DUMPSARENA

Qlik Sense Data Architect Certification Exam 2024

Qlik QSDA2024

Version Demo

Total Demo Questions: 7

Total Premium Questions: 74

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, Data Modeling	21
Topic 2, Data Transformation	19
Topic 3, Data Loading	26
Topic 4, Data Governance	7
Topic 5, Mix Questions	1
Total	74

QUESTION NO: 1

A service-management app contains a Requests table with RequestID, CreatedDate, and ResolvedDate. Users must analyze request counts by a common calendar while distinguishing requests created during a period from requests resolved during a period.

A selection in the calendar must be able to apply to either date role without creating multiple calendar tables or ambiguous associations.

Which data-model approach should the data architect use?

A. Create separate master calendars for CreatedDate and ResolvedDate

B. Join CreatedDate and ResolvedDate into one field in the Requests table

C. Create a canonical date link table with RequestID, CanonicalDate, and DateType

D. Rename CreatedDate and ResolvedDate to CalendarDate in the Requests table

ANSWER: C

Explanation:

Create a canonical date link table containing RequestID, a canonical date field, and a date-type field such as Created or Resolved. The canonical date field associates with one master calendar, while the date-type field identifies the business role represented by each link-table record.

This design allows users to select a calendar period and then distinguish CreatedDate from ResolvedDate through the date-type field. Leaving both date fields associated directly to one calendar creates multiple associations between the same tables and can result in circular references or ambiguous analysis. Separate calendar tables make common time analysis more difficult.

QUESTION NO: 2

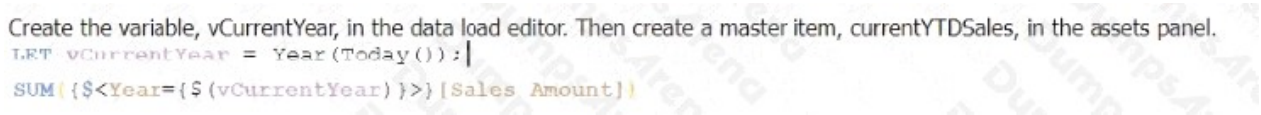
Sales managers need to see an overview of historical performance and highlight the current year's metrics. The app has the following requirements:

• Display the current year's total sales

• Total sales displayed must respond to the user's selections

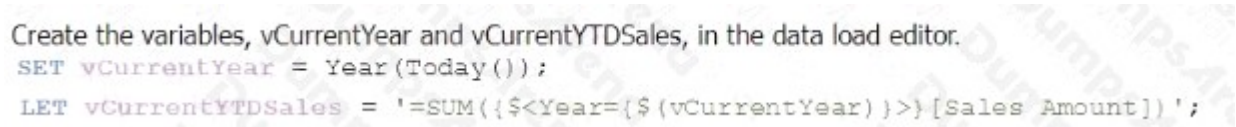
Which variables should a data architect create to meet these requirements?

A)

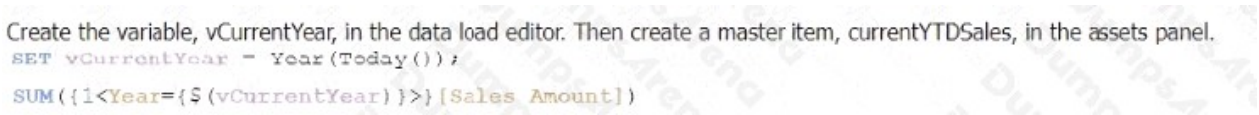


B)

C)



D)



A. Option A

B. Option B

C. Option C

D. Option D

ANSWER: C

Explanation:

Option C is correct because it defines a reusable current-year value and a sales-measure expression that preserves the user's active selection state. The current-year variable is based on `Year(Today())`, so its value is derived from the date when the reload executes rather than being manually maintained. This lets the app's definition move to a new calendar year on a subsequent reload.

The total-sales variable uses `Sum()` together with set analysis targeting the current year. Its set identifier is `$`, which represents the current selection state. Consequently, selections on associated fields—such as region, product, customer, or sales representative—remain in effect while the measure restricts the calculation to records for the current year. The variable can then be expanded in a KPI or chart expression, providing a consistent current-year sales calculation throughout the app.

Qlik documents that `SET` assigns script text to a variable and `LET` assigns an evaluated expression result; these statements support the required variable definitions. Qlik also documents that the `$` set identifier represents the current selection state, which is essential for a measure that must respond to user selections. See Qlik SET statement documentation and Qlik set analysis documentation.

QUESTION NO: 3

A Customer table has a Region field that is NULL for customers whose region has not yet been assigned. Business users need to select an explicit Unassigned value in a filter pane and analyze these customers separately from customers assigned to a region.

Which script setting should the data architect use before loading the Customer table?

A. NullAsValue Region;

B. NullInterpret Region;

C. Alt(Region,'Unassigned') AS Region

D. NullAsNull Region;

ANSWER: A

Explanation:

`NullAsValue Region;` is correct because it converts NULL values in the specified field into a selectable field value when the table is loaded. The displayed null representation can then be configured with `NullValue`, such as Unassigned.

NullInterpret is used when interpreting a specified text string as a NULL value, not when making existing NULLs selectable. Alt() can supply an alternative numeric expression but is not the appropriate general field-level method for representing NULL dimension values. NullAsNull preserves NULL behavior rather than creating a selectable value.

QUESTION NO: 4

Exhibit.

