

# Databricks

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

Databricks Certified Data Engineer Professional Exam



**NEW QUESTION 1**

Review the following error traceback:

Which statement describes the error being raised?

- A. The code executed was PySpark but was executed in a Scala notebook.
- B. There is no column in the table named `heartrateheartrateheartrate`
- C. There is a type error because a column object cannot be multiplied.
- D. There is a type error because a DataFrame object cannot be multiplied.
- E. There is a syntax error because the `heartrate` column is not correctly identified as a column.

**Answer:** E

**Explanation:**

The error being raised is an `AnalysisException`, which is a type of exception that occurs when Spark SQL cannot analyze or execute a query due to some logical or semantic error<sup>1</sup>. In this case, the error message indicates that the query cannot resolve the column name `'heartrateheartrateheartrate'` given the input columns `'heartrate'` and `'age'`. This means that there is no column in the table named `'heartrateheartrateheartrate'`, and the query is invalid. A possible cause of this error is a typo or a copy-paste mistake in the query. To fix this error, the query should use a valid column name that exists in the table, such as `'heartrate'`.

References: `AnalysisException`

**NEW QUESTION 2**

A junior data engineer has been asked to develop a streaming data pipeline with a grouped aggregation using DataFrame `df`. The pipeline needs to calculate the average humidity and average temperature for each non-overlapping five-minute interval. Events are recorded once per minute per device.

Streaming DataFrame `df` has the following schema:

`"device_id INT, event_time TIMESTAMP, temp FLOAT, humidity FLOAT"` Code block:

Choose the response that correctly fills in the blank within the code block to complete this task.

- A. `to_interval("event_time", "5 minutes").alias("time")`
- B. `window("event_time", "5 minutes").alias("time")`
- C. `"event_time"`
- D. `window("event_time", "10 minutes").alias("time")`
- E. `lag("event_time", "10 minutes").alias("time")`

**Answer:** B

**Explanation:**

This is the correct answer because the `window` function is used to group streaming data by time intervals. The `window` function takes two arguments: a time column and a window duration. The window duration specifies how long each window is, and must be a multiple of 1 second. In this case, the window duration is `"5 minutes"`, which means each window will cover a non-overlapping five-minute interval. The `window` function also returns a struct column with two fields: `start` and `end`, which represent the start and end time of each window. The `alias` function is used to rename the struct column as `"time"`. Verified References:

[Databricks Certified Data Engineer Professional], under `"Structured Streaming"` section; Databricks Documentation, under `"WINDOW"` section.

<https://www.databricks.com/blog/2017/05/08/event-time-aggregation-watermarking-apache-sparks-structured-streaming.html>

**NEW QUESTION 3**

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 4**

A Delta Lake table in the Lakehouse named `customer_parsams` is used in churn prediction by the machine learning team. The table contains information about customers derived from a number of upstream sources. Currently, the data engineering team populates this table nightly by overwriting the table with the current valid values derived from upstream data sources.

Immediately after each update succeeds, the data engineer team would like to determine the difference between the new version and the previous of the table.

Given the current implementation, which method can be used?

- A. Parse the Delta Lake transaction log to identify all newly written data files.
- B. Execute `DESCRIBE HISTORY customer_churn_params` to obtain the full operation metrics for the update, including a log of all records that have been added or modified.
- C. Execute a query to calculate the difference between the new version and the previous version using Delta Lake's built-in versioning and time travel functionality.
- D. Parse the Spark event logs to identify those rows that were updated, inserted, or deleted.

**Answer:** C

**Explanation:**

Delta Lake provides built-in versioning and time travel capabilities, allowing users to query previous snapshots of a table. This feature is particularly useful for

understanding changes between different versions of the table. In this scenario, where the table is overwritten nightly, you can use Delta Lake's time travel feature to execute a query comparing the latest version of the table (the current state) with its previous version. This approach effectively identifies the differences (such as new, updated, or deleted records) between the two versions. The other options do not provide a straightforward or efficient way to directly compare different versions of a Delta Lake table.

References:

? Delta Lake Documentation on Time Travel: Delta Time Travel

? Delta Lake Versioning: Delta Lake Versioning Guide

#### 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

Which of the following technologies can be used to identify key areas of text when parsing Spark Driver log4j output?

- A. Regex
- B. Julia
- C. pyspark.ml.feature
- D. Scala Datasets
- E. C++

**Answer:** A

#### Explanation:

Regex, or regular expressions, are a powerful way of matching patterns in text. They can be used to identify key areas of text when parsing Spark Driver log4j output, such as the log level, the timestamp, the thread name, the class name, the method name, and the message. Regex can be applied in various languages and frameworks, such as Scala, Python, Java, Spark SQL, and Databricks notebooks. References:

? <https://docs.databricks.com/notebooks/notebooks-use.html#use-regular-expressions>

? <https://docs.databricks.com/spark/latest/spark-sql/udf-scala.html#using-regular-expressions-in-udfs>

? [https://docs.databricks.com/spark/latest/sparkr/functions/regexp\\_extract.html](https://docs.databricks.com/spark/latest/sparkr/functions/regexp_extract.html)

? [https://docs.databricks.com/spark/latest/sparkr/functions/regexp\\_replace.html](https://docs.databricks.com/spark/latest/sparkr/functions/regexp_replace.html)

#### NEW QUESTION 7

A data engineer is configuring a pipeline that will potentially see late-arriving, duplicate records.

