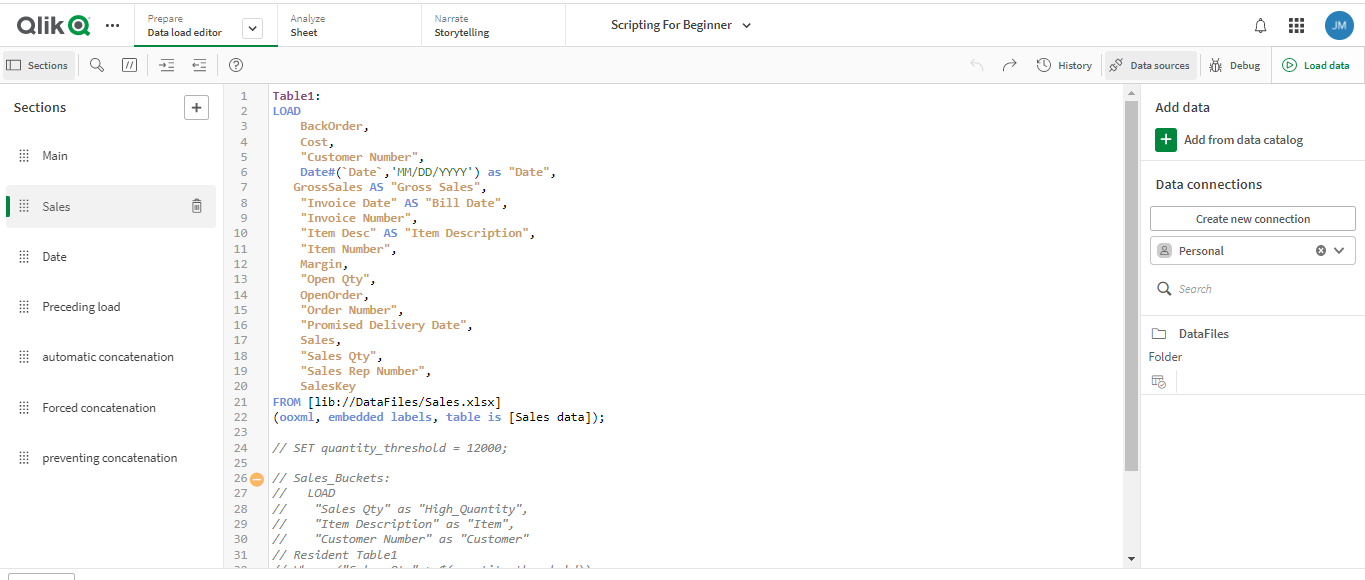
Scripting for Beginners

1. **Scripting in the data load editor**

Qlik Sense uses 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.

**2.Data load editor**

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



**3.Script editor**

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

1. **LOAD and SELECT statements**

You can load data into Qlik Sense 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.

Transforming data

You can transform and manipulate data in the data load editor before using the data in your

app.

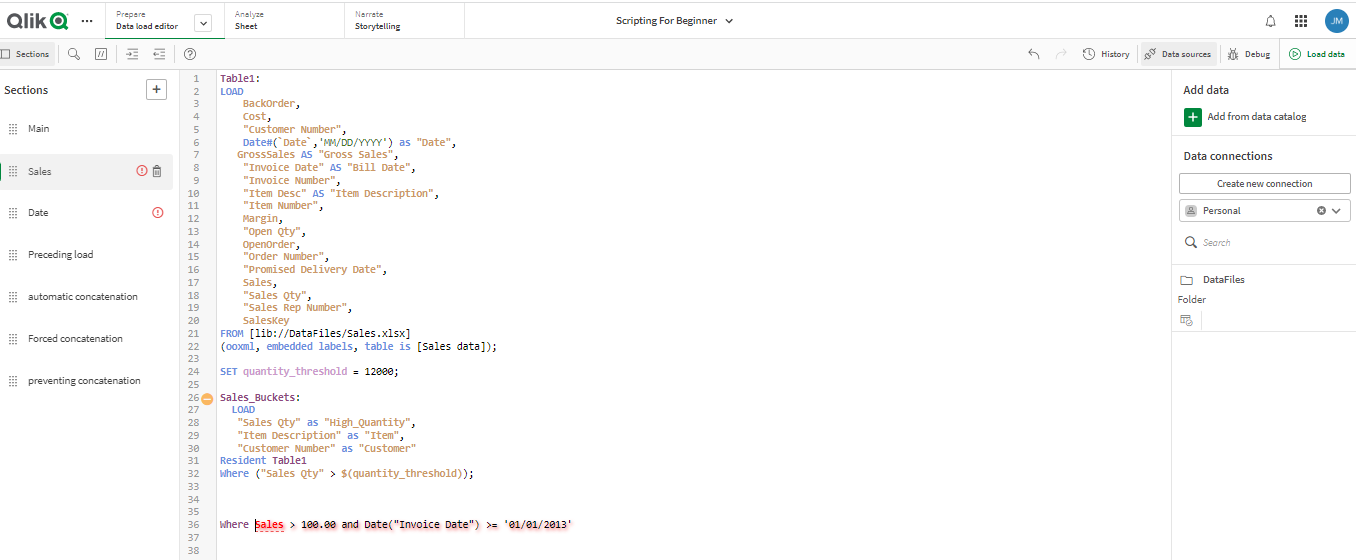
**5.Selecting and loading data**

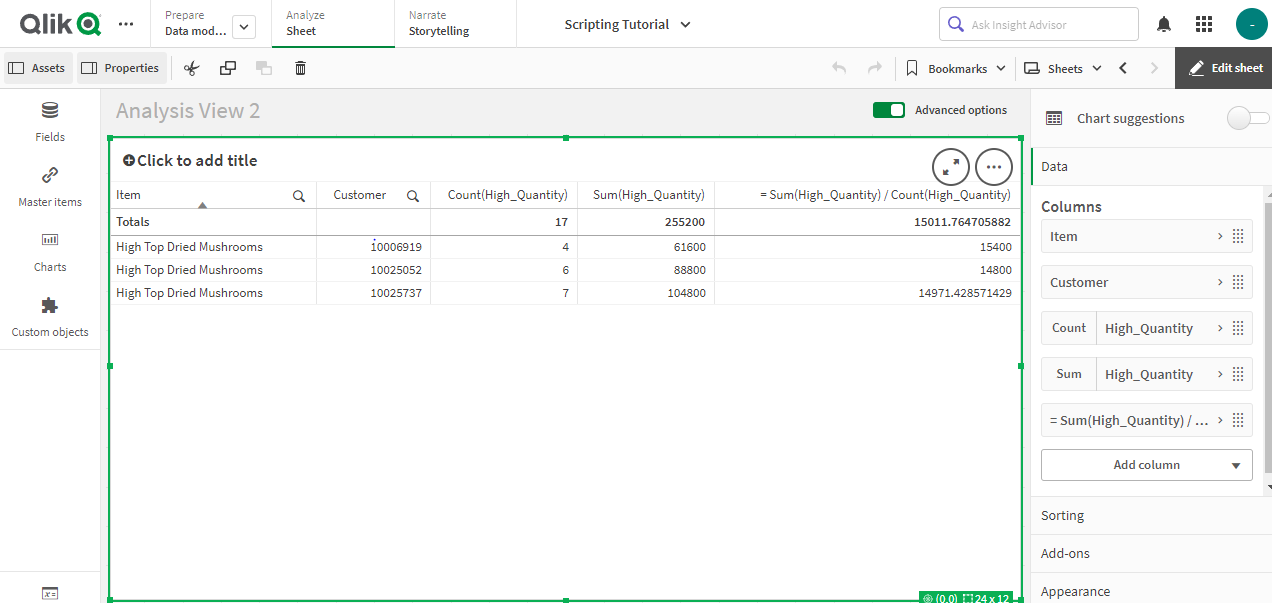
Loading data from files, such as Microsoft Excel or any other supported file format, is easily

