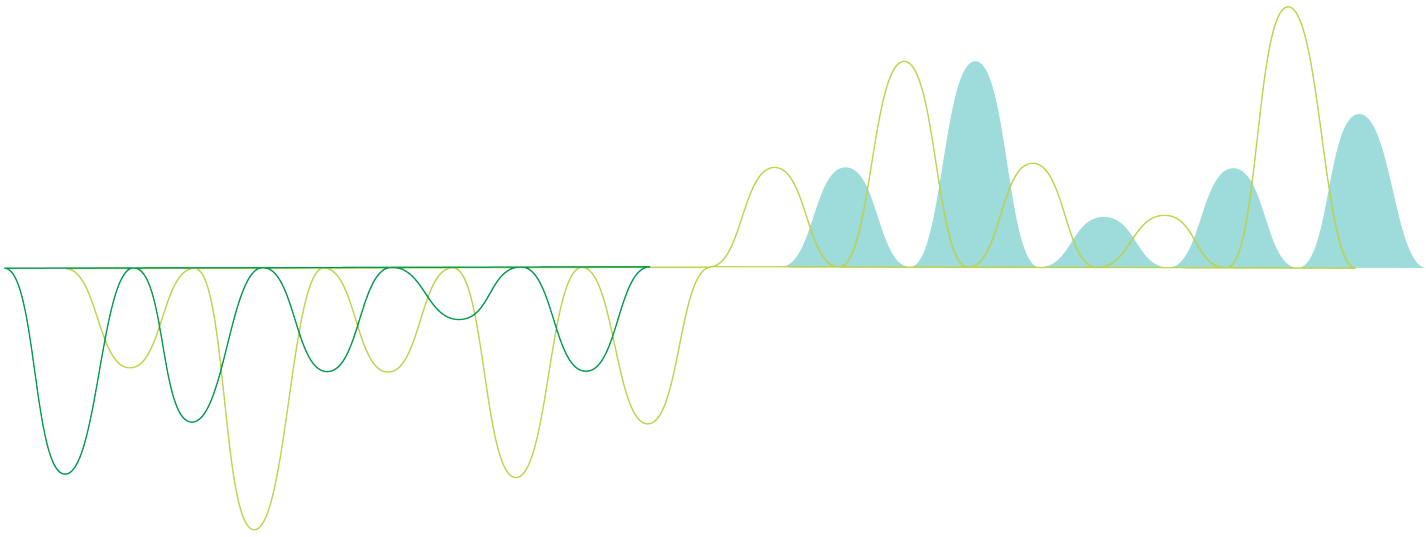
Introduktionskurs – Nästa steg inom skript

Qlik Sense®
November 2023



Contents

1 Välkommen till introduktionskursen!

Välkommen till denna introduktionskurs. Här får du lära dig mer om avancerade skript i Qlik Sense.

När du har bekantat dig med det grundläggande om skript kan du börja utföra mer avancerade operationer på dina data när du laddar dem till Qlik Sense. Det kan inkludera till exempel att omvandla data med korstabeller, rensa data och skapa och ladda data från Qlik-datafiler som kallas QVD-filer.

1.1 Det här får du lära dig

After completing this tutorial, you should be comfortable with loading data using some of the more advanced scripting functions in Qlik Sense.

1.2 Vem bör gå den här kursen?

Du bör vara bekant med grunderna för skript i Qlik Sense. Det vill säga, du har laddat data och ändrat på data med skript.

Om du inte redan har gjort det rekommenderar vi att du går igenom kursen om skript för nybörjare.

Du behöver åtkomst till Skriptredigeraren och bör ha behörighet att ladda data till Qlik Sense Enterprise on Windows.

Instruktionerna gäller även generellt för Qlik Sense Cloud Business.

1.3 Paketets innehåll

Zip-filen som du hämtat innehåller följande datafiler som du behöver för att slutföra introduktionskursen:

- *Cutlery.xlsx*
- *Data.xlsx*
- *Events.txt*
- *Employees.xlsx*
- *Intervals.txt*
- *Product.xlsx*
- *Salesman.xlsx*
- *Transactions.csv*
- *Winedistricts.txt*

Paketet innehåller även en kopia av appen *Avancerad skriptvägledning*. Ytterligare skriptavsnitt i appen innehåller skripten för de andra appar som du skapar i den här kursen. Du kan ladda upp appen till din hubb.

Vi rekommenderar att du bygger appen själv enligt anvisningarna i kursen så att du får ut så mycket som möjligt att aktiviteten. Dessutom måste du ha laddat upp och anslutit till dina datafiler enligt anvisningarna i kursen för att dataladdningarna ska fungera.

Om du stöter på problem kan appen hjälpa dig att felsöka. Vi har angett vilka skriptsegment som hör till varje lektion.

1.4 Lektioner i den här introduktionskursen

Det bör ta cirka 3–4 timmar att genomföra den här kursen, beroende på hur stor erfarenhet du har av Qlik Sense. Ämnena är avsedda att genomföras i ordning. Du kan dock göra en paus och komma tillbaka när som helst. Du slipper dessutom att göra test.

Omvandla data

Använda prefixet Crosstable

Kombinera tabeller med Join och Keep

Använda postöverskridande funktioner Peek, Previous och Exists

Matcha intervaller och iterativ inläsning

Datatvätt

Hantera hierarkiska data

QVD-filer

1.5 Vill du lära dig mer?

- Om du vill lära dig mer finns det en uppsjö av möjligheter i Qlik.
- Onlinehjälpen för Qlik är tillgänglig.
- Utbildning, inklusive kostnadsfria onlinekurser, finns i Qlik Continuous Classroom.
- Diskussionsforum, bloggar med mera finns i Qlik Community.

2 Satserna LOAD och SELECT

Du kan ladda data i Qlik Sense med hjälp av satserna LOAD och SELECT. Varje sats genererar en intern tabell. LOAD används för att ladda data från filer och SELECT används för att ladda data från databaser.

I den här introduktionskursen kommer du att använda data från filer, så du kommer att använda LOAD-satser.

Du kan också använda en föregående LOAD om du vill kunna ändra på innehållet i laddade data. Byta namn på fält måste du till exempel göra i en LOAD-sats. SELECT-satser tillåter inte ändringar av fältnamn.

Följande regler gäller när du laddar data i Qlik Sense:

- Qlik Sense gör ingen skillnad mellan tabeller som genererats med en LOAD-sats och tabeller som genererats med en SELECT-sats. Det spelar därför ingen roll om tabellerna, när flera tabeller laddas, laddas med LOAD- eller SELECT-satser eller en kombination av de båda.
- Fältens ordning i satsen eller i den ursprungliga tabellen i databasen är inte viktig för Qlik Senses logik.
- Fältnamn är skiftlägeskänsliga och de används för att skapa associationer mellan datatabeller. Därför är det ibland nödvändigt att byta namn på fält i laddningsskriptet för att få en önskad datamodell.

3 Omvandla data

Du kan omvandla och ändra på data i Skriptredigeraren innan du använder dem i din app.

En av fördelarna med att ändra på data är att du kan välja att bara ladda en underuppsättning av data från en fil, t.ex. ett fåtal valda kolumner från en tabell, för att göra datahanteringen mer effektiv. Du kan också ladda data mer än en gång för att dela upp rådata i flera nya logiska tabeller. Du kan ladda data från fler än en källa och slå ihop den till en tabell i Qlik Sense.

Följande övningar visar hur du laddar data med prefixet Crosstable. Du får även lära dig att koppla samman tabeller, använda postöverskridande funktioner som Peek och Previous samt ladda samma rad flera gånger med hjälp av While Load.

3.1 Använda prefixet Crosstable

Korstabeller är en vanlig tabelltyp, som består av en datamatris med värden mellan två ortogonala listor av rubrikdata. När du har en korstabell med data kan du använda prefixet Crosstable för att omvandla data och skapa önskade fält.

Prefixet Crosstable

I följande *Product*-tabell har du en kolumn per månad och en rad per produkt.

Produkttabell

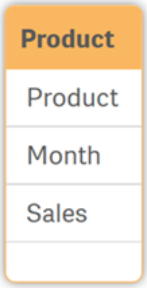
Produkt	Jan 2014	Feb 2014	Mars 2014	Apr 2014	Maj 2014	Jun 2014
A	100	98	100	83	103	82
B	284	279	297	305	294	292
C	50	53	50	54	49	51

När du laddar tabellen får du en tabell med ett fält för *Product* och ett fält för varje månad.

Product-tabellen med fältet Product och ett fält för varje månad

Om du vill analysera dessa data är det mycket enklare att ha alla tal i ett fält och alla månader i ett annat. I detta fall handlar det om en tabell med tre kolumner och en kolumn för varje kategori (*Product, Month, Sales*).

Product tabell med fälten Product, Month och Sales



Crosstable-prefixet omvandlar data till en tabell med en kolumn för *Month* och en annan för *Sales*. Ett annat sätt att uttrycka det på är att säga att det tar fältnamn och omvandlar dem till fältvärden.

Gör följande:

1. Skapa en ny app och kalla den för *Introduktionskurs om avancerade skript*.
2. Lägg till ett nytt skriptavsnitt i **Skriptredigeraren**.
3. Ge delavsnittet namnet *Product*.
4. Klicka på **Välj data** under **AttachedFiles** i menyn till höger.
5. Ladda upp och välj sedan *Product.xlsx*.
6. Välj tabellen *Product* i fönstret **Välj data från**.

   > *Se till att **Inbäddade fältnamn** under **Fältnamn** har valts för att inkludera namnen på tabellfälten när du laddar data.*

7. Klicka på **Infoga skript**.

   Ditt skript bör se ut så här:

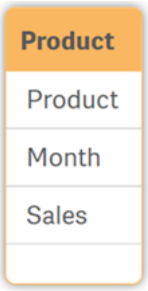
   ```
   LOAD
       Product,
       "Jan 2014",
       "Feb 2014",
       "Mar 2014",
       "Apr 2014",
       "May 2014",
       "Jun 2014"
   FROM [lib://AttachedFiles/Product.xlsx]
   (ooxml, embedded labels, table is Product);
   ```

8. Klicka på **Ladda data**.
9. Öppna **datamodellvyn**. Datamodellen ser ut så här:

Product-tabellen med fältet Product och ett fält för varje månad

10. Klicka på fliken *Product* i **Skriptredigeraren**.
11. Ange följande över LOAD-satsen:
    `CrossTable(Month, Sales)`
12. Klicka på **Ladda data**.
13. Öppna **datamodellvyn**. Datamodellen ser ut så här:

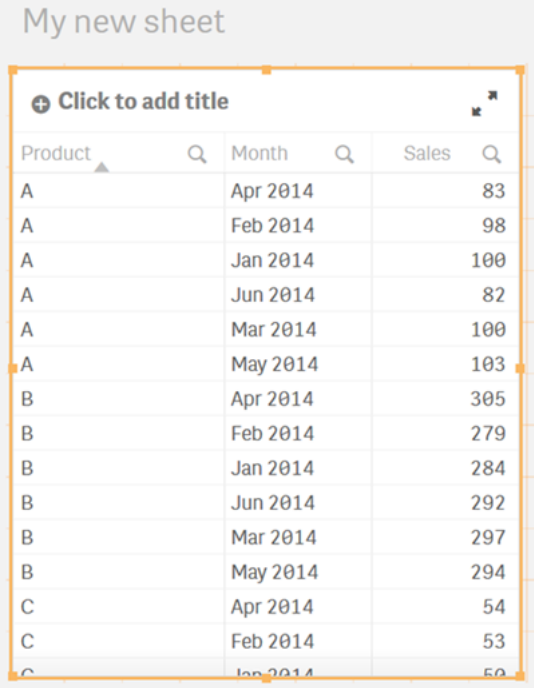
    Product tabell med fälten Product, Month och Sales



Observera att indata normalt bara har en kolumn som kvalificerande fält; som en intern nyckel (*Product* i exemplet ovan). Det går dock att ha flera. I sådana fall måste alla kvalificerande fält räknas upp före attributfälten i LOAD-satsen, och Crosstable-prefixets tredje parameter måste användas för att definiera antalet kvalificerande fält. Det går inte att ha en föregående LOAD eller ett prefix framför Crosstable-nyckelordet. Det går dock att använda automatisk konkatenering.

