DUMPSARENA

Implementing Data Engineering Solutions Using Microsoft Fabric

Microsoft DP-700

Version Demo

Total Demo Questions: 17

Total Premium Questions: 175

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, Implement and manage an analytics solution	63
Topic 2, Ingest and transform data	76
Topic 3, Monitor and optimize an analytics solution	36
Total	175

QUESTION NO: 1 - (DRAG DROP)

DRAG DROP

You have a Fabric eventhouse that contains a KQL database. The database contains a table named TaxiData. The following is a sample of the data in TaxiData.

VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	PULocationID	DOLocationID	payment_type	total_amount
2	2022-06-06T11:08:32Z	2022-06-06T11:22:17Z	1	0.17	231	50	2	7.12
2	2022-06-06T11:12:05Z	2022-06-06T11:20:43Z	1	1.02	161	163	1	10.56
1	2022-06-06T11:15:00Z	2022-06-06T11:25:32Z	1	1.07	142	230	2	17.12
2	2022-06-06T11:29:54Z	2022-06-06T11:49:34Z	2	2.07	162	236	2	12.01
1	2022-06-06T11:50:50Z	2022-06-06T12:07:24Z	2	2.65	140	142	1	7.89

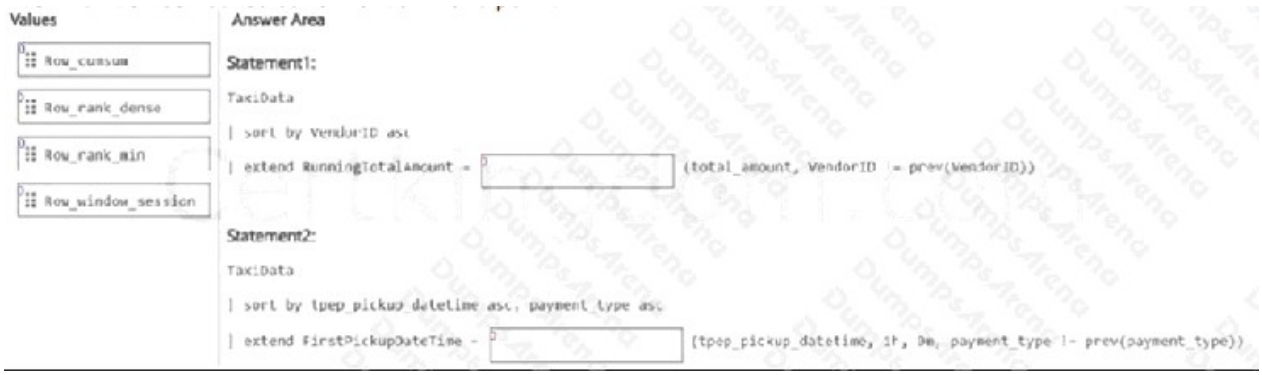
You need to build two KQL queries. The solution must meet the following requirements:

One of the queries must partition RunningTotalAmount by VendorID.

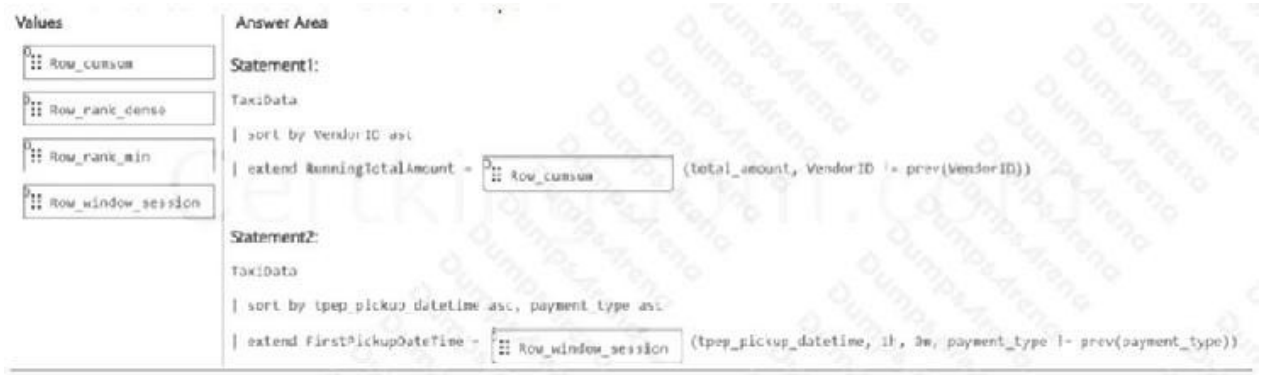
The other query must create a column named FirstPickupDateTime that shows the first value of each hour from tpep_pickup_datetime partitioned by payment_type.

How should you complete each query? To answer, drag the appropriate values the correct targets. Each value 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:

The correct query completion uses `row_cumsum` for `RunningTotalAmount`. In KQL, `row_cumsum` calculates a cumulative sum across serialized rows. Its optional restart predicate makes it especially suitable for a partitioned running total. Here, `total_amount` is the numeric value accumulated, while `VendorID != prev(VendorID)` identifies the boundary between vendor groups. When the vendor value changes from the preceding row, the cumulative calculation restarts. Since the rows are sorted by `VendorID`, all rows for a vendor are processed together and each vendor receives its own running total. Microsoft documents this cumulative-row behavior and restart condition in the row_cumsum documentation.

