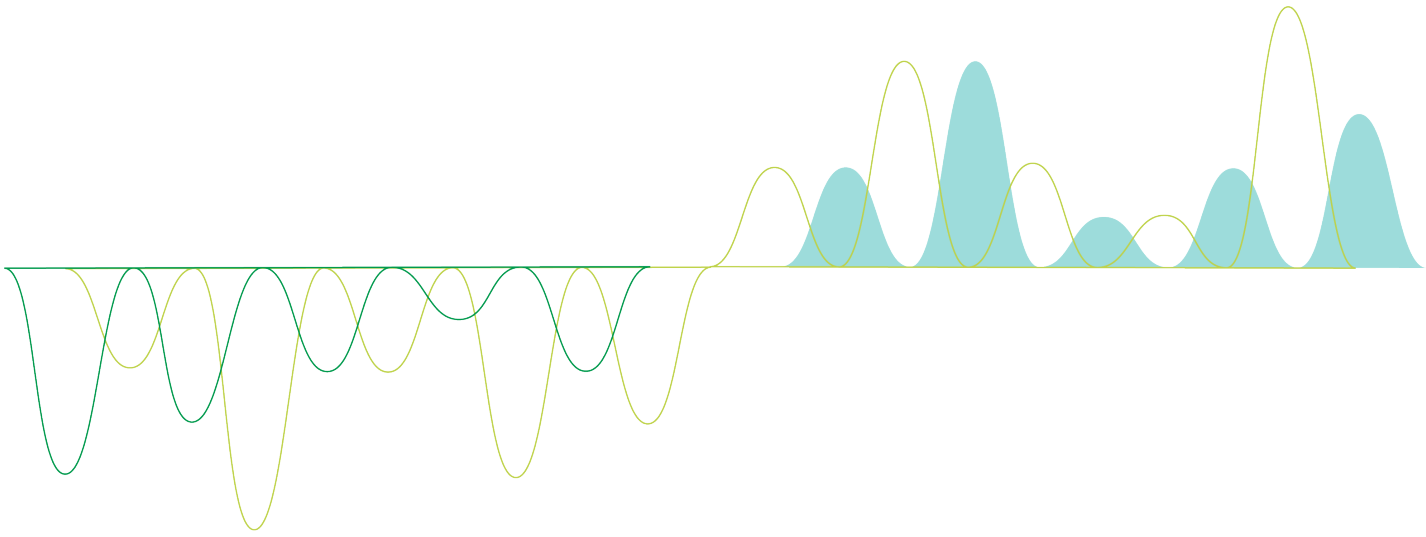
Tutorial - Nächste Schritte bei der Skripterstellung

Qlik Sense®
February 2024



Contents

1 Herzlich willkommen!

Willkommen bei diesem Tutorial, in dem die erweiterte Skripterstellung in Qlik Sense erläutert wird.

Nachdem Sie sich mit den Grundlagen der Skripterstellung vertraut gemacht haben, können Sie mit komplizierteren Datenvorgängen beginnen, während Sie Ihre Daten in Qlik Sense laden. Dazu gehört zum Beispiel das Umwandeln von Daten mithilfe von Kreuztabellen, das Bereinigen von Daten und das Erstellen und Laden von Daten aus Qlik-Datendateien, die als QVD-Dateien bezeichnet werden.

1.1 Lerninhalte

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

1.2 Zielgruppe für diesen Kurs

Sie sollten mit den Grundlagen der Skripterstellung in Qlik Sense vertraut sein. Es wird davon ausgegangen, dass Sie bereits Daten anhand von Skripts geladen und bearbeitet haben.

Wenn dies nicht der Fall ist, wird empfohlen, das Tutorial „Skripterstellung für Anfänger“ durchzuarbeiten.

Sie benötigen Zugriff auf den Dateneditor und müssen berechtigt sein, Daten in Qlik Sense Enterprise on Windows zu laden.

Die Anweisungen gelten allgemein auch für Qlik Sense Cloud Business.

1.3 Paketinhalt

Das heruntergeladene Paket enthält die folgenden Datendateien, die Sie zum Bearbeiten des Tutorials benötigen:

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

Das Paket enthält auch eine Kopie der App *Erweitertes Skript – Tutorial*. Zusätzliche Skriptabschnitte in der App enthalten die Skripts für die anderen Apps, die Sie in diesem Tutorial erstellen. Sie können die App in Ihren Hub hochladen.

Sie sollten die App wie im Tutorial beschrieben selbst erstellen, um den Lernerfolg zu maximieren. Zudem müssen Sie die Datendateien wie im Tutorial beschrieben hochladen und verbinden, damit die Datenladevorgänge funktionieren.

Wenn Sie Probleme haben, kann Sie die App aber bei der Problembehebung unterstützen. Wir haben angegeben, welche Skriptsegmente zu den einzelnen Lektionen gehören.

1.4 Lektionen in diesem Tutorial

Je nach Ihrer Erfahrung mit Qlik Sense dauert die Bearbeitung dieses Tutorials 3-4 Stunden. Die Themen sollten ihrer Reihenfolge nach bearbeitet werden. Sie können die Arbeit daran jedoch jederzeit unterbrechen und später fortsetzen. Glücklicherweise umfasst das Tutorial keine Tests.

Umwandeln von Daten

Verwenden des Crosstable -Zusatzes

Kombinieren von Tabellen mit Join und Keep

Verwenden von Inter-Record-Funktionen: Peek, Previous und Exists

Einordnung in Intervalle und iterative Ladevorgänge

Datenpflege durch Mapping

Handhabung hierarchischer Daten

QVD-Dateien

1.5 Weitere Informationsquellen und Ressourcen

- Qlik bietet eine Vielzahl von Ressourcen, wenn Sie noch mehr erfahren möchten.
- Qlik Online-Hilfe ist verfügbar.
- Schulungen, einschließlich kostenloser Online-Kurse, stehen im Qlik Continuous Classroom zur Verfügung.
- Diskussionsforen, Blogs und mehr finden Sie in der Qlik Community.

2 LOAD- und SELECT-Anweisungen

Sie können mithilfe von LOAD- und SELECT-Befehlen Daten in Qlik Sense laden. Durch jeden dieser Anweisungen wird eine interne Tabelle erstellt. LOAD wird zum Laden von Daten aus Dateien verwendet, während mit SELECT Daten aus Datenbanken geladen werden.

In diesem Tutorial verwenden Sie Daten aus Dateien. Daher werden LOAD-Anweisungen verwendet.

Sie können auch ein vorangestellte LOAD verwenden, um den Inhalt der geladenen Daten beeinflussen zu können. Beispielsweise muss das Umbenennen von Feldern in einer LOAD-Anweisung erfolgen, während die SELECT-Anweisung keine Änderungen an Feldnamen zulässt.

Beim Laden von Daten in Qlik Sense gelten die folgenden Regeln:

- Qlik Sense unterscheidet nicht zwischen Tabellen, die durch eine LOAD- oder eine SELECT-Anweisung generiert werden. Wenn mehrere Tabellen geladen werden, ist es daher unerheblich, ob die Tabellen durch LOAD- oder SELECT-Anweisungen oder eine Kombination beider Anweisungen geladen wurden.
- Die Reihenfolge der Felder in der zugrunde liegenden Datenbank ist für Qlik Sense nicht relevant.
- Feldnamen unterscheiden zwischen Groß- und Kleinschreibung und werden verwendet, um Zuordnungen zwischen Datentabellen einzurichten. Aus diesem Grund müssen Felder im Ladeskript manchmal umbenannt werden, um das gewünschte Datenmodell zu erhalten.

3 Umwandeln von Daten

Sie können Daten im Dateneditor umwandeln und bearbeiten, bevor Sie die Daten in Ihrer App verwenden.

Einer der Vorteile der Datenbearbeitung ist, dass Sie beschließen können, nur eine Teilmenge der Daten aus einer Datei zu laden, z. B. einige wenige ausgewählte Spalten einer Tabelle. So können Sie die Datenverarbeitung effizienter gestalten. Sie können die Daten auch mehrmals laden, um die Rohdaten in mehrere neue interne Tabellen aufzuspalten. Außerdem können Sie auch Daten aus mehreren Quellen laden und sie zu einer Tabelle in Qlik Sense zusammenführen.

In den folgenden Übungen wird gezeigt, wie Sie Daten anhand des Crosstable-Zusatzes laden. Zudem erfahren Sie, wie Sie Tabellen verbinden, Inter-Record-Funktionen wie Peek und Previous, verwenden und über While Load eine Reihe mehrmals einlesen können.

3.1 Verwenden des Crosstable-Zusatzes

Kreuztabellen sind ein häufig verwendeter Tabellentyp, bei dem eine Matrix von Werten zwischen zwei senkrecht aufeinander stehenden Wertelisten steht. Bei Kreuztabellen mit Daten kann der Zusatz Crosstable verwendet werden, um die Daten umzuformen und die gewünschten Felder zu erstellen.

Crosstable-Zusatz

In der folgenden *Product*-Tabelle stehen Ihnen eine Spalte pro Monat und eine Zeile pro Produkt zur Verfügung.

Produkttabelle

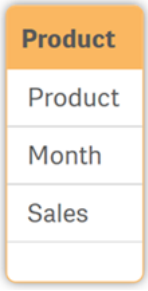
Produkt	Jan 2014	Feb 2014	Mar 2014	Apr 2014	May 2014	Jun 2014
A	100	98	100	83	103	82
B	284	279	297	305	294	292
C	50	53	50	54	49	51

Wenn Sie die Tabelle laden, ergibt sich eine Tabelle mit einem Feld für *Product* und einem Feld für jeden der Monate.

Tabelle Product mit Feld Product und je einem Feld für die Monate

Wenn Sie diese Daten analysieren möchten, ist es viel einfacher, alle Zahlen in einem Feld und alle Monate in einem anderen Feld aufzulisten. Es ergäbe sich also eine Tabelle mit drei Spalten – jeweils eine pro Kategorie (*Product, Month, Sales*).

Tabelle Product mit den Feldern Product, Month und Sales



Der Crosstable-Zusatz wandelt die Daten in eine Tabelle mit einer Spalte für *Month* und einer weiteren für *Sales* um. Kurz gesagt: Feldnamen werden in Feldwerte umgewandelt.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie eine neue App und nennen Sie sie *Erweiterte Skripterstellung – Tutorial*.
2. Fügen Sie einen neuen Skriptabschnitt im **Dateneditor** hinzu.
3. Nennen Sie den Abschnitt *Product*.
4. Klicken Sie im rechten Menü unter **AttachedFiles** auf **Daten auswählen**.
5. Laden Sie *Product.xlsx* hoch und wählen Sie die Datei aus.
6. Wählen Sie die Tabelle *Product* im Fenster **Daten auswählen aus**.

   *Achten Sie unter **Feldnamen** darauf, dass der Eintrag **Eingebettete Feldnamen** ausgewählt ist, damit die Namen der Tabellenfelder beim Laden der Daten eingeschlossen sind.*

7. Klicken Sie auf **Skript einfügen**.

   Ihr Skript sollte folgendermaßen aussehen:

```
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);
```

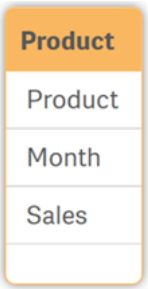
8. Klicken Sie auf **Daten laden**.
9. Öffnen Sie die **Datenmodellansicht**. Das Datenmodell sieht folgendermaßen aus:

   Tabelle Product mit Feld Product und je einem Feld für die Monate

10. Klicken Sie im **Dateneditor** auf die Registerkarte *Product*.
11. Geben Sie Folgendes oberhalb der LOAD-Anweisung ein:
    `CrossTable(Month, Sales)`
12. Klicken Sie auf **Daten laden**.
13. Öffnen Sie die **Datenmodellansicht**. Das Datenmodell sieht folgendermaßen aus:

    Tabelle Product mit den Feldern Product, Month und Sales

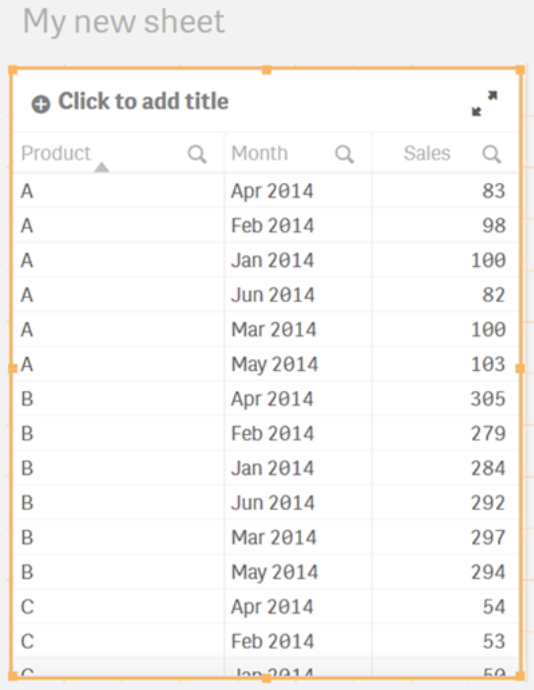
    

    Beachten Sie, dass die Datenquelle in der Regel nur eine Spalte als Qualifizierung umfasst: als interner Schlüssel (im obigen Beispiel *Product*). Es sind jedoch auch mehrere möglich. Dazu müssen alle Qualifizierungsfelder im LOAD-Befehl vor den Attributfeldern aufgeführt werden, und der Crosstable-

Zusatz kann verwendet werden, um die Anzahl der Qualifizierungsfelder zu definieren. Es ist nicht möglich, ein preceding LOAD oder einen Zusatz vor dem Schlüsselwort Crosstable zu erstellen. Sie können aber automatische Zusammenfassung verwenden.