I en tabell i Qlik Sense ser dina data ut så här:

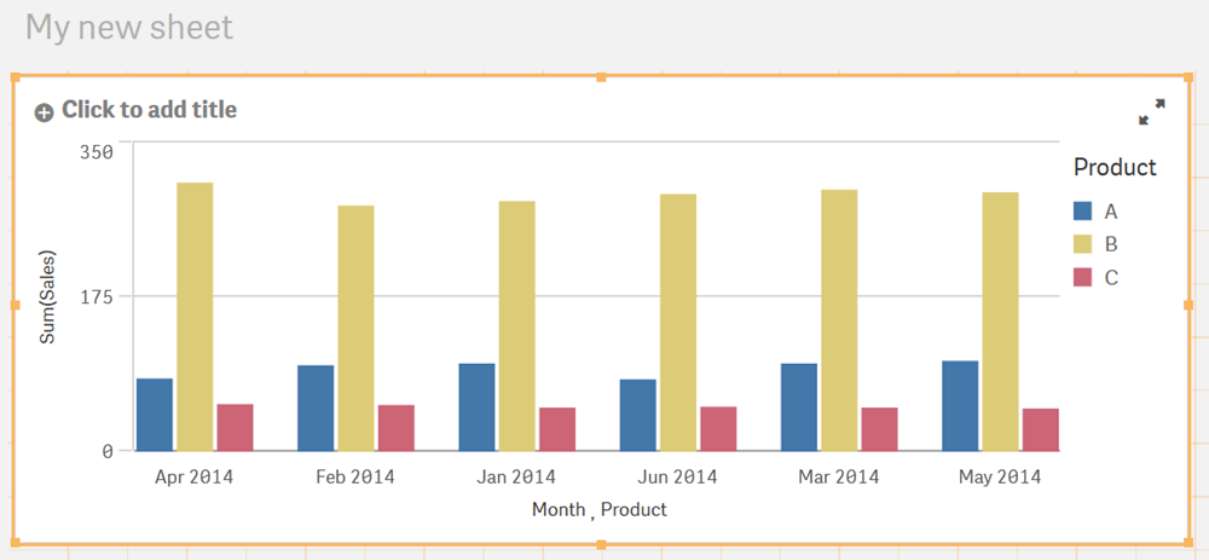
Tabellen visar data som har laddats med prefixet Crosstable



Product	Month	Sales
A	Apr 2014	83
A	Feb 2014	98
A	Jan 2014	100
A	Jun 2014	82
A	Mar 2014	100
A	May 2014	103
B	Apr 2014	305
B	Feb 2014	279
B	Jan 2014	284
B	Jun 2014	292
B	Mar 2014	297
B	May 2014	294
C	Apr 2014	54
C	Feb 2014	53

Nu kan du till exempel skapa ett stapeldiagram med hjälp av dessa data:

Stapeldiagrammet visar data som laddats med Crosstable-prefixet



Mer information om Crosstable finns i det här blogginlägget i Qlik Community: Korstabelladdning. Beteenden diskuteras i ett sammanhang som avser QlikView. Logiken gäller dock i samma utsträckning för Qlik Sense.

Numerisk tolkning fungerar inte för attributfälten. Detta innebär att om du har månader som kolumnrubriker kommer dessa inte att tolkas automatiskt. Du kan avhjälpa problemet genom att använda Crosstable-prefixet för att skapa en tillfällig tabell och köra ett andra pass på den så att tolkningarna blir som i följande exempel.

Observera att detta bara är ett exempel. Det finns inga övningar för dessa data att göra i Qlik Sense.

```
tmpData:
Crosstable (MonthText, Sales)
LOAD Product, [Jan 2014], [Feb 2014], [Mar 2014], [Apr 2014], [May 2014], [Jun 2014]
FROM ...

Final:
LOAD Product,
Date(Date#(MonthText,'MMM YYYY'),'MMM YYYY') as Month,
Sales
Resident tmpData;
Drop Table tmpData;
```

Rensa minnescachen

Du kan ta bort tabeller som du skapar för att rensa cacheminnet. Om du läser in till en tillfällig tabell som i föregående avsnitt bör du ta bort den när den inte behövs längre. Exempel:

```
DROP TABLE Table1, Table2, Table3, Table4;
DROP TABLES Table1, Table2, Table3, Table4;
```

Du kan även släppa fält. Exempel:

```
DROP FIELD Field1, Field2, Field3, Field4;
DROP FIELDS Field1, Field2, Field3, Field4;
DROP FIELD Field1 from Table1;
DROP FIELDS Field1 from Table1;
```

Som du kan se kan nyckelorden TABLE och FIELD vara singular eller plural.

3.2 Kombinera tabeller med Join och Keep

En operation som slår samman två tabeller till en. Posterna i den skapade tabellen är kombinationer av posterna i de ursprungliga tabellerna, vanligtvis på så sätt att de två poster som bidrar till någon given kombination i den skapade tabellen har ett gemensamt värde för ett eller flera gemensamma fält – en så kallad natural join. I Qlik Sense kan joins göras i skriptet. Resultatet blir en logisk tabell.

Det är möjligt att länka tabeller redan i skriptet. Qlik Sense-logiken ser då inte de separata tabellerna, utan bara den interna tabell som länkningen resulterat i. Ibland kan detta vara nödvändigt, men det finns nackdelar:

- De inlästa tabellerna blir ofta större och Qlik Sense arbetar långsammare.
- En del information kan gå förlorad; det går ofta inte längre att få frekvensuppgifter (antal poster) om den ursprungliga tabellen.

Keep-funktionen, som reducerar den ena tabellen eller båda tabellerna till det gemensamma snittet av tabelldata innan tabellerna sparas i Qlik Sense, har utvecklats för att minska antalet fall där man måste använda funktionen explicit join.

I denna dokumentation används vanligen termen join för att beteckna länkningar som gjordes innan de interna tabellerna skapades. Associationen, som görs när de interna tabellerna har skapats, är emellertid i allt väsentligt också en länkning (join).

Join

En länkning skapas enklast med hjälp av Join-prefixet i skriptet, som länkar samman den interna tabellen med en annan namngiven tabell eller med den senast skapade tabellen. Länkningen blir en outer join, som skapar alla möjliga kombinationer av värden från två tabeller.

Exempel:

```
LOAD  a, b, c from table1.csv;
join LOAD a, d from table2.csv;
```

Den interna tabell som blir resultatet får fälten a, b, c och d. Antalet poster varierar beroende på fältvärdena i de två tabellerna.

Fälten som länkar tabellerna måste ha identiska namn. Hur många fält man använder för länkningen spelar ingen roll. Vanligen ska tabellerna ha ett eller flera fält gemensamt. Om det inte finns något gemensamt fält kommer länkningen att resultera i den kartesiska produkten av tabellerna. Det är också möjligt att länka tabeller där alla fält är gemensamma, men det finns oftast ingen mening med det. Join-prefixet använder sig av den senast skapade interna tabellen såvida inte namnet på en tidigare inläst tabell har angetts i Join-satsen. Ordningen på de två satserna är således inte godtycklig.

Använda Join

Det explicita Join prefixet i Qlik Sense-skriptspråket skapar en fullständig länkning (full join) mellan de båda tabellerna. Resultatet är en tabell. Sådana länkningar ger ofta mycket stora tabeller.

Gör följande:

1. Öppna appen *Avancerad skriptvägledning*.
2. Lägg till ett nytt skriptavsnitt i **Skriptredigeraren**.
3. Kalla avsnittet *Transactions*.
4. Klicka på **Välj data** under **AttachedFiles** i menyn till höger.
5. Ladda upp och välj sedan *Transactions.csv*.

   *Se till att **Inbäddade fältnamn** under **Fältnamn** har valts för att inkludera namnen på tabellfälten när du laddar data.*
6. Klicka på **Infoga skript** i fönstret **Välj data från**.
7. Ladda upp och välj sedan *Salesman.xlsx*.

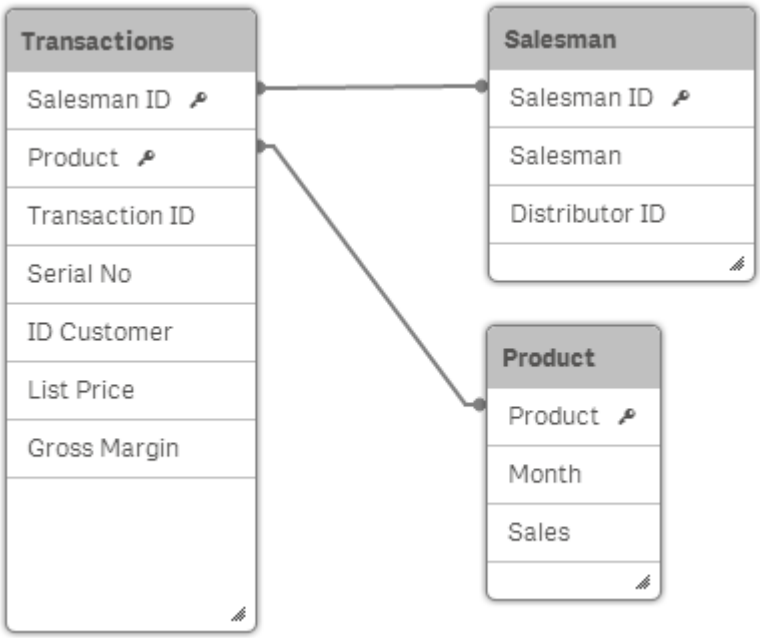
8. Klicka på **Infoga skript** i fönstret **Välj data från**.

   Ditt skript bör se ut så här:

```
LOAD
    "Transaction ID",
    "Salesman ID",
    Product,
    "Serial No",
    "ID Customer",
    "List Price",
    "Gross Margin"
FROM [lib://AttachedFiles/Transactions.csv]
(txt, codepage is 28591, embedded labels, delimiter is ',', msq);

LOAD
    "Salesman ID",
    Salesman,
    "Distributor ID"
FROM [lib://AttachedFiles/Salesman.xlsx]
(ooxml, embedded labels, table is Salesman);
```

9. Klicka på **Ladda data**.
10. Öppna **datamodellvyn**. Datamodellen ser ut så här:

    Datamodell: Tabellerna Transactions, Salesman och Product



Att hålla isär tabellerna *Transactions* och *Salesman* ger kanske inte det resultat som krävs. Det kan vara bättre att slå samman de två tabellerna.

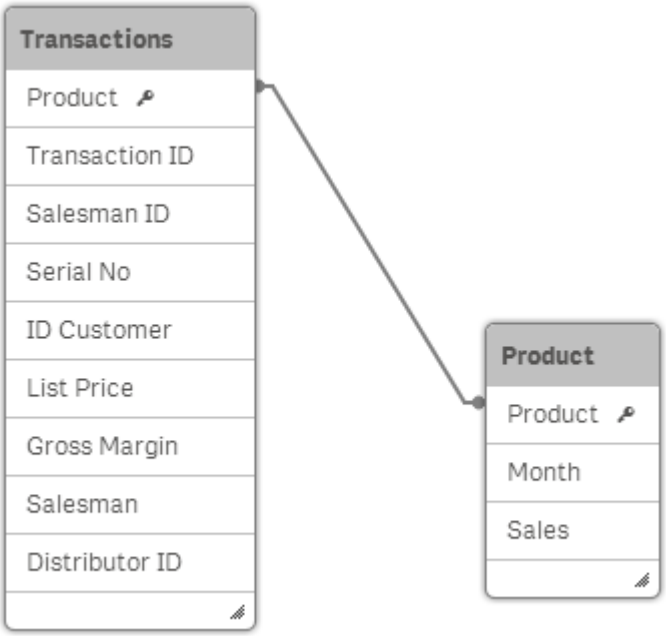
Gör följande:

1. Ge den sammanslagna tabellen ett namn genom att lägga till följande rad ovanför den första LOAD-satsen:
   ```
   Transactions:
   ```
2. Om du vill koppla ihop tabellerna *Transactions* och *Salesman* lägger du till följande rad ovanför den andra LOAD-satsen:
   ```
   Join(Transactions)
   ```

   Ditt skript bör se ut så här:

   ```
   Transactions:
   LOAD
       "Transaction ID",
       "Salesman ID",
       Product,
       "Serial No",
       "ID Customer",
       "List Price",
       "Gross Margin"
   FROM [lib://AttachedFiles/Transactions.csv]
   (txt, codepage is 28591, embedded labels, delimiter is ',', msq);

   Join(Transactions)
   LOAD
       "Salesman ID",
       Salesman,
       "Distributor ID"
   FROM [lib://AttachedFiles/Salesman.xlsx]
   (ooxml, embedded labels, table is Salesman);
   ```
3. Klicka på **Ladda data**.
4. Öppna **datamodellvyn**. Datamodellen ser ut så här:

Datamodell:Tabellerna Transactions och Product



Alla fälten i tabellerna *Transactions* och *Salesman* har nu kombinerats till en enda *Transactions*-tabell.

Mer information om när Join ska användas finns i följande blogginlägg i Qlik Community: To Join or not to Join *(Använda länkning eller inte) och* Mapping as an Alternative to Joining *(Mappning som ett alternativ till länkning). Beteenden diskuteras i ett sammanhang som avser QlikView. Logiken gäller dock i samma utsträckning för Qlik Sense.*

Keep

En av finesserna med Qlik Sense är just att programmet gör associationer mellan tabeller istället för att skapa länkningar, vilket minskar utrymmet i minnet, ökar hastigheten och ger större flexibilitet. Keep-funktionaliteten har utformats för att minska antalet fall där explicit länkning måste användas.