The correct completion for `FirstPickupDateTime` uses `row_window_session`. This KQL function assigns a session-start value to rows based on a timestamp expression and session boundaries. With `tpep_pickup_datetime` as the timestamp, `1h` as the maximum session duration, and the payment-type comparison as the restart expression, it produces the first pickup datetime for the relevant hourly session and begins a new grouping whenever the payment type changes. The resulting value can therefore be stored directly in `FirstPickupDateTime` and identifies the initial pickup value for each required time window within the payment-type partition. The behavior of session start values, duration controls, and restart predicates is described in Microsoft's row_window_session documentation.

QUESTION NO: 2

You have a Fabric workspace named Workspace1 that uses version control. Workspace! and Azure DevOps are integrated.

In Azure DevOps. developers create a branch named Branch1 to test extract, transform, and load (ETL) updates.

You need to connect Workspace1 to Branch1. The solution must ensure that all the existing content in Branch1 is available in Workspace1.

What should you do?

A. From Workspace1. select Source control, and then select Sync

B. From Workspace1, select Source control, select Current branch, select Branch1, and then switch branches.

C. From Azure DevOps, merge the contents of the main branch into Branch1.

D. From Workspace1, select Source control, and then select Branch out to new workspace

ANSWER: B

Explanation:

Selecting **Source control**, choosing **Current branch**, selecting **Branch1**, and then switching branches is the appropriate action. A Fabric workspace that is already connected to an Azure DevOps repository can change its associated working branch through the branch selector in the Source control experience. When the workspace switches to Branch1, Fabric synchronizes the workspace view with that branch so that the items and definitions already committed in Branch1 become available in Workspace1. This lets developers use the existing branch as the isolated environment for testing ETL changes while retaining Git-based version control and collaboration.

The workspace must be switched to the intended existing branch rather than merely performing a synchronization operation against whichever branch is currently selected. The branch-switching workflow is designed for exactly this scenario: moving a Git-connected Fabric workspace from its current branch to another branch in the same repository and aligning the workspace content with that branch. For more information, see Microsoft Fabric Git integration branching strategies and Git integration process in Microsoft Fabric.

QUESTION NO: 3

You have a Fabric eventstream that receives temperature readings from IoT devices.

You need to retain only readings where the temperature is greater than 80 degrees and calculate the average temperature for each device during one-minute intervals before writing the results to a lakehouse. Which two transform operators should you use? Each correct answer presents part of the solution. NOTE: Each correct selection is worth one point.

A. Filter

B. Aggregate

C. Union

D. Expand

E. Manage fields

ANSWER: A B

Explanation:

The **Filter** transformation retains only events that satisfy a specified condition. Configuring a filter for temperatures greater than 80 removes the unwanted readings before they reach downstream processing and storage.

The **Aggregate** transformation calculates values across multiple events in a configured time window. Configure a one-minute window, group by the device identifier, and select the average aggregation for the temperature field. This produces one average temperature value per device for each interval.

For more information, see Process events by using transformations in Fabric eventstreams.

QUESTION NO: 4

You have a Fabric workspace that contains a warehouse named Warehouse1.

While monitoring Warehouse1, you discover that query performance has degraded during the last 60 minutes.

You need to isolate all the queries that were run during the last 60 minutes. The results must include the username of the users that submitted the queries and the query statements. What should you use?

A. the Microsoft Fabric Capacity Metrics app

B. views from the queryinsights schema

C. Query activity

D. the sys.dm_exec_requests dynamic management view

ANSWER: B

Explanation:

views from the queryinsights schema are designed to provide persisted query-history and performance-troubleshooting data for a Fabric warehouse. In particular, query-insights views expose details about completed requests, including request timing, execution information, the submitting user, and the submitted SQL command text. This makes the data queryable with T-SQL, so an administrator can filter the history to requests whose submission or end time falls within the preceding 60 minutes and return precisely the user names and statements needed for the investigation.

This approach is especially appropriate when performance degradation has already occurred and the goal is to isolate a historical set of queries rather than inspect only the current workload. Query Insights retains historical execution information and supports correlating query text with request-level metadata, enabling targeted analysis of the workload active during the affected period. Microsoft documents the Query Insights schema and its history views for monitoring and troubleshooting Fabric Warehouse query execution at Query Insights in Fabric Data Warehouse. The available Warehouse monitoring capabilities and query-history details are also described in Monitor Fabric Data Warehouse.

QUESTION NO: 5

You have a Fabric warehouse named DW1 that contains a Type 2 slowly changing dimension (SCD) dimension table named DimCustomer. DimCustomer contains 100 columns and 20 million rows.

The columns are of various data types, including int, varchar, date, and varbinary.

You need to identify incoming changes to the table and update the records when there is a change.

The solution must minimize resource consumption.

What should you use to identify changes to attributes?

