DUMPSBOSS.

Implementing Analytics Solutions Using Microsoft Fabric

Microsoft DP-600

Version Demo

Total Demo Questions: 10

Total Premium Questions: 140

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

QUESTION NO: 1 - (HOTSPOT)

You need to migrate the Research division data for Productline2. The solution must meet the data preparation requirements. How should you complete the code? To answer, select the appropriate options in the answer area

NOTE: Each correct selection is worth one point.

ANSWER:

Explanation:

In the answered screenshot, the first dropdown (the `format()` for the write) is set to **delta**, and the second dropdown (the `save()` destination) is set to **Tables/productline2**. Writing in Delta is the right call for Fabric data preparation because Delta tables give you schema enforcement, ACID transactions, and better reliability for downstream analytics compared to writing back out as CSV or Parquet files. In a Fabric Lakehouse, saving to the `Tables/` area is also important because that's what creates/maintains a managed table experience rather than just dropping files into the `Files/` area.

The issue is the destination path. The question specifically says you need to migrate the **Research division** data for **Productline2**. That implies the table should be stored under a Research-specific location. Since the answer choices include **Tables/research/productline2**, that is the option that matches the division requirement. Selecting **Tables/productline2** would place the data in a generic location and doesn't reflect the Research division separation the question calls for.

So the correct completion is to keep **delta** as the format, but change the `save()` path to **Tables/research/productline2**. For more background on Delta Lake and why it's used for reliable table storage, see the Delta documentation at https://learn.microsoft.com/en-us/azure/databricks/delta/. For Fabric Lakehouse concepts (Tables vs Files), see https://learn.microsoft.com/en-us/fabric/data-engineering/lakehouse-overview.

QUESTION NO: 2 - (DRAG DROP)

You have a Fabric tenant that contains a lakehouse named Lakehouse1

Readings from 100 IoT devices are appended to a Delta table in Lakehouse1. Each set of readings is approximately 25 KB. Approximately 10 GB of data is received daily.

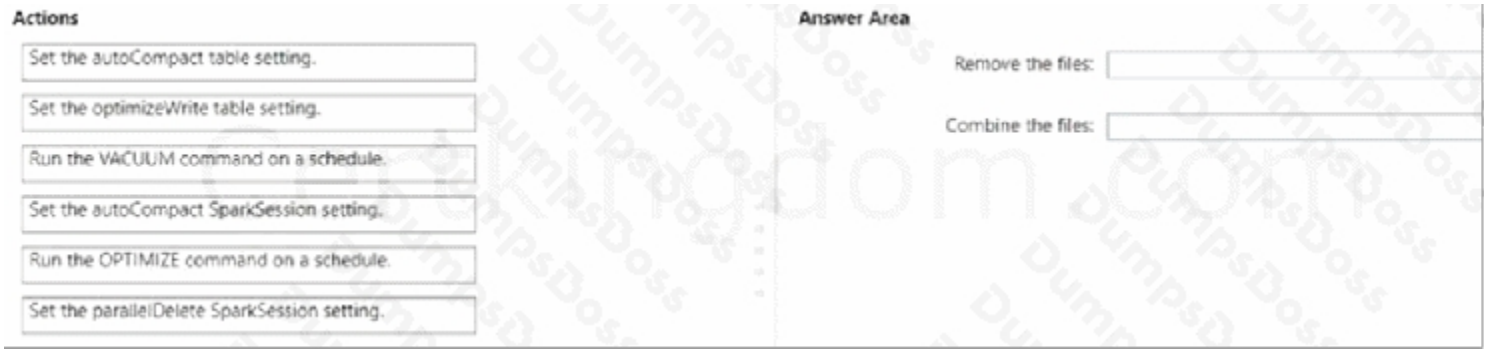
All the table and SparkSession settings are set to the default.

You discover that queries are slow to execute. In addition, the lakehouse storage contains data and log files that are no longer used.

You need to remove the files that are no longer used and combine small files into larger files with a target size of 1 GB per file.

What should you do? To answer, drag the appropriate actions to the correct requirements. Each action may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

NOTE: Each correct selection is worth one point.



ANSWER:

Explanation:

In the answer area, both drop zones (“Remove the files” and “Combine the files”) are blank, meaning no actions were actually selected. That makes the current response incomplete/incorrect because the question requires you to pick actions that (1) delete unused files and (2) compact lots of small files into larger ones.

For **removing files that are no longer used**, the correct Delta Lake operation is **VACUUM**. Delta tables keep old data files around so time travel and rollback can work. Over time, that leaves behind data files (and related artifacts) that are no longer referenced by the current table state. Running **VACUUM** removes those unreferenced files (subject to the retention policy),

which helps reduce storage bloat. In practice, you typically run it on a schedule as part of maintenance. See: Delta Lake VACUUM.

