = kumulativ fördelning (standard)

Denna funktion hänger samman med funktionen **NORMINV** på följande sätt:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Exempel och resultat:

Exempel	Resultat
NORMDIST(0.5, 0, 1)	Returnerar 0,6915

NORMINV

NORMINV() returnerar inversen av den normala kumulativa fördelningen för det angivna medelvärdet och standardavvikelsen.

Syntax:

```
NORMINV(prob, mean, standard_dev)
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
prob	En sannolikhet som har samband med normalfördelningen. Måste vara ett tal mellan 0 och 1.
mean	Ett värde som anger fördelningens aritmetiska medelvärde.
standard_dev	Ett positivt värde som anger fördelningens standardavvikelse.

Denna funktion hänger samman med funktionen **NORMDIST** på följande sätt:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Exempel och resultat:

Exempel	Resultat
NORMINV(0.6914625, 0, 1)	Returnerar 0,5000

TDIST

TDIST returnerar sannolikheten hos studentens t-fördelning där ett numeriskt värde är ett beräknat värde av t vars sannolikhet ska kalkyleras.

Syntax:

```
TDIST(value, degrees_freedom, tails)
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
value	Det värde där fördelningen ska utvärderas – det får inte vara negativt.
degrees_freedom	Ett positivt heltal som anger frihetsgrad.
tails	Måste vara antingen 1 (ensidig fördelning) eller 2 (dubbelsidig fördelning).

Denna funktion hänger samman med funktionen **TINV** på följande sätt:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value`

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Exempel och resultat:

Exempel	Resultat
TDIST(1, 30, 2)	Returnerar 0,3253

TINV

TINV() returnerar t-värdet hos studentens t-fördelning som en funktion av sannolikhet och frihetsgraden.

Syntax:

```
TINV(prob, degrees_freedom)
```

Returnerad datatyp: tal

Argument:

Argument

Argument	Beskrivning
prob	En dubbelsidig sannolikhet som har samband med t-fördelningen. Måste vara ett tal mellan 0 och 1.
degrees_ freedom	Ett heltal som anger frihetsgrad.

Begränsningar:

Alla argument måste vara numeriska, annars returneras NULL.

Denna funktion hänger samman med funktionen **TDIST** på följande sätt:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value.`

Exempel och resultat:

Exempel	Resultat
TINV(0.3253086, 30)	Returnerar 1,0000

5.24 Strängfunktioner

Den här delen beskriver funktioner för att hantera och modifiera strängar.

Alla funktioner kan användas både i dataladdningsskriptet och i diagramuttryck, utom **Evaluate** som endast kan användas i dataladdningsskriptet.

Strängfunktioner - översikt

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

Capitalize

Capitalize() returnerar strängen med alla ord med inledande versal.

```
Capitalize (text)
```

Chr

Chr() returnerar Unicode-tecknet som motsvarar indatavärdets heltal.

```
Chr (int)
```

Evaluate

Evaluate() undersöker om indatatexten kan utvärderas som ett giltigt Qlik Sense-uttryck och returnerar i så fall värdet för uttrycket som en sträng. Om indatasträngen inte är ett giltigt uttryck returneras NULL.

```
Evaluate (expression_text)
```

FindOneOf

FindOneOf() söker en sträng för att hitta positionen för eventuella tecken som ingår i en uppsättning angivna tecken. Enbart positionen för det första av dessa tecken returneras, om inte ett tredje argument (med ett värde större än 1) tillhandahålls. Om ingen motsvarighet påträffas, returneras **0**.

```
FindOneOf (text, char_set[, count])
```

Hash128

Hash128() returnerar en 128-bitars hashning av de kombinerade indatauttrycksvärdena. Resulterar i en 22-teckensträng.

```
Hash128 (expr{, expression})
```

Hash160

Hash160() returnerar en 160-bitars hashning av de kombinerade indatauttrycksvärdena. Resulterar i en 27-teckensträng.

```
Hash160 (expr{, expression})
```

Hash256

Hash256() returnerar en 256-bitars hashning av de kombinerade indatauttrycksvärdena. Resulterar i en 43-teckensträng.

```
Hash256 (expr{, expression})
```

Index

Index() söker en sträng för att hitta startpositionen för den n:e förekomsten av en angiven delsträng. Ett valbart tredje argument ger värdet för n, vilket annars är 1. Ett negativt värde söker från slutet av strängen. Strängens positioner är numrerade från **1** och uppåt.

```
Index  (text, substring[, count])
```

KeepChar

KeepChar() returnerar en sträng som består av första strängen, ”text”, minus alla eventuella tecken som INTE ingår i den andra strängen, ”keep_chars”.

```
KeepChar (text, keep_chars)
```

Left

Left() returnerar en sträng som består av de första (längst till vänster) tecknen i indatasträngen, där antalet tecken fastställs av det andra argumentet.

```
Left (text, count)
```

Len

Len() returnerar längden för indatasträngen.

```
Len (text)
```

LevenshteinDist

LevenshteinDist() returnerar Levenshtein-avståndet mellan två strängar. Det definieras som det minsta antalet enkelteckensredigeringar (infogningar, borttagningar eller ersättningar) som krävs för att förvandla en sträng till en annan. Funktionen är användbar vid ungefärliga strängjämförelser.

```
LevenshteinDist (text1, text2)
```

Lower

Lower() konverterar alla tecken i indatasträngen till gemener.

```
Lower (text)
```

LTrim

LTrim() returnerar strängen rensad på inledande blankstegstecken.

```
LTrim (text)
```

Mid

Mid() returnerar den del av indatasträngen som börjar vid positionen för det tecken som definieras av det andra argumentet, ”start”, och returnerar det antal tecken som definieras av det tredje argumentet ”count”. Om ”count” utelämnas returneras resten av indatasträngen. Det första tecknet i indatasträngen numreras som 1.

```
Mid (text, start[, count])
```

Ord

Ord() returnerar talnumret för Unicode-koden för det första tecknet i indatasträngen.

```
Ord (text)
```

PurgeChar

PurgeChar() returnerar en sträng som består av de tecken som finns i indatasträngen (”text”), utom alla eventuella tecken som visas i det andra argumentet (”remove_chars”).

```
PurgeChar (text, remove_chars)
```

Repeat

Repeat() bildar en sträng som består av indatasträngen som upprepats det antal gånger som definieras av det andra argumentet.

```
Repeat (text[, repeat_count])
```

Replace

Replace() returnerar en sträng efter att ha ersatt alla förekomster av en given delsträng inom indatasträngen med en annan delsträng. Funktionen är inte rekursiv och fungerar från vänster till höger.

```
Replace (text, from_str, to_str)
```

Right

Right() returnerar en sträng som består av de sista (längst till höger) tecknen av indatasträngen, där antalet tecken fastställs av det andra argumentet.

```
Right (text, count)
```

RTrim

RTrim() returnerar strängen rensad på avslutande blankategstecken.

```
RTrim (text)
```

SubField

Subfield() används för att extrahera delsträngkomponenter från ett överordnat strängfält där de ursprungliga postfälten består av två eller fler delar, som separeras av en avgränsare.

```
SubField (text, delimiter[, field_no ])
```

SubStringCount

SubstringCount() returnerar antalet förekomster av angiven delsträng i indatasträngtexten. Om det inte blir någon träff returneras 0.

```
SubStringCount (text, substring)
```

TextBetween

TextBetween() returnerar texten i indatasträngen som finns mellan de tecken som definierats som avgränsare.

```
TextBetween (text, delimiter1, delimiter2[, n])
```

Trim

Trim() returnerar strängen rensad på inledande och avslutande blankstegstecken.

```
Trim (text)
```

Upper

Upper() konverterar alla tecken i indatasträngen till versaler för alla texttecken i uttrycket. Tal och symboler ignoreras.

```
Upper (text)
```

Capitalize

Capitalize() returnerar strängen med alla ord med inledande versal.

