

# Databricks

## Exam Questions Databricks-Certified-Professional-Data-Engineer

Databricks Certified Data Engineer Professional Exam



**NEW QUESTION 1**

A Databricks job has been configured with 3 tasks, each of which is a Databricks notebook. Task A does not depend on other tasks. Tasks B and C run in parallel, with each having a serial dependency on Task A.

If task A fails during a scheduled run, which statement describes the results of this run?

- A. Because all tasks are managed as a dependency graph, no changes will be committed to the Lakehouse until all tasks have successfully been completed.
- B. Tasks B and C will attempt to run as configured; any changes made in task A will be rolled back due to task failure.
- C. Unless all tasks complete successfully, no changes will be committed to the Lakehouse; because task A failed, all commits will be rolled back automatically.
- D. Tasks B and C will be skipped; some logic expressed in task A may have been committed before task failure.
- E. Tasks B and C will be skipped; task A will not commit any changes because of stage failure.

**Answer: D**

**Explanation:**

When a Databricks job runs multiple tasks with dependencies, the tasks are executed in a dependency graph. If a task fails, the downstream tasks that depend on it are skipped and marked as Upstream failed. However, the failed task may have already committed some changes to the Lakehouse before the failure occurred, and those changes are not rolled back automatically. Therefore, the job run may result in a partial update of the Lakehouse. To avoid this, you can use the transactional writes feature of Delta Lake to ensure that the changes are only committed when the entire job run succeeds.

Alternatively, you can use the Run if condition to configure tasks to run even when some or all of their dependencies have failed, allowing your job to recover from failures and

continue running. References:

? transactional writes: <https://docs.databricks.com/delta/delta-intro.html#transactional-writes>

? Run if: <https://docs.databricks.com/en/workflows/jobs/conditional-tasks.html>

**NEW QUESTION 2**

A junior data engineer seeks to leverage Delta Lake's Change Data Feed functionality to create a Type 1 table representing all of the values that have ever been valid for all rows in a bronze table created with the property `delta.enableChangeDataFeed = true`. They plan to execute the following code as a daily job:

Which statement describes the execution and results of running the above query multiple times?

- A. Each time the job is executed, newly updated records will be merged into the target table, overwriting previous values with the same primary keys.
- B. Each time the job is executed, the entire available history of inserted or updated records will be appended to the target table, resulting in many duplicate entries.
- C. Each time the job is executed, the target table will be overwritten using the entire history of inserted or updated records, giving the desired result.
- D. Each time the job is executed, the differences between the original and current versions are calculated; this may result in duplicate entries for some records.
- E. Each time the job is executed, only those records that have been inserted or updated since the last execution will be appended to the target table giving the desired result.

**Answer: B**

**Explanation:**

Reading table's changes, captured by CDF, using `spark.read` means that you are reading them as a static source. So, each time you run the query, all table's changes (starting from the specified `startingVersion`) will be read.

**NEW QUESTION 3**

The business intelligence team has a dashboard configured to track various summary metrics for retail stores. This includes total sales for the previous day alongside totals and averages for a variety of time periods. The fields required to populate this dashboard have the following schema:

For Demand forecasting, the Lakehouse contains a validated table of all itemized sales updated incrementally in near real-time. This table named `products_per_order`, includes the following fields:

Because reporting on long-term sales trends is less volatile, analysts using the new dashboard only require data to be refreshed once daily. Because the dashboard will be queried interactively by many users throughout a normal business day, it should return results quickly and reduce total compute associated with each materialization.

Which solution meets the expectations of the end users while controlling and limiting possible costs?

- A. Use the Delta Cache to persist the `products_per_order` table in memory to quickly refresh the dashboard with each query.
- B. Populate the dashboard by configuring a nightly batch job to save the required data to quickly update the dashboard with each query.
- C. Use Structure Streaming to configure a live dashboard against the `products_per_order` table within a Databricks notebook.
- D. Define a view against the `products_per_order` table and define the dashboard against this view.

**Answer: D**

**Explanation:**

Given the requirement for daily refresh of data and the need to ensure quick response times for interactive queries while controlling costs, a nightly batch job to pre-compute and save the required summary metrics is the most suitable approach.

? By pre-aggregating data during off-peak hours, the dashboard can serve queries quickly without requiring on-the-fly computation, which can be resource-intensive and slow, especially with many users.

? This approach also limits the cost by avoiding continuous computation throughout the day and instead leverages a batch process that efficiently computes and stores the necessary data.

? The other options (A, C, D) either do not address the cost and performance requirements effectively or are not suitable for the use case of less frequent data refresh and high interactivity.

References:

? Databricks Documentation on Batch Processing: [Databricks Batch Processing](#)

? Data Lakehouse Patterns: [Data Lakehouse Best Practices](#)

**NEW QUESTION 4**

A data engineer needs to capture pipeline settings from an existing pipeline in the workspace, and use them to create and version a JSON file to create a new pipeline. Which command should the data engineer enter in a web terminal configured with the Databricks CLI?

