Tutorial - Scripting for Beginners

Qlik Sense®
November 2026
Copyright © 1993-2026 QlikTech International AB. All rights reserved.

© 2026 QlikTech International AB. All rights reserved. All company and/or product names may be trade names, trademarks and/or registered trademarks of the respective owners with which they are associated.

Contents

1 Welcome to this tutorial! 4
1.1 What you will learn 4
1.2 Who should take this course 4
1.3 Package contents 4
1.4 Lessons in this tutorial 5
1.5 Further reading and resources 5
2 Scripting in the data load editor 6
2.1 Data load editor 6
2.2 Script editor 7
Accessing syntax help for commands and functions 7
Commenting in the script 8
Indenting code 8
Inserting a prepared test script 8
3 LOAD and SELECT statements 9
4 Selecting and loading data 10
5 Renaming fields 19
6 Reducing data 22
7 Transforming data 28
7.1 Resident LOAD 28
7.2 Preceding LOAD 32
8 Concatenation 34
8.1 Automatic concatenation 34
8.2 Forced concatenation 38
8.3 Preventing concatenation 39
9 Circular references 41
9.1 Resolving circular references 42
10 Synthetic keys 44
10.1 Resolving synthetic keys 46
11 Using data in an application 48
11.1 Adding a chart 48
11.2 Adding dimensions and measures 49
Creating and adding dimensions 49
Creating and adding measures 50
11.3 Thank you! 54

1 Welcome to this tutorial!

Welcome to this tutorial, which will introduce you to basic scripting in Qlik Sense.

Before you can create visualizations in your application in Qlik Sense, you have to load your data. Knowing how to use load scripts allows you to prepare and manipulate your data when you load it into your application.

You can load data using the data manager or the data load editor. You use the data load editor when you want to create, edit and run a data load script.

1.1 What you will learn

After completing this tutorial, you should be comfortable with loading data using scripts, editing scripts, and transforming data.

1.2 Who should take this course

You should be familiar with the basics of Qlik Sense. That is, you have created applications and visualizations.

You require access to the data load editor and should be allowed to load data in Qlik Sense Enterprise on Windows.

1.3 Package contents

The zip package that you downloaded contains the following data files that you need to complete the tutorial:

- *Customers.xlsx*
- *Dates.xlsx*
- *Region.txt*
- *Sales.xlsx*

Additionally, the package contains a copy of the *Scripting Tutorial* application. You can upload the application to your hub.

We recommend building the application yourself as described in the tutorial to maximize your learning. Additionally, you would have to upload and connect to your data files as described in the tutorial for the data loads to work.

However, if you run into problems, the application may help you troubleshoot. We have indicated which script segments are associated with each lesson.

1.4 Lessons in this tutorial

Depending on your experience with Qlik Sense, this tutorial should take 3-4 hours to complete. The topics are designed to be completed in sequence. However, you can step away and return at any time. There are, mercifully, no tests.

- Introduction to data loading
- LOAD and SELECT statements
- Selecting and loading data
- Renaming fields
- Reducing data
- Transforming data
- Concatenation
- Circular references
- Synthetic keys
- Using data in an application

1.5 Further reading and resources

- ≤ Qlik offers a wide variety of resources when you want to learn more.
- Qlik online help is available.
- Training, including free online courses, is available on ≤ Qlik Learning.
- Discussion forums, blogs, and more can be found in ≤ Qlik Community.

2 Scripting in the data load editor

Applications use a data load script, which is managed in the Data load editor, to connect to and retrieve data from various data sources. A data source can be a data file, for example an Excel file or a .csv file. A data source can also be a database, for example a Google BigQuery or Salesforce database.

You can also load data using the Data manager, but when you want to create, edit and run a data load script you use the data load editor.

In the script, the fields and tables to load are specified. Scripting is often used to specify what data to load from your data sources. You can also manipulate the data structure by using script statements.

During the data load, Qlik Sense identifies common fields from different tables (key fields) to associate the data. The resulting data structure of the data in the application can be monitored in the data model viewer. Changes to the data structure can be achieved by renaming fields to obtain different associations between tables.

After the data has been loaded into an application, it is stored in the application.

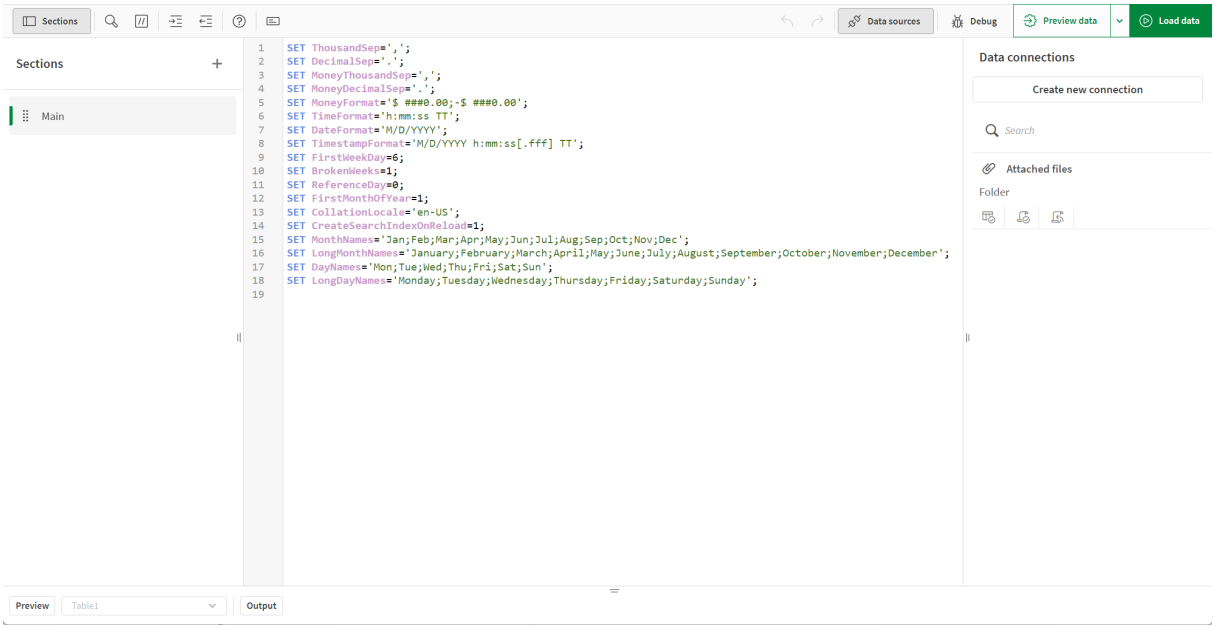
2.1 Data load editor

You can create scripts to load data in Data load editor. The editor is available from the drop-down menu in the application.

When you open the data load editor, the script editor appears in the middle of your screen. Script sections are shown as tabs in the left menu. The application automatically creates the **Main** section. Data connections are shown in the right menu.

The script must be written using the Qlik Sense script syntax. Qlik Sense syntax keywords are highlighted in blue.

Data load editor



2.2 Script editor

There are a number of functions available in the editor to assist you in developing the load script.

Accessing syntax help for commands and functions

There are several ways to access syntax help for a Qlik Sense syntax keyword.

Accessing the help portal

You can access detailed help in the Qlik helpsite in two different ways.

- Click ⓘ in the toolbar to enter syntax help mode. In syntax help mode you can click on a syntax keyword (marked in blue and underlined) to access syntax help.
- Place the cursor inside or at the end of the keyword and press Ctrl+H.
- *You cannot edit the script in syntax help mode.*

Using the autocomplete function

If you start to type a Qlik Sense script keyword, you get an autocomplete list of matching keywords to select from. The list is narrowed down as you continue to type, and you can select from templates with suggested syntax and parameters. A tooltip displays the syntax of the function, including parameters and additional statements, as well as a link to the help portal description of the statement or function.

You can also use the keyboard shortcut Ctrl+Space to show the keyword list, and Ctrl+Shift+Space to show a tooltip.

Commenting in the script

You can insert comments in the script code, or deactivate parts of the script code by using comment marks. All text on a line that follows to the right of // (two forward slashes) will be considered a comment and will not be executed when the script is run.

The data load editor toolbar contains a shortcut for commenting or uncommenting code. Click //, or press Ctrl + K to comment or uncomment code.

Indenting code

You can indent the code to increase readability. Click to indent the text (increase indentation) or, click to outdent the text (decrease indentation).

Inserting a prepared test script

You can insert a prepared test script that will load a set of inline data fields. You can use this to quickly create a data set for test purposes. Press Ctrl + 00 to insert the test script.

3 LOAD and SELECT statements

You can load data into applications using the LOAD and SELECT statements. Each of these statements generates an internal table. LOAD is used to load data from files, while SELECT is used to load data from databases.

In this tutorial, you will be using data from files, so you will be using LOAD statements.

You can also use a preceding LOAD to be able to manipulate the content of the data loaded. For example, renaming fields has to be done in a LOAD statement, whereas the SELECT statement does not permit any changes to field names.

The following rules apply when loading data into applications:

- Applications do not differentiate between tables generated by a LOAD or a SELECT statement. This means that if several tables are loaded, it does not matter whether the tables are loaded by LOAD or SELECT statements or by a mix of the two.
- The order of the fields in the statement or in the original table in the database is unimportant to the application logic.
- Field names are case sensitive and are used to establish associations among data tables. Due to this, at times it is necessary to rename fields in the load script to achieve a desired data model.

4 Selecting and loading data

Loading data from files, such as Microsoft Excel or any other supported file format, is easily done by using the data selection dialog in the data load editor.

Do the following:

1. Open Qlik Sense.
2. Create a new application.
3. Name the application *Scripting Tutorial*, and then click **Create**.
4. Open the application.

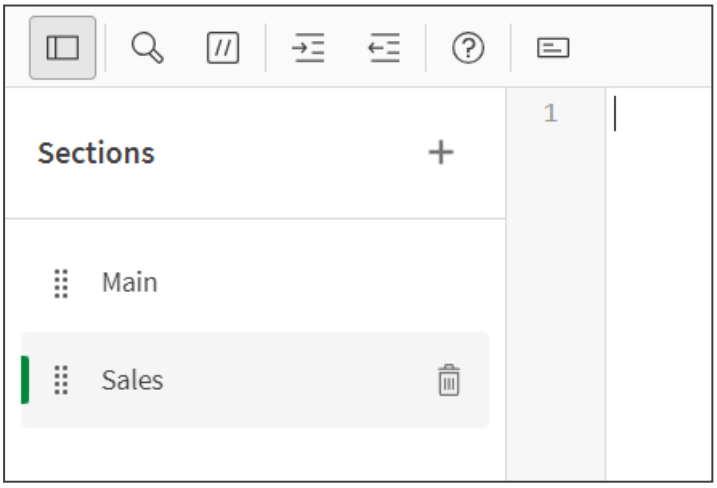
   > *Before you load data into your application for the first time, there is an option to use **Add data** to easily load data from files. However, in this tutorial we want to see the script, so we will use the data load editor.*

5. Open Data load editor from the drop-down menu in the top toolbar.
6. Click + in the left menu to add a new script section below the section named *Main*.

   Using more than one section makes it easy to keep your script organized. The script section will execute in order when you load data.

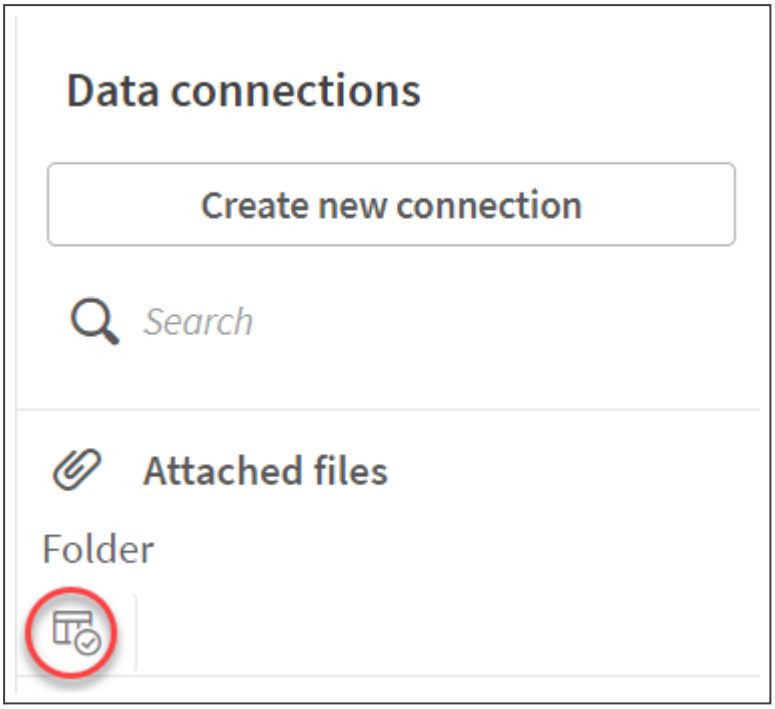
7. Give the section a name by typing *Sales*.

   New Sales tab in Scripting Tutorial application

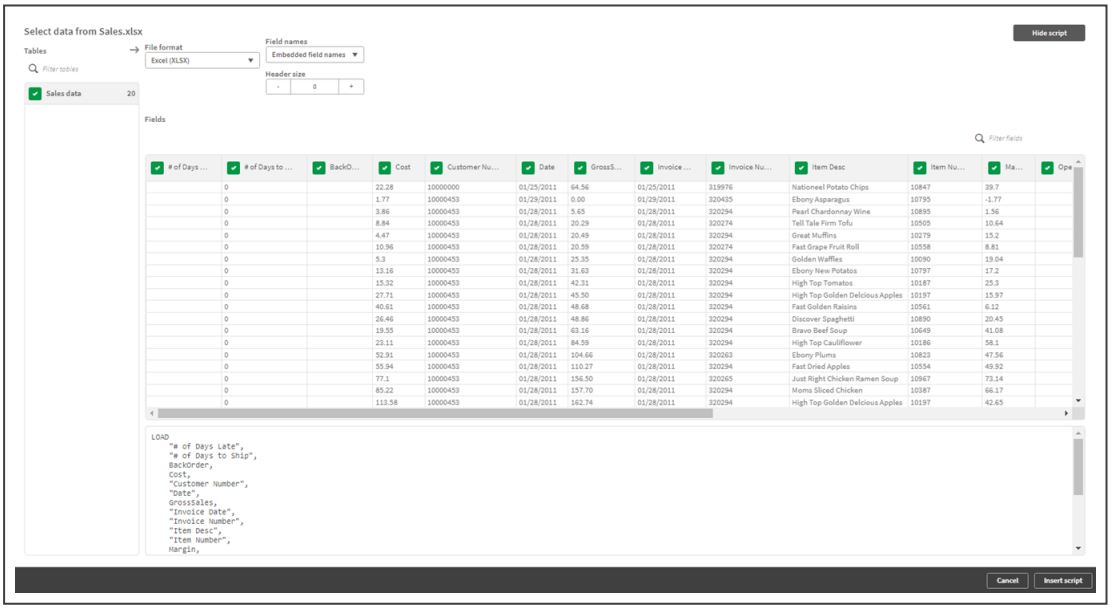
8. Under **AttachedFiles** in the right menu, click **Select data**.

