PassQueen
a.k.a dumpsqueen.com

Qlik Sense Data Architect Certification Exam 2022

Qlik QSDA2022

Version Demo

Total Demo Questions: 12

Total Premium Questions: 125

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Design Applications	20
Topic 2, Load Data	67
Topic 3, Design Data Models	38
Total	125

QUESTION NO: 1

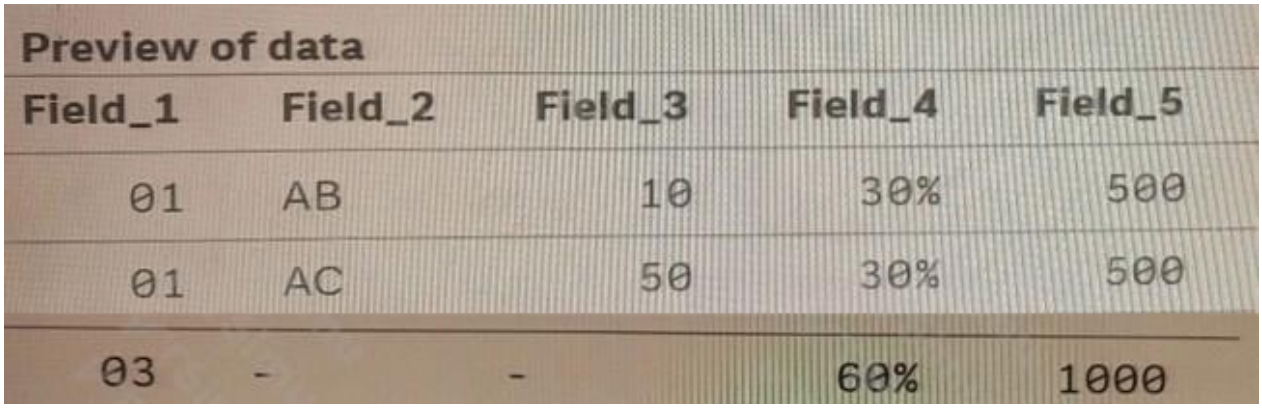
A data architect executes the following script:

```
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?

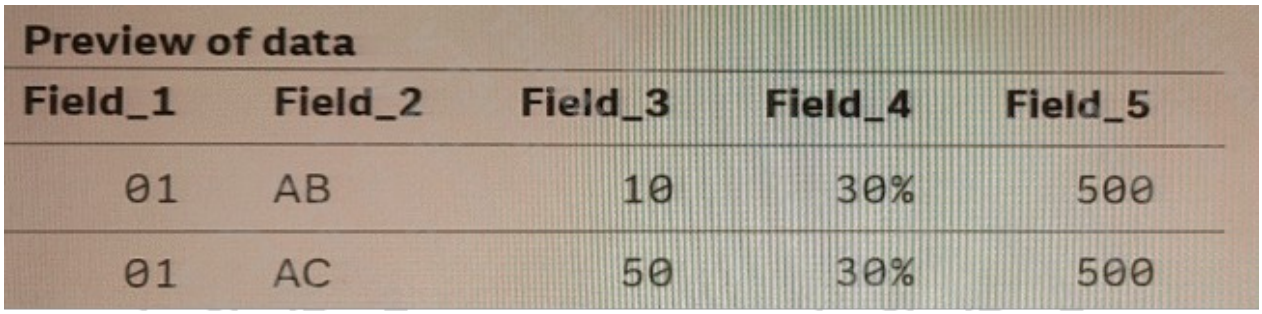
A)



Preview of data

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

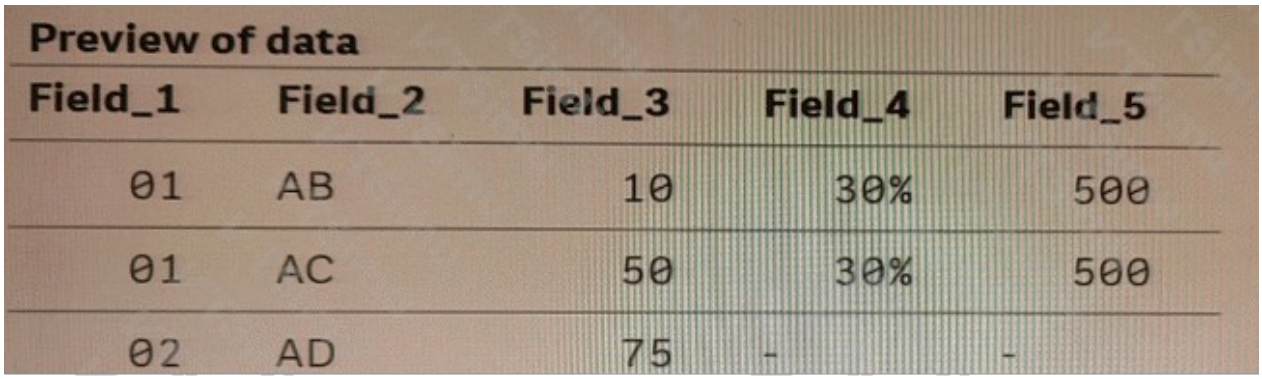
B)



Preview of data

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

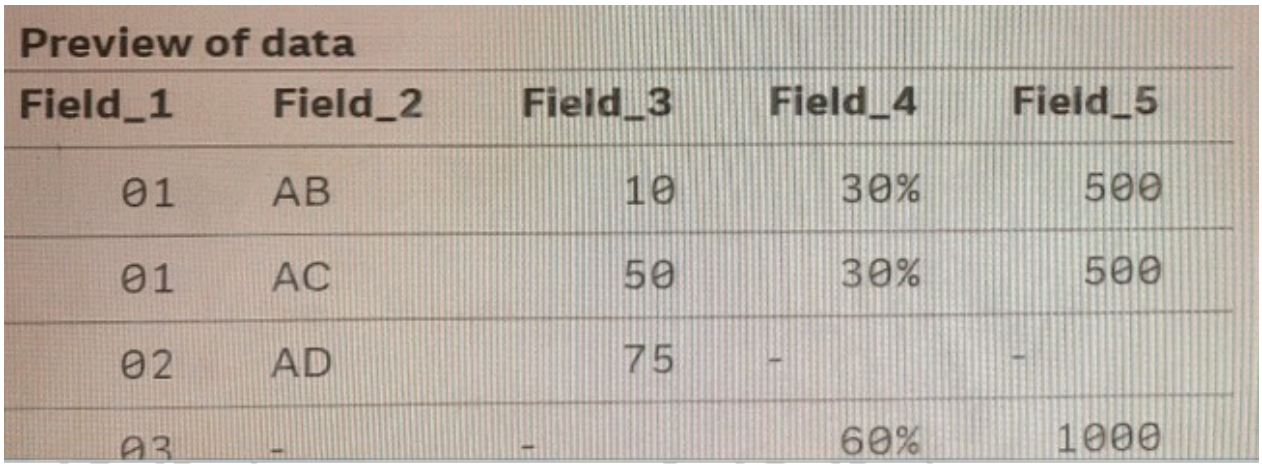
C)



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	-	-

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	-	-
03	-	-	60%	1000

A. Option A

B. Option B

C. Option C

D. Option D

ANSWER: D

Explanation:

Option D is correct. In Qlik Sense, the final contents of a resident table are determined by the order in which script statements execute and by how each LOAD statement transforms, filters, or appends records from its input. Table labels identify the target table being created or modified, while field expressions are evaluated for every source record processed by the LOAD statement. Where the script uses preceding loads, the lower LOAD runs first and supplies its result to the LOAD above it; this processing order is essential when following the resulting values shown for Table A. Qlik also applies automatic concatenation when consecutive loads produce the same field set unless the script explicitly directs otherwise. The displayed result in Option D reflects the records and field values produced after the script has completed all of these operations, rather than an intermediate result from an individual statement. This follows Qlik's documented script execution model for LOAD statements and table concatenation behavior. See the Qlik documentation for preceding loads and Concatenate.