När prefixet Keep används mellan två LOAD- eller SELECT-satser reduceras den ena tabellen eller båda tabellerna till snittet av tabelldata innan de lagras i Qlik Sense. Prefixet Keep måste alltid inledas med något av följande nyckelord: Inner, Left eller Right. Urvalet av poster från tabellerna görs på samma sätt som i en motsvarande koppling. Skillnaden är att tabellerna inte länkas utan lagras i Qlik Sense som två tabeller med olika namn.

Inner

Prefixen Join och Keep i dataladdningsskriptet kan föregås av prefixet Inner.

Om det används framför Join, anger det att länkningen mellan de båda tabellerna ska vara en inre länkning (inner join). Den tabell som blir resultatet innehåller enbart kombination mellan de två tabellerna med en fullständig datauppsättning från båda sidor.

Om de används framför Keep, anger det att tabellerna ska reduceras till det gemensamma snittet av deras datamängder innan de lagras i Qlik Sense.

Exempel:

I de här exemplen använder vi källtabellerna *Table1* och *Table2*.

Observera att det här bara är exempel. Det finns inga övningar för dessa data att göra i Qlik Sense.

Table 1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Inner Join

Först utför vi en Inner Join på tabellerna. Det resulterar i att *VTable* bara innehåller en rad, den enda posten som förekommer i båda tabellerna, med kombinerade data från båda tabellerna.

```
VTable:
SELECT * from Table1;
inner join SELECT * from Table2;
```

VTable

A	B	C
1	aa	xx

Inner Keep

Om vi i stället utför en Inner Keep har vi fortfarande två tabeller. De två tabellerna associeras via det gemensamma fältet *A*.

```
VTab1:
SELECT * from Table1;
VTab2:
inner keep SELECT * from Table2;
```

VTab1

A	B
1	aa

VTab2

A	C
1	xx

Left

Prefixen Join och Keep i dataladdningsskriptet kan föregås av prefixet left.

Om det används framför Join, anger det att länkningen mellan de båda tabellerna ska vara vänsterriktad (left join). En sådan länkning resulterar i en tabell som endast innehåller kombinationer där datauppsättningen från den första tabellen är fullständig.

Om det används framför Keep, anger det att den andra tabellen ska reduceras till det gemensamma snittet med den första tabellen innan den lagras i Qlik Sense.

Exempel:

I de här exemplen använder vi källtabellerna *Table1* och *Table2*.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Först utför vi en Left Join på tabellerna. Resultatet blir att *VTable* innehåller alla rader från *Table1*, kombinerade med fält från matchande rader i *Table2*.

```
VTable:
SELECT * from Table1;
left join SELECT * from Table2;
```

VTable

A	B	C
1	aa	xx
2	cc	-
3	ee	-

Om vi i stället utför en Left Keep har vi fortfarande två tabeller. De två tabellerna associeras via det gemensamma fältet *A*.

```
VTab1:
SELECT * from Table1;
VTab2:
left keep SELECT * from Table2;
```

VTab1

A	B
1	aa
2	cc
3	ee

VTab2

A	C
1	xx

Right

Prefixen Join och Keep i Qlik Sense-skriptspråket kan föregås av prefixet right.

Om det används framför Join, anger det att länkningen mellan de båda tabellerna ska vara högerriktad (right join). En sådan länkning resulterar i en tabell som endast innehåller kombinationer där datamängden från den andra tabellen är fullständig.

Om det används framför Keep, anger det att den första tabellen ska reduceras till det gemensamma snittet med den andra tabellen innan den lagras i Qlik Sense.

Exempel:

I de här exemplen använder vi källtabellerna *Table1* och *Table2*.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Först utför vi en Right Join på tabellerna. Resultatet blir att *VTable* innehåller alla rader från *Table2*, kombinerade med fält från matchande rader i *Table1*.

```
VTable:
SELECT * from Table1;
right join SELECT * from Table2;
```

VTable

A	B	C
1	aa	xx
4	-	yy

Om vi i stället utför en Right Keep har vi fortfarande två tabeller. De två tabellerna associeras via det gemensamma fältet *A*.

```
VTab1:
SELECT * from Table1;
VTab2:
right keep SELECT * from Table2;
```

VTab1

A	B
1	aa

VTab2

A	C
1	xx
4	yy

3.3 Använda postöverskridande funktioner Peek, Previous och Exists

Dessa funktioner används när ett värde från tidigare inlästa poster behövs för utvärdering av den aktuella posten.

I den här delen av introduktionskursen ska vi undersöka funktionerna Peek(), Previous() och Exists().

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.

Syntax:
```
Peek(fieldname [ , row [ , tablename ] ] )
```

Raden måste vara ett heltal. 0 anger första posten, 1 andra posten osv. Negativa tal markerar ordningen från slutet av tabellen. -1 anger den senaste lästa posten.

Om ingen rad angivits, antas -1.

Tablename är en tabelletikett utan avslutande kolon. Om inget *tablename* har angetts antas den aktuella tabellen. Om det används utanför **LOAD**-satsen eller refererar till en annan tabell, måste *tablename* anges.

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(expression)
```

Previous()-funktionen kan nästlas om man vill få tillgång till poster som ligger flera steg bakåt. 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.

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.

Syntax:
```
Exists(field [, expression ] )
```

Fältet måste finnas i de data som laddats hittills av skriptet. *Expression* är ett uttryck som utvärderar fältvärdet som ska hittas i den angivna fältet. Om uttrycket utelämnas, antas den aktuella postens värde i det angivna fältet.

Använda Peek() och Previous()

I sin enklaste form används Peek() och Previous() för att identifiera specifika värden i en tabell. Här är ett exempel på data i tabellen *Employees* som du kommer att ladda i den här övningen.

Exempel på data från tabellen Employees

Datum	Anställd	Slutade
1/1/2011	6	0

Datum	Anställd	Slutade
2/1/2011	4	2
3/1/2011	6	1
4/1/2011	5	2

För närvarande hämtar den endast data för månad, anställningar och uppsägningar. Vi ska nu lägga till fält för *Employee Count* och *Employee Var* med hjälp av funktionerna Peek() och Previous(), för att se den månatliga skillnaden i det totala antalet medarbetare.

Gör följande:

1. Öppna appen *Avancerad skriptvägledning*.
2. Lägg till ett nytt skriptavsnitt i **Skriptredigeraren**.
3. Kalla delavsnittet *Employees*.
4. Klicka på **Välj data** under **AttachedFiles** i menyn till höger.
5. Ladda upp och välj sedan *Employees.xlsx*.

   Se till att Embedded field names under Field names har valts för att inkludera namnen på tabellfälten när du laddar data.

6. Klicka på **Infoga skript** i fönstret **Välj data från**.

   Ditt skript bör se ut så här:

   ```
   LOAD
       "Date",
       Hired,
       Terminated
   FROM [lib://AttachedFiles/Employees.xlsx]
   (ooxml, embedded labels, table is Sheet1);
   ```

7. Ändra skriptet så att det ser ut så här:

   ```
   [Employees Init]:
   LOAD
       rowno() as Row,
       Date(Date) as Date,
       Hired,
       Terminated,
       If(rowno()=1, Hired-Terminated,  peek([Employee Count], -1)+(Hired-Terminated)) as
   [Employee Count]
   FROM [lib://AttachedFiles/Employees.xlsx]
   (ooxml, embedded labels, table is Sheet1);
   ```

   Datumen i fältet *Date* i arket Excel har formatet MM/DD/ÅÅÅÅ. Date-funktionen tillämpas på *Date*-fältet för att säkerställa att datumen tolkas rätt med hjälp av systemvariablernas format.

Peek()-funktionen gör att du kan identifiera alla värden som lästs in för ett definierat fält. Vi börjar med att titta om rowno() är lika med 1 i uttrycket. Om det är lika med 1, finns inte *Employee Count*, därför fyller vi i fältet med skillnaden mellan *Hired* minus *Terminated*.

Om rowno() är större än 1, så tittar vi på förra månadens *Employee Count* och lägger till det talet till skillnaden mellan den månadens *Hired*-medarbetare minus *Terminated*-medarbetare.

Observera även att vi använder (-1) i funktionen Peek(). Det gör att Qlik Sense använder posten ovanför den aktuella posten. Om (-1) inte anges kommer Qlik Sense att visa den föregående posten.

8. Lägg till följande i slutet av skriptet:

```
[Employee Count]:
LOAD
       Row,
       Date,
       Hired,
       Terminated,
       [Employee Count],
       If(rowno()=1,0,[Employee Count]-Previous([Employee Count])) as [Employee Var]
Resident [Employees Init] Order By Row asc;
Drop Table [Employees Init];
```

Previous()-funktionen gör att du kan identifiera det senaste värdet som lästs in för ett definierat fält. Vi börjar med att titta om rowno() är lika med 1 i uttrycket. Om det är lika med 1 vet vi att Employee Var inte finns eftersom det inte finns någon post för föregående månads *Employee Count*. Vi anger helt enkelt 0 för värdet.

Om rowno() är större än 1 vet vi att det kommer att finnas en *Employee Var*, så vi tittar på den föregående månadens *Employee Count* och subtraherar det talet från den aktuella månadens *Employee Count* för att få värdet i *Employee Var*-fältet.

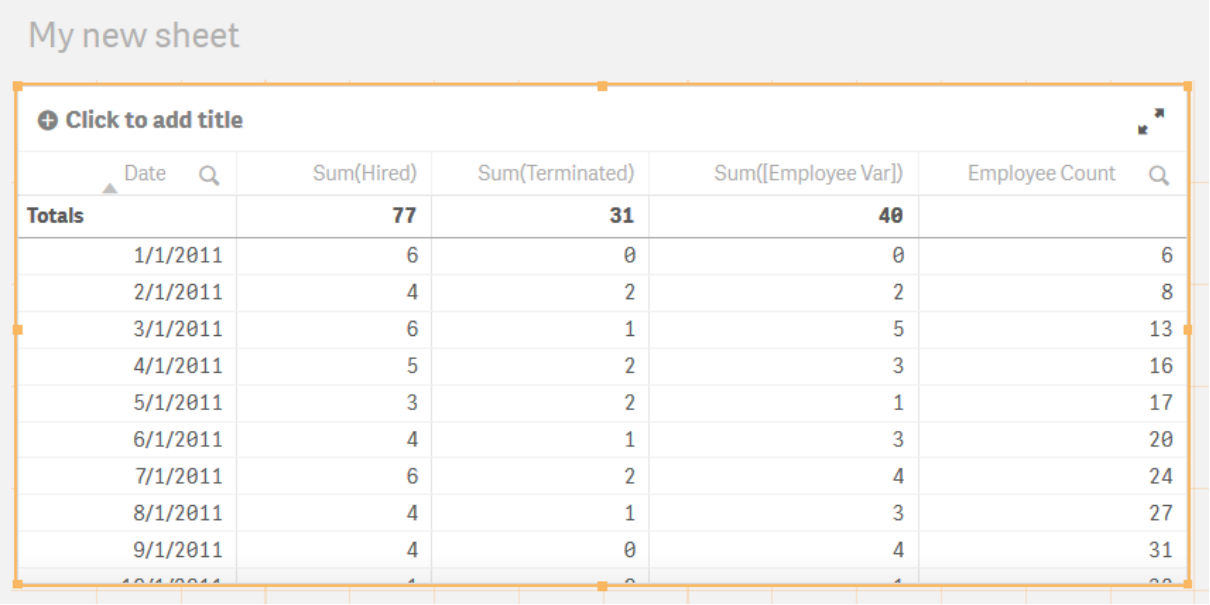
Ditt skript bör se ut så här:

```
[Employees Init]:
LOAD
    rowno() as Row,
    Date(Date) as Date,
    Hired,
    Terminated,
    If(rowno()=1, Hired-Terminated,  peek([Employee Count], -1)+(Hired-Terminated)) as
[Employee Count]
FROM [lib://AttachedFiles/Employees.xlsx]
(ooxml, embedded labels, table is Sheet1);

[Employee Count]:
LOAD
    Row,
    Date,
    Hired,
    Terminated,
    [Employee Count],
    If(rowno()=1,0,[Employee Count]-Previous([Employee Count])) as [Employee Var]
Resident [Employees Init] Order By Row asc;
```