- A. Use the `get` command to capture the settings for the existing pipeline; remove the `pipeline_id` and rename the pipeline; use this in a `create` command
- B. Stop the existing pipeline; use the returned settings in a `reset` command

- C. Use the alone command to create a copy of an existing pipeline; use the get JSON command to get the pipeline definition; save this to git
- D. Use list pipelines to get the specs for all pipelines; get the pipeline spec from the return results parse and use this to create a pipeline

**Answer:** A

**Explanation:**

The Databricks CLI provides a way to automate interactions with Databricks services. When dealing with pipelines, you can use the databricks pipelines get --pipeline-id command to capture the settings of an existing pipeline in JSON format. This JSON can then be modified by removing the pipeline\_id to prevent conflicts and renaming the pipeline to create a new pipeline. The modified JSON file can then be used with the databricks pipelines create command to create a new pipeline with those settings. References:

? Databricks Documentation on CLI for Pipelines: Databricks CLI - Pipelines

**NEW QUESTION 5**

The data engineer team is configuring environment for development testing, and production before beginning migration on a new data pipeline. The team requires extensive testing on both the code and data resulting from code execution, and the team want to develop and test against similar production data as possible.

A junior data engineer suggests that production data can be mounted to the development testing environments, allowing pre production code to execute against production data. Because all users have

Admin privileges in the development environment, the junior data engineer has offered to configure permissions and mount this data for the team.

Which statement captures best practices for this situation?

- A. Because access to production data will always be verified using passthrough credentials it is safe to mount data to any Databricks development environment.
- B. All developer, testing and production code and data should exist in a single unified workspace; creating separate environments for testing and development further reduces risks.
- C. In environments where interactive code will be executed, production data should only be accessible with read permissions; creating isolated databases for each environment further reduces risks.
- D. Because delta Lake versions all data and supports time travel, it is not possible for user error or malicious actors to permanently delete production data, as such it is generally safe to mount production data anywhere.

**Answer:** C

**Explanation:**

The best practice in such scenarios is to ensure that production data is handled securely and with proper access controls. By granting only read access to production data in development and testing environments, it mitigates the risk of unintended data modification. Additionally, maintaining isolated databases for different environments helps to avoid accidental impacts on production data and systems. References:

? Databricks best practices for securing data:

<https://docs.databricks.com/security/index.html>

**NEW QUESTION 6**

A Databricks job has been configured with 3 tasks, each of which is a Databricks notebook. Task A does not depend on other tasks. Tasks B and C run in parallel, with each having a serial dependency on task A.

If tasks A and B complete successfully but task C fails during a scheduled run, which statement describes the resulting state?

- A. All logic expressed in the notebook associated with tasks A and B will have been successfully completed; some operations in task C may have completed successfully.
- B. All logic expressed in the notebook associated with tasks A and B will have been successfully completed; any changes made in task C will be rolled back due to task failure.
- C. All logic expressed in the notebook associated with task A will have been successfully completed; tasks B and C will not commit any changes because of stage failure.
- D. Because all tasks are managed as a dependency graph, no changes will be committed to the Lakehouse until all tasks have successfully been completed.
- E. Unless all tasks complete successfully, no changes will be committed to the Lakehouse; because task C failed, all commits will be rolled back automatically.

**Answer:** A

**Explanation:**

The query uses the CREATE TABLE USING DELTA syntax to create a Delta Lake table from an existing Parquet file stored in DBFS. The query also uses the LOCATION keyword to specify the path to the Parquet file as /mnt/finance\_eda\_bucket/tx\_sales.parquet. By using the LOCATION keyword, the query creates an external table, which is a table that is stored outside of the default warehouse directory and whose metadata is not managed by Databricks. An external table can be created from an existing directory in a cloud storage system, such as DBFS or S3, that contains data files in a supported format, such as Parquet or CSV. The resulting state after running the second command is that an external table will be created in the storage container mounted to /mnt/finance\_eda\_bucket with the new name prod.sales\_by\_store. The command will not change any data or move any files in the storage container; it will only update the table reference in the metastore and create a new Delta transaction log for the renamed table. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "ALTER TABLE RENAME TO" section; Databricks Documentation, under "Create an external table" section.

**NEW QUESTION 7**

A Delta Lake table was created with the below query:

Realizing that the original query had a typographical error, the below code was executed: ALTER TABLE prod.sales\_by\_stor RENAME TO prod.sales\_by\_store

Which result will occur after running the second command?

- A. The table reference in the metastore is updated and no data is changed.
- B. The table name change is recorded in the Delta transaction log.
- C. All related files and metadata are dropped and recreated in a single ACID transaction.
- D. The table reference in the metastore is updated and all data files are moved.
- E. A new Delta transaction log is created for the renamed table.

**Answer:** A

**Explanation:**

The query uses the CREATE TABLE USING DELTA syntax to create a Delta Lake table from an existing Parquet file stored in DBFS. The query also uses the LOCATION keyword to specify the path to the Parquet file as /mnt/finance\_eda\_bucket/tx\_sales.parquet. By using the LOCATION keyword, the query creates an external table, which is a table that is stored outside of the default warehouse directory and whose metadata is not managed by Databricks. An external table can

be created from an existing directory in a cloud storage system, such as DBFS or S3, that contains data files in a supported format, such as Parquet or CSV. The result that will occur after running the second command is that the table reference in the metastore is updated and no data is changed. The metastore is a service that stores metadata about tables, such as their schema, location, properties, and partitions. The metastore allows users to access tables using SQL commands or Spark APIs without knowing their physical location or format. When renaming an external table using the ALTER TABLE RENAME TO command, only the table reference in the metastore is updated with the new name; no data files or directories are moved or changed in the storage system. The table will still point to the same location and use the same format as before. However, if renaming a managed table, which is a table whose metadata and data are both managed by Databricks, both the table reference in the metastore and the data files in the default warehouse directory are moved and renamed accordingly. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “ALTER TABLE RENAME TO” section; Databricks Documentation, under “Metastore” section; Databricks Documentation, under “Managed and external tables” section.

