



# Zelfstudie - Volgende stappen in scriptgebruik

Qlik Sense®

November 2024

Copyright © 1993-} QlikTech International AB. Alle rechten voorbehouden.



<b>1 Welkom bij deze zelfstudie!</b>	<b>5</b>
1.1 Wat u gaat leren	5
1.2 Voor wie is deze zelfstudie bedoeld	5
1.3 Inhoud pakket	5
1.4 Lessen van deze zelfstudie	6
1.5 Meer informatie en bronnen	6
<b>2 De opdrachten LOAD en SELECT</b>	<b>7</b>
<b>3 Gegevens transformeren</b>	<b>8</b>
3.1 Het prefix Crosstable gebruiken	8
Prefix Crosstable	8
De geheugencache leegmaken	12
3.2 Tabellen samenvoegen met Join en Keep	12
Join	13
Join gebruiken	13
Keep	16
Inner	17
Left	18
Right	19
3.3 Inter-recordfuncties gebruiken: Peek, Previous en Exists	21
Peek()	21
Previous()	21
Exists()	22
Peek() en Previous() gebruiken	22
Exists() gebruiken	25
3.4 Afstemmingsintervallen en iteratief laden	29
Het prefix IntervalMatch() gebruiken	29
Een While-lus en iteratief laden van IterNo() gebruiken	31
Open en gesloten intervallen	33
<b>4 Gegevens opschonen</b>	<b>34</b>
4.1 Tabellen toewijzen	34
Regels:	34
4.2 Functies en opdrachten Mapping	34
4.3 Prefix Mapping	35
4.4 Functie ApplyMap()	35
4.5 Functie MapSubstring()	38
4.6 Map ... Using	40
<b>5 Hiërarchische gegevens verwerken</b>	<b>42</b>
5.1 Prefix Hierarchy	42
5.2 Prefix HierarchyBelongsTo	44
Autorisatie	45
<b>6 QVD-bestanden</b>	<b>48</b>
6.1 QVD-bestanden maken	49
Store	49
6.2 Gegevens uit QVD-bestanden lezen	50
Buffer	52

---

6.3 Hartelijk dank! .....	54
---------------------------	----

# 1 Welkom bij deze zelfstudie!

Welkom bij deze zelfstudie waarin u leert geavanceerde scripts samen te stellen in Qlik Sense.

Zodra u bekend bent met de basisbeginselen van scriptgebruik, kunt u beginnen met het uitvoeren van meer geavanceerde bewerkingen van uw gegevens als u deze in Qlik Sense laadt. Hieronder valt bijvoorbeeld het transformeren van gegevens met kruistabellen, gegevens opschonen en gegevens maken en laden vanuit Qlik-gegevensbestanden die bekend staan als QVD-bestanden.

## 1.1 Wat u gaat leren

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

## 1.2 Voor wie is deze zelfstudie bedoeld

U bent vertrouwd met de basisbeginselen van scriptgebruik in Qlik Sense. Met andere woorden: u hebt gegevens geladen en gemanipuleerd met behulp van scripts.

Als u dat nog niet hebt gedaan, raden we u aan om de zelfstudie Scriptgebruik voor beginners af te ronden.

U moet toegang hebben tot de editor voor laden van gegevens en u moet toestemming hebben om gegevens te laden in Qlik Sense Enterprise on Windows.

De instructies zijn over het algemeen ook van toepassing op Qlik Sense Cloud Business.

## 1.3 Inhoud pakket

Het zip-pakket dat u hebt gedownload bevat de volgende gegevensbestanden die u nodig hebt om de zelfstudie af te ronden:

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

Het pakket bevat ook een kopie van de app *Zelfstudie voor geavanceerd scriptgebruik*. Extra scriptsecties in de app bevatten de scripts voor andere apps die u in deze zelfstudie maakt. U kunt de app uploaden naar uw hub.

We raden u aan om de app zelf te bouwen en dat te doen zoals beschreven in de zelfstudie, zodat u hier zoveel mogelijk van leert. U moet daarnaast de app uploaden en deze aan de gegevensbestanden koppelen om de gegevensloads te laten functioneren, zoals beschreven in de zelfstudie.




Als u echter tegen problemen aanloopt, kan de app u helpen deze op te lossen. We hebben aangegeven welke scriptsegmenten aan welke les gekoppeld zijn.

## 1.4 Lessen van deze zelfstudie

U hebt 3 tot 4 uur nodig om deze zelfstudie af te ronden, afhankelijk van uw ervaring met Qlik Sense. De onderwerpen zijn bedoeld om achtereenvolgens door te nemen. U kunt echter op elk gewenst moment stoppen om de zelfstudie later weer te hervatten. Er zijn gelukkig geen toetsen.

- Gegevens transformeren
- Het prefix Crosstable gebruiken
- Tabellen samenvoegen met Join en Keep
- Inter-recordfuncties gebruiken: Peek, Previous en Exists
- Afstemmingsintervallen en iteratief laden
- Gegevens opschonen
- Hiërarchische gegevens verwerken
- QVD-bestanden

## 1.5 Meer informatie en bronnen

-  [Qlik](#) biedt een groot aantal bronnen, voor het geval u nog meer wilt leren.
- [Qlik online help](#) is beschikbaar.
- Training, inclusief gratis online cursussen, is beschikbaar in de  [Qlik Continuous Classroom](#).
- Discussieforums, blogs, en meer kunt u vinden in de  [Qlik Community](#).

## 2 De opdrachten LOAD en SELECT

U kunt gegevens in Qlik Sense laden via de LOAD- en SELECT-opdrachten. Elk van deze opdrachten genereert een interne tabel. LOAD wordt gebruikt voor het laden van gegevens uit bestanden, terwijl SELECT wordt gebruikt voor het laden van gegevens uit databases.

In deze zelfstudie gebruikt u gegevens uit bestanden en dus LOAD-opdrachten.

U kunt ook een voorafgaande LOAD gebruiken om de inhoud van de geladen gegevens te kunnen bewerken. Zo moet het hernoemen van velden gebeuren aan de hand van een LOAD-opdracht; SELECT-opdrachten staan geen wijzigingen aan veldnamen toe.

De volgende regels zijn van toepassing bij het laden van gegevens in Qlik Sense:

- Qlik Sense maakt geen onderscheid tussen tabellen die zijn gegenereerd via een LOAD- of SELECT-opdracht. Dit betekent dat als er meerdere tabellen worden geladen, het niet uitmaakt of dit met LOAD- of SELECT-instructies of beide is gebeurd.
- De volgorde van de velden in de opdracht of in de oorspronkelijke tabel in de database is niet belangrijk in de Qlik Sense-logica.
- Veldnamen zijn hoofdlettergevoelig en worden gebruikt om koppelingen tussen gegevenstabellen te creëren. Hierdoor is het soms nodig om velden in het load-script een nieuwe naam te geven, om op die manier het gewenste gegevensmodel te bewerkstelligen.

## 3 Gegevens transformeren

U kunt gegevens transformeren en manipuleren in de editor voor het laden van gegevens voordat u gegevens in uw app gebruikt.

Een van de voordelen van het bewerken van gegevens is dat u ervoor kunt kiezen om slechts een subset van de gegevens uit een bestand te laden, bijvoorbeeld enkele specifieke kolommen van een tabel, om de gegevens op efficiëntere wijze te kunnen hanteren. U kunt de gegevens ook meer dan één keer laden om de onbewerkte gegevens op te splitsen in verschillende nieuwe logische tabellen. Het is tevens mogelijk om gegevens uit meerdere bronnen te laden en samen te voegen tot één tabel in Qlik Sense.

De volgende oefeningen laten u zien hoe u gegevens kunt laden met behulp van de Crosstable-prefix. U leert tevens hoe u tabellen kunt samenvoegen, inter-recordfuncties zoals Peek en Previous kunt gebruiken en dezelfde rij verschillende malen kunt laden met While Load.

### 3.1 Het prefix Crosstable gebruiken

Kruistabellen zijn een veel voorkomend type tabel die bestaat uit een matrix van waarden tussen twee rechthoekige lijsten met veldnaamgegevens. Wanneer u een kruistabel met gegevens hebt, kan het prefix Crosstable worden gebruikt voor het transformeren van de gegevens en het maken van de gewenste velden.

#### Prefix Crosstable

In de volgende *Product*-tabel hebt u één kolom per maand en één rij per product.

Tabel Product						
Product	Jan 2014	Feb 2014	Mar 2014	Apr 2014	Mei 2014	Jun 2014
A	100	98	100	83	103	82
B	284	279	297	305	294	292
C	50	53	50	54	49	51

Als deze tabel wordt geladen, resulteert dat in een uitvoer van een tabel met één veld voor *Product* en één veld voor elk van de maanden.



*Product-tabel met Product-veld en een veld voor elk van de maanden*

Product
Product
Jan 2014
Feb 2014
Mar 2014
Apr 2014
May 2014
Jun 2014

Als u deze gegevens wilt analyseren, is het veel gemakkelijker als alle getallen in één veld staan en alle maanden in een ander. In dit geval is dat een tabel met drie kolommen waarvan elke kolom een categorie bevat (*Product, Month, Sales*).

*Product-tabel met velden Product, Month en Sales*

Product
Product
Month
Sales

Het prefix *Crosstable* converteert de gegevens naar een tabel met één kolom voor *Month* en een andere voor *Sales*. Een andere manier om het uit te drukken is door te zeggen dat veldnamen worden geconverteerd naar veldwaarden.

#### Doe het volgende:

1. Maak een nieuwe app en noem deze *Zelfstudie voor geavanceerd scriptgebruik*.
2. Voeg een nieuwe scriptsectie toe in de **Editor voor het laden van gegevens**.
3. Geef de sectie *Product* een naam.
4. Onder **AttachedFiles** in het rechtermenu klikt u op **Gegevens selecteren**.
5. Upload en selecteer *Product.xlsx*.
6. Selecteer de *Product*-tabel in het venster **Selecteer gegevens uit**.



Selecteer alle velden en zorg ervoor dat **Ingesloten veldnamen** onder **Veldnamen** is geselecteerd om de namen van de tabelvelden mee te nemen bij het laden van de gegevens.