done by using the data selection dialog in the data load editor.

**6.Resident LOAD**

You can use the Resident source qualifier in a LOAD statement to load data from a previously loaded table.

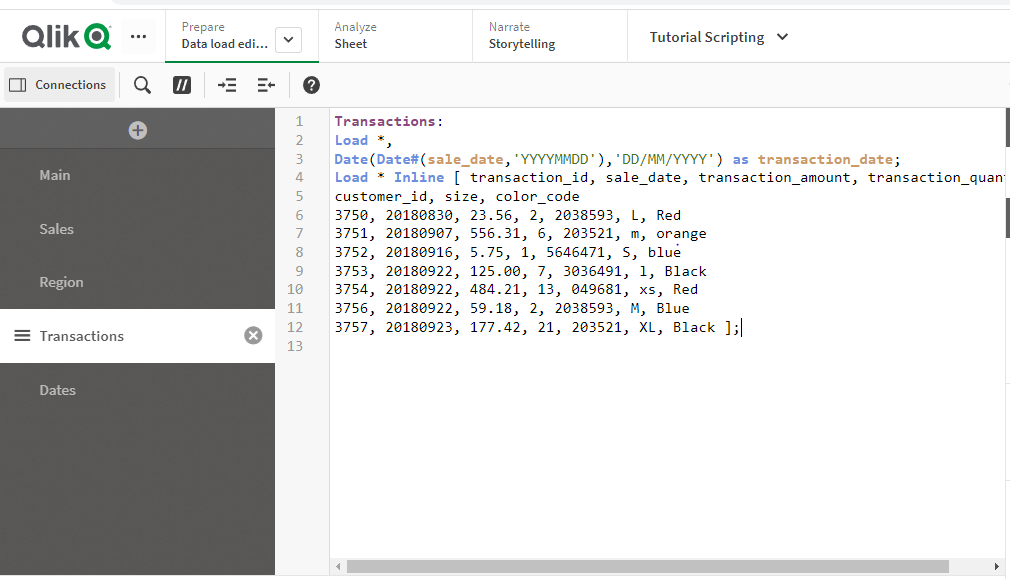


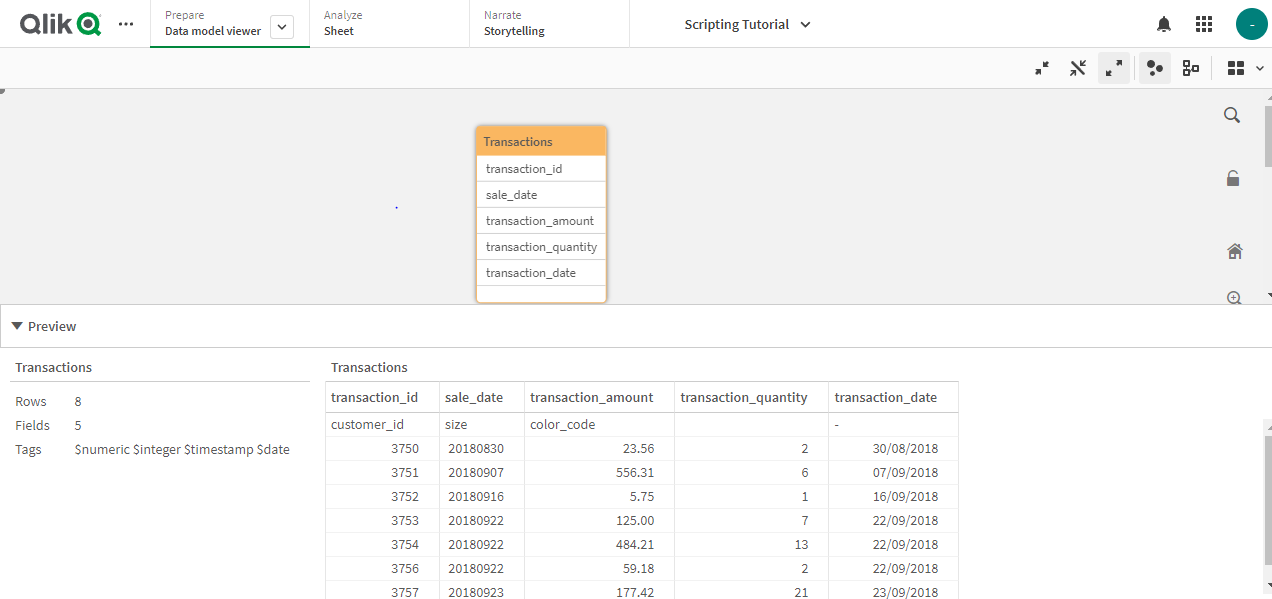


**Preceding LOAD**

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.

