DUMPSBOSS.

QlikView 11 Developer Certification Examination (qv_developer_01)

Qlik QV-Developer-01

Version Demo

Total Demo Questions: 9

Total Premium Questions: 90

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Data Modeling	21
Topic 2, Data Loading	36
Topic 3, User Interface Design	28
Topic 4, Security	2
Topic 5, QlikView Server and Publisher	2
Topic 6, Deployment	1
Total	90

QUESTION NO: 1

Refer to the exhibit to the right.

```
1  QUALIFY *;
2
3  SalesPerson:
4  LOAD DepartmentID,
5       SalesPersonID,
6       Name;
7  SQL SELECT DepartmentID,
8       SalesPersonID,
9       Name
10 FROM dbo.SalesPerson;
11
12 Left Join
13 LOAD DepartmentID,
14      Name;
15 SQL SELECT DepartmentID,
16      Name
17 FROM dbo.Department;
```

When a customer reloads the script displayed in the exhibit, it fails to respond and complete, and displays an error message.

OBJECT OUT OF MEMORY.

How should the developer resolve the logic error in the script?

A. change the script to use an Inner Join instead of Left Join

B. change the QUALIFY * command to list out only the fields that need to be qualified

C. change the Left Join command to include the name of the table being joined enclosed in parentheses

D. change the DepartmentID field in the SalesPerson table to match the qualified field name of DepartmentID in the Department table

ANSWER: B

Explanation:

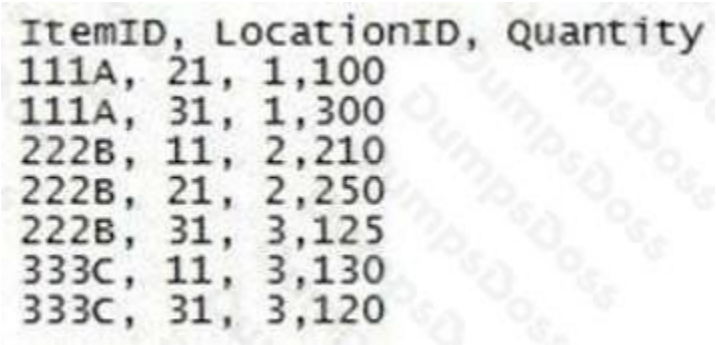
“change the QUALIFY * command to list out only the fields that need to be qualified” is correct because QlikView’s `QUALIFY *` statement qualifies every field subsequently loaded with its source table name. Qualification is useful for preventing unintended associations between tables that happen to contain identically named fields. However, fields intended to act as join keys must retain the same unqualified name in both tables so that QlikView can identify the intended matching field during the join.

In this script, qualifying all fields also qualifies the common key fields. Consequently, the left join has no shared field name on which to match records. A join without a common field effectively produces a Cartesian combination of the participating tables, causing the number of rows to grow dramatically and potentially exhausting available memory. Replacing `QUALIFY *` with an explicit list of only the non-key fields that require qualification preserves the intended common join field while still protecting against unwanted associations. This allows QlikView to perform the join using the proper key and avoids the

excessive data expansion that leads to the out-of-memory condition. Qlik documents the behavior of the QUALIFY statement and its use for field-name qualification in the QlikView QUALIFY documentation; join behavior is described in the QlikView Join documentation.

QUESTION NO: 2

Refer to the exhibit to the right and the Load Script provided below. Exhibit.



```
ItemID, LocationID, Quantity
111A, 21, 1,100
111A, 31, 1,300
222B, 11, 2,210
222B, 21, 2,250
222B, 31, 3,125
333C, 11, 3,130
333C, 31, 3,120
```

LOAD ItemID,

LocationID,

Quantity

FROM

[inventory.txt]

(txt, codepage is 1252, embedded labels, delimiter is ',', msq);

When the developer runs the script, which values will the field Quantity contain?

A. NULL

B. 1; 2; 3

C. 1100; 1300; 2210; 2250; 3125; 3130; 3120

D. 1,100; 1,300; 2,210; 2,250; 3,125; 3,130; 3,120

ANSWER: B

Explanation:

1; 2; 3 is correct because the load statement explicitly declares a comma as the delimiter for `inventory.txt`. In the exhibited source data, the Quantity values include commas as thousands separators, such as `1,100`. Because those values are not protected as a single text value by an appropriate text qualifier, QlikView interprets each comma as a column boundary. Consequently, when QlikView reads the three named fields—ItemID, LocationID, and Quantity—it assigns only the portion before the comma to Quantity. Thus, values such as `1,100`, `2,210`, and `3,125` produce Quantity values beginning with `1`, `2`, and `3`, respectively. Repeated source rows can produce repeated values, but the distinct field values are 1, 2, and 3. The `embedded labels` setting correctly uses the first source row as field names; it does not alter delimiter handling within subsequent data rows. QlikView's script syntax documents the text-file format specification and delimiter setting for LOAD statements. See the QlikView LOAD statement documentation and the QlikView delimited-file format documentation.