```
Drop Table [Employees Init];
```

9. Klicka på **Ladda data**.
   Skapa en tabell i ett nytt ark i appöversikten med hjälp av *Date, Hired, Terminated, Employee Count* och *Employee Var* som kolumnerna i tabellen. Den resulterande tabellen bör se ut så här:

   Tabell efter användning av Peek och Previous i skript



Date	Sum(Hired)	Sum(Terminated)	Sum([Employee Var])	Employee Count
Totals	**77**	**31**	**40**	
1/1/2011	6	0	0	6
2/1/2011	4	2	2	8
3/1/2011	6	1	5	13
4/1/2011	5	2	3	16
5/1/2011	3	2	1	17
6/1/2011	4	1	3	20
7/1/2011	6	2	4	24
8/1/2011	4	1	3	27
9/1/2011	4	0	4	31

Peek() och Previous() gör att du kan rikta in dig på definierade rader i en tabell. Den största skillnaden mellan de två funktionerna är att med Peek() funktionen kan användaren titta på ett fält som inte har lästs in i skriptet tidigare, medan användaren med Previous()-funktionen endast kan titta på fält som redan har lästs in. Previous() använder LOAD-satsens indata medan Peek() använder LOAD-satsens utdata. (Samma som skillnaden mellan RecNo() och RowNo().) Det innebär att de två funktionerna kommer att bete sig olika om du har en Where-sats.

Funktionen Previous() skulle alltså vara bättre för att visa aktuellt värde jämfört med föregående värde. I exemplet beräknade vi variansen för medarbetare från månad till månad.

Funktionen Peek() skulle vara bättre för ett fält som inte har laddats in i tabellen tidigare eller för att fokusera på en specifik rad. Detta visades i exemplet när vi beräknade *Employee Count* genom att titta på föregående månads *Employee Count* och sedan lade till skillnaden mellan medarbetare som anställdes och medarbetare som slutade för den aktuella månaden. Kom ihåg att *Employee Count* inte var ett fält i den ursprungliga filen.

Mer information om när Peek() och Previous() används finns i det här blogginlägget i Qlik Community: Peek() vs Previous() – When to Use Each. *Beteenden diskuteras i ett sammanhang som avser QlikView. Logiken gäller dock i samma utsträckning för Qlik Sense.*

Använda Exists()

Exists()-funktionen används ofta med Where-satsen i skriptet för att ladda data, om relaterade data redan har laddats i datamodellen.

I följande exempel använder vi även funktionen Dual() för att tilldela numeriska värden till strängar.

Gör följande:

1. Skapa en ny app och ge den ett namn.
2. Lägg till ett nytt skriptavsnitt i **Skriptredigeraren**.
3. Kalla delavsnittet *People*.
4. Ange följande skript:

```
//Add dummy people data
PeopleTemp:
LOAD * INLINE [
PersonID, Person
1, Jane
2, Joe
3, Shawn
4, Sue
5, Frank
6, Mike
7, Gloria
8, Mary
9, Steven,
10, Bill
];

//Add dummy age data
AgeTemp:
LOAD * INLINE [
PersonID, Age
1, 23
2, 45
3, 43
4, 30
5, 40
6, 32
7, 45
8, 54
9,
10, 61
11, 21
12, 39
];

//LOAD new table with people
People:
NoConcatenate LOAD
    PersonID,
```

```
    Person
Resident PeopleTemp;

Drop Table PeopleTemp;

//Add age and age bucket fields to the People table
Left Join (People)
LOAD
    PersonID,
    Age,
       If(IsNull(Age) or Age='', Dual('No age', 5),
        If(Age<25, Dual('Under 25', 1),
         If(Age>=25 and Age <35, Dual('25-34', 2),
          If(Age>=35 and Age<50, Dual('35-49' , 3),
           If(Age>=50, Dual('50 or over', 4)
            ))))) as AgeBucket
Resident AgeTemp
Where Exists(PersonID);

DROP Table AgeTemp;
```

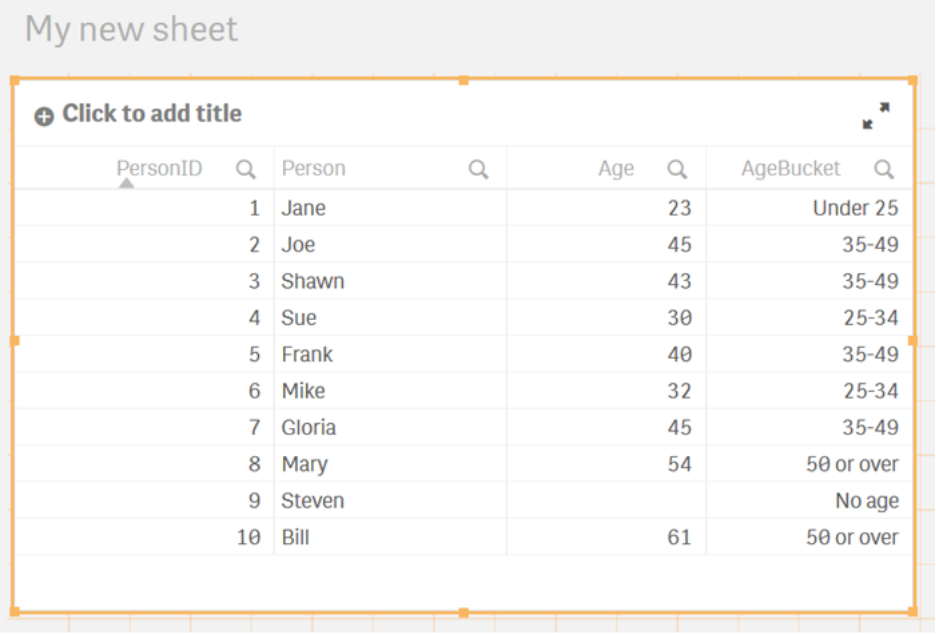
5. Klicka på **Ladda data**.

   I skriptet läses fälten *Age*och *AgeBucket* in endast om *PersonID* redan har lästs in i datamodellen.

   Observera i *AgeTemp*-tabellen att det finns åldrar angivna för *PersonID* 11 och 12, men eftersom dessa ID:n inte har laddats in i datamodellen (i tabellen *People*) utesluts de på grund av Where Exists (PersonID)-satsen. Den här satsen kan även skrivas så här: Where Exists(PersonID, PersonID).

   Skriptets utdata ser ut så här:

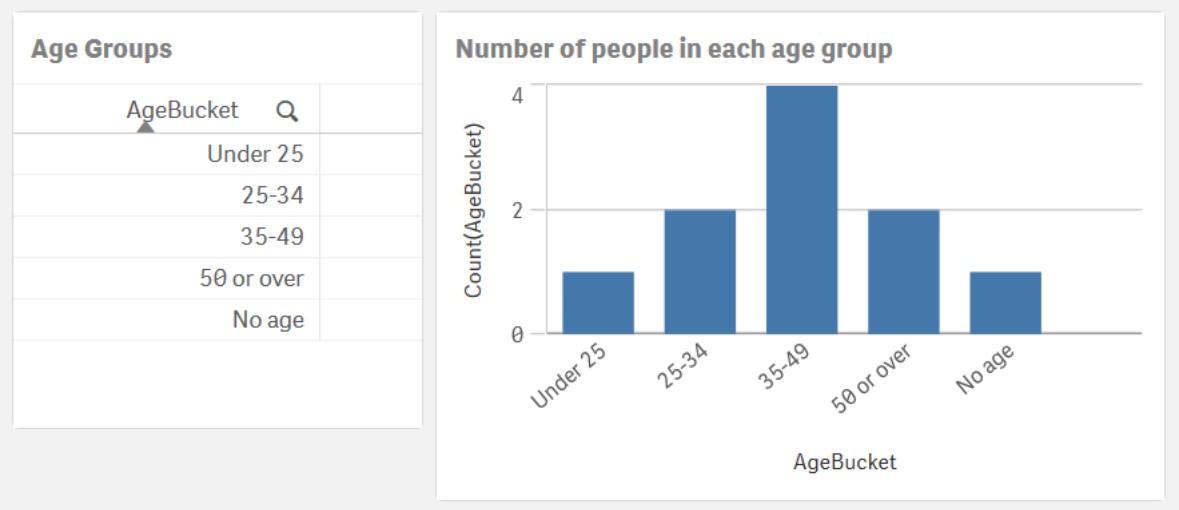
   Tabell efter användning av Exists i skript



PersonID	Person	Age	AgeBucket
1	Jane	23	Under 25
2	Joe	45	35-49
3	Shawn	43	35-49
4	Sue	30	25-34
5	Frank	40	35-49
6	Mike	32	25-34
7	Gloria	45	35-49
8	Mary	54	50 or over
9	Steven		No age
10	Bill	61	50 or over

Om inga *PersonID* i *AgeTemp*-tabellen hade lästs in i datamodellen skulle inte fälten *Age*och *AgeBucket* ha kopplats till tabellen *People*. Du kan använda funktionen Exists() för att undvika överblivna poster/data i datamodellen, det vill säga *Age*- och *AgeBucket*-fält som inte har några personer associerade.

6. Skapa ett nytt ark och ge det ett namn.
7. Öppna det nya arket och klicka på **Redigera ark**.
8. Lägg till en standardtabell till arket med dimensionen *AgeBucket* och kalla visualiseringen *Åldersgrupper*.
9. Lägg till ett stapeldiagram till arket med dimensionen *AgeBucket* och måttet *Count([AgeBucket])*. Ge visualiseringen namnet *Number of people in each age group*.
10. Justera egenskaperna för tabellen och stapeldiagrammet som du vill ha dem och klicka på **Klart**. Arket bör se ut ungefär så här:
Ark med grupperingar efter ålder



Dual()-funktionen är mycket användbar i skriptet eller i ett diagramuttryck när du behöver tilldela en sträng ett numeriskt värde.

I skriptet har du en applikation som läser in åldrar och du har bestämt dig för att lägga de åldrarna i buckets så att du kan skapa visualiseringar baserade på buckets för åldrar jämfört med verkliga åldrar. Det finns en bucket för personer under 25, en för personer mellan 25 och 35 och så vidare. Med hjälp av Dual()-funktionen kan buckets för ålder tilldelas numeriska värden som sedan kan användas för att sortera dessa buckets i en listbox eller ett diagram. Så precis som i apparket placeras ”No age” sist i listan.

Mer information om Exists() och Dual() finns i det här blogginlägget i Qlik Community: Dual & Exists – användbara funktioner

3.4 Matcha intervaller och iterativ inläsning

Prefixet Intervalmatch framför en LOAD- eller SELECT-sats används för att länka diskreta numeriska värden till ett eller flera numeriska intervall. Detta är en mycket kraftfull funktion, som kan användas i exempelvis produktionsmiljö.

Använda prefixet IntervalMatch()

Den mest grundläggande intervallmatchningen är när du har en lista med tal eller datum (händelser) i en tabell och en lista över intervall i en andra tabell. Målet är att länka de två tabellerna. Oftast är detta ett många-till-många-förhållande, det vill säga att ett intervall kan ha många datum som hör till det och ett datum kan höra till många intervall. För att lösa detta måste du skapa en kopplingstabell mellan de två ursprungliga tabellerna. Detta kan du göra på flera sätt.

Det enklaste sättet att lösa det här problemet i Qlik Sense är att använda prefixet IntervalMatch() framför antingen en LOAD- eller en SELECT-sats. LOAD/SELECT-satsen får bara innehålla två fält, nämligen fälten From och To som anger intervallen. Prefixet IntervalMatch() genererar då alla kombinationer mellan de inlästa intervallen och ett tidigare inläst numeriskt fält, som angetts som parameter för prefixet.

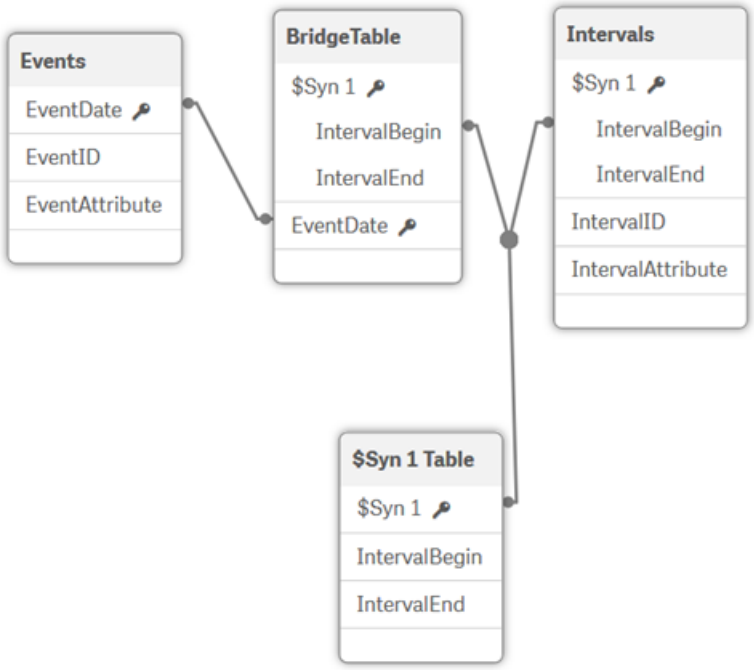
Gör följande:

1. Skapa en ny app och ge den ett namn.
2. Lägg till ett nytt skriptavsnitt i **Skriptredigeraren**.
3. Kalla avsnitten *Events*.
4. Klicka på **Välj data** under **AttachedFiles** i menyn till höger.
5. Ladda upp och välj sedan *Events.txt*.
6. Klicka på **Infoga skript** i fönstret **Välj data från**.
7. Ladda upp och välj sedan *Intervals.txt*.
8. Klicka på **Infoga skript** i fönstret **Välj data från**.
9. I skriptet ger du den första tabellen namnet *Events*, och den andra tabellen namnet *Intervals*.
10. Lägg till en IntervalMatch-sats i slutet av skriptet för att skapa en tredje tabell som kopplar samman de två första tabellerna:
    ```
    BridgeTable:
    IntervalMatch (EventDate)
    LOAD distinct IntervalBegin, IntervalEnd
    Resident Intervals;
    ```
11. Ditt skript bör se ut så här:
    ```
    Events:
    LOAD
        EventID,
        EventDate,
        EventAttribute
    FROM [lib://AttachedFiles/Events.txt]
    (txt, utf8, embedded labels, delimiter is '\t', msq);
    ```