Load \*

****

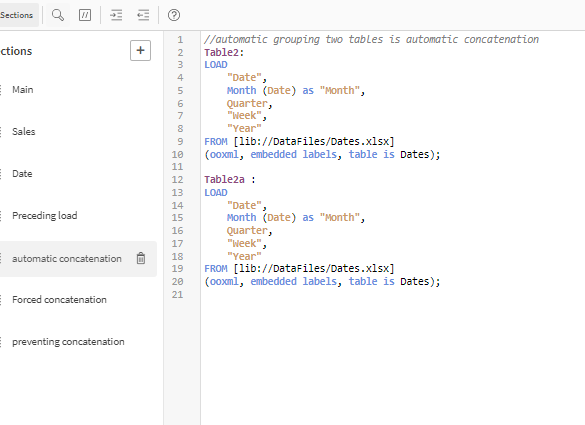


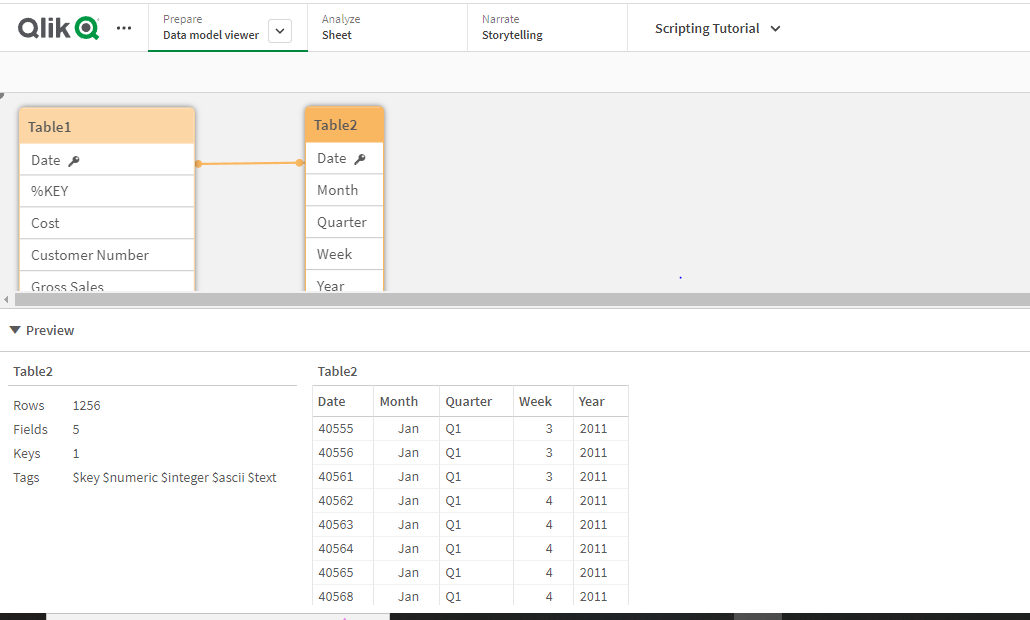
**Concatenation**

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

**Automatic concatenation**

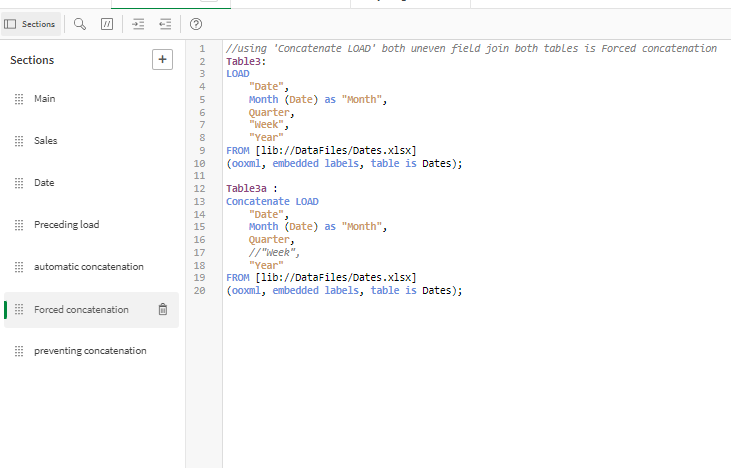
If the field names and the number of fields of two or more loaded tables are exactly the same, Qlik Sense will automatically concatenate the content of the different statements into one table.





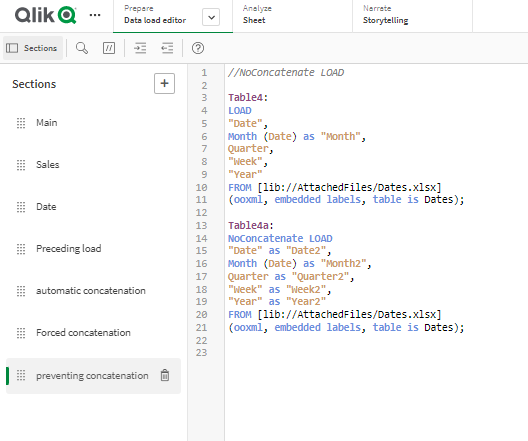
**Forced concatenation**

Even if two or more tables do not have exactly the same set of fields, it is still possible to force Qlik Sense to concatenate the two tables.



**Preventing concatenation**

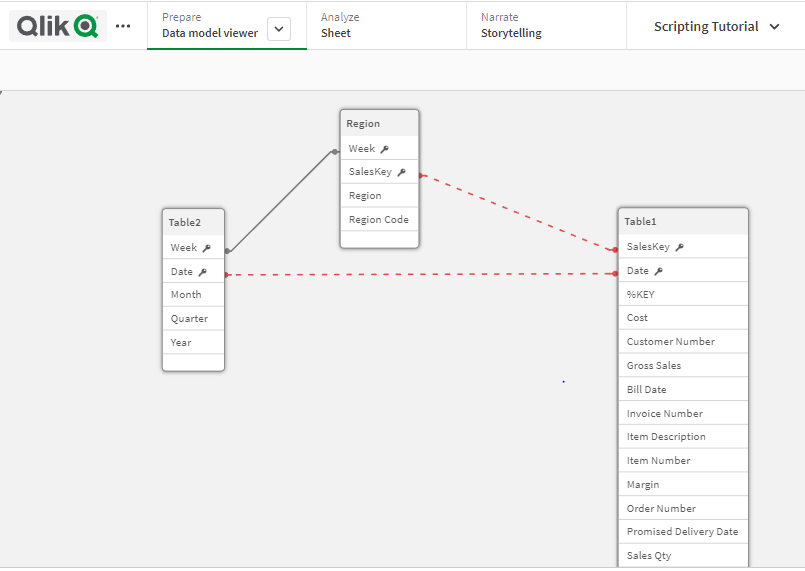
If the field names and the number of fields of two or more loaded tables are exactly the same, Qlik Sense will automatically concatenate the content of the different statements into one table. This can be prevented with a NoConcatenate statement.



Day -2

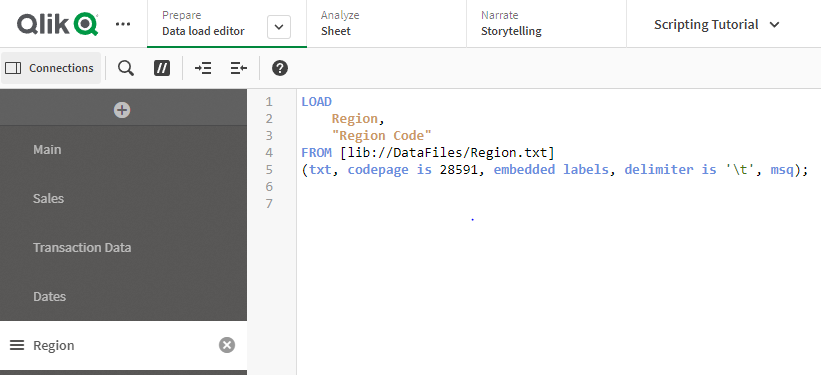
**Circular references:**

Circular reference is nothing but formation of loop between multiple tables.