7. Klik op **Script invoegen**.

Uw script zou er als volgt moeten uitzien:

```
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. Klik op **Gegevens laden**.
9. Open de **gegevensmodelviewer**. Het gegevensmodel ziet er als volgt uit:  
*Product-tabel met Product-veld en een veld voor elk van de maanden*

Product
Product
Jan 2014
Feb 2014
Mar 2014
Apr 2014
May 2014
Jun 2014

10. Klik op het tabblad *Product* in de **editor voor laden van gegevens**.
11. Voer boven de LOAD-opdracht het volgende in:  
`CrossTable(Month, Sales)`
12. Klik op **Gegevens laden**.
13. Open de **gegevensmodelviewer**. Het gegevensmodel ziet er als volgt uit:  
*Product-tabel met velden Product, Month en Sales*

Product
Product
Month
Sales

Let op: invoergegevens hebben normaal gesproken slechts één kolom als kwalificerend veld, als een interne sleutel (*Product* in het bovenstaande voorbeeld). Maar u kunt er meerdere hebben. Als dat het geval is, moeten alle kwalificerende velden worden opgesomd vóór de

### 3 Gegevens transformeren

kenmerk velden in de LOAD-opdracht en moet de derde parameter voor het prefix Crosstable worden gebruikt voor het definiëren van het aantal kwalificerende velden. U kunt geen voorafgaande LOAD of een prefix vóór het trefwoord Crosstable te hebben. U kunt echter gebruik maken van automatisch aaneenschakelen.

In een tabel in Qlik Sense zien uw gegevens er als volgt uit:

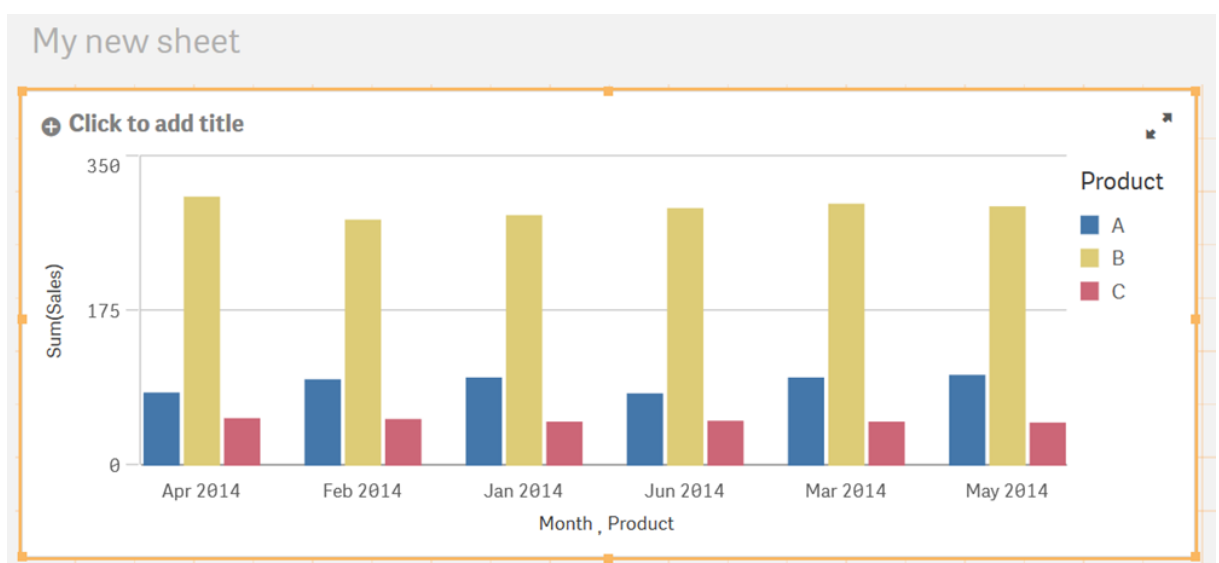
*Tabel toont gegevens geladen met het prefix Crosstable*

My new sheet

Click to add title			
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	
C	Jan 2014	50	

U kunt nu bijvoorbeeld een staafdiagram maken met behulp van de gegevens:

*Staafdiagram toont gegevens geladen met de prefix Crosstable*





Zie voor meer informatie over Crosstable deze blogpost in Qlik Community: [De crosstable-load](#). De gedragingen worden besproken in de context van QlikView 12. De logica geldt echter net zo voor Qlik Sense.

De numerieke interpretatie werkt niet voor de kenmerkelden. Dit betekent dat als u maanden hebt als kolomkoppen, deze niet automatisch worden geïnterpreteerd. Dit kan worden opgelost door gebruik te maken van het prefix Crosstable om een tijdelijke tabel te maken en hierop een tweede bewerking uit te voeren om de interpretaties te maken zoals te zien in het volgende voorbeeld:

Let op: dit is slechts een voorbeeld. Er zijn geen bijbehorende oefeningen in 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;
```

### De geheugencache leegmaken

U kunt tabellen die u maakt verwijderen om de geheugencache leeg te maken. Als u laadt in een tijdelijke tabel zoals in de voorgaande sectie, kunt u deze het beste verwijderen als u deze niet langer nodig hebt. Bijvoorbeeld:

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

U kunt ook velden neerzetten. Bijvoorbeeld:

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

Zoals u kunt zien, zijn de trefwoorden TABLE en FIELD, en kunnen deze in het enkelvoud of meervoud staan.

## 3.2 Tabellen samenvoegen met Join en Keep

Een join is een bewerking waarmee twee tabellen worden samengevoegd tot één tabel. De records van de resulterende tabel zijn een combinatie van de records in de oorspronkelijke tabellen. Deze combinaties zijn doorgaans zodanig dat de twee records die een combinatie vormen in de resulterende tabel een gemeenschappelijke waarde hebben voor een of meerdere gemeenschappelijke velden: een zogeheten natuurlijke join. In Qlik Sense kunt u joins maken in het script, resulterend in logische tabellen.

Het is mogelijk tabellen al in het script samen te voegen. De Qlik Sense-logica ziet dan niet de afzonderlijke tabellen, maar het resultaat van de join: één interne tabel. In sommige situaties is dit nodig, maar er zijn nadelen:

- De geladen tabellen worden vaak groter en Qlik Sense werkt trager.
- Een aantal gegevens kan verloren gaan: de frequentie (het aantal records) binnen de oorspronkelijke tabel is mogelijk niet meer beschikbaar.

De functie Keep reduceert een of beide tabellen tot de gemeenschappelijke tabelgegevens voordat de tabellen in Qlik Sense worden opgeslagen. Dit vermindert het aantal gevallen waarbij expliciete joins nodig zijn.



*In deze documentatie wordt de term join meestal gebruikt voor koppelingen die zijn gemaakt voordat de interne tabellen worden gemaakt. De associatie die plaatsvindt nadat de logische tabellen zijn gemaakt, is echter in wezen ook een join.*

### Join

De eenvoudigste manier om een join te maken is met het prefix Join in het script, waarmee de interne tabel met een andere benoemde tabel of met de laatste eerder gemaakte tabel wordt samengevoegd. Het gaat hier om een outer join, waarbij alle mogelijke combinaties van waarden uit de twee tabellen worden gemaakt.

#### Voorbeeld:

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

De resulterende interne tabel bevat de velden a, b, c en d. Het aantal records is afhankelijk van de veldwaarden van de twee tabellen.



*De namen van de velden voor de koppeling moeten exact gelijk zijn. Het aantal velden voor de koppeling is willekeurig. De tabellen hebben doorgaans één of enkele velden gemeenschappelijk. Zonder gemeenschappelijke velden wordt het cartesische product van de tabellen weergegeven. Alle velden gemeenschappelijk is ook mogelijk, maar heeft meestal weinig zin. Tenzij een tabelnaam van een eerder geladen tabel is opgegeven in de Join-opdracht, wordt voor de prefix Join de laatste eerder gemaakte tabel gebruikt. De volgorde van de twee opdrachten is dus niet willekeurig.*

### Join gebruiken

Het expliciete prefix Join in de scripttaal van Qlik Sense zorgt voor een volledige koppeling van de twee tabellen. Het resultaat is één tabel. Zulke joins resulteren vaak in zeer grote tabellen.

### Doe het volgende:

1. Open de app *Zelfstudie voor geavanceerd scriptgebruik*.
2. Voeg een nieuwe scriptsectie toe in de **Editor voor het laden van gegevens**.
3. Roep de sectie *Transactions* aan.
4. Onder **AttachedFiles** in het rechtermenu klikt u op **Gegevens selecteren**.
5. Upload en selecteer *Transactions.csv*.



Selecteer alle velden en zorg ervoor dat **Ingesloten veldnamen** onder **Veldnamen** is geselecteerd om de namen van de tabelvelden mee te nemen bij het laden van de gegevens.

6. In het venster **Selecteer gegevens uit** klikt u op **Script invoeren**.
7. Upload en selecteer *Salesman.xlsx*.
8. In het venster **Selecteer gegevens uit** klikt u op **Script invoeren**.

Uw script zou er als volgt moeten uitzien:

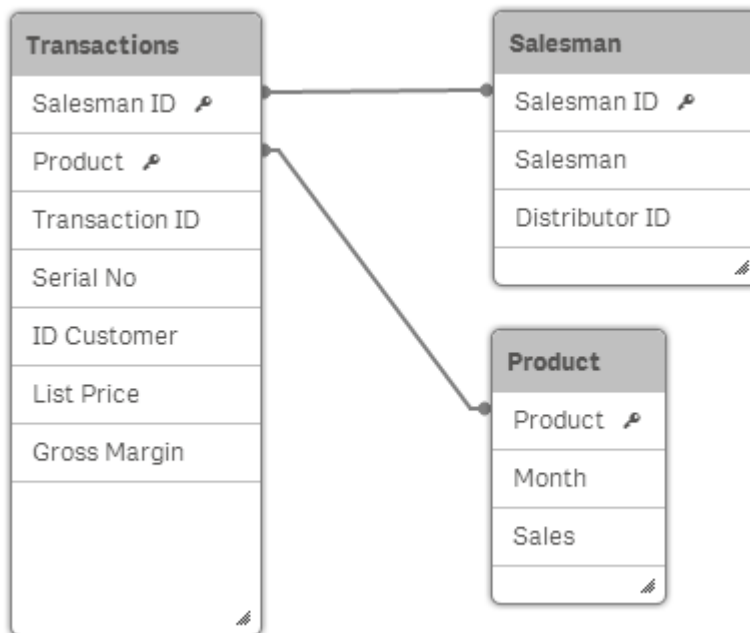
```
LOAD
    "Transaction ID",
    "Salesman ID",
    Product,
    "Serial No",
    "ID Customer",
    "List Price",
    "Gross Margin"
FROM [lib://AttachedFiles/Transactions.csv]
(txt, codepage is 28591, embedded labels, delimiter is ',', msq);

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

9. Klik op **Gegevens laden**.
10. Open de **gegevensmodelviewer**. Het gegevensmodel ziet er als volgt uit:

### 3 Gegevens transformeren

Gegevensmodel: Tabellen *Transactions*, *Salesman* en *Product*



Als de tabellen *Transactions* en *Salesman* gescheiden zijn, levert dit echter mogelijk niet het vereiste resultaat op. Het is wellicht beter om de twee tabellen samen te voegen.

#### Doe het volgende:

1. Om de samengevoegde tabel een naam te geven plaatst u de volgende regel boven de eerste LOAD-opdracht:  
Transactions:
2. Als u de tabellen *Transactions* en *Salesman* wilt samenvoegen, voegt u de volgende regel toe boven de tweede LOAD-opdracht:  
Join(Transactions)

Uw script zou er als volgt moeten uitzien:

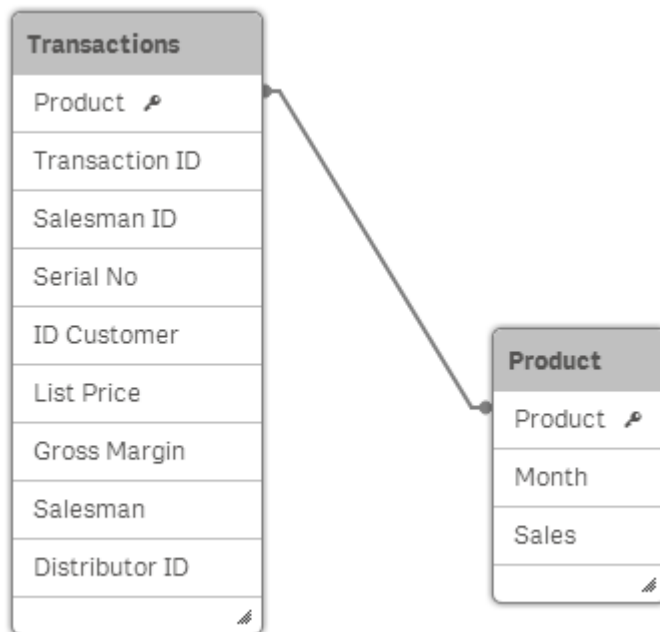
```
Transactions:
LOAD
    "Transaction ID",
    "Salesman ID",
    Product,
    "Serial No",
    "ID Customer",
    "List Price",
    "Gross Margin"
FROM [lib://AttachedFiles/Transactions.csv]
(txt, codepage is 28591, embedded labels, delimiter is ',', msq);