```
Intervals:
LOAD
    IntervalID,
    IntervalAttribute,
    IntervalBegin,
    IntervalEnd
FROM [lib://AttachedFiles/Intervals.txt]
(txt, utf8, embedded labels, delimiter is '\t', msq);

BridgeTable:
IntervalMatch (EventDate)
LOAD distinct IntervalBegin, IntervalEnd
Resident Intervals;
```

12. Klicka på **Ladda data**.
13. Öppna **datamodellvyn**. Datamodellen ser ut så här:

    Datamodell:Tabellerna Events, BridgeTable, Intervals och $Syn1



Datamodellen innehåller en sammansatt nyckel (fälten *IntervalBegin* och *IntervalEnd*) som tar formen av en syntetisk Qlik Sense-nyckel.

De grundläggande tabellerna är:

- *Events*-tabellen som innehåller exakt en post per händelse.
- *Intervals*-tabellen som innehåller exakt en post per intervall.
- Kopplingstabellen som innehåller exakt en post per kombination av händelse och intervall, och som länkar de två andra tabellerna.

Observera att en händelse kan höra till flera intervaller om intervallerna överlappar varandra. Ett intervall kan såklart ha flera händelser som hör till det.

Den här datamodellen är optimal i den mening att den är normaliserad och kompakt. *Events*-tabellen och *Intervals*-tabellen är båda oförändrade och innehåller det ursprungliga antalet poster. Alla Qlik Sense-beräkningar som används med de här tabellerna, till exempel Count(EventID), kommer att fungera och utvärderas korrekt.

Du kan läsa mer om IntervalMatch() i följande blogginlägg i Qlik Community: Using IntervalMatch() *(Använda IntervalMatch())*

Använda en While-slinga och iterativ inläsning IterNo()

Du kan åstadkomma nästan samma kopplingstabell med hjälp av en While-slinga och IterNo() som skapar uppräkningsbara värden mellan det lägre och det övre gränsvärdet för intervallet.

En slinga i LOAD-satsen kan skapas med hjälp av While-satsen. Exempel:

```
LOAD Date, IterNo() as Iteration From … While IterNo() <= 4;
```

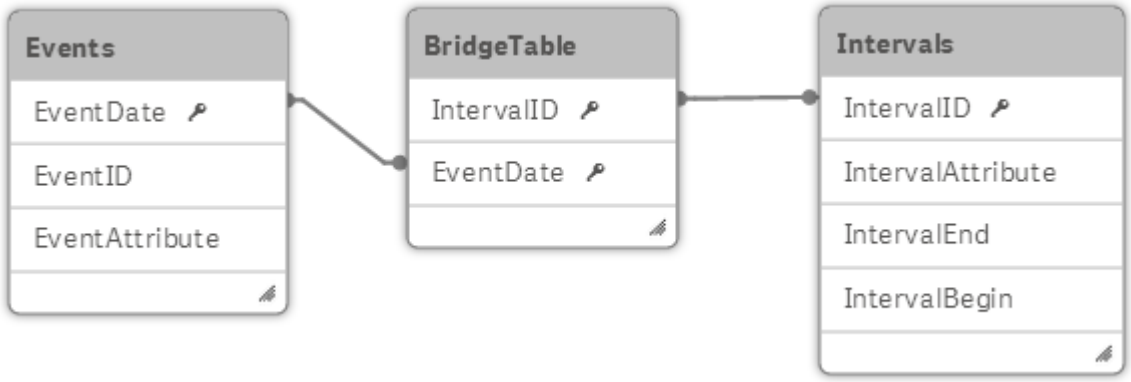
En sådan LOAD-sats skapar en slinga för varje indatapost och läser in denna om och om igen så länge uttrycket i While-satsen är sant. IterNo()-funktionen returnerar ”1” i den första iterationen, ”2” i den andra och så vidare.

Du har en primärnyckel för intervallen, IntervalID, så den enda skillnaden i skriptet är hur kopplingstabellen skapas:

Gör följande:

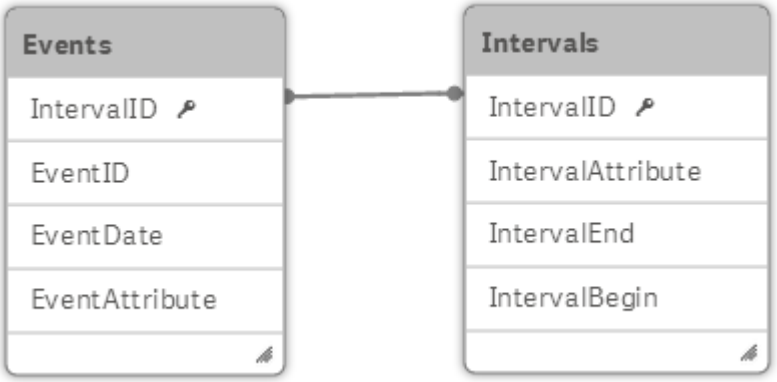
1. Ersätt de befintliga Bridgetable-satserna med följande skript:
   ```
   BridgeTable:
   LOAD distinct * Where Exists(EventDate);
   LOAD IntervalBegin + IterNo() - 1 as EventDate, IntervalID
       Resident Intervals
       While IntervalBegin + IterNo() - 1 <= IntervalEnd;
   ```
2. Klicka på **Ladda data**.
3. Öppna **datamodellvyn**. Datamodellen ser ut så här:

Datamodell:Tabellerna Events, BridgeTable och Intervals



Normalt är lösningen med tre tabeller bäst, eftersom du då kan ha ett många-till-många-förhållande mellan intervall och händelser. En vanlig situation är dock att du vet att en händelse bara kan höra till ett enda intervall. I sådana fall är kopplingstabellen inte nödvändig. *IntervalID* kan lagras direkt i händelsetabellen. Det finns flera sätt att åstadkomma detta, men det mest användbara är att koppla Bridgetable till *Events*-tabellen.

4. Lägg till följande kod i slutet av skriptet:
```
Join (Events)
LOAD EventDate, IntervalID
Resident BridgeTable;

Drop Table BridgeTable;
```
5. Klicka på **Ladda data**.
6. Öppna **datamodellvyn**. Datamodellen ser ut så här:
Datamodell:Tabellerna Events och Intervals



Öppna och stängda intervaller

Om en tabell är öppen eller stängd avgörs av slutpunkterna, beroende på om dessa är inkluderade i intervallet eller inte.

- Om slutpunkterna ingår är det ett stängt intervall:
  $[a,b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$
- Om slutpunkterna inte ingår är det ett öppet intervall:
  $]a,b[= \{x \in \mathbb{R} \mid a < x < b\}$
- Om en slutpunkt ingår är det ett halvöppet intervall:
  $[a,b[= \{x \in \mathbb{R} \mid a \leq x < b\}$

I ditt fall, där intervallen kan överlappa varandra och ett tal kan höra till fler än ett intervall, behöver du vanligtvis använda stängda intervall.

I vissa fall vill du kanske dock inte ha överlappande intervall, utan att ett tal bara ska höra till ett intervall. I sådana fall kommer det att orsaka problem om en punkt är slutet på ett intervall men på samma gång början på nästa. Ett tal med det värdet kommer att ingå i båda intervallen. I den här situationen bör du använda halvöppna intervall.

En praktisk lösning på det här problemet är att subtrahera ett mycket litet tal från slutvärdet på varje intervall, så att du får intervall som är slutna men som inte överlappar varandra. Om talen är datum blir det enklast att göra detta genom att använda DayEnd()-funktionen som returnerar den sista millisekunden av dagen:

```
Intervals:
LOAD…, DayEnd(IntervalEnd – 1) as IntervalEnd From Intervals;
```

Du kan även subtrahera ett litet tal manuellt. Om du gör det måste du se till att talet du subtraherar inte är för litet, eftersom operationen avrundas till 52 signifikanta binära siffror (14 decimalsiffor). Om talet du subtraherar är för litet blir skillnaden inte signifikant och du kommer fortfarande att använda det ursprungliga talet.

4 Datatvätt

Det finns tillfällen då källdata som du laddar i Qlik Sense inte nödvändigtvis ser ut som du vill ha dem i Qlik Sense-appen. Qlik Sense tillhandahåller en rad olika funktioner och satser som du kan använda för att omvandla data till ett format som fungerar för dig.

Mappning kan användas i ett Qlik Sense-skript för att ersätta eller ändra fältvärden eller namn när skriptet körs, så mappning kan användas för att rensa upp data och göra dem mer enhetliga eller för att ersätta delar av ett fältvärde, eller hela fältvärdet.

Vid laddning av data från olika tabeller har inte fältvärden som betecknar samma sak alltid samma namn. Eftersom en sådan brist på överensstämmelse hindrar associationer, bör problemet åtgärdas. En bra metod för att göra detta är att skapa en mappningstabell för att jämföra fältvärden.

4.1 Mappningstabeller

Tabeller som laddas via Mapping-laddning eller Mapping-val behandlas annorlunda än andra tabeller. De lagras i en separat del av minnet och används bara som mappningstabeller när skriptet körs. När skriptet har körts utelämnas dessa tabeller automatiskt.

Regler:

- En mappningstabell måste ha två kolumner, varav den första ska innehålla värdena som ska jämföras, den andra de önskade mappningsvärdena.
- Kolumnerna måste döpas, men då namnen inte har någon koppling till fältnamn i vanliga interna tabeller är de inte relevanta i sig.

4.2 Mapping-funktioner och -satser

Följande mappningsfunktioner/-satser kommer att tas upp i denna introduktionskurs:

• Prefixet Mapping

• ApplyMap()

• MapSubstring()

• Satsen Map … Using

• Satsen Unmap

4.3 Prefixet Mapping

Prefixet Mapping används i ett skript för att skapa en mappningstabell. Mappningstabellen kan sedan användas med funktionen ApplyMap(), funktionen MapSubstring() eller satsen Map … Using.

Gör följande:

1. Skapa en ny app och ge den ett namn.
2. Lägg till ett nytt skriptavsnitt i **Skriptredigeraren**.
3. Kalla avsnittet *Countries*.
4. Ange följande skript:

```
CountryMap:
MAPPING LOAD * INLINE [
Country, NewCountry
U.S.A., US
U.S., US
United States, US
United States of America, US
];
```

I *CountryMap*-tabellen sparas två kolumner: *Country* och *NewCountry*. I kolumnen *Country* sparas de olika sätt på vilka ett land har angetts i fältet *Country*. I kolumnen *NewCountry* sparas det sätt på vilket värdena ska mappas. Den här mappningstabellen kommer att användas för att spara enhetliga *US*-landsvärden i fältet *Country*. Om exempelvis *U.S.A.* har sparats i fältet *Country*, mappa det som *US*.

4.4 ApplyMap() funktion

Använd ApplyMap() när du vill ersätta data i ett fält utifrån en mappningstabell som skapats tidigare. Mappningstabellen måste laddas innan funktionen ApplyMap() kan användas. Data i tabellen *Data.xlsx* som du ska ladda ser ut så här:

Data table

ID	Namn	Land	Code
1	John Black	U.S.A.	SDFGBS1DI
2	Steve Johnson	U.S.A.	2ABC
3	Mary White	United States	DJY3DFE34
4	Susan McDaniels	u	DEF5556
5	Dean Smith	USA	KSD111DKFJ1

Observera att landet har skrivits på olika sätt. För att göra landsfältet enhetligt laddas mappningstabellen och funktionen **ApplyMap()** används.

Gör följande:

1. Under skriptet du skrev ovan, väljer och laddar du *Data.xlsx*. Infoga sedan skriptet.
2. Infoga följande ovanför LOAD-satsen du just har skapat:

```
Data:
```

Ditt skript bör se ut så här:

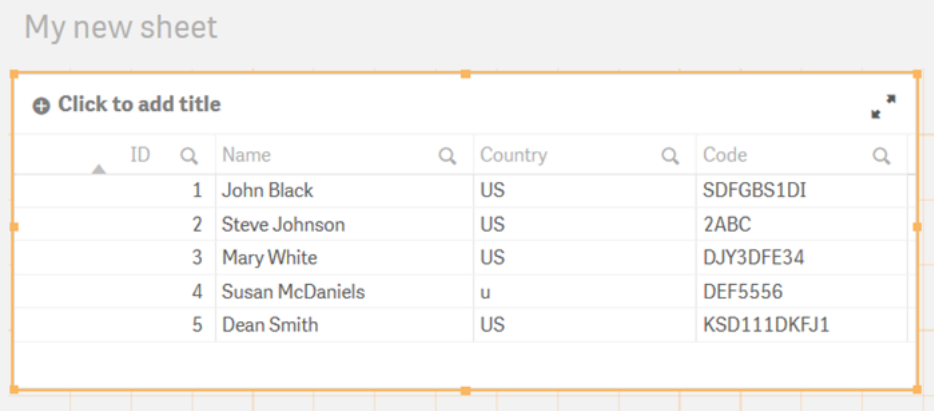
```
CountryMap:
MAPPING LOAD * INLINE [
    Country, NewCountry
    U.S.A., US
    U.S., US
    United States, US
    United States of America, US
];

Data:
LOAD
    ID,
    Name,
    Country,
    Code
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

3. Ändra den rad som innehåller `Country`, på följande sätt:
   `ApplyMap('CountryMap', Country) as Country,`

   Den första parametern i funktionen ApplyMap() har kartnamnet omslutet av enkla citattecken. Den andra parametern är det fält som innehåller de data som ska ersättas.