A. a direct attributes comparison for the attributes in the source table.

B. a hash function to compare the attributes in the DimCustomer table.

C. a direct attributes comparison across the attributes in the DimCustomer table.

D. a hash function to compare the attributes in the source table.

ANSWER: D

Explanation:

A hash function to compare the attributes in the source table is appropriate because it reduces a potentially very wide change comparison to a comparison of one deterministic value. For a Type 2 slowly changing dimension, the incoming source row can have a hash computed from the tracked business attributes, using consistent data-type conversion, column ordering, delimiters, and null handling. That value can then be compared with the hash retained or calculated for the current DimCustomer row. A different hash identifies that the tracked attribute set has changed, so the existing dimension version can be expired and a new version inserted.

This approach is especially beneficial for a dimension with 100 columns and 20 million rows: it avoids repeatedly evaluating a long set of individual attribute predicates during change detection. SQL hashing functionality such as `HASHBYTES` produces a binary hash from input data and supports modern algorithms including SHA2_256, making it suitable for creating a compact change-detection signature. Microsoft's Fabric dimensional-modeling guidance describes Type 2 dimensions as retaining history through multiple versions of a dimension member. See HASHBYTES (Transact-SQL) and Dimensional modeling in Microsoft Fabric.

QUESTION NO: 6 - (DRAG DROP)

DRAG DROP

You are implementing the following data entities in a Fabric environment:

Entity1: Available in a lakehouse and contains data that will be used as a core organization entity

Entity2: Available in a semantic model and contains data that meets organizational standards

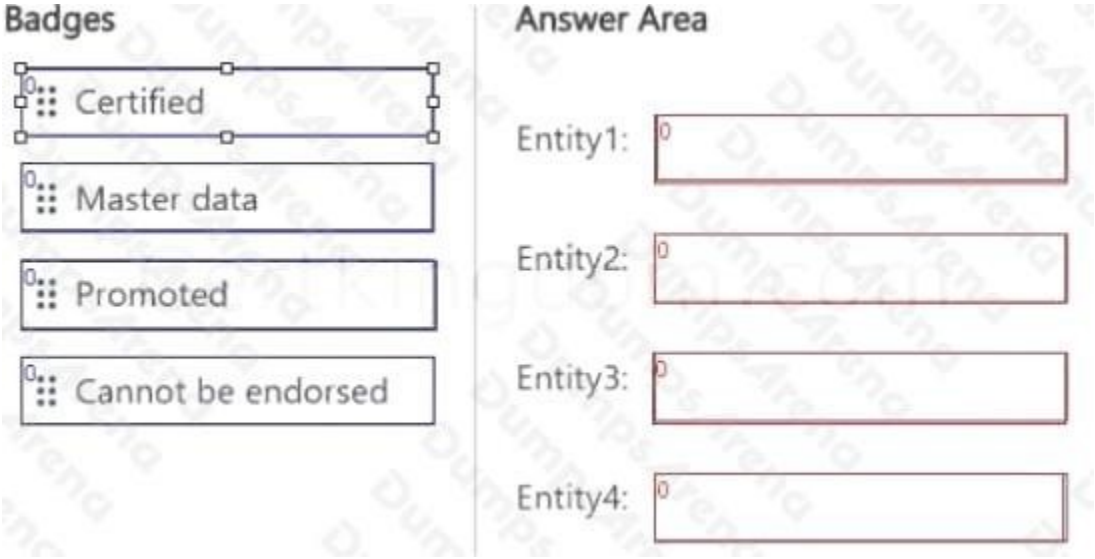
Entity3: Available in a Microsoft Power BI report and contains data that is ready for sharing and reuse Entity4: Available in a Power BI dashboard and contains approved data for executive-level decision making

Your company requires that specific governance processes be implemented for the data.

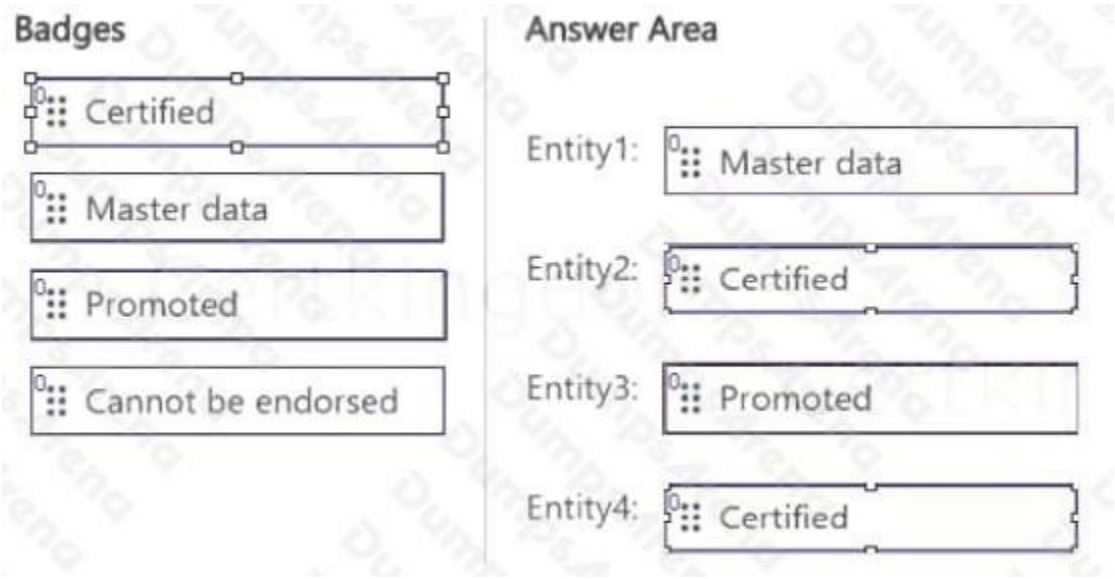
You need to apply endorsement badges to the entities based on each entity's use case.