Join(Transactions)
LOAD
```

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

3. Klik op **Gegevens laden**.
4. Open de **gegevensmodelviewer**. Het gegevensmodel ziet er als volgt uit:

*Gegevensmodel: Tabellen Transactions en Product*



Alle velden van de tabellen *Transactions* en *Salesman* worden nu samengevoegd in een enkele tabel *Transactions*.



Zie voor meer informatie over het gebruik van Join deze blogposts in Qlik Community: [Samenvoegen of niet samenvoegen](#), [Toewijzen als alternatief voor samenvoegen](#). De gedragingen worden besproken in de context van QlikView 12. De logica geldt echter net zo voor Qlik Sense.

## Keep

Een van belangrijkste functies in Qlik Sense is de mogelijkheid om tabellen met elkaar te associëren in plaats van ze samen te voegen. Dit vereist minder geheugen, verhoogt de snelheid en biedt een enorme flexibiliteit. De functionaliteit van Keep is ontworpen om het aantal gevallen te beperken waarin expliciete joins moeten worden gebruikt.

Het prefix Keep tussen twee LOAD- of SELECT-opdrachten reduceert een of beide tabellen tot de gemeenschappelijke tabelgegevens voordat ze in Qlik Sense worden opgeslagen. Het prefix Keep moet worden voorafgegaan door een van de volgende trefwoorden Inner, Left of Right. De selectie



van records uit de tabellen gebeurt net zoals bij een overeenkomstige join. De twee tabellen worden echter niet samengevoegd maar als twee afzonderlijk benoemde tabellen in Qlik Sense opgeslagen.

### Inner

De prefixen Join en Keep in het load-script kunnen worden voorafgegaan door het prefix Inner.

Indien gebruikt voor Join, geeft het prefix aan dat een inner join moet worden gebruikt tussen de twee tabellen. De resulterende tabel bevat dan alleen combinaties tussen de twee tabellen met een volledige gegevensset van beide kanten.

Indien gebruikt voor Keep, geeft de opdracht aan dat de twee tabellen worden gereduceerd tot gegevens die passen bij beide tabellen, voordat deze worden opgeslagen in Qlik Sense.

#### Voorbeeld:

In deze voorbeelden gebruiken we de brontabellen *Table1* en *Table2*.

Onthoud dat dit slechts voorbeelden zijn. Er zijn geen bijbehorende oefeningen in Qlik Sense.

Table 1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

### Inner Join

Eerst voeren we een Inner Join uit op de tabellen, wat een *VTable* oplevert die één rij, het enige record dat in beide tabellen voorkomt, met gegevens uit beide tabellen bevat.

VTable:

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

VTable

A	B	C
1	aa	xx

### Inner Keep

Als we in plaats daarvan een Inner Keep uitvoeren, hebben we nog steeds twee tabellen. De twee tabellen zijn via het gemeenschappelijke veld *A* met elkaar verbonden.

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

VTab1

A	B
1	aa

VTab2

A	C
1	xx

### Left

De prefixen Join en Keep in het script voor het laden van gegevens kunnen worden voorafgegaan door het prefix left.

Indien gebruikt voor Join, geeft het prefix aan dat een left join moet worden gebruikt tussen de twee tabellen. De resulterende tabel bevat dan alleen combinaties tussen de twee tabellen met een volledige gegevensset van de eerste tabel.

Indien gebruikt voor Keep, geeft dit aan dat de tweede tabel wordt gereduceerd tot gegevens die passen bij de eerste tabel, voordat deze wordt opgeslagen in Qlik Sense.

#### Voorbeeld:

In deze voorbeelden gebruiken we de brontabellen *Table1* en *Table2*.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Eerst voeren we een Left Join uit op de tabellen, wat resulteert in een *VTable* die alle rijen uit *Table1* bevat, gecombineerd met velden uit de bijpassende rijen in *Table2*.

VTable:

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

VTable

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

Als we in plaats daarvan een Left Keep uitvoeren, hebben we nog steeds twee tabellen. De twee tabellen zijn via het gemeenschappelijke veld *A* met elkaar verbonden.

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

De prefixen Join en Keep in de scripttaal van Qlik Sense kunnen worden voorafgegaan door het prefix *right*.

### 3 Gegevens transformeren

---

Indien gebruikt voor Join, geeft het prefix aan dat een right join moet worden gebruikt tussen de twee tabellen. De resulterende tabel bevat dan alleen combinaties tussen de twee tabellen met een volledige gegevensset van de tweede tabel.

Indien gebruikt voor Keep, geeft dit aan dat de eerste tabel wordt gereduceerd tot gegevens die passen bij de eerste tabel, voordat deze wordt opgeslagen in Qlik Sense .

#### Voorbeeld:

In deze voorbeelden gebruiken we de brontabellen *Table1* en *Table2*.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Eerst voeren we een Right Join uit op de tabellen, wat resulteert in een *VTable* die alle rijen uit *Table2* bevat, gecombineerd met velden uit de bijpassende rijen in *Table1*.

VTable:

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

VTable

A	B	C
1	aa	xx
4	-	yy

Als we in plaats daarvan een Right Keep uitvoeren, hebben we nog steeds twee tabellen. De twee tabellen zijn via het gemeenschappelijke veld A met elkaar verbonden.

VTab1:

```
SELECT * from Table1;
```

VTab2:

```
right keep SELECT * from Table2;
```

VTab1

A	B
1	aa

VTab2

A	C
1	xx
4	yy

### 3.3 Inter-recordfuncties gebruiken: Peek, Previous en Exists

Deze functies worden gebruikt wanneer een waarde uit eerder geladen records van gegevens nodig is voor de evaluatie van de huidige record.

In dit gedeelte van de zelfstudie gaan we de functies Peek(), Previous() en Exists() nader onderzoeken.

#### Peek()

**Peek()** retourneert de waarde van een veld in een tabel voor een rij die al is geladen. Het rijnummer kan worden opgegeven, net als de tabel. Als er geen rijnummer is opgegeven, wordt het laatst geladen record gebruikt.

**Syntaxis:**

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

Rij moet een geheel getal zijn. 0 geeft de eerste record aan, 1 de tweede, enzovoort. Negatieve nummers geven de volgorde vanaf het eind van de tabel aan. -1 geeft de laatste gelezen record aan.

Als geen rij wordt vermeld, wordt -1 verondersteld.

*Tablename* is een tabellabel zonder de dubbele punt. Als geen *tablename* wordt vermeld, wordt de huidige tabel verondersteld. Als de tabelnaam buiten de **LOAD**-opdracht wordt gebruikt of als de tabelnaam verwijst naar een andere tabel, moet de *tablename* worden opgenomen.

#### Previous()

**Previous()** retourneert de waarde van de uitdrukking **expr** met gebruik van de gegevens uit de vorige invoerrecord die niet is genegeerd wegens een **where**-clausule. In de eerste record van een interne tabel retourneert de functie NULL.

**Syntaxis:**

`Previous(expression)`

De functie `Previous()` kan worden genest om toegang te krijgen tot eerdere records. Gegevens worden rechtstreeks opgehaald uit de invoerbron. Zo kan ook worden verwezen naar velden die niet zijn geladen in Qlik Sense en dus niet zijn opgeslagen in de bijbehorende database.

### Exists()

**Exists()** bepaalt of een specifieke veldwaarde al is geladen in het veld in het script voor het laden van gegevens. De functie retourneert `TRUE` of `FALSE`, zodat deze kan worden gebruikt in de **where**-clausule van een **LOAD**-opdracht of een **IF**-opdracht.

#### Syntaxis:

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

Het veld moet bestaan in de gegevens die tot nu toe zijn geladen door het script. *Expression* is een uitdrukking die wordt geëvalueerd tot de veldwaarde die moet worden gezocht in het opgegeven veld. Als niets is opgegeven, wordt uitgegaan van de waarde van de huidige record in het opgegeven veld.

### Peek() en Previous() gebruiken

In hun simpelste vorm worden `Peek()` en `Previous()` gebruikt voor het identificeren van specifieke waarden binnen een tabel. Hier vindt u een voorbeeld van de gegevens in de tabel *Employees* die u in deze oefening zult laden.

Voorbeeldgegevens van tabel *Employees*

Date	Aangenomen	Beëindigd
1/1/2011	6	0
2/1/2011	4	2
3/1/2011	6	1
4/1/2011	5	2

Op dit moment verzamelt deze uitsluitend gegevens voor maand, aangenomen werknemers en ontslagen werknemers, dus gaan we velden toevoegen voor *Employee Count* en *Employee Var*, met de functies `Peek()` en `Previous()`, om het maandelijkse verschil in het totale aantal werknemers te bekijken.

#### Doe het volgende:

1. Open de app *Zelfstudie voor geavanceerd scriptgebruik*.
2. Voeg een nieuwe scriptsectie toe in de **Editor voor het laden van gegevens**.
3. Roep de sectie *Employees* aan.
4. Onder **AttachedFiles** in het rechtermenu klikt u op **Gegevens selecteren**.
5. Upload en selecteer *Employees.xlsx*.



Zorg ervoor dat *Embedded field names* onder *Field names* is geselecteerd om de namen van de tabelvelden mee te nemen bij het laden van de gegevens.

6. In het venster **Selecteer gegevens uit** klikt u op **Script invoeren**.

Uw script zou er als volgt moeten uitzien:

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

7. Wijzig het script zodat dit er nu als volgt uit ziet:

```
[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);
```

De datums in het veld *Date* in het Excel-werkblad hebben de indeling MM/DD/JJJJ. De functie *Date* wordt toegepast op het veld *Date* om ervoor te zorgen dat de datums correct worden geïnterpreteerd met behulp van de indeling van de systeemvariabelen.

Met de functie *Peek()* kunt u elke willekeurige waarde identificeren die is geladen voor een gedefinieerd veld. In de uitdrukking kijken we eerst of de *rowno()* gelijk is aan 1. Als dit gelijk is aan 1, bestaat er geen *Employee Count*, dus vullen we het veld met het verschil van *Hired* min *Terminated*.

Als de *rowno()* groter is dan 1, kijken we naar de *Employee Count* voor de afgelopen maand en tellen we dat getal op bij het verschil tussen *Hired* en *Terminated* werknemers voor die maand.

Zoals u kunt zien, maken we in de functie *Peek()* gebruik van een (-1). Dit geeft aan dat Qlik Sense naar de record boven de huidige record moet kijken. Als de (-1) niet is opgegeven, gaat Qlik Sense ervan uit dat u de voorgaande record wilt bekijken.

8. Voeg het volgende toe aan het eind van uw script:

```
[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];
```

Met de functie Previous() kunt u de laatste waarde identificeren die is geladen voor een gedefinieerd veld. In de uitdrukking kijken we eerst of de rowno() gelijk is aan 1. Als deze gelijk is aan 1, weten we dat er geen Employee Var is omdat er geen record voor *Employee Count* is voor de afgelopen maand. We voeren dus simpelweg 0 in als waarde.

Als rowno() groter is dan 1, weten we dat er een *Employee Var* is, dus kijken we naar *Employee Count* voor afgelopen maand en trekken dat getal af van *Employee Count* voor de huidige maand om de waarde te maken in het veld *Employee Var*.

Uw script zou er als volgt moeten uitzien:

```
[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. Klik op **Gegevens laden**.

In een nieuw werkblad in het app-overzicht maakt u een tabel met behulp van *Date*, *Hired*, *Terminated*, *Employee Count* en *Employee Var* als de kolommen van de tabel. De resulterende tabel zou er als volgt uit moeten zien:



### 3 Gegevens transformeren

De tabel na het gebruik van Peek en Previous in het script

My new sheet

Click to add title				
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() en Previous() staan u toe gedefinieerde rijen binnen in een tabel te specificeren. Het grootste verschil tussen de twee functies is dat de functie Peek() de gebruiker in staat stelt een veld te bekijken dat niet eerder in het script was geladen, terwijl de functie Previous() alleen een eerder geladen veld kan bekijken. Previous() werkt op basis van de invoer van de LOAD-opdracht, terwijl Peek() werkt op basis van de uitvoer van de LOAD-opdracht. (Hetzelfde als het verschil tussen RecNo() en RowNo()) Dit betekent dat de twee functies zich anders gedragen als u een Where-clausule hebt.

Dus de functie Previous() is beter als u de huidige waarde in vergelijking met de vorige waarde moet weergeven. In het voorbeeld hebben we het verschil in aantallen werknemers van maand tot maand berekend.

