PassQueen
a.k.a dumpsqueen.com

Implementing Data Engineering Solutions Using Azure Databricks

Microsoft DP-750

Version Demo

Total Demo Questions: 10

Total Premium Questions: 106

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Implement data ingestion and transformation solutions	62
Topic 2, Manage and optimize data engineering workloads	44
Total	106

QUESTION NO: 1 - (SIMULATION)

You have an Azure Databricks workspace that is enabled for Unity Catalog and contains a catalog named finance, finance contains two schemas named default and procurement.

You need to create a table named assets in the procurement schema, assets must contain the following columns:

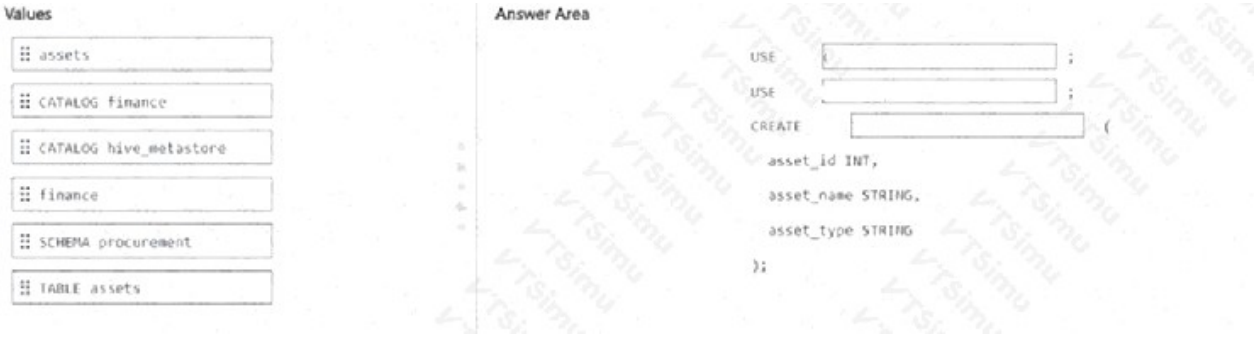
• asset.id

• asset, type

• asset_name

How should you complete the SQL statement? To answer, drag the appropriate values to 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:

Place the values as follows:

This selects the `finance` catalog and its `procurement` schema before creating the `assets` table.

QUESTION NO: 2

You have an Azure Databricks workspace.

Users report that a Databricks notebook that runs each day takes longer than expected to run.

When reading the Directed Acyclic Graph (DAG), you discover the following issues concerning the Apache Spark stage:

• Most tasks in the stage finish quickly.

• A few tasks in the stage run more slowly.

• The CPU is underutilized at the end of the stage.

• The slow tasks process many more input records.

• The stage is blocked while it waits for the few slow tasks.

What is the root cause of the issues?

A. caching

B. shuffling

C. spilling

D. skewing

ANSWER: D

Explanation:

skewing is the root cause because the tasks in a Spark stage process separate data partitions, and the reported input-record imbalance shows that a small number of partitions are much larger than the rest. This is data skew: a key distribution, join, aggregation, or partitioning operation has concentrated disproportionately many records into only a few partitions. The tasks assigned to normal-sized partitions complete early, but Spark cannot mark the stage complete until every task has finished. Consequently, executors that completed their work become idle and cluster CPU utilization drops while the remaining straggler tasks process the oversized partitions. The stage duration is therefore determined by the slowest partitions rather than by the average task duration. Azure Databricks documents that skew occurs when data is unevenly distributed across partitions and that Adaptive Query Execution can dynamically optimize skewed joins by splitting skewed partitions. Reviewing task-level input sizes and durations in the Spark UI is an appropriate way to identify this pattern and validate the imbalance. See Dynamically handle skew join in Azure Databricks and Troubleshoot slow jobs in Azure Databricks.

QUESTION NO: 3

You have an Azure Databricks workspace that is enabled for Unity Catalog. A group named DataReaders must query a managed Delta table named production.curated.Customers.

DataReaders currently has no Unity Catalog privileges.

Which three privileges must you grant to DataReaders? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. USE CATALOG on production

B. USE SCHEMA on production.curated

C. SELECT on production.curated.Customers

D. MODIFY on production.curated.Customers

E. CREATE TABLE on production.curated

F. READ FILES on production.curated.Customers

ANSWER: A B C

Explanation:

Grant `USE CATALOG` on production, `USE SCHEMA` on curated, and `SELECT` on Customers. Unity Catalog requires users to traverse the catalog and schema namespaces before accessing an object. The SELECT privilege authorizes reading the table data.