Select data window



9. Upload and then select *Sales.xlsx*. A data preview window opens.

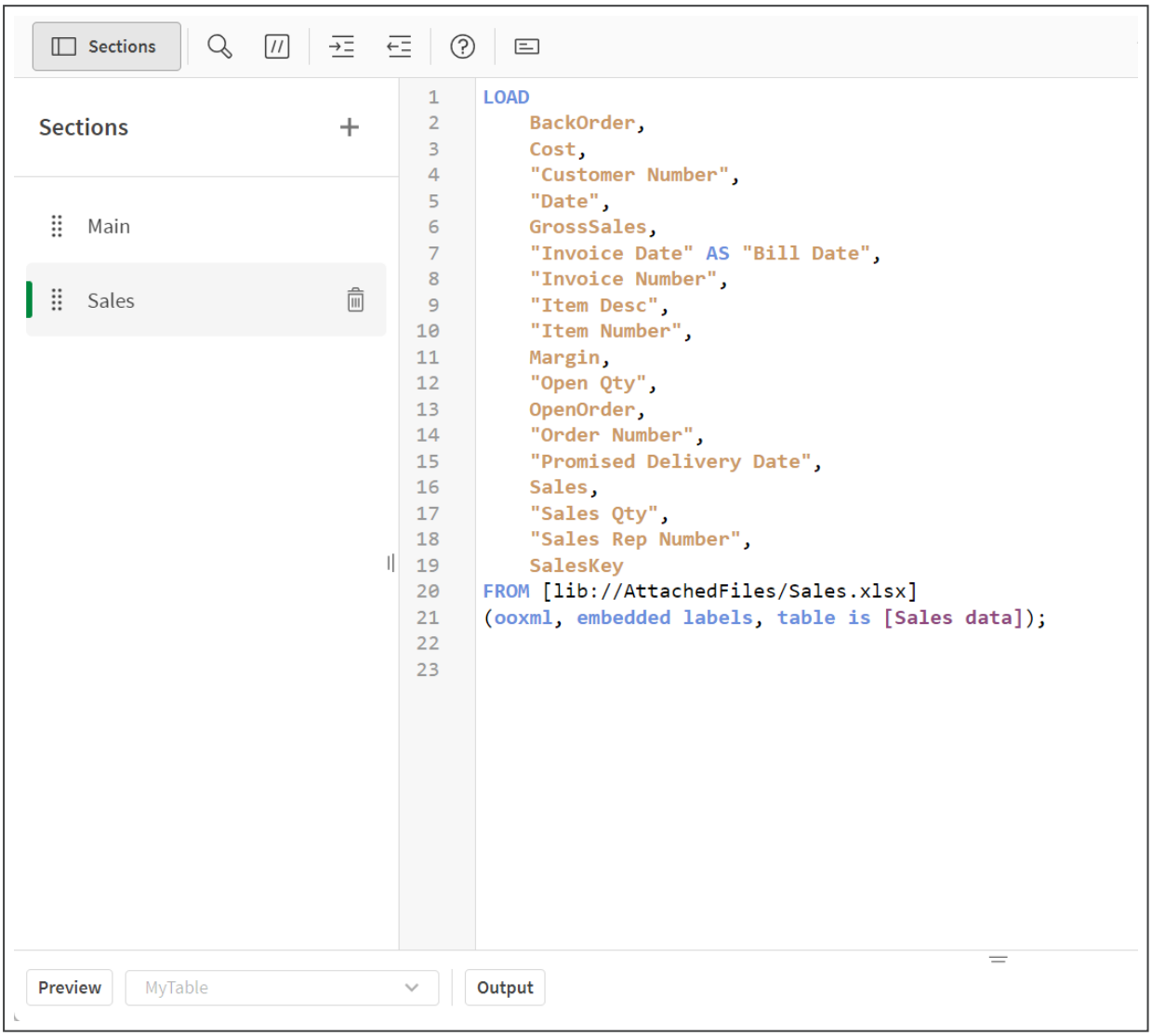
   Data preview window for Sales data file



10. Deselect the fields *# of Days Late* and *# of Days to Ship*. You might need to click on the field headings to see the complete field names.
11. Search for *date* in the **Filter fields** search field.
12. Click on the heading *Invoice Date* and type *Bill Date* to rename the field.

13. Click **Insert script**. The load script is inserted into the *Sales* section of the script editor. Note that the application puts double quotes around field names that contain a space.
    Your script should look like this:

    Load script in Sales tab

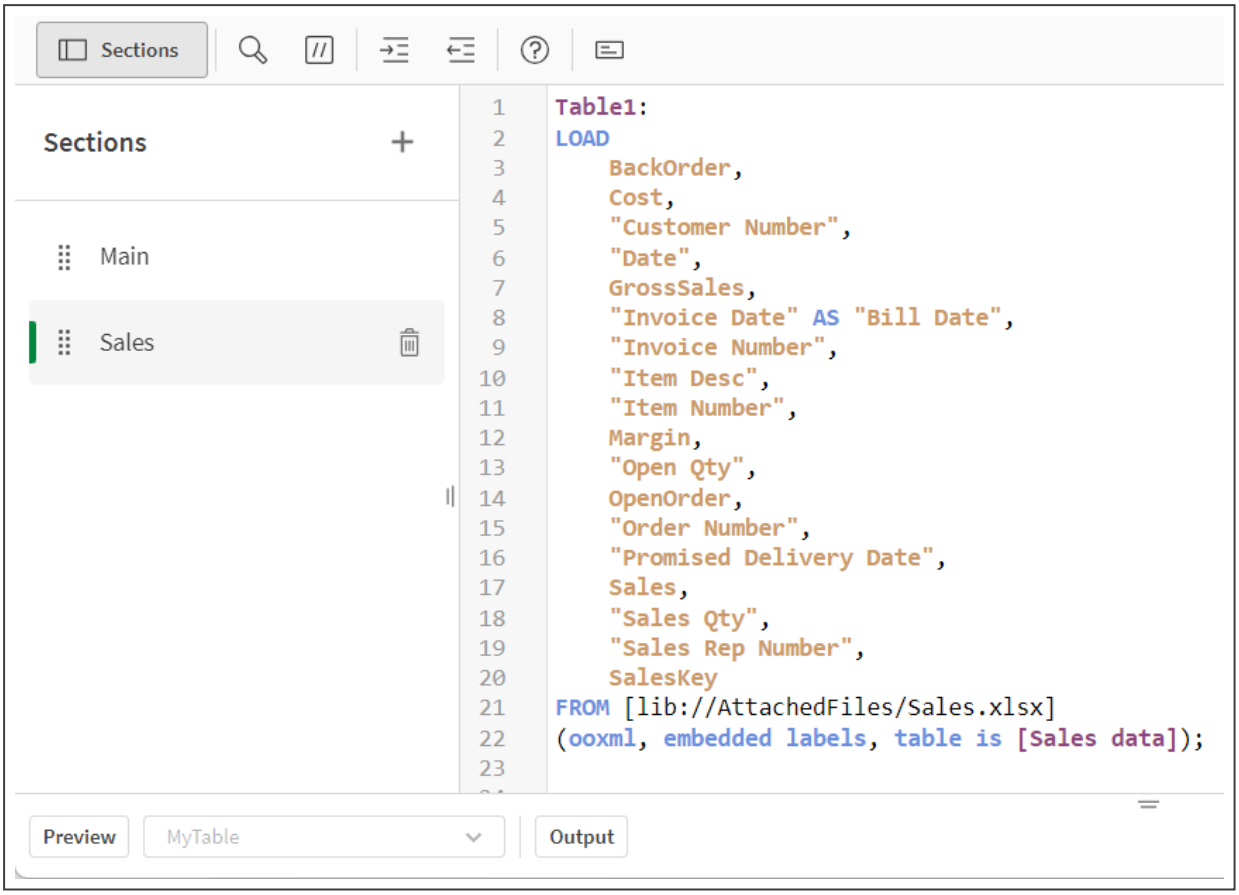


14. Add the following row above the *LOAD* statement to name the table *Table1*:
    `Table1:`

15. Now adjust the script to ensure that the dates are interpreted correctly. Change the *Date* field to the following:
    `Date(`Date`,'MM/DD/YYYY') as "Date",`

    Your script should look like this:

Updated load scriptin Sales tab

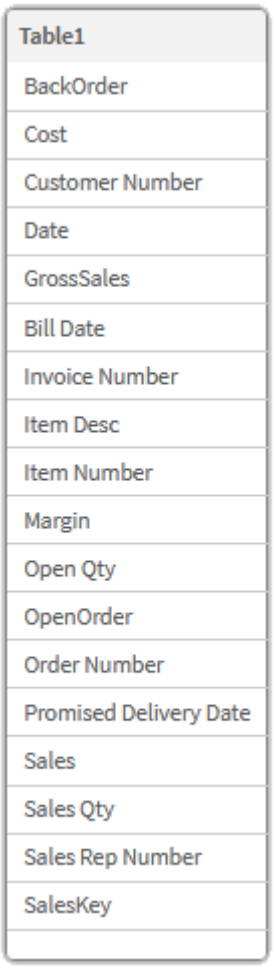


16. In the upper right corner, click **Load data**.

    This will load the data into the application. A script execution progress window is displayed. When it is finished you will see a summary of possible errors and synthetic keys even if there are none.

17. Click **Close**.
18. Open the data model viewer from the drop-down menu in the top toolbar. By clicking the data model viewer will open in a new tab.
19. Select and in the top menu to show the table view that is used in this tutorial. If your table is not displayed properly, you can remove the existing load script and build the script again.

Table view in data model viewer of Sales data



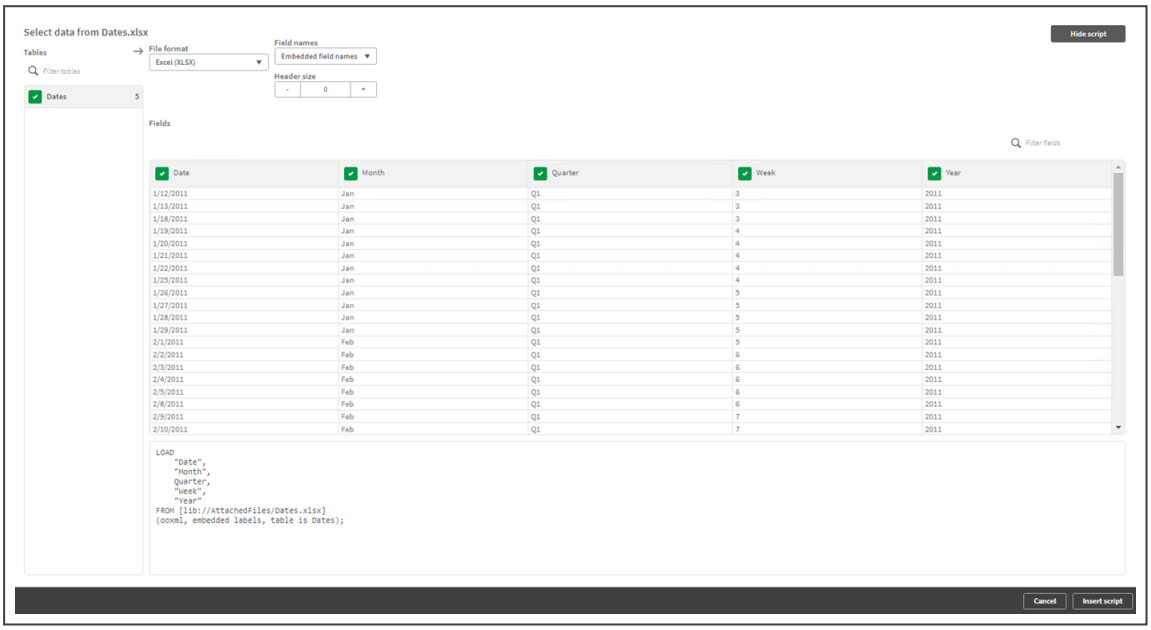
Now, let's load another table called *Dates*. After we load the table, the application will connect it with the *Sales* table on the *Date* field.

20. Open the **Data load editor**.
21. Click + to add a new script section.
22. Name the section *Dates*. If the new section *Dates* is not already placed below *Sales*, move the pointer over ⣿, and then drag the section down below the section *Sales* to rearrange the order.
23. Click on the top row of the script and click //.
    Make sure // is added into the script.
24. Add the following text after //:
    ```
    Loading data from Dates.xlsx
    ```
    The top line of your script should now look like this:
    ```
    // Loading data from Dates.xlsx
    ```
25. Under **AttachedFiles** in the right menu, click **Select data**.

> *Under **Field names**, make sure that **Embedded field names** is selected to include the names of the table fields when you load the data.*

26. Upload and then select *Dates.xlsx*. A data preview window opens.

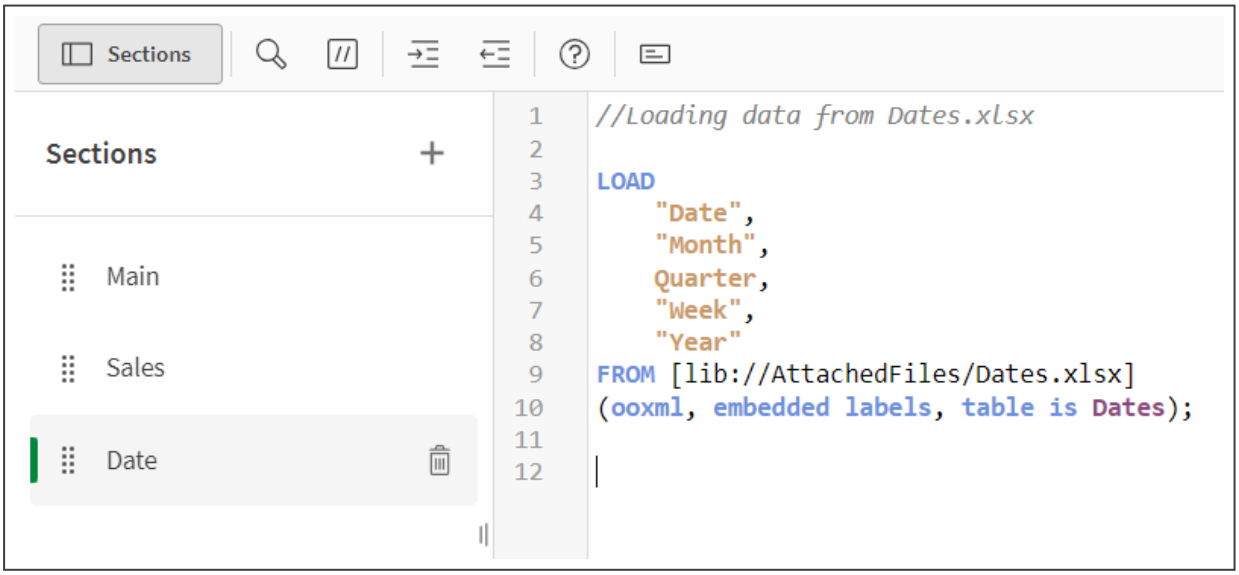
Data preview window for Dates data file



27. Click **Insert script**.

Your script should look like this:

Load script in Dates tab



28. Add the following on the row above the *LOAD* statement to name the table *Table2*:

```
Table2:
```

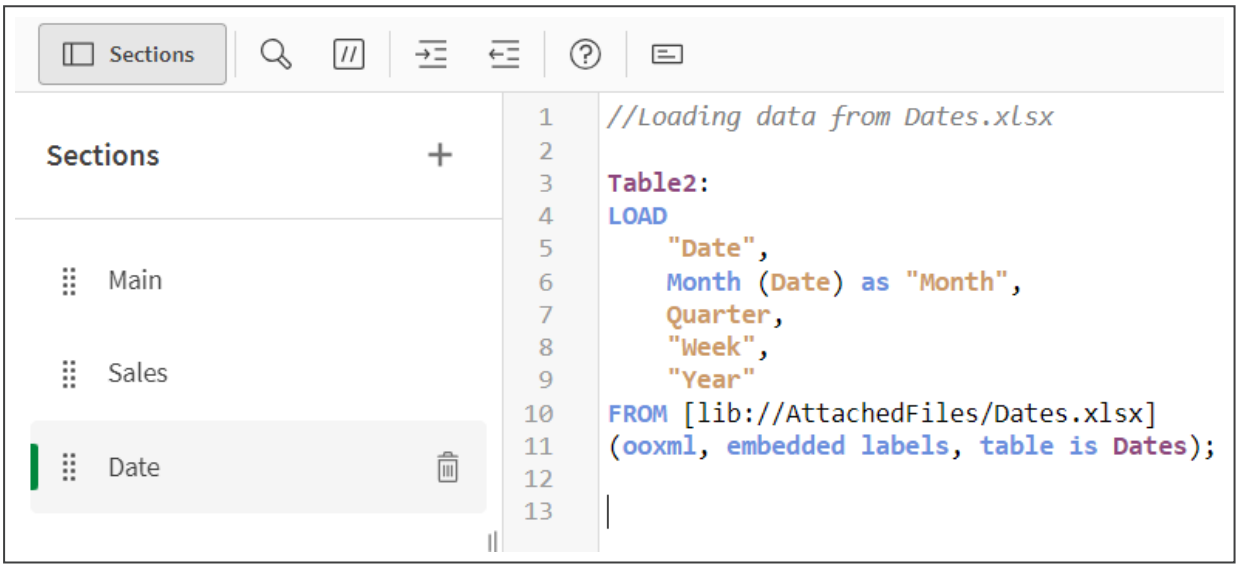
29. To ensure that the *Month* column in the file *Dates.xlsx* is interpreted correctly in the application, we need to apply the *Month* function to the *Date* field.

    Change the *Month* field to the following:

    ```
    Month (Date) as "Month",
    ```

    Your script should look like this:

    Updated load script in Dates tab



    Now you have created a script to load the selected data from the file *Dates.xlsx*. It is time to load the data into the application.

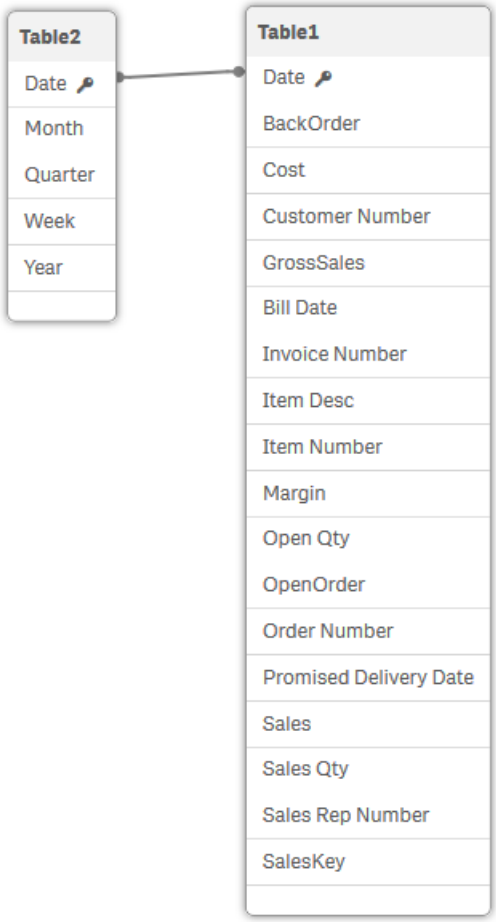
30. In the upper right corner, click **Load data**.