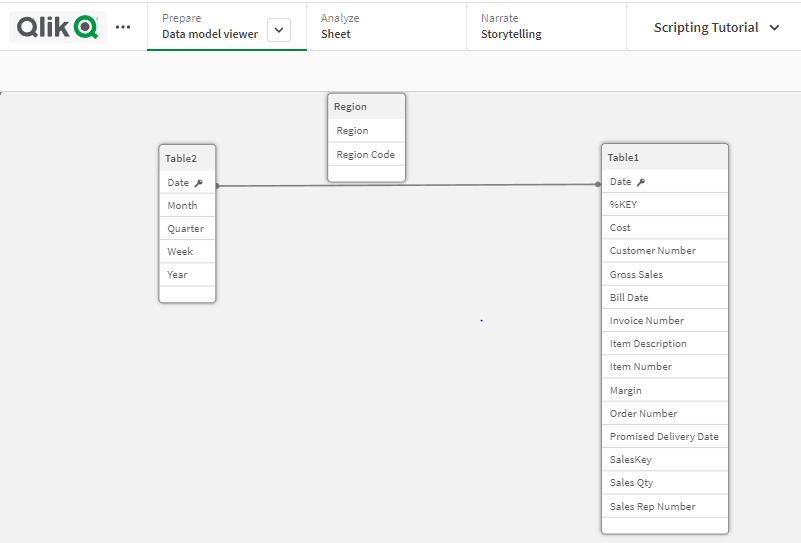


**Resolving circular references:**

Circular reference issue can be solved by changing the name of the field or removing the field from the table.



I have deleted the week and sales key fields in the date load editor.

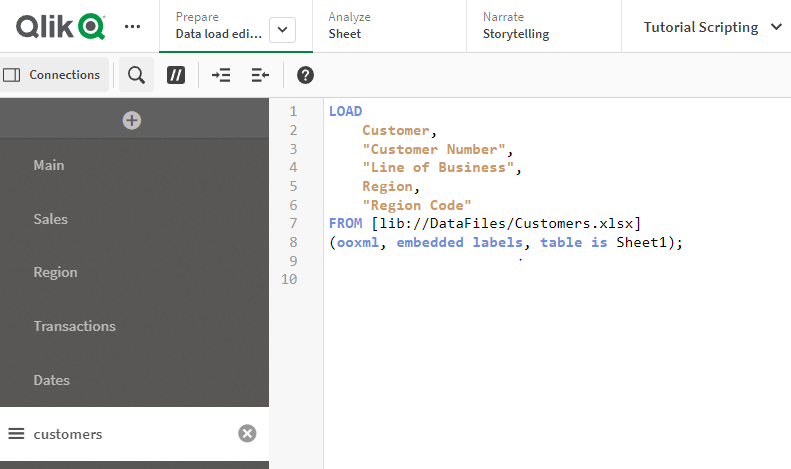


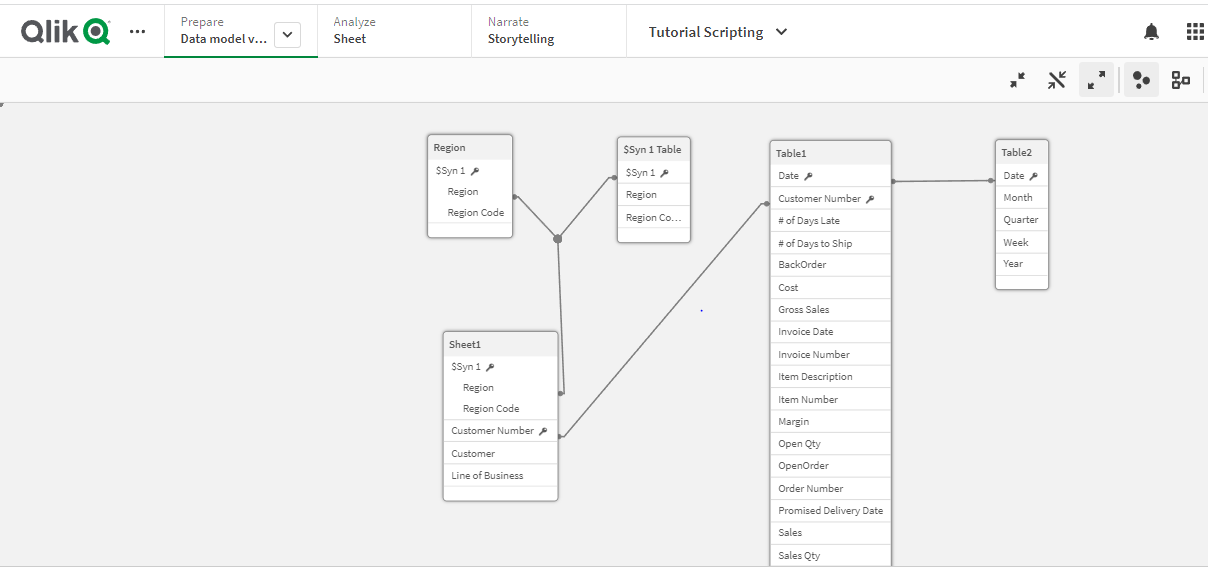
**Synthetic keys:**

When two or more internal tables have two or more fields in common, this implies a

composite key relationship.

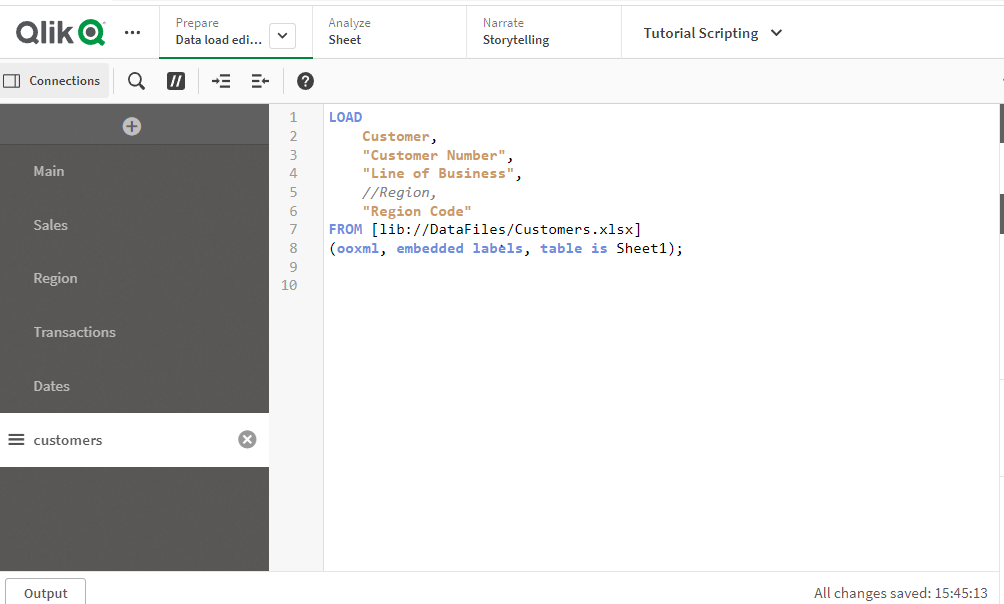
When two are more tables are linked with two or more keys, then the synthetic key error occurs.