QUESTION NO: 2

A data architect needs to revise an existing app.

The number of data rows has grown rapidly recently. While the app is in production, users are becoming increasingly unhappy about the response times when they make selections

Which two methods should be used to improve performance? (Select two.)

A. Use dynamic script generation with variables

B. Denormalize the schema

C. Make sure any UI variables are preceded by '='

D. Use flags in the data model to simplify set analysis

E. Create master items for all complex expressions

ANSWER: B D

Explanation:

Denormalize the schema and use flags in the data model to simplify set analysis are appropriate performance improvements for an app in which growing data volume is affecting interactive selection response. A denormalized Qlik data model reduces the number of table associations that Qlik's associative engine must traverse when users select values. Where it is practical and does not introduce incorrect relationships or unnecessary duplication, combining related lookup attributes into a fact-oriented table can produce a simpler model with fewer hops and better interactive responsiveness. Qlik recommends designing a data model that is as simple as possible and avoiding unnecessary snowflake-style structures.

Flags provide pre-calculated fields that represent commonly used business conditions, such as current period, prior period, active status, or a target category. Expressions can then select the flag directly rather than repeatedly resolving more complex conditional logic or date-based set analysis at calculation time. This reduces expression complexity and makes chart calculations more efficient and easier to maintain as the app grows. Both techniques address workload experienced during user interaction: simplifying associative navigation through the model and reducing repeated calculation work in visualizations. See Qlik guidance on app performance and understanding data models.

QUESTION NO: 3

A published app contains Salary and Bonus fields. All authorized users must be able to analyze the same records, but a group of users must NOT be able to see these fields.

Which two actions should the data architect take? (Select two.)

A. Add an OMIT field to the Section Access table

B. Create authorization records for the affected users

C. Use a reduction field for Salary and Bonus

D. Drop Salary and Bonus from the application data model

E. Apply conditional show expressions to every chart

ANSWER: A B

Explanation:

An **OMIT field** in the Section Access authorization table identifies fields that a user is not permitted to access. The data architect must also create the applicable Section Access authorization records using values such as `ACCESS` and `USERID` for the affected users. This restricts access to the named fields without reducing the data records available to those users. Section access is evaluated when the app is opened, and OMIT is used for field-level restriction rather than row-level data reduction. Qlik describes OMIT and Section Access configuration in its Managing security with section access documentation.

QUESTION NO: 4

Refer to the exhibits.

The first table shows the source table (Original table).

This data represents the stocks stored every month for each product:

• The relevant fields are productid, qty, and date.

• The date field represents the calendar months using

• The qty field shows the product stock fluctuation from the current month versus the previous month. If there is no fluctuation between months, there are no new entries in the table.

The second table shows a Pivot table visualization the data analyst needs to create in the app displaying per each product the monthly trend of available stock.

For performance reasons, the data analyst requests the data architect to calculate the running stock quantity of each product for every month in the script.

Which approach should the data architect use?

A. 2 RIGHT JOIN the Combined table with the Original table to populate the missing qty values

B. 2. LEFT JOIN the Combined table with the Original table to populate the missing qty values

C. 2. RIGHT JOIN the Calendar table back to the Original table to populate the missing qty values

D. 2. LEFT JOIN the Calendar table back to the Original table to populate the missing qty values

ANSWER: D

Explanation:

“2. LEFT JOIN the Calendar table back to the Original table to populate the missing qty values” is the appropriate approach because calculating a monthly running stock requires a continuous monthly series for every product. The calendar supplies the missing month records, while the original data supplies the recorded month-to-month quantity changes. A left join preserves the complete month scaffold and attaches each available quantity fluctuation to its matching product and month. Missing quantities can then be treated as zero before calculating the cumulative stock quantity in chronological order for each product, commonly using an inter-record calculation such as `RangeSum(Peek(...), qty)` after sorting by product and date.

