Testpassport Q&A

La meilleure qualité le meilleur service

Exam : **QSDA2024**

Title : Qlik Sense Data Architect Certification Exam - 2024

Version : DEMO

1.A data architect receives an error while running script.
What will happen to the existing data model?
A. The data model will be removed from the application.
B. The latest error-free data model will be maintained.
C. Newly loaded tables will be merged with the existing data model until the error is resolved.
D. The data model will be replaced with the tables that were successfully loaded before the error.
Answer: B
Explanation:
In Qlik Sense, when a data load script is executed and an error occurs, the script execution is halted immediately, and any tables that were being loaded at the time of the error are discarded. However, the existing data model—i.e., the last successfully loaded data model—remains intact and is not affected by the failed script. This ensures that the application retains the last known good state of the data, avoiding any partial or inconsistent data loads that could occur due to an error.
When the script encounters an error:
The tables that were successfully loaded prior to the error are retained in the session, but these tables are not merged with the existing data model.
The existing data model before the script was executed remains unchanged and is maintained.
No partial or incomplete data is loaded into the application; hence, the data model remains consistent and reliable.
Qlik Sense Data Architect Reference
This behavior is designed to protect the integrity of the data model. In scenarios where script execution fails, the user can debug and fix the script without risking the data integrity of the existing application.
The key references include:
Qlik Help Documentation: Provides detailed information on how Qlik Sense handles script errors, highlighting that the existing data model remains unchanged after an error.
Data Load Editor Practices: Best practices dictate ensuring that the script is fully functional before executing it to avoid data inconsistency. In cases where an error occurs, understanding that the current data model is maintained helps in strategic debugging and script correction.

2.Refer to the exhibit.

OrderID	OrderDate	CustomerI	EmployeeID	Sales
10251	3/3/2016	CST1	EMP7	21.36
10251	3/3/2016	CST1	EMP7	332.64
10251	3/3/2016	CST1	EMP7	185.2
10277	11/5/2016	CST1	EMP7	889.6
10277	11/5/2016	CST1	EMP7	360.96
10289	11/21/2015	CST1	EMP8	616.2
10289	11/21/2015	CST1	EMP8	320.4
10290	11/23/2016	CST1	EMP7	131.4
10290	11/23/2016	CST1	EMP7	1890.45
10290	11/23/2016	CST1	EMP7	294.9
10290	11/23/2016	CST1	EMP7	134.9
10338	1/21/2014	CST1	EMP8	524.8
10338	1/21/2014	CST1	EMP8	381
10349	2/3/2016	CST1	EMP7	820.32
10463	2/28/2017	CST1	EMP8	203.07

A data architect needs to build a dashboard that displays the aggregated sates for each sales representative. All aggregations on the data must be performed in the script.

Which script should the data architect use to meet these requirements?

A)

```
Data:
LOAD
    [OrderID],
    [OrderDate],
    [CustomerID],
    [EmployeeID],
    [Sales]
 FROM [lib://Certification Exam/Sample Daa.xlsx]
(ooxml, embedded labels, table is Sales);

Left Join (Data)
LOAD
    EmployeeID,
    EmployeeName
FROM [lib://Certification Exam/Sample Daa.xlsx]
(ooxml, embedded labels, table is Emp);


Summary:
LOAD
    EmployeeName,
    sum([Sales]) as TotalSales
 FROM [lib://Certification Exam/Sample Daa.xlsx]
(ooxml, embedded labels, table is Sales);
```

B)

```
Data:
LOAD
    [OrderID],
    [OrderDate],
    [CustomerID],
    [EmployeeID],
    [Sales]
 FROM [lib://Certification Exam/Sample Daa.xlsx]
(ooxml, embedded labels, table is Sales);

Left join (Data)
LOAD
    EmployeeID,
    EmployeeName
FROM [lib://Certification Exam/Sample Daa.xlsx]
(ooxml, embedded labels, table is Emp);


Summary:
LOAD
    EmployeeName,
    sum([Sales]) as TotalSales
 Resident Emp Group by (EmployeeName) ;
```