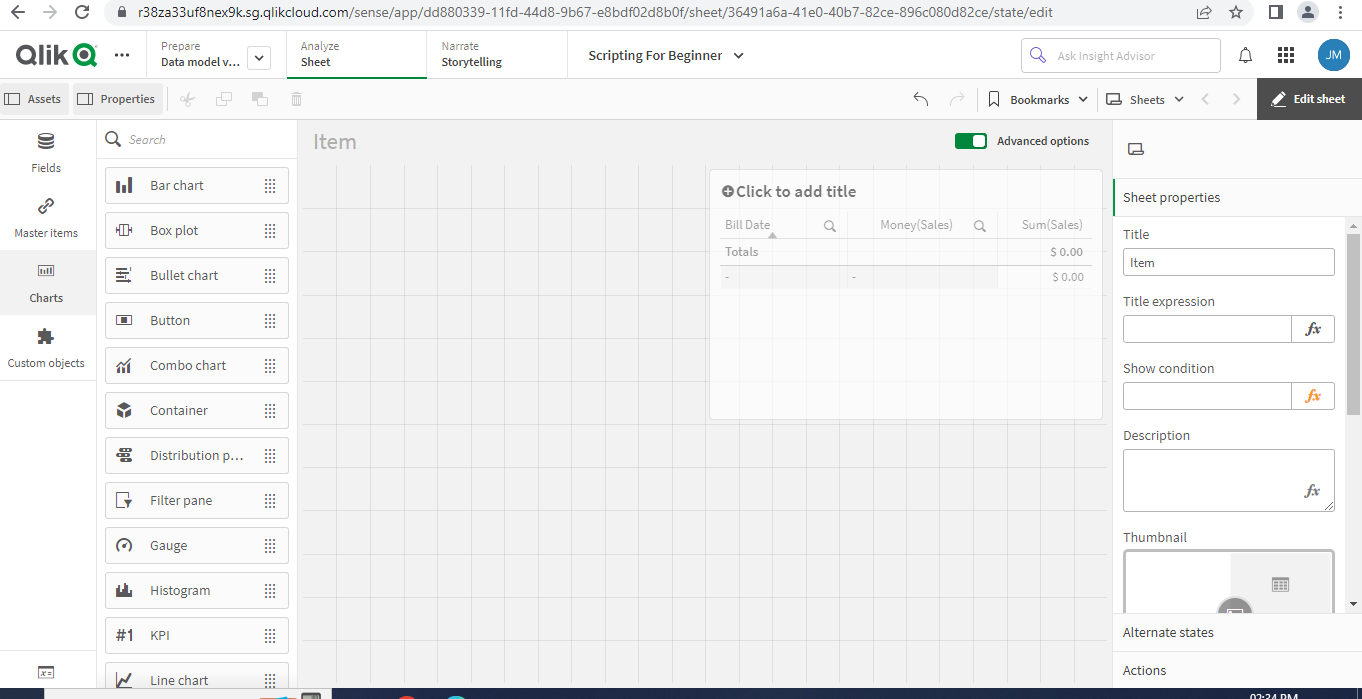
**Resolving synthetic keys:**

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.

 I have comment the region filed in sheet 1.

Using data in an app

1. Adding a chart



1. Adding dimensions and measures

Filter Chart:

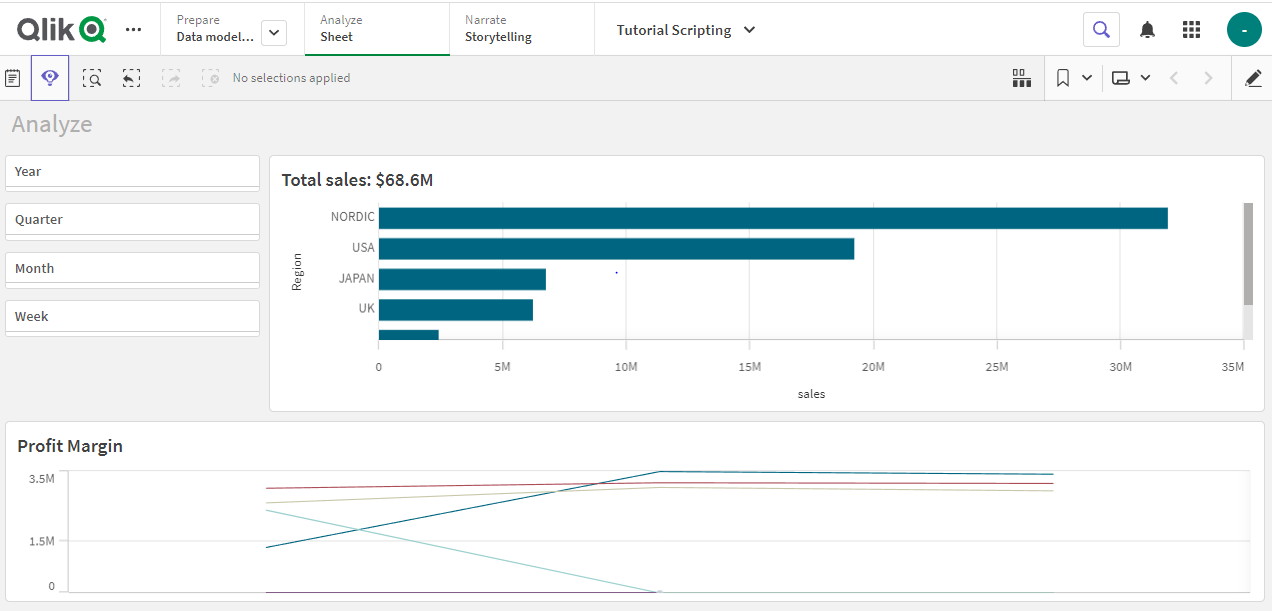
I have added the year, quarter, month and week in dimensions.

Bar Chart:

I have added the region in dimensions and sum (sales) in measures.

Line Chart:

I have added the year and month in dimensions and sum (margin) in measures.