4. Klicka på **Ladda data**.
   Den resulterande tabellen ser ut så här:

   Tabell som visar data som laddats med funktionen ApplyMap()



ID	Name	Country	Code
1	John Black	US	SDFGBS1DI
2	Steve Johnson	US	2ABC
3	Mary White	US	DJY3DFE34
4	Susan McDaniels	u	DEF5556
5	Dean Smith	US	KSD111DKFJ1

   De olika stavningarna av *United States* har alla ändrats till *US*. Det finns en post som inte har stavats korrekt, så funktionen ApplyMap() ändrade inte det här fältvärdet. Med hjälp av funktionen ApplyMap() kan du använda den tredje parametern för att lägga till ett standarduttryck om mappningstabellen inte har något matchande värde.

5. Lägg till `'US'` som den tredje parametern i funktionen ApplyMap() för att hantera sådana fall där landet kan ha angetts felaktigt:
   `ApplyMap('CountryMap', Country, 'US') as Country,`

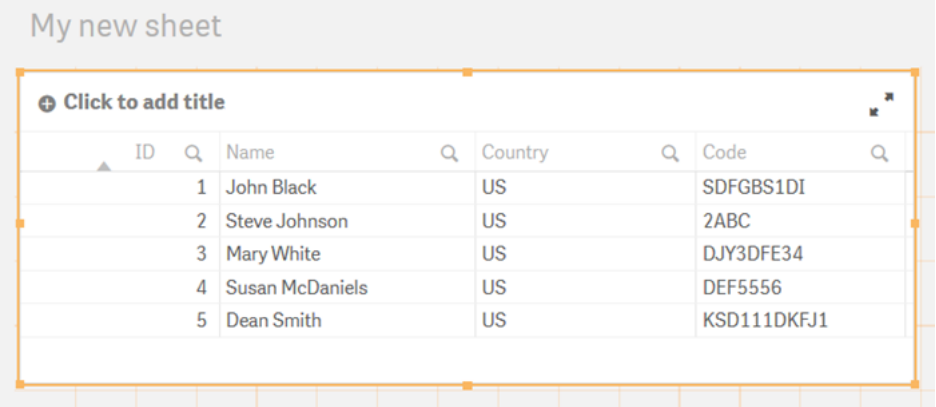
   Ditt skript bör se ut så här:

```
CountryMap:
MAPPING LOAD * INLINE [
    Country, NewCountry
    U.S.A., US
    U.S., US
    United States, US
    United States of America, US
];

Data:
LOAD
    ID,
    Name,
    ApplyMap('CountryMap', Country, 'US') as Country,
    Code
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

6. Klicka på **Ladda data**.

   Den resulterande tabellen ser ut så här:

   Tabell som visar data som laddats med funktionen ApplyMap



Du kan läsa mer om ApplyMap() i följande blogginlägg i Qlik Community: Don't join - use Applymap instead *(Länka inte – använd Applymap istället)*

4.5 Funktionen MapSubstring()

Med funktionen MapSubstring() kan du mappa delar av ett fält.

I den tabell som skapats med ApplyMap() vill vi nu att siffrorna ska skrivas som text, så att funktionen MapSubstring() används för att ersätta numeriska data med text.

För att det ska gå att göra detta måste en mappningstabell först skapas.

Gör följande:

1. Lägg till följande rader i skriptet efter avsnittet *CountryMap*, men före avsnittet *Data*.

```
CodeMap:
MAPPING LOAD * INLINE [
F1, F2
1, one
2, two
3, three
4, four
5, five
11, eleven
];
```

   I tabellen *CodeMap* mappas numren 1 till 5 samt 11.

2. I avsnittet *Data* i skriptet ändrar du satsen `Code` enligt följande:

```
MapSubString('CodeMap', Code) as Code
```

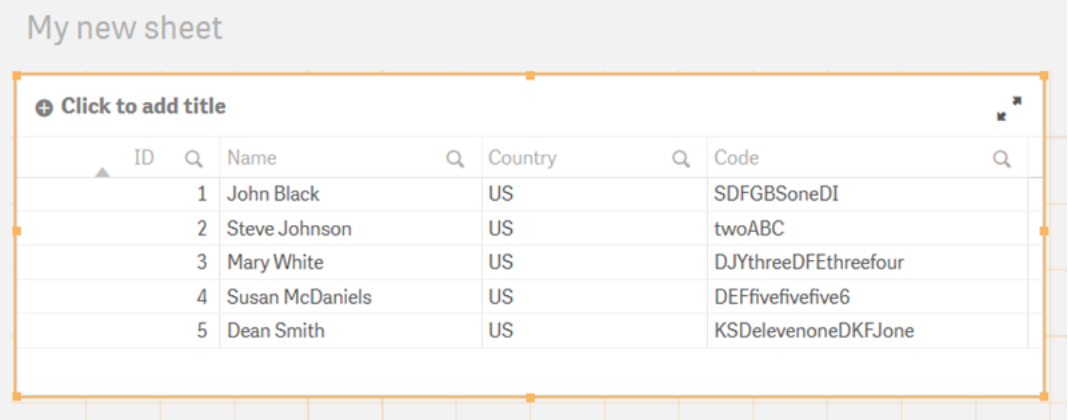
   Ditt skript bör se ut så här:

```
CountryMap:
MAPPING LOAD * INLINE [
    Country, NewCountry
    U.S.A., US
    U.S., US
    United States, US
    United States of America, US
];

CodeMap:
MAPPING LOAD * INLINE [
F1, F2
1, one
2, two
3, three
4, four
5, five
11, eleven
];

Data:
LOAD
    ID,
    Name,
    ApplyMap('CountryMap', Country, 'US') as Country,
    MapSubString('CodeMap', Code) as Code
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

3. Klicka på **Ladda data**.

   Den resulterande tabellen ser ut så här:

Tabell som visar data som laddats med funktionen MapSubString



ID	Name	Country	Code
1	John Black	US	SDFGBSoneDI
2	Steve Johnson	US	twoABC
3	Mary White	US	DJYthreeDFEthreefour
4	Susan McDaniels	US	DEFfivefivefive6
5	Dean Smith	US	KSDelevenoneDKFJone

De numeriska tecknen ersattes med text i fältet *Code*. Om en siffra visas mer än en gång som i fallet med ID=3 och ID=4, upprepas också texten. ID=4. *Susan McDaniels* hade siffran 6 i sin kod. Eftersom 6 inte mappades i *CodeMap*-tabellen förblir den oförändrad. ID=5, *Dean Smith*, hade 111 i sin kod. Denna har mappats som "elevenone".

Du kan läsa mer om MapSubstring() i följande blogginlägg i Qlik Community: Mapping … and not the geographical kind *(Använda maps ... och det handlar inte om geografi)*

4.6 Map … Using

Satsen Map … Using kan också användas för att tillämpa en mappning på ett fält. Den fungerar dock lite annorlunda än ApplyMap(). Medan ApplyMap() hanterar mappningen varje gång fältnamnet påträffas, hanterar Map … Using mappningen när värdet har sparats under fältnamnet i den interna tabellen.

Vi ska titta på ett exempel. Anta att vi laddar fältet *Country* flera gånger i skriptet och vill tillämpa en karta varje gång fältet laddades. Funktionen ApplyMap() kan användas på det sätt som illustrerades tidigare i den här introduktionskursen eller också kan Map … Using användas.

Om Map … Using används tillämpas kartan på fältet när fältet är sparat i den interna tabellen. Så i exemplet nedan tillämpas kartan på fältet *Country* i tabellen Data1 men skulle inte tillämpas på fältet *Country2* i tabellen *Data2*. Detta beror på att satsen Map … Using enbart tillämpas på fält med namnet *Country*. När fältet *Country2* sparas i den interna tabellen har det inte längre namnet *Country*. Om du vill att kartan ska tillämpas på *Country2*-tabellen skulle du behöva använda funktionen ApplyMap().

Satsen Unmap avslutar satsen Map … Using, vilket innebär att om *Country* skulle laddas in efter satsen Unmap så skulle *CountryMap* inte tillämpas.

Gör följande:

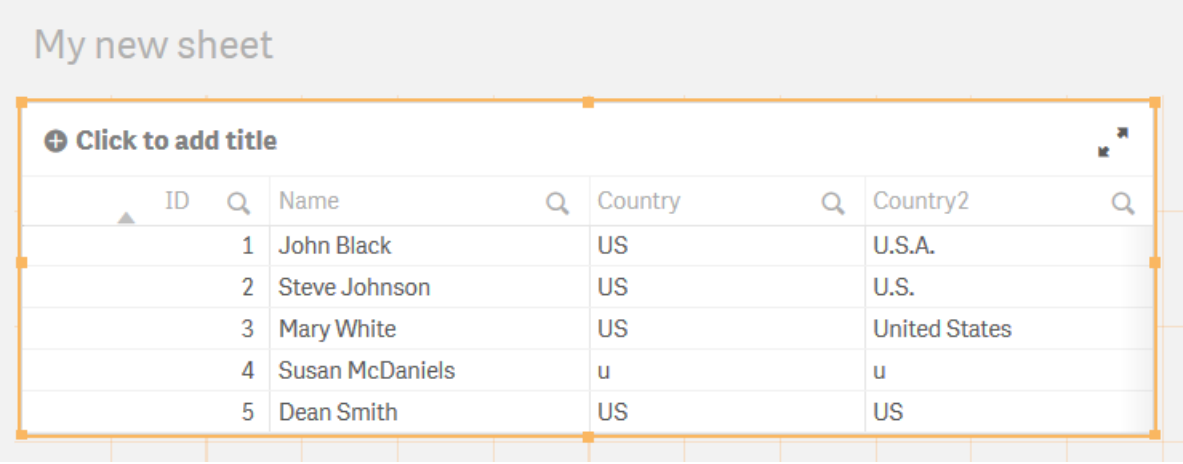
1. Ersätt skriptet för tabellen *Data* med följande:

```
Map Country Using CountryMap;
Data1:
    LOAD
    ID,
    Name,
    Country
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);

Data2:
LOAD
    ID,
    Country as Country2
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);
UNMAP;
```

2. Klicka på **Ladda data**.

   Den resulterande tabellen ser ut så här:

   Tabell som visar data som laddats med funktionen Map … Using



5 Hantera hierarkiska data

Hierarkier är en viktig del av alla Business Intelligence-lösningar. De används för att beskriva dimensioner som naturligt innehåller olika precisionsnivåer. Vissa är enkla och intuitiva medan andra är komplexa och kräver mycket tankearbete för att kunna modelleras korrekt.

Från övre delen av en hierarki till botten är medlemmarna progressivt allt mer detaljerade. Till exempel, om en dimension har nivåerna Marknad, Land, Delstat respektive Ort, visas medlemmen Nord- och Sydamerika på hierarkins översta nivå, medlemmen USA visas på den andra nivån, medlemmen Kalifornien visas på den tredje nivån och San Francisco visas på den nedersta nivån. Kalifornien är mer specifikt än USA och San Fransisco är mer specifikt än Kalifornien.

Att spara hierarkier i en relationsmodell är en vanlig utmaning med många olika lösningar. Det finns flera metoder:

- Horisontell hierarki
- Modell för angränsande lista
- Metod för sökvägsuppräkning
- Modell för kapslade uppsättningar
- Lista över överordnade element

För syftena med den här introduktionskursen kommer vi att skapa en lista med överordnade element eftersom den presenterar hierarkin i en form som kan användas direkt i en fråga. Mer information om andra metoder finns i Qlik Community.

5.1 Prefixet Hierarchy

Prefixet Hierarchy är ett skriptkommando som du placerar framför en LOAD- eller SELECT-sats som laddar en tabell med angränsande noder. Satsen LOAD måste innehålla minst tre fält: Ett ID som är en unik nyckel för noden, en referens till det överordnade objektet och ett namn.