#### NEW QUESTION 8

A junior data engineer is migrating a workload from a relational database system to the Databricks Lakehouse. The source system uses a star schema, leveraging foreign key constraints and multi-table inserts to validate records on write.

Which consideration will impact the decisions made by the engineer while migrating this workload?

- A. All Delta Lake transactions are ACID compliance against a single table, and Databricks does not enforce foreign key constraints.
- B. Databricks only allows foreign key constraints on hashed identifiers, which avoid collisions in highly-parallel writes.
- C. Foreign keys must reference a primary key field; multi-table inserts must leverage Delta Lake's upsert functionality.
- D. Committing to multiple tables simultaneously requires taking out multiple table locks and can lead to a state of deadlock.

**Answer:** A

#### Explanation:

In Databricks and Delta Lake, transactions are indeed ACID-compliant, but this compliance is limited to single table transactions. Delta Lake does not inherently enforce foreign key constraints, which are a staple in relational database systems for maintaining referential integrity between tables. This means that when migrating workloads from a relational database system to Databricks Lakehouse, engineers need to reconsider how to maintain data integrity and relationships that were previously enforced by foreign key constraints. Unlike traditional relational databases where foreign key constraints help in maintaining the consistency across tables, in Databricks Lakehouse, the data engineer has to manage data consistency and integrity at the application level or through careful design of ETL processes. References:

? Databricks Documentation on Delta Lake: Delta Lake Guide

? Databricks Documentation on ACID Transactions in Delta Lake: ACID Transactions in Delta Lake

#### NEW QUESTION 9

A Structured Streaming job deployed to production has been experiencing delays during peak hours of the day. At present, during normal execution, each microbatch of data is processed in less than 3 seconds. During peak hours of the day, execution time for each microbatch becomes very inconsistent, sometimes exceeding 30 seconds. The streaming write is currently configured with a trigger interval of 10 seconds.

Holding all other variables constant and assuming records need to be processed in less than 10 seconds, which adjustment will meet the requirement?

- A. Decrease the trigger interval to 5 seconds; triggering batches more frequently allows idle executors to begin processing the next batch while longer running tasks from previous batches finish.
- B. Increase the trigger interval to 30 seconds; setting the trigger interval near the maximum execution time observed for each batch is always best practice to ensure no records are dropped.
- C. The trigger interval cannot be modified without modifying the checkpoint directory; to maintain the current stream state, increase the number of shuffle partitions to maximize parallelism.
- D. Use the trigger once option and configure a Databricks job to execute the query every 10 seconds; this ensures all backlogged records are processed with each batch.
- E. Decrease the trigger interval to 5 seconds; triggering batches more frequently may prevent records from backing up and large batches from causing spill.

**Answer:** E

#### Explanation:

The adjustment that will meet the requirement of processing records in less than 10 seconds is to decrease the trigger interval to 5 seconds. This is because triggering batches more frequently may prevent records from backing up and large batches from causing spill. Spill is a phenomenon where the data in memory exceeds the available capacity and has to be written to disk, which can slow down the processing and increase the execution time<sup>1</sup>. By reducing the trigger interval, the streaming query can process smaller batches of data more quickly and avoid spill. This can also improve the latency and throughput of the streaming job<sup>2</sup>.

The other options are not correct, because:

? Option A is incorrect because triggering batches more frequently does not allow idle executors to begin processing the next batch while longer running tasks from previous batches finish. In fact, the opposite is true. Triggering batches more frequently may cause concurrent batches to compete for the same resources and cause contention and backpressure<sup>2</sup>. This can degrade the performance and stability of the streaming job.

? Option B is incorrect because increasing the trigger interval to 30 seconds is not a good practice to ensure no records are dropped. Increasing the trigger interval means that the streaming query will process larger batches of data less frequently, which can increase the risk of spill, memory pressure, and timeouts<sup>12</sup>. This can also increase the latency and reduce the throughput of the streaming job.

? Option C is incorrect because the trigger interval can be modified without modifying the checkpoint directory. The checkpoint directory stores the metadata and state of the streaming query, such as the offsets, schema, and configuration<sup>3</sup>. Changing the trigger interval does not affect the state of the streaming query, and does not require a new checkpoint directory. However, changing the number of shuffle partitions may affect the state of the streaming query, and may require a new checkpoint directory<sup>4</sup>.

? Option D is incorrect because using the trigger once option and configuring a Databricks job to execute the query every 10 seconds does not ensure that all backlogged records are processed with each batch. The trigger once option means that the streaming query will process all the available data in the source and then stop<sup>5</sup>. However, this does not guarantee that the query will finish processing within 10 seconds, especially if there are a lot of records in the source. Moreover, configuring a Databricks job to execute the query every 10 seconds may cause overlapping or missed batches, depending on the execution time of the query.