Syntax:

```
Capitalize(text)
```

Returnerad datatyp: sträng

Exempel: Diagramuttryck

Exempel	Resultat
`Capitalize ( 'star trek' )`	Returnerar 'Star Trek'
`Capitalize ( 'AA bb cC Dd')`	Returnerar 'Aa Bb Cc Dd'

Exempel: Laddningsskript

```
Load  String, Capitalize(String) Inline [String rHode iSland washingTon d.C. new york];
```

Resultat

Sträng	Capitalize(String)
rHode iSland	Rhode Island
washingTon d.C.	Washington D.C.
new york	New York

Chr

Chr() returnerar Unicode-tecknet som motsvarar indatavärdets heltal.

Syntax:

```
Chr(int)
```

Returnerad datatyp: sträng

Exempel och resultat:

Exempel	Resultat
Chr(65)	Returnerar strängen 'A'

Exempel	Resultat
Chr(163)	Returnerar strängen '£'
Chr(35)	Returnerar strängen '#'

Evaluate

Evaluate() undersöker om indatatexten kan utvärderas som ett giltigt Qlik Sense-uttryck och returnerar i så fall värdet för uttrycket som en sträng. Om indatasträngen inte är ett giltigt uttryck returneras NULL.

Syntax:

```
Evaluate(expression_text)
```

Returnerad datatyp: dual

> *Denna strängfunktion kan inte användas i diagramuttryck.*

Exempel och resultat:

Funktionsexempel	Resultat
`Evaluate ( 5 * 8 )`	Returnerar '40'

Laddningsskriptsexempel

```
Load  Evaluate(String) as Evaluated, String Inline [String 4 5+3 0123456789012345678 Today()
];
```

Resultat

Sträng	Utvärderat
4	4
5+3	8
0123456789012345678	0123456789012345678
Today()	2022-02-02

FindOneOf

FindOneOf() söker en sträng för att hitta positionen för eventuella tecken som ingår i en uppsättning angivna tecken. Enbart positionen för det första av dessa tecken returneras, om inte ett tredje argument (med ett värde större än 1) tillhandahålls. Om ingen motsvarighet påträffas, returneras **0**.

Syntax:

```
FindOneOf(text, char_set[, count])
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
char_set	En teckenuppsättning att söka efter i text.
count	Definierar vilka förekomster av något av tecknen som sökningen ska inkludera. Till exempel, söker värdet 2 efter den andra förekomsten.

Exempel: Diagramuttryck

Exempel	Resultat
`FindOneOf( 'my example text string', 'et%s')`	Returnerar ”4” eftersom ”e” är det fjärde tecknet i exempelsträngen.
`FindOneOf( 'my example text string', 'et%s', 3)`	Returnerar ”12” eftersom sökningen avser något av tecknen e, t, % eller s, och ”t” är den tredje förekomsten i position 12 i exempelsträngen.
`FindOneOf( 'my example text string', '¤%&')`	Returnerar ”0” eftersom inget av tecknen ¤, % eller & förekommer i exempelsträngen.

Exempel: Laddningsskript

```
Load * Inline [SearchFor, Occurrence et%s,1 et%s,3 ¤%&,1]
```

Resultat

SearchFor	Förekomst	FindOneOf('my example text string', SearchFor, Occurrence)
et%s	1	4
et%s	3	12
¤%&	1	0

Hash128

Hash128() returnerar en 128-bitars hashning av de kombinerade indatauttrycksvärdena. Resulterar i en 22-teckensträng.

Syntax:

```
Hash128(expr{, expression})
```

Returnerad datatyp: sträng

Exempel: Diagramuttryck

Exempel	Resultat
Hash128 ('abc', 'xyz', '123')	Returnerar 'MA&5]6+3=:>:>G%S<U*S2+'.
Hash128 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Returnerar 'G7*=6GKPJ(Z+)^KM?<$'A+'.

Exempel: Laddningsskript

```
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 ];
```

Resultat

Region	År	Månad	Hash128
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2+
EU	2022	01	B40^K&[T@!;VB'XR]<5=/$
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!
US	2022	02	C6@#]4#_G-(]J7EQY#KRW0

Hash160

Hash160() returnerar en 160-bitars hashning av de kombinerade indatauttrycksvärdena. Resulterar i en 27-teckensträng.

Syntax:

```
Hash160(expr{, expression})
```

Returnerad datatyp: sträng

Exempel: Diagramuttryck

Exempel	Resultat
Hash160 ('abc', 'xyz', '123')	Returnerar 'MA&5]6+3=:>;>G%S<U*S2I:`=X*'.
Hash160 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Returnerar 'G7*=6GKPJ (Z+)^KM?<$'AI.)?U$'.

Exempel: Laddningsskript

```
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 ];
```

Resultat

Region	År	Månad	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() returnerar en 256-bitars hashning av de kombinerade indatauttrycksvärdena. Resulterar i en 43-teckensträng.

Syntax:

```
Hash256(expr{, expression})
```

Returnerad datatyp: sträng

Exempel: Diagramuttryck

Exempel	Resultat
Hash256 ('abc', 'xyz', '123')	Returnerar 'MA&5]6+3=:>;>G%S<U*S2I:`=X*A.IO*8N\%Y7Q;YEJ'.
Hash256 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Returnerar 'G7*=6GKPJ(Z+)^KM?<$'AI.)?U$#X2RB [:0ZP=+Z`F:'.

Exempel: Laddningsskript

```
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 ];
```

Resultat

Region	År	Månad	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() söker en sträng för att hitta startpositionen för den n:e förekomsten av en angiven delsträng. Ett valbart tredje argument ger värdet för n, vilket annars är 1. Ett negativt värde söker från slutet av strängen. Strängens positioner är numrerade från **1** och uppåt.

Syntax:

```
Index(text, substring[, count])
```

Returnerad datatyp: heltal

Argument:

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
substring	En teckensträng att söka efter i text.
count	Definierar vilken förekomst av **substring** som sökningen ska göras för. Till exempel, söker värdet 2 efter den andra förekomsten.

Exempel och resultat:

Exempel	Resultat
Index('abcdefg', 'cd')	Returnerar 3
Index('abcdabcd', 'b', 2)	Returnerar 6 (den andra förekomsten av ”b”)
Index('abcdabcd', 'b',-2)	Returnerar 2 (den andra förekomsten av ”b” räknat bakifrån)
`Left( Date, Index( Date,'-') -1)` where **Date** = 1997-07-14	Returnerar 1997
`Mid( Date, Index( Date, '-', 2 ) -2, 2 )` where **Date** = 1997-07-14	Returnerar 07

Exempel: Skript

```
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 ];
```

KeepChar

KeepChar() returnerar en sträng som består av första strängen, ”text”, minus alla eventuella tecken som INTE ingår i den andra strängen, ”keep_chars”.

Syntax:

```
KeepChar(text, keep_chars)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
keep_chars	En sträng som innehåller tecknen i text som ska behållas.

Exempel: Diagramuttryck

Exempel	Resultat
`KeepChar ( 'a1b2c3','123' )`	Returnerar 123
`KeepChar ( 'a1b2c3','1234' )`	Returnerar 123
`KeepChar ( 'a1b22c3','1234' )`	Returnerar 1223
`KeepChar ( 'a1b2c3','312' )`	Returnerar 123

Exempel: Laddningsskript

```
T1: Load  *,  keepchar(String1, String2) as KeepChar;   Load * inline [ String1, String2
'a1b2c3', '123' ];
```

Resultat

Qlik Sense-tabellen visar utdata vid användning av *KeepChar*-funktionen i laddningsskriptet.