Performing this work in the load script materializes the monthly stock balance as a field in the data model. The pivot table can therefore use the precomputed result directly rather than repeatedly evaluating a running-total expression at visualization calculation time. This is especially beneficial when the app has many products, months, or concurrent users. Qlik script joins combine tables on their common field names, and a left join retains all rows from the table being extended, which is essential when the generated calendar provides the full required time grain. See Qlik’s documentation for Join script prefixes and the Peek function.

QUESTION NO: 5

Refer to the exhibit.

```
239
240     CompaniesDetails:
241     LOAD [CompanyID],
242         Age(Today(),CompanyStarted) as CompanyAge;
243     LOAD
244         [CompanyID],
245         [CompanyStarted]
246
247      FROM [lib://MyDataFiles/DAExam1.xlsx]
248     (ooxml, embedded labels, table is [Company Names]);
249
250     let a = FieldName(2 ,'CompaniesDetails');
251     Trace $(a);
252
253     CompaniesDesc:
254     LOAD
255         [CompanyID],
256         [CompanyDescription],
257         [CompanyName]
258      FROM [lib://MyDataFiles/DAExam1.xlsx]
259     (ooxml, embedded labels, table is [Company Names]);
260
```

A business analyst reports that the 'CompanyAge' field does NOT display for users The data architect examines the LOAD script and wants to place the breakpoint in the script to check the field name.

Which line number should the data architect use?

A. 242 because field name appears in the output Panel of the debugger

B. 251 because the field name appears in the Variable panel of the debugger

C. 251 because the field name appears in the Qlik Log files

D. 251 because the field name appears in the Output panel of the debugger

ANSWER: D

Explanation:

251 because the field name appears in the Output panel of the debugger is correct. To verify whether *CompanyAge* is being created with the intended name, the breakpoint must be placed after the LOAD statement that produces the field. At line 251, the statement has executed, so the Script Debugger can expose the resulting output and allow the architect to inspect the fields generated by that load operation. The Output panel is the appropriate debugger view for confirming the fields and values produced during execution, including whether an alias or expression has resulted in the expected *CompanyAge* field name. This is especially useful when a field is calculated, renamed with `AS`, or carried forward from a preceding load, because the field visible to users is determined by the loaded result rather than solely by a source-column name. Qlik's script debugger supports pausing at selected script lines and examining execution results to troubleshoot load-script behavior. See the Qlik Sense documentation on debugging scripts and the LOAD statement reference.

QUESTION NO: 6

A data architect is adding section access to reduce a published app by COUNTRY.

Which two actions should the data architect take? (Select two.)

A. Place the Section Access load before Section Application

B. Use the same uppercase COUNTRY field name in both sections

C. Load Section Access after all application tables are loaded

D. Use a different field name in the application data for COUNTRY

E. Create a data island containing the authorization values

ANSWER: A B

Explanation:

The `Section Access` portion of the script must be loaded before `Section Application`. This allows Qlik Sense to process the authorization table separately from the application data while reloading the app.

The reduction field must use the same uppercase field name in both sections. For example, if the authorization table uses `COUNTRY`, the application data must contain the corresponding field as `COUNTRY`. Matching the field name enables Qlik Sense to associate authorization values with application records and perform data reduction when users open the app. See Qlik guidance on managing security with section access.

QUESTION NO: 7

Refer to the exhibit.

What are the values of the variables vLoadStart and vLoadComplete after the script executes?