    When you click **Load data**, the data is loaded into the application and the script is saved.

31. When the script execution is finished, click **Close**.
32. Open the **Data model viewer**.

    Now you can see that a connection has been made between the two fields named *Date* in the two tables.

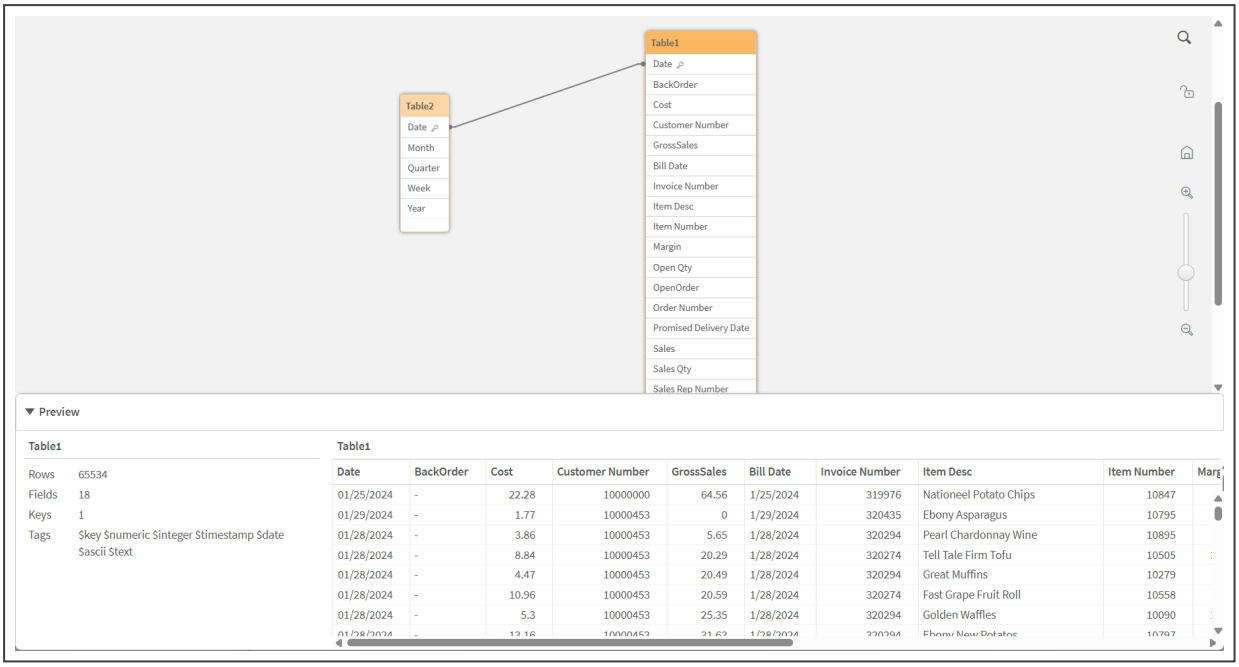
Table view in data model viewer



33. Click **Preview** in the bottom left corner. Click on the name of the table *Table2*.

    This will display information about the table. In the **Preview** field you can see that 628 rows of data have been loaded into the internal table *Table2*. If you instead click on a field in the table, you will see information about the field.

Table preview in data model viewer



The data is now available to use in visualizations in an application. We will show you how later in this tutorial.

5 Renaming fields

In the previous topic, we showed you how to rename fields in the data preview window. In the heading of the data preview window, you renamed *Invoice Date* to *Bill Date*. When you inserted the load script, you could see that the field would be renamed using the keyword *AS*.

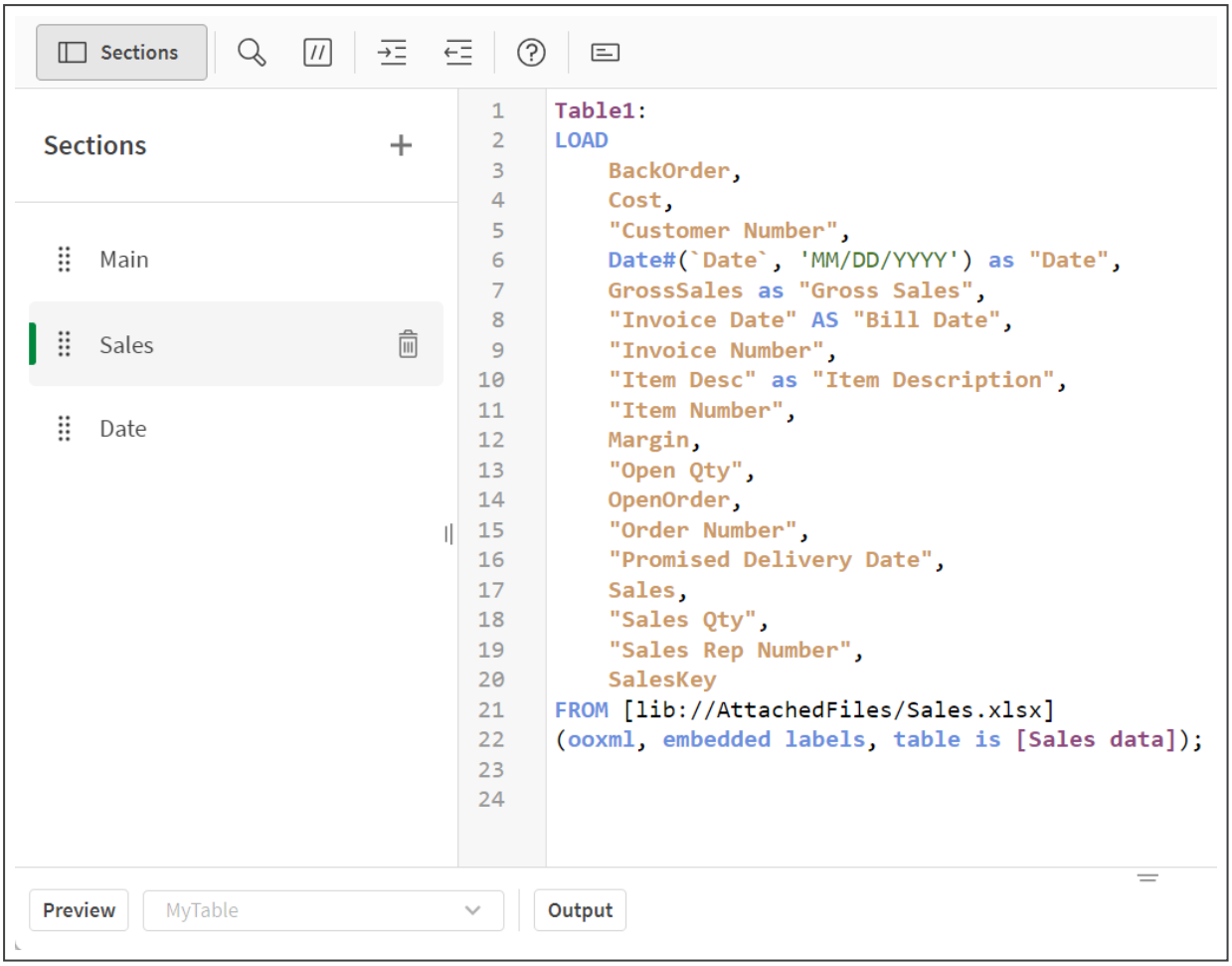
We can also perform this action directly in the script.

Do the following:

1. Open the **Data load editor** in the *Scripting Tutorial* application.
2. Click the *Sales* tab.
3. In the load script, make the following changes. Note that you have to include parentheses around fields that contain a space.
    i. Change *GrossSales,* to:
    ```
    GrossSales AS "Gross Sales",
    ```
    ii. Change *"Item Desc",* to:
    ```
    "Item Desc" AS "Item Description",
    ```

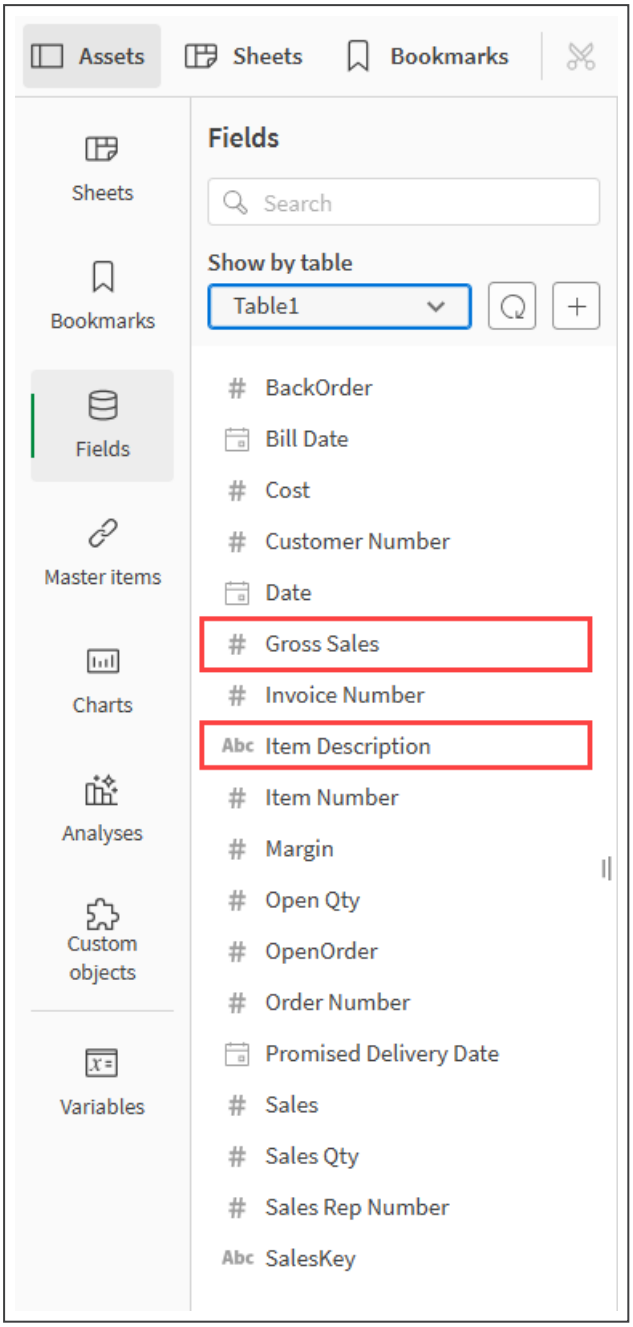
    Your script should look like this:

Load script window with script to rename fields



4. Click **Load data**. The data is loaded.
5. Open the **Data model viewer**. Confirm that the fields have been renamed.
6. You can also view all your fields in an application. Click the **Analysis/Sheet** tab in the top toolbar. The application opens in sheet view.
7. Click **Edit sheet**, and then click **Fields** in the assets panel. You can see the field names that you changed. You can use any of these fields in the visualizations that you create in your application.

Renamed fields in analysis view



6 Reducing data

The application provides several different ways to reduce the amount of data that you load into your application. You can, for example, filter data from files or from data connectors.

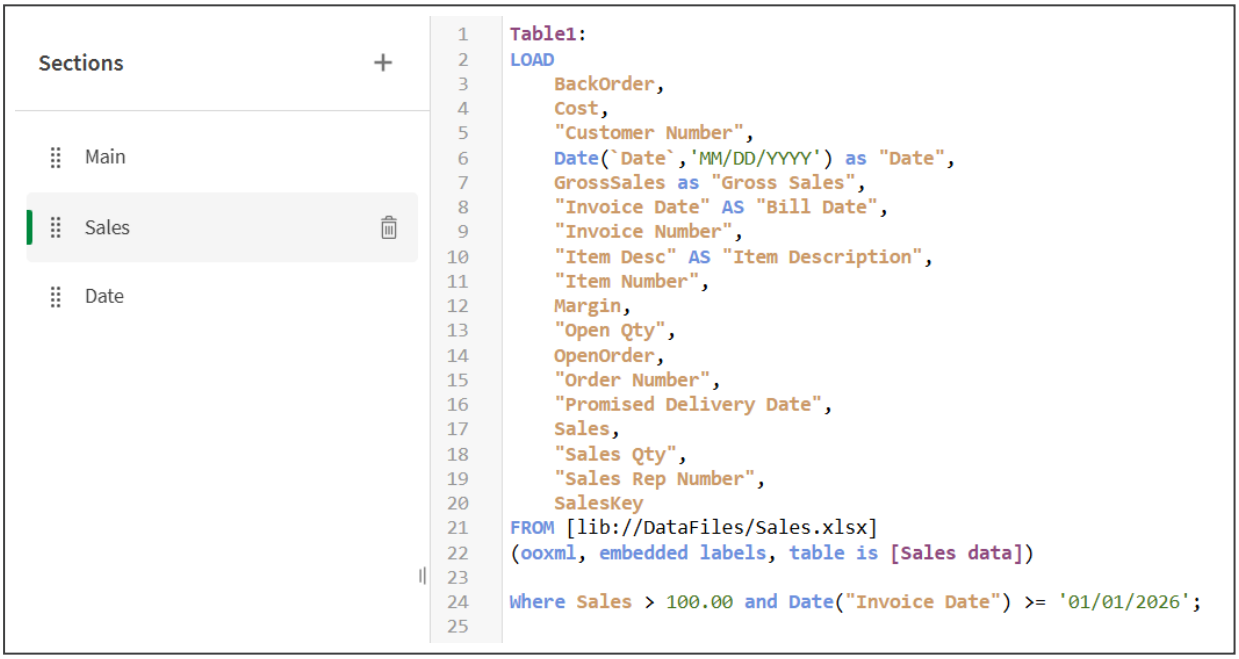
You can also reduce data directly in the load script.

Do the following:

1. Open the **Data load editor** in the *Scripting Tutorial* application.
2. Click **Load data**.

   Based on the load script that you have written so far, the application loads 65,534 lines from the *Sales.xlsx* data file into *Table1*. Note that *Sales data* is the name of the tab that contains your table in the original *Sales.xlsx* file.

   Data load progress window

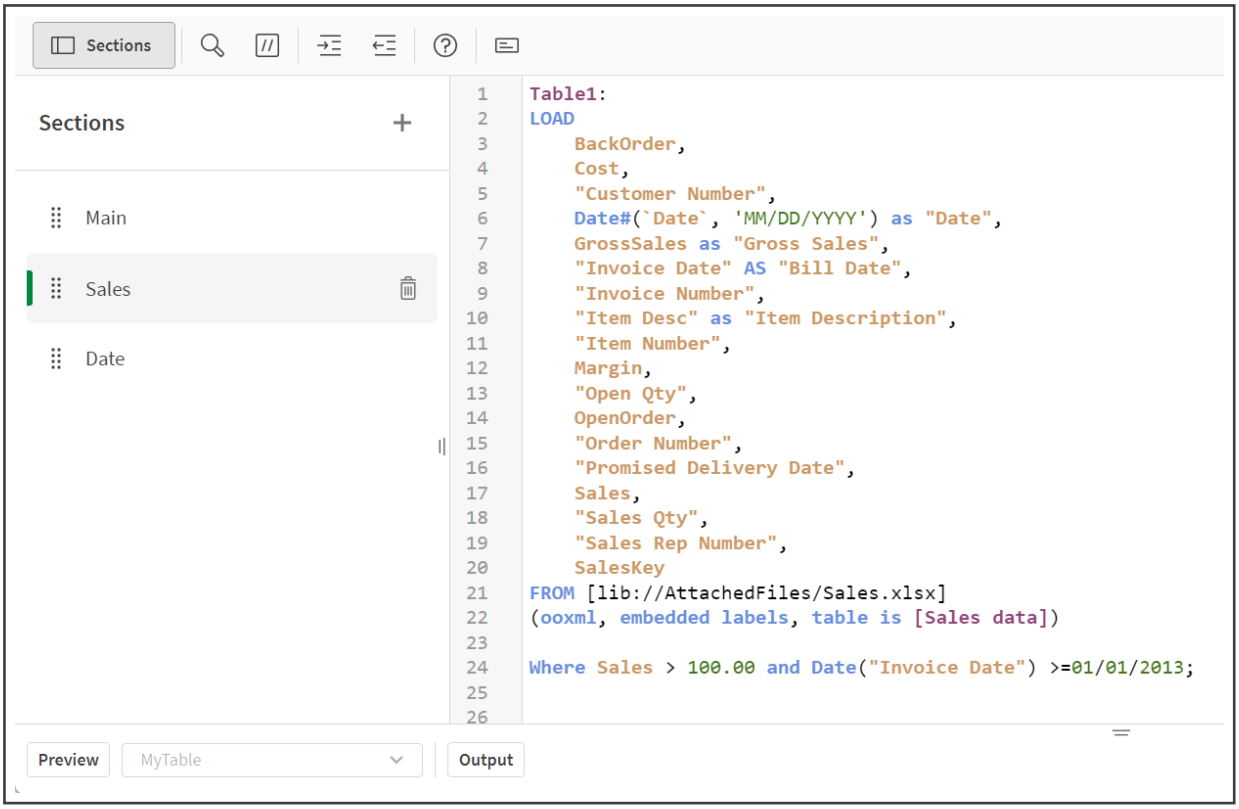


3. In the data load editor, click the *Sales* tab.
4. Delete the semicolon at the end of this line:
   `(ooxml, embedded labels, table is [Sales data]);`
5. Add the following line at the end of the load script:
   `Where Sales > 100.00 and Date("Invoice Date") >= '01/01/2026';`

   This tells the application to only load data where sales are greater than $100.00. It also uses the Date function to load data where the date is equal to or greater than January 1, 2026.