String1	String2	KeepChar
a1b2c3	123	123

Se även:

PurgeChar (page 794)

Left

Left() returnerar en sträng som består av de första (längst till vänster) tecknen i indatasträngen, där antalet tecken fastställs av det andra argumentet.

Syntax:

```
Left(text, count)
```

Returnerad datatyp: sträng

Argument:

Argument	Beskrivning
text	Den ursprungliga strängen.
count	Definierar antal tecken som ska inkluderas från vänster sida av strängen **text**.

Exempel: Diagramuttryck

Exempel	Resultat
`Left('abcdef', 3)`	Returnerar 'abc'

Exempel: Laddningsskript

```
T1: Load  *,  left(Text,Start) as Left;        Load * inline [ Text, Start 'abcdef', 3 '2021-07-14', 4 '2021-07-14', 2 ];
```

Resultat

Qlik Sense-tabellen visar utdata vid användning av *Left*-funktionen i laddningsskriptet.

Text	Start	Left
abcdef	3	abc
2021-07-14	4	2021
2021-07-14	2	20

Se även *Index (page 787)*, som ger möjlighet till en mer komplex stränganalys.

Len

Len() returnerar längden för indatasträngen.

Syntax:

```
Len(text)
```

Returnerad datatyp: heltal

Exempel: Diagramuttryck

Exempel	Resultat
`Len('Peter')`	Returnerar '5'

Exempel: Laddningsskript

```
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 ];
```

Resultat

Sträng	NewString
this is a sample text string	This is a sample text string
capitalize first letter only	Capitalize first letter only

LevenshteinDist

LevenshteinDist() returnerar Levenshtein-avståndet mellan två strängar. Det definieras som det minsta antalet enkelteckensredigeringar (infogningar, borttagningar eller ersättningar) som krävs för att förvandla en sträng till en annan. Funktionen är användbar vid ungefärliga strängjämförelser.

Syntax:

```
LevenshteinDist(text1, text2)
```

Returnerad datatyp: heltal

Exempel: Diagramuttryck

Exempel	Resultat
`LevenshteinDist('Kitten','Sitting')`	Returnerar '3'

Exempel: Laddningsskript

Laddningsskript

```
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;
```

Resultat

ID	String_1	String_2	LevenshteinDistance
1	Silver	Sliver	2
2	Silver	SSiver	2
3	Silver	SSiveer	3
4	Gold	Fetstil	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() konverterar alla tecken i indatasträngen till gemener.

Syntax:

```
Lower(text)
```

Returnerad datatyp: sträng

Exempel: Diagramuttryck

Exempel	Resultat
`Lower('abcD')`	Returnerar 'abcd'

Exempel: Laddningsskript

```
Load  String, Lower(String) Inline [String rHode iSland washingTon d.C. new york];
```

Resultat

Sträng	Lower(String)
rHode iSland	rhode island
washingTon d.C.	washington d.c.
new york	new york

LTrim

LTrim() returnerar strängen rensad på inledande blankstegstecken.

Syntax:

```
LTrim(text)
```

Returnerad datatyp: sträng

Exempel: Diagramuttryck

Exempel	Resultat
`LTrim( ' abc' )`	Returnerar 'abc'
`LTrim( 'abc ' )`	Returnerar 'abc '

Exempel: Laddningsskript

```
Set verbatim=1;  T1:  Load *, len(LtrimString) as LtrimStringLength;       Load *, ltrim
(String) as LtrimString; Load *, len(String) as StringLength;                Load * Inline [
String '   abc    ' ' def  '];
```

Satsen "Set verbatim=1" finns med i exemplet för att säkerställa att utrymmena inte rensas automatiskt före demonstration av ltrim-funktionen. Mer information finns i Verbatim (page 144).

Resultat

Sträng	StringLength	LtrimStringLength
def	6	5
abc	10	7

Se även:

- *RTrim (page 797)*

Mid

Mid() returnerar den del av indatasträngen som börjar vid positionen för det tecken som definieras av det andra argumentet, ”start”, och returnerar det antal tecken som definieras av det tredje argumentet ”count”. Om ”count” utelämnas returneras resten av indatasträngen. Det första tecknet i indatasträngen numreras som 1.

Syntax:

```
Mid(text, start[, count])
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
start	Heltal som definierar positionen för det första tecknet i text som ska inkluderas.
count	Definierar stränglängden för utdatasträngen. Om den utelämnas inkluderas alla tecken från positionen som definieras av **start**.

Exempel: Diagramuttryck

Exempel	Resultat
`Mid('abcdef',3 )`	Returnerar 'cdef'
`Mid('abcdef',3, 2 )`	Returnerar 'cd'

Exempel: Laddningsskript

```
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 ];
```

Resultat

Qlik Sense-tabellen visar utdata vid användning av *Mid*-funktionen i laddningsskriptet.

Text	Start	Mid1	Count	Mid2
abcdef	2	bcdef	3	bcd
abcdef	3	cdef	2	cd
210714	2	10714	3	107
210714	3	0714	2	07

Se även:

Index (page 787)

Ord

Ord() returnerar talnumret för Unicode-koden för det första tecknet i indatasträngen.

Syntax:

```
Ord(text)
```

Returnerad datatyp: heltal

Exempel och resultat:

Exempel: Diagramuttryck

Exempel	Resultat
`Ord('A')`	Returnerar heltalet 65.
`Ord('Ab')`	Returnerar heltalet 65.

Exempel: Laddningsskript

```
//Guqin (Chinese: 古琴) - 7-stringed zithers T2: Load *,  ord(Chinese) as OrdUnicode,
        ord(Western) as OrdASCII;                    Load * inline [ Chinese, Western 古琴,
Guqin ];
```

Resultat:

Chinese	Western	OrdASCII	OrdUnicode
古琴	Guqin	71	21476

PurgeChar

PurgeChar() returnerar en sträng som består av de tecken som finns i indatasträngen (”text”), utom alla eventuella tecken som visas i det andra argumentet (”remove_chars”).

Syntax:

```
PurgeChar(text, remove_chars)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
remove_chars	En sträng som innehåller tecknen i text som ska tas bort.

Returnerad datatyp: sträng

Exempel: Diagramuttryck

Exempel	Resultat
`PurgeChar ( 'a1b2c3','123' )`	Returnerar ”abc”.
`PurgeChar ( 'a1b2c3','312' )`	Returnerar ”abc”.

Exempel: Laddningsskript

```
T1: Load  *,  purgechar(String1, String2) as PurgeChar;   Load * inline [ String1, String2
'a1b2c3', '123' ];
```

Resultat

Qlik Sense-tabellen visar utdata vid användning av *PurgeChar*-funktionen i laddningsskriptet.

String1	String2	PurgeChar
a1b2c3	123	abc

Se även:

KeepChar (page 787)

Repeat

Repeat() bildar en sträng som består av indatasträngen som upprepats det antal gånger som definieras av det andra argumentet.

Syntax:

```
Repeat(text[, repeat_count])
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
repeat_count	Definierar antalet gånger tecknen i strängen **text** ska upprepas i utdatasträngen.

Exempel: Diagramuttryck

Exempel	Resultat
`Repeat( ' * ', rating )` when **rating** = 4	Returnerar '****'

Exempel: Laddningsskript

```
T1: Load *, repeat(String,2) as Repeat; Load * inline [ String hello world! hOw aRe you? ];
```

Resultat

Sträng	Upprepa
hello world!	hello world!hello world!
hOw aRe you?	hOw aRe you?hOw aRe you?

Replace

Replace() returnerar en sträng efter att ha ersatt alla förekomster av en given delsträng inom indatasträngen med en annan delsträng. Funktionen är inte rekursiv och fungerar från vänster till höger.

Syntax:

```
Replace(text, from_str, to_str)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
from_str	En sträng som kan förekomma en eller flera gånger inom indatasträngen **text**.
to_str	Strängen som ersätter alla förekomster av **from_str** inom strängen **text**.