In addition to de-duplicating records within the batch, which of the following approaches allows the data engineer to deduplicate data against previously processed records as it is inserted into a Delta table?

- A. Set the configuration `delta.deduplicate = true`.
- B. VACUUM the Delta table after each batch completes.
- C. Perform an insert-only merge with a matching condition on a unique key.
- D. Perform a full outer join on a unique key and overwrite existing data.
- E. Rely on Delta Lake schema enforcement to prevent duplicate records.

**Answer:** C

#### Explanation:

To deduplicate data against previously processed records as it is inserted into a Delta table, you can use the merge operation with an insert-only clause. This allows you to insert new records that do not match any existing records based on a unique key, while ignoring duplicate records that match existing records. For example, you can use the following syntax:

```
MERGE INTO target_table USING source_table ON target_table.unique_key = source_table.unique_key WHEN NOT MATCHED THEN INSERT *
```

This will insert only the records from the source table that have a unique key that is not present in the target table, and skip the records that have a matching key.

This way, you can avoid inserting duplicate records into the Delta table.

References:

? <https://docs.databricks.com/delta/delta-update.html#upsert-into-a-table-using-merge>

? <https://docs.databricks.com/delta/delta-update.html#insert-only-merge>

#### NEW QUESTION 8

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 9**

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 10**

Which statement describes the correct use of pyspark.sql.functions.broadcast?

- A. It marks a column as having low enough cardinality to properly map distinct values to available partitions, allowing a broadcast join.
- B. It marks a column as small enough to store in memory on all executors, allowing a broadcast join.
- C. It caches a copy of the indicated table on attached storage volumes for all active clusters within a Databricks workspace.
- D. It marks a DataFrame as small enough to store in memory on all executors, allowing a broadcast join.
- E. It caches a copy of the indicated table on all nodes in the cluster for use in all future queries during the cluster lifetime.

**Answer:** D

**Explanation:**

<https://spark.apache.org/docs/3.1.3/api/python/reference/api/pyspark.sql.functions.broadcast.html>

The broadcast function in PySpark is used in the context of joins. When you mark a DataFrame with broadcast, Spark tries to send this DataFrame to all worker nodes so that it can be joined with another DataFrame without shuffling the larger DataFrame across the nodes. This is particularly beneficial when the DataFrame is small enough to fit into the memory of each node. It helps to optimize the join process by reducing the amount of data that needs to be shuffled across the cluster, which can be a very expensive operation in terms of computation and time.

The pyspark.sql.functions.broadcast function in PySpark is used to hint to Spark that a DataFrame is small enough to be broadcast to all worker nodes in the cluster. When this hint is applied, Spark can perform a broadcast join, where the smaller DataFrame is sent to each executor only once and joined with the larger DataFrame on each executor. This can significantly reduce the amount of data shuffled across the network and can improve the performance of the join operation. In a broadcast join, the entire smaller DataFrame is sent to each executor, not just a specific column or a cached version on attached storage. This function is particularly useful when one of the DataFrames in a join operation is much smaller than the other, and can fit comfortably in the memory of each executor node. References:

? Databricks Documentation on Broadcast Joins: Databricks Broadcast Join Guide

? PySpark API Reference: pyspark.sql.functions.broadcast

**NEW QUESTION 10**

Incorporating unit tests into a PySpark application requires upfront attention to the design of your jobs, or a potentially significant refactoring of existing code. Which statement describes a main benefit that offset this additional effort?



- A. Improves the quality of your data
- B. Validates a complete use case of your application
- C. Troubleshooting is easier since all steps are isolated and tested individually
- D. Yields faster deployment and execution times
- E. Ensures that all steps interact correctly to achieve the desired end result

**Answer:** A

#### NEW QUESTION 11

A data engineer is testing a collection of mathematical functions, one of which calculates the area under a curve as described by another function. Which kind of the test does the above line exemplify?

- A. Integration
- B. Unit
- C. Manual
- D. functional

**Answer:** B

#### Explanation:

A unit test is designed to verify the correctness of a small, isolated piece of code, typically a single function. Testing a mathematical function that calculates the area under a curve is an example of a unit test because it is testing a specific, individual function to ensure it operates as expected.

References:

? Software Testing Fundamentals: Unit Testing

#### NEW QUESTION 13

A junior data engineer has been asked to develop a streaming data pipeline with a grouped aggregation using DataFrame df. The pipeline needs to calculate the average humidity and average temperature for each non-overlapping five-minute interval. Incremental state information should be maintained for 10 minutes for late-arriving data.

