**Basics of Qlik Sense Scripting:**

Data Loading:

In Qlik Sense, you load data from various data sources, including Excel files, CSV files, databases, and more. You use the LOAD statement to load data into tables. For example:

**MyTable:**

**LOAD**

**Field1,**

**Field2**

**FROM [DataSourcePath];**

This code loads data from a data source (specified in [DataSourcePath]) into a table named MyTable.

Transformations:

You can perform various data transformations on loaded data. Common transformations include renaming fields, deriving new fields, and aggregating data. Here's an example that renames fields and creates a calculated field:

**MyTable:**

**LOAD**

**OldFieldName1 AS NewFieldName1,**

**OldFieldName2 AS NewFieldName2,**

**Field1 + Field2 AS SumField**

**FROM [DataSourcePath];**

Data Concatenation:

You can concatenate tables using the CONCATENATE keyword to combine data from multiple sources or perform data joins:

**Concatenate**

**LOAD**

**FieldA,**

**FieldB**

**FROM [AnotherDataSource];**

Data Filtering:

Use the WHERE clause to filter data while loading it. For example:

**MyTable:**

**LOAD**

**Field1,**

**Field2**

**FROM [DataSourcePath]**

**WHERE Field3 = 'Value';**

This loads only the rows where Field3 equals 'Value' into MyTable.

Data Model:

Qlik Sense automatically builds a data model based on the tables you've loaded. This model defines how different fields are related to each other. You can create associations, explicitly defining how tables are linked using the JOIN keyword.

Script Variables:

You can define script variables for dynamic control of data loading or calculations:

**SET vStartDate = '2023-01-01';**

**SET vEndDate = '2023-12-31';**

You can then use these variables in your script for dynamic filtering or calculations.

Comments:

You can add comments in your script using // for single-line comments or /\* ... \*/ for multi-line comments:

**// This is a single-line comment**

**/\***

**This is a multi-line comment.**

**It can span multiple lines.**

**\*/**

**Operators**

**Arithmetic Operators:**

Addition (+): Used to add two or more values.

TotalSales + Profit

Subtraction (-): Used to subtract one value from another.

TotalExpenses - Costs

Multiplication (\*): Used to multiply values.

Quantity \* UnitPrice

Division (/): Used to divide one value by another.

Revenue / Customers

Modulus (%): Returns the remainder after division.

Orders % 5

**Comparison Operators:**

Equal to (==): Used to compare two values for equality.

Country == 'USA'

Not equal to (!= or <>): Used to check if two values are not equal.

Product <> 'Widget'

Greater than (>): Checks if one value is greater than another.

Sales > 1000

Less than (<): Checks if one value is less than another.

Age < 30

Greater than or equal to (>=): Checks if one value is greater than or equal to another.

Score >= 90

Less than or equal to (<=): Checks if one value is less than or equal to another.

Price <= 50

**Logical Operators:**

AND (&&): Used to combine two or more conditions. Returns true if all conditions are true.

(Region = 'North' && Sales > 1000)

OR (||): Used to combine two or more conditions. Returns true if at least one condition is true.

(Category = 'Electronics' || Category = 'Clothing')

NOT (!): Negates a condition.

NOT (Discount > 0.1)

**String Concatenation Operator:**

Concatenation (& or +): Used to concatenate strings.

'First Name: ' & FirstName

**Null Operator:**

Null (IsNull()): Used to check for null values.

IsNull(Sales)

**More Functions**

**Data Conversion:**

Date(): Converts a numeric or text representation of a date into a Qlik Sense date format.

Date('2023-10-15', 'YYYY-MM-DD')

Num(): Converts a value to a numeric representation. It can be used to convert numbers stored as text to numeric values.

Num('12345.67')

Text(): Converts a value to a text representation.

Text(123.45)

Timestamp(): Converts a date and time value to a Qlik Sense timestamp format.

Timestamp('2023-10-15 12:30:00', 'YYYY-MM-DD hh:mm:ss')

Interval(): Converts a numeric value into an interval format. Useful for dealing with time durations.

Interval(3600, 'hh:mm:ss')

This function converts the numeric value 3600 into a time interval in hours, minutes, and seconds.

Dual(): Creates a dual value, which associates both a numeric and text interpretation with a single value.

Dual('Green', 1)

This function creates a dual value 'Green' with a numeric value of 1.

ToNum(): Converts a field or value to a numeric format. This function is often used to explicitly convert data types within load scripts.

ToNum(Quantity)

This function converts the Quantity field to a numeric format.

ToText(): Converts a field or value to a text format. It's used to ensure data is treated as text when needed.