In einer Tabelle in Qlik Sense sehen Ihre Daten wie folgt aus:

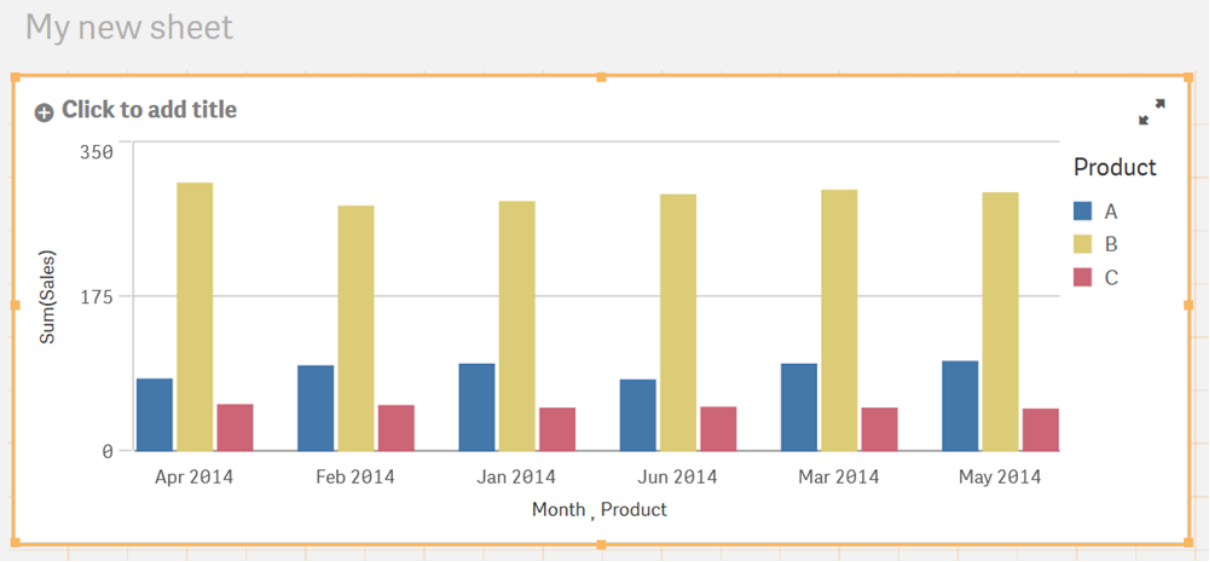
Tabelle mit den Daten, die mit dem Crosstable-Zusatz geladen wurden



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

Sie können jetzt beispielsweise ein Balkendiagramm mit den Daten erstellen:

Balkendiagramm mit den Daten, die mit dem Crosstable-Zusatz geladen wurden



Weitere Informationen zur Verwendung von Crosstable finden Sie in folgendem Blog-Eintrag in Qlik Community: The Crosstable Load *(Der Crosstable-Ladevorgang). Die Verhaltensweisen werden im Kontext von QlikView besprochen. Die Logik gilt aber ebenso für Qlik Sense.*

Numerische Interpretation ist für Attributfelder nicht möglich. Das bedeutet, dass diese nicht automatisch interpretiert werden, wenn Monate als Spaltenüberschrift verwendet werden. Eine Übergangslösung besteht darin, mit dem Crosstable-Zusatz eine temporäre Tabelle zu erstellen und einen zweiten Lauf durchzuführen, damit die Interpretationen dem folgenden Beispiel entsprechen.

Beachten Sie, dass es sich hierbei nur um ein Beispiel handelt. Es sind keine in Qlik Sense zu bearbeitenden Übungen vorhanden.

```
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;
```

Löschen des Cache-Speichers

Sie können erstellte Tabellen löschen, um den Zwischenspeicher zu löschen. Wenn Sie wie im vorherigen Abschnitt eine temporäre Tabelle laden, sollten Sie diese löschen, wenn sie nicht mehr benötigt wird. Hier ein Beispiel:

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

Sie können auch Felder löschen. Hier ein Beispiel:

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

Wie Sie sehen, können die Schlüsselwörter TABLE und FIELD Singular oder Plural sein.

3.2 Kombinieren von Tabellen mit Join und Keep

Bei der Verknüpfung werden zwei Tabellen zu einer Tabelle kombiniert. Dabei bilden jeweils zwei Datensätze aus den Ursprungstabellen, meist zwei Datensätze, die einen Wert in einem Feld gemeinsam haben, einen neuen Datensatz in der entstehenden Tabelle. In Qlik Sense können Verknüpfungen im Skript definiert werden. Dabei werden programminterne Tabellen erzeugt.

Durch den join-Zusatz im Skript ist es möglich, Tabellen beim Einlesen durch Joins zusammenzuschließen. In diesem Fall speichert Qlik Sense nur eine einzelne interne Tabelle als Ergebnis des Joins. Dies ist in einigen Situationen erforderlich, jedoch bestehen auch Nachteile.

- Die internen Tabellen werden fast immer größer und Qlik Sense arbeitet entsprechend langsamer.
- Einige Informationen gehen möglicherweise verloren, z. B. die Häufigkeit (Anzahl der Datensätze) in der zugrunde liegenden Tabelle.

Der Keep-Zusatz bewirkt, dass vor dem Speichern in Qlik Sense die beiden Tabellen verglichen werden und dass aufgrund dieses Vergleichs in einer oder in beiden Tabellen bestimmte Datensätze wegfallen. Dadurch kann in den meisten Fällen auf einen Join verzichtet werden.

In diesem Handbuch wird der Begriff „Join“ nur für Joins gebraucht, die vor dem Speichern der Tabellen entstehen. Die Verknüpfung von Tabellen, nachdem diese angelegt wurden, ist im Grunde genommen aber auch nichts anderes als ein Join.

Join

Am einfachsten erstellen Sie Joins durch einen Join-Zusatz im Skript, mit dem die interne Tabelle mit einer anderen benannten Tabelle oder mit der zuletzt erstellten Tabelle zusammengefügt wird. Dieser kombiniert die Tabelle mit einer bereits erstellten Tabelle durch einen Outer Join, d. h. in der entstehenden Tabelle sind alle möglichen Kombinationen von Datensätzen enthalten.

Beispiel:

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

Die entstehende Tabelle enthält die Felder a, b, c und d. Die Anzahl der Datensätze hängt von den Feldwerten der beiden Tabellen ab.

Die Namen der Felder, über die der Join gemacht wird, müssen exakt übereinstimmen. Der Join kann über beliebig viele Felder gemacht werden. Meistens haben die Tabellen nur ein oder wenige Felder gemeinsam. Haben die Tabellen gar kein Feld gemeinsam, so ergibt der Join das karthesische Produkt der Tabellen. Es ist möglich, dass die Tabellen in allen Feldnamen übereinstimmen, dies ergibt jedoch im Allgemeinen keinen Sinn. Sofern im Join-Befehl mit Join-Zusatz kein anderer Tabellenname einer zuvor geladenen Tabelle angegeben ist, bezieht er sich jeweils auf die direkt zuvor angelegte Tabelle. Die Reihenfolge der Befehle ist daher nicht beliebig.

Verwenden von Join

Der explizite Zusatz Join im Qlik Sense-Skript bewirkt eine vollständige Zusammenfügung der beiden Tabellen, d. h. die beiden Tabellen werden zu einer zusammengeschlossen. Solche Verknüpfungen können oft zu sehr großen Tabellen führen.

Gehen Sie folgendermaßen vor:

1. Öffnen Sie die App *Erweitertes Skript-Tutorial*.
2. Fügen Sie einen neuen Skriptabschnitt im **Dateneditor** hinzu.
3. Rufen Sie den Abschnitt *Transactions* auf.
4. Klicken Sie im rechten Menü unter **AttachedFiles** auf **Daten auswählen**.
5. Laden Sie *Transactions.csv* hoch und wählen Sie die Datei aus.

   *Achten Sie unter **Feldnamen** darauf, dass der Eintrag **Eingebettete Feldnamen** ausgewählt ist, damit die Namen der Tabellenfelder beim Laden der Daten eingeschlossen sind.*

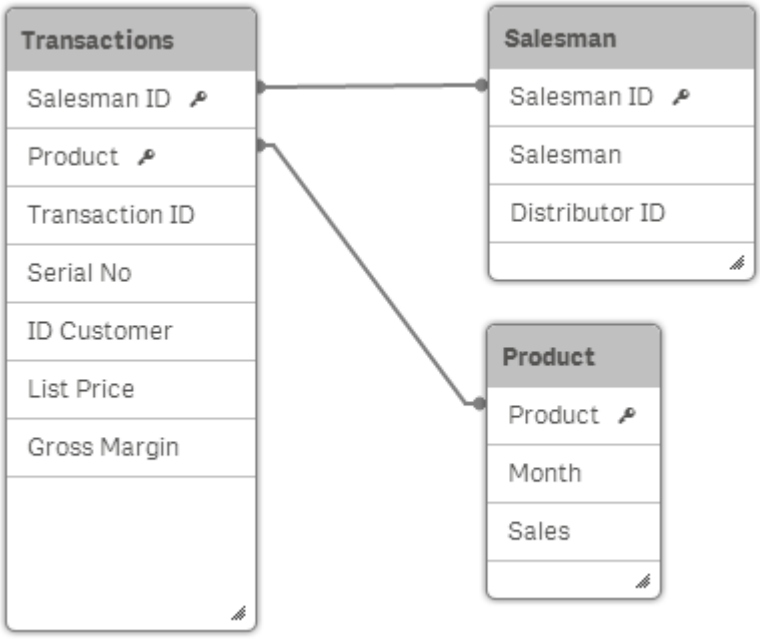
6. Klicken Sie im Fenster **Daten auswählen aus** auf **Skript einfügen**.
7. Laden Sie *Salesman.xlsx* hoch und wählen Sie die Datei aus.
8. Klicken Sie im Fenster **Daten auswählen aus** auf **Skript einfügen**.

   Ihr Skript sollte folgendermaßen aussehen:

```
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. Klicken Sie auf **Daten laden**.
10. Öffnen Sie die **Datenmodellansicht**. Das Datenmodell sieht folgendermaßen aus:

Datenmodell: Tabellen Transactions, Salesman und Product



Jedoch entsprechen getrennte *Transactions*- und *Salesman*-Tabellen möglicherweise nicht dem geforderten Ergebnis. Dann ist es besser, die beiden Tabellen zusammenzufügen.

Gehen Sie folgendermaßen vor:

1. Um einen Namen für die verknüpfte Tabelle festzulegen, fügen Sie die folgende Zeile oberhalb der ersten LOAD-Anweisung hinzu:
   ```
   Transactions:
   ```
2. Um die Tabellen *Transactions* und *Salesman* zu verknüpfen, fügen Sie die folgende Zeile über der zweiten LOAD-Anweisung hinzu:
   ```
   Join(Transactions)
   ```

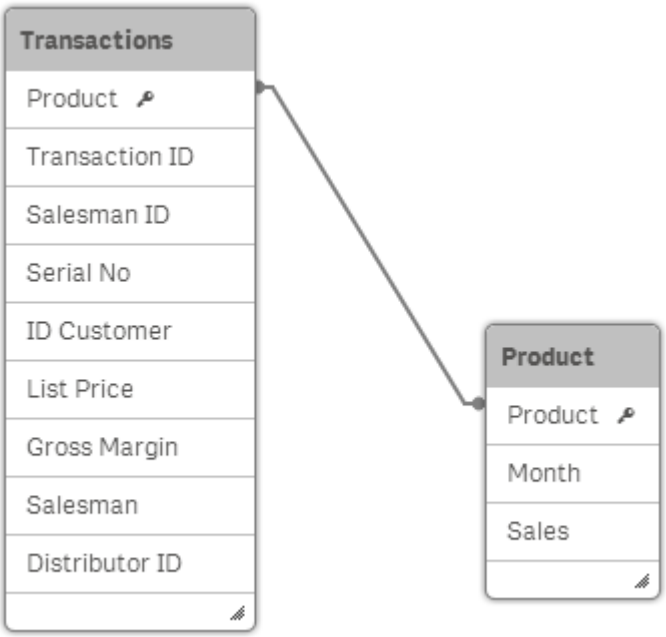
   Ihr Skript sollte folgendermaßen aussehen:

   ```
   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. Klicken Sie auf **Daten laden**.
4. Öffnen Sie die **Datenmodellansicht**. Das Datenmodell sieht folgendermaßen aus:

   Datenmodell: Tabellen Transactions und Product



   Alle Felder der Tabellen *Transactions* und *Salesman* werden nun in einer einzigen *Transactions*-Tabelle kombiniert.

> *Weitere Informationen zur Verwendung von Join finden Sie in den folgenden Blog-Einträgen in Qlik Community:* To Join or not to Join *(Verknüpfen oder nicht verknüpfen),* Mapping as an Alternative to Joining*(Mapping als Alternative für die Verknüpfung). Die Verhaltensweisen werden im Kontext von QlikView besprochen. Die Logik gilt aber ebenso für Qlik Sense.*

Keep

Einer der großen Vorteile von Qlik Sense besteht aber gerade darin, dass automatisch Verknüpfungen zwischen Tabellen hergestellt werden, ohne dass ein Join gemacht wird. Dadurch wird Speicherplatz gespart, die Zugriffszeiten verkürzen sich, und das System behält eine hohe Flexibilität. Durch die Keep-Funktion reduziert sich die Zahl der Fälle, in denen explizite Joins benötigt werden, nochmals.

Der Zusatz Keep zwischen zwei LOAD- oder SELECT-Anweisungen bewirkt, dass vor dem Speichern in Qlik Sense eine oder beide Tabellen auf die Schnittmenge der Tabellendaten reduziert werden. Dem Keep-Zusatz muss stets einer der Zusätze Inner, Left oder Right vorangehen. Die Auswahl der Datensätze in den Tabellen erfolgt nach denselben Regeln wie bei dem entsprechenden Join. Die Tabellen werden jedoch nicht zusammengeschlossen, sondern als zwei Tabellen unter verschiedenen Namen in Qlik Sense gespeichert.