Prefixet omvandlar en laddad tabell till en tabell med expanderade noder, dvs. en tabell som har ett antal ytterligare kolumner, en för varje nivå i hierarkin.

Gör följande:

1. Skapa en ny app och ge den ett namn.
2. Lägg till ett nytt skriptavsnitt i **Skriptredigeraren**.
3. Kalla avsnittet *Wine*.
4. Klicka på **Välj data** under **AttachedFiles** i menyn till höger.
5. Ladda upp och välj sedan *Winedistricts.txt*.
6. Avmarkera fälten *Lbound* och *RBound* i fönstret **Välj data från**, så att de inte laddas.
7. Klicka på **Infoga skript**.
8. Ange följande över LOAD-satsen:

```
Hierarchy (NodeID, ParentID, NodeName)
```

Ditt skript bör se ut så här:

```
Hierarchy (NodeID, ParentID, NodeName)
LOAD
    NodeID,
    ParentID,
    NodeName
FROM [lib://AttachedFiles/Winedistricts.txt]
(txt, utf8, embedded labels, delimiter is '\t', msq);
```

9. Klicka på **Ladda data**.
10. Använd avsnittet **Granska** i **datamodellvyn** för att visa den resulterande tabellen.

    Den resulterande expanderade nodtabellen har exakt samma antal poster som källtabellen: En per nod. Den expanderade nodtabellen är mycket praktisk eftersom den uppfyller ett antal krav för analys av en hierarki i en relationsmodell:

    - Alla nodnamn finns i en och samma kolumn, så att den kan användas för sökningar.
    - Dessutom har de olika nodnivåerna expanderats till ett fält vardera; fält som kan användas i hierarkiska grupper eller som dimensioner i pivottabeller.
    - Dessutom har de olika nodnivåerna expanderats till ett fält vardera; fält som kan användas i hierarkiska grupper.
    - Den kan fås att innehålla en sökväg som är unik för noden, och som listar alla överordnade element i rätt ordning.
    - Den kan fås att innehålla nodens djup, dvs. avståndet från roten.

    Den resulterande tabellen ser ut så här:

    Tabell som visar exempeldata som har laddats med prefixet Hierarchy

My new sheet

NodeID	ParentID	NodeName	NodeName1	NodeName2	NodeName3	NodeName4	NodeName5	NodeName6
289	288	Bas-Médoc	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
290	289	Listrac	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
291	289	Pauillac	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
292	289	Saint-Estèphe	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
293	289	Saint-Julien	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
294	288	Haut-Médoc	The World	Europe	France	Bordeaux	Médoc	Haut-Médoc
295	294	Margaux	The World	Europe	France	Bordeaux	Médoc	Haut-Médoc

5.2 Prefixet HierarchyBelongsTo

Prefixet HierarchyBelongsTo är, liksom prefixet Hierarchy, ett skriptkommando som du placerar framför en LOAD- eller SELECT-sats som laddar en tabell med angränsande noder.

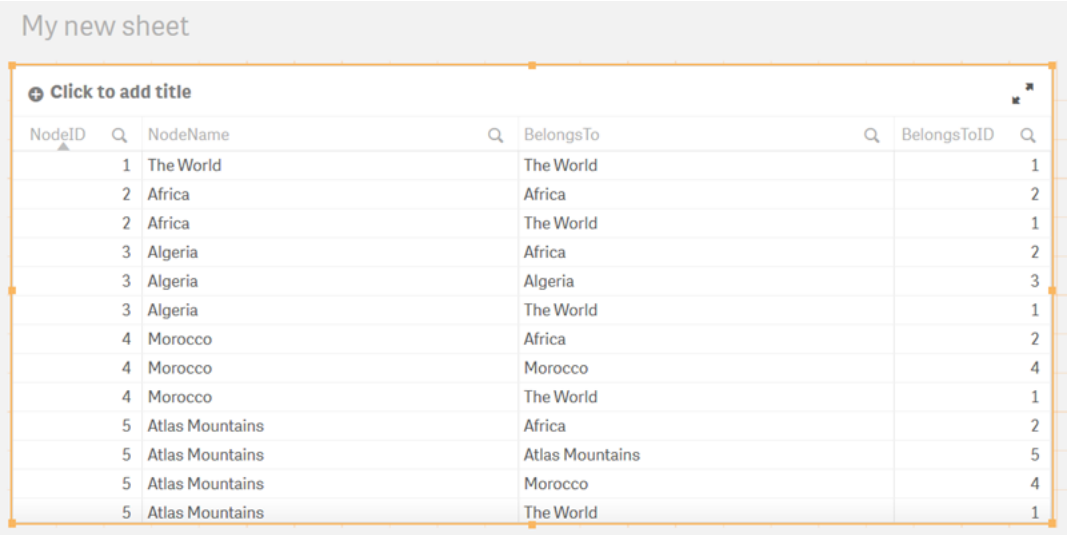
Även här måste LOAD-satsen innehålla minst tre fält: Ett ID som är en unik nyckel för noden, en referens till det överordnade objektet och ett namn. Prefixet omvandlar den laddade tabellen till en överordnad tabell, en tabell som har varje kombination av ett överordnat och en underordnat element listad som en separat post. Därför är det väldigt enkelt att hitta alla överordnade eller alla underordnade element till en viss nod.

Gör följande:

1. Ändra satsen Hierarchy i **Skriptredigeraren** så att den ser ut på följande sätt:
   `HierarchyBelongsTo (NodeID, ParentID, NodeName, BelongsToID, BelongsTo)`
2. Klicka på **Ladda data**.
3. Använd avsnittet **Granska** i **datamodellvyn** för att visa den resulterande tabellen.
   Den överordnade tabellen uppfyller ett antal krav för analys av en hierarki i en relationsmodell:
   - Om nod-ID:t representerar de enskilda noderna, representerar det överordnade ID:t hela träden och underträden i hierarkin.
   - Alla nodnamn finns både i rollen som noder och i rollen som träd, och båda kan användas för sökningar.
   - Den kan utformas så att den innehåller djupskillnaden mellan djupet på noden och djupet på det överordnade elementet, det vill säga avståndet från underträdets rot.

   Den resulterande tabellen ser ut så här:

   Tabellen visar data som har laddats med prefixet HierarchyBelongsTo


My new sheet

Click to add title

NodeID	NodeName	BelongsTo	BelongsToID
1	The World	The World	1
2	Africa	Africa	2
2	Africa	The World	1
3	Algeria	Africa	2
3	Algeria	Algeria	3
3	Algeria	The World	1
4	Morocco	Africa	2
4	Morocco	Morocco	4
4	Morocco	The World	1
5	Atlas Mountains	Africa	2
5	Atlas Mountains	Atlas Mountains	5
5	Atlas Mountains	Morocco	4
5	Atlas Mountains	The World	1

Auktorisering

Det är inte ovanligt att en hierarki används för auktorisering. Ett exempel är en organisationshierarki. Varje chef bör ha rätt att se allting som rör den egna avdelningen, inklusive alla underavdelningar. Men de bör inte nödvändigtvis ha rättigheten att se andra avdelningar.

Exempel på en organisationshierarki

Detta innebär att olika personer har tillåtelse att se olika underträd i organisationen. Behörighetstabellen kan se ut på följande sätt:

Behörighetstabell

ACCESS	NTNAME	PERSON	POSITION	PERMISSIONS
USER	ACME\JRL	John	CPO	HR
USER	ACME\CAH	Carol	CEO	CEO
USER	ACME\JER	James	Director Engineering	Engineering
USER	ACME\DBK	Diana	CFO	Ekonomi
USER	ACME\RNL	Bob	COO	Sales
USER	ACME\LFD	Larry	CTO	Produkt

I det här fallet har *Carol* tillåtelse att se allt som berör *CEO* och under; *Larry* har tillåtelse att se *Product*-organisationen och *James* har tillåtelse att se enbart *Engineering*-organisationen.

Exempel:

Hierarkin lagras ofta i en tabell med angränsande noder. I det här exemplet kan du lösa detta genom att ladda tabellen med angränsande noder med en HierarchyBelongsTo-sats och kalla det överordnade fältet *Tree*.

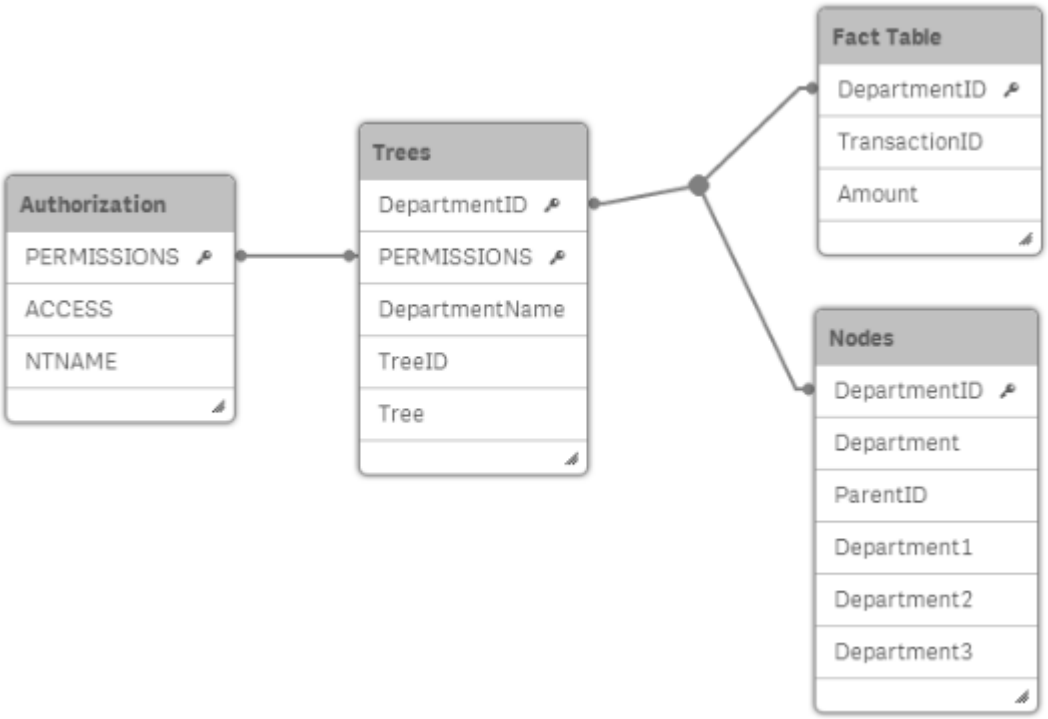
Om du vill använda Section Access laddar du en kopia av *Tree* med versaler och kallar det nya fältet *PERMISSIONS*. Slutligen ska du ladda behörighetstabellen. De två sista stegen kan utföras med följande skriptrader. Observera att tabellen TempTrees är tabellen som skapats av HierarchyBelongsTo-satsen.

Observera att det här bara är ett exempel. Det finns ingen övning för dessa data att göra i Qlik Sense.

```
Trees:
LOAD *,
    Upper(Tree) as PERMISSIONS
    Resident TempTrees;
Drop Table TempTrees;

Section Access;
Authorization:
LOAD   ACCESS,
       NTNAME,
       UPPER(Permissions) as PERMISSIONS
From Organization;
Section Application;
```

Det här exemplet skulle skapa följande datamodell:

Datamodell: Tabellerna Authorization, Trees, Fact och Nodes



6 QVD-filer

En QVD (QlikView Data)-fil är en fil som innehåller en tabell med data som har exporterats från Qlik Sense eller QlikView. QVD är ett Qlik-originalformat och kan bara skrivas till och läsas av Qlik Sense eller QlikView. Detta filformat är optimerat för hastighet vid datainläsning från ett Qlik Sense-skript, men är samtidigt mycket kompakt. Datainläsning från en QVD-fil är vanligtvis 10-100 gånger snabbare än inläsning från andra datakällor.

QVD-filer kan läsas in på två olika sätt: standard (snabbt) och optimerat (snabbare). Vilket sätt som används bestäms automatiskt av Qlik Sense-skriptmotorn. Det optimerade läget kan endast användas när alla inlästa fält läses utan någon typ av transformering (formler som påverkar fälten). Filnamnen kan dock ändras. En Where-sats som gör att Qlik Sense packar upp posterna leder också till att optimerad laddning inaktiveras.

En QVD-fil innehåller exakt en datatabell och består av tre delar:

- Ett XML-huvud (i teckenuppsättningen UTF-8) som beskriver tabellens fält, layouten för den efterföljande informationen, samt vissa ytterligare metadata.
- Symboltabeller i byte-komprimerat format.
- Faktiska tabelldata i bit-komprimerat format.

QVD-filer har många användningsområden. Fyra huvudsakliga användningsområden kan omedelbart identifieras. Fler än ett användningsområde kan gälla på samma gång:

- Snabbare dataladdningshastighet

  Genom att buffra de block av indata som inte har ändrats respektive ändrats långsamt i QVD-filer kan skriptexekveringen snabbas upp väsentligt för stora datamängder.