   Your script should look like this:

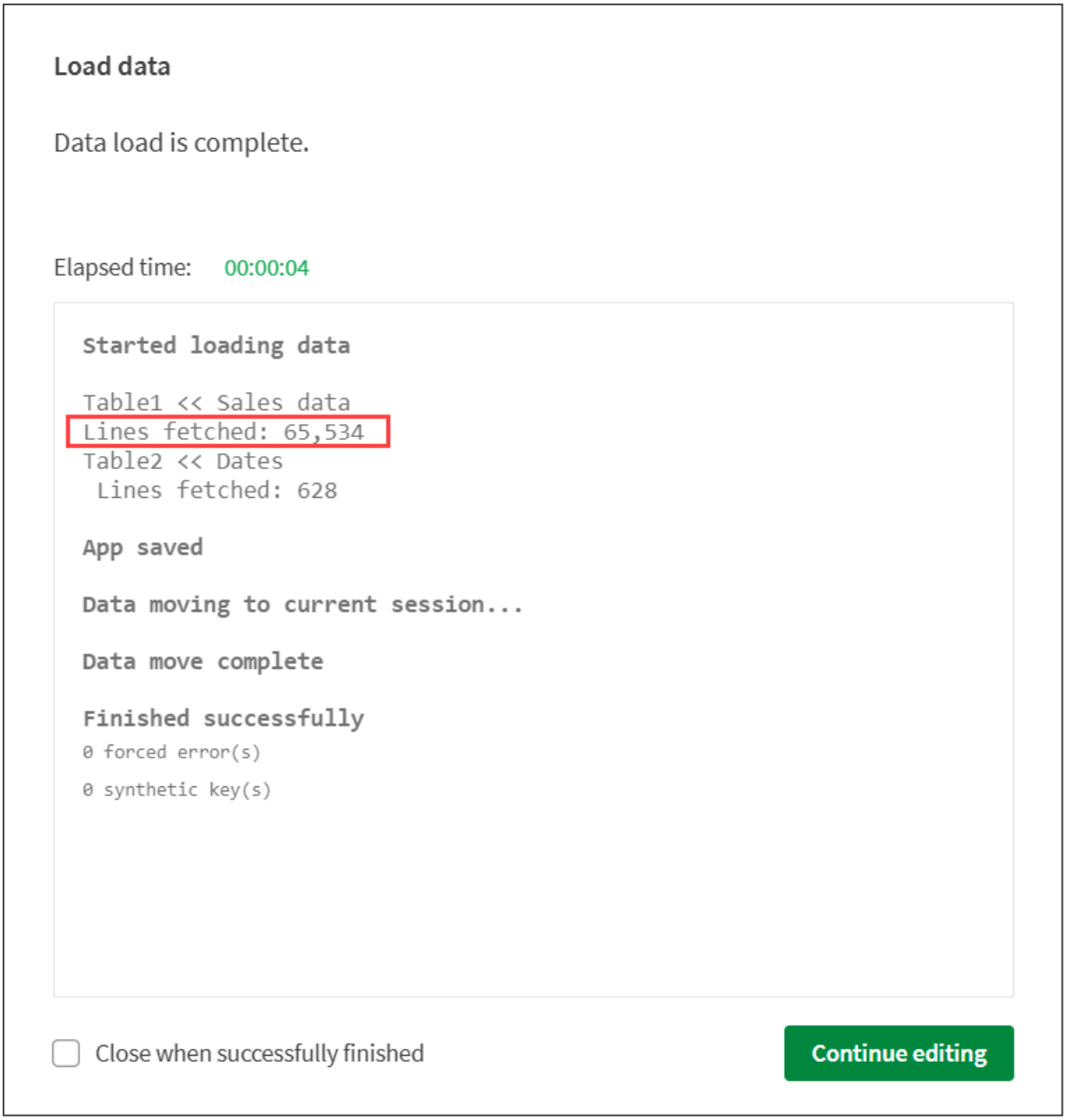
Load script window with script to reduce amount of data loaded



6. Click **Load data**.

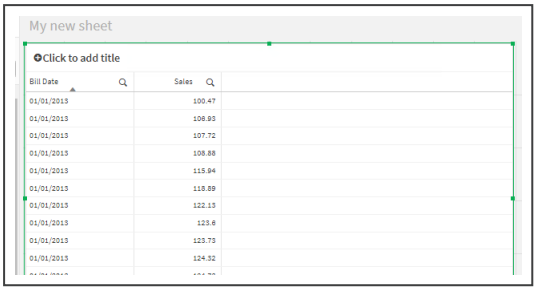
   Based on your updated load script, the application now loads fewer lines from the *Sales.xlsx* data file.

Data load progress window with reduced data load



7. If you then add the data to a straight table in your application, you can see that only the data that conforms to the conditions that you created was loaded.

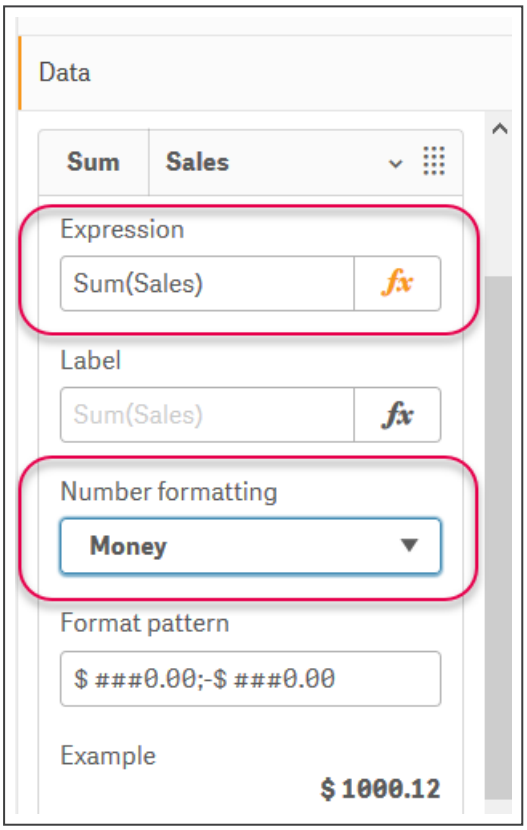
Table containing Bill Date and Sales fields



Note that we added the *Sales* field as a dimension. This is so that the *Sales* values are shown individually. If we had added *Sales* as a measure the values would have instead been aggregated per date.

Typically, you would add *Sales* as a measure. With measures you have the option to display values as currency amounts (for example, dollars) by applying number formatting to the column.

Number formatting applied to Sales measure

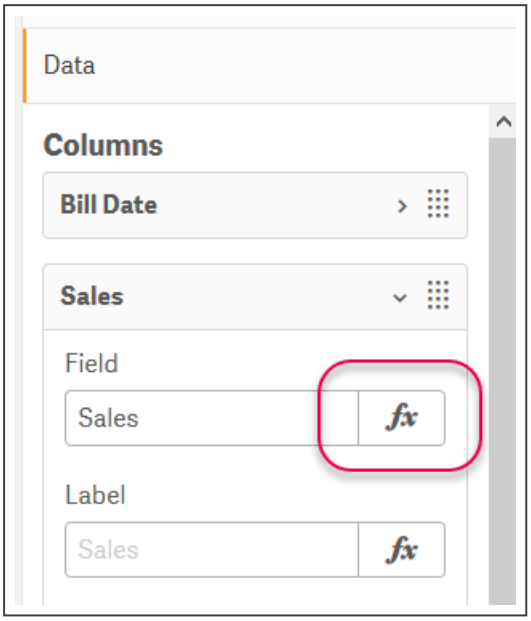


However, as we are using *Sales* as a dimension, we need a different approach. In this case, we will use a chart expression. Even though we are not discussing chart functions in detail in this tutorial, this is a good opportunity for a quick example.

When we use *Sales* as a dimension, the values are displayed as numeric, as shown in the table above.

To fix this, you can open the chart expression editor *Sales* field by clicking **fx**, and then using the *Money* function.

Opening the expression editor



Enter the following in the expression editor.

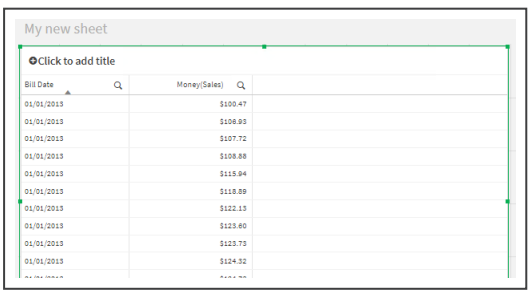
```
=Money( Sales )
```

Expression editor



The values in the field will now display as monetary units. Those units (in this case, dollars) are specified in the **Main** section of the load script.

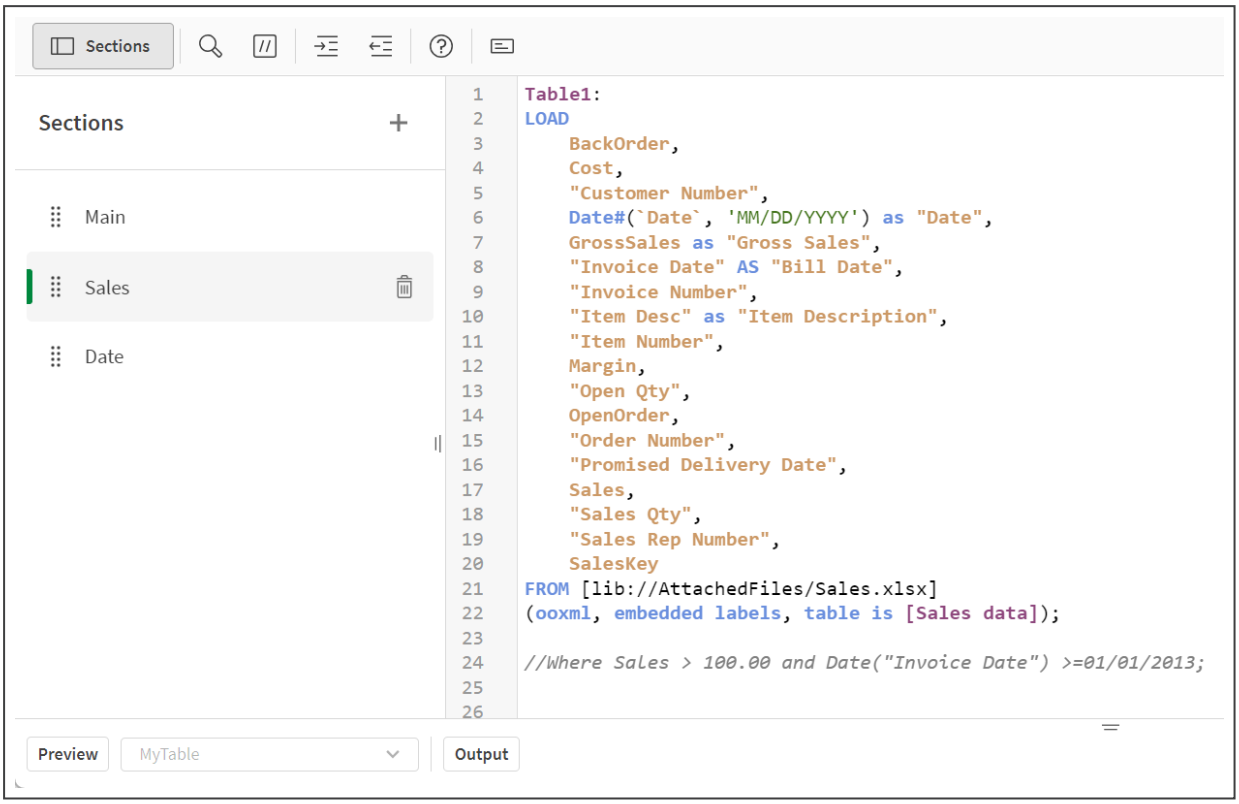
Table containing Bill Date and Sales fields. Sales now shown in dollars



8. Now that we have completed this example, comment out the Where statement. Don't forget to add a semicolon to the end of your first LOAD statement.

Your script should look like this:

Load script window with Where statement commented out



7 Transforming data

You can transform and manipulate data using many different techniques in Data load editor.

One of the advantages to data manipulation is that you can choose to load only a subset of the data from a file, such as a few chosen columns from a table, to make the data handling more efficient. You can also load the data more than once to split up the raw data into several new logical tables. It is also possible to load data from more than one source and merge it into one table in the application.

In this topic, you will perform some basic data transformation using a Resident load, and then a Preceding load.

7.1 Resident LOAD

You can use the Resident source qualifier in a LOAD statement to load data from a previously loaded table. This is also useful when you want to perform calculations on data loaded with a SELECT statement where you do not have the option to use scripting functions, such as date or numeric value handling.

In this example, you will create a new table called *Sales_Buckets* and then load the data from *Table1* using a resident load. In the *Sales_Buckets* table, you will create a variable called *quantity_threshold*, and then use a Where statement to only load data that meets that threshold.

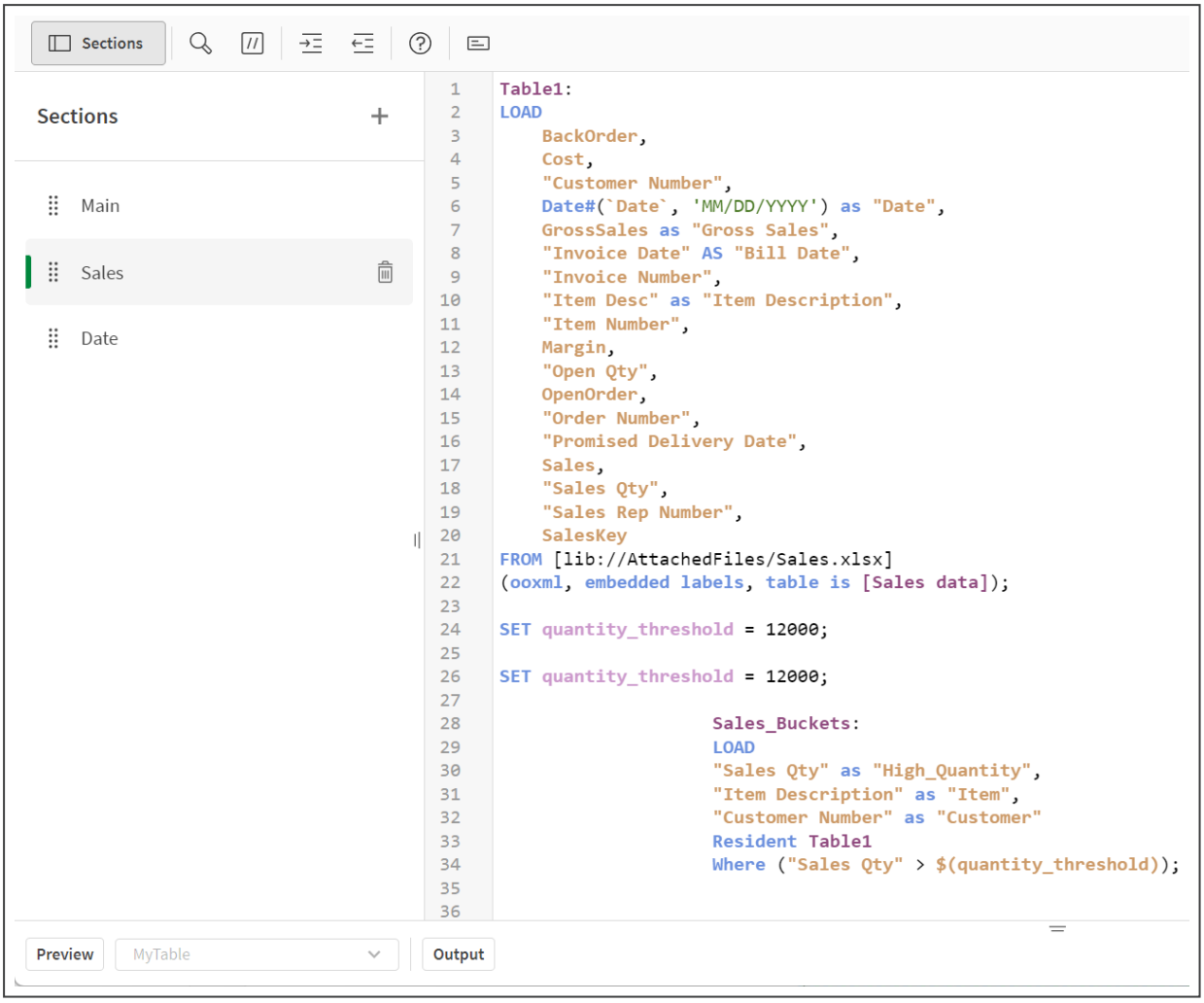
1. Open the **Data load editor** in the *Scripting Tutorial* application.
2. Click the *Sales* tab.
3. Add the following to the end of your script:

```
SET quantity_threshold = 12000;

Sales_Buckets:
LOAD
"Sales Qty" as "High_Quantity",
"Item Description" as "Item",
"Customer Number" as "Customer"
Resident Table1
Where ("Sales Qty" > $(quantity_threshold));
```

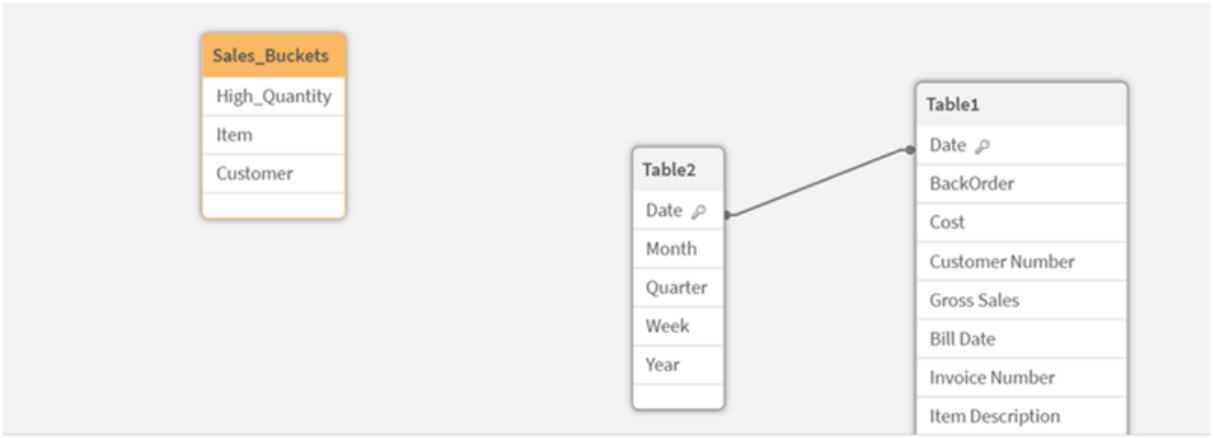
Your script should look like this:

Load script window with script to create new table called Sales_Buckets



4. Click **Load data**.
5. Open the **Data model viewer**. You can see that you created a new table called *Sales_Buckets* with the data loaded according to the fields that you specified, and the threshold that you set.