ToText(CustomerID)

This function converts the CustomerID field to a text format.

DualValue(): Converts a value to a dual value, allowing you to associate both a numeric and text representation.

DualValue(42, 'Answer to the Ultimate Question')

This function creates a dual value with both numeric and text representations.

**String Functions:**

Concat(): Combines multiple strings into a single string, with an optional delimiter.

Concat(Product, ', ')

This function concatenates values in the Product field, separated by a comma and space.

Replace(): Replaces a substring with another substring within a string.

Replace(Description, 'old', 'new')

This function replaces occurrences of 'old' with 'new' in the Description field.

SubFieldCount(): Counts the number of occurrences of a substring within a string.

SubFieldCount(Ingredients, ', ')

This function counts the number of ingredients in a list separated by commas and spaces.

Left() and Right(): Extracts a specified number of characters from the left or right side of a string.

Left(Name, 3) // Extracts the first 3 characters of the Name field

Right(PostalCode, 5) // Extracts the rightmost 5 characters of the PostalCode field

Mid(): Extracts a substring from a string, starting at a specified position and extending for a specified number of characters.

Mid(Text, 3, 5) // Extracts 5 characters from the 3rd position in the Text field

Upper() and Lower(): Convert text to uppercase or lowercase, respectively.

Upper(Title) // Converts the Title field to uppercase

Lower(Description) // Converts the Description field to lowercase

Trim(): Removes leading and trailing spaces from a string.

Trim(Whitespace) // Removes leading and trailing spaces from the Whitespace field

Len(): Returns the length of a string.

Len(Comment) // Returns the number of characters in the Comment field

WildMatch(): Tests if a string matches a wildcard expression. Useful for pattern matching.

WildMatch(Name, 'J\*n')

This function checks if the Name field matches the pattern 'J\*n', which would match 'John' and 'Jan' but not 'Robert'.

**Number Functions:**

Round(): Rounds a number to a specified number of decimal places.

Round(1234.5678, 2) // Returns 1234.57

Floor() and Ceil(): Round a number down or up to the nearest integer, respectively.

Floor(5.8) // Returns 5

Ceil(5.1) // Returns 6

RangeSum(): Calculates the sum of values within a range.

RangeSum(1, 2, 3, 4, 5) // Returns 15

**Conditional Statements:**

IF(): Returns one value if a condition is true and another if it's false.

IF(Sales > 1000, 'High', 'Low')

Switch(): Evaluates a series of conditions and returns the result of the first true condition.

Switch(Category = 'A', 'Category A', Category = 'B', 'Category B', 'Other')

**Loops:**

FOR Loop: You can create a loop by setting a script variable and then using it in your script to iterate through values.

FOR i = 1 TO 10

MyTable:

LOAD

Field1,

Field2

FROM [DataSourcePath$i];

NEXT

WHILE Loop: You can create a loop with a WHILE statement, controlling it based on script variables or field values.

SET vCounter = 1;

WHILE $(vCounter) <= 10

MyTable:

LOAD

Field1,

Field2

FROM [DataSourcePath$(vCounter)];

SET vCounter = $(vCounter) + 1;

WEND

**Aggregate Functions**

SUM(): Calculates the sum of a numeric field.

SUM(Sales)

COUNT(): Counts the number of records in a field.

COUNT(CustomerID)

AVG(): Calculates the average of a numeric field.

AVG(Profit)

MIN(): Returns the minimum value in a numeric field.

MIN(Price)

MAX(): Returns the maximum value in a numeric field.

MAX(Temperature)

RANGE functions:

- RANGE functions allow you to perform custom range aggregations, where you can specify the boundaries of the ranges.

- RANGESUM(): Calculates the sum of values within specified ranges.

RANGESUM(0, 100, Sales) // Sum of Sales between 0 and 100

- RANGEAVE(): Calculates the average of values within specified ranges.

RANGEAVE(0, 100, Age) // Average Age between 0 and 100

- RANGECOUNT(): Counts the number of values within specified ranges.

RANGECOUNT(10, 20, Price) // Count of Prices between 10 and 20

AGGR(): Creates aggregated expressions, allowing you to perform calculations based on a set of dimensions. This is particularly useful for creating calculated dimensions or aggregations not covered by standard functions.

AGGR(SUM(Sales), Customer)

Percentile(): Calculates the value at a specified percentile for a given field. You can specify the desired percentile as a parameter (e.g., 25 for the 25th percentile).