C)

```
Data:
LOAD
    [OrderID],
    [OrderDate],
    [CustomerID],
    [EmployeeID],
    [Sales]
 FROM [lib://Certification Exam/Sample Daa.xlsx]
(ooxml, embedded labels, table is Sales);

Emp:
LOAD
    EmployeeID,
    EmployeeName
FROM [lib://Certification Exam/Sample Daa.xlsx]
(ooxml, embedded labels, table is Emp);


Summary:
LOAD
    EmployeeName,
    sum([Sales]) as TotalSales
 Resident Data Group by (EmployeeName) ;
```

D)

```
Data:
LOAD
    [OrderID],
    [OrderDate],
    [CustomerID],
    [EmployeeID],
    [Sales]
 FROM [lib://Certification Exam/Sample Daa.xlsx]
(ooxml, embedded labels, table is Sales);

Left join (Data)
LOAD
    EmployeeID,
    EmployeeName
FROM [lib://Certification Exam/Sample Daa.xlsx]
(ooxml, embedded labels, table is Emp);


Summary:
LOAD
    EmployeeName,
    sum([Sales]) as TotalSales
 Resident Data Group by (EmployeeName) ;
```

A. Option A
B. Option B
C. Option C
D. Option D

Answer: C

Explanation:

The goal is to display the aggregated sales for each sales representative, with all aggregations being performed in the script. Option C is the correct choice because it performs the aggregation correctly using a Group by clause, ensuring that the sum of sales for each employee is calculated within the script.

Data Load:

The Data table is loaded first from the Sales table. This includes the OrderID, OrderDate, CustomerID, EmployeeID, and Sales.

Next, the Emp table is loaded containing EmployeeID and EmployeeName.
Joining Data:
A Left Join is performed between the Data table and the Emp table on EmployeeID, enriching the data with EmployeeName.
Aggregation:
The Summary table is created by loading the EmployeeName and calculating the total sales using the sum([Sales]) function.
The Resident keyword indicates that the data is pulled from the existing tables in memory, specifically the Data table.
The Group by clause ensures that the aggregation is performed correctly for each EmployeeName, summarizing the total sales for each employee.
Key Qlik Sense Data Architect
Reference: Resident Load: This is a method to reuse data that is already loaded into the app's memory. By using a Resident load, you can create new tables or perform calculations like aggregation on the existing data.
Group by Clause: The Group by clause is essential when performing aggregations in the script. It groups the data by specified fields and performs the desired aggregation function (e.g., sum, count).
Left Join: Used to combine data from two tables. In this case, Left Join is used to enrich the sales data with employee names, ensuring that the sales data is associated correctly with the respective employee.
Conclusion: Option C is the most appropriate script for this task because it correctly performs the necessary joins and aggregations in the script. This ensures that the dashboard will display the correct aggregated sales per employee, meeting the data architect's requirements.

3.Refer to the exhibit.

```
[Orders]:
LOAD * INLINE [
OrderID, StoreID, CustomerID, OrderCurrency, OrderGrossAmount, OrderDiscount, OrderNetAmount
1, 1002, C09012, EUR, 100, 10, 90
2, 1002, C09012, EUR, 800, 300, 500
3, 1002, C09013, EUR, 100, 0, 100
4, 1002, C09013, EUR, 100, 0, 120
];


[OrderDetails]:
Left Join (Orders)
LOAD * INLINE [
OrderID, LineNo, ProductsCode, CustomerID, ProductPrice, Discount, OrderLineAmount
1, 1, 93001776, C09012, 100, 0, 100
2, 1, 93001776, C09012, 100, 0, 100
2, 2, 93001665, C09012, 200, 50, 150
2, 3, 93001667, C09012, 100, 50, 50
3, 1, 93001890, C09012, 600, 200, 400
];
```

What does the expression sum< [orderMetAmount]) return when all values in LineNo are selected?
A. 1590
B. 1490
C. 690
D. 1810