Sales_Buckets table in data model viewer



▼ Preview

Sales_Buckets

Rows	11
Fields	3
Tags	$numeric $integer $ascii $text

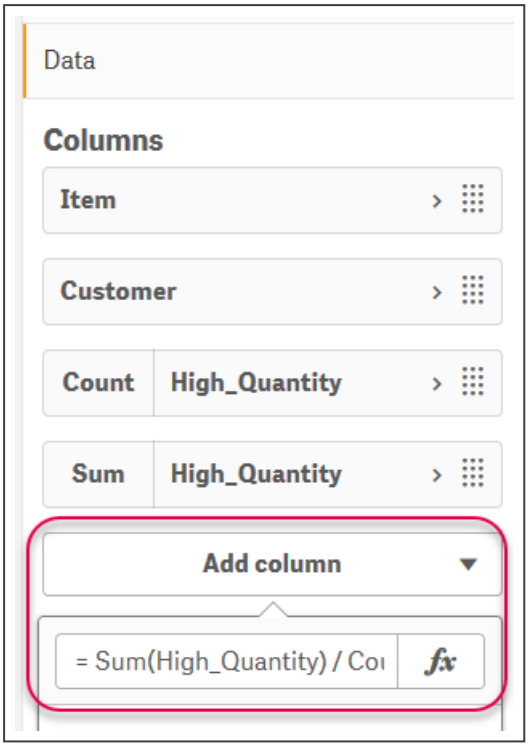
Sales_Buckets

High_Quantity	Item	Customer
16000	High Top Dried Mushrooms	10025737
12500	Blue Label Canned Peas	10017036
16000	High Top Dried Mushrooms	10025052
13600	High Top Dried Mushrooms	10006919
16000	High Top Dried Mushrooms	10006919
13600	High Top Dried Mushrooms	10025737
13600	High Top Dried Mushrooms	10025052
13600	High Top Dried Mushrooms	10025052

6. Add the data to a table in your application. Add *Item* and *Customer* as dimensions. Add *High_Quantity* as a measure aggregated on *Count*, and then again aggregated on *Sum*. Next, add a new column as a measure with the following formula:

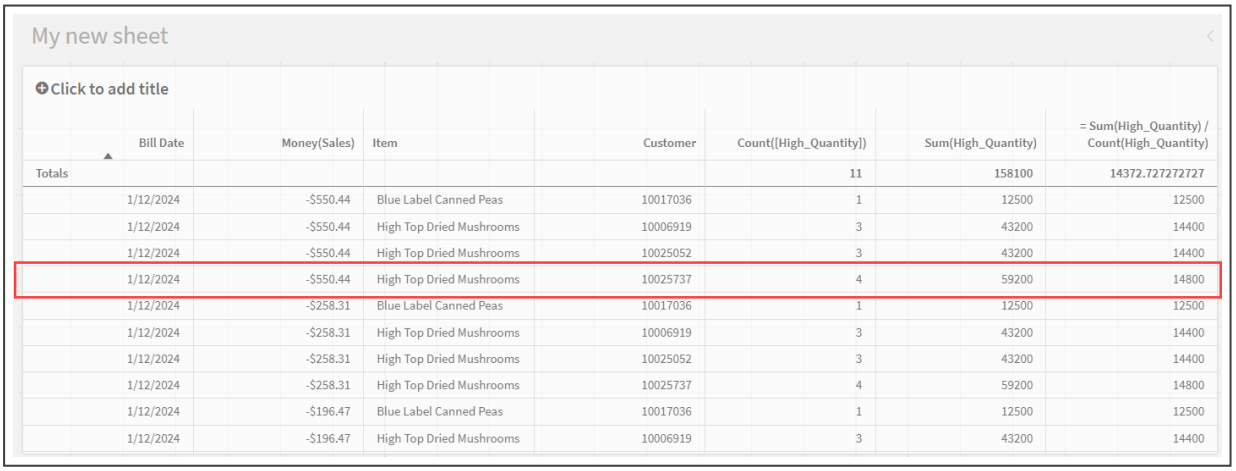
```
= Sum(High_Quantity) / Count(High_Quantity)
```

New measure with formula



Your table shows, for example, that *Customer* 10025737 has made 4 large orders of *High Top Dried Mushrooms*, with an average quantity of 14,800. To perform sorts on the data in the fields, close **Edit** mode by clicking **Done**.

Table showing customers that have made large orders



Bill Date	Money(Sales)	Item	Customer	Count([High_Quantity])	Sum(High_Quantity)	= Sum(High_Quantity) / Count(High_Quantity)
Totals				11	158100	14372.727272727
1/12/2024	-$550.44	Blue Label Canned Peas	10017036	1	12500	12500
1/12/2024	-$550.44	High Top Dried Mushrooms	10006919	3	43200	14400
1/12/2024	-$550.44	High Top Dried Mushrooms	10025052	3	43200	14400
1/12/2024	-$550.44	High Top Dried Mushrooms	10025737	4	59200	14800
1/12/2024	-$258.31	Blue Label Canned Peas	10017036	1	12500	12500
1/12/2024	-$258.31	High Top Dried Mushrooms	10006919	3	43200	14400
1/12/2024	-$258.31	High Top Dried Mushrooms	10025052	3	43200	14400
1/12/2024	-$258.31	High Top Dried Mushrooms	10025737	4	59200	14800
1/12/2024	-$196.47	Blue Label Canned Peas	10017036	1	12500	12500
1/12/2024	-$196.47	High Top Dried Mushrooms	10006919	3	43200	14400

7. Now that we have completed this example, comment out the script for the *quantity_threshold* variable and the *Sales_Buckets* table.

   The end of your script should now appear as follows:

Commented out script

```
(ooxml, embedded labels, table is [Sales data]);

// SET quantity_threshold = 12000;

//                Sales_Buckets:
//                LOAD
//                "Sales Qty" as "High_Quantity",
//                "Item Description" as "Item",
//                "Customer Number" as "Customer"
//                Resident Table1
//                Where ("Sales Qty" > $(quantity_threshold));
```

7.2 Preceding LOAD

A preceding load allows you to perform transformations and apply filters so that you can load data in one pass. Basically, it is a LOAD statement that loads from the LOAD or SELECT statement below, without specifying a source qualifier such as From or Resident that you would normally do. You can stack any number of LOAD statements this way. The statement at the bottom will be evaluated first, then the statement above that, and so on until the top statement has been evaluated.

As mentioned earlier in this tutorial, you can load data into applications using the LOAD and SELECT statements. Each of these statements generates an internal table. LOAD is used to load data from files or from an inline table, while SELECT is used to load data from databases. You have used data from files in this tutorial. In this example, you will use an inline table. However, it is worth noting that a preceding load can be used above a SELECT statement to manipulate your data. The basics are the same as you will see here using LOAD.

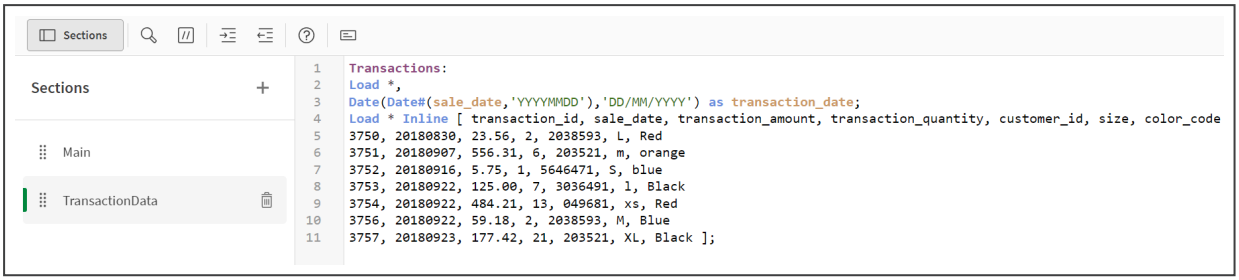
This example is not related to the data we are loading in this tutorial. It is only used to show an example of what a preceding load can look like. You will create an inline table in the data load editor called *Transactions*. Date interpretation will be performed in the preceding LOAD, where a new field called *transaction_date* will be created. This field is created from the *sale_date* field.

1. Create a new application and call it *ReformatDate*.
2. Open the data load editor, and then create a new tab called *TransactionData*.
3. Add the following script:

   ```
   Transactions:
   Load *,
   Date(Date#(sale_date,'YYYYMMDD'),'DD/MM/YYYY') as transaction_date;
   Load * Inline [ transaction_id, sale_date, transaction_amount, transaction_quantity,
   customer_id, size, color_code
   3750, 20180830, 23.56, 2, 2038593, L, Red
   3751, 20180907, 556.31, 6, 203521, m, orange
   3752, 20180916, 5.75, 1, 5646471, S, blue
   3753, 20180922, 125.00, 7, 3036491, l, Black
   3754, 20180922, 484.21, 13, 049681, xs, Red
   3756, 20180922, 59.18, 2, 2038593, M, Blue
   3757, 20180923, 177.42, 21, 203521, XL, Black ];
   ```

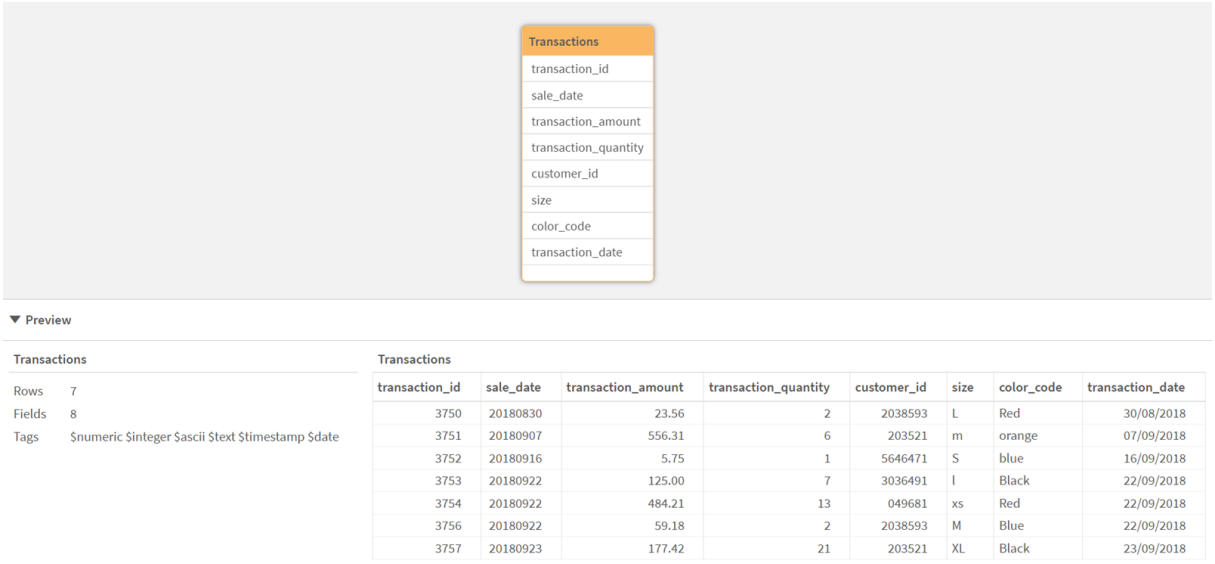
   Your script should look like this:

Load script with preceding load



4. Click **Load data**.
5. Open the **Data model viewer**. Select and expand the *Transactions* table. You can see that all the fields were loaded as specified by the * in the preceding load statement. A new field called *transaction_date* was created. The field has the reformatted date.

   New field called transaction_date in data model viewer



8 Concatenation

Concatenation is an operation that takes two tables and combines them into one.

The two tables are added to each other by stacking one on top of the other, with a column for each distinct column name. The data is not changed and the resulting table contains the same number of records as the two original tables together. Several concatenate operations can be performed sequentially, so that the resulting table is concatenated from more than two tables.

8.1 Automatic concatenation

If the field names and the number of fields of two or more loaded tables are exactly the same, the application will automatically concatenate the content of the different statements into one table.

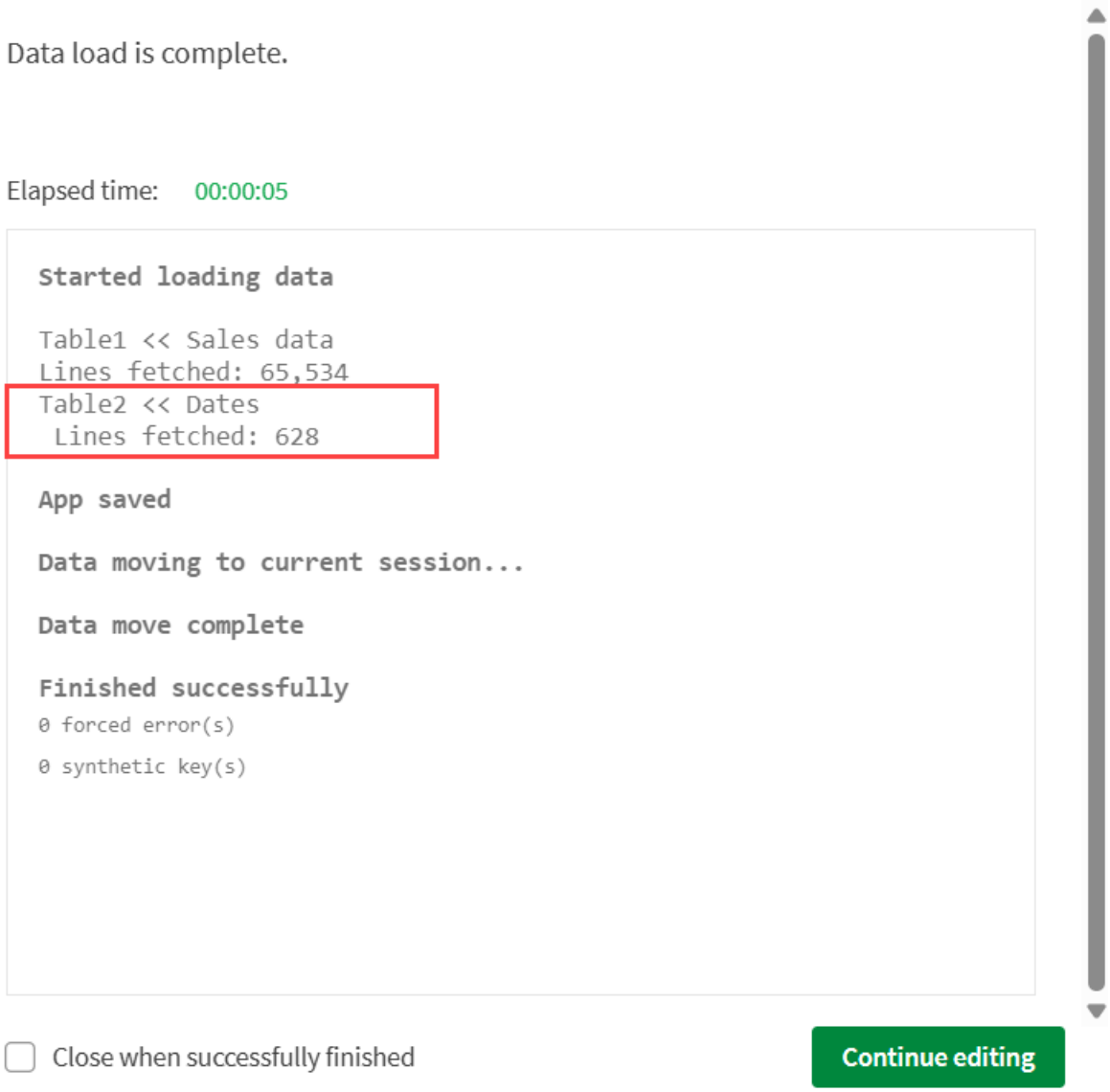
The number and names of the fields must be exactly the same for automatic concatenation to take place. The order of the two LOAD statements is arbitrary, but the table will be given the name of the table that is loaded first.