`MODIFY` permits write operations and is not required for read-only access. `CREATE TABLE` permits creating tables in a schema, while `READ FILES` applies to governed file access through an external location or volume rather than querying an existing managed table.

QUESTION NO: 4

You have an Azure Databricks workspace that is enabled for Unity Catalog and contains a Delta table named Orders

You load the Orders table into an Apache Spark DataFrame named df.

You need to create a DataFrame that excludes rows where the order amount is null.

Solution: You run the following expression.

df-fillna(0, subset=[' order_amount '])

Does this meet the goal?

A. Yes

B. No

ANSWER: B

Explanation:

No is correct because the stated goal is to remove records whose `order_amount` value is null, not to assign those records a replacement value. In PySpark, `DataFrame.fillna()` is intended to replace null or missing values with a supplied value. Even if the expression were written with valid method-call syntax as `df.fillna(0, subset=["order_amount"])`, it would retain every row and replace null values in the specified column with zero. Therefore, it does not create a DataFrame that excludes the null-order rows.

To meet the requirement, use a row-filtering operation such as `df.dropna(subset=["order_amount"])`, which returns a new DataFrame after dropping rows that contain null values in the specified subset of columns. Alternatively, a filter using `col("order_amount").isNotNull()` expresses the same inclusion criterion. Spark DataFrames are immutable, so either operation produces a transformed DataFrame that must be assigned to a variable if it will be used later. See the PySpark documentation for DataFrame.dropna and DataFrame.fillna.

QUESTION NO: 5 - (HOTSPOT)

You have an Azure Databricks workspace that is enabled for Unity Catalog and contains a Delta table named db1.sales_orders.

dbl sales_orders is updated nightly and has change data feed (CDF) enabled.

You need to ingest all the changes from the dbl.sales.ordets table, including inserts, updates, and deletes, into a downstream pipeline.

How should you complete the PsySpark code segment? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

Answer Area

ANSWER:

Explanation:

Change Data Feed is the Delta Lake feature intended for downstream change-processing scenarios. A streaming reader must enable it with `readChangeFeed` set to `true`. With this setting, the stream reads the table's recorded data changes and supplies change metadata, including the `_change_type` column. That metadata identifies whether each emitted record represents an insert, delete, update preimage, or update postimage, allowing the downstream pipeline to apply the operation correctly.

The pipeline must also retain delete events because the requirement explicitly says to ingest inserts, updates, and deletes. Setting `ignoreDeletes` to `false` means delete activity is not filtered out. This makes the intent of the stream explicit: it is consuming the full set of relevant changes rather than omitting delete operations. Together, the completed reader is conceptually `spark.readStream.format("delta").option("readChangeFeed", "true").option("ignoreDeletes", "false").table("db1.sales_orders")`.

Delta Change Data Feed is designed to make incremental processing efficient by reading changes made after a chosen version or timestamp, rather than repeatedly processing the entire table. A production pipeline can additionally use a checkpoint and, when required, a `startingVersion` or `startingTimestamp` to define its initial position. Microsoft's documentation describes enabling CDF reads through the `readChangeFeed` option and the change records returned by the feature: Use Delta Lake change data feed on Azure Databricks. The related streaming behavior and available Delta options are documented at Stream from Delta Lake.

QUESTION NO: 6

You have an Azure Databricks workspace that is enabled for Unity Catalog.

You have 500 GB of sales data stored as multiple CSV files in cloud storage.

You plan to load the data into a Delta table.

You need to ingest the bulk data by using a solution that meets the following requirements:

• Minimize how long it takes to implement the solution.

• Minimize the amount of custom code required.

What should you use?

A. Auto Loader

B. Apache Spark read APIs

C. file upload

D. COPY INTO

ANSWER: D

Explanation:

`COPY INTO` is the appropriate choice for a simple, bulk ingestion of CSV files from cloud storage into an existing Delta table when fast implementation and minimal custom code are priorities. It is a SQL command, so the load can be implemented with a concise statement that identifies the target Delta table, the source storage location, and CSV format options. This avoids building and maintaining a separate Spark application or a streaming ingestion pipeline.