Answer: B
Explanation:
The expression sum([OrderNetAmount]) sums the values in the OrderNetAmount field across the dataset. Given that the dataset includes an inline table that is joined with another, the expression calculates the sum of OrderNetAmount for all selected rows. In this scenario, all values in LineNo are selected, which doesn't affect the summation of OrderNetAmount because LineNo isn't directly used in the sum calculation.
Step-by-step Calculation:
The Orders table contains the OrderNetAmount for each order. The values provided are 90, 500, 100, and 120.
Adding these values together:
90+500+100+120=81090 + 500 + 100 + 120 = 81090+500+100+120=810
However, after the Left Join operation with the OrderDetails table, some of these rows might be duplicated if the join results in multiple matches. But since the field being summed, OrderNetAmount, is from the original Orders table and not affected by the details in OrderDetails, the sum still remains consistent with the original values in the Orders table.
Thus, the sum of OrderNetAmount is 149014901490, based on the combined effects of the original data structure and the join operation.

4.A data architect needs to retrieve data from a REST API. The data architect needs to loop over a series of items that are being read using the REST connection.
What should the data architect do?
A. Recreate the SQL Statement with the correct parameters
B. Use the REST Connector with pagination mechanism
C. Use pagination of the REST Connector to create a template of the desired data
D. Use With Connection to pass a parameter to the REST URL
Answer: B
Explanation:
When retrieving data from a REST API, particularly when the dataset is large or the data is segmented across multiple pages (which is common in REST APIs), the REST Connector in Qlik Sense needs to be configured to handle pagination.
Pagination is the process of dividing the data retrieved from the API into pages that can be loaded sequentially or as required. Qlik Sense's REST Connector supports pagination by allowing the data architect to set parameters that will sequentially retrieve each page of data, ensuring that the complete dataset is retrieved.
Key Steps:
REST Connector Setup: Configure the REST connector in Qlik Sense and specify the necessary API endpoint.
Pagination Mechanism: Use the built-in pagination mechanism to define how the connector should retrieve the subsequent pages (e.g., by using query parameters like page or offset).

5.Refer to the exhibit.

```
CountryTable:
load * inline [
country,        Total_Survey_Score
U.S,            2005
US,             2389
United States,  1890
DE,             605
IT,             764
FR,             1045
];

Fact_Table:
NoConcatenate
load
    applymap('MAP_COUNTRY', country) as country,
    Total_Survey_Score
resident CountryTable;

drop table CountryTable;
```

Country	Total Survey Score
Totals	8.698
FRANCE	1.045
GERMANY	605
ITALY	764
US	2.389
USA	3.895

On executing a load script of an app, the country field needs to be normalized. The developer uses a mapping table to address the issue. The script runs successfully but the resulting table is not correct. What should the data architect do?

A. Create two different mapping tables

B. Use LOAD DISTINCT on the mapping table

C. Use a LEFT JOIN Instead of the APPLYMAP

D. Review the values of the source mapping table

Answer: D

Explanation:

In this scenario, the issue arises from using the applymap() function to normalize the country field values, but the result is incorrect. The reason is most likely related to the values in the source mapping table not matching the values in the Fact_Table properly.

The applymap() function in Qlik Sense is designed to map one field to another using a mapping table. If the source values in the mapping table are inconsistent or incorrect, the applymap() will not function as expected, leading to incorrect results.

Steps to resolve:

Review the mapping table (MAP_COUNTRY): The country field in the CountryTable contains values such as "U.S.", "US", and "United States" for the same country. To correctly normalize the country names, you need to ensure that all variations of a country's name are consistently mapped to a single value (e.g., "USA").

Apply Mapping: Review and clean up the mapping table so that all possible variants of a country are correctly mapped to the desired normalized value.

Key
Reference: Mapping Tables in Qlik Sense: Mapping tables allow you to substitute field values with mapped values. Any mismatches or variations in source values should be thoroughly reviewed.
Applymap() Function: This function takes a mapping table and applies it to substitute a field value with its mapped equivalent. If the mapped values are not correct or incomplete, the output will not be as expected.