Do the following:

1. Open the **Data load editor** in the *Scripting Tutorial* application.
2. Click the **Date** tab.
3. Click **Load data**.

   Based on the load script that you have written so far, the application loads 628 lines from the *Dates.xlsx* data file into *Table2*.

Data load progress window



4. On a new line in the script in the section *Dates*, copy and paste the LOAD statement for *Table2*. This will cause the data to be loaded twice. Name the second table *Table2a*.

   You could also delete the existing script, and copy and paste the following:

```
Table2:
                                    LOAD
                                    "Date",
                                    Month (Date) as "Month",
                                    Quarter,
                                    "Week",
                                    "Year"
```

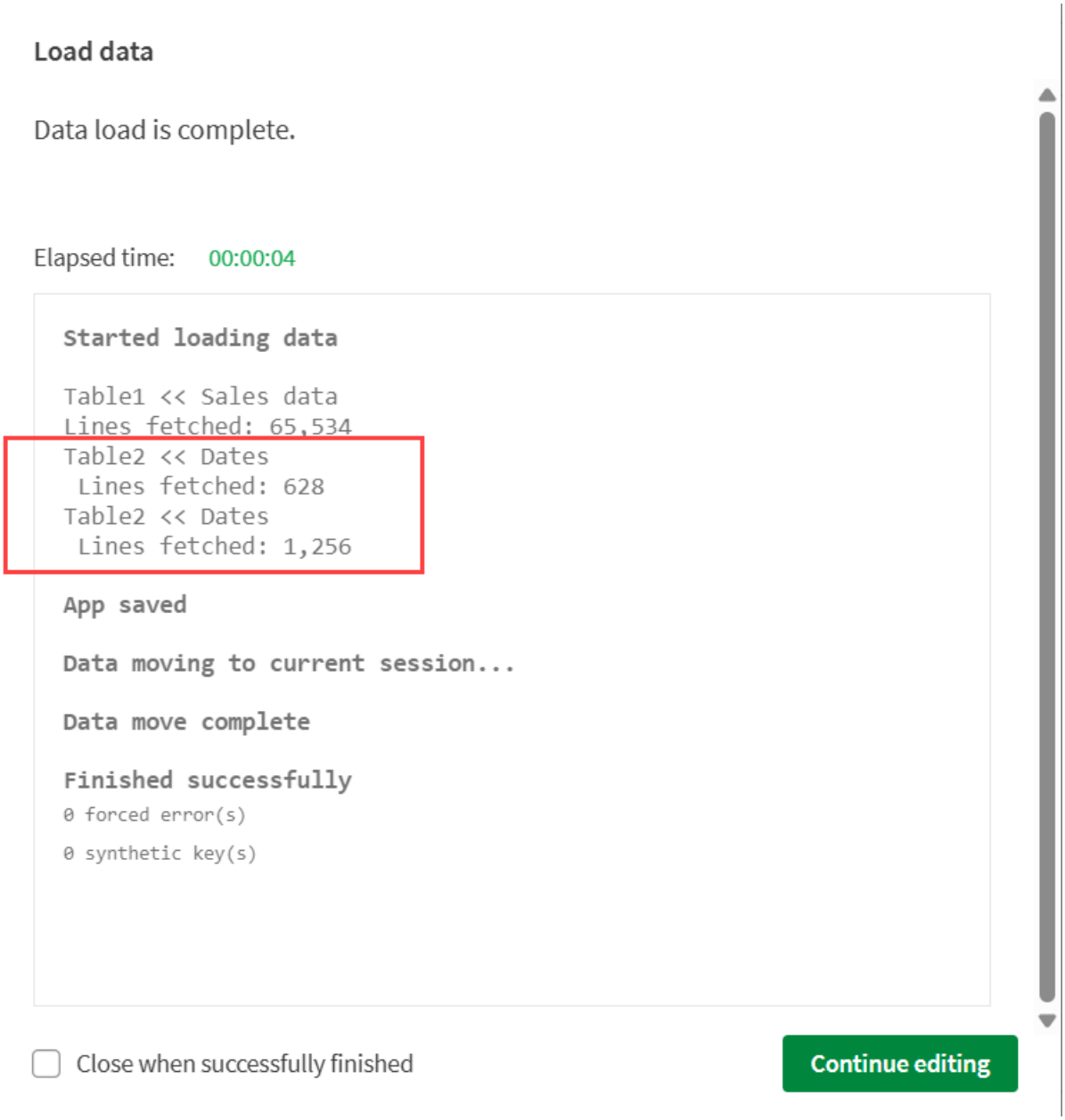
```
FROM [lib://AttachedFiles/Dates.xlsx]
(ooxml, embedded labels, table is Dates);

Table2a:
LOAD
"Date",
Month (Date) as "Month",
Quarter,
"Week",
"Year"
FROM [lib://AttachedFiles/Dates.xlsx]
(ooxml, embedded labels, table is Dates);
```

5. Click **Load data**.

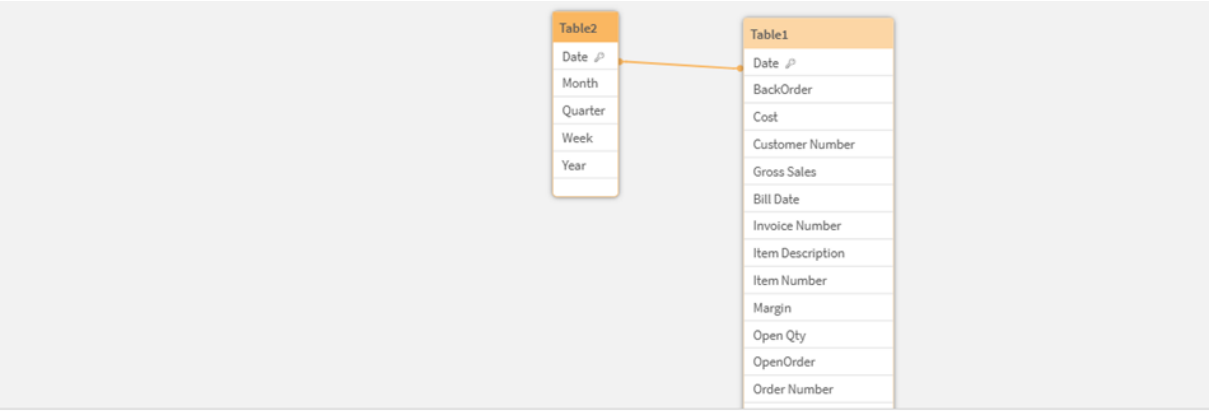
   The application does not load *Table2* and then *Table2a*. Instead, it recognizes that *Table2a* has the same field names and the number of fields as *Table2*. It then adds the data of *Table2a* to *Table2*, and deletes table *Table2a*. The result is that *Table2* now has 1,256 lines.

Concatenation in data load progress window



6. Open the **Data model viewer**.
7. Click **Show preview**.
   Only *Table2* was created. Select *Table2*. The table has 1256 rows.

Data model viewer showing Table2



▼ Preview

Table2

Rows	1256
Fields	5
Keys	1
Tags	$key $numeric $integer $timestamp $date $ascii $text

Table2

Date	Month	Quarter	Week	Year
01/12/2011	Jan	Q1	3	2011
01/13/2011	Jan	Q1	3	2011
01/18/2011	Jan	Q1	3	2011
01/19/2011	Jan	Q1	4	2011
01/20/2011	Jan	Q1	4	2011
01/21/2011	Jan	Q1	4	2011
01/22/2011	Jan	Q1	4	2011
01/25/2011	Jan	Q1	4	2011

8.2 Forced concatenation

Even if two or more tables do not have exactly the same set of fields, it is still possible to force the application to concatenate the two tables. This is done with the Concatenate prefix in the script, which concatenates a table with another named table or with the most recently created table.

Do the following:

1. Edit the LOAD statement for *Table2a*, adding Concatenate and commenting out *Week*.
   Your script should now appear as follows:

```
Table2a:
                                    Concatenate LOAD
                                    "Date",
                                    Month (Date) as "Month",
                                    Quarter,
                                    // "Week",
                                    "Year"
```

```
FROM [lib://AttachedFiles/Dates.xlsx]
(ooxml, embedded labels, table is Dates);
```

By commenting out *Week*, we make sure that the tables are not identical.

2. Click **Load data**.
3. Open the **Data model viewer**.
   Now you can see that *Table2a* has not been created.
4. Click *Table2* in the data model viewer, and then click **Preview**.
   The table has the fields *Date*, *Month*, *Quarter*, *Week*, and *Year*. The field *Week* is still showing, because it was loaded from *Table2*.
5. Click *Week* in *Table2*. The preview shows that the number of non-null values for the field is 628. However, it you click any of the other fields, you see that number of non-null values is 1256. *Week* was loaded just once, from *Table2*. The number of values, or records, is the sum of the number of records in *Table2* and *Table2a*.

8.3 Preventing concatenation

If the field names and the number of fields of two or more loaded tables are exactly the same, the application will automatically concatenate the content of the different statements into one table. This can be prevented with a NoConcatenate statement. The table loaded with the associated LOAD or SELECT statement will then not be concatenated with the existing table.

Do the following:

1. To be able to separate the content of the two tables completely, add NoConcatenate to the LOAD statement in *Table2a*, and rename the fields so that the application does not create a synthetic key based on the matching fields. Uncomment *Week* in *Table2* so that the two tables have the same fields.
   Your script should now appear as follows:

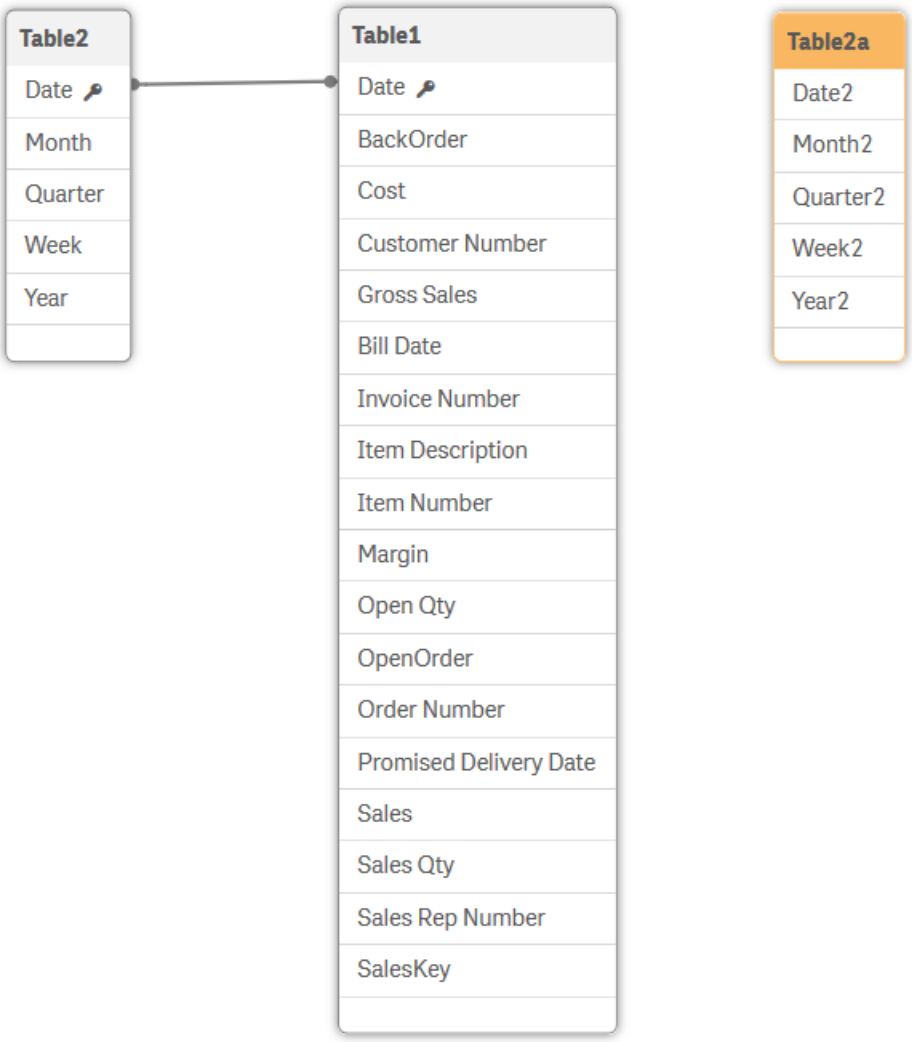
```
Table2:
                                    LOAD
                                    "Date",
                                    Month (Date) as "Month",
                                    Quarter,
                                    "Week",
                                    "Year"
                                    FROM [lib://AttachedFiles/Dates.xlsx]
                                    (ooxml, embedded labels, table is Dates);

                                    Table2a:
                                    NoConcatenate LOAD
                                    "Date" as "Date2",
                                    Month (Date) as "Month2",
                                    Quarter as "Quarter2",
                                    "Week" as "Week2",
                                    "Year" as "Year2"
```

```
        FROM [lib://AttachedFiles/Dates.xlsx]
(ooxml, embedded labels, table is Dates);
```

2. Click **Load data**.
3. Open the **Data model viewer**.
   Now you can see that the two tables are completely separated.

   Data model viewer showing Table2 and Table 2a



4. Now that we have finished demonstrating concatenation, we no longer need *Table2a*. Delete all the rows in the LOAD statement for *Table2a*, and then click **Load data**.

9 Circular references

If there are circular references (loops) in a data structure, the tables are associated in such a way that there is more than one path of associations between two fields.

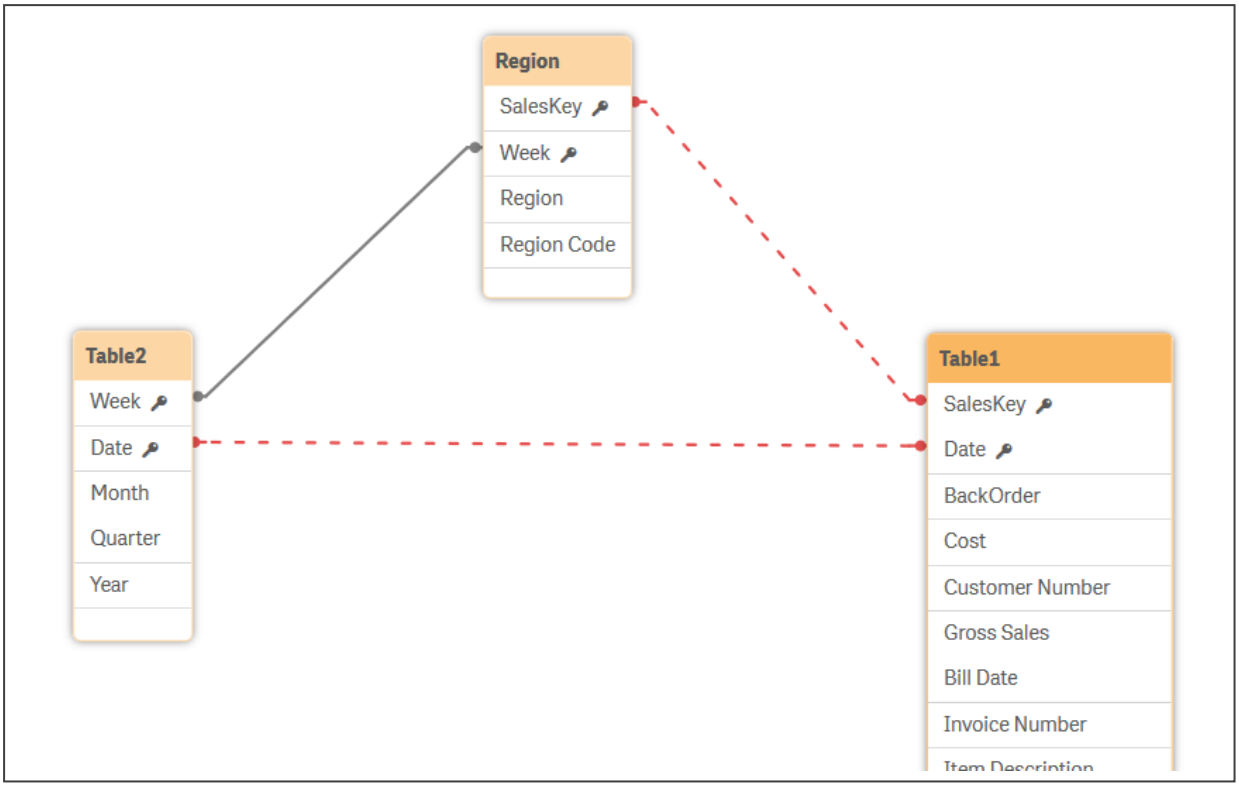
Do the following:

1. Open the **Data load editor** in the *Scripting Tutorial* application.
2. Click + to add a new script section.
3. Name the section *Region*.
4. Under **AttachedFiles** in the right menu, click **Select data**.
5. Upload and then select *Region.txt*. A data preview window opens.
6. Select all fields and make sure that **Embedded field names** under **Field names** is selected to include the names of the table fields when you load the data.
7. Click **Insert script**.
8. Click **Load data**.