Which badge should you apply to each entity? To answer, drag the appropriate badges the correct entities. Each badge 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:

Endorsement badges make the intended level of trust and reuse clear to people discovering Fabric and Power BI content. Entity1 should receive the **Master data** badge because it is a core organizational entity held in a lakehouse. Master data represents the shared business entities that provide a consistent organizational foundation, such as customer, product, employee, or location information. Identifying that entity as master data communicates that it is an authoritative, reusable business record rather than simply a useful individual asset.

Entity2 should receive the **Certified** badge. A semantic model that meets organizational standards has the formal quality and governance characteristics expected of certified content. Certification identifies content that an authorized certifier has reviewed and approved as a trusted source for use across the organization. Microsoft describes certification as the organization's authoritative signal for content that meets defined quality requirements. See Endorsement in Power BI.

Entity3 should receive the **Promoted** badge because the report is ready for sharing and reuse. Promotion lets content owners highlight useful, valuable content so that other users can find and confidently reuse it. It is the appropriate discovery and reuse signal for a report that is ready to be shared broadly.

Entity4 should receive the **Certified** badge. Since the dashboard contains approved data used for executive-level decisions, the badge needs to convey a formally governed and trusted source. Certified content provides that clear organizational trust signal, helping decision-makers identify approved content for important reporting and analysis. For Fabric governance concepts, see Microsoft Fabric governance overview.

QUESTION NO: 7

You have an Azure subscription that contains a blob storage account named sa1. Sa1 contains two files named Filelxsv and File2.csv.

You have a Fabric tenant that contains the items shown in the following table.

You need to configure Pipeline1 to perform the following actions:

• At 2 PM each day, process Filel.csv and load the file into flhl.

• At 5 PM each day. process File2.csv and load the file into flhl.

The solution must minimize development effort. What should you use?

A. a job definition

B. a data pipeline schedule

C. a data pipeline trigger

D. an activator

ANSWER: B

Explanation:

a data pipeline schedule is the appropriate choice because Fabric Data Factory lets you schedule a pipeline to run automatically at defined times. Pipeline scheduling is intended for recurring, time-based orchestration, such as executing ingestion work every day at specific hours. Pipeline1 can use its existing copy or ingestion logic to load the specified source file into the target lakehouse, while schedules define when the executions occur. Configuring daily schedule entries for 2 PM and 5 PM minimizes development effort because it uses built-in orchestration rather than requiring custom code or a separate event-processing solution. The pipeline can also be designed to accept a file-name parameter, allowing the scheduled executions to supply the relevant file name while reusing the same ingestion implementation. Fabric supports scheduling pipeline runs through the pipeline scheduling experience, including recurrence configuration and start times. This directly matches a requirement based on predictable clock times rather than a condition occurring in a data stream. See Schedule a pipeline run in Microsoft Fabric and Parameters in Microsoft Fabric Data Factory.

QUESTION NO: 8

You are building a Fabric notebook named MasterNotebookl in a workspace. MasterNotebookl contains the following code.

```
DAG = {
    "activities": [
        {   "name": "execute_notebook_1",
            "path": "notebook_01",
            "timeoutPerCellInSeconds": 600,
            "args": {
                        "input_value": "999"
                },
            "retry": 1,
            "retryIntervalInSeconds": 30

        },
        {   "name": "execute_notebook_2",
            "path": "notebook_02",
            "timeoutPerCellInSeconds": 400,
            "args": {
                        "input_value": "888"
                    },
            "retry": 1,
            "retryIntervalInSeconds": 30
        },
        {   "name": "execute_notebook_3",
            "path": "notebook_03",
            "timeoutPerCellInSeconds": 600,
            "args": {
                        "input_value": "777"
                    },
            "retry": 1,
            "retryIntervalInSeconds": 30
        }
        ],
    "timeoutInSeconds": 43200,
    "concurrency": 0
}
```

You need to ensure that the notebooks are executed in the following sequence:

1. Notebook_03

2. Notebook.Ol

3. Notebook_02

Which two actions should you perform? Each correct answer presents part of the solution. NOTE: Each correct selection is worth one point.