References: Memory Management Overview, Structured Streaming Performance Tuning Guide, Checkpointing, Recovery Semantics after Changes in a Streaming Query, Triggers

#### NEW QUESTION 10

A junior data engineer has manually configured a series of jobs using the Databricks Jobs UI. Upon reviewing their work, the engineer realizes that they are listed as the "Owner" for each job. They attempt to transfer "Owner" privileges to the "DevOps" group, but cannot successfully accomplish this task.

Which statement explains what is preventing this privilege transfer?



- A. Databricks jobs must have exactly one owner; "Owner" privileges cannot be assigned to a group.
- B. The creator of a Databricks job will always have "Owner" privileges; this configuration cannot be changed.
- C. Other than the default "admins" group, only individual users can be granted privileges on jobs.
- D. A user can only transfer job ownership to a group if they are also a member of that group.
- E. Only workspace administrators can grant "Owner" privileges to a group.

**Answer:** E

**Explanation:**

The reason why the junior data engineer cannot transfer "Owner" privileges to the "DevOps" group is that Databricks jobs must have exactly one owner, and the owner must be an individual user, not a group. A job cannot have more than one owner, and a job cannot have a group as an owner. The owner of a job is the user who created the job, or the user who was assigned the ownership by another user. The owner of a job has the highest level of permission on the job, and can grant or revoke permissions to other users or groups. However, the owner cannot transfer the ownership to a group, only to another user. Therefore, the junior data engineer's attempt to transfer "Owner" privileges to the "DevOps" group is not possible. References:

? Jobs access control: <https://docs.databricks.com/security/access-control/table-acls/index.html>

? Job permissions: <https://docs.databricks.com/security/access-control/table-acls/privileges.html#job-permissions>

**NEW QUESTION 10**

What statement is true regarding the retention of job run history?

- A. It is retained until you export or delete job run logs
- B. It is retained for 30 days, during which time you can deliver job run logs to DBFS or S3
- C. It is retained for 60 days, during which you can export notebook run results to HTML
- D. It is retained for 60 days, after which logs are archived
- E. It is retained for 90 days or until the run-id is re-used through custom run configuration

**Answer:** C

**NEW QUESTION 12**

An upstream system has been configured to pass the date for a given batch of data to the Databricks Jobs API as a parameter. The notebook to be scheduled will use this parameter to load data with the following code:

```
df = spark.read.format("parquet").load(f"/mnt/source/{date}")
```

Which code block should be used to create the date Python variable used in the above code block?

- A. `date = spark.conf.get("date")`
- B. `input_dict = input() date= input_dict["date"]`
- C. `import sys date = sys.argv[1]`
- D. `date = dbutils.notebooks.getParam("date")`
- E. `dbutils.widgets.text("date", "null") date = dbutils.widgets.get("date")`

**Answer:** E

**Explanation:**

The code block that should be used to create the date Python variable used in the above code block is:

```
dbutils.widgets.text("date", "null") date = dbutils.widgets.get("date")
```

This code block uses the `dbutils.widgets` API to create and get a text widget named "date" that can accept a string value as a parameter<sup>1</sup>. The default value of the widget is "null", which means that if no parameter is passed, the date variable will be "null". However, if a parameter is passed through the Databricks Jobs API, the date variable will be assigned the value of the parameter. For example, if the parameter is "2021-11-01", the date variable will be "2021-11-01". This way, the notebook can use the date variable to load data from the specified path.

The other options are not correct, because:

? Option A is incorrect because `spark.conf.get("date")` is not a valid way to get a parameter passed through the Databricks Jobs API. The `spark.conf` API is used to get or set Spark configuration properties, not notebook parameters<sup>2</sup>.

? Option B is incorrect because `input()` is not a valid way to get a parameter passed through the Databricks Jobs API. The `input()` function is used to get user input from the standard input stream, not from the API request<sup>3</sup>.

? Option C is incorrect because `sys.argv1` is not a valid way to get a parameter passed through the Databricks Jobs API. The `sys.argv` list is used to get the command-line arguments passed to a Python script, not to a notebook<sup>4</sup>.

? Option D is incorrect because `dbutils.notebooks.getParam("date")` is not a valid way to get a parameter passed through the Databricks Jobs API. The `dbutils.notebooks` API is used to get or set notebook parameters when running a notebook as a job or as a subnotebook, not when passing parameters through the API<sup>5</sup>.

References: Widgets, Spark Configuration, `input()`, `sys.argv`, Notebooks

**NEW QUESTION 13**

A CHECK constraint has been successfully added to the Delta table named `activity_details` using the following logic:

A batch job is attempting to insert new records to the table, including a record where `latitude = 45.50` and `longitude = 212.67`.

Which statement describes the outcome of this batch insert?

- A. The write will fail when the violating record is reached; any records previously processed will be recorded to the target table.
- B. The write will fail completely because of the constraint violation and no records will be inserted into the target table.
- C. The write will insert all records except those that violate the table constraints; the violating records will be recorded to a quarantine table.
- D. The write will include all records in the target table; any violations will be indicated in the boolean column named `valid_coordinates`.
- E. The write will insert all records except those that violate the table constraints; the violating records will be reported in a warning log.