De functie Peek() is beter als u zich richt op een veld dat nog niet eerder in de tabel is geladen of als u een specifieke rij moet weergeven. Dit hebben we laten zien in het voorbeeld waarbij we *Employee Count* berekenden door *Employee Count* van de voorafgaande maand te bekijken en het verschil tussen aangenomen en ontslagen werknemers voor de huidige maand daarbij op te tellen. Onthoud dat *Employee Count* geen veld in het oorspronkelijke bestand vormde



Zie voor meer informatie over wanneer u Peek() en Previous() gebruikt deze blogpost in Qlik Community: [Peek\(\) vs Previous\(\) – When to Use Each](#). De gedragingen worden besproken in de context van QlikView 12. De logica geldt echter net zo voor Qlik Sense.

### Exists() gebruiken

De functie Exists() wordt vaak gebruikt met de Where-clausule in het script om gegevens te laden als er al gerelateerde gegevens in het gegevensmodel zijn geladen.

In het volgende voorbeeld gaan we tevens de functie `Dual()` gebruiken om numerieke waarden toe te wijzen aan tekenreeksen.

### Doe het volgende:

1. Maak een nieuwe app en geef deze een naam.
2. Voeg een nieuwe scriptsectie toe in de **Editor voor het laden van gegevens**.
3. Roep de sectie *People* aan.
4. Voer het volgende script in:

```
//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;
```

### 3 Gegevens transformeren

```
//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;
```

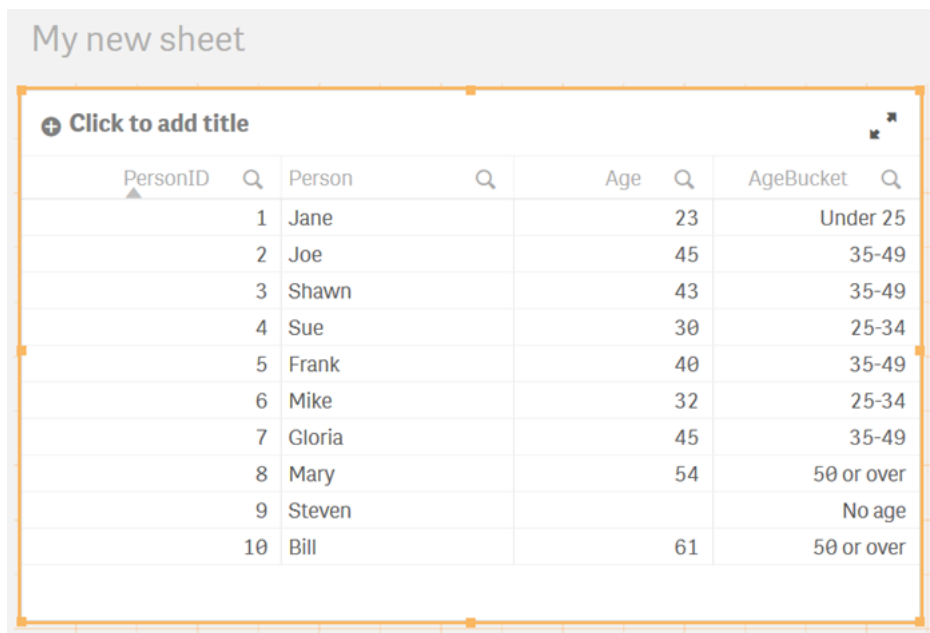
#### 5. Klik op **Gegevens laden**.

In het script worden de velden *Age* en *AgeBucket* alleen geladen als *PersonID* al is geladen in het gegevensmodel.

Zoals u ziet, worden in de tabel *AgeTemp* leeftijden vermeld voor *PersonID* 11 en 12, maar aangezien deze id's niet in het gegevensmodel zijn geladen (in de tabel *People*), worden zij uitgesloten door de *Where Exists(PersonID)*-clausule. Deze clausule kan ook als volgt worden geschreven: *Where Exists(PersonID, PersonID)*.

De uitvoer van het script ziet er als volgt uit:

*De tabel na het gebruik van Exists in het script*



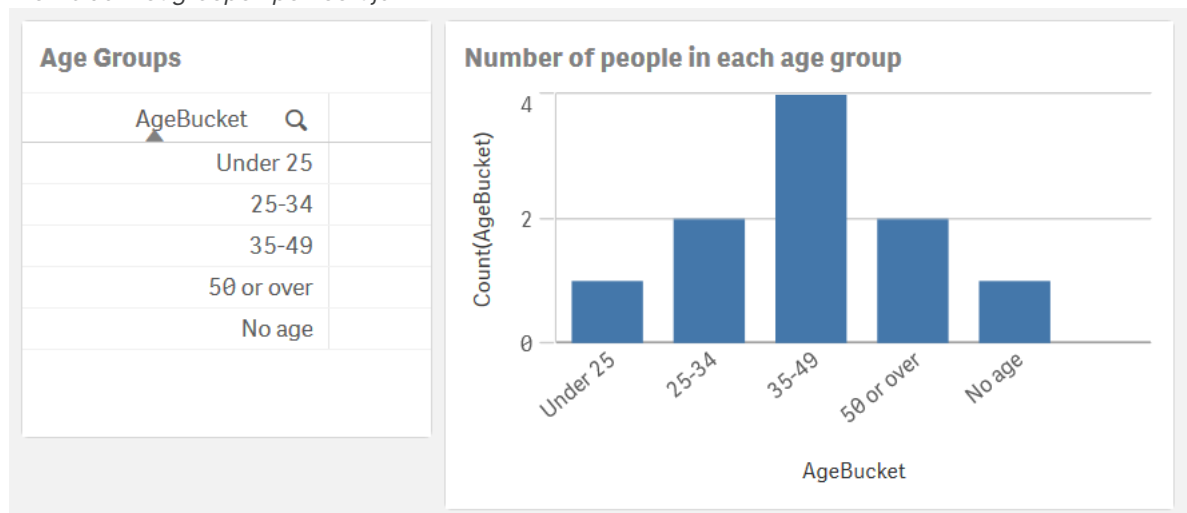
PersonID	Person	Age	AgeBucket
1	Jane	23	Under 25
2	Joe	45	35-49
3	Shawn	43	35-49
4	Sue	30	25-34
5	Frank	40	35-49
6	Mike	32	25-34
7	Gloria	45	35-49
8	Mary	54	50 or over
9	Steven		No age
10	Bill	61	50 or over

Als geen van de waarden voor *PersonID* in de tabel *AgeTemp* in het gegevensmodel is geladen, zijn de velden *Age* en *AgeBucket* niet samengevoegd tot de tabel *People*. Het gebruik van de functie *Exists()* kan helpen voorkomen dat er weesrecords/-gegevens ontstaan in het gegevensmodel. Met andere woorden dat de velden *Age* en *AgeBucket* geen bijbehorende personen hebben.

6. Maak een nieuw werkblad en geef dit een naam.
7. Open het nieuwe werkblad en klik op **Het werkblad bewerken**.
8. Voeg een standaardtabel toe aan het werkblad met de dimensie *AgeBucket* en geef de visualisatie de naam *Leeftijdsgroepen*.
9. Voeg een staafdiagram toe aan het werkblad met de dimensie *AgeBucket* en de meting *Count([AgeBucket])*. Geef de visualisatie *Number of people in each age group* een naam.
10. Pas de eigenschappen van de tabel en het staafdiagram aan uw voorkeuren aan en klik daarna op **Gereed**.

U zou nu een werkblad moeten hebben dat hierop lijkt:

*Werkblad met groepen per leeftijd*



De functie *Dual()* is nuttig in het script of in een diagramuitdrukking, als een numerieke waarde moet worden toegewezen aan een tekenreeks.

In het script hierboven hebt u een toepassing die leeftijden laadt en u hebt besloten om deze leeftijden op te nemen in buckets zodat u visualisaties kunt maken op basis van de leeftijdsbuckets en deze kunt vergelijken met de daadwerkelijke leeftijden. Er is een bucket voor mensen jonger dan 25, mensen tussen 25 en 35 enzovoort. Via de functie *Dual()* kunnen aan de leeftijdsbuckets een numerieke waarde worden toegewezen die kan worden gebruikt voor het sorteren van de leeftijdsbuckets in een lijstvak of in een diagram. Dus, net als in het werkblad van de app, wordt "Geen leeftijd" aan het einde van de lijst geplaatst door de sorteerfunctie.



Zie voor meer informatie over *Exists()* en *Dual()* deze blogpost in Qlik Community: [Dual en Exists – nuttige functies](#)

## 3.4 Afstemmingsintervallen en iteratief laden

Het prefix `IntervalMatch` bij een opdracht `LOAD` of `SELECT` wordt gebruikt om discrete numerieke waarden aan een of meer numerieke intervallen te koppelen. Dit is een zeer krachtige functie die u bijvoorbeeld in productieomgevingen kunt gebruiken.

### Het prefix `IntervalMatch()` gebruiken

De meest elementaire intervalafstemming is wanneer u een lijst met getallen of datums (gebeurtenissen) hebt in de ene tabel en een lijst met intervallen in een tweede. Het doel is om de twee tabellen te koppelen. Gewoonlijk is dit een veel-op-veel-relatie, oftewel één interval kan vele bijbehorende datums hebben en een datum kan tot vele intervallen behoren. Om dit op te lossen, moet u een overbruggingstabel maken tussen de twee oorspronkelijke tabellen. Er zijn verschillende manieren om dit te doen.

De eenvoudigste manier om dit probleem op te lossen in Qlik Sense is om het prefix `IntervalMatch()` vóór een opdracht `LOAD` of `SELECT` te plaatsen. De `LOAD`-/`SELECT`-opdracht hoeft slechts twee velden te bevatten, namelijk de velden "From" en "To" voor het definiëren van de intervallen. Het prefix `IntervalMatch()` genereert vervolgens alle combinaties tussen de geladen intervallen en een eerder geladen numeriek veld, dat is opgegeven als parameter voor het prefix.

#### Doe het volgende:

1. Maak een nieuwe app en geef deze een naam.
2. Voeg een nieuwe scriptsectie toe in de **Editor voor het laden van gegevens**.
3. Roep de secties *Events* aan.
4. Onder **AttachedFiles** in het rechtermenu klikt u op **Gegevens selecteren**.
5. Upload en selecteer *Events.txt*.
6. In het venster **Selecteer gegevens uit** klikt u op **Script invoeren**.
7. Upload en selecteer *Intervals.txt*.
8. In het venster **Selecteer gegevens uit** klikt u op **Script invoeren**.
9. Noem in het script de eerste tabel *Events* en de tweede tabel *Intervals*.
10. Voeg aan het einde van het script een `IntervalMatch` toe om een derde tabel te maken en de eerste twee tabellen te overbruggen:

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

11. Uw script zou er als volgt moeten uitzien:

```
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)

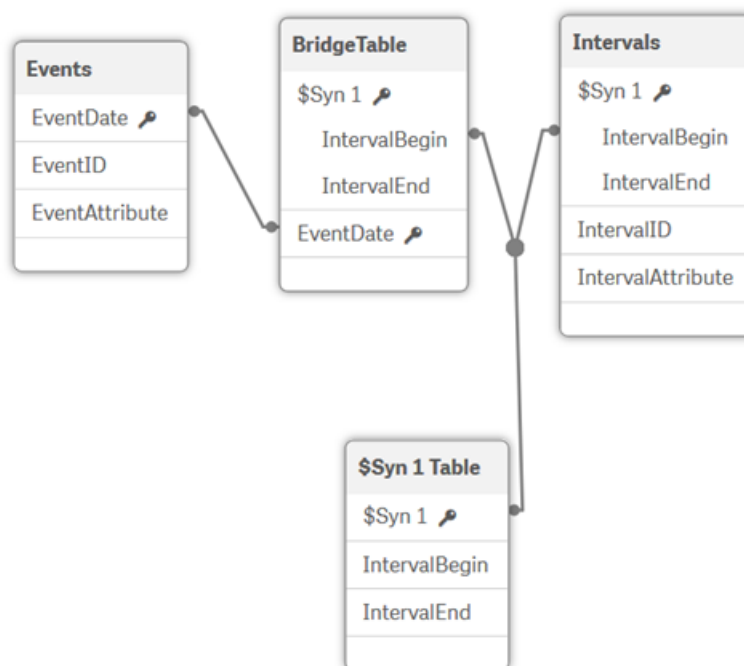
LOAD distinct IntervalBegin, IntervalEnd

Resident Intervals;

12. Klik op **Gegevens laden**.

13. Open de **gegevensmodelviewer**. Het gegevensmodel ziet er als volgt uit:

*Gegevensmodel: Tabellen Events, BridgeTable, Intervals en \$Syn1*



Het gegevensmodel bevat een samengestelde sleutel (de velden *IntervalBegin* en *IntervalEnd*) die zichzelf manifesteert als een synthetische sleutel in Qlik Sense.

De basistabellen zijn:

- De tabel *Events* die precies één record per gebeurtenis bevat.
- De tabel *Intervals* die exact één record per interval bevat.

- De overbruggingstabel die exact één record per combinatie van gebeurtenis en interval bevat en die de twee eerdere tabellen aan elkaar koppelt.

Een gebeurtenis kan overigens bij verschillende intervallen behoren als de intervallen elkaar overlappen. En een interval kan natuurlijk verschillende gebeurtenissen hebben die hier deel van uitmaken.

Dit gegevensmodel is optimaal in die zin dat het genormaliseerd en compact is. De tabel *Events* en de tabel *Intervals* zijn beide ongewijzigd en bevatten het oorspronkelijke aantal records. Alle berekeningen in Qlik Sense die worden uitgevoerd op deze tabellen, zoals `Count(EventID)`, werken en worden op correcte wijze geëvalueerd.



Zie voor meer informatie over `IntervalMatch()` deze blogpost in Qlik Community: [IntervalMatch\(\)](#) gebruiken

### Een While-lus en iteratief laden van `IterNo()` gebruiken

U kunt bijna dezelfde overbruggingstabel verkrijgen via een While-lus en `IterNo()`, waarmee telbare waarden worden gemaakt tussen de boven- en ondergrenzen van het interval.

Een lus binnen in de opdracht `LOAD` kan worden gemaakt met behulp van de clause `While`. Bijvoorbeeld:

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

Een dergelijke `LOAD`-opdracht voert een lusbewerking uit op elke invoerrecord en laadt deze steeds opnieuw zolang de uitdrukking in de `While`-clause waar is. De functie `IterNo()` geeft "1" retour bij de eerste iteratie, "2" bij de tweede enzovoort.

U hebt een primaire sleutel voor de intervallen, de `IntervalID`, dus het enige verschil in het script schuilt in de manier waarop de overbruggingstabel wordt gemaakt:

#### Doe het volgende:

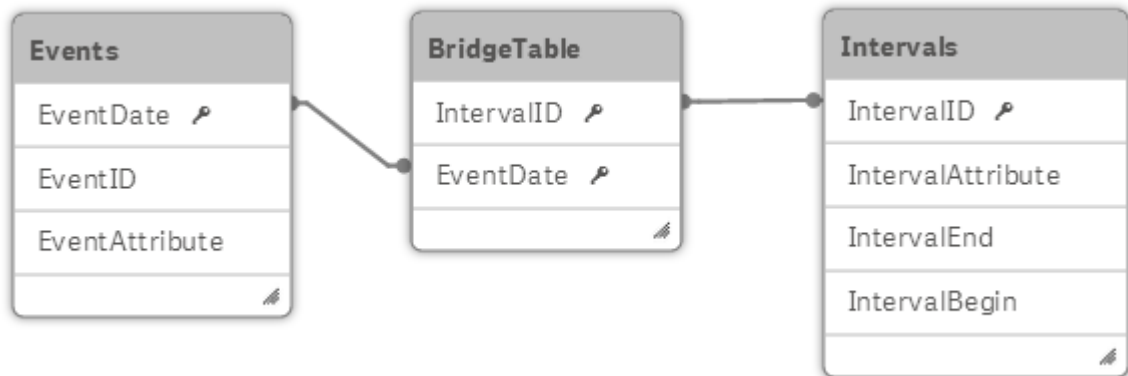
1. Vervang de bestaande opdrachten `Bridgetable` door het volgende script:

```
BridgeTable:
LOAD distinct * where Exists(EventDate);
LOAD IntervalBegin + IterNo() - 1 as EventDate, IntervalID
  Resident Intervals
  while IntervalBegin + IterNo() - 1 <= IntervalEnd;
```

2. Klik op **Gegevens laden**.
3. Open de **gegevensmodelviewer**. Het gegevensmodel ziet er als volgt uit:

### 3 Gegevens transformeren

*Gegevensmodel: Tabellen Events, BridgeTable en Intervals*



Over het algemeen is de oplossing met drie tabellen de beste, omdat het veel-op-veel-relaties mogelijk maakt tussen intervallen en gebeurtenissen. Maar een veelvoorkomende situatie is dat u weet dat een gebeurtenis slechts tot een enkel interval kan behoren. In dergelijke gevallen is een overbruggingstabel echt niet nodig. De *IntervalID* kan rechtstreeks in de tabel van de gebeurtenis worden opgeslagen. Er zijn verschillende manieren om dit voor elkaar te krijgen, maar de meest nuttige is samenvoeging van Bridgetable met de tabel *Events*.

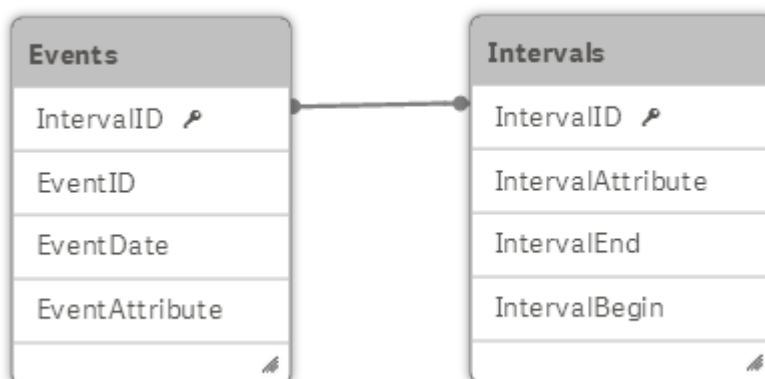
4. Voeg het volgende toe aan het eind van uw script:

```
Join (Events)  
LOAD EventDate, IntervalID  
Resident BridgeTable;
```

```
Drop Table BridgeTable;
```

5. Klik op **Gegevens laden**.
6. Open de **gegevensmodelviewer**. Het gegevensmodel ziet er als volgt uit:

*Gegevensmodel: Tabellen Events en Intervals*





### Open en gesloten intervallen

Of een interval open of gesloten is wordt bepaald door de eindpunten, afhankelijk van of deze al dan niet in het interval zijn opgenomen.

- Als de eindpunten zijn opgenomen, is het een gesloten interval:  
 $[a,b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$
- Als de eindpunten niet zijn opgenomen, is het een open interval:  
 $]a,b[ = \{x \in \mathbb{R} \mid a < x < b\}$
- Als één eindpunt is opgenomen, is het een halfopen interval:  
 $[a,b[ = \{x \in \mathbb{R} \mid a \leq x < b\}$

Als u een situatie hebt waarin de intervallen elkaar overlappen en een getal bij meer dan één interval kan behoren, moet u gewoonlijk gesloten intervallen gebruiken.

In sommige situaties wilt u echter geen overlappende intervallen en wilt u dat een getal bij slechts één interval behoort. U komt dit in de problemen als één punt het einde van het ene interval en tegelijkertijd het begin van het volgende vormt. Een getal met deze waarde wordt aan beide intervallen toegewezen. Daarom wilt u halfopen intervallen.

Een praktische oplossing voor dit probleem is het aftrekken van een zeer kleine hoeveelheid van de eindwaarde van alle intervallen, waardoor gesloten, maar niet-overlappende intervallen ontstaan. Als uw getallen datums zijn, kunt u dit het gemakkelijkst doen door de functie `DayEnd()` te gebruiken, die de laatste milliseconde van de dag retourneert:

```
Intervals:  
LOAD..., DayEnd(IntervalEnd - 1) as IntervalEnd From Intervals;
```

U kunt ook handmatig een kleine hoeveelheid aftrekken. Als u dat doet, moet u ervoor zorgen dat de afgetrokken hoeveelheid niet te klein is omdat de bewerking wordt afgerond op 52 significante binaire cijfers (14 decimale cijfers). Als u een te kleine hoeveelheid gebruikt, is het verschil niet significant en krijgt u weer het oorspronkelijke getal.

## 4 Gegevens opschonen

Er zijn momenten waarop de brongegevens die u in Qlik Sense laadt niet de vorm hebben zoals u ze in de Qlik Sense-app wilt opslaan. Qlik Sense biedt u een keur aan functies en opdrachten waarmee u uw gegevens kunt transformeren in een indeling die geschikt is voor uw doeleinden.

Mapping kan worden gebruikt in een Qlik Sense-script om veldwaarden of -namen te vervangen of wijzigen als het script wordt uitgevoerd, dus kan mapping worden gebruikt voor het opschonen van gegevens en het meer consistent maken hiervan of om een veldwaarde geheel of gedeeltelijk te vervangen.

Als u gegevens uit verschillende tabellen laadt, zijn de namen van veldwaarden die hetzelfde betekenen niet altijd consistent. Aangezien dit gebrek aan consistentie associaties in de weg staat, moet dit probleem worden opgelost. Dit kunt u op elegante wijze doen door een toewijzingstabel te maken om veldwaarden te vergelijken.

### 4.1 Tabellen toewijzen

Tabellen geladen met Mapping load of Mapping select worden anders behandeld dan andere tabellen. Zij worden opgeslagen in een afzonderlijk gebied van het geheugen en uitsluitend als toewijzingstabellen gebruikt tijdens de uitvoering van het script. Nadat het script is uitgevoerd, worden deze tabellen automatisch verwijderd.

#### Regels:

- Een toewijzingstabel moet uit twee kolommen bestaan, de eerste met vergelijkingswaarden en de tweede met de gewenste toewijzingswaarden.
- De twee kolommen moeten een naam hebben, waarbij de namen op zichzelf niet van belang zijn. De kolomnamen hebben geen verbinding met veldnamen in reguliere interne tabellen.

### 4.2 Functies en opdrachten Mapping

De volgende functies/opdrachten voor mapping komen aan bod in deze zelfstudie:

- Prefix Mapping
- ApplyMap()
- MapSubstring()
- Opdracht Map ... Using
- Opdracht Unmap

### 4.3 Prefix Mapping

Het prefix Mapping wordt gebruikt in een script om een toewijzingstabel te maken. Deze toewijzingstabel kan vervolgens worden gebruikt met de functie `ApplyMap()`, de functie `MapSubstring()` of de opdracht `Map ... Using`.

#### Doe het volgende:

1. Maak een nieuwe app en geef deze een naam.
2. Voeg een nieuwe scriptsectie toe in de **Editor voor het laden van gegevens**.
3. Roep de sectie *Countries* aan.
4. Voer het volgende script in:

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

In de tabel *CountryMap* worden twee kolommen opgeslagen: *Country* en *NewCountry*. De kolom *Country* bevat de verschillende manieren waarop land is ingevoerd in het veld *Country*. In de kolom *NewCountry* wordt opgeslagen hoe de waarden worden toegewezen. Deze toewijzingstabel wordt gebruikt voor het opslaan van consistente *US* landwaarden in het veld *Country*. Als bijvoorbeeld *U.S.A.* is opgeslagen in het veld *Country*, wijst u het toe aan *US*.

### 4.4 Functie ApplyMap()

Gebruik `ApplyMap()` om gegevens in een veld te vervangen op basis van een eerder gemaakte toewijzingstabel. De toewijzingstabel moet worden geladen voordat de functie `ApplyMap()` kan worden gebruikt. De gegevens in de tabel *Data.xlsx* die u gaat laden, zien er als volgt uit:

#### Gegevenstabel

ID	Naam	Land:	Code
1	John Black	VS	SDFGBS1DI
2	Steve Johnson	V.S.	2ABC
3	Mary White	Verenigde Staten	DJY3DFE34
4	Susan McDaniels	u	DEF5556
5	Dean Smith	VS	KSD111DKFJ1

U ziet dat het land op verschillende manieren is gespecificeerd. Teneinde het landveld consistent te maken, wordt de toewijzingstabel geladen en wordt vervolgens de functie **`ApplyMap()`** gebruikt.

### Doe het volgende:

1. Onder het script dat u hierboven hebt ingevoerd, selecteert en laadt u *Data.xlsx*. Voeg vervolgens het script in.
2. Voeg het volgende toe boven de nieuw gemaakte opdracht LOAD:

Data:

Uw script zou er als volgt moeten uitzien:

```
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. Wijzig als volgt de regel die country, bevat:  
ApplyMap('CountryMap', Country) as Country,

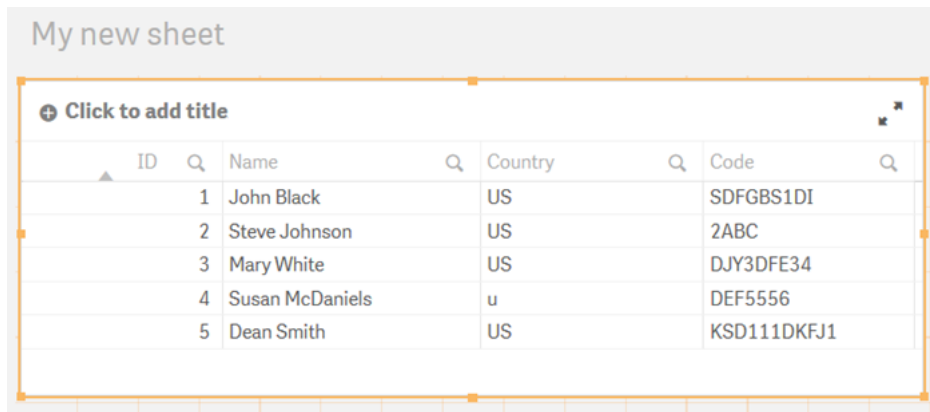
De eerste parameter van de functie ApplyMap() bevat de toewijzingsnaam tussen enkele aanhalingstekens. De tweede parameter is het veld met de gegevens die moeten worden vervangen.

4. Klik op **Gegevens laden**.

De resulterende tabel ziet er als volgt uit:

## 4 Gegevens opschonen

Tabel met de gegevens geladen met de functie `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 verschillende spellingen van *United States* zijn allemaal gewijzigd in *US*. Er is één record die niet correct is gespeld, dus heeft de functie `ApplyMap()` die veldwaarde niet gewijzigd. Via de functie `ApplyMap()` kunt u de derde parameter gebruiken om een standaarduitdrukking toe te voegen als de toewijzingstabel geen overeenkomstige waarde bevat.

- Voeg 'us' toe als derde parameter van de functie `ApplyMap()` om gevallen af te handelen waarbij de naam van het land verkeerd is gespeld:

`ApplyMap('CountryMap', Country, 'US') as Country,`

Uw script zou er als volgt moeten uitzien:

```
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);
```

- Klik op **Gegevens laden**.

De resulterende tabel ziet er als volgt uit:

Tabel met de gegevens geladen met de functie *ApplyMap*

My new sheet

Click to add title

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



Zie voor meer informatie over *ApplyMap()* deze blogpost in Qlik Community: [Niet samenvoegen - gebruik in plaats daarvan Applymap](#)

### 4.5 Functie MapSubstring()

Met de functie *MapSubstring()* kunt u onderdelen toewijzen aan een veld.

In de tabel die door *ApplyMap()* is gemaakt, willen we nu de getallen uitschrijven als tekst, dus wordt de functie *MapSubstring()* gebruikt voor het vervangen van de numerieke gegevens door tekst.

Hiertoe moet eerst een toewijzingstabel worden gemaakt.

#### Doe het volgende:

1. Voeg de volgende scriptregels toe na de sectie *CountryMap*, maar vóór de sectie *Data*.

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

In de tabel *CodeMap* worden de getallen 1 t/m 5 en 11 toegewezen.

2. Wijzig in de sectie *Data* van het script de opdracht code als volgt:

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

Uw script zou er als volgt moeten uitzien:

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

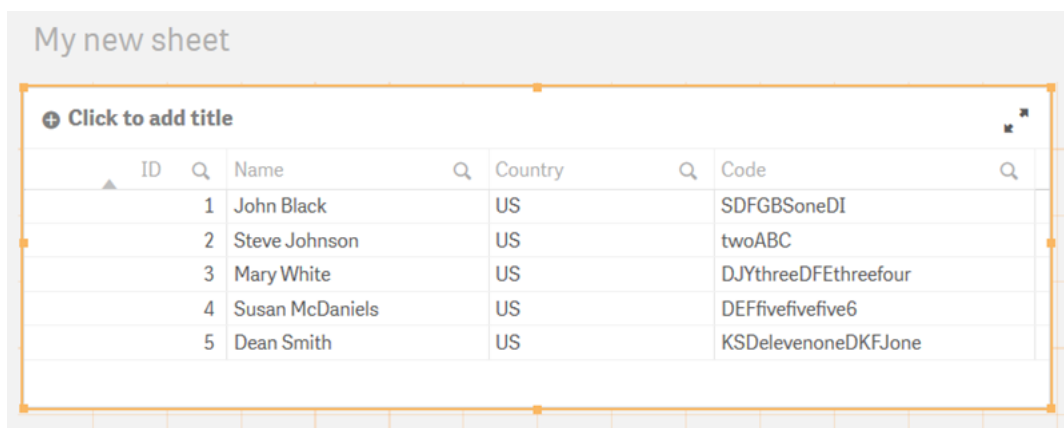
CodeMap:
MAPPING LOAD * INLINE [
    F1, F2
    1, one
    2, two
    3, three
    4, four
    5, five
    11, eleven
];

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. Klik op **Gegevens laden**.

De resulterende tabel ziet er als volgt uit:

*Tabel met de gegevens geladen met de functie 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 numerieke tekens zijn vervangen door tekst in het veld *Code*. Als een getal meer dan eenmaal voorkomt, zoals bij ID=3 en ID=4, wordt de tekst eveneens herhaald. ID=4. *Susan McDaniels* had een 6 in haar code. Aangezien 6 niet was toegewezen in de tabel *CodeMap*, blijft deze ongewijzigd. ID=5, *Dean Smith*, had 111 in zijn code. Dit is toegewezen als 'elevenone'.



Zie voor meer informatie over `MapSubstring()` deze blogpost in Qlik Community:  
[Toewijzen](#)

### 4.6 Map ... Using

De opdracht `Map ... Using` kan ook worden gebruikt om een toewijzing op een veld toe te passen. Dat werkt wel iets anders dan `ApplyMap()`. Terwijl `ApplyMap()` de toewijzing uitvoert telkens wanneer de veldnaam wordt aangetroffen, handelt `Map ... Using` de toewijzing af als de waarde wordt opgeslagen onder de veldnaam in de interne tabel.

Laten we eens een voorbeeld gaan bekijken. Stel dat we het veld *Country* meerdere keren in het script laden en een toewijzing willen toepassen telkens wanneer het veld wordt geladen. De functie `ApplyMap()` zou kunnen worden gebruikt zoals eerder geïllustreerd in deze zelfstudie of `Map ... Using` kan worden gebruikt.

Als `Map ... Using` wordt gebruikt, wordt de toewijzing toegepast op het veld als het veld wordt opgeslagen in de interne tabel. In het onderstaande voorbeeld wordt de toewijzing toegepast op het veld *Country* in de tabel *Data1*, maar wordt deze niet toegepast op het veld *Country2* in de tabel *Data2*. Dit komt doordat de opdracht `Map ... Using` alleen wordt toegepast op velden met de naam *Country*. Als het veld *Country2* wordt opgeslagen in de interne tabel, heeft dit niet langer de naam *Country*. Als u de toewijzing wilt toepassen op de tabel *Country2*, moet u de functie `ApplyMap()` gebruiken.

De opdracht `Unmap` beëindigt de opdracht `Map ... Using`, zodat, als *Country* zou worden geladen na de opdracht `Unmap`, de opdracht *CountryMap* niet zou worden toegepast.