A. vLoadComplete: current system date and time when the script ended

B. vLoadStart: system date when the app was opened vLoadComplete 'now()'

C. vLoadStart: system date when the app was opened vLoadComplete: current system date and time when the script ended

D. vLoadStart: current system date and time when the script started vLoadComplete: current system date and time when the script ended

ANSWER: D

Explanation:

`vLoadStart: current system date and time when the script started vLoadComplete: current system date and time when the script ended` is correct because the script evaluates the `now()` expression at each assignment. The first assignment occurs at the beginning of the reload script, so `vLoadStart` receives the timestamp at which that part of the reload begins. The later assignment occurs after the intervening script statements have executed, so `vLoadComplete` receives the timestamp at that later point, effectively recording when the script completes. In Qlik load scripting, `LET` evaluates the expression on its right side and stores the resulting value in the variable; it does not store the textual function call for later evaluation. The `now()` function returns the current timestamp when called under its default timing behavior, making separate calls useful for reload duration logging and audit information. The stored values are Qlik timestamp values, whose display can depend on the application's configured timestamp format. See Qlik's documentation for now() and for the LET statement.

QUESTION NO: 8

A transaction table contains a long text-based CustomerID field that is repeated in millions of rows. A data architect wants to reduce memory consumption in the app.

Which two statements are true about using the AutoNumber function? (Select two.)

A. AutoNumber creates an integer representation of a loaded value

B. AutoNumber values can vary across separate reloads or apps

C. AutoNumber preserves the original text value in the same field

D. AutoNumber automatically creates a persistent cross-app key

E. AutoNumber can only be used in chart expressions

ANSWER: A B

Explanation:

AutoNumber() replaces distinct source values with internally generated integer values during the load, which can reduce memory consumption when long keys occur repeatedly in a large fact table. However, AutoNumber values are generated for the current load and can differ between reloads or separate apps. Therefore, they should not be used as persistent keys for separately generated QVD files or for associations between independently loaded applications. Qlik documents AutoNumber behavior in the AutoNumber function documentation.

QUESTION NO: 9

A customer has a dataset that contains latitude and longitude data for service points around the country. The data is retrieved using the following statement:

```
Locations:
LOAD LocationName, Lat, Long;
SQL SELECT LocationName, Lat, Long FROM Locations;
```

It must be clear to the end user that this is geographic data. Drag and drop, map-based visualization of this data is required. Which two steps should the data architect take to support this data? (Select two.)

A. Define Location as a master item, and set the tag to $geodata

B. Add GeoMakePoint(Lat, Long) AS Point to the preceding load

C. Add GeoMakePoint(Lat, Long) AS Point to Location's preceding load

D. Add the following to the end of the script:
TAG FIELD LocationName WITH '$geodata', '$related';
TAG FIELD Point WITH '$geodata', '$related';

E. Add the following to the end of the script:
TAG FIELD LocationName WITH '$geoname', '$relates_Point';
TAG FIELD Point WITH '$geopoint', '$relates_LocationName', '$hidden';

ANSWER: B E

Explanation:

Adding `GeoMakePoint(Lat, Long) AS Point` in a preceding load creates a Qlik geographic point from the separate latitude and longitude fields. A point field is required so that Qlik Sense can recognize the coordinate pair as a single geographic location and use it directly in map visualizations. The latitude value must be supplied first and the longitude value second. Creating this point also supports the intended drag-and-drop experience because map layers can use a recognized geospatial point rather than requiring users to manually configure two numeric fields.

The field tags establish the semantic relationship Qlik Sense needs for automatic geographic interpretation. Tagging `LocationName` with `$geoname` identifies it as the human-readable name for a geographic entity, while `$relates_Point` links that name to the point field. Tagging `Point` with `$geopoint` identifies it as geographic point data, and `$relates_LocationName` completes the relationship in the other direction. The `$hidden` tag keeps the technical point field out of normal field selection while preserving its use by map visualizations. Together, the generated point and these

system tags make the service locations understandable and readily usable on a map. See Qlik's GeoMakePoint documentation and TAG statement documentation.