Percentile(25, Sales) // 25th percentile of Sales

FirstSortedValue(): Returns the first sorted value of a field based on a sorting expression. This is useful when you want to retrieve the top or bottom values based on a specific sort order.

FirstSortedValue(Product, -Sales) // Product with the highest Sales

TopN() and BottomN(): These functions allow you to find the top or bottom N values in a field, based on a specified measure.

TopN(5, Country, Sales) // Top 5 Countries by Sales

BottomN(3, Product, Profit) // Bottom 3 Products by Profit

**Grouping and Aggregating by Dimensions:**

In Qlik Sense, you often aggregate data based on one or more dimensions. For example, you can calculate the total sales by product category or by year. You can use the GROUP BY clause to achieve this:

LOAD

Product,

Category,

Year,

SUM(Sales) as TotalSales

GROUP BY

Product,

Category,

Year;

This script loads data and groups it by product, category, and year while calculating the total sales for each combination.

**Set Analysis with Aggregations:**

Set Analysis is a powerful feature in Qlik Sense that allows you to perform calculations based on specific selections made by users. You can use it with aggregate functions to create complex expressions that respond to user interactions.

For example, you can use Set Analysis to calculate the sales for a selected year, irrespective of other selections:

SUM({<Year={'2023'}>} Sales)

This expression sums the sales for the year 2023, regardless of other selections.

**Aggregating Over Time Periods:**

You can perform time-based aggregations by using Qlik Sense's built-in date and time functions. For instance, you can calculate the rolling 12-month sales:

LOAD

Date,

SUM(Sales) as Rolling12MonthSales

RESIDENT YourTable

ORDER BY Date;

In this example, the data is ordered by date, and the script calculates the sum of sales for each date, effectively creating a rolling 12-month aggregation.

**Aggregation Functions in Charts:**

When creating visualizations, you can apply aggregation functions directly within chart expressions. For instance, you can calculate the average sales for each product category in a bar chart by using AVG(Sales) as an expression.

**Advanced Set Analysis:**

You can create complex Set Analysis expressions with aggregate functions to define more advanced selections. This allows you to build custom aggregations that respond to user interactions. For instance, you can calculate the sum of sales for the selected products and the previous year's sales:

SUM({<Product=P(Product), Year=P(Year-1)>} Sales)

Here, P(Product) and P(Year-1) represent the products and the previous year selected by the user.

**Nested Aggregations:**

In some cases, you might need to perform aggregations within aggregations. This can be achieved by using aggregation functions within aggregation functions, allowing you to calculate more complex measures.

AVG(AGGR(SUM(Sales), Product, Year))

This expression calculates the average sales across all products and years.

**Descriptive Statistical Functions**

In Qlik Sense, descriptive statistical functions allow you to analyze and summarize data, providing insights into the distribution and characteristics of your data. These functions are often used to calculate common statistics such as mean, median, standard deviation, and more. Here are some key descriptive statistical functions in Qlik Sense:

AVG(): Calculates the average (mean) of a numeric field.

AVG(Sales)

SUM(): Calculates the sum of a numeric field.

SUM(Revenue)

MIN(): Returns the minimum value in a numeric field.

MIN(Temperature)

MAX(): Returns the maximum value in a numeric field.

MAX(Salary)

MEDIAN(): Calculates the median of a numeric field. The median is the middle value of a sorted dataset.

MEDIAN(Age)

RANGE(): Calculates the range of a numeric field, which is the difference between the maximum and minimum values.

RANGE(Price)

STDEV(): Calculates the sample standard deviation of a numeric field. Standard deviation measures the dispersion or variability in data.

STDEV(Population)

VARIANCE(): Calculates the sample variance of a numeric field. Variance measures the average squared difference from the mean.

VARIANCE(Sales)

COUNT(): Counts the number of records in a field.

COUNT(CustomerID)

COUNTD(): Counts the distinct values in a field. Useful for calculating the number of unique items.

COUNTD(Product)

Q1() and Q3(): Calculate the first quartile (25th percentile) and third quartile (75th percentile) of a numeric field, respectively.

Q1(Age)

Q3(Income)

CORRELATION(): Calculates the correlation coefficient between two numeric fields, which measures the strength and direction of the linear relationship between the fields.

CORRELATION(Price, Quantity)

PERCENTILE(): Calculates the value at a specified percentile for a given field. You can specify the desired percentile as a parameter (e.g., 25 for the 25th percentile).

PERCENTILE(25, Sales) // 25th percentile of Sales

RANK(): Assigns a rank to each value in a field based on a specified sorting order. You can use this function to identify the rank of values in your data.