#### Doe het volgende:

1. Vervang het script voor de tabel *Data* door het volgende:

```
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. Klik op **Gegevens laden**.

De resulterende tabel ziet er als volgt uit:



## 4 Gegevens opschonen

Tabel met de gegevens geladen met de functie Map ... Using

My new sheet

Click to add title			
ID	Name	Country	Country2
1	John Black	US	U.S.A.
2	Steve Johnson	US	U.S.
3	Mary White	US	United States
4	Susan McDaniels	u	u
5	Dean Smith	US	US

## 5 Hiërarchische gegevens verwerken

Hiërarchieën maken een belangrijk onderdeel uit van alle Business Intelligence-oplossingen. Zij worden gebruikt voor het beschrijven van dimensies die van nature verschillende granulariteitsniveaus bevatten. Sommige zijn simpel en intuïtief, terwijl andere complex zijn en veel denkwerk bij de modellering vereisen.

Vanaf de top van een hiërarchie tot aan de onderkant, zijn de leden in toenemende mate meer gedetailleerd. In een dimensie met de niveaus Markt, Land, Staat en Plaats bijvoorbeeld, verschijnt het lid Noord-, Midden- en Zuid-Amerika op het hoogste niveau van de hiërarchie, het lid V.S. op het tweede niveau, het lid California op het derde niveau en San Francisco op het laagste niveau. California is specifiekier dan V.S. en San Francisco is specifiekier dan California.

Het opslaan van hiërarchieën in een relationeel model is een veelvoorkomend probleem met meerdere oplossingen. Er zijn verschillende benaderingen:

- De horizontale hiërarchie
- Het nabijheidslijstmodel
- De padopsommingsmethode
- Het model van geneste gegevensverzamelingen
- De voorgangerslijst

In het kader van deze zelfstudie gaan we een voorgangerslijst maken omdat deze de hiërarchie presenteert in een vorm die direct bruikbaar is in een query. Meer informatie over de andere benaderingen is te vinden in Qlik Community.

### 5.1 Prefix Hierarchy

Het prefix Hierarchy is een scriptopdracht die u vóór een LOAD- of SELECT-opdracht plaatst waarmee een tabel met nabijgelegen knooppunten wordt geladen. De opdracht LOAD moet uit ten minste drie velden bestaan: Een id als unieke sleutel voor het knooppunt, een verwijzing naar het bovenliggende element en een naam.

Het prefix transformeert een geladen tabel in een tabel met uitgevouwen knooppunten; een tabel met een aantal extra kolommen - één voor elk niveau van de hiërarchie.

#### Doe het volgende:

1. Maak een nieuwe app en geef deze een naam.
2. Voeg een nieuwe scriptsectie toe in de **Editor voor het laden van gegevens**.
3. Roep de sectie *Wine* aan.
4. Onder **AttachedFiles** in het rechtermenu klikt u op **Gegevens selecteren**.
5. Upload en selecteer *Winedistricts.txt*.
6. Leeg in het venster **Gegevens selecteren van** de velden *Lbound* en *Rbound*, zodat deze niet worden geladen.

## 5 Hiërarchische gegevens verwerken

- Klik op **Script invoegen**.
- Voer boven de LOAD-opdracht het volgende in:  
Hierarchy (NodeID, ParentID, NodeName)

Uw script zou er als volgt moeten uitzien:

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

- Klik op **Gegevens laden**.
- Gebruik de sectie **Voorbeeld** van de **gegevensmodelviewer** om de resulterende tabel te bekijken.

De resulterende tabel met uitgevouwen knooppunten bevat exact hetzelfde aantal records als de brontabel: één per knooppunt. De tabel met uitgevouwen knooppunten is zeer praktisch omdat het aan een aantal vereisten voor analyse van een hiërarchie in een relationeel model voldoet:

- Alle namen van knooppunten bestaan in één en dezelfde kolom, zodat deze kan worden gebruikt voor zoekopdrachten.
- Daarnaast zijn de verschillende knooppuntniveaus elk in één veld uitgevouwen; velden die kunnen worden gebruikt in drill-down-groepen of als dimensies in draaitabellen.
- Daarnaast zijn de verschillende knooppuntniveaus uitgevouwen in één veld elk; velden die kunnen worden gebruikt in drill-down groepen.
- In de tabel kan tevens een pad worden opgenomen dat uniek is voor het knooppunt, met alle voorgangers in de juiste volgorde.
- De tabel kan de diepte van het knooppunt bevatten, oftewel de afstand tot het hoofdelement.

De resulterende tabel ziet er als volgt uit:

*Tabel toont een voorbeeld van de gegevens geladen met het prefix 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 Prefix HierarchyBelongsTo

Net zoals het prefix Hierarchy is het prefix HierarchyBelongsTo een scriptopdracht die u vóór een opdracht LOAD of SELECT plaatst om nabijgelegen knooppunten te laden.

Ook hier moet de opdracht LOAD ten minste drie velden bevatten: Een id die een unieke sleutel is voor het knooppunt, een referentie naar het bovenliggende element en een naam. Het prefix transformeert de geladen tabel in een voorgangerstabel, een tabel die elke combinatie van een voorganger en een afstammeling bevat als afzonderlijke record. Daarom is het heel gemakkelijk om alle voorgangers of alle afstammelingen van een specifiek knooppunt te vinden.

#### Doe het volgende:

1. Pas de opdracht Hierarchy in de **editor voor laden van gegevens** als volgt aan:  
HierarchyBelongsTo (NodeID, ParentID, NodeName, BelongsToID, BelongsTo)
2. Klik op **Gegevens laden**.
3. Gebruik de sectie **Voorbeeld** van de **gegevensmodelviewer** om de resulterende tabel te bekijken.

De voorgangerstabel voldoet aan een aantal vereisten voor analyse van een hiërarchie in een relationeel model:

- Als de knooppunt-id de afzonderlijke knooppunten vertegenwoordigt, vertegenwoordigt de voorganger-id de volledige structuren en substructuren van de hiërarchie.
- Alle knooppuntnamen bestaan zowel in de rol als knooppunten als in de rol als structuren, en beide kunnen worden gebruikt voor zoekopdrachten.
- De tabel kan het diepteverschil tussen de knooppunt diepte en de voorganger diepte bevatten, oftewel de afstand van het hoofdelement tot de substructuur.

De resulterende tabel ziet er als volgt uit:

## 5 Hiërarchische gegevens verwerken

Tabel toont gegevens geladen met het prefix *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		

### Autorisatie

Het is niet ongebruikelijk dat een hiërarchie wordt gebruikt voor autorisatie. Een voorbeeld is een organisatiehiërarchie. Alle managers moeten het recht hebben om alles te zien wat betrekking heeft op hun eigen afdeling, met inbegrip van subafdelingen. Maar zij hoeven niet noodzakelijkerwijs het recht te hebben om andere afdelingen te zien.

*Voorbeeld organisatiehiërarchie*



Dit betekent dat verschillende personen verschillende substructuren van de organisatie kunnen bekijken. De autorisatietabel zou er als volgt kunnen uitzien:

## 5 Hiërarchische gegevens verwerken

Autorisatietabel

TOEGANG	NTNAME	PERSOON	POSITIE	RECHTEN
GEBRUIKER	ACME\JRL	John	CPO	HR
GEBRUIKER	ACME\CAH	Carol	CEO	CEO
GEBRUIKER	ACME\JER	James	Hoofdtechnicus	Technicus
GEBRUIKER	ACME\DBK	Diana	CFO	Financiën
GEBRUIKER	ACME\RNL	Bob	COO	Sales
GEBRUIKER	ACME\LFD	Larry	CTO	Product

In dit geval mag *Carol* alles zien dat betrekking heeft op de *CEO* en eronder; *Larry* mag de organisatie *Product* zien en *James* mag alleen de organisatie *Engineering* zien.

### Voorbeeld:

De hiërarchie wordt vaak opgeslagen in een tabel met nabijgelegen knooppunten. In dit voorbeeld kunt u dit oplossen door de tabel met nabijgelegen knooppunten te laden met een `HierarchyBelongsTo` en het voorgangersveld de naam *Tree* te geven.

Als u `Section Access` wilt gebruiken, moet u een kopie in hoofdletters van *Tree* laden en dit nieuwe veld de naam `PERMISSIONS` geven. Ten slotte moet u de autorisatietabel laden. Deze laatste twee stappen kunt u uitvoeren met de volgende scriptregels. Onthoud dat de tabel `TempTrees` de tabel is die door de opdracht `HierarchyBelongsTo` is gemaakt.

Let op: dit is slechts een voorbeeld. Er is geen bijbehorende oefening in Qlik Sense.

Trees:

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

Section Access;

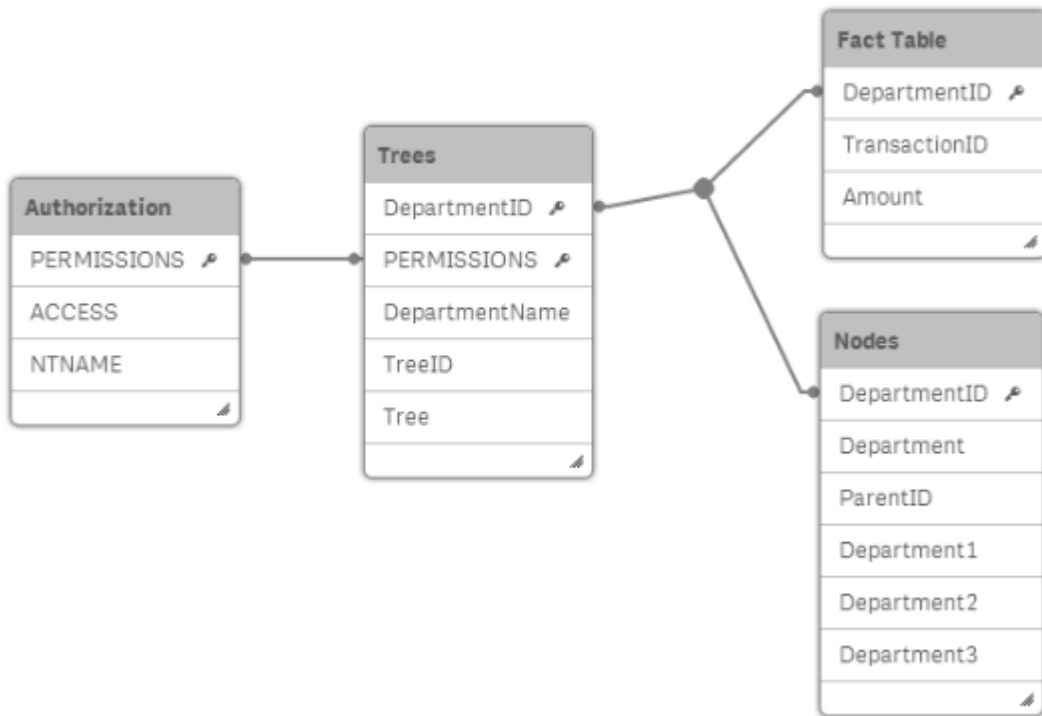
Authorization:

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

Dit voorbeeld zou in het volgende gegevensmodel resulteren:

## 5 Hiërarchische gegevens verwerken

Gegevensmodel: Tabellen Authorization, Trees, Fact en Nodes



## 6 QVD-bestanden

Een QVD (QlikView Data)-bestand is een bestand dat een tabel bevat met gegevens die vanuit Qlik Sense of QlikView 12 zijn geëxporteerd. QVD is een speciale Qlik-indeling die uitsluitend kan worden geschreven naar en gelezen door Qlik Sense of QlikView 12. De bestandsindeling is geoptimaliseerd om snel gegevens in een Qlik Sense-script te kunnen lezen, maar is toch zeer compact. Gegevens uit een QVD-bestand worden 10-100 keer sneller gelezen dan gegevens uit een andere gegevensbron.

QVD-bestanden kunnen in twee modi worden gelezen: standaard (snel) en geoptimaliseerd (sneller). De geselecteerde modus wordt automatisch bepaald door de Qlik Sense-script-engine. De geoptimaliseerde modus kan alleen worden toegepast als alle geladen velden worden gelezen zonder dat er transformaties (formules die op de velden worden toegepast) nodig zijn. Alleen het hernoemen van velden is toegestaan. Een Where-clausule die ertoe leidt dat de records worden uitgepakt in Qlik Sense zorgt er eveneens voor dat geoptimaliseerd laden wordt uitgeschakeld.

Een QVD-bestand bevat precies één gegevenstabel en bestaat uit de volgende drie delen:

- Een XML-koptekst (in UTF-8-tekenset) met daarin een beschrijving van de velden, de gegevensopmaak en bepaalde andere metagegevens.
- Symbooltabellen in een byte-stuffed indeling.
- Eigenlijke tabelgegevens in een bit-stuffed indeling.

QVD-bestanden kunnen voor vele doeleinden worden gebruikt. Vier daarvan zijn direct te onderkennen. Vaak is er meer dan één van toepassing:

- Sneller gegevens laden  
Door niet of langzaam veranderende blokken gegevensinvoer in QVD-bestanden te bufferen, wordt de scriptuitvoering van grote gegevensverzamelingen aanmerkelijk versneld.
- Lagere belasting van databaseservers  
Ook de hoeveelheid op te halen gegevens uit externe gegevensbronnen kan aanzienlijk worden gereduceerd. Hierdoor is er minder netwerkverkeer nodig en wordt de belasting van externe databases teruggebracht. Als meerdere Qlik Sense-scripts dezelfde gegevens delen, hoeven deze maar één keer uit de brondatabase te worden geladen in een QVD-bestand. De andere toepassingen kunnen via dit QVD-bestand gebruikmaken van dezelfde gegevens.
- Gegevens uit meerdere Qlik Sense-toepassingen consolideren.  
Met de Binary-scriptopdracht kunt u gegevens uit slechts één enkele Qlik Sense-toepassing in een andere laden, maar bij QVD-bestanden kunt u gegevens uit een willekeurige hoeveelheid Qlik Sense-toepassingen door middel van een Qlik Sense-script met elkaar combineren. Hierdoor wordt het mogelijk om binnen één toepassing gelijksoortige gegevens van bijvoorbeeld verschillende bedrijfsafdelingen te combineren.
- Incrementeel laden



In veel voorkomende gevallen kan de QVD-functionaliteit worden gebruikt voor incrementeel laden, d.w.z. dat uitsluitend nieuwe records uit een database worden geladen.



Om te zien hoe de Qlik Community Qlik toepassingsautomatisering benut om QVD-laadtijden te verbeteren, zie [QVD's uitsplitsen met behulp van een automatisering om ladingen te verbeteren](#)

### 6.1 QVD-bestanden maken

Een QVD-bestand kan op twee manieren worden gemaakt:

- Door het expliciet te maken en een naam te geven met de opdracht Store in het Qlik Sense-script.  
U geeft in het script aan dat u een eerder gelezen tabel of deel ervan wilt exporteren naar een expliciet genoemd bestand op een locatie van uw keuze.
- Door dit automatisch te maken en beheren vanuit een script.  
Als u een load- of select-opdracht van het prefix Buffer voorziet, maakt Qlik Sense automatisch een QVD-bestand dat bij het herladen van gegevens onder bepaalde voorwaarden in plaats van de originele gegevensbron kan worden gebruikt.

Tussen de verschillende aangemaakte QVD-bestanden bestaan geen verschillen wat betreft de leessnelheid.

#### Store

Deze scriptopdracht maakt een specifiek genoemd QVD-, CSV-, of txt-bestand.

##### Syntaxis:

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

De opdracht kan alleen velden exporteren uit één gegevenstabel. Als velden uit diverse tabellen moeten worden geëxporteerd, moet eerst een expliciete join in het script worden opgegeven om de te exporteren gegevenstabel te maken.

De tekstwaarden worden in de indeling UTF-8 naar het CSV-bestand geëxporteerd. U kunt een scheidingsteken opgeven, zie **LOAD**. De opslagopdracht voor een CSV -bestand ondersteunt geen BIFF -export.

##### Voorbeelden:

```
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';
```

**Doe het volgende:**

1. Open de app *Zelfstudie voor geavanceerd scriptgebruik*.
2. Klik op de scriptsectie *Product*.
3. Voeg het volgende toe aan het eind van uw script:

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

Uw script zou er als volgt moeten uitzien:

```
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. Klik op **Gegevens laden**.