**Answer:** B

**Explanation:**

The CHECK constraint is used to ensure that the data inserted into the table meets the specified conditions. In this case, the CHECK constraint is used to ensure that the latitude and longitude values are within the specified range. If the data does not meet the specified conditions, the write operation will fail completely and no records will be inserted into the target table. This is because Delta Lake supports ACID transactions, which means that either all the data is written or none of it is written. Therefore, the batch insert will fail when it encounters a record that violates the constraint, and the target table will not be updated. References:

? Constraints: <https://docs.delta.io/latest/delta-constraints.html>

? ACID Transactions: <https://docs.delta.io/latest/delta-intro.html#acid-transactions>

### NEW QUESTION 18

Which statement regarding stream-static joins and static Delta tables is correct?

- A. Each microbatch of a stream-static join will use the most recent version of the static Delta table as of each microbatch.
- B. Each microbatch of a stream-static join will use the most recent version of the static Delta table as of the job's initialization.
- C. The checkpoint directory will be used to track state information for the unique keys present in the join.
- D. Stream-static joins cannot use static Delta tables because of consistency issues.
- E. The checkpoint directory will be used to track updates to the static Delta table.

**Answer:** A

#### Explanation:

This is the correct answer because stream-static joins are supported by Structured Streaming when one of the tables is a static Delta table. A static Delta table is a Delta table that is not updated by any concurrent writes, such as appends or merges, during the execution of a streaming query. In this case, each microbatch of a stream-static join will use the most recent version of the static Delta table as of each microbatch, which means it will reflect any changes made to the static Delta table before the start of each microbatch. Verified References: [Databricks Certified Data Engineer Professional], under “Structured Streaming” section; Databricks Documentation, under “Stream and static joins” section.

### NEW QUESTION 22

The Databricks workspace administrator has configured interactive clusters for each of the data engineering groups. To control costs, clusters are set to terminate after 30 minutes of inactivity. Each user should be able to execute workloads against their assigned clusters at any time of the day. Assuming users have been added to a workspace but not granted any permissions, which of the following describes the minimal permissions a user would need to start and attach to an already configured cluster.

- A. "Can Manage" privileges on the required cluster
- B. Workspace Admin privileges, cluster creation allowe
- C. "Can Attach To" privileges on the required cluster
- D. Cluster creation allowe
- E. "Can Attach To" privileges on the required cluster
- F. "Can Restart" privileges on the required cluster
- G. Cluster creation allowe
- H. "Can Restart" privileges on the required cluster

**Answer:** D

#### Explanation:

<https://learn.microsoft.com/en-us/azure/databricks/security/auth-authz/access-control/cluster-acl>  
<https://docs.databricks.com/en/security/auth-authz/access-control/cluster-acl.html>

### NEW QUESTION 25

A data engineer, User A, has promoted a new pipeline to production by using the REST API to programmatically create several jobs. A DevOps engineer, User B, has configured an external orchestration tool to trigger job runs through the REST API. Both users authorized the REST API calls using their personal access tokens.

Which statement describes the contents of the workspace audit logs concerning these events?

- A. Because the REST API was used for job creation and triggering runs, a Service Principal will be automatically used to identity these events.
- B. Because User B last configured the jobs, their identity will be associated with both the job creation events and the job run events.
- C. Because these events are managed separately, User A will have their identity associated with the job creation events and User B will have their identity associated with the job run events.
- D. Because the REST API was used for job creation and triggering runs, user identity will not be captured in the audit logs.
- E. Because User A created the jobs, their identity will be associated with both the job creation events and the job run events.

**Answer:** C

#### Explanation:

The events are that a data engineer, User A, has promoted a new pipeline to production by using the REST API to programmatically create several jobs, and a DevOps engineer, User B, has configured an external orchestration tool to trigger job runs through the REST API. Both users authorized the REST API calls using their personal access tokens. The workspace audit logs are logs that record user activities in a Databricks workspace, such as creating, updating, or deleting objects like clusters, jobs, notebooks, or tables. The workspace audit logs also capture the identity of the user who performed each activity, as well as the time and details of the activity. Because these events are managed separately, User A will have their identity associated with the job creation events and User B will have their identity associated with the job run events in the workspace audit logs. Verified References: [Databricks Certified Data Engineer Professional], under “Databricks Workspace” section; Databricks Documentation, under “Workspace audit logs” section.

### NEW QUESTION 30

A table named user\_ltv is being used to create a view that will be used by data analysts on various teams. Users in the workspace are configured into groups, which are used for setting up data access using ACLs.

The user\_ltv table has the following schema:

email STRING, age INT, ltv INT

The following view definition is executed:

```
CREATE VIEW email_ltv AS
SELECT
CASE WHEN
    is_member('marketing') THEN email
    ELSE 'REDACTED'
END AS email,
ltv
FROM user_ltv
```

An analyst who is not a member of the marketing group executes the following query: `SELECT * FROM email_ltv`  
Which statement describes the results returned by this query?