QUESTION NO: 10

Refer to the exhibits.

While debugging an app, a developer loads data from an application layer QVD file.

In the process of separating a concatenated key into two parts, some split results are missing data

What should the data architect do?

A. Utilize a combination of LEFT(), MID(), and RIGHT() functions to capture the key components

B. In the SUBFIELD function, replace the '_' with a '|' or '_' character.

C. Utilize a combination of LEFT(), MID(), and RIGHTO functions to capture the key components

D. In the SUBFIELD function, replace the '-' with a '|' or '_' character

E. Instruct the developer of the QVD file to correct the generation of the ConcatKeyAlpha field

F. Wrap an IF() function around the SUBFIELDQ functions to check and adapt to null values character

G. Instruct the developer of the QVD file to correct the generation of the ConcatKeyAlpha field

ANSWER: D

Explanation:

In the SUBFIELD function, replace the '-' with a '|' or '_' character is correct because `SubField()` can split a value only when its delimiter argument matches the separator actually present in the concatenated key. A hyphen is often part of legitimate business values, such as codes, names, or identifiers. If it is treated as the separator when the QVD's `ConcatKeyAlpha` values were instead constructed with a pipe or underscore, the requested subfield position may not exist for some rows. Qlik then returns a null result for that requested part, which appears as missing split data. Using the same delimiter as the one used to construct the key ensures that each component is consistently identified and returned. The separator should also be chosen so it cannot occur in either source component; this prevents accidental extra splits and preserves the two-part key structure. Qlik documents that `SubField(text, delimiter, field_no)` extracts fields based on the specified delimiter, and that a missing requested field produces NULL. See the Qlik SubField function documentation and Qlik string-function reference.

QUESTION NO: 11

A data architect needs to update only selected tables during development without executing every standard LOAD statement in a large app.

Which two actions are required to use a partial reload? (Select two.)

A. Prefix the relevant LOAD statements with Add, Replace, or Merge

B. Start the reload as a partial reload

C. Add a WHERE EXISTS clause to every LOAD statement

D. Store every table as a QVD before starting the reload

E. Use a Binary load as the final script statement

ANSWER: A B

Explanation:

A partial reload executes only load statements that use the **Add**, **Replace**, or **Merge** prefixes. The relevant LOAD statements must therefore be prefixed appropriately. The reload must also be started as a partial reload, rather than as a standard full reload. Standard LOAD statements without these prefixes are ignored during a partial reload. This can reduce development reload time when only selected data tables need to be updated. Qlik documents partial reload behavior in the Add prefix documentation and the Replace prefix documentation.

QUESTION NO: 12

A data architect needs to arrange data to create an app with a map where multiple location points consolidate into hexagonal areas based on postal codes

The areas will be color coded based on the number of vendors in the location.

Which GeoAnalytics operation should the data architect use?

A. Binning

B. Intersect

C. AddressLookup

D. Simplify

ANSWER: A

Explanation:

Binning is the appropriate GeoAnalytics operation because it spatially aggregates many individual point locations into consistently sized geographic cells, including hexagonal bins. This reduces visual clutter from overlapping or densely distributed vendor locations while preserving a meaningful view of their geographic distribution. The resulting bin geometry can be loaded into Qlik Sense and used as the area layer of a map.

During the binning process, values associated with the points can be aggregated for each cell. In this case, the relevant measure is the count of vendors within each hexagonal area. That count can then drive the map layer's color expression, allowing users to immediately identify areas with relatively high or low vendor concentration. Postal-code information may be retained as an associated attribute or used in the data preparation logic, while the hexagonal spatial aggregation provides the requested consolidated map areas.

Qlik's GeoAnalytics documentation describes Binning as an operation for aggregating geographic point data into grid areas, including hexagons: Qlik GeoAnalytics Binning documentation. For general configuration of area layers and color-based measures in maps, see Qlik Sense map properties.