Exempel och resultat:

Exempel	Resultat
`Replace('abccde','cc','xyz')`	Returnerar 'abxyzde'

Se även:

Right

Right() returnerar en sträng som består av de sista (längst till höger) tecknen av indatasträngen, där antalet tecken fastställs av det andra argumentet.

Syntax:

```
Right(text, count)
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
text	Den ursprungliga strängen.
count	Definierar det antal tecken som ska inkluderas från delen längst till höger i strängen **text**.

Exempel: Diagramuttryck

Exempel	Resultat
`Right('abcdef', 3)`	Returnerar 'def'

Exempel: Laddningsskript

```
T1: Load  *,  right(Text,Start) as Right;        Load * inline [ Text, Start 'abcdef', 3
'2021-07-14', 4 '2021-07-14', 2 ];
```

Resultat

Qlik Sense-tabellen visar utdata vid användning av *Right*-funktionen i laddningsskriptet.

Text	Start	Right
abcdef	3	def
2021-07-14	4	7-14
2021-07-14	2	14

RTrim

RTrim() returnerar strängen rensad på avslutande blankategstecken.

Syntax:

```
RTrim(text)
```

Returnerad datatyp: sträng

Exempel: Diagramuttryck

Exempel	Resultat
`RTrim( ' abc' )`	Returnerar ' abc'
`RTrim( 'abc ' )`	Returnerar 'abc'

Exempel: Laddningsskript

```
Set verbatim=1;    T1:  Load *, len(RtrimString) as RtrimStringLength;         Load *, rtrim
(String) as RtrimString; Load *, len(String) as StringLength;              Load * Inline [
String '   abc    ' ' def  '];
```

Satsen ”Set verbatim=1” finns med i exemplet för att säkerställa att utrymmena inte rensas automatiskt före demonstration av rtrim-funktionen. Mer information finns i Verbatim (page 144).

Resultat

Sträng	StringLength	RtrimStringLength
def	6	4
abc	10	6

Se även:

- *LTrim (page 792)*

SubField

Subfield() används för att extrahera delsträngkomponenter från ett överordnat strängfält där de ursprungliga postfälten består av två eller fler delar, som separeras av en avgränsare.

Subfield()-funktionen kan användas till exempel för att extrahera förnamn och efternamn från en lista av poster som innehåller fullständiga namn, komponentdelarna i ett sökvägsnamn eller för att extrahera data från kommaavgränsade tabeller.

Om du använder funktionen **Subfield()** i en **LOAD**-sats med den valbara parametern field_no utelämnad kommer en fullständig post genereras för varje delsträng. Om flera fält laddas med hjälp av **Subfield()** skapas de kartesiska produkterna för alla kombinationer.

Syntax:

```
SubField(text, delimiter[, field_no ])
```

Returnerad datatyp: sträng

Argument:

Argument

Argument	Beskrivning
text	Den ursprungliga strängen. Detta kan vara en hårdkodad text, en variabel, en dollarteckensexpansion eller ett annat uttryck.
delimiter	Ett tecken med indata-**text** som delar in strängen i dess komponentdelar.
field_no	Det valbara tredje argumentet är ett heltal som vilka av delsträngarna i den överordnade strängen **text** som ska returneras. Använd värdet 1 om du vill returnera den första delsträngen, 2 om du vill returnera den delsträngen och så vidare. • Om **field_no** är ett positivt värde extraheras understrängar från vänster till höger. • Om **field_no** är ett negativt värde extraheras understrängar från höger till vänster.

SubField() kan användas istället för att använda komplexa kombinationer av funktioner, som t.ex. Len(), Right(), Left(), Mid() och andra strängfunktioner.

Exempel: Skript- och diagramuttryck som använder SubField

Exempel – skript- och diagramuttryck

Grundläggande exempel

Exempel	Resultat
`SubField(S, ';' ,2)`	Returnerar 'cde' om **S** är 'abc;cde;efg'.
`SubField(S, ';' ,1)`	Returnerar en tom sträng om **S** är en tom sträng.
`SubField(S, ';' ,1)`	Returnerar en tom sträng om **S** är ';'.
Anta att du har en variabel som innehåller ett sökvägsnamn vMyPath, `Set vMyPath=\Users\ext_jrb\Documents\Qlik\Sense\Apps;.`	I ett text- och bilddiagram kan du lägga till mått, t.ex. `SubField(vMyPath, '\',-3)`, som resulterar i 'Qlik', eftersom det är den tredje delsträngen till höger om variabeln vMyPath.

Skriptexempel 1

Laddningsskript

Ladda följande skriptuttryck och data i skriptredigeraren.

```
FullName:                                    LOAD * inline [ Name 'Dave Owen' 'Joe Tem' ];  SepNames:  Lo
(Name, ' ',1) as FirstName, SubField(Name, ' ',-1) as Surname Resident FullName; Drop Table
FullName;
```

Skapa en visualisering

Skapa en tabellvisualisering i ett Qlik Sense-ark med **Name**, **FirstName** och **SurName** som dimensioner.

Resultat

Name	FirstName	SurName
Dave Owen	Dave	Owen
Joe Tem	Joe	Tem

Förklaring

Funktionen **SubField()** extraherar den första delsträngen i **Name** genom att ställa in argumentet **field_no** på 1. Eftersom värdet i **field_no** är positivt, används ordningen vänster till höger vid extrahering av delsträngen. Ett andra funktionsanrop extraherar den andra delsträngen genom att ställa in argumentet **field _no** på -1, vilket extraherar delsträngen i ordningen höger till vänster.

Skriptexempel 2

Laddningsskript

Ladda följande skriptuttryck och data i skriptredigeraren.

```
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 '|');
```

Skapa en visualisering

Skapa en tabellvisualisering i ett Qlik Sense-ark med **Instrument**, **Player** och **Project** som dimensioner.

Resultat

Instrument	Player	Project
Guitar	Mike	Music
Guitar	Mike	Video
Guitar	Mike	OST
Guitar	Neil	Music
Guitar	Neil	Video

Instrument	Player	Project
Guitar	Neil	OST
Synth	Jen	Music
Synth	Jen	Video
Synth	Jen	OST
Synth	Jo	Music
Synth	Neil	Music
Synth	Neil	Video
Synth	Neil	OST

Förklaring

Det här exemplet visar hur användning av flera instanser av funktionen **Subfield()**, där parametern field_no har utelämnats, inom samma **LOAD**-sats skapar kartesiska produkter av alla kombinationerna. Alternativet **DISTINCT** används för att undvika att skapa dubbletter av poster.

SubStringCount

SubstringCount() returnerar antalet förekomster av angiven delsträng i indatasträngtexten. Om det inte blir någon träff returneras 0.

Syntax:

```
SubStringCount(text, sub_string)
```

Returnerad datatyp: heltal

Argument:

Argument	Beskrivning
text	Den ursprungliga strängen.
sub_string	En sträng som kan förekomma en eller flera gånger inom indatasträngen **text**.

Exempel: Diagramuttryck

Exempel	Resultat
`SubStringCount ( 'abcdefgcdxyz', 'cd' )`	Returnerar '2'
`SubStringCount ( 'abcdefgcdxyz', 'dc' )`	Returnerar '0'

Exempel: Laddningsskript