For **combining small files into larger files** with a target size (here, 1 GB), the right tool is **OPTIMIZE**. OPTIMIZE performs file compaction by rewriting many small files into fewer larger files, which improves query performance by reducing file listing overhead and the number of file reads. This is exactly what you want when you're ingesting lots of small appends (25 KB records, ~10 GB/day) and queries are slowing down due to file fragmentation. Running **OPTIMIZE** on a schedule is a common best practice. See: Delta Lake OPTIMIZE.

Settings like *optimizeWrite* and *autoCompact* can help reduce small-file creation during ingestion, but they don't replace the need for periodic OPTIMIZE (for existing fragmentation) and VACUUM (for removing unreferenced files). The question explicitly asks to remove unused files and compact to ~1 GB files, which maps cleanly to VACUUM and OPTIMIZE.

QUESTION NO: 3

Note: This section contains one or more sets of questions with the same scenario and problem. Each question presents a unique solution to the problem. You must determine whether the solution meets the stated goals. More than one solution in the set might solve the problem. It is also possible that none of the solutions in the set solve the problem.

After you answer a question in this section, you will NOT be able to return. As a result, these questions do not appear on the Review Screen.

Your network contains an on-premises Active Directory Domain Services (AD DS) domain named contoso.com that syncs with a Microsoft Entra tenant by using Microsoft Entra Connect.

You have a Fabric tenant that contains a semantic model.

You enable dynamic row-level security (RLS) for the mode! and deploy the model to the Fabric service.

You query a measure that includes the username () function, and the query returns a blank result. You need to ensure that the measure returns the user principal name (UPNJ of a user.

Solution: You add user objects to the list of synced objects in Microsoft Entra Connect. Does this meet the goal?

A. Yes

B. No

ANSWER: B

Explanation:

Correct answer: No. Adding more user objects to the Microsoft Entra Connect sync scope won't fix `USERNAME()` returning blank in the Fabric/Power BI service. In the service, the identity functions behave a bit differently than on a local desktop model, and `USERNAME()` often doesn't return what you expect (or can return blank) depending on the connection/auth context.

If your goal is specifically to get the user principal name (UPN), the more reliable approach is to use `USERPRINCIPALNAME()` in your DAX, because it's designed to return the signed-in user's UPN in the Power BI/Fabric service. Syncing "more" users doesn't change what the service passes into DAX for the current viewer; it just changes which identities exist in Entra ID.

So, this solution doesn't meet the goal because it targets directory sync, not the DAX identity function behavior. Reference: https://learn.microsoft.com/en-us/dax/userprincipalname-function-dax and https://learn.microsoft.com/en-us/power-bi/enterprise/service-admin-rls

QUESTION NO: 4

You need to ensure that Contoso can use version control to meet the data analytics requirements and the general requirements. What should you do?

A. Store all the semantic models and reports in Data Lake Gen2 storage.

B. Modify the settings of the Research workspaces to use a GitHub repository.

C. Store all the semantic models and reports in Microsoft OneDrive.

D. Modify the settings of the Research division workspaces to use an Azure Repos repository.

ANSWER: B

Explanation:

In Microsoft Fabric, the practical way to get real version control (branches, pull requests, history, and collaboration) for items like notebooks, lakehouses, semantic models, and reports is to connect the workspace to a Git repository using Git integration. That's what actually gives you source control behavior—simply storing PBIX/metadata files in OneDrive or ADLS Gen2 doesn't give you proper branching and PR-based review.

So the right move is to go into the workspace settings for the Research workspaces and connect them to a GitHub repo. Once connected, Fabric can sync workspace items to the repo and teams can manage changes like they would with normal code. This aligns with typical "general requirements" around governance and controlled deployments.

Azure Repos can also be used in some Microsoft products, but for Fabric's Git integration the commonly supported and tested path is GitHub. That's why the GitHub option is the best fit here.

References: https://learn.microsoft.com/en-us/fabric/cicd/git-integration/overview

QUESTION NO: 5

You have a Fabric tenant that contains a warehouse.

A user discovers that a report that usually takes two minutes to render has been running for 45 minutes and has still not rendered.

You need to identify what is preventing the report query from completing. Which dynamic management view (DMV) should you use?