Inner

Vor den Zusätzen Join und Keep kann im Datenladeskript der Zusatz Inner gestellt werden.

Vor Join bewirkt er, dass ein Inner Join hergestellt wird. Die dadurch entstehende Tabelle enthält nur die Datensätze, zu denen in der jeweils anderen Tabelle ein passender Datensatz vorhanden ist.

Vor einem Keep-Zusatz bewirkt „Inner“, dass von beiden Tabellen nur die Datensätze in Qlik Sense gespeichert werden, für die in der jeweils anderen Tabelle ein passender Datensatz vorhanden ist.

Beispiel:

In diesen Beispielen werden die Quelltabellen *Table1* und *Table2* verwendet.

Beachten Sie, dass es sich hierbei nur um Beispiele handelt. Es sind keine in Qlik Sense zu bearbeitenden Übungen vorhanden.

Table 1

A	**B**
1	aa
2	cc
3	ee

Table2

A	**C**
1	xx
4	yy

Inner Join

Zunächst wird für die Tabellen ein Inner Join durchgeführt. Daraus ergibt sich *VTable* mit lediglich einer einzigen Zeile, der einzige Datensatz, der in beiden Tabellen vorhanden ist, wobei die Daten von beiden Tabellen kombiniert werden.

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

VTable

A	B	C
1	aa	xx

Inner Keep

Wird stattdessen ein Inner Keep durchgeführt, bleiben weiterhin zwei Tabellen bestehen. Die beiden Tabellen sind über das gemeinsame Feld *A* miteinander verknüpft.

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

VTab1

A	B
1	aa

VTab2

A	C
1	xx

Left

Vor Join und Keep kann im Datenladeskript der Zusatz left gestellt werden.

Vor einem Join-Zusatz bewirkt "left", dass ein Left Join hergestellt wird. Die dadurch entstehende Tabelle enthält die Datensätze der ersten Tabelle ergänzt mit den passenden Sätzen der zweiten Tabelle, sofern vorhanden.

Vor einem Keep-Zusatz bewirkt „Left“, dass von der zweiten Tabelle nur die Datensätze in Qlik Sense gespeichert werden, für die ein passender Datensatz in der ersten Tabelle existiert.

Beispiel:

In diesen Beispielen werden die Quelltabellen *Table1* und *Table2* verwendet.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Zunächst wird für die Tabellen ein Left Join durchgeführt, wodurch eine *VTable* mit allen Zeilen aus *Table1* entsteht, die mit den passenden Feldern in *Table2* kombiniert wird.

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

VTable

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

Wird stattdessen ein Left Keep durchgeführt, bleiben weiterhin zwei Tabellen bestehen. Die beiden Tabellen sind über das gemeinsame Feld *A* miteinander verknüpft.

```
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

Vor Join und Keep kann in der Qlik Sense-Skriptsprache der Zusatz right platziert werden.

Vor einem Join-Zusatz bewirkt „Right“, dass ein Right Join hergestellt wird. Die dadurch entstehende Tabelle enthält die Datensätze der zweiten Tabelle, ergänzt durch die passenden Sätze der ersten Tabelle, sofern vorhanden.

Vor einem Keep-Zusatz bewirkt „Right“, dass von der ersten Tabelle nur die Datensätze in Qlik Sense gespeichert werden, für die ein passender Datensatz in der zweiten Tabelle existiert.

Beispiel:

In diesen Beispielen werden die Quelltabellen *Table1* und *Table2* verwendet.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Zunächst wird für die Tabellen ein Right Join durchgeführt, wodurch eine *VTable* mit allen Zeilen aus *Table2* entsteht, die mit den passenden Feldern in *Table1* kombiniert wird.

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

VTable

A	B	C
1	aa	xx
4	-	yy

Wird stattdessen ein Right Keep durchgeführt, bleiben weiterhin zwei Tabellen bestehen. Die beiden Tabellen sind über das gemeinsame Feld *A* miteinander verknüpft.

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

VTab1

A	B
1	aa

VTab2

A	C
1	xx
4	yy

3.3 Verwenden von Inter-Record-Funktionen: Peek, Previous und Exists

Diese Funktionen werden benötigt, wenn zur Evaluation des aktuellen Datensatzes Werte aus vorangehenden Datensätzen herangezogen werden sollen.

In diesem Teil des Tutorials werden die Funktionen Peek(), Previous() und Exists() besprochen.

Peek()

Peek() gibt den Wert eines Feldes in einer Tabelle für eine Zeile zurück, die bereits geladen wurde. Die Zeilennummer kann wie die Tabelle festgelegt werden. Wenn keine Zeilennummer angegeben ist, wird der letzte zuvor geladene Datensatz verwendet.

Syntax:

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

Zeile muss eine ganze Zahl sein. 0 steht für den ersten Datensatz, 1 für den zweiten usw. Mithilfe von negativen Zahlen können die Datensätze vom unteren Ende der Tabelle aus gezählt werden. -1 bezeichnet den letzten gelesenen Datensatz.

Ist keine Zeile angegeben, wird -1 angenommen.

Tablename ist eine Tabellenbezeichnung ohne abschließenden Doppelpunkt. Fehlt *tablename*, wird die aktuelle Tabelle verwendet. Wird die Funktion außerhalb der **LOAD**-Anweisung verwendet oder bezieht sie sich auf eine andere Tabelle, muss *tablename* explizit angegeben werden.

Previous()

Previous() liefert den Wert der **expr**-Formel, wobei für die Berechnung Daten aus dem letzten Datensatz verwendet werden, der nicht durch einen **where**-Zusatz ausgeschlossen wurde. Im ersten Datensatz einer internen Tabelle liefert diese Funktion NULL.

Syntax:

```
Previous(expression)
```

Die Previous()-Funktion kann verschachtelt sein, um auf weiter zurückliegende Datensätze zuzugreifen. Die Daten werden direkt aus der Eingabequelle abgerufen, wodurch auch ein Bezug auf Felder möglich ist, die nicht in Qlik Sense geladen wurden, d. h. selbst wenn sie nicht in der assoziativen Datenbank gespeichert wurden.

Exists()

Exists() bestimmt, ob ein spezifischer Feldwert bereits in das Feld im Datenladeskript geladen wurde. Die Funktion gibt TRUE oder FALSE zurück, und kann deshalb in der **where**-Bedingung eines **LOAD**-Befehls oder eines **IF**-Befehls verwendet werden.

Syntax:

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

Das Feld muss in den bisher vom Skript geladenen Daten vorhanden sein. *Expression* ist eine Formel, die den zu suchenden Wert im angegebenen Feld ergibt. Wird sie weggelassen, wird der aktuelle Wert des Datensatzes angenommen.

Verwenden von Peek() und Previous()

In ihrer einfachsten Form werden Peek() und Previous() verwendet, um spezielle Werte in einer Tabelle zu ermitteln. Es folgt ein Beispiel der Daten in der Tabelle *Employees*, die Sie in dieser Übung laden.

Beispiel der Daten aus der Tabelle „Employees“

Datum	Eingestellt	Entlassen
1/1/2011	6	0
2/1/2011	4	2
3/1/2011	6	1
4/1/2011	5	2

Derzeit werden hier nur Daten für Monat, Einstellungen und Kündigungen erfasst. Daher werden wir Felder für *Employee Count* und *Employee Var* mit den Funktionen Peek() und Previous() hinzufügen, um die monatliche Differenz bei der gesamten Belegschaft zu ermitteln.

Gehen Sie folgendermaßen vor:

1. Öffnen Sie die App *Erweitertes Skript-Tutorial*.
2. Fügen Sie einen neuen Skriptabschnitt im **Dateneditor** hinzu.
3. Rufen Sie den Abschnitt *Employees* auf.
4. Klicken Sie im rechten Menü unter **AttachedFiles** auf **Daten auswählen**.
5. Laden Sie *Employees.xlsx* hoch und wählen Sie die Datei aus.

   Achten Sie unter Field names darauf, dass der Eintrag Embedded field names ausgewählt ist, damit die Namen der Tabellenfelder beim Laden der Daten eingeschlossen sind.

6. Klicken Sie im Fenster **Daten auswählen aus** auf **Skript einfügen**.

Ihr Skript sollte folgendermaßen aussehen:

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

7. Bearbeiten Sie das Skript, sodass es folgendermaßen aussieht:

```
[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);
```

   Die Datumsangaben im *Date*-Feld des Excel-Arbeitsblatts werden im Format MM/TT/JJJJ angegeben. Um sicherzustellen, dass das Datum unter Verwendung des Formats aus den Systemvariablen ordnungsgemäß interpretiert wird, wird die Funktion Date auf das Feld *Date* angewendet.

   Mit der Funktion Peek() können Sie neue Werte, die in ein definiertes Feld geladen wurden, ermitteln. In der Formel wird zunächst geprüft, ob rowno() 1 entspricht. Wenn er 1 entspricht, gibt es keinen Wert für *Employee Count*. Daher werden wir die Differenz von *Hired* minus *Terminated* in das Feld einpflegen.

   Wenn der Wert für rowno() größer als 1 ist, prüfen wir den Wert für *Employee Count* im letzten Monat und nutzen diesen, um die Differenz zwischen dem Mitarbeiterwert für *Hired* in diesem Monat minus *Terminated* hinzuzufügen.

   Beachten Sie, dass wir in der Funktion Peek() den Wert (-1) verwenden. Dadurch wird Qlik Sense aufgefordert, den Wert über dem aktuellen Wert zu prüfen. Falls nicht (-1) angegeben ist, geht Qlik Sense davon aus, dass Sie den vorherigen Wert prüfen möchten.

8. Fügen Sie Folgendes zum Ende des Skripts hinzu:

```
[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];
```

   Mit der Funktion Previous() können Sie den neuesten Wert, der in ein definiertes Feld geladen wurde, ermitteln. In dieser Formel wird zunächst geprüft, ob rowno() 1 entspricht. Falls der Wert gleich 1 ist, wissen wir, dass es keinen Wert für Employee Var gibt, da es keine Aufzeichnungen für den Vormonatswert *Employee Count* gibt. Wir geben also einfach 0 als Wert ein.

Ist der Wert für rowno() größer als 1, wissen wir, dass es einen Wert für *Employee Var* gibt. Wir prüfen daher den Vormonatswert für *Employee Count* und ziehen diese Zahl vom Vormonatswert für *Employee Count* ab, um den Wert im Feld *Employee Var* zu erstellen.

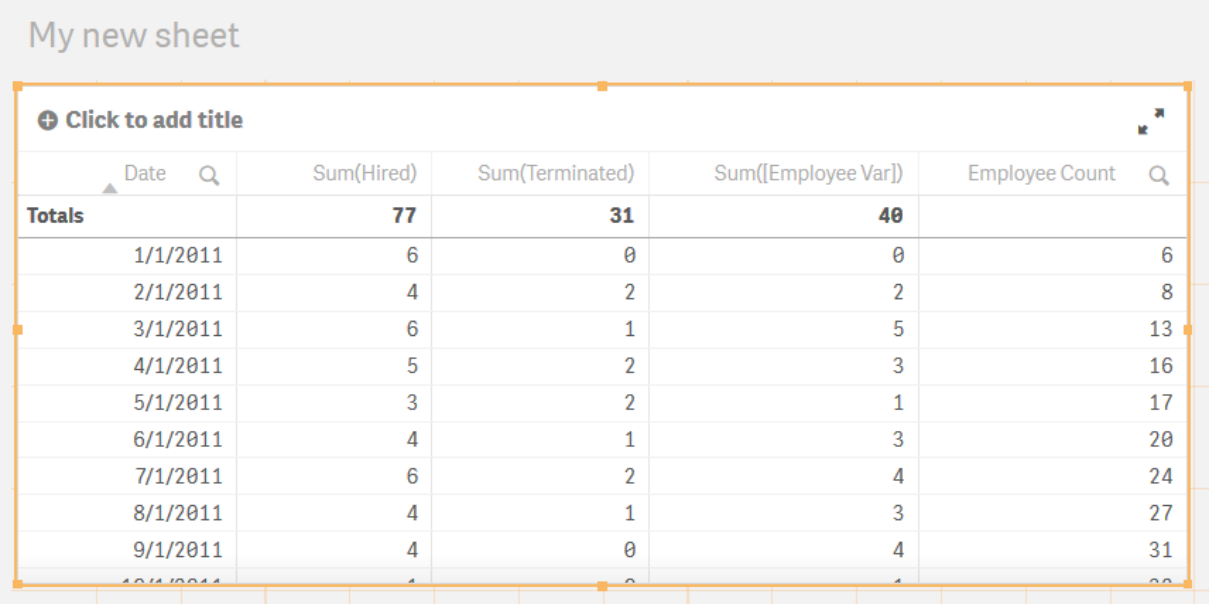
Ihr Skript sollte folgendermaßen aussehen:

```
[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. Klicken Sie auf **Daten laden**.

   Erstellen Sie in einem neuen Arbeitsblatt in der App-Übersicht eine Tabelle mit *Date, Hired, Terminated, Employee Count* und *Employee Var* als Spalten der Tabelle. Die sich ergebende Tabelle sollte folgendermaßen aussehen:

Tabelle nach Verwendung von Peek und Previous im 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

Mit Peek() und Previous() können Sie definierte Zeilen innerhalb einer Tabelle festlegen. Der größte Unterschied zwischen den beiden Funktionen liegt darin, dass mit der Funktion Peek() der Benutzer ein Feld prüfen kann, das nicht zuvor in das Skript geladen wurde. Über die Funktion Previous() kann nur ein zuvor geladenes Feld geprüft werden. Previous() reagiert auf den Input der LOAD-Anweisung und Peek() reagiert auf den Output der LOAD-Anweisung. (Genau wie der Unterschied zwischen RecNo() und RowNo().) Das bedeutet, dass die beiden Funktionen unterschiedlich arbeiten, wenn Sie eine Where-Klausel verwenden.