   This time it appears that something has gone wrong with your data load. A circular reference has been created. The **Data load progress** window will show an error message, stating that a circular reference was found during the load. However, the load is completed and the application is saved.
9. Open the **Data model viewer**.

   You can drag the tables around to reorganize them in a way so that the connections between the tables are easy to see.

Data model viewer showing circular reference



The red dotted lines indicate that a circular reference has been created. This is something that you should avoid, because it may cause ambiguities in the data interpretation.

9.1 Resolving circular references

To be able to understand what caused the circular references, let us take a closer look at your tables in the **Data model viewer**.

If you look at *Table1* and the *Table2* in the screenshot above, you can see that they have the *Date* field in common. You can also see that *Table1* and *Region* have the *SalesKey* field in common. Finally, notice that *Table2* and *Region* have the field *Week* in common. This means that a loop, a circular reference, has been created. Since this can cause problems in the data analysis later on, let us remove it.

The easiest way to resolve this is to rename or remove one of the fields. In our case, we have loaded some data that we do not need for our application, and we can remove it.

Do the following:

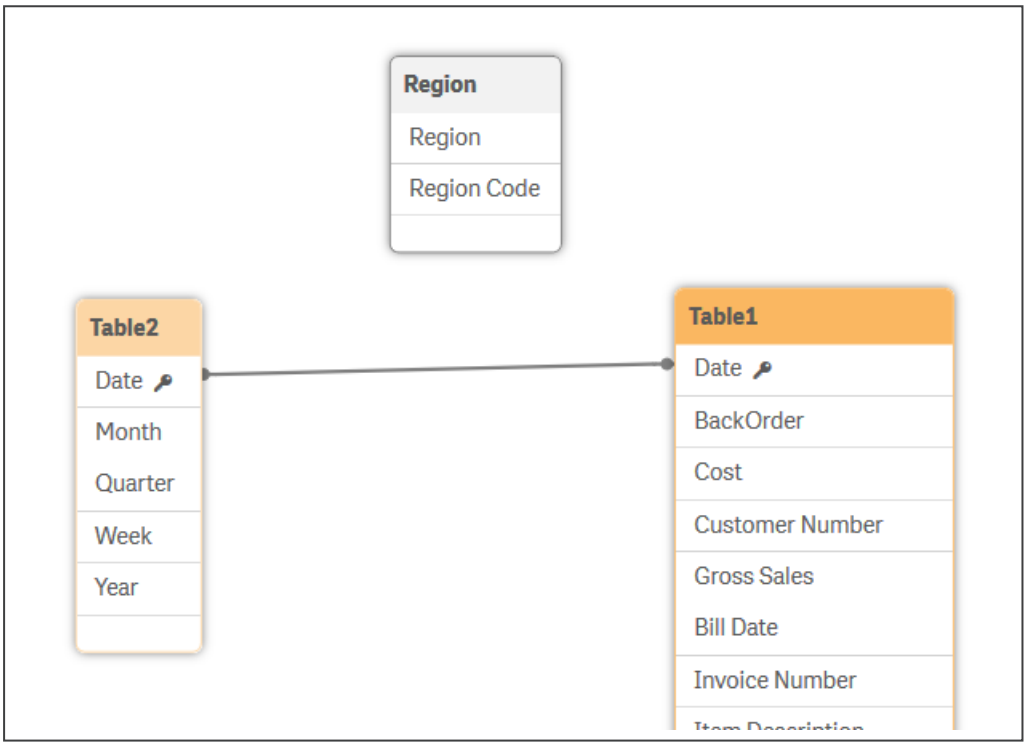
1. Open the **Data load editor**.
2. Click on the section *Region* and delete the following two rows in the *LOAD* statement:
   ```
   "Week",
   SalesKey
   ```
3. Make sure to also remove the comma after *"Region Code"*.
   Your script should now appear as follows:

```
LOAD
    Region,
    "Region Code"
FROM [lib://AttachedFiles/Region.txt]
(txt, codepage is 28591, embedded labels, delimiter is '\t', msq);
```

4. Click **Load data**.
5. Open the **Data model viewer**.

   The undesired references to *Region* have been removed.

   Data model viewer showing that circular reference has been removed



10 Synthetic keys

When two or more internal tables have two or more fields in common, this implies a composite key relationship. Applications handle this through synthetic keys. These keys are anonymous fields that represent all occurring combinations of the composite key.

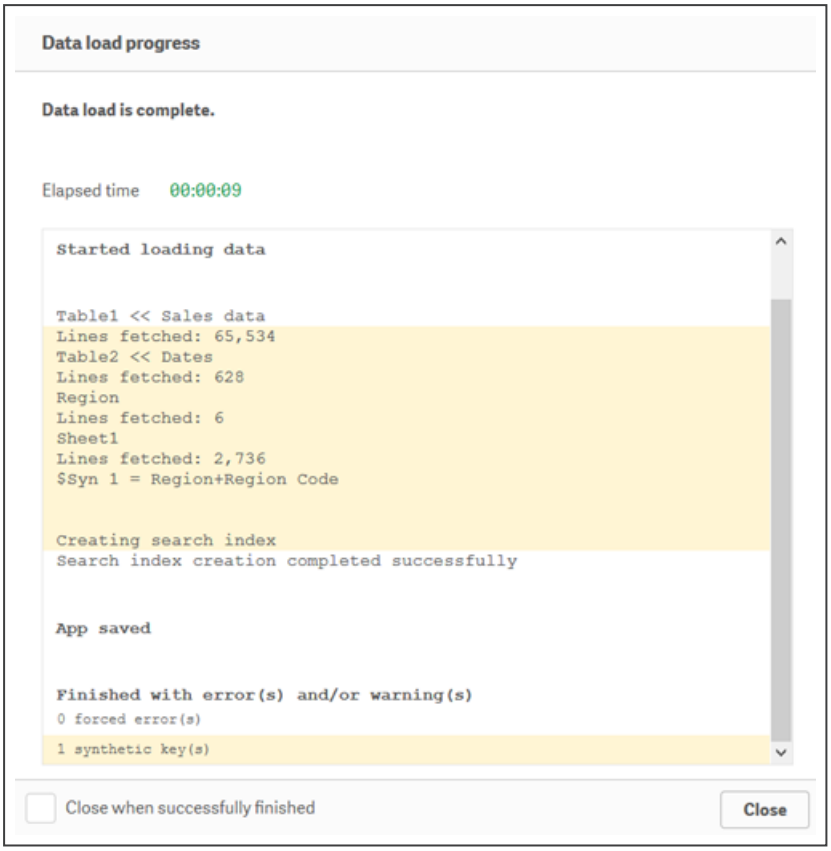
An increased number of composite keys can result in elevated memory usage, and can affect performance. This may also depend on data amounts, table structure, and other factors. Whenever there are several synthetic keys that are dependent on each other, it is good practice to remove them.

Now it is time to load our final set of data.

Do the following:

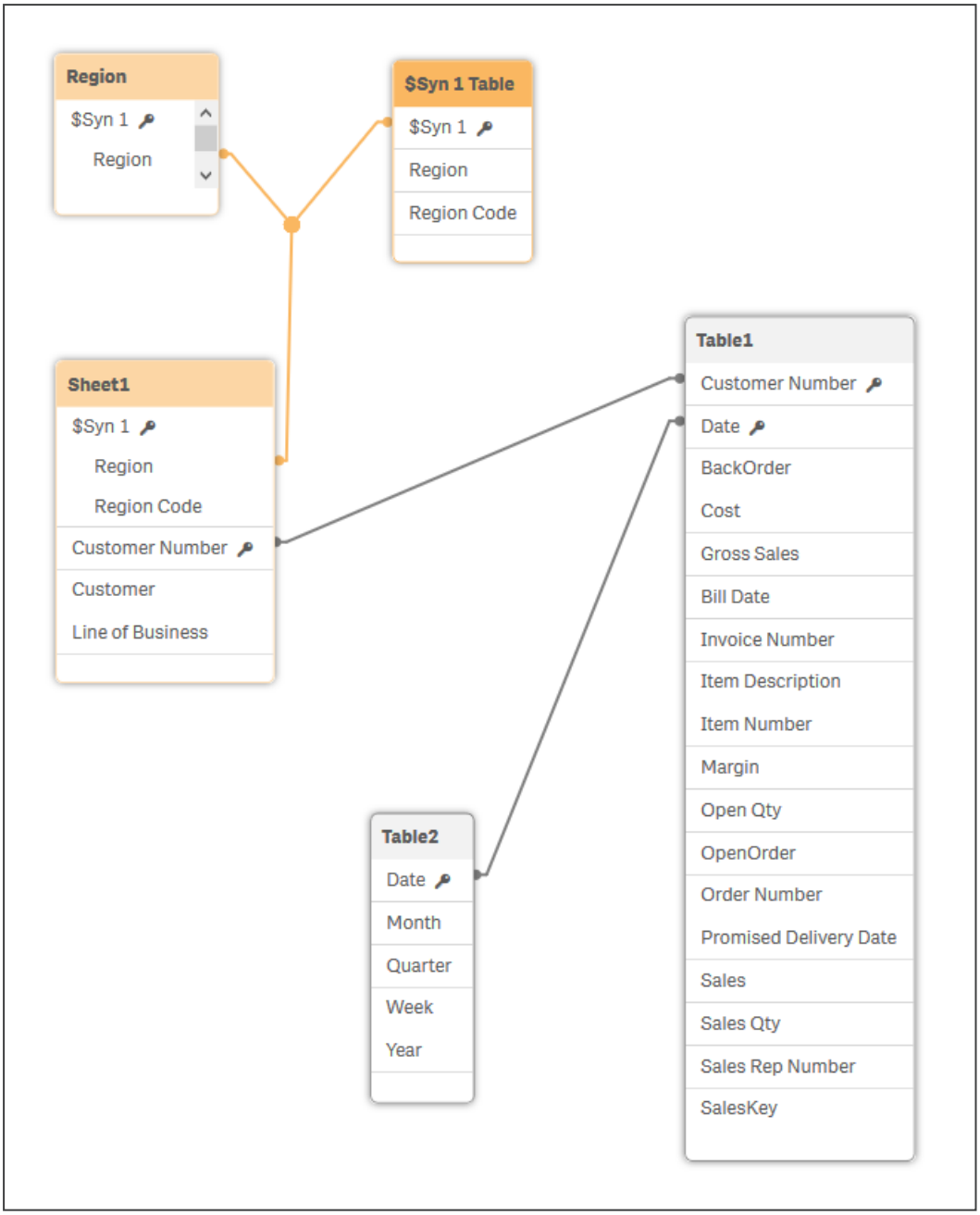
1. Open the **Data load editor** in the *Scripting Tutorial* application.
2. Click + to add a new script section.
3. Name the section *Customers*.
4. Under **AttachedFiles** in the right menu, click **Select data**.
5. Upload and then select *Customers.xlsx*. The data preview window opens.
6. Select *Sheet1*.
7. Click **Insert script**.
8. Click **Load data**.
   Now you can see in the data load progress window that a synthetic keys was created.

Data load progress window with synthetic key warning



9. Open the **Data model viewer**.
   We can see that a synthetic key has been created by seeing that a new table, *$Syn 1 Table*, has been created. It contains all the fields, *Region* and *Region code*, that the connected tables *Sheet1* and *Region* have in common. In this case it makes the connections a bit confusing and misleading, so it is not desirable to keep.

Data model viewer showing synthetic key



10.1 Resolving synthetic keys

The easiest way to eliminate synthetic keys is to rename one or more fields in the tables. This can be done when loading the data. Now we will go through the steps of how to remove a synthetic key.

Do the following:

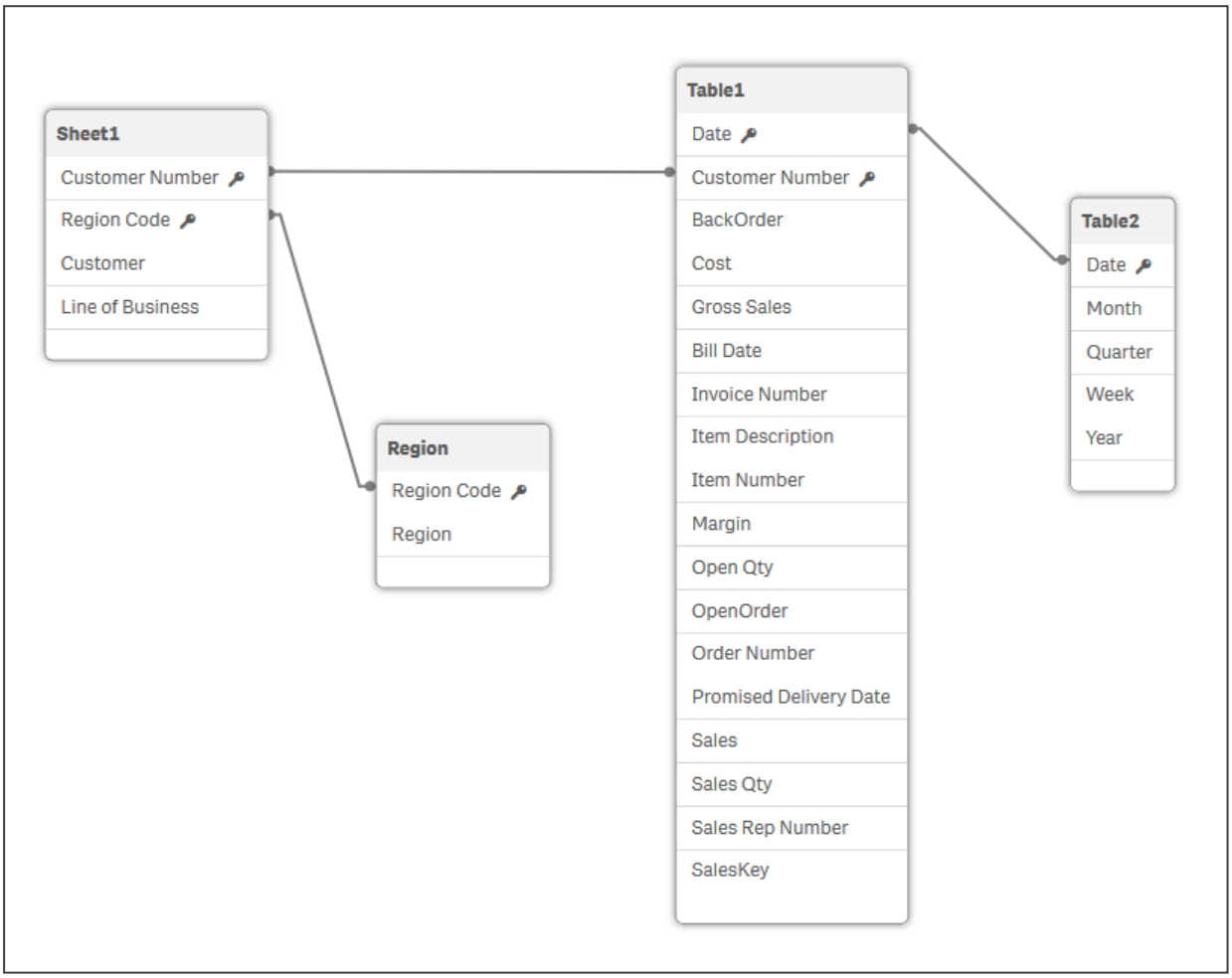
1. Open the **Data load editor**.
2. Click the section *Customers* and delete the row in the **LOAD** statement saying:

```
Region,
```

3. Click **Load data**.
4. Open the **Data model viewer**.

   The synthetic key is has been removed.

   Data model viewer showing that synthetic key has been removed



11 Using data in an application

To finish off this tutorial, it is time for you to put your loaded data into a visualization in your application.