A. Split the Directed Acyclic Graph (DAG) definition into three separate definitions.

B. Change the concurrency to 3.

C. Move the declaration of Notebook_03 to the top of the Directed Acyclic Graph (DAG) definition.

D. Move the declaration of Notebook_02 to the bottom of the Directed Acyclic Graph (DAG) definition.

E. Add dependencies to the execution of Note boo k_02.

F. Add dependencies to the execution of Notebook_03.

G. Add a dependency to the execution of Notebook_01 so that it depends on Notebook_03.

ANSWER: E G

Explanation:

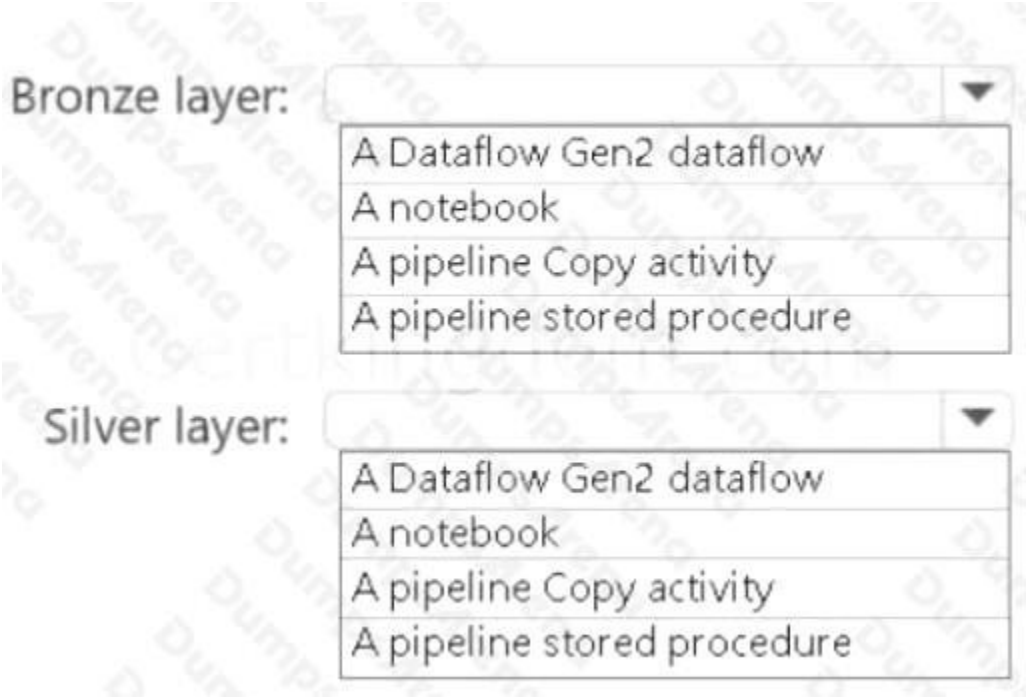
To enforce the required order, define explicit DAG dependencies: Notebook_01 must depend on Notebook_03, and Notebook_02 must depend on Notebook_01. Therefore, add a dependency to the Notebook_01 activity that names `Notebook_03`, and add a dependency to the Notebook_02 activity that names `Notebook_01`. Fabric evaluates activities with no unmet dependencies as runnable, and starts a dependent activity only after all activities listed in its `dependencies` collection complete successfully. This produces the required chain: Notebook_03 runs first, completing it makes Notebook_01 eligible to run, and completion of Notebook_01 then makes Notebook_02 eligible to run. The dependency declarations express the execution requirement directly and remain reliable regardless of scheduling behavior or the order in which activities are listed in the DAG definition. Microsoft documents that `mssparkutils.notebook.runMultiple()` supports running notebooks through a DAG and that dependencies control execution relationships between activities. See Run multiple notebooks with Microsoft Spark utilities and Fabric notebook utilities documentation.

QUESTION NO: 9 - (HOTSPOT)

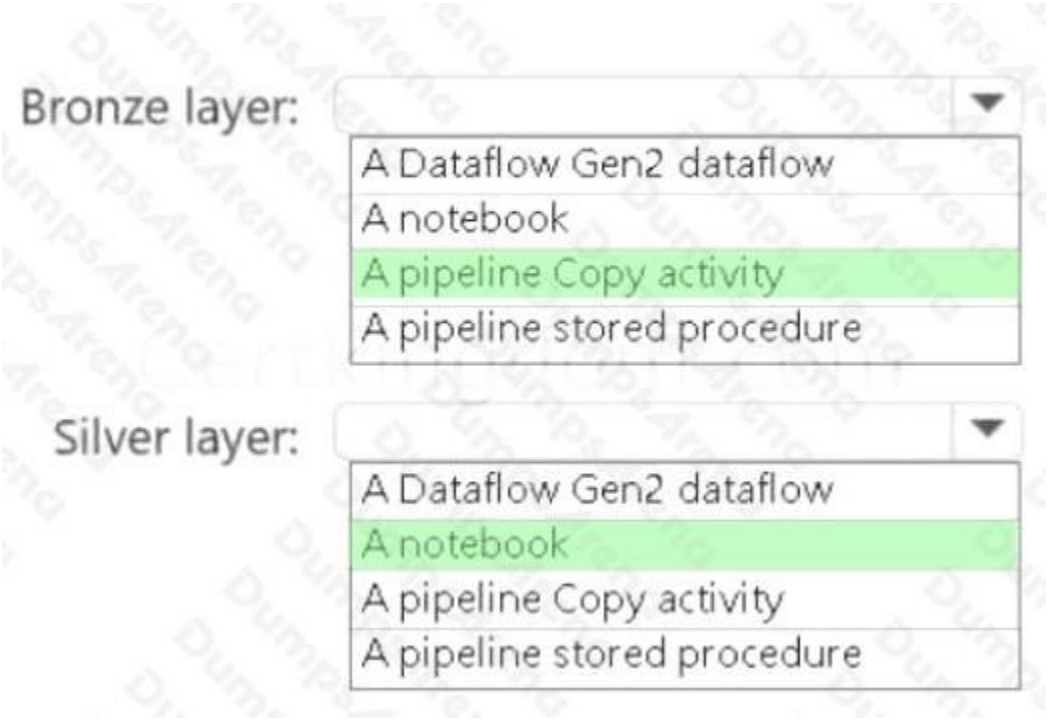
HOTSPOT

You need to recommend a method to populate the POS1 data to the lakehouse medallion layers.