Streaming DataFrame df has the following schema:

"device\_id INT, event\_time TIMESTAMP, temp FLOAT, humidity FLOAT" Code block:

Choose the response that correctly fills in the blank within the code block to complete this task.

- A. withWatermark("event\_time", "10 minutes")
- B. awaitArrival("event\_time", "10 minutes")
- C. await("event\_time + '10 minutes'")
- D. slidingWindow("event\_time", "10 minutes")
- E. delayWrite("event\_time", "10 minutes")

**Answer:** A

#### Explanation:

The correct answer is A. withWatermark("event\_time", "10 minutes"). This is because the question asks for incremental state information to be maintained for 10 minutes for late-arriving data. The withWatermark method is used to define the watermark for late data. The watermark is a timestamp column and a threshold that tells the system

how long to wait for late data. In this case, the watermark is set to 10 minutes. The other options are incorrect because they are not valid methods or syntax for watermarking in Structured Streaming. References:

? Watermarking: <https://docs.databricks.com/spark/latest/structured-streaming/watermarks.html>

? Windowed aggregations: <https://docs.databricks.com/spark/latest/structured-streaming/window-operations.html>

#### NEW QUESTION 18

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 23

The data engineering team is migrating an enterprise system with thousands of tables and views into the Lakehouse. They plan to implement the target architecture using a series of bronze, silver, and gold tables. Bronze tables will almost exclusively be used by production data engineering workloads, while silver tables will be used to support both data engineering and machine learning workloads. Gold tables will largely serve business intelligence and reporting purposes. While personal identifying information (PII) exists in all tiers of data, pseudonymization and anonymization rules are in place for all data at the silver and gold levels.

The organization is interested in reducing security concerns while maximizing the ability to collaborate across diverse teams. Which statement exemplifies best practices for implementing this system?

- A. Isolating tables in separate databases based on data quality tiers allows for easy permissions management through database ACLs and allows physical separation of default storage locations for managed tables.
- B. Because databases on Databricks are merely a logical construct, choices around database organization do not impact security or discoverability in the Lakehouse.
- C. Storing all production tables in a single database provides a unified view of all data assets available throughout the Lakehouse, simplifying discoverability by granting all users view privileges on this database.
- D. Working in the default Databricks database provides the greatest security when working with managed tables, as these will be created in the DBFS root.
- E. Because all tables must live in the same storage containers used for the database they're created in, organizations should be prepared to create between dozens and thousands of databases depending on their data isolation requirements.

**Answer:** A

**Explanation:**

This is the correct answer because it exemplifies best practices for implementing this system. By isolating tables in separate databases based on data quality tiers, such as bronze, silver, and gold, the data engineering team can achieve several benefits. First, they can easily manage permissions for different users and groups through database ACLs, which allow granting or revoking access to databases, tables, or views. Second, they can physically separate the default storage locations for managed tables in each database, which can improve performance and reduce costs. Third, they can provide a clear and consistent naming convention for the tables in each database, which can improve discoverability and usability. Verified References: [Databricks Certified Data Engineer Professional], under "Lakehouse" section; Databricks Documentation, under "Database object privileges" section.

**NEW QUESTION 26**

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 28**

The business reporting team requires that data for their dashboards be updated every hour. The total processing time for the pipeline that extracts transforms and load the data for their pipeline runs in 10 minutes.

Assuming normal operating conditions, which configuration will meet their service-level agreement requirements with the lowest cost?

- A. Schedule a job to execute the pipeline once an hour on a dedicated interactive cluster.
- B. Schedule a Structured Streaming job with a trigger interval of 60 minutes.
- C. Schedule a job to execute the pipeline once an hour on a new job cluster.
- D. Configure a job that executes every time new data lands in a given directory.

**Answer:** C

**Explanation:**

Scheduling a job to execute the data processing pipeline once an hour on a new job cluster is the most cost-effective solution given the scenario. Job clusters are ephemeral in nature; they are spun up just before the job execution and terminated upon completion, which means you only incur costs for the time the cluster is active. Since the total processing time is only 10 minutes, a new job cluster created for each hourly execution minimizes the running time and thus the cost, while also fulfilling the requirement for hourly data updates for the business reporting team's dashboards.

References:

? Databricks documentation on jobs and job clusters: <https://docs.databricks.com/jobs.html>

**NEW QUESTION 29**

To reduce storage and compute costs, the data engineering team has been tasked with curating a series of aggregate tables leveraged by business intelligence dashboards, customer-facing applications, production machine learning models, and ad hoc analytical queries.

The data engineering team has been made aware of new requirements from a customer-facing application, which is the only downstream workload they manage entirely. As a result, an aggregate table used by numerous teams across the organization will need to have a number of fields renamed, and additional fields will also be added.

Which of the solutions addresses the situation while minimally interrupting other teams in the organization without increasing the number of tables that need to be managed?