QUESTION NO: 3

Refer to the exhibit to the right.

SalesOrderID	SalesAmount	City	Country
1	329	Chicago	US
2	931	New York	US
4	906	L.A.	US
5	17	LA	USA
6	547	Los Angeles	US
7	526	Berlin	DE
8	474	Hong Kong	CN
9	791	HongKong	CN
10	404	London	UK
11	198	London	US
12	606	Berlin	GER
13	141	Moscow	RU
14	223	NY	US
15	383	N.Y.	US

Which two data quality issues exist within the data set displayed in the exhibit? (Choose two.)

A. multiple spellings of the same City

B. the City value London is associated with two different Country values

C. the Country values and SalesOrderID values do not relate properly

D. the SalesAmount field name does not contain a space

E. inconsistent Country abbreviations

F. duplicate key values

ANSWER: A E

Explanation:

The displayed values show two classic dimensional-data standardization problems: **multiple spellings of the same City** and **inconsistent Country abbreviations**. A city is normally a descriptive attribute used for grouping, filtering, and associating records. When the same real-world city appears under variant spellings, punctuation, or capitalization, QlikView treats those values as distinct field values. This fragments aggregations and selections: a chart by city can split sales that should belong to one location, and a user selecting one spelling may omit records held under another.

Country abbreviations require the same type of normalization. A country represented by different codes or textual abbreviations is not a single consistent business value to QlikView; it becomes several separate symbols in the field. Standardizing those values during the load process, commonly with mapping tables and the `ApplyMap()` function, establishes a canonical city and country value before the data model is built. QlikView documents mapping loads and `ApplyMap()` as tools for replacing field values during script execution: QlikView ApplyMap documentation. Consistent field values also support reliable associations and accurate aggregations in the resulting data model; see QlikView mapping-load documentation.

QUESTION NO: 4

Which chart should a designer use to display orders, sales, and average order value grouped by year, quarter, region, product, and salesperson?

A. Pivot Table

B. Straight Table

C. Grid Chart

D. Mekko Chart

E. Bar Chart

ANSWER: A

Explanation:

Pivot Table is the appropriate choice because it is designed to present several measures across multiple, related dimensions in a compact, hierarchical layout. In this case, orders, sales, and average order value are measures, while year, quarter, region, product, and salesperson are dimensions that users may need to explore at different levels of detail. A pivot table lets the designer arrange dimensions as rows and columns, reorder them interactively, and expand or collapse dimension groups. For example, a user can begin with yearly totals, expand a year into quarters, then investigate results by region, product, and salesperson without needing a separate chart for each perspective.

This layout is particularly useful for operational and sales analysis because it supports comparison of several measures side by side while retaining the dimensional hierarchy that gives those measures context. QlikView pivot tables also support partial sums, which can make aggregations at levels such as year, quarter, or region readily visible. This makes the Pivot Table well suited to a detailed multidimensional report rather than a chart focused primarily on visualizing one relationship or distribution. Qlik's documentation describes pivot tables as tables in which dimensions and expressions can be shown as rows and columns and rearranged interactively. See QlikView Pivot Table documentation and QlikView chart documentation for pivot tables.

QUESTION NO: 5

A customer needs two independent selection contexts in the same QlikView document. One set of charts will show actual sales and another set will show budget sales for a different user-selected region.

Which two actions should a developer take to implement this requirement? (Choose two.)

A. Define an alternate state in Document Properties.

B. Assign the alternate state to the applicable sheet objects.

C. Create a separate QVW document for each selection context.

D. Use a bookmark as the only method for maintaining independent selections.

E. Rename all fields used by the budget charts.

ANSWER: A B

Explanation:

The developer should define an alternate state in Document Properties and assign that state to the objects that require an independent selection context. Selections made in objects assigned to the alternate state are maintained separately from selections in the default state.

Expressions can also explicitly reference a state by using its identifier in set analysis. This enables a chart to calculate using a named alternate state even when the chart itself is assigned to another state. Alternate states are intended for comparative analysis, such as comparing different regions, periods, or scenarios within one document. See the QlikView alternate states documentation.

QUESTION NO: 6

A customer who operates a large national sales organization needs to divide a Sales.qvw document into region-specific documents using the Region field (North, South, East, and West).

Each region should have access to region-specific data.

Which instruction should a developer give to the Server/Publisher administrator to meet the customer's needs?

A. create a QlikView Server task to reduce the Sales.qvw document based on the Region field and distribute to region-specific folders with appropriate file permissions

B. create a QlikView Publisher task to reduce the Sales.qvw document based on the Region field and distribute to region-specific folders with appropriate file permissions

C. create four QlikView Server tasks to reload data from the Sales.qvw document with data only for each region and distribute to region-specific folders with appropriate file permissions

D. create a QlikView Publisher task to reduce data from the Sales.qvw document to create four QVX files with data only for each region and distribute to region-specific folders with appropriate file permissions

ANSWER: B

Explanation:

The instruction to create a QlikView Publisher task to reduce the Sales.qvw document based on the Region field and distribute to region-specific folders with appropriate file permissions is correct. QlikView Publisher is designed to create and distribute managed document copies from a source document. Its document-reduction capability can use a field such as Region to generate separate reduced QVW documents, each containing only the data associated with one field value—for example, North, South, East, or West.