- Snabbare dataladdningshastighet

  Genom att buffra de block av indata som inte har ändrats respektive ändrats långsamt i QVD-filer kan skriptexekveringen snabbas upp väsentligt för stora datamängder.

- Minskat tryck på databasservrar

  Mängden data som hämtas från externa datakällor kan minskas avsevärt. Detta minskar arbetstrycket på de externa databaserna och mängden nätverkstrafik. En annan fördel är att när flera Qlik Sense-skript delar samma data, behöver data bara laddas en gång från källdatabasen till en QVD-fil. De andra applikationerna kan använda data genom denna QVD-fil.

- Konsolidera data från flera Qlik Sense-applikationer.

  Med Binary-skriptsatsen går det att ladda data från en enda Qlik Sense-applikation till en annan, men med QVD-filer kan ett Qlik Sense-skript kombinera data från flera olika Qlik Sense-applikationer. Detta möjliggör konsolidering av likartade data från olika affärsenheter med mera.

- Inkrementell laddning

  I många fall kan QVD-funktionaliteten användas för att förenkla inkrementell laddning genom att bara ladda nytillkomna poster i en växande databas.

För att se hur Qlik-forumet använder Qlik Application Automation för att förbättra QVD-laddningstiderna kan du läsa i Hur du delar upp QVD:er med en automatisering för att förbättra omladdningar

6.1 Skapa QVD-filer

En QVD-fil kan skapas på två sätt:

- Skapas och namnges explicit med hjälp av Store-kommandot i Qlik Sense-skriptet.
  Ange i skriptet att en tabell, eller delar av en tabell, som tidigare har lästs in ska exporteras till en namngiven fil på en viss plats.
- Skapa och sköta underhåll automatisk direkt från skriptet.
  Om en load- eller select-sats föregås av prefixet Buffer, skapar Qlik Sense automatiskt en QVD-fil som under vissa omständigheter kan användas istället för den ursprungliga datakällan vid laddning.

De resulterande QVD-filerna skiljer sig inte åt vad gäller inläsningshastighet.

Store

Detta skriptuttryck skapar en namngiven QVD-, CSV- eller txt-fil.

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

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**. Satsen store (spara) till en CSV -fil stöder inte BIFF -export.

Exempel:

```
Store mytable into [lib://AttachedFiles/xyz.qvd];
Store * from mytable into [lib://FolderConnection/xyz.qvd];
Store myfield from mytable into 'lib://FolderConnection/xyz.qvd';
Store myfield as renamedfield, myfield2 as renamedfield2 from mytable into
[lib://AttachedFiles/xyz.qvd];
Store mytable into 'lib://FolderConnection/myfile.txt';
Store * from mytable into 'lib://FolderConnection/myfile.csv';
```

Gör följande:

1. Öppna appen *Avancerad skriptvägledning*.
2. Klicka på skriptavsnittet *Product*.
3. Lägg till följande i slutet av skriptet:

```
Store * from Product into [lib://AttachedFiles/ProductData.qvd](qvd);
```

Ditt skript bör se ut så här:

```
CrossTable(Month, Sales)
LOAD
    Product,
    "Jan 2014",
    "Feb 2014",
    "Mar 2014",
    "Apr 2014",
    "May 2014"
FROM [lib://AttachedFiles/Product.xlsx]
(ooxml, embedded labels, table is Product);

Store * from Product into [lib://AttachedFiles/ProductData.qvd](qvd);
```

4. Klicka på **Ladda data**.

   Filen *Product.qvd* bör nu finnas i listan med filer.

   Den här datafilen är resultatet av ett **Crosstable**-skript och är en tabell med tre kolumner, med en kolumn för varje kategori (Product, Month, Sales). Den här datafilen kan nu användas för att ersätta hela *Product* -skriptavsnittet.

6.2 Läsa in data från QVD-filer

En QVD-fil kan laddas eller nås av Qlik Sense på följande sätt:

- Läsa in en QVD-fil som explicit datakälla. Det går att referera till QVD-filerna genom en load-sats i Qlik Sense-skriptet, precis som med vilken annan typ av textfil som helst (csv, fix, dif, biff och så vidare).

  Exempel:

  ```
  LOAD * from 'lib://FolderConnection/xyz.qvd' (qvd);
  LOAD fieldname1, fieldname2 from [lib://FolderConnection/xyz.qvd] (qvd);
  LOAD fieldname1 as newfieldname1, fieldname2 as newfieldname2 from
  [lib://AttachedFiles/xyz.qvd](qvd);
  ```

- Automatisk inläsning av buffrade QVD-filer. När buffer-prefixet används för load- eller select-satser, behöver du inte uttryckligen ange hur inläsningen ska ske. Qlik Sense avgör i vilken utsträckning data från QVD-filen ska användas eller om data istället ska hämtas med hjälp av den ursprungliga LOAD- eller SELECT-satsen.
- Nå QVD-filer från skriptet. Ett antal skriptfunktioner (alla börjar med QVD) kan användas för att hämta olika typer av information för data som återfinns i XML-huvudet till en QVD-fil.

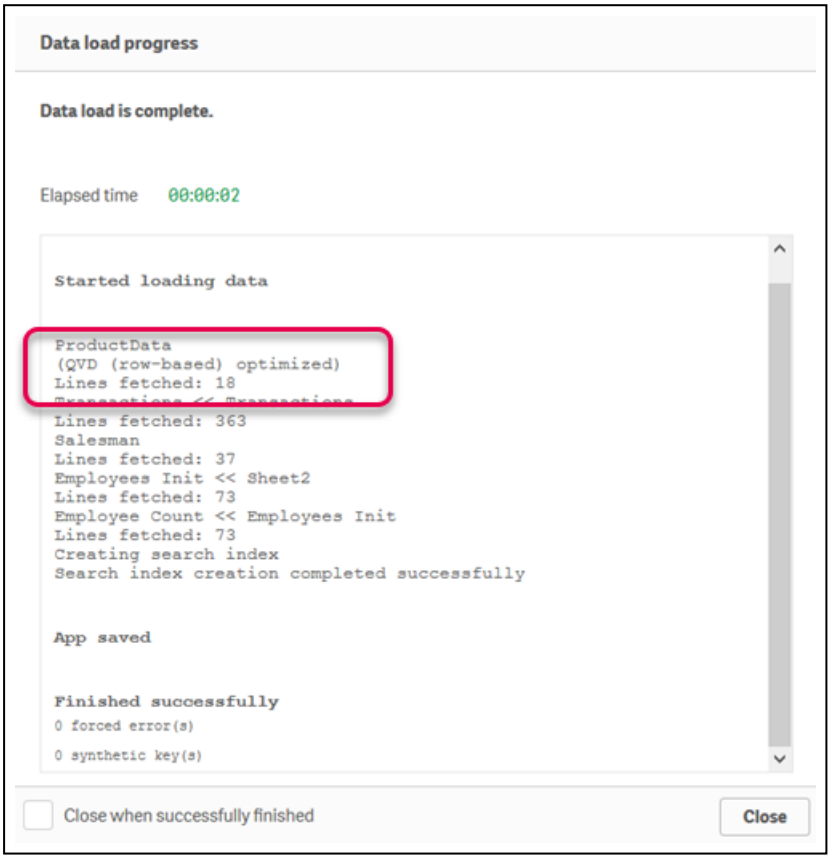
Gör följande:

1. Kommentera ut hela skriptet i *Product*-skriptavsnittet.
2. Ange följande skript:

```
Load * from [lib://AttachedFiles/ProductData.qvd](qvd);
```

3. Klicka på **Ladda data**.
   Data laddas från filen QVD.

   Förloppsfönster för dataladdning



Om du vill veta mer om att använda QVD-filer för inkrementell laddning läser du det här blogginlägget i Qlik Community: Översikt över inkrementell laddning i Qlik

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.

Exempel:

```
Buffer load * from MyTable;
```

stale [after] amount [(days | hours)]

Amount är ett tal som anger tidsperioden. Decimals kan användas. Om ingen enhet anges förutsätts dagar.

Alternativet stale after används oftast med databaskällor där det inte finns någon enkel tidsangivelse för ursprungsdata. En stale after-sats anger helt enkelt en tidsperiod efter att QVD-bufferten skapades då 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.

Exempel:

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

Incremental

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 följande nya poster läses in från originalkällan. Slutligen skapas en uppdaterad QVD-fil.

Observera att alternativet incremental bara kan användas tillsammans med LOAD-satser och textfiler och att inkrementell laddning inte kan användas där gamla data ändras eller tas bort.

Exempel:

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

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. Store-satsen ska användas om du vill behålla innehållet i bufferten som en QVD- eller CSV-fil.

Gör följande:

1. Skapa en ny app och ge den ett namn.
2. Lägg till ett nytt skriptavsnitt i **Skriptredigeraren**.
3. Klicka på **Välj data** under **AttachedFiles** i menyn till höger.
4. Ladda upp och välj sedan *Cutlery.xlsx*.
5. Klicka på **Infoga skript** i fönstret **Välj data från**.
6. Kommentera ut fälten i load-satsen och ändra den till följande:
   ```
   Buffer LOAD *
   ```

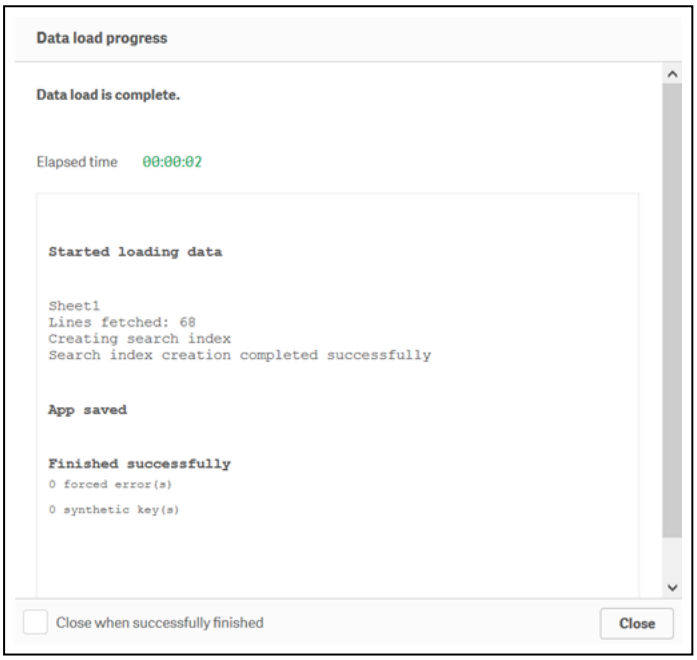
   Ditt skript bör se ut så här:

```
Buffer LOAD *
       //     "date",
       //     item,
       //     quantity
       FROM [lib://AttachedFiles/Cutlery.xlsx]
       (ooxml, embedded labels, table is Sheet1);
```

7. Klicka på **Ladda data**.

   Den första gången du laddar data laddas de från *Cutlery.xlsx*.

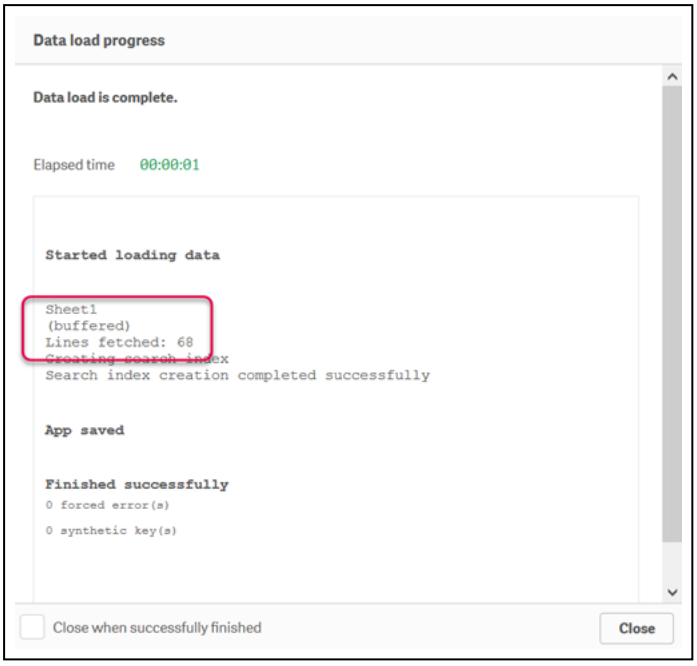
   Förloppsfönster för dataladdning



   Buffer-satsen skapar även en QVD-fil och lagrar den i Qlik Sense. I en Qlik Sense Enterprise on Windows-driftsättning lagras den i en katalog på Qlik Sense-servern.

8. Klicka på **Ladda data** igen.
9. Den här gången laddas data från QVD-filen som skapades av Buffer-satsen när du laddade data första gången.

Förloppsfönster för dataladdning



6.3 Tack!

Nu när du är klar med den här introduktionskursen vet du lite mer om skript i Qlik Sense. Besök gärna vår webbplats för mer information om den ytterligare utbildningar som finns tillgängliga.