- A. Send all users notice that the schema for the table will be changing; include in the communication the logic necessary to revert the new table schema to match historic queries.
- B. Configure a new table with all the requisite fields and new names and use this as the source for the customer-facing application; create a view that maintains the original data schema and table name by aliasing select fields from the new table.
- C. Create a new table with the required schema and new fields and use Delta Lake's deep clone functionality to sync up changes committed to one table to the corresponding table.
- D. Replace the current table definition with a logical view defined with the query logic currently writing the aggregate table; create a new table to power the customer-facing application.
- E. Add a table comment warning all users that the table schema and field names will be changing on a given date; overwrite the table in place to the specifications of the customer-facing application.

**Answer:** B

**Explanation:**

This is the correct answer because it addresses the situation while minimally interrupting other teams in the organization without increasing the number of tables that need to be managed. The situation is that an aggregate table used by numerous teams across the organization will need to have a number of fields renamed, and additional fields will also be added, due to new requirements from a customer-facing application. By configuring a new table with all the requisite fields and

new names and using this as the source for the customer-facing application, the data engineering team can meet the new requirements without affecting other teams that rely on the existing table schema and name. By creating a view that maintains the original data schema and table name by aliasing select fields from the new table, the data engineering team can also avoid duplicating data or creating additional tables that need to be managed. Verified References: [Databricks Certified Data Engineer Professional], under “Lakehouse” section; Databricks Documentation, under “CREATE VIEW” section.

### NEW QUESTION 33

When evaluating the Ganglia Metrics for a given cluster with 3 executor nodes, which indicator would signal proper utilization of the VM's resources?

- A. The five Minute Load Average remains consistent/flat
- B. Bytes Received never exceeds 80 million bytes per second
- C. Network I/O never spikes
- D. Total Disk Space remains constant
- E. CPU Utilization is around 75%

**Answer:** E

#### Explanation:

In the context of cluster performance and resource utilization, a CPU utilization rate of around 75% is generally considered a good indicator of efficient resource usage. This level of CPU utilization suggests that the cluster is being effectively used without being overburdened or underutilized.

? A consistent 75% CPU utilization indicates that the cluster's processing power is being effectively employed while leaving some headroom to handle spikes in workload or additional tasks without maxing out the CPU, which could lead to performance degradation.

? A five Minute Load Average that remains consistent/flat (Option A) might indicate underutilization or a bottleneck elsewhere.

? Monitoring network I/O (Options B and C) is important, but these metrics alone don't provide a complete picture of resource utilization efficiency.

? Total Disk Space (Option D) remaining constant is not necessarily an indicator of proper resource utilization, as it's more related to storage rather than computational efficiency.

References:

? Ganglia Monitoring System: Ganglia Documentation

? Databricks Documentation on Monitoring: Databricks Cluster Monitoring

### NEW QUESTION 37

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 38

A data architect has designed a system in which two Structured Streaming jobs will concurrently write to a single bronze Delta table. Each job is subscribing to a different topic from an Apache Kafka source, but they will write data with the same schema. To keep the directory structure simple, a data engineer has decided to nest a checkpoint directory to be shared by both streams.

The proposed directory structure is displayed below:

Which statement describes whether this checkpoint directory structure is valid for the given scenario and why?

- A. No; Delta Lake manages streaming checkpoints in the transaction log.
- B. Yes; both of the streams can share a single checkpoint directory.
- C. No; only one stream can write to a Delta Lake table.
- D. Yes; Delta Lake supports infinite concurrent writers.
- E. No; each of the streams needs to have its own checkpoint directory.

**Answer:** E

#### Explanation:

This is the correct answer because checkpointing is a critical feature of Structured Streaming that provides fault tolerance and recovery in case of failures.

Checkpointing stores the current state and progress of a streaming query in a reliable storage system, such as DBFS or S3. Each streaming query must have its



own checkpoint directory that is unique and exclusive to that query. If two streaming queries share the same checkpoint directory, they will interfere with each other and cause unexpected errors or data loss. Verified References: [Databricks Certified Data Engineer Professional], under “Structured Streaming” section; Databricks Documentation, under “Checkpointing” section.

**NEW QUESTION 40**

When scheduling Structured Streaming jobs for production, which configuration automatically recovers from query failures and keeps costs low?

- A. Cluster: New Job Cluster; Retries: Unlimited;Maximum Concurrent Runs: Unlimited
- B. Cluster: New Job Cluster; Retries: None;Maximum Concurrent Runs: 1
- C. Cluster: Existing All-Purpose Cluster; Retries: Unlimited;Maximum Concurrent Runs: 1
- D. Cluster: Existing All-Purpose Cluster; Retries: Unlimited;Maximum Concurrent Runs: 1
- E. Cluster: Existing All-Purpose Cluster; Retries: None;Maximum Concurrent Runs: 1

**Answer:** D

**Explanation:**

The configuration that automatically recovers from query failures and keeps costs low is to use a new job cluster, set retries to unlimited, and set maximum concurrent runs to 1. This configuration has the following advantages:

? A new job cluster is a cluster that is created and terminated for each job run. This means that the cluster resources are only used when the job is running, and no idle costs are incurred. This also ensures that the cluster is always in a clean state and has the latest configuration and libraries for the job<sup>1</sup>.