What should you recommend for each layer? To answer, select the appropriate options in the answer area. NOTE: Each correct selection is worth one point.



ANSWER:



Explanation:

A pipeline Copy activity is the appropriate method for the Bronze layer because this layer is the raw ingestion zone of a medallion architecture. POS1 data should be transferred from its source into the lakehouse efficiently and retained with minimal alteration, preserving the source data for traceability and later reprocessing. In Microsoft Fabric, Copy activity is designed for orchestrated data movement between supported sources and destinations, including Lakehouse storage. It provides a managed way to schedule, monitor, and run ingestion as part of a Fabric Data Factory pipeline. Microsoft documents this capability in Copy activity in Microsoft Fabric.

A notebook is the appropriate method for the Silver layer because Silver is where the raw Bronze data is refined into clean, validated, and structured data. A Fabric notebook can use Spark and Delta Lake operations to apply data-quality rules, standardize column values and schemas, remove or handle invalid records, deduplicate data, and merge incremental changes into curated Silver tables. Notebooks are well suited to this work because they provide flexible, code-based transformation logic and can read and write Lakehouse Delta tables directly. Fabric notebooks support Spark-based engineering workflows as described in How to use notebooks in Microsoft Fabric. This separation keeps ingestion simple in Bronze while making the Silver layer a dependable, analytics-ready foundation for downstream Gold-level reporting and business aggregations.

QUESTION NO: 10

You need to create a workflow for the new book cover images.

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

NOTE: Each correct selection is worth one point.

A. a notebook that uses Apache Spark Structured Streaming

B. a time-based schedule

C. an activator item

D. a data pipeline

E. a streaming dataflow

F. a blob storage action

ANSWER: E F

Explanation:

A blob storage action and a streaming dataflow provide the event-driven ingestion and processing pieces needed for a workflow that responds to newly added book-cover images. The blob storage action captures the creation of a new blob as an event, so processing can begin when an image arrives rather than waiting for a periodic polling interval. This is appropriate when the workflow must react to individual image arrivals and preserve the event context, such as the blob name, path, or creation time.

A streaming dataflow then provides the low-code stream-processing layer for the incoming event data. It can continuously transform, filter, enrich, and route the event records for downstream use in Fabric. In practice, the blob-created event identifies the image that has arrived, and the streaming dataflow processes the associated event stream as events are received. This combination supports a responsive, near-real-time workflow for new image arrivals. Fabric Eventstreams supports event-driven streaming architectures and integrates with streaming dataflows for continuous transformation and routing. See the Microsoft Fabric Eventstreams overview and the Microsoft Fabric streaming dataflows overview.

QUESTION NO: 11

You need to recommend a solution for handling old files. The solution must meet the technical requirements. What should you include in the recommendation?

A. a data pipeline that includes a Copy data activity

B. a notebook that runs the VACUUM command

C. a notebook that runs the OPTIMIZE command

D. a data pipeline that includes a Delete data activity

ANSWER: B

Explanation:

A notebook that runs the VACUUM command is appropriate because Delta tables retain obsolete data files after operations such as updates, deletes, merges, and file rewrites. These files are not part of the current version of the table, but they may remain in OneLake during the configured retention period to support Delta time travel and concurrent operations. VACUUM performs Delta-aware cleanup: it identifies files that are no longer referenced by the transaction log and removes only those files that are eligible under the retention policy. Running the command in a Fabric notebook also makes the cleanup operationally repeatable and suitable for scheduling as part of ongoing lakehouse maintenance. The retention period must be selected carefully because reducing it can limit the ability to query earlier table versions. Microsoft recommends table maintenance practices that combine file optimization when needed with vacuuming to manage obsolete files and storage consumption. See Optimize and vacuum tables in Microsoft Fabric and Lakehouse and Delta tables in Microsoft Fabric.

QUESTION NO: 12

You have a Fabric notebook named Notebook1 that performs data-quality validation. Notebook1 is executed by a data pipeline named Pipeline1.

You need Pipeline1 to run different downstream activities depending on whether the validation succeeds or fails. Which two actions should you perform? Each correct answer presents part of the solution. NOTE: Each correct selection is worth one point.

A. Use notebookutils.notebook.exit() to return the validation result.

B. Add an If Condition activity that evaluates the notebook activity exit value.

C. Configure a schedule directly on Notebook1.

D. Use a Copy data activity to evaluate the notebook result.

E. Enable high concurrency for Notebook1.

ANSWER: A B

Explanation:

Use `notebookutils.notebook.exit()` in Notebook1 to return a value when the notebook execution completes. The returned value becomes the notebook activity exit value available to the pipeline run.