```
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 ];
```

Resultat

Strings	SubStringCount_AB
aB/cd/ef/gh/Abc/abandoned	3
ABC:DEF:GHI:AB:CD:EF:GH	2

TextBetween

TextBetween() returnerar texten i indatasträngen som finns mellan de tecken som definierats som avgränsare.

Syntax:

```
TextBetween(text, delimiter1, delimiter2[, n])
```

Returnerad datatyp: sträng

Argument:

Argument	Beskrivning
text	Den ursprungliga strängen.
delimiter1	Anger det första avgränsartecknet (eller strängen) som ska sökas efter i **text**.
delimiter2	Anger det andra avgränsartecknet (eller strängen) som ska sökas efter i **text**.
n	Definierar vilken förekomst av avgränsarparet som ska sökas emellan. Till exempel returnerar värdet 2 tecknen mellan den andra förekomsten av avgränsare1 och den andra förekomsten av avgränsare2.

Exempel: Diagramuttryck

Exempel	Resultat
`TextBetween('<abc>', '<', '>')`	Returnerar 'abc'
`TextBetween('<abc><de>', '<', '>',2)`	Returnerar 'de'
`TextBetween('abc', '<', '>')` `TextBetween('<a<b', '<', '>')`	Båda exemplen returnerar NULL. Om någon av avgränsarna inte hittas i strängen returneras NULL.
`TextBetween('<>', '<', '>')`	Returnerar en sträng med längden noll.
`TextBetween('<abc>', '<', '>', 2)`	Returnerar NULL, eftersom n är större än antalet förekomster av avgränsarna.

Exempel: Laddningsskript

```
Load *, textbetween(Text,'<','>') as TextBetween, textbetween(Text,'<','>',2) as
SecondTextBetween; Load * inline [ Text <abc><de> <def><ghi><jkl> ];
```

Resultat

Text	TextBetween	SecondTextBetween
<abc><de>	abc	de
<def><ghi><jkl>	def	ghi

Trim

Trim() returnerar strängen rensad på inledande och avslutande blankstegstecken.

Syntax:

```
Trim(text)
```

Returnerad datatyp: sträng

Exempel och resultat:

Exempel: Diagramuttryck

Exempel	Resultat
`Trim( ' abc' )`	Returnerar 'abc'
`Trim( 'abc ' )`	Returnerar 'abc'
`Trim( ' abc ' )`	Returnerar 'abc'

Exempel: Laddningsskript

```
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');
```

Satsen "Set verbatim=1" finns med i exemplet för att säkerställa att utrymmena inte rensas automatiskt före demonstration av trim-funktionen. Mer information finns i Verbatim (page 144).

Resultat:

Sträng	StringLength	TrimStringLength
def	6	3
abc	10	3

Upper

Upper() konverterar alla tecken i indatasträngen till versaler för alla texttecken i uttrycket. Tal och symboler ignoreras.

Syntax:

```
Upper(text)
```

Returnerad datatyp: sträng

Exempel: Diagramuttryck

Exempel	Resultat
`Upper(' abcD')`	Returnerar 'ABCD'

Exempel: Laddningsskript

```
Load  String,Upper(String) Inline [String rHode iSland washingTon d.C. new york];
```

Resultat

Sträng	Upper(String)
rHode iSland	RHODE ISLAND
washingTon d.C.	WASHINGTON D.C.
new york	NEW YORK

5.25 Systemfunktioner

Systemfunktionerna låter dig nå egenskaperna för systemet, enheten och Qlik Sense-appen.

Översikt av systemfunktioner

En del av funktionerna beskrivs mer ingående efter översikten. För de här funktionerna kan du klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

Author()

Den här funktionen returnerar en textsträng som anger egenskapen upphovsman för den aktuella appen. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

Egenskapen Upphovsman kan inte anges i den aktuella versionen av Qlik Sense. Om du migrerar ett QlikView-dokument kvarstår egenskapen upphovsman.

ClientPlatform()

Den här funktionen returnerar användaragentsträngen i klientwebbläsaren. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

Exempel:

```
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/35.0.1916.114 Safari/537.36
```

ComputerName

Denna funktion returnerar en sträng som anger namnet på aktuell dator enligt operativsystemet. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

Om datorns namn består av fler än 15 tecken ingår bara de 15 första i strängen.

```
ComputerName( )
```

DocumentName

Den här funktionen returnerar en textsträng som anger namnet på den aktuella Qlik Sense-appen, utan sökväg men med filtillägg. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

```
DocumentName( )
```

DocumentPath

Denna skriptfunktion returnerar en sträng som innehåller den kompletta sökvägen till den aktuella Qlik Sense-appen. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

```
DocumentPath( )
```

Funktionen stöds inte i standardläget.

DocumentTitle

Den här funktionen returnerar en textsträng som anger namnet på den aktuella Qlik Sense-appen. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

```
DocumentTitle( )
```

EngineVersion

Denna funktion returnerar komplett information om Qlik Sense-motorversionen i form av en sträng.

```
EngineVersion ()
```

GetCollationLocale

Den här skriptfunktionen returnerar kulturnamnet för den sorteringsspråkvariant som används. Om variabeln CollationLocale inte har ställts in returneras användarens faktiska datorspråkvariant.

```
GetCollationLocale( )
```

GetObjectField

GetObjectField() returnerar namnet på dimensionen. **Index** är ett valfritt heltal som anger dimensionen som ska returneras.

```
GetObjectField - diagramfunktion([index])
```

GetRegistryString

Den här funktionen returnerar värdet för en nyckel i Windows-registret. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

```
GetRegistryString(path, key)
```

Funktionen stöds inte i standardläget.

IsPartialReload

Den här funktionen returnerar -1 (True) om den aktuella återinläsningen är partiell, annars 0 (False).

```
IsPartialReload ()
```

OSUser

Den här funktionen returnerar en sträng som innehåller namnet på den användare som är kopplad för närvarande. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

```
OSUser( )
```

I Qlik Sense Desktop och Qlik Sense Mobile Client Managed returnerar den här funktionen alltid "Personal\Me".

ProductVersion

Denna funktion returnerar komplett information om Qlik Sense-version och byggnummer i form av en sträng.

Funktionen är utfasad och ersatt med **EngineVersion()**.

```
ProductVersion ()
```

ReloadTime

Den här funktionen returnerar en tidsangivelse för när dataladdningen senast avslutades. Den kan användas på både dataladdningsskriptet och i ett diagramuttryck.

```
ReloadTime( )
```

StateName

StateName() returnerar namnet på det alternativa tillståndet för visualiseringen där den används. StateName kan till exempel användas för att skapa visualiseringar med text och färger som är dynamiska för att återspegla när tillståndet för en visualisering ändras. Denna funktion kan användas i diagramuttryck men kan inte användas för att fastställa det tillstånd som uttrycket syftar på.