? Setting retries to unlimited means that the job will automatically restart the query in case of any failure, such as network issues, node failures, or transient errors. This improves the reliability and availability of the streaming job, and avoids data loss or inconsistency<sup>2</sup>.

? Setting maximum concurrent runs to 1 means that only one instance of the job can run at a time. This prevents multiple queries from competing for the same resources or writing to the same output location, which can cause performance degradation or data corruption<sup>3</sup>.

Therefore, this configuration is the best practice for scheduling Structured Streaming jobs for production, as it ensures that the job is resilient, efficient, and consistent.

References: Job clusters, Job retries, Maximum concurrent runs

**NEW QUESTION 45**

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 49**

A production workload incrementally applies updates from an external Change Data Capture feed to a Delta Lake table as an always-on Structured Stream job. When data was initially migrated for this table, OPTIMIZE was executed and most data files were resized to 1 GB. Auto Optimize and Auto Compaction were both turned on for the streaming production job. Recent review of data files shows that most data files are under 64 MB, although each partition in the table contains at least 1 GB of data and the total table size is over 10 TB.

Which of the following likely explains these smaller file sizes?

- A. Databricks has autotuned to a smaller target file size to reduce duration of MERGE operations
- B. Z-order indices calculated on the table are preventing file compaction
- C. Bloom filter indices calculated on the table are preventing file compaction
- D. Databricks has autotuned to a smaller target file size based on the overall size of data in the table
- E. Databricks has autotuned to a smaller target file size based on the amount of data in each partition

**Answer:** A

**Explanation:**

This is the correct answer because Databricks has a feature called Auto Optimize, which automatically optimizes the layout of Delta Lake tables by coalescing small files into larger ones and sorting data within each file by a specified column. However, Auto Optimize also considers the trade-off between file size and merge performance, and may choose a smaller target file size to reduce the duration of merge operations, especially for streaming workloads that frequently update existing records. Therefore, it is possible that Auto Optimize has autotuned to a smaller target file size based on the characteristics of the streaming production job. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Auto Optimize” section. <https://docs.databricks.com/en/delta/tune-file-size.html#autotune-table> 'Autotune file size based on workload'

**NEW QUESTION 51**

The data engineering team maintains a table of aggregate statistics through batch nightly updates. This includes total sales for the previous day alongside totals and averages for a variety of time periods including the 7 previous days, year-to-date, and quarter-to-date. This table is named store\_sales\_summary and the schema is as follows:

The table daily\_store\_sales contains all the information needed to update store\_sales\_summary. The schema for this table is: store\_id INT, sales\_date DATE, total\_sales FLOAT If daily\_store\_sales is implemented as a Type 1 table and the total\_sales column might be adjusted after manual data auditing, which approach is the safest to generate accurate reports in the store\_sales\_summary table?

- A. Implement the appropriate aggregate logic as a batch read against the daily\_store\_sales table and overwrite the store\_sales\_summary table with each Update.
- B. Implement the appropriate aggregate logic as a batch read against the daily\_store\_sales table and append new rows nightly to the store\_sales\_summary table.
- C. Implement the appropriate aggregate logic as a batch read against the daily\_store\_sales table and use upsert logic to update results in the



store\_sales\_summary table.

D. Implement the appropriate aggregate logic as a Structured Streaming read against the daily\_store\_sales table and use upsert logic to update results in the store\_sales\_summary table.

E. Use Structured Streaming to subscribe to the change data feed for daily\_store\_sales and apply changes to the aggregates in the store\_sales\_summary table with each update.

**Answer:** E

**Explanation:**

The daily\_store\_sales table contains all the information needed to update store\_sales\_summary. The schema of the table is:

store\_id INT, sales\_date DATE, total\_sales FLOAT

The daily\_store\_sales table is implemented as a Type 1 table, which means that old values are overwritten by new values and no history is maintained. The total\_sales column might be adjusted after manual data auditing, which means that the data in the table may change over time.

The safest approach to generate accurate reports in the store\_sales\_summary table is to use Structured Streaming to subscribe to the change data feed for daily\_store\_sales and apply changes to the aggregates in the store\_sales\_summary table with each update. Structured Streaming is a scalable and fault-tolerant stream processing engine built on Spark SQL. Structured Streaming allows processing data streams as if they were tables or DataFrames, using familiar operations such as select, filter, groupBy, or join. Structured Streaming also supports output modes that specify how to write the results of a streaming query to a sink, such as append, update, or complete. Structured Streaming can handle both streaming and batch data sources in a unified manner.

The change data feed is a feature of Delta Lake that provides structured streaming sources that can subscribe to changes made to a Delta Lake table. The change data feed captures both data changes and schema changes as ordered events that can be processed by downstream applications or services. The change data feed can be configured with different options, such as starting from a specific version or timestamp, filtering by operation type or partition values, or excluding no-op changes.