**Using the Crosstable prefix**

Cross tables are a common type of table featuring a matrix of values between two orthogonal

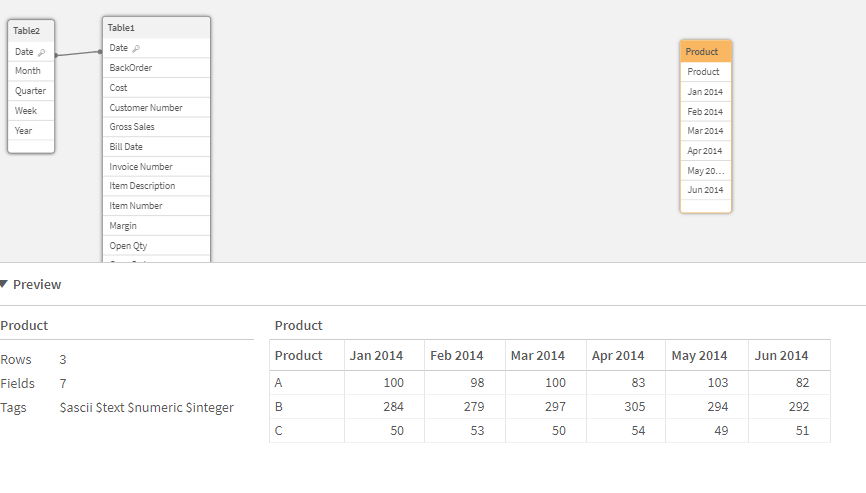
lists of header data. Whenever you have a cross table of data, you can use the Crosstable

prefix **to transform the data and create the desired fields.**

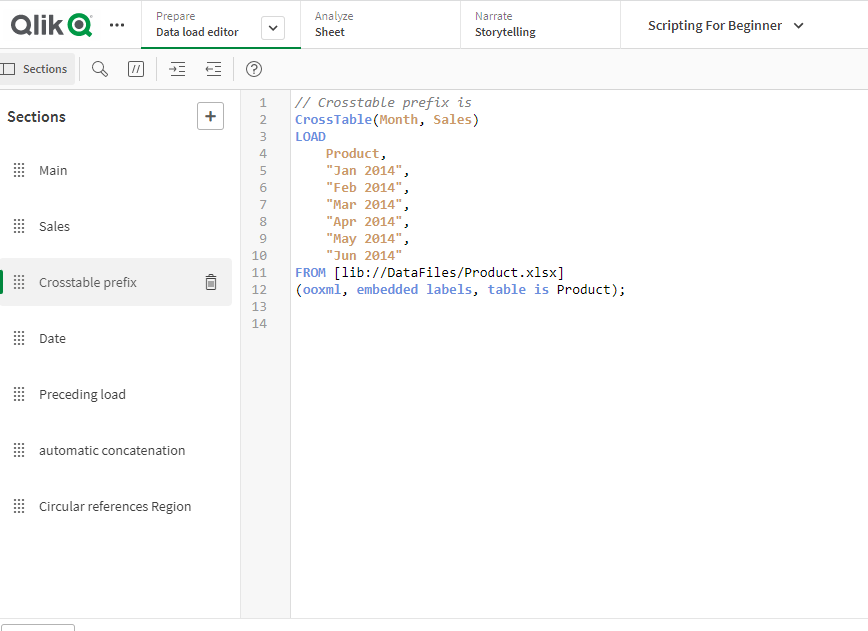
Enter the following above the LOAD statement:

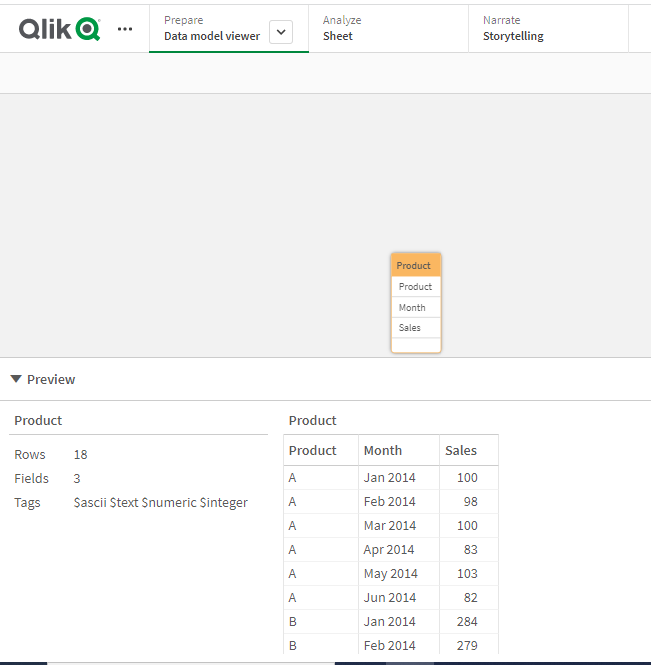
**Script-”CrossTable(,)”**

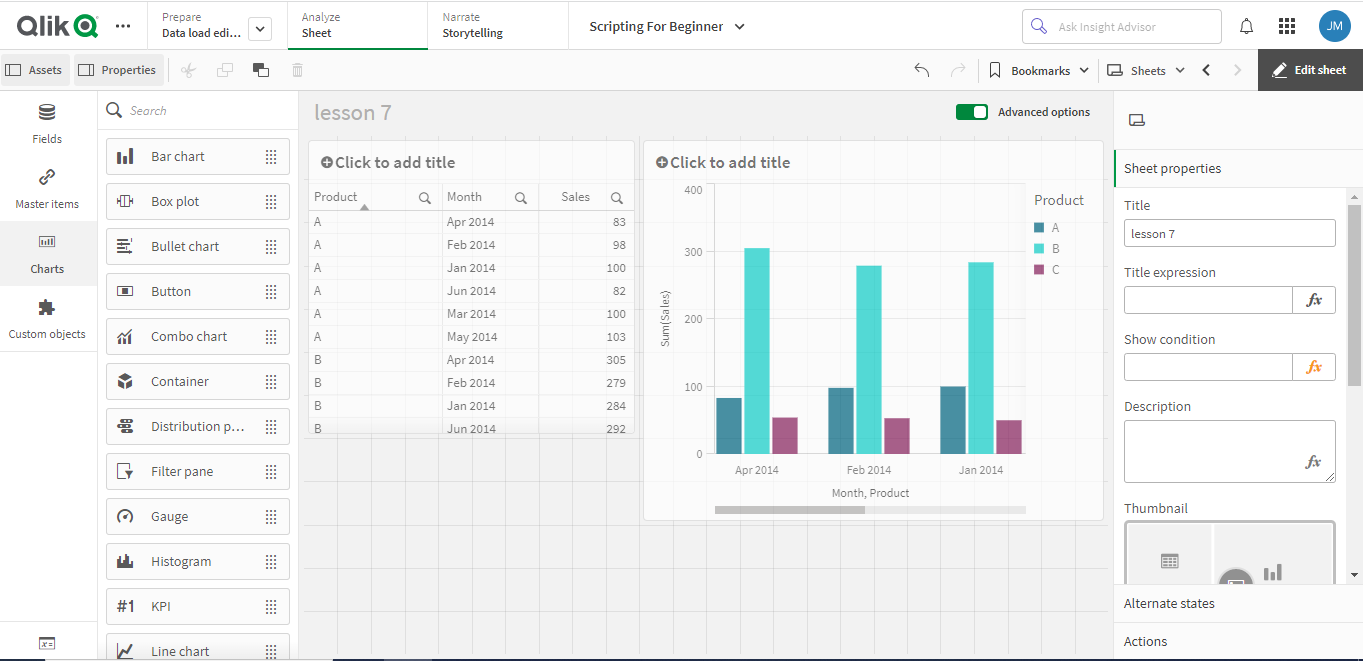
Before apply cross table in script:



Apply in script:







**Day 3**

**Combining tables with Join and Keep**

**A join is an operation that uses two tables and combines them into one.** The records of the resulting table are combinations of records in the original tables, usually in such a way that the two records contributing to any given combination.

**Join**

The simplest way to make a join is with the Join prefix in the script, which joins the internal table with another named table or with the last previously created table. The join will be an outer join, creating all possible combinations of values from the two tables.

**Example:**

LOAD

a, b, c from table1.csv;

Join LOAD a, d from table2.csv;

**Using Join**

The explicit Join prefix in the Qlik Sense script language performs a full join of the two tables. The result is one table. Such joins can often result in very large tables.

**Keep**

One of the main features of Qlik Sense is its ability to make associations between tables instead of joining them, which reduces space in memory, increases speed and gives enormous flexibility.

* The Keep functionality has been designed to reduce the number of cases where explicit joins need to be used.
* The Keep prefix must always be preceded by one of the keywords Inner, Left or Right.

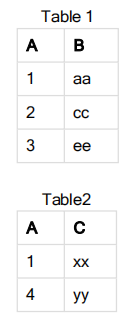
**Inner**

The Join and Keep prefixes in the data load script can be preceded by the prefix Inner.

* If used before Join, it specifies that the join between the two tables should be an inner join.
* The resulting table contains only combinations between the two tables with a full data set from both sides.

**Example:**

In these examples we use the source tables Table1 and Table2.



**Inner Join**

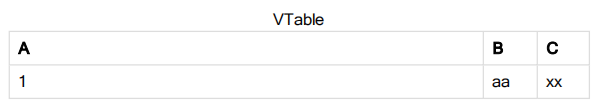
First, we perform an Inner Join on the tables, resulting in SampleTable, containing only one row, the only record

existing in both tables, with data combined from both tables.

**Sample Table:**

**SELECT \* from Table1;**

**inner join SELECT \* from Table2;**



**Inner Keep**

If we perform an Inner Keep instead, we will still have two tables. The two tables are associated via the

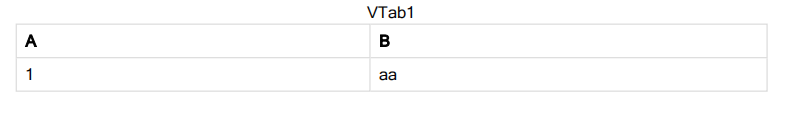
common field A.

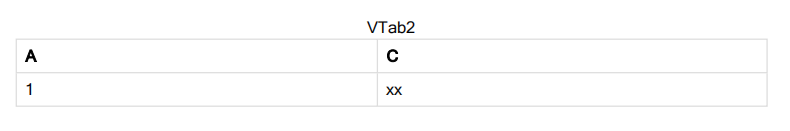
**VTab1:**

**SELECT \* from Table1;**

**VTab2:**

**inner keep SELECT \* from Table2;**





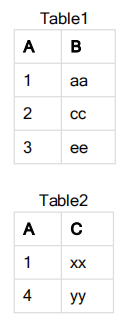
Left JOIN

The Join and Keep prefixes in the data load script can be preceded by the prefix left.

* If used before Join, it specifies that the join between the two tables should be a left join.
* The resulting table only contains combinations between the two tables with a full data set from the first table.
* If used before Keep, it specifies that the second table should be reduced to its common intersection with the first table before being stored in Qlik Sense.

**Example:**

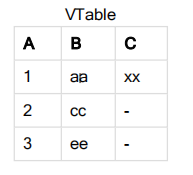
In these examples we use the source tables Table1 and Table2.



**VTable:**

**SELECT \* from Table1;**

**left join SELECT \* from Table2; LEFTJOIN**



**LEFT KEEP**

If we perform a Left Keep instead, we will still have two tables. The two tables are associated via the

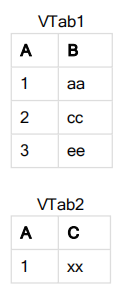
common field A.

**VTab1:**

**SELECT \* from Table1;**

**VTab2:**

**left keep SELECT \* from Table2;**



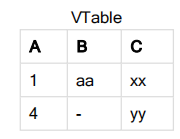
**RIGHT JOIN**

First, we perform a Right Join on the tables, resulting in VTable, containing all rows from Table2, combined with fields from matching rows in Table1.

**VTable:**

**SELECT \* from Table1;**

**right join SELECT \* from Table2;**



**RIGHT KEEP**

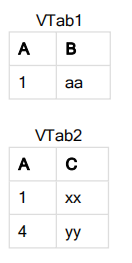
If we perform a Right Keep instead, we will still have two tables. The two tables are associated via the common field A.

**VTab1:**

**SELECT \* from Table1;**

**VTab2:**

**right keep SELECT \* from Table2;**



Using inter-record functions: Peek, Previous, and Exists

These functions are used when a value from previously loaded records of data is needed for the evaluation of the current record.

**Peek()**

The Function is used to view the values of specific row values is peek().

If no row number is specified, the last previously loaded record will be used.

**Syntax:**

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

**Previous()** finds the value of the **expr** expression using data from the previous input record that has not been discarded because of a **where** clause. In the first record of an internal table, the function will return.

NULL.

**Syntax:**

Previous(expression)

**Exists()** determines whether a specific field value has already been loaded into the field in the data load

script. The function returns TRUE or FALSE, so can be used in the **where** clause of a **LOAD** statement or an **IF** statement.