- A. Three columns will be returned, but one column will be named "redacted" and contain only null values.
- B. Only the email and ltv columns will be returned; the email column will contain all null values.
- C. The email and ltv columns will be returned with the values in user ltv.
- D. The email, ag
- E. and ltv columns will be returned with the values in user ltv.
- F. Only the email and ltv columns will be returned; the email column will contain the string "REDACTED" in each row.

**Answer:** E

**Explanation:**

The code creates a view called `email_ltv` that selects the `email` and `ltv` columns from a table called `user_ltv`, which has the following schema: `email STRING`, `age INT`, `ltv INT`. The code also uses the `CASE WHEN` expression to replace the email values with the string "REDACTED" if the user is not a member of the marketing group. The user who executes the query is not a member of the marketing group, so they will only see the email and ltv columns, and the email column will contain the string "REDACTED" in each row. Verified References: [Databricks Certified Data Engineer Professional], under "Lakehouse" section; Databricks Documentation, under "CASE expression" section.

**NEW QUESTION 33**

An upstream system is emitting change data capture (CDC) logs that are being written to a cloud object storage directory. Each record in the log indicates the change type (insert, update, or delete) and the values for each field after the change. The source table has a primary key identified by the field `pk_id`. For auditing purposes, the data governance team wishes to maintain a full record of all values that have ever been valid in the source system. For analytical purposes, only the most recent value for each record needs to be recorded. The Databricks job to ingest these records occurs once per hour, but each individual record may have changed multiple times over the course of an hour. Which solution meets these requirements?

- A. Create a separate history table for each `pk_id` resolve the current state of the table by running a union all filtering the history tables for the most recent state.
- B. Use merge into to insert, update, or delete the most recent entry for each `pk_id` into a bronze table, then propagate all changes throughout the system.
- C. Iterate through an ordered set of changes to the table, applying each in turn; rely on Delta Lake's versioning ability to create an audit log.
- D. Use Delta Lake's change data feed to automatically process CDC data from an external system, propagating all changes to all dependent tables in the Lakehouse.
- E. Ingest all log information into a bronze table; use merge into to insert, update, or delete the most recent entry for each `pk_id` into a silver table to recreate the current table state.

**Answer:** B

**Explanation:**

This is the correct answer because it meets the requirements of maintaining a full record of all values that have ever been valid in the source system and recreating the current table state with only the most recent value for each record. The code ingests all log information into a bronze table, which preserves the raw CDC data as it is. Then, it uses merge into to perform an upsert operation on a silver table, which means it will insert new records or update or delete existing records based on the change type and the `pk_id` columns. This way, the silver table will always reflect the current state of the source table, while the bronze table will keep the history of all changes. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Upsert into a table using merge" section.

**NEW QUESTION 35**

The following code has been migrated to a Databricks notebook from a legacy workload:

```
%sh
git clone https://github.com/foo/data_loader;
python ./data_loader/run.py;
mv ./output /dbfs/mnt/new_data
```

The code executes successfully and provides the logically correct results, however, it takes over 20 minutes to extract and load around 1 GB of data. Which statement is a possible explanation for this behavior?

- A. %sh triggers a cluster restart to collect and install Gi
- B. Most of the latency is related to cluster startup time.
- C. Instead of cloning, the code should use %sh pip install so that the Python code can get executed in parallel across all nodes in a cluster.
- D. %sh does not distribute file moving operations; the final line of code should be updated to use %fs instead.
- E. Python will always execute slower than Scala on Databrick
- F. The run.py script should be refactored to Scala.
- G. %sh executes shell code on the driver nod
- H. The code does not take advantage of the worker nodes or Databricks optimized Spark.

**Answer:** E



**Explanation:**

<https://www.databricks.com/blog/2020/08/31/introducing-the-databricks-web-terminal.html>

The code is using %sh to execute shell code on the driver node. This means that the code is not taking advantage of the worker nodes or Databricks optimized Spark. This is why the code is taking longer to execute. A better approach would be to use Databricks libraries and APIs to read and write data from Git and DBFS, and to leverage the parallelism and performance of Spark. For example, you can use the Databricks Connect feature to run your Python code on a remote Databricks cluster, or you can use the Spark Git Connector to read data from Git repositories as Spark DataFrames.

**NEW QUESTION 38**

The data architect has decided that once data has been ingested from external sources into the

Databricks Lakehouse, table access controls will be leveraged to manage permissions for all production tables and views.

The following logic was executed to grant privileges for interactive queries on a production database to the core engineering group.

GRANT USAGE ON DATABASE prod TO eng; GRANT SELECT ON DATABASE prod TO eng;

Assuming these are the only privileges that have been granted to the eng group and that these users are not workspace administrators, which statement describes their privileges?

- A. Group members have full permissions on the prod database and can also assign permissions to other users or groups.
- B. Group members are able to list all tables in the prod database but are not able to see the results of any queries on those tables.
- C. Group members are able to query and modify all tables and views in the prod database, but cannot create new tables or views.
- D. Group members are able to query all tables and views in the prod database, but cannot create or edit anything in the database.
- E. Group members are able to create, query, and modify all tables and views in the prod database, but cannot define custom functions.