By using Structured Streaming to subscribe to the change data feed for daily\_store\_sales, one can capture and process any changes made to the total\_sales column due to manual data auditing. By applying these changes to the aggregates in the store\_sales\_summary table with each update, one can ensure that the reports are always consistent and accurate with the latest data. Verified References: [Databricks Certified Data Engineer Professional], under “Spark Core” section; Databricks Documentation, under “Structured Streaming” section; Databricks Documentation, under “Delta Change Data Feed” section.

**NEW QUESTION 52**

A Data engineer wants to run unit's tests using common Python testing frameworks on python functions defined across several Databricks notebooks currently used in production.

How can the data engineer run unit tests against function that work with data in production?

- A. Run unit tests against non-production data that closely mirrors production
- B. Define and unit test functions using Files in Repos
- C. Define units test and functions within the same notebook
- D. Define and import unit test functions from a separate Databricks notebook

**Answer:** A

**Explanation:**

The best practice for running unit tests on functions that interact with data is to use a dataset that closely mirrors the production data. This approach allows data engineers to validate the logic of their functions without the risk of affecting the actual production data. It's important to have a representative sample of production data to catch edge cases and ensure the functions will work correctly when used in a production environment.

References:

? Databricks Documentation on Testing: Testing and Validation of Data and Notebooks

**NEW QUESTION 54**

A Spark job is taking longer than expected. Using the Spark UI, a data engineer notes that the Min, Median, and Max Durations for tasks in a particular stage show the minimum and median time to complete a task as roughly the same, but the max duration for a task to be roughly 100 times as long as the minimum.

Which situation is causing increased duration of the overall job?

- A. Task queueing resulting from improper thread pool assignment.
- B. Spill resulting from attached volume storage being too small.
- C. Network latency due to some cluster nodes being in different regions from the source data
- D. Skew caused by more data being assigned to a subset of spark-partitions.
- E. Credential validation errors while pulling data from an external system.

**Answer:** D

**Explanation:**

This is the correct answer because skew is a common situation that causes increased duration of the overall job. Skew occurs when some partitions have more data than others, resulting in uneven distribution of work among tasks and executors. Skew can be caused by various factors, such as skewed data distribution, improper partitioning strategy, or join operations with skewed keys. Skew can lead to performance issues such as long-running tasks, wasted resources, or even task failures due to memory or disk spills. Verified References: [Databricks Certified Data Engineer Professional], under “Performance Tuning” section; Databricks Documentation, under “Skew” section.

**NEW QUESTION 55**

The data engineer team has been tasked with configured connections to an external database that does not have a supported native connector with Databricks. The external database already has data security configured by group membership. These groups map directly to user group already created in Databricks that represent various teams within the company.

A new login credential has been created for each group in the external database. The Databricks Utilities Secrets module will be used to make these credentials available to Databricks users.

Assuming that all the credentials are configured correctly on the external database and group membership is properly configured on Databricks, which statement describes how teams can be granted the minimum necessary access to using these credentials?

- A. “Read” permissions should be set on a secret key mapped to those credentials that will be used by a given team.
- B. No additional configuration is necessary as long as all users are configured as administrators in the workspace where secrets have been added.
- C. “Read” permissions should be set on a secret scope containing only those credentials that will be used by a given team.
- D. “Manage” permission should be set on a secret scope containing only those credentials that will be used by a given team.

**Answer:** C

**Explanation:**

In Databricks, using the Secrets module allows for secure management of sensitive information such as database credentials. Granting 'Read' permissions on a secret key that maps to database credentials for a specific team ensures that only members of that team can access these credentials. This approach aligns with the principle of least privilege, granting users the minimum level of access required to perform their jobs, thus enhancing security.

References:

? Databricks Documentation on Secret Management: Secrets

**NEW QUESTION 59**

A DLT pipeline includes the following streaming tables:

Raw\_lot ingest raw device measurement data from a heart rate tracking device. Bgm\_stats incrementally computes user statistics based on BPM measurements from raw\_lot.

How can the data engineer configure this pipeline to be able to retain manually deleted or updated records in the raw\_lot table while recomputing the downstream table when a pipeline update is run?

- A. Set the skipChangeCommits flag to true on bpm\_stats
- B. Set the SkipChangeCommits flag to true raw\_lot
- C. Set the pipelines, reset, allowed property to false on bpm\_stats
- D. Set the pipelines, reset, allowed property to false on raw\_lot

**Answer:** D

**Explanation:**

In Databricks Lakehouse, to retain manually deleted or updated records in the raw\_lot table while recomputing downstream tables when a pipeline update is run, the property pipelines.reset.allowed should be set to false. This property prevents the system from resetting the state of the table, which includes the removal of the history of changes, during a pipeline update. By keeping this property as false, any changes to the raw\_lot table, including manual deletes or updates, are retained, and recomputation of downstream tables, such as bpm\_stats, can occur with the full history of data changes intact. References:

? Databricks documentation on DLT pipelines: <https://docs.databricks.com/data-engineering/delta-live-tables/delta-live-tables-overview.html>

**NEW QUESTION 60**

The marketing team is looking to share data in an aggregate table with the sales organization, but the field names used by the teams do not match, and a number of marketing specific fields have not been approved for the sales org.