Add an If Condition activity to Pipeline1 and use the notebook activity output exit value in the condition expression. The If Condition activity can then direct execution to the appropriate true or false branch for successful and failed validation outcomes.

For more information, see Notebook utilities in Microsoft Fabric and If Condition activity in Fabric Data Factory.

QUESTION NO: 13 - (SIMULATION)

You have two Fabric notebooks named Load_Salesperson and Load_Orders that read data from Parquet files in a lakehouse. Load_Salesperson writes to a Delta table named dim_salesperson. Load.Orders writes to a Delta table named fact_orders and is dependent on the successful execution of Load_Salesperson.

You need to implement a pattern to dynamically execute Load_Salesperson and Load_Orders in the appropriate order by using a notebook.

How should you complete the code? To answer, drag the appropriate values the correct targets. Each value 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: See the explanation for the answer

Explanation:

Use `mssparkutils.notebook.runMultiple()` with a DAG. Define `Load_Orders` with a dependency on `Load_Salesperson`.

```
DAG = {
    "activities": [
        {
            "name": "Load_Salesperson",
            "path": "Load_Salesperson"
        },
        {
            "name": "Load_Orders",
            "path": "Load_Orders",
            "dependencies": ["Load_Salesperson"]
        }
    ]
}
```

`mssparkutils.notebook.runMultiple(DAG)`
For the first activity, use `Load_Salesperson` as both the activity `name` and notebook `path`.
For the second activity, use `Load_Orders` as both the activity `name` and notebook `path`.
Set `dependencies` for `Load_Orders` to `["Load_Salesperson"]`.
Invoke the DAG by using `mssparkutils.notebook.runMultiple(DAG)`.

This ensures that the orders notebook runs only after the salesperson notebook completes successfully.

QUESTION NO: 14

You have a Fabric warehouse named Warehouse1 that contains a table named Sales.

You need to implement row-level security so that each sales representative can query only the rows assigned to their Microsoft Entra ID account.

Which two objects should you create? Each correct answer presents part of the solution. NOTE: Each correct selection is worth one point.

A. FUNCTION

B. SECURITY POLICY

C. VIEW

D. CONSTRAINT

E. DATABASE SCOPED CREDENTIAL

ANSWER: A B

Explanation:

A row-level security implementation in a Fabric warehouse requires an inline table-valued **FUNCTION** and a **SECURITY POLICY**. The function contains the filter predicate that evaluates the current user and determines whether each row is accessible. For example, the function can compare the current user identity to a sales representative mapping.

The security policy applies the function to the protected table as a filter predicate. This causes the predicate to be evaluated automatically whenever a user queries Sales. A view does not provide enforced row-level security, and granting SELECT controls object access but does not filter rows. For more information, see Row-level security in Fabric Data Warehouse.

QUESTION NO: 15

You have a Fabric workspace that contains a lakehouse named Lakehouse1. Lakehouse1 contains a

Delta table named Table1.

You analyze Table1 and discover that Table1 contains 2,000 Parquet files of 1 MB each. You need to minimize how long it takes to query Table1. What should you do?

A. Disable V-Order and run the OPTIMIZE command.

B. Disable V-Order and run the VACUUM command.

C. Run the OPTIMIZE and VACUUM commands.

ANSWER: C

Explanation:

Run the OPTIMIZE and VACUUM commands. is the correct choice because the table has a small-files problem. Reading 2,000 individual 1 MB Parquet files creates substantial metadata, file-listing, task-scheduling, and file-opening overhead. That overhead can dominate query duration even though the table's total data volume is relatively small.

In a Fabric lakehouse, `OPTIMIZE` compacts many small Delta/Parquet data files into fewer, larger files. This reduces the number of files the SQL analytics endpoint or Spark engine must discover and read, improving scan efficiency and therefore reducing query time. Fabric supports optimized writes and table maintenance for Delta tables, including compaction through `OPTIMIZE`. V-Order is enabled by default for Fabric lakehouse tables and should be retained so rewritten files receive Fabric's read-performance optimizations.

After compaction, `VACUUM` removes obsolete files that are no longer referenced by the Delta table transaction log, subject to the configured retention period. Running it as part of table maintenance keeps the storage layout clean after the rewrite. See Lakehouse table maintenance in Microsoft Fabric and Delta optimization and V-Order in Microsoft Fabric.

QUESTION NO: 16

You have a Fabric workspace that contains a takehouse and a semantic model named Model1.

You use a notebook named Notebook1 to ingest and transform data from an external data source. You need to execute Notebook1 as part of a data pipeline named Pipeline1. The process must meet the following requirements:

Run daily at 07:00 AM UTC.

Attempt to retry Notebook1 twice if the notebook fails.

After Notebook1 executes successfully, refresh Model1.

Which three actions should you perform? Each correct answer presents part of the solution. NOTE: Each correct selection is worth one point.

A. Set the Retry setting of the Notebook activity to 2.

B. Place the Semantic model refresh activity after the Notebook activity and link the activities by using an On completion condition.

C. Place the Semantic model refresh activity after the Notebook activity and link the activities by using the On success condition.