**Answer:** D

**Explanation:**

The GRANT USAGE ON DATABASE prod TO eng command grants the eng group the permission to use the prod database, which means they can list and access the tables and views in the database. The GRANT SELECT ON DATABASE prod TO eng command grants the eng group the permission to select data from the tables and views in the prod database, which means they can query the data using SQL or DataFrame API. However, these commands do not grant the eng group any other permissions, such as creating, modifying, or deleting tables and views, or defining custom functions. Therefore, the eng group members are able to query all tables and views in the prod database, but cannot create or edit anything in the database. References:

? Grant privileges on a database: <https://docs.databricks.com/en/security/auth-authz/table-acls/grant-privileges-database.html>

? Privileges you can grant on Hive metastore objects: <https://docs.databricks.com/en/security/auth-authz/table-acls/privileges.html>

**NEW QUESTION 39**

Which statement describes integration testing?

- A. Validates interactions between subsystems of your application
- B. Requires an automated testing framework
- C. Requires manual intervention
- D. Validates an application use case
- E. Validates behavior of individual elements of your application

**Answer:** D

**Explanation:**

This is the correct answer because it describes integration testing. Integration testing is a type of testing that validates interactions between subsystems of your application, such as modules, components, or services. Integration testing ensures that the subsystems work together as expected and produce the correct outputs or results. Integration testing can be done at different levels of granularity, such as component integration testing, system integration testing, or end-to-end testing. Integration testing can help detect errors or bugs that may not be found by unit testing, which only validates behavior of individual elements of your application. Verified References: [Databricks Certified Data Engineer Professional], under "Testing" section; Databricks Documentation, under "Integration testing" section.

**NEW QUESTION 42**

The security team is exploring whether or not the Databricks secrets module can be leveraged for connecting to an external database.

After testing the code with all Python variables being defined with strings, they upload the password to the secrets module and configure the correct permissions for the currently active user. They then modify their code to the following (leaving all other variables unchanged).

```
password = dbutils.secrets.get(scope="db_creds", key="jdbc_password")

print(password)

df = (spark
      .read
      .format("jdbc")
      .option("url", connection)
      .option("dbtable", tablename)
      .option("user", username)
      .option("password", password)
      )
```

Which statement describes what will happen when the above code is executed?

- A. The connection to the external table will fail; the string "redacted" will be printed.
- B. An interactive input box will appear in the notebook; if the right password is provided, the connection will succeed and the encoded password will be saved to DBFS.
- C. An interactive input box will appear in the notebook; if the right password is provided, the connection will succeed and the password will be printed in plain text.
- D. The connection to the external table will succeed; the string value of password will be printed in plain text.
- E. The connection to the external table will succeed; the string "redacted" will be printed.

**Answer:** E



**Explanation:**

This is the correct answer because the code is using the `dbutils.secrets.get` method to retrieve the password from the secrets module and store it in a variable. The secrets module allows users to securely store and access sensitive information such as passwords, tokens, or API keys. The connection to the external table will succeed because the password variable will contain the actual password value. However, when printing the password variable, the string “redacted” will be displayed instead of the plain text password, as a security measure to prevent exposing sensitive information in notebooks. Verified References: [Databricks Certified Data Engineer Professional], under “Security & Governance” section; Databricks Documentation, under “Secrets” section.

**NEW QUESTION 43**

A user wants to use DLT expectations to validate that a derived table report contains all records from the source, included in the table validation\_copy. The user attempts and fails to accomplish this by adding an expectation to the report table definition.

Which approach would allow using DLT expectations to validate all expected records are present in this table?

- A. Define a SQL UDF that performs a left outer join on two tables, and check if this returns null values for report key values in a DLT expectation for the report table.
- B. Define a function that performs a left outer join on validation\_copy and report and report, and check against the result in a DLT expectation for the report table
- C. Define a temporary table that perform a left outer join on validation\_copy and report, and define an expectation that no report key values are null
- D. Define a view that performs a left outer join on validation\_copy and report, and reference this view in DLT expectations for the report table

**Answer:** D

**Explanation:**

To validate that all records from the source are included in the derived table, creating a view that performs a left outer join between the validation\_copy table and the report table is effective. The view can highlight any discrepancies, such as null values in the report table's key columns, indicating missing records. This view can then be referenced in DLT (Delta Live Tables) expectations for the report table to ensure data integrity. This approach allows for a comprehensive comparison between the source and the derived table.

References:

? Databricks Documentation on Delta Live Tables and Expectations: Delta Live Tables Expectations

**NEW QUESTION 44**

Which statement describes Delta Lake optimized writes?

- A. A shuffle occurs prior to writing to try to group data together resulting in fewer files instead of each executor writing multiple files based on directory partitions.
- B. Optimized writes logical partitions instead of directory partitions partition boundaries are only represented in metadata fewer small files are written.
- C. An asynchronous job runs after the write completes to detect if files could be further compacted; yes, an OPTIMIZE job is executed toward a default of 1 GB.
- D. Before a job cluster terminates, OPTIMIZE is executed on all tables modified during the most recent job.

**Answer:** A

**Explanation:**

Delta Lake optimized writes involve a shuffle operation before writing out data to the Delta table. The shuffle operation groups data by partition keys, which can lead to a reduction in the number of output files and potentially larger files, instead of multiple smaller files. This approach can significantly reduce the total number of files in the table, improve read performance by reducing the metadata overhead, and optimize the table storage layout, especially for workloads with many small files.