Het bestand *Product.qvd* zou zich nu in de lijst met bestanden moeten bevinden.

Dit gegevensbestand is het resultaat van het **Crosstable**-script en is een tabel met drie kolommen, één kolom voor elke categorie (Product, Month, Sales). Het gegevensbestand zou nu kunnen worden gebruikt voor het vervangen van de hele *Product*-scriptsectie.

## 6.2 Gegevens uit QVD-bestanden lezen

Een QVD-bestand kan op de volgende manieren door Qlik Sense worden gelezen of geopend:

- Een QVD-bestand laden als een expliciete gegevensbron. Er kan naar QVD-bestanden worden verwezen met een load-opdracht in het Qlik Sense-script, net als naar andere typen tekstbestanden (csv, fix, dif, biff enzovoort).

**Voorbeelden:**

```
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);
```

- Automatisch laden van gebufferde QVD-bestanden. Als u het prefix buffer in de load- of select-opdracht gebruikt, zijn er geen expliciete leesopdrachten nodig. Qlik Sense bepaalt automatisch in welke mate gegevens uit het QVD-bestand of uit de oorspronkelijke LOAD- of SELECT-opdracht worden gebruikt.

- QVD-bestanden via het script benaderen. Een aantal scriptfuncties (alle functies die beginnen met QVD) kan worden gebruikt om informatie op te halen uit de XML-koptekst van een QVD-bestand.

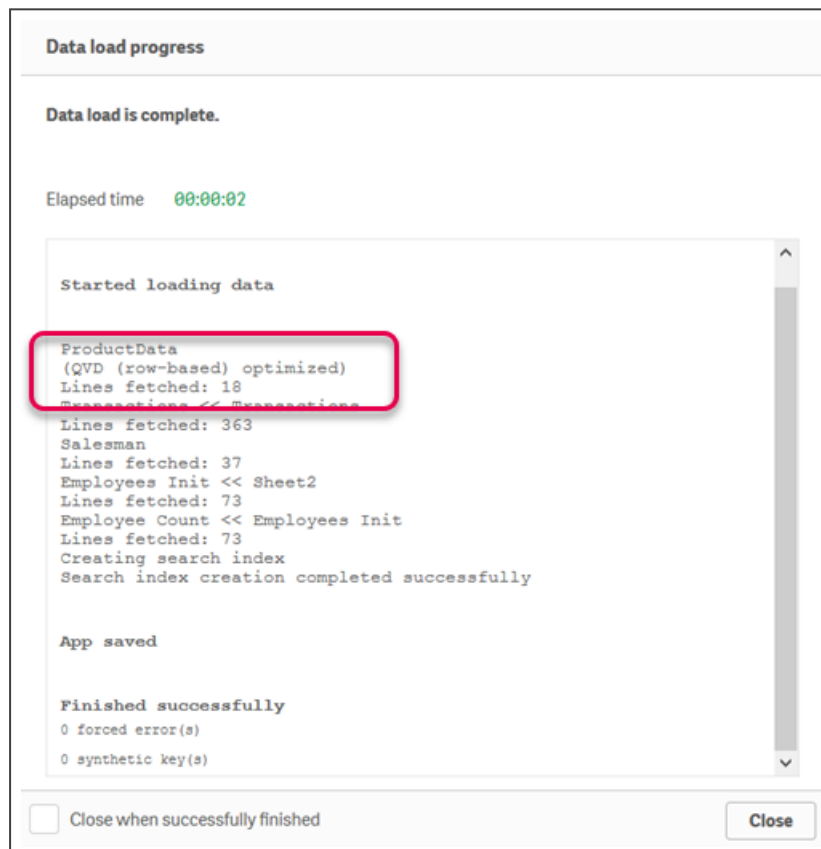
### Doe het volgende:

1. Schakel het hele script uit in de scriptsectie *Product*.
2. Voer het volgende script in:  
`Load * from [lib://AttachedFiles/ProductData.qvd](qvd);`

3. Klik op **Gegevens laden**.

De gegevens wordt geladen uit het QVD-bestand.

*Venster met voortgang laden van gegevens*



Voor meer informatie over het gebruik van QVD-bestanden voor incrementeel laden, bekijkt u deze blogpost in Qlik Community: [Overzicht van Incrementeel laden met Qlik](#)

## Buffer

QVD-bestanden kunnen automatisch worden gemaakt en beheerd met behulp van het prefix Buffer. Dit prefix kan worden gebruikt bij de meeste LOAD- en SELECT-opdrachten in een script. Hiermee wordt aangegeven dat QVD-bestanden worden gebruikt als cache/buffer om het resultaat van de opdracht op te slaan.

### Syntaxis:

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

Als deze opties niet zijn gedefinieerd, wordt de QVD-buffer die bij de eerste uitvoering van het script werd gemaakt, voor onbepaalde tijd gebruikt.

### Voorbeeld:

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

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

Amount is een getal dat de tijdsperiode aangeeft. Het gebruik van Decimals is toegestaan. Als er geen eenheid is aangegeven, wordt verondersteld dat het om dagen gaat.

De optie stale after wordt meestal gebruikt voor databasebronnen waarbij de gegevens niet zijn voorzien van een tijdstempel. Een stale after-clausule geeft simpelweg een tijdsperiode aan vanaf het tijdstip van aanmaken van de QVD-buffer waarna de buffer niet meer als geldig wordt beschouwd. Voor deze tijd wordt de QVD-buffer gebruikt als gegevensbron, daarna wordt de oorspronkelijke gegevensbron gebruikt. Het QVD-bufferbestand wordt dan automatisch bijgewerkt en er begint een nieuwe periode.

### Voorbeeld:

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

### Incremental

Met de optie incremental wordt aangegeven dat alleen een gedeelte van een onderliggend bestand hoeft te worden gelezen. De vorige grootte van het bestand wordt opgeslagen in de XML-header van het QVD-bestand. Dit is vooral nuttig bij logbestanden. Alle records die eerder zijn geladen, worden gelezen uit het QVD-bestand. Nieuwe bestanden worden gelezen uit de originele bron en vervolgens wordt een bijgewerkt QVD-bestand gemaakt.

Let op! De optie incremental kan alleen worden gebruikt met LOAD-opdrachten en tekstbestanden, en incremental load kan niet worden gebruikt als oude gegevens zijn gewijzigd of verwijderd!

### Voorbeeld:

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

Een QVD-buffer wordt normaal gesproken verwijderd als er tijdens de volledige uitvoering van het script in de app waarvoor de buffer werd gemaakt, niet langer naar wordt verwezen of als de app waarvoor de buffer werd gemaakt, niet meer bestaat. De Store-opdracht moet worden gebruikt als u de inhoud van de buffer wilt bewaren als een QVD- of CSV-bestand.

### Doe het volgende:

1. Maak een nieuwe app en geef deze een naam.
2. Voeg een nieuwe scriptsectie toe in de **Editor voor het laden van gegevens**.
3. Onder **AttachedFiles** in het rechtermenu klikt u op **Gegevens selecteren**.
4. Upload en selecteer *Cutlery.xlsx*.
5. In het venster **Selecteer gegevens uit** klikt u op **Script invoeren**.
6. Schakel de velden in de laadinstructie uit en verander de laadinstructie in het volgende:

```
Buffer LOAD *
```

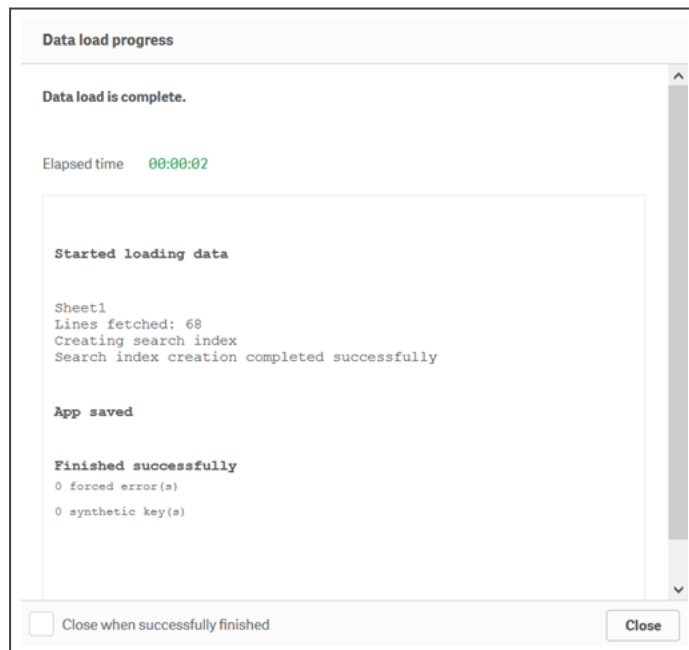
Uw script zou er als volgt moeten uitzien:

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

7. Klik op **Gegevens laden**.

Als u voor de eerste keer gegevens laadt, worden deze geladen vanuit *Cutlery.xlsx*.

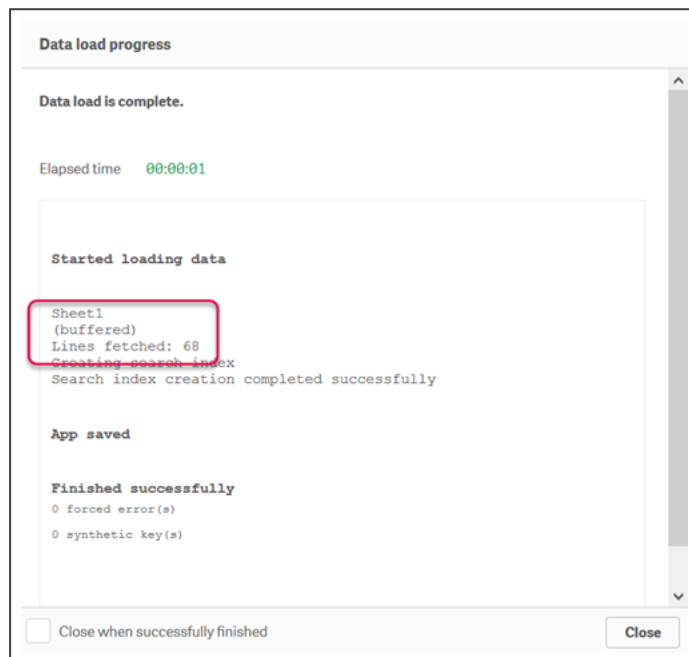
*Venster met voortgang laden van gegevens*



De Buffer-instructie maakt ook een QVD-bestand en slaat dit op in Qlik Sense. In een Qlik Sense Enterprise on Windows-implementatie wordt dit opgeslagen in een directory op de Qlik Sense-server.

8. Klik nogmaals op **Gegevens laden**.
9. De gegevens worden dit keer geladen vanuit het QVD-bestand dat door de Buffer-instructie is gemaakt toen u de gegevens voor de eerste keer hebt geladen.

*Venster met voortgang laden van gegevens*



### 6.3 Hartelijk dank!

U hebt deze zelfstudie nu voltooid en hopelijk beschikt u nu over wat meer kennis van het gebruiken van scripts in Qlik Sense. Bezoek onze website voor meer informatie over de verdere training die beschikbaar is.