Which of the following solutions addresses the situation while emphasizing simplicity?

- A. Create a view on the marketing table selecting only these fields approved for the sales team alias the names of any fields that should be standardized to the sales naming conventions.
- B. Use a CTAS statement to create a derivative table from the marketing table configure a production job to propagate changes.
- C. Add a parallel table write to the current production pipeline, updating a new sales table that varies as required from marketing table.
- D. Create a new table with the required schema and use Delta Lake's DEEP CLONE functionality to sync up changes committed to one table to the corresponding table.

**Answer:** A

**Explanation:**

Creating a view is a straightforward solution that can address the need for field name standardization and selective field sharing between departments. A view allows for presenting a transformed version of the underlying data without duplicating it. In this scenario, the view would only include the approved fields for the sales team and rename any fields as per their naming conventions.

References:

? Databricks documentation on using SQL views in Delta Lake: <https://docs.databricks.com/delta/quick-start.html#sql-views>

**NEW QUESTION 63**

Which statement regarding spark configuration on the Databricks platform is true?

- A. Spark configuration properties set for an interactive cluster with the Clusters UI will impact all notebooks attached to that cluster.
- B. When the same spark configuration property is set for an interactive to the same interactive cluster.
- C. Spark configuration set within a notebook will affect all SparkSession attached to the same interactive cluster
- D. The Databricks REST API can be used to modify the Spark configuration properties for an interactive cluster without interrupting jobs.

**Answer:** A

**Explanation:**

When Spark configuration properties are set for an interactive cluster using the Clusters UI in Databricks, those configurations are applied at the cluster level. This means that all notebooks attached to that cluster will inherit and be affected by these configurations. This approach ensures consistency across all executions within that cluster, as the Spark configuration properties dictate aspects such as memory allocation, number of executors, and other vital execution parameters. This centralized configuration management helps maintain standardized execution environments across different notebooks, aiding in debugging and performance optimization.

References:

? Databricks documentation on configuring clusters: <https://docs.databricks.com/clusters/configure.html>

**NEW QUESTION 65**

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 66**

Which Python variable contains a list of directories to be searched when trying to locate required modules?

- A. importlib.resource path
- B. ,sys.path
- C. os-path
- D. pypi.path
- E. pylib.source

**Answer: B**

**NEW QUESTION 69**

The data governance team is reviewing code used for deleting records for compliance with GDPR. They note the following logic is used to delete records from the Delta Lake table named users.

```
DELETE FROM users
WHERE user_id IN
    (SELECT user_id FROM delete_requests)
```

Assuming that user\_id is a unique identifying key and that delete\_requests contains all users that have requested deletion, which statement describes whether successfully executing the above logic guarantees that the records to be deleted are no longer accessible and why?

- A. Yes; Delta Lake ACID guarantees provide assurance that the delete command succeeded fully and permanently purged these records.
- B. No; the Delta cache may return records from previous versions of the table until the cluster is restarted.
- C. Yes; the Delta cache immediately updates to reflect the latest data files recorded to disk.
- D. No; the Delta Lake delete command only provides ACID guarantees when combined with the merge into command.
- E. No; files containing deleted records may still be accessible with time travel until a vacuum command is used to remove invalidated data files.

**Answer: E**

**Explanation:**

The code uses the DELETE FROM command to delete records from the users table that match a condition based on a join with another table called delete\_requests, which contains all users that have requested deletion. The DELETE FROM command deletes records from a Delta Lake table by creating a new version of the table that does not contain the deleted records. However, this does not guarantee that the records to be deleted are no longer accessible, because Delta Lake supports time travel, which allows querying previous versions of the table using a timestamp or version number. Therefore, files containing deleted records may still be accessible with time travel until a vacuum command is used to remove invalidated data files from physical storage. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Delete from a table" section; Databricks Documentation, under "Remove files no longer referenced by a Delta table" section.

**NEW QUESTION 70**

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 71**

The data engineering team has configured a job to process customer requests to be forgotten (have their data deleted). All user data that needs to be deleted is stored in Delta Lake tables using default table settings.

The team has decided to process all deletions from the previous week as a batch job at 1am each Sunday. The total duration of this job is less than one hour.

Every Monday at 3am, a batch job executes a series of VACUUM commands on all Delta Lake tables throughout the organization.

The compliance officer has recently learned about Delta Lake's time travel functionality. They are concerned that this might allow continued access to deleted data.

Assuming all delete logic is correctly implemented, which statement correctly addresses this concern?