References:

? Databricks documentation on Delta Lake performance tuning: <https://docs.databricks.com/delta/optimizations/auto-optimize.html>

**NEW QUESTION 48**

A small company based in the United States has recently contracted a consulting firm in India to implement several new data engineering pipelines to power artificial intelligence applications. All the company's data is stored in regional cloud storage in the United States.

The workspace administrator at the company is uncertain about where the Databricks workspace used by the contractors should be deployed.

Assuming that all data governance considerations are accounted for, which statement accurately informs this decision?

- A. Databricks runs HDFS on cloud volume storage; as such, cloud virtual machines must be deployed in the region where the data is stored.
- B. Databricks workspaces do not rely on any regional infrastructure; as such, the decision should be made based upon what is most convenient for the workspace administrator.
- C. Cross-region reads and writes can incur significant costs and latency; whenever possible, compute should be deployed in the same region the data is stored.
- D. Databricks leverages user workstations as the driver during interactive development; as such, users should always use a workspace deployed in a region they are physically near.
- E. Databricks notebooks send all executable code from the user's browser to virtual machines over the open internet; whenever possible, choosing a workspace region near the end users is the most secure.

**Answer:** C

**Explanation:**

This is the correct answer because it accurately informs this decision. The decision is about where the Databricks workspace used by the contractors should be deployed. The contractors are based in India, while all the company's data is stored in regional cloud storage in the United States. When choosing a region for deploying a Databricks workspace, one of the important factors to consider is the proximity to the data sources and sinks. Cross-region reads and writes can incur significant costs and latency due to network bandwidth and data transfer fees. Therefore, whenever possible, compute should be deployed in the same region the data is stored to optimize performance and reduce costs. Verified References: [Databricks Certified Data Engineer Professional], under “Databricks Workspace” section; Databricks Documentation, under “Choose a region” section.

**NEW QUESTION 49**

An external object storage container has been mounted to the location `/mnt/finance_eda_bucket`.

The following logic was executed to create a database for the finance team:

After the database was successfully created and permissions configured, a member of the finance team runs the following code:

If all users on the finance team are members of the finance group, which statement describes how the `tx_sales` table will be created?

- A. A logical table will persist the query plan to the Hive Metastore in the Databricks control plane.
- B. An external table will be created in the storage container mounted to `/mnt/finance_eda_bucket`.

- C. A logical table will persist the physical plan to the Hive Metastore in the Databricks control plane.
- D. An managed table will be created in the storage container mounted to /mnt/finance eda bucket.
- E. A managed table will be created in the DBFS root storage container.

**Answer:** A

**Explanation:**

<https://docs.databricks.com/en/lakehouse/data-objects.html>

**NEW QUESTION 50**

All records from an Apache Kafka producer are being ingested into a single Delta Lake table with the following schema:

key BINARY, value BINARY, topic STRING, partition LONG, offset LONG, timestamp LONG

There are 5 unique topics being ingested. Only the "registration" topic contains Personal Identifiable Information (PII). The company wishes to restrict access to PII. The company also wishes to only retain records containing PII in this table for 14 days after initial ingestion. However, for non-PII information, it would like to retain these records indefinitely.

Which of the following solutions meets the requirements?

- A. All data should be deleted biweekly; Delta Lake's time travel functionality should be leveraged to maintain a history of non-PII information.
- B. Data should be partitioned by the registration field, allowing ACLs and delete statements to be set for the PII directory.
- C. Because the value field is stored as binary data, this information is not considered PII and no special precautions should be taken.
- D. Separate object storage containers should be specified based on the partition field, allowing isolation at the storage level.
- E. Data should be partitioned by the topic field, allowing ACLs and delete statements to leverage partition boundaries.

**Answer:** B

**Explanation:**

Partitioning the data by the topic field allows the company to apply different access control policies and retention policies for different topics. For example, the company can use the Table Access Control feature to grant or revoke permissions to the registration topic based on user roles or groups. The company can also use the DELETE command to remove records from the registration topic that are older than 14 days, while keeping the records from other topics indefinitely.

Partitioning by the topic field also improves the performance of queries that filter by the topic field, as they can skip reading irrelevant partitions. References:

? Table Access Control: [https://docs.databricks.com/security/access-control/table-](https://docs.databricks.com/security/access-control/table-acls/index.html)

[acls/index.html](https://docs.databricks.com/security/access-control/table-acls/index.html)

? DELETE: <https://docs.databricks.com/delta/delta-update.html#delete-from-a-table>

**NEW QUESTION 51**

.....

## Thank You for Trying Our Product

### We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

### Databricks-Certified-Professional-Data-Engineer Practice Exam Features:

- \* Databricks-Certified-Professional-Data-Engineer Questions and Answers Updated Frequently
- \* Databricks-Certified-Professional-Data-Engineer Practice Questions Verified by Expert Senior Certified Staff
- \* Databricks-Certified-Professional-Data-Engineer Most Realistic Questions that Guarantee you a Pass on Your FirstTry
- \* Databricks-Certified-Professional-Data-Engineer Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year

**100% Actual & Verified — Instant Download, Please Click**  
**[Order The Databricks-Certified-Professional-Data-Engineer Practice Test Here](#)**