RANK(Sales, 1, 1) // Rank of Sales, ascending order

**Filtering and Sorting**

In Qlik Sense scripting, filtering and sorting are important operations that help you prepare your data for visualization and analysis. Here's how to perform filtering and sorting in your Qlik Sense script:

**Filtering Data:**

MyTable:

LOAD

Field1,

Field2

FROM [DataSourcePath]

WHERE Field3 = 'Value';

In this example, only the rows where Field3 is equal to 'Value' will be loaded into the MyTable table.

You can use various comparison operators such as =, >, <, >=, and <=, as well as logical operators like AND and OR to create complex filter conditions.

**Sorting Data:**

Sorting data in Qlik Sense script allows you to control the order in which data is loaded. You can use the ORDER BY clause to specify the sorting order. Here's how to sort data:

MyTable:

LOAD

Field1,

Field2

FROM [DataSourcePath]

ORDER BY Field1 ASC, Field2 DESC;

In this example, data will be sorted by Field1 in ascending (ASC) order and then by Field2 in descending (DESC) order. You can use ASC for ascending order and DESC for descending order. You can also sort by multiple fields.

**Removing Duplicates:**

If you want to remove duplicate rows from your data, you can use the DISTINCT keyword in your script. Here's how to do it:

MyTable:

LOAD DISTINCT

Field1,

Field2

FROM [DataSourcePath];

This will load only distinct combinations of Field1 and Field2, eliminating duplicate rows.

**Renaming and Reordering Fields:**

You can rename and reorder fields during the loading process in your script. Use the AS keyword for renaming and reordering fields as needed.

MyTable:

LOAD

Field1 AS NewField1,

Field2 AS NewField2,

Field3

FROM [DataSourcePath];

This code renames Field1 to NewField1, Field2 to NewField2, and keeps Field3 as is in the MyTable table.

**Limiting the Number of Rows:**

If you want to limit the number of rows loaded from your data source, you can use the TOP keyword with the LOAD statement.

MyTable:

LOAD TOP 100

Field1,

Field2

FROM [DataSourcePath];

This code loads only the top 100 rows from the data source.

**Merging and Concatenating Data Sets**

**Merging Datasets:**

Merging data involves combining rows from multiple tables based on a common key or field. Qlik Sense provides the `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN`, and `FULL OUTER JOIN` operations to merge data. Here's how to perform an `INNER JOIN`:

MasterData:

LOAD

CustomerID,

CustomerName

FROM [MasterData.csv];

TransactionData:

LOAD

CustomerID,

TransactionAmount

FROM [TransactionData.csv];

MergedData:

LOAD

CustomerID,

CustomerName,

TransactionAmount

RESIDENT MasterData;

INNER JOIN

LOAD

CustomerID,

TransactionAmount

RESIDENT TransactionData;

In this example, the `INNER JOIN` combines data from `MasterData` and `TransactionData` based on the common field `CustomerID`. The resulting `MergedData` table contains the combined data.

**Concatenating Datasets:**

Concatenation is used to stack tables on top of each other, one below the other, to create a single table. Use the `CONCATENATE` keyword to concatenate tables:

Sales2019:

LOAD

Date,

Sales

FROM [SalesData2019.csv];

Sales2020:

LOAD

Date,

Sales

FROM [SalesData2020.csv];

AllSales:

CONCATENATE

LOAD

Date,

Sales

RESIDENT Sales2019;

CONCATENATE

LOAD

Date,

Sales

RESIDENT Sales2020;

In this example, `AllSales` contains all the sales data from both `Sales2019` and `Sales2020`.

**Joining Datasets:**

Joining data is similar to merging, but it combines data from multiple tables based on common keys, typically creating a larger, more detailed dataset. Here's how to perform an `INNER JOIN` to join two tables:

Customers:

LOAD

CustomerID,

CustomerName

FROM [Customers.csv];

Orders:

LOAD

OrderID,

CustomerID,

OrderDate,

OrderTotal

FROM [Orders.csv];

JoinedData:

LOAD

CustomerID,

CustomerName,

OrderID,

OrderDate,

OrderTotal

RESIDENT Customers;

INNER JOIN

LOAD

CustomerID,

OrderID,

OrderDate,

OrderTotal

RESIDENT Orders;

In this example, the `INNER JOIN` combines the `Customers` and `Orders` tables based on the common field `CustomerID` to create the `JoinedData` table with a more detailed view of customer orders.

You can use different types of joins (`LEFT JOIN`, `RIGHT JOIN`, `FULL OUTER JOIN`) and specify multiple join conditions as needed for your data integration.