A. sys.dm-exec_requests

B. sys.dn_.exec._sessions

C. sys.dm._exec._connections

D. sys.dm_pdw_exec_requests

ANSWER: D

Explanation:

In a Fabric Warehouse, the most useful place to start when a query is “stuck” is the DMV that shows you what requests are currently running and what state they’re in. `sys.dm_pdw_exec_requests` does exactly that: it lists active (and recent) requests and includes status details that help you spot things like long-running execution, queuing, waits, or blocking that can keep a report from finishing.

The other options (the `sys.dm_exec_*` DMVs) are SQL Server-style DMVs and aren’t the right fit for the dedicated/warehouse-style engine behavior you’re troubleshooting here. If you want to know what’s preventing completion, you need the request-level view that’s designed for the warehouse engine, and that’s `sys.dm_pdw_exec_requests`.

Reference: https://learn.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-pdw-exec-requests-transact-sql

QUESTION NO: 6 - (HOTSPOT)

You have a Microsoft Power BI project that contains a file named definition.pbir. definition.pbir contains the following JSON.

```
{
  "version": "1.0",
  "datasetReference": {
    "byPath": {
      "path": "../Sales.Dataset"
    },
    "byConnection": null
  }
}
```

For each of the following statements, select Yes if the statement is true. Otherwise, select No. NOTE:

Each correct selection is worth one point.

Answer Area

Statements	Yes	No
`definition.pbir` is in the PBIR-Legacy format.	○	○
The semantic model referenced by `definition.pbir` is located in the Power BI service.	○	○
When the related report is opened, Power BI Desktop will open the semantic model in full edit mode.	○	○

ANSWER:

Explanation:

Let's break down what the `definition.pbir` JSON is telling you and then map that to each Yes/No choice. The key part is `datasetReference`. In the snippet, the dataset is referenced using `byPath` with a relative path: `"path": "../Sales.Dataset"`, and `byConnection` is explicitly `null`. That combination means the report is pointing to a semantic model stored alongside the report in the Power BI Project structure (a local project artifact), not to a remote semantic model in the Power BI service.

Statement 1: "`definition.pbir` is in the PBIR-Legacy format." The presence of a versioned schema (`"version": "1.0"`) and the modern `datasetReference` object is consistent with the current PBIR definition format rather than the older legacy layout. So the correct choice is **No**.

Statement 2: "The semantic model referenced by `definition.pbir` is located in the Power BI service." Because the reference is `byPath` to a local `.Dataset` folder and not a service connection (`byConnection`), the semantic model is not in the service. So the correct choice is **No**.

Statement 3: "When the related report is opened, Power BI Desktop will open the semantic model in full edit mode." Since the report is tied to a local semantic model in the project (not a remote/live connection to the service), Desktop can open and edit the model fully. So the correct choice is **Yes**.

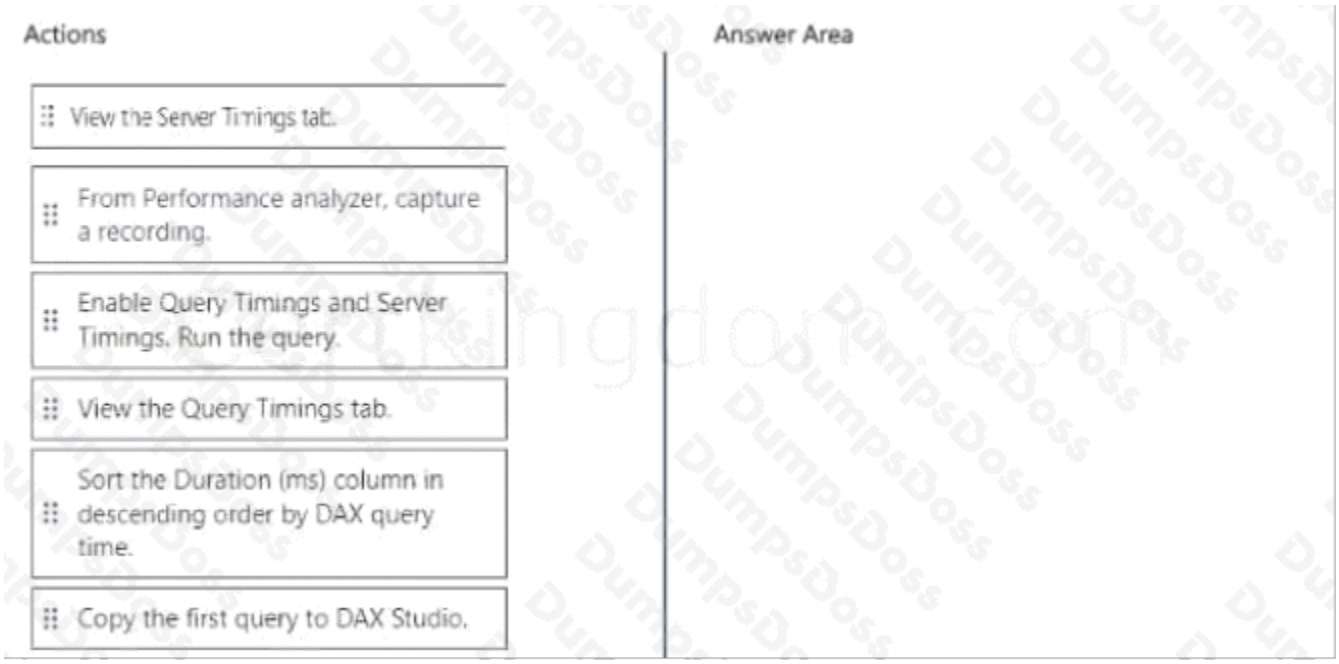
References: Power BI projects (PBIP) overview, About semantic models (datasets) and report connections.