D. From the Schedule settings of Notebook1, set the time zone to UTC.

E. From the Schedule settings of Pipeline1, set the time zone to UTC.

F. Set the Retry setting of the Semantic model refresh activity to 2.

ANSWER: A C E

Explanation:

Set the Retry setting of the Notebook activity to 2. Pipeline activities in Fabric have retry behavior that is configured on the individual activity. A retry value of 2 causes the notebook activity to be retried up to two times after a failure, directly applying the resiliency requirement to Notebook1.

Place the Semantic model refresh activity after the Notebook activity and link the activities by using the On success condition. Pipeline dependencies control when a downstream activity can start. An On success dependency ensures that the semantic model refresh begins only after Notebook1 has completed successfully, preserving the required execution order and preventing refresh before successful ingestion and transformation.

From the Schedule settings of Pipeline1, set the time zone to UTC. The schedule belongs to the pipeline that orchestrates the notebook and semantic model refresh activities. Configuring its time zone as UTC ensures that a daily trigger configured for 07:00 runs at 07:00 UTC consistently. For details, see Run and schedule pipelines in Microsoft Fabric and Notebook activity in Microsoft Fabric.

QUESTION NO: 17 - (HOTSPOT)

HOTSPOT

You plan to process the following three datasets by using Fabric:

Dataset1: This dataset will be added to Fabric and will have a unique primary key between the source and the destination. The unique primary key will be an integer and will start from 1 and have an increment of 1.

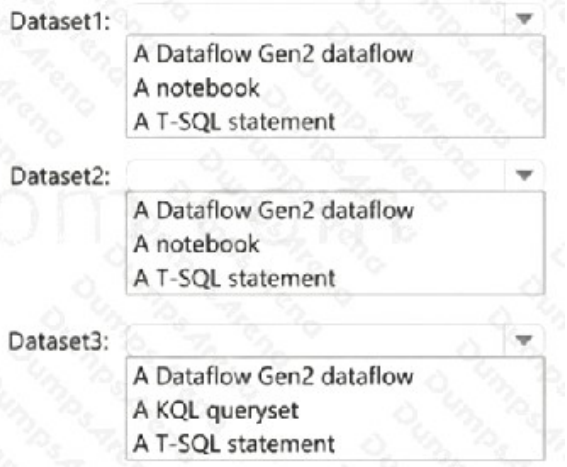
Dataset2: This dataset contains semi-structured data that uses bulk data transfer. The dataset must be handled in one process between the source and the destination. The data transformation process will include the use of custom visuals to understand and work with the dataset in development mode.

Dataset3. This dataset is in a takehouse. The data will be bulk loaded. The data transformation process will include row-based windowing functions during the loading process.

You need to identify which type of item to use for the datasets. The solution must minimize

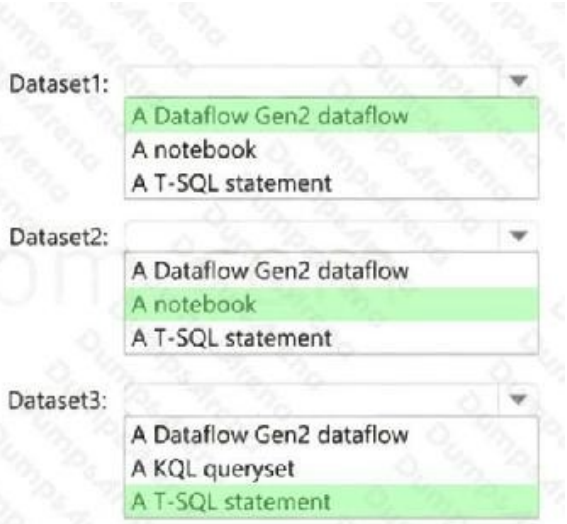
development effort and use built-in functionality, when possible. What should you identify for each dataset? To answer, select the appropriate options in the answer area. NOTE: Each correct selection is worth one point.

Answer Area



ANSWER:

Answer Area



Explanation:

Use a Dataflow Gen2 dataflow for Dataset1, a notebook for Dataset2, and a T-SQL statement for Dataset3. A Dataflow Gen2 dataflow is the best fit for Dataset1 because it uses the low-code Power Query experience and includes an Index Column transformation. This transformation can generate the required integer key directly in the data-preparation flow, with a starting value of 1 and an increment of 1. That provides a consistent generated key as the data moves from the source into Fabric, while minimizing custom development. The Power Query index-column feature is documented at Add an index column, and Dataflow Gen2 is designed to prepare and load data into Fabric destinations.

A notebook is appropriate for Dataset2 because notebooks provide a single development surface for reading, transforming, and writing semi-structured bulk data. They also support interactive development and custom visualizations, allowing engineers to inspect data shapes, validate transformation logic, and explore results during development. Fabric notebooks support Python, Scala, SQL, and R workloads, as described in How to use Microsoft Fabric notebooks.

A T-SQL statement is appropriate for Dataset3 because the data is loaded into a lakehouse and the required transformation includes row-based window functions as part of the load process. Fabric SQL supports standard windowing expressions such as ROW_NUMBER, RANK, LAG, and LEAD, enabling ordered, partition-aware calculations directly in the transformation query. The relevant syntax and behavior are described in SELECT - OVER clause.