```
StateName - diagramfunktion()
```

EngineVersion

Denna funktion returnerar komplett information om Qlik Sense-motorversionen i form av en sträng.

Syntax:

```
EngineVersion()
```

IsPartialReload

Den här funktionen returnerar -1 (True) om den aktuella återinläsningen är partiell, annars 0 (False).

Syntax:

```
IsPartialReload()
```

ProductVersion

Denna funktion returnerar komplett information om Qlik Sense-version och byggnummer i form av en sträng. Funktionen är utfasad och ersatt med **EngineVersion()**.

Syntax:

```
ProductVersion()
```

StateName - diagramfunktion

StateName() returnerar namnet på det alternativa tillståndet för visualiseringen där den används. StateName kan till exempel användas för att skapa visualiseringar med text och färger som är dynamiska för att återspegla när tillståndet för en visualisering ändras. Denna funktion kan användas i diagramuttryck men kan inte användas för att fastställa det tillstånd som uttrycket syftar på.

Syntax:

```
StateName ()
```

Example 1:

Dynamisk text

```
='Region - ' & if(StateName() = '$', 'Default', StateName())
```

Example 2:

Dynamiska färger

```
if(StateName() = 'Group 1', rgb(152, 171, 206),
      if(StateName() = 'Group 2', rgb(187, 200, 179),
            rgb(210, 210, 210)
      )
)
```

5.26 Tabellfunktioner

Tabellfunktionerna returnerar information om den datatabell som för närvarande läses. Om inget tabellnamn har angetts och funktionen används inom en LOAD-sats antas den aktuella tabellen.

Alla funktioner kan användas i dataladdningsskriptet, men endast NoOfRows kan användas i ett diagramuttryck.

Tabellfunktioner - översikt

En del av funktionerna beskrivs mer ingående efter översikten. För de här funktionerna kan du klicka på namnet på funktionen i syntaxen för att omedelbart få åtkomst till detaljerna för den specifika funktionen.

FieldName

Skriptfunktionen FieldName returnerar namnet på fältet med det angivna numret inom en tidigare inläst tabell. Om funktionen används inom en LOAD-sats får den inte referera till den tabell som för närvarande används.

```
FieldName (field_number ,table_name)
```

FieldNumber

Skriptfunktionen FieldNumber returnerar numret på ett angivet fält i en tidigare inläst tabell. Om funktionen används inom en LOAD-sats får den inte referera till den tabell som för närvarande används.

```
FieldNumber (field_name ,table_name)
```

NoOfFields

Skriptfunktionen NoOfFields returnerar antalet fält i en tidigare inläst tabell. Om funktionen används inom en LOAD-sats får den inte referera till den tabell som för närvarande används.

```
NoOfFields (table_name)
```

NoOfRows

Funktionen NoOfRows returnerar antalet rader (poster) i en tidigare inläst tabell. Om funktionen används inom en LOAD-sats får den inte referera till den tabell som för närvarande används.

```
NoOfRows (table_name)
```

NoOfTables

Den här skriptfunktionen returnerar antalet tidigare inlästa tabeller.

```
NoOfTables()
```

TableName

Den här skriptfunktionen returnerar namnet på tabellen med det angivna numret.

```
TableName(table_number)
```

TableNumber

Den här skriptfunktionen returnerar numret på den angivna tabellen. Den första tabellen har nummer 0.

Om table_name inte finns returneras NULL.

```
TableNumber(table_name)
```

Exempel:

I det här exemplet vill vi skapa en tabell med information om tabeller och fält som har laddats.

Först ska vi ladda några exempeldata. Detta skapar de två tabeller som kommer att användas för att illustrera tabellfunktionerna som beskrivs i detta avsnitt.

```
Characters:
Load Chr(RecNo()+Ord('A')-1) as Alpha, RecNo() as Num autogenerate 26;

ASCII:
Load
 if(RecNo()>=65 and RecNo()<=90,RecNo()-64) as Num,
 Chr(RecNo()) as AsciiAlpha,
 RecNo() as AsciiNum
autogenerate 255
 Where (RecNo()>=32 and RecNo()<=126) or RecNo()>=160 ;
```

Sedan itererar vi igenom tabellerna som har laddats med hjälp av funktionen NoOfTables och sedan genom fälten i varje tabell, med hjälp av funktionen NoOfFields och laddar information med hjälp av tabellfunktionerna.

```
//Iterate through the loaded tables
For t = 0 to NoOfTables() - 1

//Iterate through the fields of table
 For f = 1 to NoOfFields(TableName($(t)))
  Tables:
  Load
   TableName($(t)) as Table,
   TableNumber(TableName($(t))) as TableNo,
   NoOfRows(TableName($(t))) as TableRows,
   FieldName($(f),TableName($(t))) as Field,
   FieldNumber(FieldName($(f),TableName($(t))),TableName($(t))) as FieldNo
   Autogenerate 1;
 Next f
Next t;
```

Tabellen Tables som skapas ser ut så här:

Load table

Table	TableNo	TableRows	Field	FieldNo
Characters	0	26	Alpha	1
Characters	0	26	Num	2

Table	TableNo	TableRows	Field	FieldNo
ASCII	1	191	Num	1
ASCII	1	191	AsciiAlpha	2
ASCII	1	191	AsciiNum	3

FieldName

Skriptfunktionen **FieldName** returnerar namnet på fältet med det angivna numret inom en tidigare inläst tabell. Om funktionen används inom en **LOAD**-sats får den inte referera till den tabell som för närvarande används.

Syntax:

```
FieldName(field_number ,table_name)
```

Argument:

Argument

Argument	Beskrivning
field_number	Fältnumret till det fält som du vill referera till.
table_name	Den tabell som innehåller det fält som du vill referera till.

Exempel:

```
LET a = FieldName(4,'tab1');
```

FieldNumber

Skriptfunktionen **FieldNumber** returnerar numret på ett angivet fält i en tidigare inläst tabell. Om funktionen används inom en **LOAD**-sats får den inte referera till den tabell som för närvarande används.

Syntax:

```
FieldNumber(field_name ,table_name)
```

Argument:

Argument

Argument	Beskrivning
field_name	Fältets namn.
table_name	Namnet på den tabell som innehåller fältet.

Om fältet field_name inte finns i table_name, eller table_name inte finns, returnerar funktionen 0.

Exempel:

```
LET a = FieldNumber('Customer','tab1');
```

NoOfFields

Skriptfunktionen **NoOfFields** returnerar antalet fält i en tidigare inläst tabell. Om funktionen används inom en **LOAD**-sats får den inte referera till den tabell som för närvarande används.

Syntax:

```
NoOfFields(table_name)
```

Argument:

Argument

Argument	Beskrivning
table_name	Tabellens namn.

Exempel:

```
LET a = NoOfFields('tab1');
```

NoOfRows

Funktionen **NoOfRows** returnerar antalet rader (poster) i en tidigare inläst tabell. Om funktionen används inom en **LOAD**-sats får den inte referera till den tabell som för närvarande används.

Syntax:

```
NoOfRows(table_name)
```

Argument:

Argument

Argument	Beskrivning
table_name	Tabellens namn.

Exempel:

```
LET a = NoOfRows('tab1');
```

5.27 Trigonometriska och hyperboliska funktioner

Den här delen beskriver funktioner för att utföra trigonometriska och hyperboliska operationer. I alla funktionerna är argumenten uttryck som resulterar i vinklar mätta i radianer, där **x** ska tolkas som ett verkligt tal.

Alla vinklar mäts i radianer.

Alla funktioner kan användas både i dataladdningsskriptet och diagramuttryck.

cos
Cosinus av **x**. Resultatet blir ett tal mellan -1 och 1.

```
cos( x )
```

acos
Invers cosinus av **x**. Funktionen är bara definierad om -1≤**x**≤1. Resultatet blir ett tal mellan 0 och π.

```
acos( x )
```

sin
Sinus av **x**. Resultatet blir ett tal mellan -1 och 1.

```
sin( x )
```

asin
Invers sinus av **x**. Funktionen är bara definierad om -1≤**x**≤1. Resultatet blir ett tal mellan - π/2 och π/2.

```
asin( x )
```

tan
Tangens av **x**. Resultatet blir ett reellt tal.

```
tan( x )
```

atan
Invers tangens av **x**. Resultatet blir ett tal mellan - π/2 och π/2.

```
atan( x )
```

atan2
Tvådimensionell generalisering av invers tangens-funktionen. Returnerar vinkeln mellan origo och den punkt som motsvaras av koordinaterna **x** och **y**. Resultatet blir ett tal mellan - π och + π.

```
atan2( y,x )
```

cosh
Cosinus hyberbolicus av **x**. Resultatet blir ett positivt reellt tal.