QUESTION NO: 7 - (DRAG DROP)

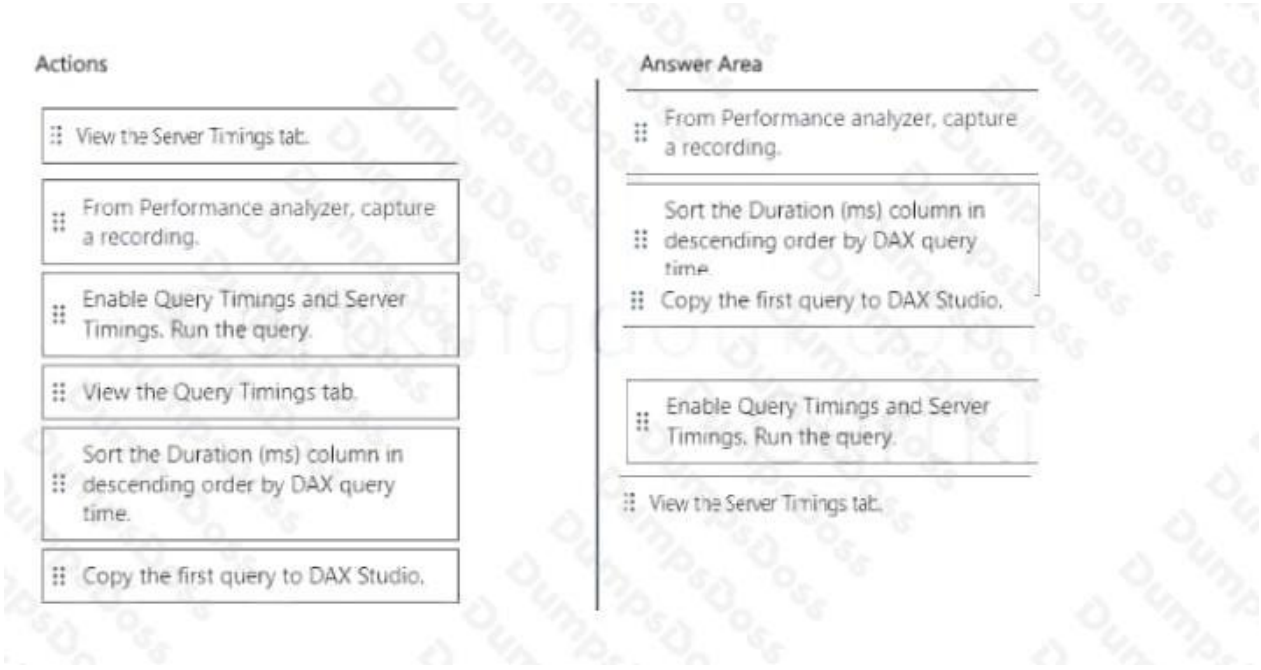
You have a Fabric tenant that contains a Microsoft Power BI report named Report 1. Report1 is slow to render. You suspect that an inefficient DAX query is being executed.

You need to identify the slowest DAX query, and then review how long the query spends in the formula engine as compared to the storage engine.

Which five actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.



ANSWER:



Explanation:

Because Report1 is slow, the first thing you want is to figure out *which* visual (and therefore which DAX query) is taking the longest. That's exactly what Power BI's Performance analyzer is for: you start a recording, interact with the report so the visuals render, and then use the captured timings to identify the worst offender. In the action list, that maps to starting the recording and then sorting the captured results by duration so the slowest DAX query rises to the top.