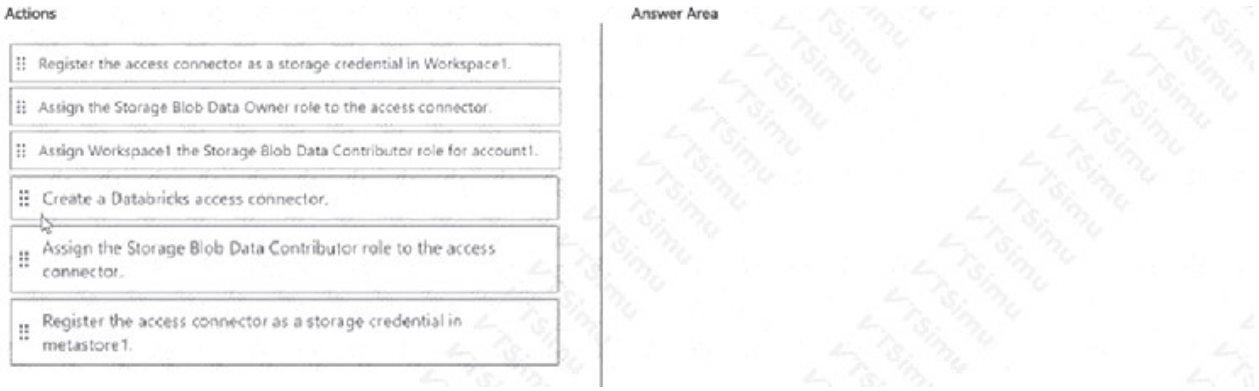
For this scenario, `COPY INTO` also provides operational safety: Databricks records which source files have been loaded and skips those files on subsequent executions. This makes the operation idempotent, enabling retries without unintentionally loading the same files again. The command supports loading from cloud object storage and common file formats, including CSV, directly into Delta Lake tables. Unity Catalog can govern access to the destination table and, where configured, the storage location through external locations and storage credentials. See COPY INTO for Azure Databricks and COPY INTO examples.

QUESTION NO: 7 - (DRAG DROP)

You have an Azure Databricks workspace named Workspace1 that is attached to a Unity Catalog metastore named metastore1

You need to register an Azure Storage account named account1 that has a hierarchical namespace enabled as an external location The external location must use a managed identity to authenticate to account1 and the solution must follow the principle of least privilege.

Which three 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:

To use a managed identity for an Azure Data Lake Storage Gen2 external location, first create a Databricks access connector. The connector is the Azure resource that provides the managed identity Unity Catalog will use when it accesses account1. Creating it first is essential because Azure role assignments must be applied to an existing identity.

Next, grant that access connector the **Storage Blob Data Contributor** role. This gives the managed identity the required Azure Storage data-plane permissions to read, write, and list data through the external location. To follow least privilege, the assignment should be scoped as narrowly as practical, normally to the container that will be represented by the external location rather than broadly granting permissions across unrelated resources. The connector identity, rather than the workspace itself, is the identity that needs the storage permission.

Finally, register the access connector as a storage credential in metastore1. A Unity Catalog storage credential is a metastore-level securable object that stores the authorization configuration used to reach cloud storage. Referencing the access connector in this credential enables Unity Catalog to use its managed identity without storing a storage account key or SAS token. Afterward, an external location can reference that storage credential and a specific ADLS Gen2 path. Microsoft documents this managed-identity pattern for Unity Catalog external locations in Use Azure managed identities with Unity Catalog and describes storage credentials and external locations in Manage external locations and storage credentials.

QUESTION NO: 8 - (HOTSPOT)

You have an Azure Databricks workspace that is enabled for Unity Catalog.

You need to create an external volume named Volume1 in an existing schema. Volume1 must expose files from an Azure Storage container. The solution must meet the following requirements:

• Ensure that authentication does NOT require storing credentials in Databricks

• Ensure that users can access the files, but NOT modify the files.

• Follow the principle of least privilege

Which type of authentication should you configure, and which permission should you grant to the users? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

ANSWER:

Explanation:

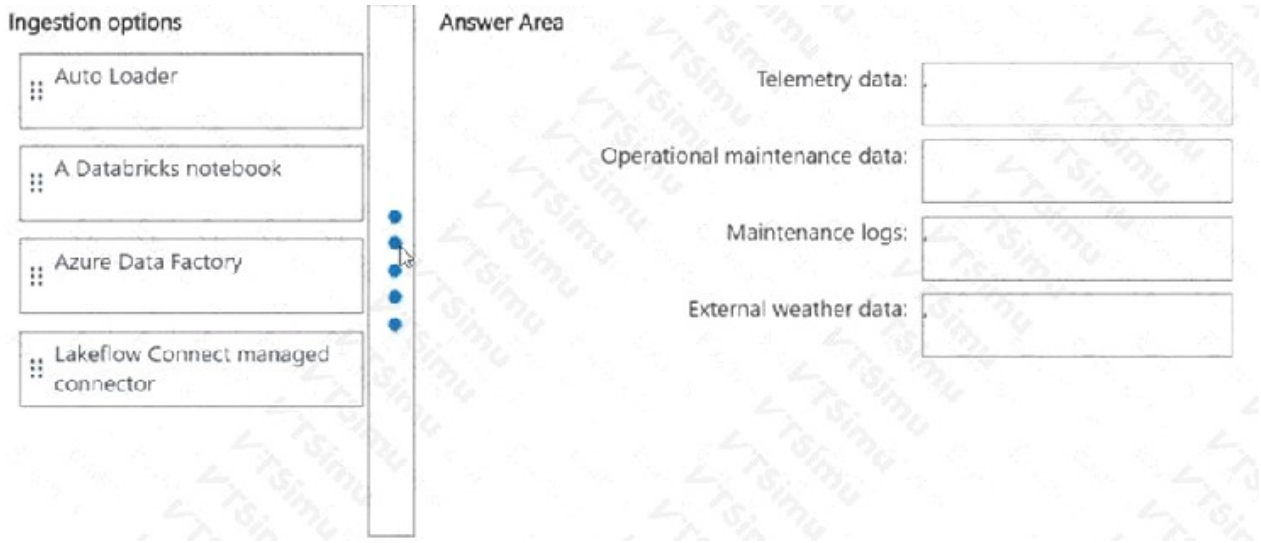
An Azure Databricks access connector is the appropriate authentication mechanism because it uses an Azure managed identity. The access connector can be used by a Unity Catalog storage credential to authorize Databricks access to the Azure Storage container that backs the external location and external volume. Azure manages the identity, so the configuration does not require placing a storage account key, SAS token, or client secret in Databricks. Permissions for that managed identity are assigned on the Azure side, using the appropriate Azure Storage RBAC role or other supported storage authorization configuration. This gives the organization centralized identity governance while avoiding the operational burden and security risk of managing long-lived secrets in the workspace.

Granting **READ VOLUME** is the least-privileged Unity Catalog permission that meets the user requirement. It authorizes users to access a volume's files for reading and to list the content available through the volume path. Because the users only need to consume the files, read-level volume access is sufficient; no file-creation, overwrite, deletion, or other modification capability is required. The permission is granted on the volume itself, so access can be scoped specifically to Volume1 rather than broadly to all storage resources or all objects in the schema. This combines secretless Azure authentication with narrowly scoped Unity Catalog authorization. Microsoft documents the access connector and managed identity pattern for Azure Databricks at Access cloud storage using managed identities. Databricks documents the Unity Catalog volume privileges, including READ VOLUME, at Unity Catalog privileges and securable objects.

QUESTION NO: 9 - (SIMULATION)

Which ingestion option should you recommend for each data source? To answer, drag the appropriate options to the correct data sources. Each option 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:

Drag the ingestion options as follows:

A Databricks notebook can implement Structured Streaming for telemetry sources such as Event Hubs. Lakeflow Connect provides managed ingestion for supported operational database sources. Auto Loader incrementally detects and ingests newly arriving log files from cloud storage. Azure Data Factory is appropriate for orchestrated ingestion from an external weather/REST data source.

QUESTION NO: 10

You use Databricks Asset Bundles to deploy a Lakeflow Jobs workflow to development and production targets.

A CI/CD pipeline must prevent an invalid bundle configuration from being deployed and must deploy the approved configuration to the production target.

Which two Databricks CLI commands should the pipeline run? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. `databricks bundle validate`

B. `databricks bundle deploy -t prod`

C. `databricks bundle init`

D. `databricks bundle run -t prod`

E. `databricks bundle destroy -t prod`

ANSWER: A B

Explanation:

Run `databricks bundle validate` and `databricks bundle deploy -t prod`. The validate command checks the bundle configuration before deployment, including resolved configuration for the selected environment. The deploy command provisions or updates the resources declared by the bundle for the production target.

`databricks bundle init` creates a new local bundle project and is not a deployment operation. `databricks bundle run` runs a resource that has already been deployed, while `databricks bundle destroy` removes deployed resources.