```
cosh( x )
```

sinh

Sinus hyberbolicus av **x**. Resultatet blir ett reellt tal.

```
sinh( x )
```

tanh

Tangens hyberbolicus av **x**. Resultatet blir ett reellt tal.

```
tanh( x )
```

acosh

Invers cosinus hyberbolicus av **x**. Resultatet blir ett positivt reellt tal.

```
acosh( x )
```

asinh

Invers csinus hyberbolicus av **x**. Resultatet blir ett reellt tal.

```
asinh( x )
```

atanh

Invers tangens hyberbolicus av **x**. Resultatet blir ett reellt tal.

```
atanh( x )
```

Exempel:

Följande skript laddar exempeldata och laddar sedan en tabell som innehåller de beräknade trigonometriska och hyperboliska operationerna för värdena.

```
SampleData:
LOAD * Inline
[Value
-1
0
1];

Results:
Load *,
cos(Value),
acos(Value),
sin(Value),
asin(Value),
tan(Value),
atan(Value),
atan2(Value, Value),
cosh(Value),
sinh(Value),
tanh(Value)
RESIDENT SampleData;

Drop Table SampleData;
```

6 Behörighetskontroll för filsystem

Av säkerhetsskäl har Qlik Sense i standardläge inte stöd för sökvägar i dataladdningsskriptet eller funktioner och variabler som exponerar filsystemet.

Eftersom sökvägar hanterades i QlikView är det dock möjligt att inaktivera standardläget och använda det bakåtkompatibla läget om du vill återanvända QlikView-dataladdningsskript.

Att inaktivera standardläge kan skapa en säkerhetsrisk genom att exponera filsystemet.

Inaktivera standardläge (page 820)

6.1 Säkerhetsaspekter vid anslutning till filbaserade ODBC- och OLE DB-dataanslutningar

ODBC- och OLE DB-dataanslutningar som använder filbaserade drivrutiner kommer att visa sökvägen till den anslutna datafilen i kopplingssträngen. Sökvägen kan exponeras när anslutningen redigeras, i dialogrutan för urval av data eller i vissa SQL-frågor. Detta gäller både i standardläge och bakåtkompatibelt läge.

Om att visa sökvägen till datafilen inte är önskvärt, rekommenderas att ansluta till datafilen med hjälp av en mappdatakoppling om det går.

6.2 Begränsningar i standardläge

Flera satser, variabler och funktioner kan inte användas eller har begränsningar i standardläget. Användning av satser som inte stöds i dataladdningsskriptet ger ett fel när det körs. Felmeddelanden återfinns i skriptloggfilen. Användning av variabler och funktioner som inte stöds ger inga felmeddelanden eller loggfilsposter. Funktionen returnerar i stället NULL.

Det finns ingen indikation på att en variabel, sats eller funktion inte stöds när du redigerar dataladdningsskriptet.

Systemvariabler

Systemvariabler

Variabel	Standardläge	Bakåtkompatibelt läge	Definition
Floppy	Stöds ej	Stöds	Returnerar enhetsbeteckningen för den första diskettenhet som påträffas, vanligen *a:*.

Variabel	Standardläge	Bakåtkompatibelt läge	Definition
CD	Stöds ej	Stöds	Returnerar bokstaven på den första cd-rom-enhet som påträffas. Om ingen cd-rom-enhet påträffas, returneras *c:*.
QvPath	Stöds ej	Stöds	Returnerar sökströngen till Qlik Sense-programfilen.
QvRoot	Stöds ej	Stöds	Returnerar rotkatalogen till Qlik Sense-programfilen.
QvWorkPath	Stöds ej	Stöds	Returnerar sökströngen till den aktuella Qlik Sense-appen.
QvWorkRoot	Stöds ej	Stöds	Returnerar rotmappen till den aktuella Qlik Sense-appen.
WinPath	Stöds ej	Stöds	Returnerar sökströngen till Windows.
WinRoot	Stöds ej	Stöds	Returnerar Windows rotmapp.
$(include=...)	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Variabeln **Include/Must_Include** specificerar en fil som innehåller text som ska inkluderas i skriptet och utvärderas som skriptkod. Den används inte för att lägga till data. Du kan spara delar av skriptkoden i en separat textfil och återanvända den i flera appar. Detta är en användardefinierad variabel.

Vanliga skriptsatser

Vanliga skriptsatser

Sats	Standardläge	Bakåtkompatibelt läge	Definition
Binary	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Satsen **binary** används för att ladda data från en annan app.
Connect	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	**CONNECT**-satsen används för att ange Qlik Sense-åtkomst till en allmän databas via OLE DB/ODBC-gränssnittet. För ODBC måste datakällan först anges med hjälp av ODBC-administratören.
Directory	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	**Directory**-satsen anger i vilken katalog datafilerna ska sökas i följande **LOAD**-satser tills en ny **Directory**-sats anges.
Execute	Stöds ej	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	**Execute**-satsen använda för att köra andra program medan Qlik Sense laddar data, exempelvis för att göra de konverteringar som krävs.
LOAD from ...	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	**LOAD**-satsen laddar fält från en fil, direkt från data i skriptet, från en tidigare inläst tabell, från en webbsida, från resultatet av en efterföljande **SELECT**-sats eller genom att generera data automatiskt.

Sats	Standardläge	Bakåtkompatibelt läge	Definition
Store into ...	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	**Store**-satsen skapar en QVD-, CSV- eller text-fil.

Kontrollsatser i skriptet

Kontrollsatser i skriptet

Sats	Standardläge	Bakåtkompatibelt läge	Definition
For each... filelist mask/dirlist mask	Indata som stöds: Sökväg som använder bibliotekskoppling Returnerade utdata: Bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem Returnerade utdata: Bibliotekskoppling eller filsystemssökväg, beroende på indata	Syntaxen filelist mask skapar en kommaavgränsad lista över alla filer i den aktuella katalog som matchar **filelist mask**. Syntaxen **dirlist mask** skapar en kommaavgränsad lista över alla mappar i den aktuella mappen som matchar filnamnsmasken.

Filfunktioner

Filfunktioner

Funktion	Standardläge	Bakåtkompatibelt läge	Definition
Attribute()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Returnerar metataggarnas värde från olika mediafiler som text.
ConnectString()	Returnerade utdata: Bibliotekskopplingsnamn	Biblioteksanslutningens namn eller faktiska anslutning, beroende på indata	Returnerar den aktiva kopplingssträngen för ODBC- eller OLE DB-anslutningar.

Funktion	Standardläge	Bakåtkompatibelt läge	Definition
FileDir()	Returnerade utdata: Bibliotekskoppling	Returnerade utdata: Bibliotekskoppling eller filsystemssökväg, beroende på indata	**FileDir**-funktionen returnerar en textsträng som anger sökvägen till katalogen som innehåller den tabellfil som håller på att läsas in.
FilePath()	Returnerade utdata: Bibliotekskoppling	Returnerade utdata: Bibliotekskoppling eller filsystemssökväg, beroende på indata	**FilePath**-funktionen returnerar en textsträng som anger den kompletta sökvägen till den tabellfil som håller på att läsas in.
FileSize()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	**FileSize**-funktionen returnerar ett heltal som anger storleken i byte för filen filename eller, om inget filename angetts, för den tabellfil som håller på att läsas in.
FileTime()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	**FileTime**-funktionen returnerar en tidsmarkör för datum och tid för den senaste ändringen av filen filename. Om inget filename har angetts, använder funktionen den senast inlästa tabellfilen.