Once you've identified the slowest query, you need deeper engine-level detail (Formula Engine vs Storage Engine). Performance analyzer alone won't break down FE vs SE time; DAX Studio's Server Timings is the standard tool for that. So

you copy the slowest query from Performance analyzer into DAX Studio, enable the timing features, run the query, and then review the Server Timings output. Server Timings is where you can see how much time is spent in the Storage Engine (VertiPaq/DirectQuery source) versus the Formula Engine (DAX evaluation and callbacks). That directly satisfies the requirement to “review how long the query spends in the formula engine as compared to the storage engine.”

In practice, Query Timings can also be useful, but it’s not the primary view for FE vs SE comparison; Server Timings is. That’s why, given you must choose only five actions from six, the most reasonable omission is “View the Query Timings tab.”

References: Power BI Desktop performance analyzer, DAX Studio – Server Timings.

QUESTION NO: 8

You have a Fabric tenant that contains a semantic model.

You need to prevent report creators from populating visuals by using implicit measures.

What are two tools that you can use to achieve the goal? Each correct answer presents a complete

solution.

NOTE: Each correct answer is worth one point.

A. Microsoft Power BI Desktop

B. Tabular Editor

C. Microsoft SQL Server Management Studio (SSMS)

D. DAX Studio

ANSWER: A B

Explanation:

To stop people from accidentally using implicit measures (like Power BI auto-creating “Sum of Sales” when you drop a column on a visual), you need a tool that can change the semantic model setting that controls implicit measures.

You can do this in **Power BI Desktop** by turning off implicit measures in the model settings (so visuals can’t generate those automatic aggregations). That pushes the model toward using only explicit measures you’ve created on purpose.

You can also do it with **Tabular Editor**, which is commonly used for Fabric/Power BI semantic model authoring and lets you edit model properties directly (including disabling implicit measures) and manage measures in a more controlled way.

SSMS and DAX Studio don’t really fit this goal: SSMS is mainly for server/admin tasks, and DAX Studio is for querying and performance analysis—it won’t enforce a “no implicit measures” rule for report authors.

References: https://learn.microsoft.com/en-us/power-bi/transform-model/desktop-implicit-measures and https://docs.tabulareditor.com/

QUESTION NO: 9

You have a Fabric tenant that contains a lakehouse.

You plan to query sales data files by using the SQL endpoint. The files will be in an Amazon Simple Storage Service (Amazon S3) storage bucket.

You need to recommend which file format to use and where to create a shortcut.

Which two actions should you include in the recommendation? Each correct answer presents part of the solution.

NOTE: Each correct answer is worth one point.

A. Create a shortcut in the Files section.

B. Use the Parquet format

C. Use the CSV format.

D. Create a shortcut in the Tables section.

E. Use the delta format.

ANSWER: B D

Explanation:

To query files smoothly through a Lakehouse SQL endpoint, you want a format that's built for analytics. Parquet is columnar, compressed, and designed for fast reads, so it's a much better fit than CSV for SQL-style reporting and large scans.

For the shortcut location, put the shortcut under **Tables** when you want the data to show up as queryable tables in the SQL endpoint experience. A shortcut under **Files** is still useful for Spark/file access, but it won't give you the same "table-like" behavior for the SQL endpoint.

References: https://learn.microsoft.com/en-us/fabric/data-engineering/lakehouse-overview and https://learn.microsoft.com/en-us/fabric/data-engineering/lakehouse-shortcuts

QUESTION NO: 10

You have a Fabric workspace that contains a Direct Query semantic model. The model queries a data source that has 500 million rows.

You have a Microsoft Power BI report named Report1 that uses the model. Report! contains visuals on multiple pages.

You need to reduce the query execution time for the visuals on all the pages.

What are two features that you can use? Each correct answer presents a complete solution. NOTE: Each correct answer is worth one point.

A. user-defined aggregations

B. automatic aggregation

C. query caching

D. OneLake integration

ANSWER: A C

Explanation:

With a DirectQuery model hitting a 500M-row source, the biggest win is to avoid sending heavy queries to the source over and over. **User-defined aggregations** let you create an in-memory aggregation table (at a higher grain like day/product/region). Many visuals can be answered from that small cache instead of scanning the huge fact table, so pages render much faster.

Query caching helps too because Power BI can reuse results for identical queries for a period of time. That's especially useful when users flip between pages or multiple visuals issue similar queries—Power BI can serve results from cache rather than re-querying the backend every time.

Automatic aggregation isn't the best fit here because it's primarily tied to Import models (and Premium features) rather than being the go-to feature for DirectQuery optimization. **OneLake integration** is great for storage and architecture, but it doesn't directly speed up DirectQuery visual queries by itself.

References: https://learn.microsoft.com/en-us/power-bi/transform-model/aggregations-advanced and https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-directquery-about#query-caching