Daher ist die Funktion Previous() besser geeignet, wenn Sie den aktuellen Wert mit dem vorherigen Wert vergleichen möchten. In diesem Beispiel habe wir die Mitarbeiterfluktuation von Monat zu Monat berechnet.

Die Funktion Peek() eignet sich besser, wenn Sie ein Feld bestimmen, das noch nicht in die Tabelle geladen wurde, oder wenn Sie eine spezielle Zeile bestimmen möchten. Das wurde in unserem Beispiel gezeigt. Hier berechneten wir den Wert für *Employee Count*, indem zum Vormonatswert für *Employee Count* die Differenz zwischen den angestellten und gekündigten Mitarbeitern im aktuellen Monat hinzugerechnet wurde. Wir erinnern uns: *Employee Count* war zuvor kein Feld in der Originaldatei.

Weitere Informationen zur Verwendung von Peek() und Previous() finden Sie in folgendem Blog-Eintrag in Qlik Community: Peek() vs Previous() – When to Use Each. Die Verhaltensweisen werden im Kontext von QlikView besprochen. Die Logik gilt aber ebenso für Qlik Sense.

Verwenden von Exists()

Die Funktion Exists() wird häufig mit der Where-Bedingung im Skript verwendet, um Daten zu laden, wenn ähnliche Daten bereits in das Datenmodell geladen wurden.

Im folgenden Beispiel verwenden wir auch die Funktion Dual(), um Strings Zahlenwerte zuzuweisen.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie eine neue App und geben Sie ihm einen Namen.
2. Fügen Sie einen neuen Skriptabschnitt im **Dateneditor** hinzu.
3. Rufen Sie den Abschnitt *People* auf.
4. Geben Sie folgendes Skript ein:

```
//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;
```

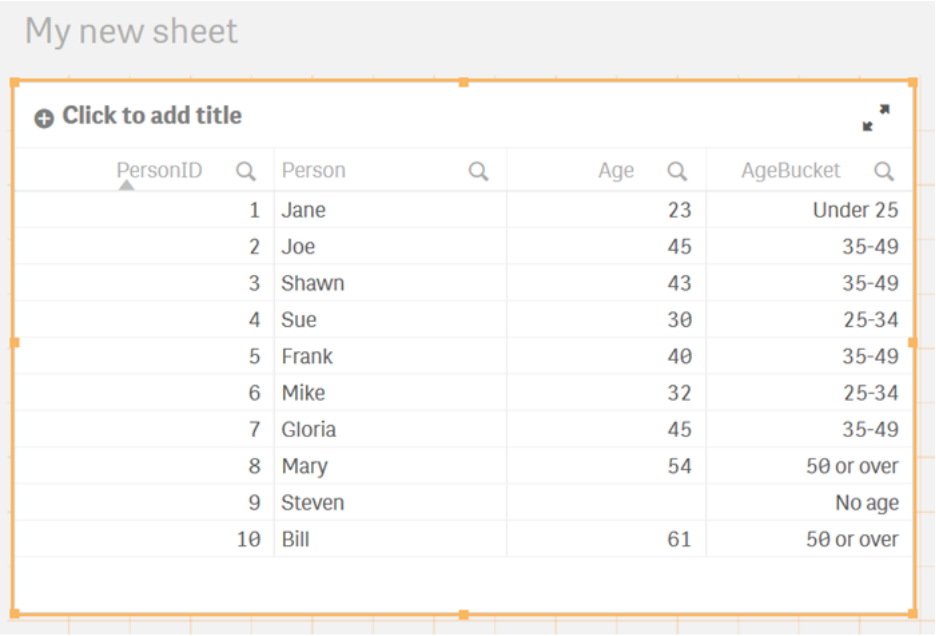
5. Klicken Sie auf **Daten laden**.

   Im Skript werden die Felder *Age* und *AgeBucket* nur eingelesen, wenn der Wert für *PersonID* bereits im Datenmodell eingelesen wurde.

   In der Tabelle *AgeTemp* ist das Alter für *PersonID* 11 und 12 aufgeführt. Da diese IDs jedoch nicht in das Datenmodell (in der Tabelle *People*) geladen wurden, sind sie von der Bedingung Where Exists (PersonID) ausgeschlossen. Diese Bedingung kann auch wie folgt geschrieben werden: Where Exists (PersonID, PersonID).

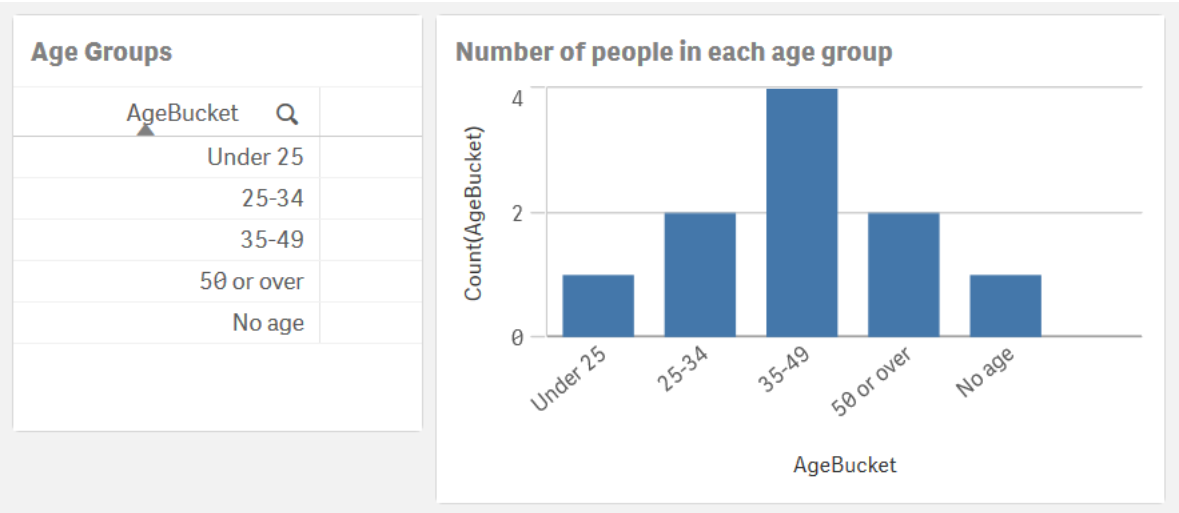
   Die Skriptausgabe gleicht der Folgenden:

   Tabelle nach Verwendung von Exists im Skript



   Wurde keiner der Wert für *PersonID* in der Tabelle *AgeTemp* eingelesen, dann werden die Felder *Age* und *AgeBucket* nicht in die Tabelle *People* eingepflegt. Mit der Funktion Exists() können Sie verwaiste Aufzeichnungen/Daten im Datenmodell verhindern – z. B. *Age*- und *AgeBucket*-Felder ohne zugehörige Personen.

6. Erstellen Sie ein neues Arbeitsblatt und geben Sie ihm einen Namen.
7. Öffnen Sie das neue Arbeitsblatt und klicken Sie auf **Arbeitsblatt bearbeiten**.
8. Fügen Sie eine Standardtabelle zum Arbeitsblatt mit der Dimension *AgeBucket* hinzu und benennen Sie die Visualisierung mit *Altersgruppen*.
9. Fügen Sie dem Arbeitsblatt ein Balkendiagramm mit der Dimension *AgeBucket* und der Kennzahl *Count ([AgeBucket])* hinzu. Nennen Sie die Visualisierung *Number of people in each age group*.
10. Passen Sie die Eigenschaften der Tabelle und des Balkendiagramms nach Ihren Wünschen an und klicken Sie dann auf **Erledigt**.
    Das Arbeitsblatt sollte folgendermaßen aussehen:
    Arbeitsblatt mit Gruppierungen nach Alter



Die Funktion Dual() ist nützlich im Skript oder in einer Diagrammformel, wenn einem String ein Zahlenwert zugewiesen werden muss.

Im obigen Skript gibt es eine Anwendung, die das Alter einliest, und Sie haben sich entschieden, diese Werte in Spannen zu unterteilen, damit Sie Visualisierungen auf Basis der Altersspannen im Vergleich zum tatsächlichen Alter erstellen können. Es gibt eine Spanne für Personen unter 25 Jahren, von 25 bis 35 Jahren usw. Mit der Funktion Dual() kann diesen Altersspannen ein Zahlenwert zugewiesen werden, welcher später zum Sortieren der Altersspannen in einer Listbox oder einem Diagramm verwendet werden kann. Wie im Arbeitsblatt der App werden die Daten ohne Altersangabe an das Ende der Liste gesetzt.

Weitere Informationen zu Exists() und Dual() finden Sie in folgendem Blog-Eintrag in Qlik Community: Dual & Exists – Useful Functions *(Dual und Vorhanden – nützliche Funktionen)*

3.4 Einordnung in Intervalle und iterative Ladevorgänge

Der Intervalmatch-Zusatz einer LOAD- oder SELECT-Anweisung wird verwendet, um diskrete numerische Werte mit einem oder mehreren numerischen Intervallen zu verknüpfen. Diese leistungsfähige Funktion kann beispielsweise in Produktionsumgebungen verwendet werden.

Verwenden des IntervalMatch()-Zusatzes

Der grundlegendste Intervallabgleich ergibt sich bei einer Liste mit Zahlen oder Daten (Ereignissen) in einer Tabelle und einer Liste mit Intervallen in einer zweiten Tabelle. Das Ziel ist die Verlinkung der beiden Tabellen. In der Regel handelt es sich dabei um eine n:n-Beziehung, d. h. einem Intervall können viele Datumswerte angehören und ein Datum kann vielen Intervallen angehören. Um diese aufzulösen, müssen Sie als Verbindung zwischen den beiden ursprünglichen Tabellen eine Brückentabelle erstellen. Dazu gibt es verschiedene Möglichkeiten.

Am einfachsten lässt sich dieses Problem in Qlik Sense lösen, indem der Zusatz IntervalMatch() vor einer LOAD- oder einer SELECT-Anweisung verwendet wird. Die LOAD/SELECT-Anweisung darf nur zwei Felder enthalten, und zwar die Felder From und To, welche die Intervalle definieren. Der Zusatz IntervalMatch() generiert daraufhin alle Kombinationen zwischen den geladenen Intervallen und einem zuvor geladenen numerischen Feld, das als Parameter für den Zusatz festgelegt wurde.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie eine neue App und geben Sie ihm einen Namen.
2. Fügen Sie einen neuen Skriptabschnitt im **Dateneditor** hinzu.
3. Rufen Sie den Abschnitt *Events* auf.
4. Klicken Sie im rechten Menü unter **AttachedFiles** auf **Daten auswählen**.
5. Laden Sie *Events.txt* hoch und wählen Sie die Datei aus.
6. Klicken Sie im Fenster **Daten auswählen aus** auf **Skript einfügen**.
7. Laden Sie *Intervals.txt* hoch und wählen Sie die Datei aus.
8. Klicken Sie im Fenster **Daten auswählen aus** auf **Skript einfügen**.
9. Nennen Sie im Skript die erste Tabelle *Events* und die zweite Tabelle *Intervals*.
10. Fügen Sie am Ende des Skripts IntervalMatch hinzu, um eine dritte Tabelle zu erstellen, welche die beiden ersten Tabellen verbindet:

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

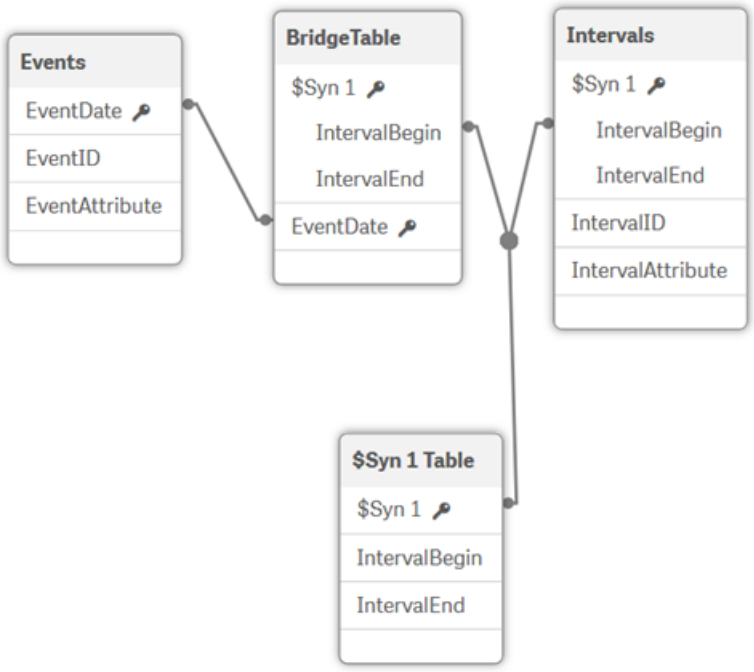
11. Ihr Skript sollte folgendermaßen aussehen:

    ```
    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. Klicken Sie auf **Daten laden**.
13. Öffnen Sie die **Datenmodellansicht**. Das Datenmodell sieht folgendermaßen aus:

Datenmodell: Tabellen Events, BridgeTable, Intervals und $Syn1



Das Datenmodell enthält einen zusammengesetzten Schlüssel (die Felder *IntervalBegin* und *IntervalEnd*), der sich als synthetischer Qlik Sense-Schlüssel manifestiert.

Die grundlegenden Tabellen sind:

- Die *Events*-Tabelle enthält genau einen Datensatz pro Ereignis.
- Die *Intervals*-Tabelle enthält genau einen Datensatz pro Intervall.
- Die Brückentabelle, die genau einen Datensatz pro Kombination aus Ereignis und Intervall enthält und die beiden vorherigen Tabellen verlinkt.