Funktion	Standardläge	Bakåtkompatibelt läge	Definition
GetFolderPath()	Stöds ej	Returnerade utdata: Absolut sökväg	Funktionen **GetFolderPath** returnerar värdet av funktionen Microsoft Windows *SHGetFolderPath*. Den här funktionen tar som indata namnet på en Microsoft Windows -mapp och returnerar den fullständiga sökvägen till mappen.
QvdCreateTime()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Denna skriptfunktion returnerar XML-huvudets tidsstämpel från en QVD-fil, om någon sådan finns, annars returneras NULL.
QvdFieldName()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Denna skriptfunktion returnerar namnet på fält nummer **fieldno** i en QVD-fil. Om fältet inte finns returneras NULL.
QvdNoOfFields()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Denna skriptfunktion returnerar antalet fält i en QVD-fil.
QvdNoOfRecords()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Denna skriptfunktion returnerar aktuellt antal poster i en QVD-fil.
QvdTableName()	Indata som stöds: Sökväg som använder bibliotekskoppling	Indata som stöds: Sökväg som använder bibliotekskoppling eller filsystem	Denna skriptfunktion returnerar namnet på tabellen i en QVD-fil.

Systemfunktioner

Systemfunktioner

Funktion	Standardläge	Bakåtkompatibelt läge	Definition
DocumentPath()	Stöds ej	Returnerade utdata: Absolut sökväg	Denna skriptfunktion returnerar en sträng som innehåller den kompletta sökvägen till den aktuella Qlik Sense-appen.
GetRegistryString()	Stöds ej	Stöds	Returnerar värdet hos en namngiven registernyckel med angiven sökväg. Funktionen kan användas både i diagram och i skriptet.

6.3 Inaktivera standardläge

Du kan inaktivera standardläge, eller med andra ord, ställa in bakåtkompatibelt läge, om du vill återanvända QlikView-dataladdningsskriptet som hänvisar till absoluta eller relativa filsökvägar samt bibliotekskopplingar.

Att inaktivera standardläge kan skapa en säkerhetsrisk genom att exponera filsystemet.

Qlik Sense

För Qlik Sense kan standardläget inaktiveras i QMC med hjälp av egenskapen **Standardläge**.

Qlik Sense Desktop

I Qlik Sense Desktop kan du ställa in standardläge/bakåtkompatibelt läge i *Settings.ini*.

Om du installerade Qlik Sense Desktop i den förvalda installationsplatsen, då finns *Settings.ini* i *C:\Users\{user}\Documents\Qlik\Sense\Settings.ini*. Om du installerade Qlik Sense Desktop i en mapp som du valt, då finns *Settings.ini* i mappen *Engine* i installationens sökväg.

Gör följande:

1. Öppna *Settings.ini* i en textredigerare.
2. Ändra *StandardReload=1* till *StandardReload=0*.
3. Spara filen och starta Qlik Sense Desktop.

Qlik Sense Desktop körs nu i bakåtkompatibelt läge.

Inställningar

Följande inställningar är tillgängliga för StandardReload:

- 1 (standardläge)
- 0 (bakåtkompatibelt läge)

7 QlikView-funktioner och satser som inte stöds i Qlik Sense

De flesta funktioner och satser som kan användas i QlikView-laddningsskript och diagramuttryck stöds också i Qlik Sense, men det finns några undantag, som beskrivs här.

7.1 Skriptsatser som inte stöds i Qlik Sense

QlikView-skriptsatser som inte stöds i Qlik Sense

Sats	Kommentarer
Command	Använd **SQL** i stället.
InputField	

7.2 Funktioner som inte stöds i Qlik Sense

I denna lista beskrivs QlikView-skript och diagramfunktioner som inte stöds i Qlik Sense.

- **GetCurrentField**
- **GetExtendedProperty**
- **Input**
- **InputAvg**
- **InputSum**
- **MsgBox**
- **NoOfReports**
- **ReportComment**
- **ReportId**
- **ReportName**
- **ReportNumber**

7.3 Prefix som inte stöds i Qlik Sense

I denna lista beskrivs QlikView-prefix som inte stöds i Qlik Sense.

- **Bundle**
- **Image_Size**
- **Info**

8 Funktioner och satser som inte rekommenderas i Qlik Sense

De flesta funktioner och satser som kan användas i QlikView-laddningsskript och diagramuttryck stöds även i Qlik Sense, men en del av dem rekommenderas inte för användning i Qlik Sense. De finns även funktioner och satser tillgängliga i tidigare versioner av Qlik Sense som har fasats ut.

Av kompatibilitetsskäl kommer de fortfarande att fungera som avsett, men man bör uppdatera koden i enlighet med rekommendationerna i det här avsnittet, eftersom de kan komma att tas bort i framtida versioner.

8.1 Skriptsatser som inte rekommenderas i Qlik Sense

Den här tabellen innehåller skriptsatser som inte rekommenderas för användning i Qlik Sense.

Skriptsatser som inte rekommenderas

Sats	Rekommendation
Command	Använd **SQL** i stället.
CustomConnect	Använd **Custom Connect** i stället.

8.2 Skriptsatsparametrar som inte rekommenderas i Qlik Sense

I den här tabellen beskrivs skriptsatsparametrar som inte rekommenderas för användning i Qlik Sense.

Skriptsatsparametrar som inte rekommenderas

Sats	Parametrar
Buffer	Använd **Incremental** i stället för: • **Inc** (rekommenderas ej) • **Incr** (rekommenderas ej)

Sats	Parametrar
LOAD	Följande parameternyckelord genereras av QlikView-filomvandlingsguider. Funktionerna behålls när data laddas in igen, men Qlik Sense tillhandahåller inte vägledning/guider för att generera satsen med dessa parametrar: • Bottom • Cellvalue • Col • Colmatch • Colsplit • Colxtr • Compound • Contain • Equal • Every • Expand • Filters • Intarray • Interpret • Length • Longer • Numerical • Pos • Remove • Rotate • Row • Rowcnd • Shorter • Start • Strcnd • Top • Transpose • Unwrap • XML: XMLSAX and Pattern is Path

8.3 Funktioner som inte rekommenderas i Qlik Sense

I den här tabellen beskrivs skript- och diagramfunktioner som inte rekommenderas för användning i Qlik Sense.

Funktioner som inte rekommenderas

Funktion	Rekommendation
NumAvg **NumCount** **NumMax** **NumMin** **NumSum**	Använd Range-funktioner i stället. *Intervallfunktioner (page 693)*
Color() **QliktechBlue** **QliktechGray**	Använd andra färgfunktioner i stället. **QliktechBlue()** kan ersättas med **RGB(8, 18, 90)** och **QliktechGray** kan ersättas med **RGB(158, 148, 137)** för att få samma färger. *Färgfunktioner (page 410)*
QlikViewVersion	Använd **EngineVersion** i stället. *EngineVersion (page 807)*
ProductVersion	Använd **EngineVersion** i stället. *EngineVersion (page 807)*
QVUser	
Year2Date	Använd **YearToDate** i stället.
Vrank	Använd **Rank** i stället.
WildMatch5	Använd **WildMatch** i stället.

Kvalificeraren ALL

I QlikView kan kvalificeraren **ALL** förekomma framför ett uttryck. Detta motsvaras av att använda **{1} TOTAL**. I ett sådant fall görs beräkningen över alla värden i fältet i dokumentet. Diagramdimensioner och aktuella urval beaktas inte. Samma värde returneras alltid oavsett det logiska tillståndet i dokumentet. Om kvalificeraren **ALL** används kommer ett set-uttryck inte att kunna användas eftersom kvalificeraren **ALL** i sig själv definierar ett set. För att kunna erbjuda bakåtkompatibilitet kommer kvalificeraren **ALL** fortfarande att fungera i denna Qlik Sense-version. Det är möjligt att den kommer att tas bort i framtida versioner.