- A. Because the vacuum command permanently deletes all files containing deleted records, deleted records may be accessible with time travel for around 24 hours.
- B. Because the default data retention threshold is 24 hours, data files containing deleted records will be retained until the vacuum job is run the following day.
- C. Because Delta Lake time travel provides full access to the entire history of a table, deleted records can always be recreated by users with full admin privileges.
- D. Because Delta Lake's delete statements have ACID guarantees, deleted records will be permanently purged from all storage systems as soon as a delete job completes.
- E. Because the default data retention threshold is 7 days, data files containing deleted records will be retained until the vacuum job is run 8 days later.

**Answer:** E

**Explanation:**

<https://learn.microsoft.com/en-us/azure/databricks/delta/vacuum>

**NEW QUESTION 76**

Two of the most common data locations on Databricks are the DBFS root storage and external object storage mounted with `dbutils.fs.mount()`.

Which of the following statements is correct?

- A. DBFS is a file system protocol that allows users to interact with files stored in object storage using syntax and guarantees similar to Unix file systems.
- B. By default, both the DBFS root and mounted data sources are only accessible to workspace administrators.
- C. The DBFS root is the most secure location to store data, because mounted storage volumes must have full public read and write permissions.
- D. Neither the DBFS root nor mounted storage can be accessed when using `%sh` in a Databricks notebook.
- E. The DBFS root stores files in ephemeral block volumes attached to the driver, while mounted directories will always persist saved data to external storage between sessions.

**Answer:** A

**Explanation:**

DBFS is a file system protocol that allows users to interact with files stored in object storage using syntax and guarantees similar to Unix file systems<sup>1</sup>. DBFS is not a physical file system, but a layer over the object storage that provides a unified view of data across different data sources<sup>1</sup>. By default, the DBFS root is accessible to all users in the workspace, and the access to mounted data sources depends on the permissions of the storage account or container<sup>2</sup>. Mounted storage volumes do not need to have full public read and write permissions, but they do require a valid connection string or access key to be provided when mounting<sup>3</sup>. Both the DBFS root and mounted storage can be accessed when using `%sh` in a Databricks notebook, as long as the cluster has FUSE enabled<sup>4</sup>. The DBFS root does not store files in ephemeral block volumes attached to the driver, but in the object storage associated with the workspace<sup>1</sup>. Mounted directories will persist saved data to external storage between sessions, unless they are unmounted or deleted<sup>3</sup>. References: DBFS, Work with files on Azure Databricks, Mounting cloud object storage on Azure Databricks, Access DBFS with FUSE

**NEW QUESTION 77**

Which is a key benefit of an end-to-end test?

- A. It closely simulates real world usage of your application.
- B. It pinpoint errors in the building blocks of your application.
- C. It provides testing coverage for all code paths and branches.
- D. It makes it easier to automate your test suite

**Answer:** A

**Explanation:**

End-to-end testing is a methodology used to test whether the flow of an application, from start to finish, behaves as expected. The key benefit of an end-to-end test is that it closely simulates real-world, user behavior, ensuring that the system as a whole operates correctly.

References:

? Software Testing: End-to-End Testing

**NEW QUESTION 79**

The data governance team is reviewing user for deleting records for compliance with GDPR. The following logic has been implemented to propagate deleted requests from the `user_lookup` table to the `user_aggregate` table.



```
(spark.read
  .format("delta")
  .option("readChangeData", True)
  .option("startingTimestamp", '2021-08-22 00:00:00')
  .option("endingTimestamp", '2021-08-29 00:00:00')
  .table("user_lookup")
  .createOrReplaceTempView("changes"))

spark.sql("""
  DELETE FROM user_aggregates
  WHERE user_id IN (
    SELECT user_id
    FROM changes
    WHERE _change_type='delete'
  )
""")
```

Assuming that user\_id is a unique identifying key and that all users have requested deletion have been removed from the user\_lookup table, which statement describes whether successfully executing the above logic guarantees that the records to be deleted from the user\_aggregates table are no longer accessible and why?

- A. No: files containing deleted records may still be accessible with time travel until a VACUUM command is used to remove invalidated data files.
- B. Yes: Delta Lake ACID guarantees provide assurance that the DELETE command succeeded fully and permanently purged these records.
- C. No: the change data feed only tracks inserts and updates not deleted records.
- D. No: the Delta Lake DELETE command only provides ACID guarantees when combined with the MERGE INTO command

**Answer:** A

**Explanation:**

The DELETE operation in Delta Lake is ACID compliant, which means that once the operation is successful, the records are logically removed from the table. However, the underlying files that contained these records may still exist and be accessible via time travel to older versions of the table. To ensure that these records are physically removed and compliance with GDPR is maintained, a VACUUM command should be used to clean up these data files after a certain retention period. The VACUUM command will remove the files from the storage layer, and after this, the records will no longer be accessible.

**NEW QUESTION 83**

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 87**

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-acls/index.html>

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

**NEW QUESTION 92**

.....

## 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](#)**