11.1 Adding a chart

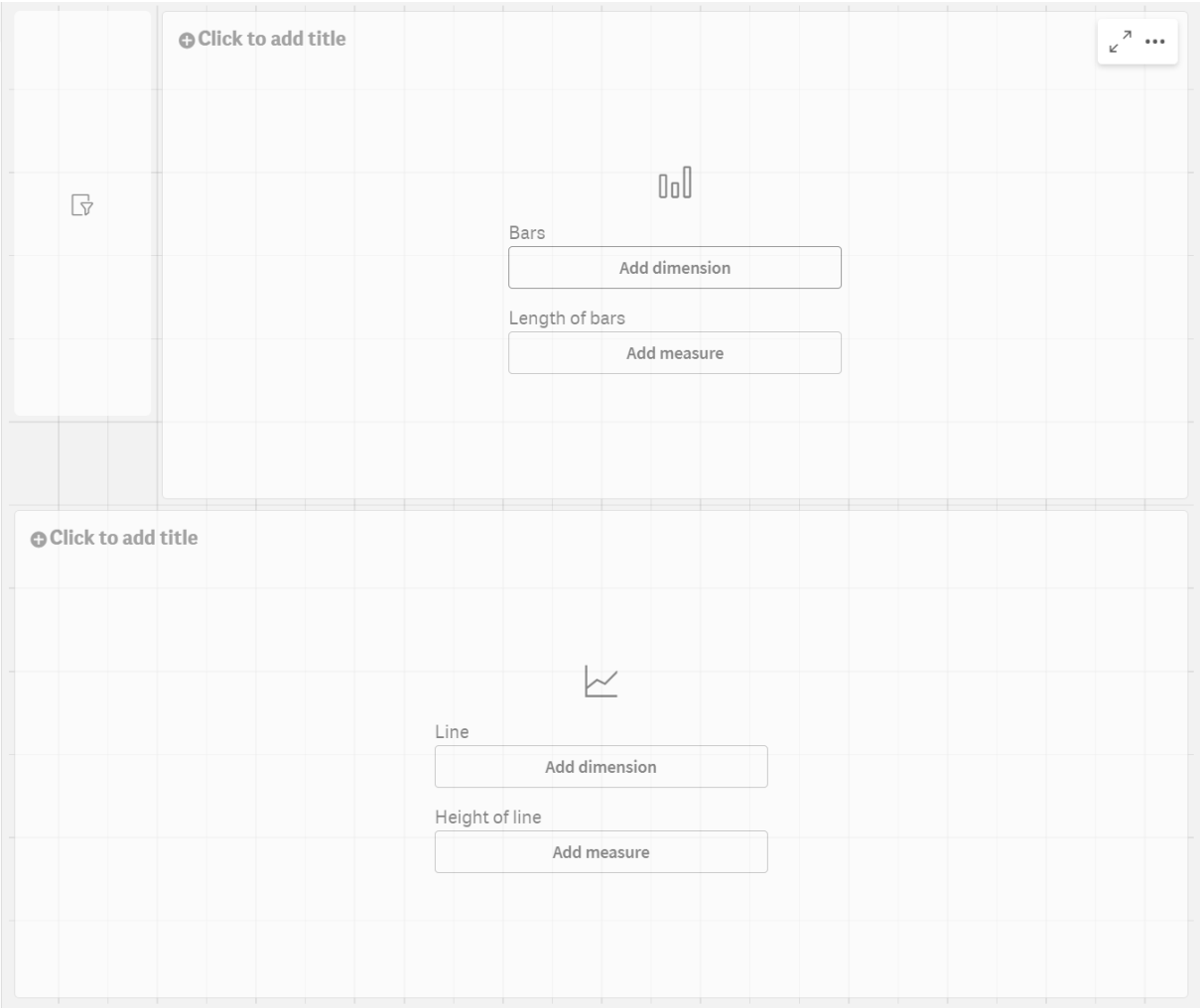
Now that your data is loaded, you can create some charts with the data. Charts are often also called visualizations. It is not until you have also added the required dimensions and measures that a chart is complete. You will start by adding the charts and then continue by adding dimensions and measures.

Do the following:

1. Create a new sheet in your *Scripting Tutorial* application.
2. In the properties panel, give the sheet the title *Product sales*.
3. Drag a filter pane from the **Charts** tab to the sheet and use the handles to resize it so that it is 3 cells wide and 4 cells tall. Place it in the top left corner of the sheet.
4. Drag a bar chart to the top right corner, make it 5 cells tall and wide enough to stretch until the side of the sheet.
5. Drag a line chart to the remaining space.

   The icons on the sheet show what sort of chart you have added. Now you can add dimensions and measures to your charts to complete them as visualizations.

Application sheet with empty charts



11.2 Adding dimensions and measures

The next step is to add dimensions and measures. Begin by adding time dimensions to the top left filter pane. The benefit of a filter pane is that you save space. Instead of having one filter pane each for *Year*, *Quarter*, *Month*, and *Week*, you will use only one filter pane for the same purpose.

Creating and adding dimensions

Do the following:

1. In the assets panel, click **Fields**. Here you find all the fields in all the tables that you have loaded in the data load editor.
2. Scroll down to the bottom of the list, and click the field *Year*. Drag it to the center of the top left filter pane.
3. In the same way, add *Quarter*, *Month* and *Week* to the filter pane.
   You have now created a filter pane with four dimensions: *Year*, *Quarter*, *Month*, and *Week*.

Creating and adding measures

Most visualizations need both dimensions and measures. A measure is the result of an aggregation expression, which is, in many cases, a common function, such as **Sum**, **Max**, **Min**, **Avg** (average), or **Count**.

In the bar chart you will show the sales by region.

Do the following:

1. In **Fields**, click the field *Region* and drag it to the center of the bar chart area.
2. Click **Add "Region"**.
3. Click the field *Sales* and drag it to the center of the bar chart area.
4. Click **Add as measure > Sum(Sales)**.
5. In the properties panel on the right-hand side, click **Appearance** and then **Presentation**. Select **Horizontal**.
   The bars are now displayed horizontally.
6. In the properties panel on the right-hand side, click **Sorting**.
   The sorting order is displayed.
7. Drag *Sum([Sales])* above *Region* so that the dimensions are sorted by *Sum([Sales])* (measure value) instead of *Region* (dimension value, alphabetically).

   The bar chart is complete, showing the sales results for the different regions. This is a basic bar chart. There are many options to enhance it in the properties panel (to the right). Just to show you one of the possibilities, let us use the title area for something more than just a title.

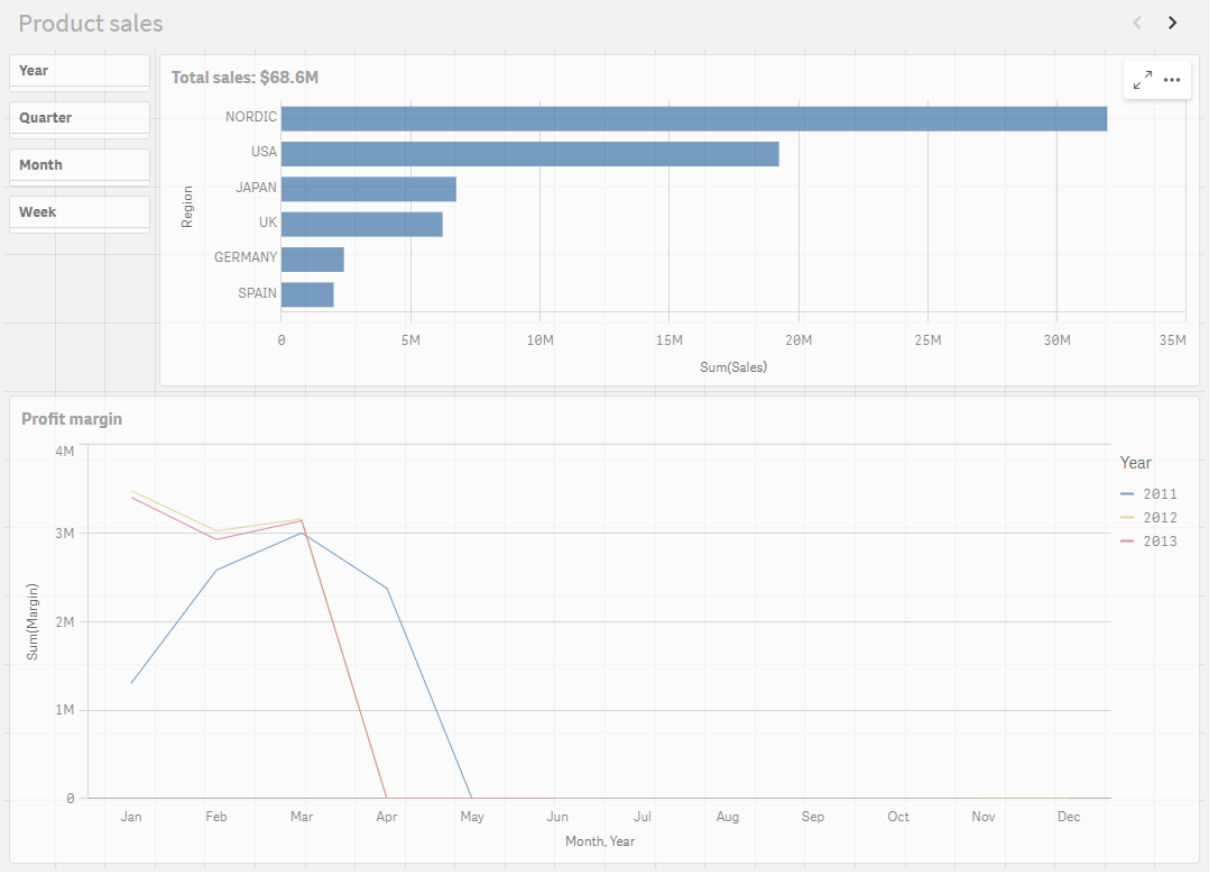
8. Paste the following into the title field of the bar chart:
   ```
   ='Total sales: $'& Round(Sum(Sales)/1000000, 0.1) & 'M'
   ```
9. Press Enter.

The final visualization on this sheet is a line chart.

Do the following:

1. In **Fields**, click the field *Month* and drag it to the center of the line chart area.
2. Click **Add "Month"**.
3. Click the field *Year* and drag it to the center of the line chart area.
4. Click **Add "Year"**.
5. Click the field *Margin* and drag it to the center of the line chart area.
6. Click **Add as measure > Sum(Margin)**.
7. Add the title *Profit margin* on top of the line chart.

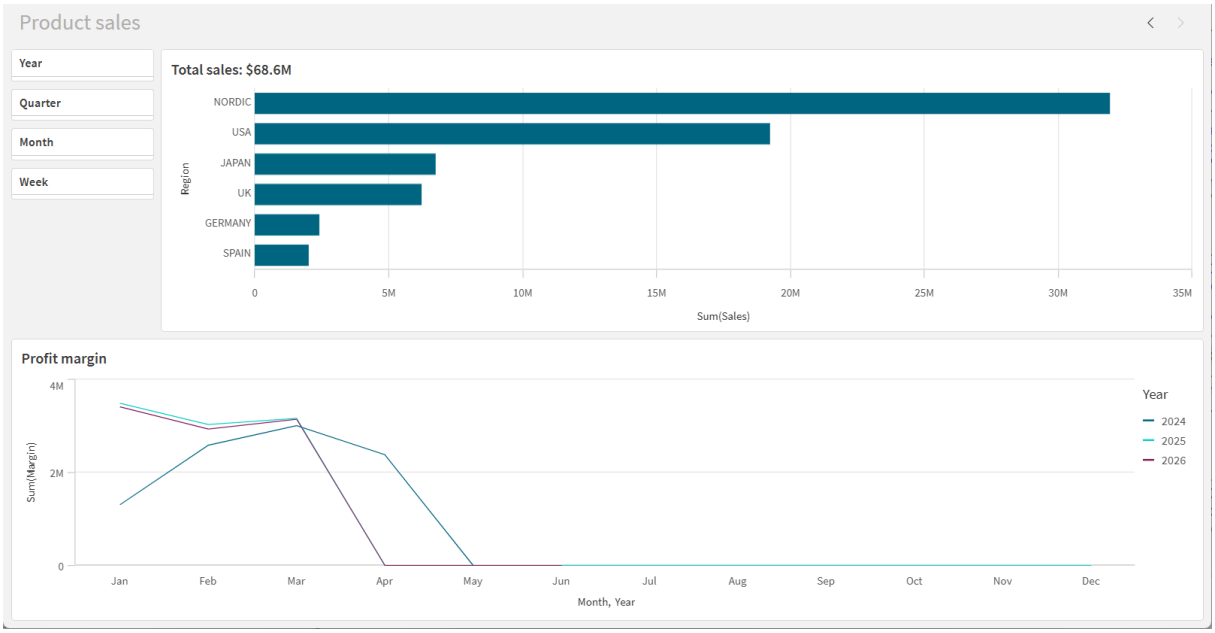
Charts with data



8. Stop editing the sheet.

   The sheet is now complete and you can begin to click around and interact with the contents of the sheet.

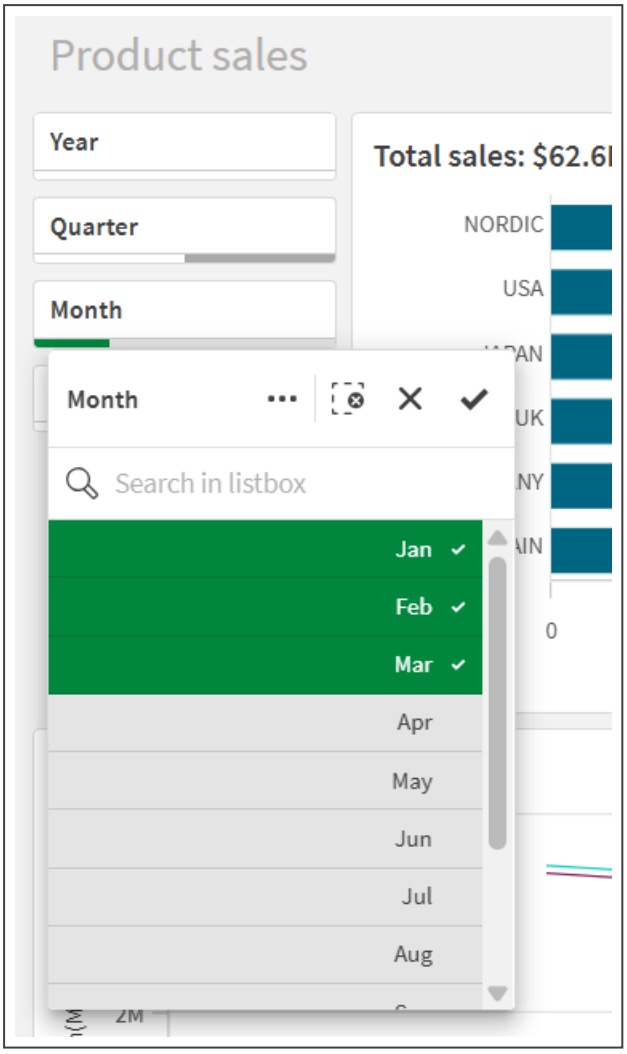
Completed sheet



Because we limited the amount of monthly sales data in our original *Sales.xlsx* file, there is little data available for our chart after the end of March for each year. You can make selections in your filter pane so that you are only comparing the first three months of each year.

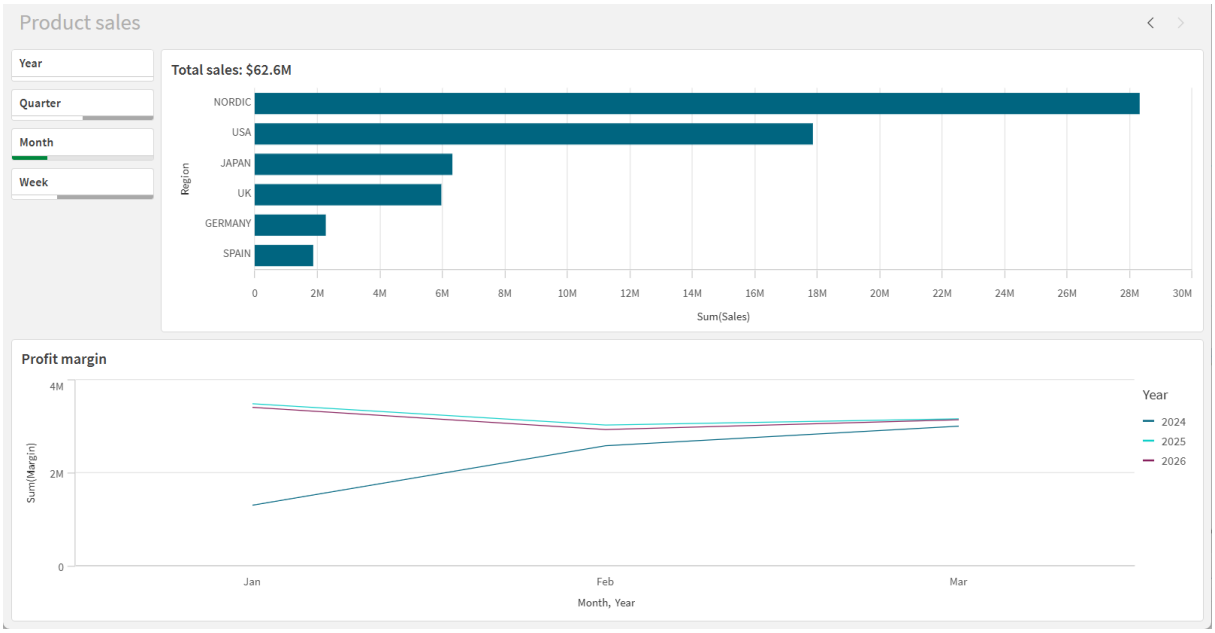
9. Click the *Month* field in the filter box, and then select *Jan*, *Feb*, and *Mar*.

Filter box



10. Close the filter pane. The *Profit margin* chart shows the data for the first three months of each year.

Profit margin chart updated based on selections



11.3 Thank you!

Now you have finished this tutorial, and hopefully you have gained some basic knowledge about scripting in applications. Please visit our website for more inspiration for your applications.