Beachten Sie, dass ein Ereignis verschiedenen Intervallen angehören kann, wenn sich die Intervalle überschneiden. Einem Intervall können selbstverständlich auch verschiedene Ereignisse angehören.

Dieses Datenmodell ist insofern optimal, da es normalisiert und kompakt ist. Die Tabelle *Events* und die Tabelle *Intervals* bleiben beide unverändert und enthalten die ursprüngliche Anzahl an Datensätzen. Alle Berechnungen von Qlik Sense, die für diese Tabellen ausgeführt werden, beispielsweise Count(EventID), funktionieren und werden korrekt evaluiert.

Weitere Informationen zur Verwendung von IntervalMatch() finden Sie in folgendem Blog-Eintrag in Qlik Community: Using IntervalMatch() *(IntervalMatch() verwenden)*

Verwenden einer While-Schleife und iterativer Ladevorgänge IterNo()

Beinahe dieselbe Brückentabelle lässt sich mithilfe einer While-Schleife und IterNo() erreichen. Hierbei werden zwischen der unteren und oberen Grenze des Intervalls abzählbare Werte erstellt.

Eine Schleife in der LOAD-Anweisung lässt sich mithilfe einer While-Schleife erstellen. Hier ein Beispiel:

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

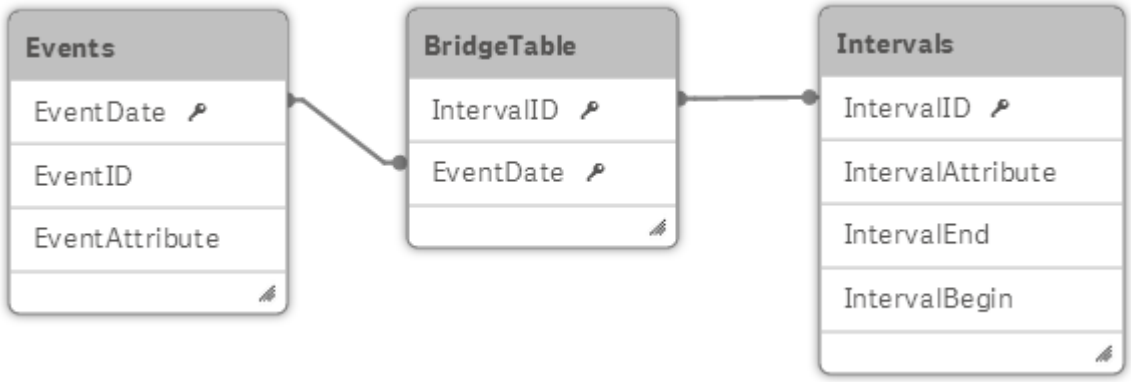
Eine derartige LOAD-Anweisung wird als Schleife für jeden Datensatz ausgeführt und immer wieder geladen, solange die Formel in der While-Bedingung zutrifft. Die Funktion IterNo() liefert "1" für die erste Iteration, "2" für die zweite usw.

Für die Intervalle verfügen Sie über einen Primärschlüssel, die IntervalID, der einzige Unterschied im Skript ergibt sich deshalb aus der Art und Weise, wie die Brückentabelle erstellt wird:

Gehen Sie folgendermaßen vor:

1. Ersetzen Sie die bestehenden Bridgetable-Anweisungen durch das folgende Skript:
   ```
   BridgeTable:
   LOAD distinct * Where Exists(EventDate);
   LOAD IntervalBegin + IterNo() - 1 as EventDate, IntervalID
       Resident Intervals
       While IntervalBegin + IterNo() - 1 <= IntervalEnd;
   ```
2. Klicken Sie auf **Daten laden**.
3. Öffnen Sie die **Datenmodellansicht**. Das Datenmodell sieht folgendermaßen aus:

Datenmodell: Tabellen Events, BridgeTable und Intervals



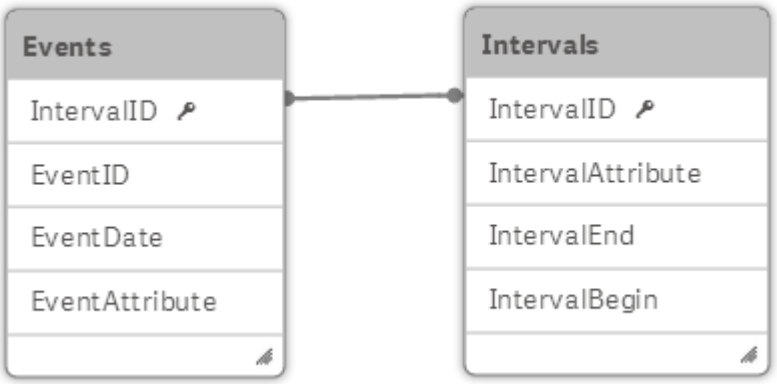
Generell ist die Lösung mit den drei Tabellen die beste, weil dadurch eine n:n-Beziehung zwischen Intervallen und Ereignissen möglich ist. Häufig ist jedoch auch bekannt, dass ein Ereignis nur zu einem einzigen Intervall gehören kann. In einem solchen Fall ist die Brückentabelle eigentlich nicht erforderlich. Die *IntervalID* kann direkt in der Ereignistabelle gespeichert werden. Es gibt verschiedene Möglichkeiten, dies zu erreichen, aber am hilfreichsten ist es, Bridgetable mit der Tabelle *Events* zu verbinden.

4. Fügen Sie am Ende Ihres Skripts folgendes Skript hinzu:

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

Drop Table BridgeTable;
```

5. Klicken Sie auf **Daten laden**.
6. Öffnen Sie die **Datenmodellansicht**. Das Datenmodell sieht folgendermaßen aus:

   Datenmodell: Tabellen Events und Intervals



Offene und abgeschlossene Intervalle

Ob ein Intervall offen oder abgeschlossen ist, hängt von den Endpunkten ab, und zwar davon, ob diese im Intervall eingeschlossen sind oder nicht.

- Sind die Endpunkte eingeschlossen, handelt es sich um ein abgeschlossenes Intervall:
  $[a,b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$
- Sind die Endpunkte nicht eingeschlossen, handelt es sich um ein offenes Intervall:
  $]a,b[= \{x \in \mathbb{R} \mid a < x < b\}$
- Ist ein Endpunkt abgeschlossen, handelt es sich um ein zur Hälfte geöffnetes Intervall:
  $[a,b[= \{x \in \mathbb{R} \mid a \leq x < b\}$

Für den Fall, dass sich die Intervalle überschneiden und eine Zahl zu mehreren Intervallen gehören kann, sind normalerweise geschlossene Intervalle erforderlich.

In bestimmten Fällen sind überlappende Intervalle jedoch möglicherweise nicht erwünscht und eine bestimmte Zahl soll nur einem bestimmten Intervall angehören. Daher ergibt sich ein Problem, wenn ein Punkt das Ende eines Intervalls und zugleich der Beginn des nächsten ist. Eine Zahl mit diesem Wert würde beiden Intervallen zugewiesen werden. Aus diesem Grund sind zur Hälfte geöffnete Intervalle wünschenswert.

Eine praktische Lösung dieses Problems ist, eine geringe Zahl vom Endwert aller Intervalle abzuziehen. Auf diese Weise werden geschlossene, aber nicht überlappende Intervalle erstellt. Wenn es sich bei den Zahlen um Daten handelt, lässt sich dies am einfachsten mithilfe der Funktion DayEnd() erzielen. Diese gibt die letzte Millisekunde des Tages zurück:

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

Sie können auch einen kleinen Betrag manuell abziehen. Beachten Sie dabei aber, dass der abgezogene Betrag nicht zu klein ist, da die Operation auf 52 signifikante Binärstellen (14 Dezimalstellen) gerundet wird. Wenn Sie einen zu kleinen Betrag verwenden, ist der Unterschied nicht signifikant, wodurch wieder die ursprüngliche Zahl herangezogen wird.

4 Datenpflege durch Mapping

Es kann vorkommen, dass die Quelldaten, die Sie in Qlik Sense laden, nicht so beschaffen sind, wie Sie sie in der Qlik Sense App haben wollen. Qlik Sense bietet viele Funktionen und Befehle, durch die Sie Ihre Daten in ein Format umwandeln können, das Ihnen weiterhilft.

Mapping kann in einem Qlik Sense Skript bei Ausführung des Skripts zum Ersetzen oder Ändern von Feldwerten oder Namen verwendet werden. So kann Mapping verwendet werden, um Daten zu löschen und konsistenter zu machen oder einen Feldwert ganz oder teilweise zu ersetzen.

In der Praxis haben Werte aus mehreren Tabellen oft unterschiedliche Bezeichnungen oder Schreibweisen, obwohl der Informationsgehalt derselbe ist. Dadurch kommen in der Software inhaltlich korrekte Verknüpfungen nicht zustande, wenn die Schreibweise nicht exakt übereinstimmt. Dieses Problem lässt sich durch Mapping beheben.

4.1 Mapping-Tabellen

Tabellen, die über Mapping load oder Mapping select geladen werden, werden anders als andere Tabellen behandelt. Sie werden lediglich während der Ausführung des Skripts für das Mapping benötigt und separat gespeichert. Nach der Ausführung des Skripts werden diese Tabellen automatisch gelöscht.

Es gilt:

- Mapping-Tabellen bestehen aus zwei Spalten: Die erste enthält Vergleichswerte, die zweite die gewünschten Mapping-Werte.
- Die zwei Spalten müssen benannt werden, doch die Namen haben keine Relevanz. Die Spaltennamen müssen keine Verbindung zu Feldnamen in normalen internen Tabellen haben.

4.2 Mapping-Funktionen und -Anweisungen

Die folgenden Mapping-Funktionen/-Anweisungen werden in diesem Tutorial behandelt:

• Mapping-Zusatz

• ApplyMap()

• MapSubstring()

• Map … Using-Anweisung

• Unmap-Anweisung

4.3 Mapping-Zusatz

Mit dem Mapping-Zusatz wird eine Mapping-Tabelle in einem Skript erstellt. Die Mapping-Tabelle kann dann mit der ApplyMap()-Funktion, der MapSubstring()-Funktion oder der Map … Using-Anweisung verwendet werden.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie eine neue App und geben Sie ihm einen Namen.
2. Fügen Sie einen neuen Skriptabschnitt im **Dateneditor** hinzu.
3. Rufen Sie den Abschnitt *Countries* auf.
4. Geben Sie folgendes Skript ein:

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

   Die Tabelle *CountryMap* speichert zwei Spalten: *Country* und *NewCountry*. Die Spalte *Country* speichert die verschiedenen Weisen, wie das Land in das Feld *Country* eingegeben wurde. Die Spalte *NewCountry* speichert, wie die Werte zugeordnet werden. Diese Mapping-Tabelle wird zum Speichern von konsistenten *US*-Länderwerten im Feld *Country* verwendet. Wenn zum Beispiel *U.S.A.* im Feld *Country* gespeichert wird, ordnen Sie ihm *US* zu.

4.4 ApplyMap()-Funktion

Verwenden Sie ApplyMap(), um Daten in einem Feld basierend auf einer zuvor erstellten Mapping-Tabelle zu ersetzen. Die Mapping-Tabelle muss geladen werden, bevor die ApplyMap()-Funktion verwendet werden kann. Die Daten in der Tabelle *Data.xlsx*, die Sie laden werden, sehen wie folgt aus:

Datentabelle

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

Beachten Sie, dass das Land auf verschiedene Weisen eingegeben wurde. Damit das Länderfeld konsistent ist, wird die Mapping-Tabelle geladen und dann die **ApplyMap()**-Funktion verwendet.

Gehen Sie folgendermaßen vor:

1. Wählen Sie unter dem oben eingegebenen Skript *Data.xlsx* aus, laden Sie die Datei und fügen Sie dann das Skript ein.
2. Geben Sie Folgendes oberhalb der soeben erstellten LOAD-Anweisung ein:

```
Data:
```

Ihr Skript sollte folgendermaßen aussehen:

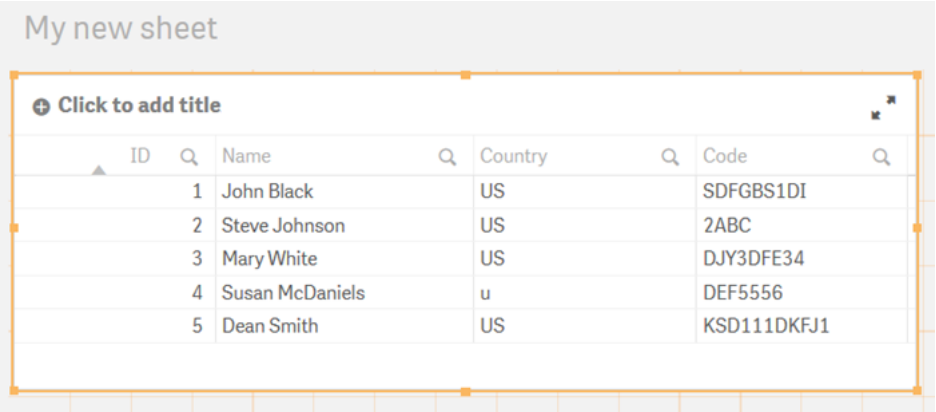
```
CountryMap:
MAPPING LOAD * INLINE [
    Country, NewCountry
    U.S.A., US
    U.S., US
    United States, US
    United States of America, US
];

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

3. Ändern Sie die Zeile mit `Country`, wie folgt:
   `ApplyMap('CountryMap', Country) as Country,`

   Beim ersten Parameter der Funktion ApplyMap() ist der Zuordnungsname von einfachen Anführungszeichen umschlossen. Der zweite Parameter ist das Feld, dessen Daten ersetzt werden sollen.