Dynamic dimension data

SPID	ChangeDate	Department
1		Dept B
2		Dept C
2	12/31/2011	Dept D

Static dimension data

SPID	Name
1	Bob
2	Cynthia

A large electronics company re-assigns sales people once per year from one Department to another.

SPID is the Salesperson ID; the SPID for each individual sales person Name remains constant. The Department for a SPID may change; each change is stored in the Dynamic Dimension data.

Four tables need to be linked correctly: a transaction table, a dynamic salesperson dimension, a static salesperson dimension, and a department dimension.

Which script prefix should the data architect use?

A. Merge

B. IntervalMatch

C. Partial Reload

D. Semantic

ANSWER: B

Explanation:

IntervalMatch is the appropriate script prefix because the department associated with a salesperson is effective for a defined period rather than permanently. Transaction records contain a discrete transaction date, while the dynamic salesperson data represents department-assignment intervals for the same stable salesperson identifier, SPID. IntervalMatch creates the associations needed to identify which effective-dated assignment applies when each transaction occurred.

In this model, the data architect can derive an end date for each assignment from the next recorded change date, then use IntervalMatch to match transaction dates to the corresponding start and end dates. Qlik also supports an extended IntervalMatch syntax using additional matching fields, which is important here: SPID can be included as the key so that a transaction is matched only to date intervals belonging to that specific salesperson. The resulting association preserves the salesperson's historical department at the time of sale, while the static salesperson dimension can continue to provide stable attributes such as the salesperson name and the department dimension can provide department details.

This is the standard Qlik load-script approach for associating discrete values, such as dates, with interval boundaries. See Qlik's documentation for IntervalMatch and its extended syntax with key fields.

QUESTION NO: 5

```
[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
];
```

Refer to the exhibit.

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:

1490 is the result returned when every `LineNo` value is selected. In Qlik Sense, a selection defines the current possible-record set through the associations in the data model. With all line numbers selected, all corresponding records represented by the exhibit remain available to the aggregation. The `Sum()` aggregation then adds the available numeric values of `orderMetAmount` at the data model's resulting record grain. This is important where order data has been associated or joined to line-level detail: an order-level value can occur once for each associated line record, and the aggregation reflects those available occurrences. Adding the resulting `orderMetAmount` values for the complete LineNo selection gives 1490. Qlik documents that `Sum()` calculates the total of an expression evaluated over the records included in the current aggregation context. The active selection state determines which records are possible, so selecting all LineNo values produces the full total shown in the exhibit. See Qlik's documentation for Sum() and for the associative selection model.

QUESTION NO: 6

A data architect loads a Sales table containing historical transactions. A new monthly source contains the same business grain and the same core fields, but it also contains a new field named PromotionCode.

The data architect must append the monthly records to Sales and retain PromotionCode for the new records. Historical records can contain null PromotionCode values.

Which script approach should the data architect use?

A. Use NoConcatenate when loading the monthly source

B. Use a Left Join from the monthly source to Sales

C. Use Concatenate (Sales) when loading the monthly source

D. Remove PromotionCode so that automatic concatenation occurs

ANSWER: C

Explanation:

Use an explicit `Concatenate (Sales)` prefix when loading the monthly source. Automatic concatenation does not occur when the field structures differ, but an explicit concatenate appends the records into Sales and creates PromotionCode as an additional field. Historical Sales records will have null values for that field.

A join would attempt to combine rows through common fields and could duplicate or alter transaction records. NoConcatenate would create a separate monthly table. Dropping PromotionCode would allow automatic concatenation only by discarding a field required for analysis.

QUESTION NO: 7

A data architect executes the following script:

```
Table_A:
LOAD * INLINE [
Field_1, Field_2, Field_3
01, AB, 10
01, AC, 50
02, AD, 75
];

Join(Table_A)
Table_B:
LOAD * INLINE [
Field_1, Field_4, Field_5
01, 30%, 500
03, 60%, 1000
];
```

What will be the result of Table.A?

A)

Preview of data

Field_1	Field_2	Field_3	Field_4	Field_5
01	AB	10	30%	500
01	AC	50	30%	500

B)

Preview of data

Field_1	Field_2	Field_3	Field_4	Field_5
01	AB	10	30%	500
01	AC	50	30%	500

C)

Preview of data

Field_1	Field_2	Field_3	Field_4	Field_5
01	AB	10	30%	500
01	AC	50	30%	500
02	AD	75	-	-
03	-	-	60%	1000

D)

Preview of data

Field_1	Field_2	Field_3	Field_4	Field_5
01	AB	10	30%	500
01	AC	50	30%	500
02	AD	75	-	-

A. OptionA

B. OptionB

C. OptionC

D. OptionD

ANSWER: D

Explanation:

The result shown in **OptionD** is correct because an unqualified `Join(Table_A)` in Qlik script performs an outer join. Qlik joins tables using every field name shared by the two tables as the join key; here, the shared field is `Field_1`. For the value `01`, the source table has two rows and the joined load has one matching row. Therefore, the values from the joined load are

appended to each of the two matching rows. The row with `Field_1 = 02` remains in the output, with null values for fields that originate only from the joined load, because no matching joined row exists. Likewise, the row with `Field_1 = 03` remains, with null values for fields that originate only from the original table, because an outer join retains nonmatching rows from both inputs. The resulting table consequently contains all matched combinations plus the unmatched row from each side. Qlik documents that `Join` without an explicit join type is an outer join and that joins are based on common field names. See the Qlik Join script prefix documentation and the Qlik documentation on combining tables with Join and Keep.