**Syntax:**

Exists(field [, expression ] )

Start Schema And Snowflake Schema:

* In star schema each dimension is represented by a single dimension table, But in Snowflake

|  |  |  |
| --- | --- | --- |
| ****Description**** | ****Star schema**** | ****Snowflake schema**** |
| ****Data model**** | Top-down approach | Bottom-up approach |
| ****Normalization/ Denormalization**** | The fact table and Dimension tables are in the Denormalized form. | The fact tables are in Denormalized form, but Dimension tables are in normalized form. |
| ****Ease of use**** | Easy to understand and low query rate. | It is a complex structure and not as easy to understand. |
| ****Ease of maintenance**** | It has redundant data and is less easy to maintain and change. | No redundancy. Hence, Snowflake is easier to maintain and change. |
| ****Dimension table**** | It contains only a single dimension table for each dimension. | It contains more than one dimension table for each dimension depending on the data. |
| ****Query performance**** | Less number of foreign keys and takes less time for execution. | More foreign keys and takes a long time for execution. |
| ****Joins**** | Less number of joins | More number of joins |
| ****Application**** | We can prefer the star schema when the dimension table has fewer rows. | We can prefer the snowflake schema when the dimension table is relatively big. So, this schema helps to reduce the size of the data. |

**DISTINCT**

Distinct keyword will eliminate the duplicate records...

Field:

A

A

B

count(field) will return 3 because you have 3 values in field

count(distinct field) will return 2 because A will be counted as 1

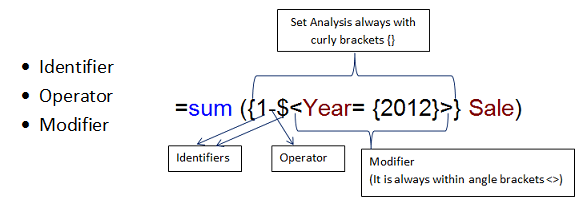
**Day 4**

**Set Analysis**

Set analysis predefines the SET OF DATA that our charts / tables use. So, using a Set Expression, we can tell our object (chart / table) to display values corresponding to various sets of data.

Some features and characteristics for Set analysis are:

* It is used to create different selection compared to the current application selections
* Must be used in aggregation function (Sum, Count….).
* Expression always begins and ends with curly brackets { }



|  |  |
| --- | --- |
| **Identifier** | **Description** |
| 0 | :Represents an empty set, no records |
| 1 | :Represents the set of all the records in the application |
| $ | :Represents the records of the current selection |
| $1 | :Represents the previous selection |

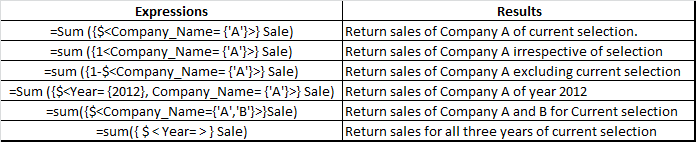
**Operator**

* It  works on set identifiers

|  |  |  |
| --- | --- | --- |
| **Operator** | **Operator Name** | **Description** |
| + | Union | Returns a set of records that belongs to union of sets. |
| – | Exclusion | Returns records that belong to the first but not the second |
| \* | Intersection | Returns records that belong to both of the set identifiers. |
| / | Symmetric Difference | Returns a set that belongs to either, but not both of the set identifiers. |

**Modifiers:**

* Modifiers are always in angle brackets <>.
* It consists multiple fields and all fields have selection criteria.
* Condition of fields within modifiers bypass the current selection criteria.



.

# **Mapping:**

The mapping prefix is used to create a mapping table that can be used to, for example, replacing field values and field names during script execution.

**Syntax:**

**Mapping**( loadstatement | selectstatement )

The **mapping** prefix can be put in front of a **LOAD.**

### **1.ApplyMap() function:**

We use this function to map or mirror the output of an expression, that evaluates during script execution, in a previously loaded table.

**Syntax:**

ApplyMap('map\_name', expression [,default\_mapping])

Where *map\_name* is the name of the table from which we want to map the fields or field values into a new table.

The *default\_mapping* is a conditional parameter which specifies the value that we use when the table evaluates for the expression condition, does not contain a certain value.

### **2.MapSubstring() function**

The MapSubstring() function maps a substring or a part of a data field from a previously loaded data table (using mapping LOAD/SELECT statements) to another table. This function is case-sensitive and returns a string type data value.

1. **Nested function**

You want to calculate the sum of the field Sales, but only include transactions with an OrderDate equal to the last year. The last year can be obtained via the aggregation function Max(TOTAL Year(OrderDate)).

The following aggregation would return the desired result:

Sum(If(Year(OrderDate)=Max(TOTAL Year(OrderDate)), Sales))

1. **Fiscal Calendar**

A fiscal year is a 12-month accounting period that a business uses for financial and tax reporting purposes. A fiscal year is also known as a financial year. A fiscal year can be different to a calendar year – it doesn't need to start on January 1 and end on December 31

Fiscal years also always end on the last day of the month, unless it is December (in which case it would simply be a calendar year).

Syntax:

Load\*,

Year(addmonths(Filedname,-6)) as [Fiscal year]

1. **DrillDown Dimensions**

When adding a dimension, you can select between creating a single or a drill-down dimension.

1.Select Drill-down as dimension type.

2.Click at least two fields from the fields list on the left-hand side to insert them as the referenced fields.

3.Type a name for the dimension.

Type a description for the dimension (optional).

If you want to specify a color, click IMG_256 in the color drop down and select a color through one of the following methods:

4.Click one of the colors in the palette.

5.Type a 6 character color code in the Hex input field: #.

6.Click IMG_257 at the bottom of the dialog, select a color in the color wheel, and optionally adjust the saturation slider.

Add tags (optional).

7.Click Create.

8.Click Done to close the dialog.

**You can create a default bookmark to handle this default selections:**

By default your charts will reflect your filter selections, so nothing additional needed.

What values do you have in Quarter?  is it QuarterName?  I'll assume it is, but then you wouldn't need to select also a year.

In your Quarter filterpane object, search:

=quarter\_field =quartername(max(total quarter\_field, 2))

(hit enter)

In your Year filterpane object, search:

=year\_field =year(max(total year\_field, 2))

(hit enter)

Create bookmark.

Right click bookmark and select as default.

**1.Current year sales,**

   Sum({$<Year={$(=Max(Year))}>}[Sales Amount])

**Last year sales amount**

Sum({$<Year={$(=Max(Year)-1)}>}[Sales Amount])

**2.Current month sales,**

pick(vcurrentyear, sum({$<Month={'$(=Month(Max(Year)-1))'}>} Sales ),

sum({$<Month ={'$(=Month(Max(Year)-1))'}>} [Sales Qty] )

**3.Current Quarter Sales,**

pick(vcurrentyear,Sum({<Quarter = {"$(=Dual('Q' & Ceil(Month(Max(Date))/3) & '-' & Year(Max(Date)),

QuarterStart(Max(Date))))"}>}Sales),

sum({<Quarter={"$(=Dual('Q' & Ceil(Month(Max(Date))/3) & '-' & Year(Max(Date)),

QuarterStart(Max(Date))))"}>}[Sales Qty]))