4. Klicken Sie auf **Daten laden**.
   Die sich ergebende Tabelle sieht folgendermaßen aus:

   Tabelle mit den Daten, die anhand der ApplyMap()-Funktion geladen wurden



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

Die verschiedenen Schreibweisen von *United States* wurden alle in *US* geändert. Es gibt einen Datensatz, der nicht richtig geschrieben wurde, weshalb die ApplyMap()-Funktion diesen Feldwert nicht geändert hat. Mithilfe der ApplyMap()-Funktion können Sie den dritten Parameter zum Hinzufügen einer Standardformel verwenden, wenn die Mapping-Tabelle keinen passenden Wert besitzt.

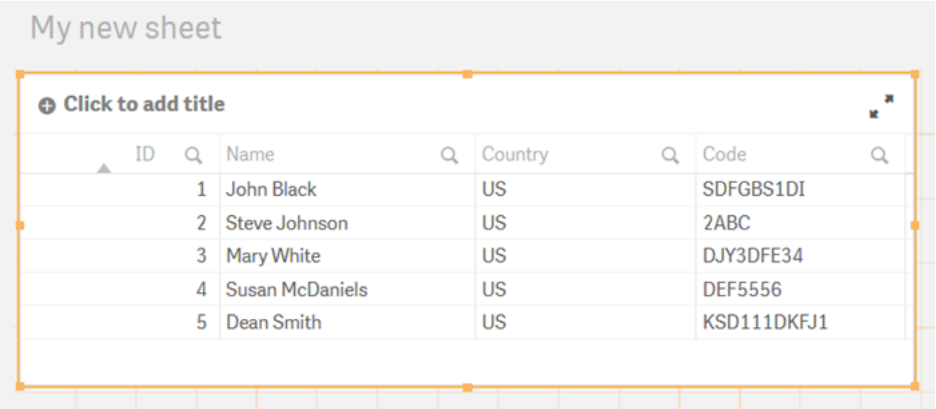
5. Fügen Sie ''US' als dritten Parameter der ApplyMap()-Funktion hinzu, um Fällen beizukommen, in denen das Land falsch eingegeben wurde:
   ```
   ApplyMap('CountryMap', Country, 'US') as Country,
   ```

   Ihr Skript sollte folgendermaßen aussehen:

   ```
   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. Klicken Sie auf **Daten laden**.
   Die sich ergebende Tabelle sieht folgendermaßen aus:

   Tabelle mit den Daten, die anhand der ApplyMap-Funktion geladen wurden



Weitere Informationen zur Verwendung von ApplyMap() finden Sie in folgendem Blog-Eintrag in Qlik Community: Don't join - use Applymap instead *(Nicht verknüpfen – stattdessen Applymap verwenden)*

4.5 MapSubstring()-Funktion

Die MapSubstring()-Funktion ermöglicht Ihnen, Teile eines Felds zuzuordnen.

In der durch ApplyMap() erstellten Tabelle wollen wir die Zahlen als Text schreiben. Dafür ersetzen wir mit der MapSubstring()-Funktion die numerischen Daten durch Text.

Zunächst muss eine Mapping-Tabelle erstellt werden.

Gehen Sie folgendermaßen vor:

1. Fügen Sie die folgenden Skriptzeilen nach dem Abschnitt *CountryMap*, aber vor dem Abschnitt *Data* hinzu.

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

   In der *CodeMap*-Tabelle erfolgt ein Durchlauf der Zahlen 1 bis 5 und 11.

2. Ändern Sie im Abschnitt *Data* des Skripts den Code-Befehl wie folgt:

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

   Ihr Skript sollte folgendermaßen aussehen:

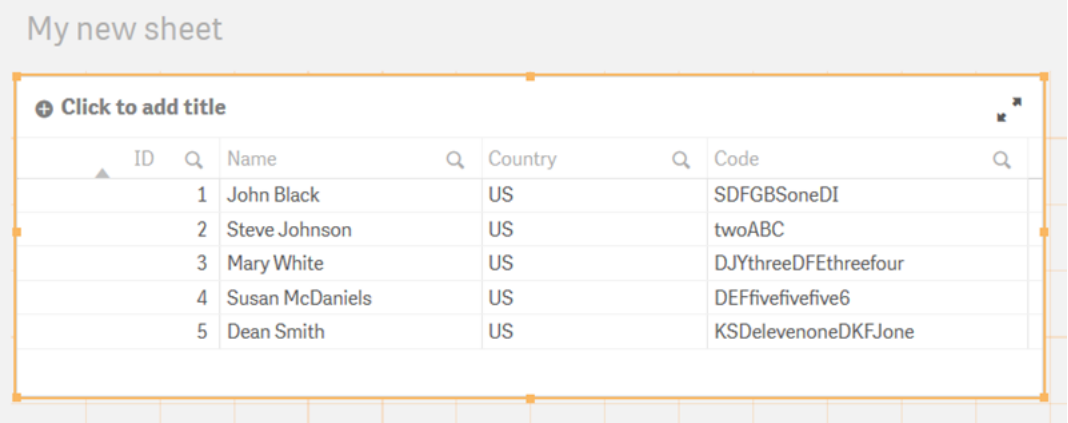
   ```
   CountryMap:
   MAPPING LOAD * INLINE [
       Country, NewCountry
       U.S.A., US
       U.S., US
       United States, US
       United States of America, US
   ];

   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. Klicken Sie auf **Daten laden**.

   Die sich ergebende Tabelle sieht folgendermaßen aus:

   Tabelle mit den Daten, die anhand der MapSubString-Funktion geladen wurden



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

   Die numerischen Zeichen wurden im *Code*-Feld durch Text ersetzt. Wenn eine Zahl wie bei ID=3 und ID=4 mehr als einmal erscheint, wird der Text auch wiederholt. ID=4. *Susan McDaniels* hatte eine 6 in ihrem Code. Da die 6 in der *CodeMap*-Tabelle nicht zugeordnet wurde, bleibt sie unverändert. ID=5, *Dean Smith* hatte 111 in seinem Code. Dies wurde als 'elevenone' zugeordnet.

Weitere Informationen zur Verwendung von MapSubstring() finden Sie in folgendem Blog-Eintrag in Qlik Community: Mapping … and not the geographical kind *(Mapping ... und zwar nicht der geografische Typ)*

4.6 Map … Using

Die Map … Using-Anweisung kann auch zum Anwenden einer Zuordnung auf ein Feld verwendet werden. Sie funktioniert aber ein wenig anders als ApplyMap(). Während ApplyMap() jedes Mal beim Stoßen auf den Feldnamen das Mapping bedient, erfolgt dies bei Map … Using nur, wenn der Wert unter dem Feldnamen in der internen Tabelle gespeichert ist.

Werfen wir einen Blick auf ein Beispiel. Angenommen, wir laden das *Country*-Feld mehrere Male im Skript und möchten jedes Mal, wenn das Feld geladen wird, eine Zuordnung anwenden. Wir können die ApplyMap()-Funktion wie zuvor in diesem Tutorial dargestellt oder Map … Using verwenden.

Bei Verwendung von Map … Using wird die Zuordnung auf das Feld angewendet, wenn dieses in der internen Tabelle gespeichert ist. Also wird im Beispiel unten die Zuordnung auf das *Country*-Feld in der Data1-Tabelle angewendet, aber nicht auf das *Country2*-Feld in der *Data2*-Tabelle. Das ist darauf zurückzuführen, dass der Map … Using-Befehl nur auf Felder mit dem Namen *Country* angewendet wird. Wenn das *Country2*-Feld in der internen Tabelle gespeichert wird, heißt es nicht länger *Country*. Wenn Sie die Zuordnung auf die Tabelle *Country2* anwenden wollen, müssen Sie die ApplyMap()-Funktion verwenden.

Der Unmap-Befehl beschließt den Map … Using-Befehl, d. h. wenn *Country* nach dem Unmap-Befehl geladen werden soll, wird *CountryMap* nicht angewendet.

Gehen Sie folgendermaßen vor:

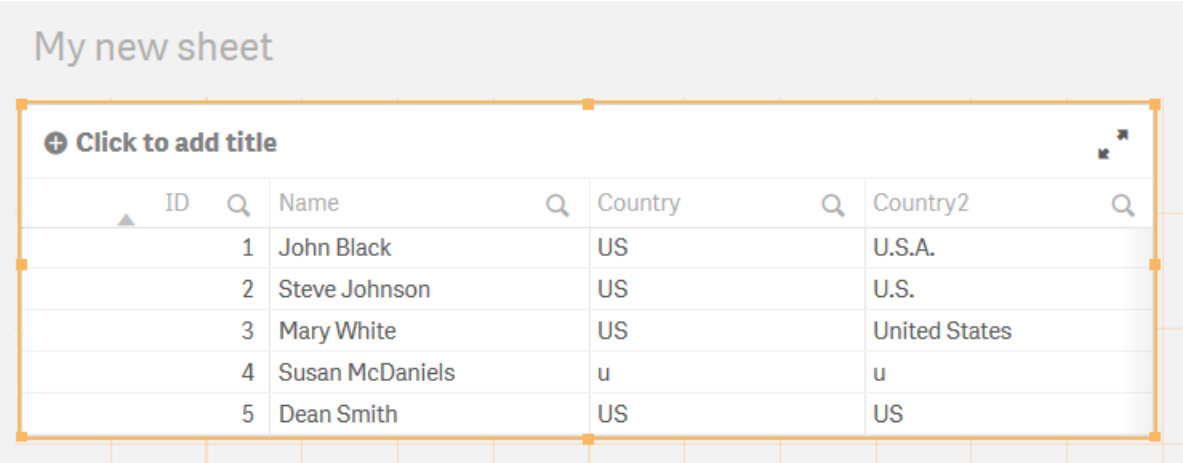
1. Ersetzen Sie das Skript für die Tabelle *Data* durch Folgendes:

```
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. Klicken Sie auf **Daten laden**.

   Die sich ergebende Tabelle sieht folgendermaßen aus:

   Tabelle mit den Daten, die anhand der Map … Using-Funktion geladen wurden



5 Handhabung hierarchischer Daten

Hierarchien sind ein wichtiger Bestandteil von Business Intelligence Lösungen, die dazu herangezogen werden, Dimensionen zu beschreiben, die von Haus aus unterschiedliche Detailtiefen aufweisen. Manche sind einfach und intuitiv, während andere recht komplex sind und eingehend durchdacht werden müssen, um richtig dargestellt zu werden.

Die einzelnen Hierarchiepunkte nehmen von oben nach unten stets an Detailtiefe zu. Beispiel: In einer Dimension mit den Ebenen Markt, Land, Bundesland und Stadt befindet sich Amerika auf der obersten Hierarchieebene, die USA befinden sich auf der zweiten Ebene, Kalifornien auf der dritten Ebene und San Francisco auf unterster Ebene. Kalifornien ist spezifischer als USA und San Francisco wiederum spezifischer als Kalifornien.

Das Speichern von hierarchischen Strukturen in einem relationalen Modell ist eine häufig anzutreffende, anspruchsvolle Aufgabe, für die es mehrere Lösungen gibt. Es gibt diverse Lösungsansätze:

- Die horizontale Hierarchie
- Das Adjazenzlisten-Modell
- Die Path Enumeration Method
- Das Nested-Sets-Modell
- Der Stammbaum

Im Rahmen dieses Tutorials erstellen wir einen Stammbaum, da dies eine hierarchische Struktur ist, die wir direkt für eine Anfrage verwenden können. Weitere Informationen zu den anderen Lösungsansätzen finden Sie in der Qlik Community.

5.1 Hierarchy -Zusatz

Der Hierarchy-Zusatz ist ein Skriptbefehl, der vor einem LOAD oder SELECT-Befehl zum Einsatz kommt, mithilfe dessen eine Tabelle mit benachbarten Knoten geladen wird. Der LOAD-Befehl muss mindestens drei Felder enthalten: eine ID als eindeutigen Schlüssel für den Knoten, eine Referenz zum übergeordneten Knoten und einen Namen.

Durch den Zusatz wird die geladene Tabelle in eine Tabelle mit erweiterten Knoten umgewandelt. Diese Tabelle verfügt über zusätzliche Spalten, eine Spalte für jede Hierarchieebene.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie eine neue App und geben Sie ihm einen Namen.
2. Fügen Sie einen neuen Skriptabschnitt im **Dateneditor** hinzu.
3. Rufen Sie den Abschnitt *Wine* auf.
4. Klicken Sie im rechten Menü unter **AttachedFiles** auf **Daten auswählen**.
5. Laden Sie *Winedistricts.txt* hoch und wählen Sie die Datei aus.
6. Deaktivieren Sie im Fenster **Daten auswählen aus** die Felder *Lbound* und *RBound*, damit sie nicht geladen werden.

7. Klicken Sie auf **Skript einfügen**.
8. Geben Sie Folgendes oberhalb der LOAD-Anweisung ein:
   ```
   Hierarchy (NodeID, ParentID, NodeName)
   ```

   Ihr Skript sollte folgendermaßen aussehen:

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

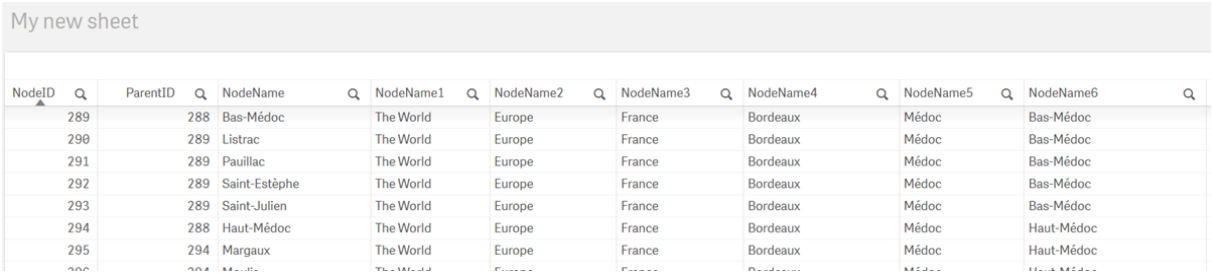
9. Klicken Sie auf **Daten laden**.
10. Verwenden Sie zum Anzeigen der resultierenden Tabelle den **Vorschau**-Abschnitt der **Datenmodellansicht**.

    Die erstellte Tabelle mit aufgeschlüsselten Ebenen verfügen über exakt so viele Datensätze wie die Quelltabelle: einen pro Knoten. Die Tabelle mit aufgeschlüsselten Ebenen ist ziemlich praktisch, da sie eine Reihe entscheidender Voraussetzungen für die Analyse einer Hierarchie in einem relationalen Modell erfüllt:

    - Alle Knotennamen sind in einer Spalte aufgeführt, sodass sie für die Suche herangezogen werden können.
    - Zudem wurden die verschiedenen Knotenebenen auf jeweils ein Feld aufgeschlüsselt. Diese Felder können in Drilldown-Gruppen oder als Dimensionen in Pivottabellen verwendet werden.
    - Zudem wurden die verschiedenen Knotenebenen auf jeweils ein Feld aufgeschlüsselt. Diese Felder können für Drilldown-Gruppen verwendet werden.
    - Sie kann einen eindeutigen Pfad für den Knoten enthalten, der alle übergeordneten Knoten in der richtigen Reihenfolge auflistet.
    - Sie kann auch die Anzahl der Ebenen zwischen dem Knoten und der Wurzel enthalten.

    Die sich ergebende Tabelle sieht folgendermaßen aus:

    Tabelle mit Beispieldaten, die mit dem Hierarchy-Zusatz geladen wurden

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 HierarchyBelongsTo-Zusatz

Wie der Hierarchy-Zusatz ist der HierarchyBelongsTo-Zusatz ein Skriptbefehl, der vor einer LOAD- oder SELECT-Anweisung zum Einsatz kommt, mithilfe dessen eine Tabelle mit benachbarten Knoten geladen wird.

Auch hier muss die LOAD-Anweisung mindestens drei Felder enthalten: eine ID als eindeutigen Schlüssel für den Knoten, eine Referenz zum übergeordneten Knoten und einen Namen. Dieser Zusatz wandelt die geladene Tabelle in eine Stammtabelle um. Diese Tabelle führt alle Kombinationen von übergeordneten und nachgeordneten Knoten in einem gesonderten Datensatz auf. Daher lassen sich alle übergeordneten und untergeordneten Knoten eines bestimmten Knotens einfach ausfindig machen.

Gehen Sie folgendermaßen vor:

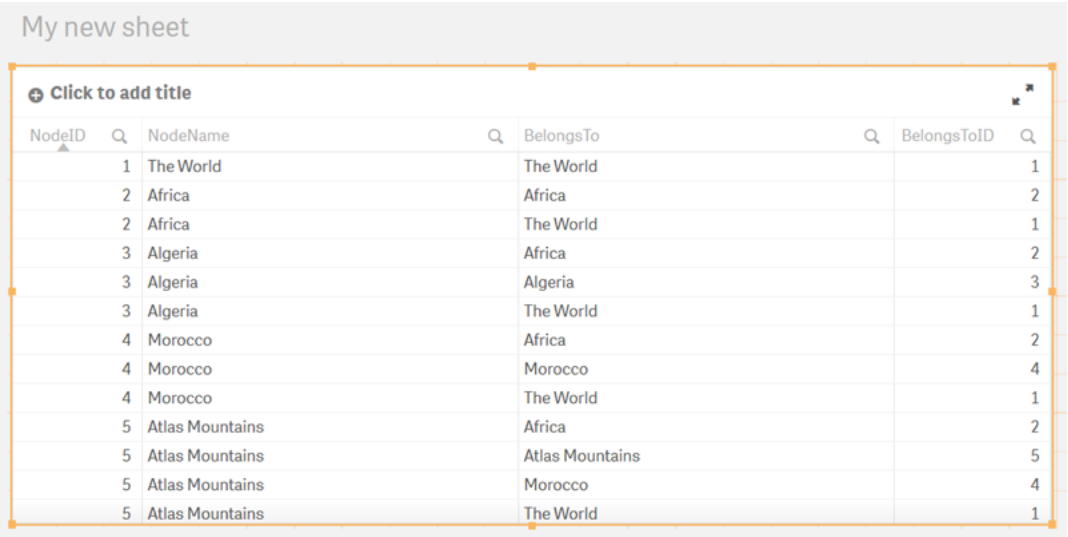
1. Ändern Sie die Hierarchy-Anweisung im **Dateneditor** wie folgt:
   `HierarchyBelongsTo (NodeID, ParentID, NodeName, BelongsToID, BelongsTo)`
2. Klicken Sie auf **Daten laden**.
3. Verwenden Sie zum Anzeigen der resultierenden Tabelle den **Vorschau**-Abschnitt der **Datenmodellansicht**.

   Die Stammtabelle erfüllt eine Reihe entscheidender Voraussetzungen für die Analyse einer Hierarchie in einem relationalen Modell:

   - Falls die Knoten-ID die einzelnen Knoten darstellt, stellt die ID der übergeordneten Knoten den gesamten Baum und die Teilbäume der Hierarchie dar.
   - Alle Knotennamen bestehen sowohl in der Rolle als Knoten als auch in der Rolle als Baum und beide können für die Suche herangezogen werden.
   - Die Stammtabelle kann auch die Tiefendifferenz zwischen der Knotentiefe und der Tiefe des übergeordneten Knotens enthalten, d. h. den Abstand von der Wurzel des Teilbaums.

   Die sich ergebende Tabelle sieht folgendermaßen aus:

   Tabelle mit den Daten, die mit dem HierarchyBelongsTo-Zusatz geladen wurden



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

Autorisierung

Hierarchien werden nicht selten für die Autorisierung verwendet. Ein Beispiel dafür ist die Organisationshierarchie. Ein Führungsverantwortlicher sollte Einsicht in alles haben, was zu seiner Abteilung gehört, auch in sämtliche Unterabteilungen. Er sollte jedoch nicht zwingend Einsicht in andere Abteilungen

haben.

Beispiel einer Organisationshierarchie

Das bedeutet, dass unterschiedliche Personen andere Teilbäume der Organisation einsehen können. Die Autorisierungstabelle sieht möglicherweise folgendermaßen aus:

Autorisierungstabelle

ACCESS	NTNAME	PERSON	POSITION	PERMISSIONS
USER	ACME\JRL	John	CPO	Personal
USER	ACME\CAH	Carol	CEO	CEO
USER	ACME\JER	James	Technischer Leiter	Technik
USER	ACME\DBK	Diana	CFO	Finanzen
USER	ACME\RNL	Bob	COO	Verkauf
USER	ACME\LFD	Larry	CTO	Produkt

In diesem Fall kann *Carol* alles ansehen, was zu *CEO* und darunter gehört; *Larry* kann die *Product*-Organisation ansehen und *James* kann lediglich die *Engineering*-Organisation einsehen.

Beispiel:

Oft wird die Hierarchie in einer Tabelle mit benachbarten Knoten gespeichert. Laden Sie zur Behebung des Problems in diesem Beispiel die Tabelle mit benachbarten Knoten anhand von HierarchyBelongsTo und nennen Sie das Feld des übergeordneten Knotens *Tree*.

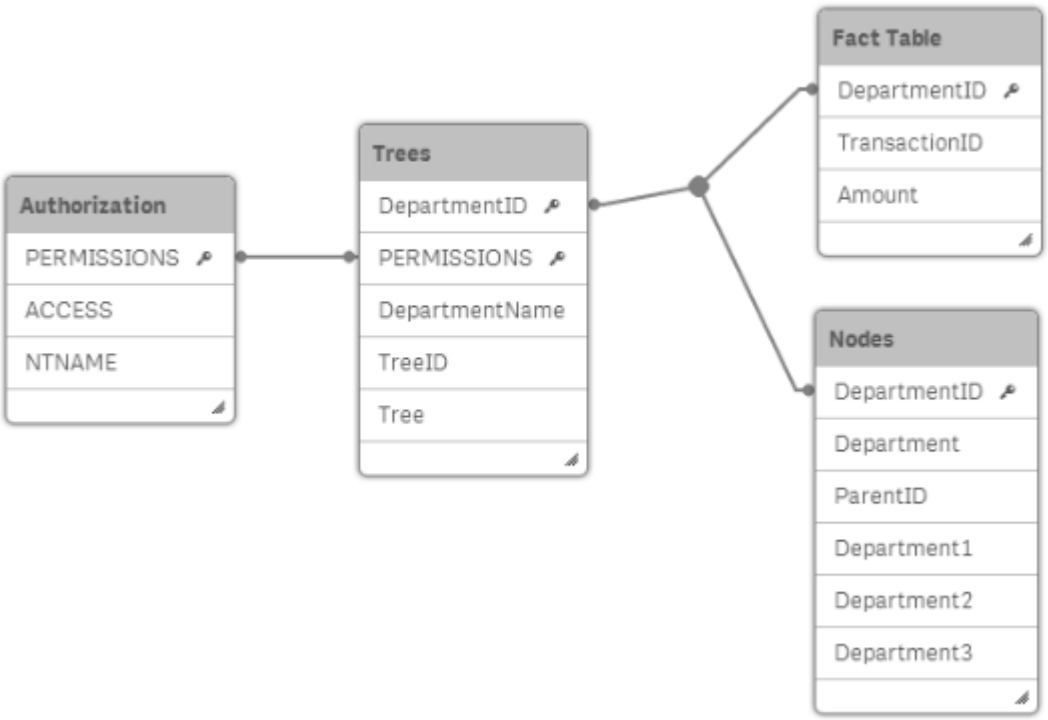
Wenn Sie Section Access verwenden möchten, laden Sie eine Kopie in Großbuchstaben von *Tree* und nennen Sie dieses neue Feld *PERMISSIONS*. Abschließend müssen Sie die Autorisierungstabelle laden. Die letzten beiden Schritte können mithilfe der folgenden Skriptzeilen durchgeführt werden. Beachten Sie, dass die TempTrees-Tabelle die Tabelle ist, die mit der HierarchyBelongsTo-Anweisung erstellt wurde.

Beachten Sie, dass es sich hierbei nur um ein Beispiel handelt. Es sind keine in Qlik Sense zu bearbeitenden Übungen vorhanden.

```
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;
```

Mit diesem Beispiel wird das folgende Datenmodell erstellt:

Datenmodell: Tabellen Authorization, Trees, Fact und Nodes



6 QVD-Dateien

QVD (QlikView Data)-Dateien enthalten Datentabellen, die aus Qlik Sense oder QlikView exportiert werden. QVD ist ein natives Qlik Format, das nur von Qlik Sense oder QlikView gelesen oder geschrieben werden kann. Das Dateiformat ist für besonders schnelles Laden aus einem Qlik Sense-Skript optimiert, aber gleichzeitig sehr kompakt. Das Einlesen von Daten aus QVD-Dateien ist etwa 10-100 Mal schneller als das Einlesen aus anderen Datenquellen.

QVD-Dateien können in zwei Modi gelesen werden: Standard (schnell) und optimiert (schneller). Der ausgewählte Modus ergibt sich automatisch aus dem Qlik Sense-Skriptmodul. Der optimierte Modus kann nur verwendet werden, wenn alle geladenen Felder oder Teile davon ohne Umformung (Formeln, die an den Feldern aktiv werden) eingelesen werden. Felder dürfen jedoch umbenannt werden. Eine Where-Bedingung, bei der Qlik Sense die Datensätze auspackt, deaktiviert auch die optimierte Ladung.

Eine QVD-Datei enthält immer genau eine Tabelle und besteht aus drei Teilen:

- Ein XML-Header (im UTF-8 Zeichensatz), der Metadaten enthält, z. B. eine Beschreibung der Felder und des Aufbaus der Tabelle.
- Symboltabellen, unter Anwendung von Bytestuffing.
- Die eigentlichen Tabellendaten, unter Anwendung von Bitstuffing.

QVD-Dateien sind in verschiedenen Situationen hilfreich. Vier häufige Anwendungsbereiche sind schnell genannt. Oft sprechen auch gleich mehrere Gründe für den Einsatz von QVD-Dateien.

- Verkürzte Ladezeiten

  Daten, die sich gar nicht oder nur in großen Zeitabständen verändern, lassen sich in QVD-Dateien puffern. Dadurch wird für die Ausführung des Skripts erheblich weniger Zeit benötigt. Dieser Effekt macht sich insbesondere bei großen Datenmengen bemerkbar.

- Entlastung für Datenbankserver

  Die aus externen Quellen geladene Datenmenge verringert sich erheblich. Datenbankserver und Netzwerke werden entlastet, weil weniger Daten bewegt werden. Wenn mehrere Qlik Sense-Skripte dieselben Daten verwenden, müssen sie nur einmal aus der Quelldatenbank in eine QVD-Datei geladen werden. Die anderen Anwendungen können über diese QVD-Datei die gleichen Daten verwenden.

- Integration von Daten aus mehreren Qlik Sense-Anwendungen

  Mit dem Skriptbefehl Binary lassen sich nur Daten aus einer Qlik Sense-Anwendung in eine andere laden. Mit QVD-Dateien jedoch kann ein Qlik Sense-Skript Daten aus beliebig vielen Qlik Sense-Anwendungen kombinieren. Ein möglicher Anwendungsbereich ist z. B. der Vergleich von Daten aus verschiedenen Abteilungen eines Betriebs.

- Inkrementelles Laden

  In vielen Fällen kann man QVD-Dateien für inkrementelle Ladevorgänge benutzen, bei denen nur neue oder veränderte Datensätze aus einer umfangreichen Datenbank gelesen werden.

Informationen dazu, wie die Qlik Community Qlik Application Automation zum Verbessern der QVD-Ladezeiten verwendet, finden Sie unter Teilen von QVDs anhand einer Automatisierung zum Verbessern von Ladevorgängen.

6.1 Erstellen von QVD-Dateien

Eine QVD-Datei kann auf zwei verschiedene Arten erstellt werden:

- Explizite Erstellung und Benennung mithilfe des Store-Befehls im Qlik Sense Skript.

  Dazu definieren Sie im Skript, dass eine bereits eingelesene Tabelle oder ein Teil davon in eine Datei exportiert werden soll, deren Namen und Speicherort Sie bestimmen.

- Automatische Generierung durch das Skript.

  Wird das Buffer-Präfix einem load- oder select-Befehl vorangestellt, erstellt Qlik Sense automatisch eine QVD-Datei, die unter bestimmten Bedingungen beim erneuten Laden der Daten anstelle der ursprünglichen Datenquelle verwendet werden kann.

Die Art und Weise des Anlegens hat keinerlei Einfluss auf die Eigenschaften der QVD-Datei, d h. die Zugriffsgeschwindigkeit ist gleich.

Store

Dieser Skriptbefehl erstellt eine ausdrücklich benannte QVD-, CSV- oder txt-Datei.

Syntax:

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

Durch den Befehl werden Werte einer Datentabelle in die neue Datei exportiert. Wenn Felder aus mehreren Tabellen exportiert werden sollen, muss zuvor im Skript eine explizite Zusammenfügung erstellt werden, um die zu exportierende Datentabelle zu generieren.

Textwerte werden im UTF-8-Format in eine CSV-Datei exportiert. Es kann ein Trennzeichen festgelegt werden (siehe **LOAD**). Der Store-Befehl in einer CSV -Datei unterstützt keinen BIFF -Export.

Beispiele:

```
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';
```

Gehen Sie folgendermaßen vor:

1. Öffnen Sie die App *Erweitertes Skript-Tutorial*.
2. Klicken Sie auf den Skriptabschnitt *Product*.
3. Fügen Sie Folgendes zum Ende des Skripts hinzu:
   ```
   Store * from Product into [lib://AttachedFiles/ProductData.qvd](qvd);
   ```

   Ihr Skript sollte folgendermaßen aussehen:

   ```
   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. Klicken Sie auf **Daten laden**.
   Die *Product.qvd*-Datei sollte nun in der Dateiliste aufgeführt werden.

   Diese Datendatei stammt aus dem **Crosstable**-Skript und umfasst drei Spalten – eine Spalte pro Kategorie (Product, Month, Sales). Diese Datendatei kann nun verwendet werden, um gesamten *Product* -Skript-Abschnitt zu ersetzen.

6.2 Daten aus QVD-Dateien einlesen

Es gibt verschiedene Möglichkeiten, QVD-Dateien mithilfe von Qlik Sense zu lesen oder darauf zuzugreifen:

- Laden einer QVD-Datei als Datenquelle. QVD-Dateien können mit einem load-Befehl im Qlik Sense-Skript genau wie andere Textdateitypen (csv, fix, dif, biff usw.) referenziert werden.

  Beispiele:

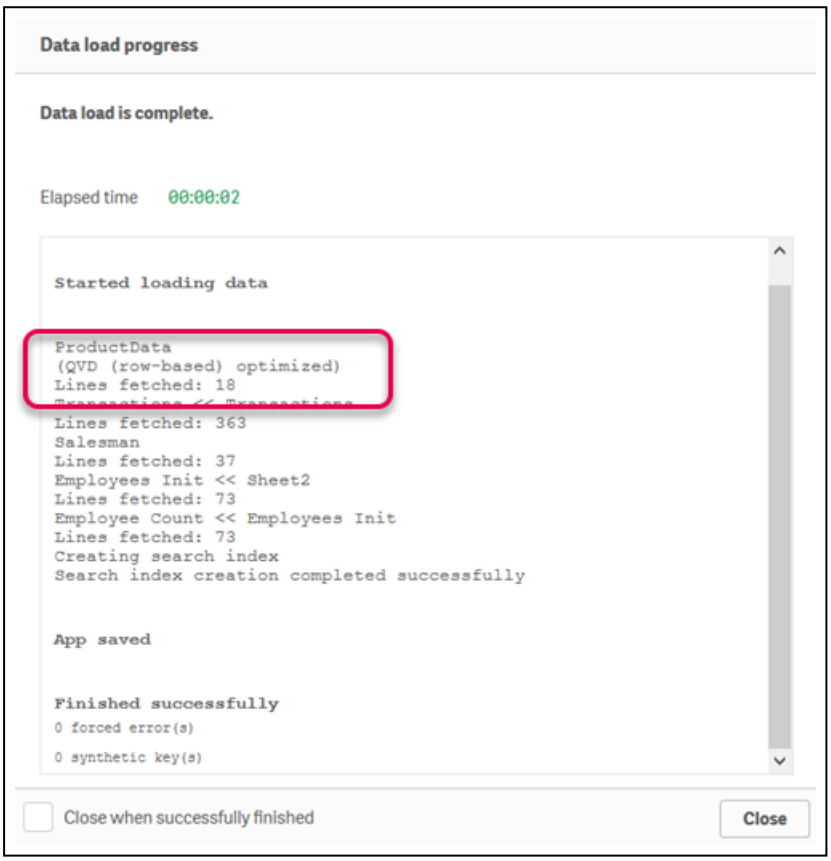
  ```
  LOAD * from 'lib://FolderConnection/xyz.qvd' (qvd);
  LOAD fieldname1, fieldname2 from [lib://FolderConnection/xyz.qvd] (qvd);
  LOAD fieldname1 as newfieldname1, fieldname2 as newfieldname2 from
  [lib://AttachedFiles/xyz.qvd](qvd);
  ```

- Automatisches Laden von gepufferten QVD-Dateien. Wenn Sie den buffer-Zusatz vor einem load- oder select-Befehl verwenden, benötigen Sie keinen separaten Befehl zum Einlesen der QVD-Datei. Qlik Sense prüft automatisch, welche Daten aus der QVD-Datei und welche aus dem zugrunde liegenden originalen LOAD oder SELECT -Befehl gelesen werden können.
- Zugriff auf QVD-Dateien über das Skript. Eine Reihe neuer Skriptfunktionen (ihre Namen beginnen alle mit QVD) helfen Ihnen, die Informationen im XML-Header der QVD-Datei auszulesen.

Gehen Sie folgendermaßen vor:

1. Kommentieren Sie das ganze Skript im Skriptabschnitt *Product* aus.
2. Geben Sie folgendes Skript ein:
   ```
   Load * from [lib://AttachedFiles/ProductData.qvd](qvd);
   ```
3. Klicken Sie auf **Daten laden**.
   Die Daten werden aus der QVD-Datei geladen.

   Fenster für Datenladefortschritt



Informationen zur Verwendung von QVD-Dateien für inkrementelles Laden finden Sie in folgendem Blog-Eintrag in Qlik Community: Overview of Qlik Incremental Loading *(Überblick über inkrementelles Laden in Qlik)*

Buffer

QVD-Dateien können mit Hilfe des Buffer-Zusatzes im Skript automatisch generiert werden. Dieser Zusatz kann zusammen mit den meisten LOAD- und SELECT-Befehlen verwendet werden. Er dient dazu, die eingelesenen Daten in einer QVD-Datei zu cachen bzw. zu buffern.

Syntax:

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

Ist keine Option definiert, wird die bei der ersten Skriptausführung generierte QVD-Datei auf unbestimmte Zeit weiterverwendet.

Beispiel:

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

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

Amount ist eine Zahl, die den Zeitraum definiert. Decimals dürfen verwendet werden. Ist keine Einheit angegeben, nimmt das Programm Tage an.

Die Option stale after ist für Datenbanken vorgesehen, deren Originaldaten keinen Zeitstempel enthalten. Eine stale after-Bedingung gibt einfach einen Zeitraum für das Erstellungsdatum des QVD-Buffers an. Danach ist er nicht mehr gültig. Vor dem Zeitrahmen wird der QVD-Buffer als Quelle für Daten verwendet, und danach gilt die ursprüngliche Datenquelle. In diesem Fall wird eine aktualisierte Version der QVD-Datei gespeichert, und die Zeitspanne beginnt von Neuem.

Beispiel:

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

Incremental

Durch die Option incremental wird nur ein Teil der Daten aus der Originaldatenquelle gelesen. Die bisherige Größe der Datei ist im XML-Header der QVD-Datei gespeichert. Dies ist insbesondere für log-Dateien nützlich. Alle bereits in der QVD-Datei vorhandenen Datensätze werden von dort eingelesen, alle neuen Datensätze aus der Originaldatenquelle. Am Ende dieses Vorgangs wird eine aktualisierte Version in der QVD-Datei gespeichert.

Beachten Sie, dass die Option incremental nur mit LOAD-Anweisungen und Textdateien verwendet werden kann und dass incremental load nicht verwendet werden kann, wenn alte Daten geändert oder gelöscht wurden.

Beispiel:

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

Normalerweise löscht das Programm automatisch QVD-Buffer, auf die während der Ausführung des Skripts nicht mehr zugegriffen wird. Auch beim Löschen der App werden die QVD-Buffer automatisch mitgelöscht. Die Store-Anweisung muss verwendet werden, wenn Sie den Inhalt des Buffers als QVD oder CSV-Datei behalten möchten.

Gehen Sie folgendermaßen vor:

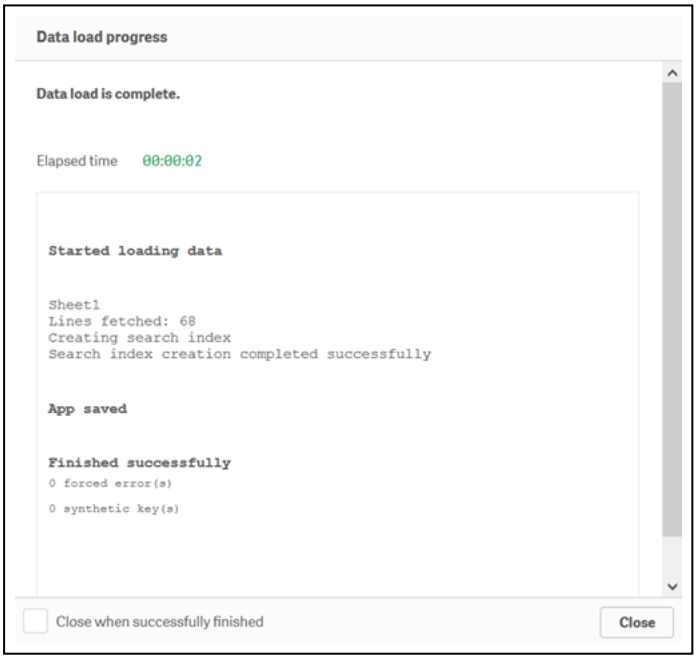
1. Erstellen Sie eine neue App und geben Sie ihm einen Namen.
2. Fügen Sie einen neuen Skriptabschnitt im **Dateneditor** hinzu.
3. Klicken Sie im rechten Menü unter **AttachedFiles** auf **Daten auswählen**.
4. Laden Sie *Cutlery.xlsx* hoch und wählen Sie die Datei aus.
5. Klicken Sie im Fenster **Daten auswählen aus** auf **Skript einfügen**.
6. Kommentieren Sie die Felder in der load-Anweisung aus und ändern Sie die load-Anweisung wie folgt:
   `Buffer LOAD *`

   Ihr Skript sollte folgendermaßen aussehen:

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

7. Klicken Sie auf **Daten laden**.
   Wenn Sie Daten zum ersten Mal laden, werden sie aus *Cutlery.xlsx* geladen.

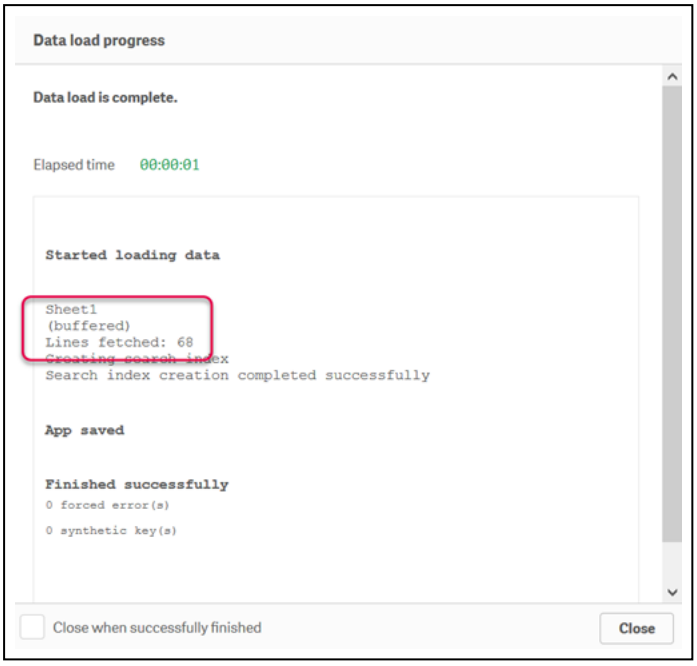
   Fenster für Datenladefortschritt

   Die Buffer-Anweisung erstellt auch eine QVD-Datei und speichert sie in Qlik Sense. In einer Qlik Sense Enterprise on Windows-Bereitstellung wird sie in einem Verzeichnis auf dem Qlik Sense-Server gespeichert.

8. Klicken Sie erneut auf **Daten laden**.
9. Dieses Mal werden die Daten aus der QVD-Datei geladen, die von der Buffer-Anweisung beim ersten Laden der Daten erstellt wurde.

Fenster für Datenladefortschritt



6.3 Vielen Dank!

Sie haben dieses Tutorial jetzt abgeschlossen und hoffentlich grundlegende Kenntnisse zur Verwendung von Skripts in Qlik Sense erworben. Nähere Informationen zu weiteren Schulungen erhalten Sie auf unserer Webseite.