The administrator can configure the Publisher task with the Region field as the reduction field and define the applicable field values. Publisher then creates the region-specific output documents as part of the distribution process. Distributing each reduced document to a dedicated folder, together with appropriate access permissions, provides users in each regional organization access only to the document that contains their regional sales data. This approach preserves a single centrally maintained source document while automating the creation and delivery of secure, data-reduced regional versions. QlikView Publisher documentation describes Publisher as the component used for distributing QlikView documents and performing tasks such as document reduction; see QlikView Publisher and the QlikView Help.

QUESTION NO: 7

Refer to the exhibit to the right.

```
1  Transaction ID,Date,Product,Amount
2  1,11/11/2011,Toy Car,136
3  2,11/11/2011,Toy Car,164
4  3,11/11/2011,Red Bicycle,181
5  4,11/11/2011,Yellow Ball,151
6  5,11/11/2011,Blue Bicycle,142
7  6,11/11/2011,Green Ball,140
8  7,11/11/2011,Wooden Bat,129
9  8,11/11/2011,Wooden Bat,103
10 9,11/11/2011,Black Glove,108
11 10,11/11/2011,Black Glove,178
12 Transaction ID,Date,Product,Amount
13 11,11/12/2011,Blue Bicycle,180
14 12,11/12/2011,Green Ball,115
15 13,11/12/2011,Wooden Bat,146
16 14,11/12/2011,Toy Car,114
17 15,11/12/2011,Toy Car,175
18 16,11/12/2011,Glove, Black,160
19 17,11/12/2011,Red Bicycle,145
20 18,11/12/2011,Wooden Bat,114
21 19,11/12/2011,Glove, Black,198
22 20,11/12/2011,Yellow Ball,124
```

A customer needs the data displayed in the exhibit loaded into QlikView for analysis.

The data is stored in a flat file and new rows are appended daily.

Which two data quality issues will a developer have to resolve? (Choose two.)

A. the Transaction ID field name contains a space

B. one or more Product field values contain spaces

C. the Date field values contain special characters

D. one or more header rows are included in the file data

E. one or more Product field values contain the file delimiter

F. the Date field name is also a function name and is a reserved word

ANSWER: D E

Explanation:

The data-quality issues that require handling are “one or more header rows are included in the file data” and “one or more Product field values contain the file delimiter.” Because rows are appended daily, a source process may append a column-heading row along with each daily batch. QlikView must prevent those repeated headings from becoming records in the data model. This is typically handled in the load script by selecting or filtering valid data rows, rather than treating every row after the initial header as transactional data. QlikView’s LOAD statement supports transformations and conditions that allow the developer to control which source records enter the model.

A product value that contains the delimiter is also a structural issue in a delimited flat file. Unless the source exports that value using valid text quoting/escaping and QlikView is configured to interpret it accordingly, QlikView will parse the embedded delimiter as a column boundary. The resulting row can have shifted field values or an incorrect number of fields, so the developer must correct the source format or configure the file interpretation and cleansing logic before analysis. These actions preserve a consistent record layout as new daily data is added. See Qlik’s documentation for the LOAD statement and the File Wizard.

QUESTION NO: 8

A developer is creating a master calendar from a numeric date field named OrderDate.

Which two functions can be used in the load script to derive calendar attributes from OrderDate? (Choose two.)

A. Year(OrderDate)

B. Month(OrderDate)

C. Sum(OrderDate)

D. Count(OrderDate)

E. ApplyMap(OrderDate)

ANSWER: A B

Explanation:

`Year(OrderDate)` returns the year component of OrderDate and can be used to create a Year field for calendar analysis. The returned value can be used for sorting, selections, and chart dimensions.

`Month(OrderDate)` returns the month component as a dual value with a textual month representation and an associated numeric month value. This makes it useful for displaying month names while maintaining chronological month sorting. Functions such as `Sum()` and `Count()` aggregate data and do not derive date attributes.

QUESTION NO: 9

A developer is testing a reload script that connects to several optional source files.

The developer needs the script to continue processing after a load error so that the ScriptError variable can be evaluated later in the script.

Which script setting should the developer use?

A. ErrorMode=0;

B. ErrorMode=1;

C. Set ScriptError=0;

D. Exit Script;

ANSWER: A

Explanation:

`ErrorMode=0;` is correct because it instructs QlikView to continue script execution when an error occurs. This allows the developer to test the value of the system variable `ScriptError` after a statement has failed and to take appropriate action, such as issuing a trace message, loading an alternative source, or stopping the script under controlled conditions.

By default, QlikView stops the reload when a script error occurs. Setting `ErrorMode` to 0 changes that behavior temporarily. A developer should restore the standard error-handling behavior when it is no longer needed. See the QlikView ErrorMode documentation.