Exam Questions Databricks-Certified-Professional-Data-Engineer

Databricks Certified Data Engineer Professional Exam

https://www.2passeasy.com/dumps/Databricks-Certified-Professional-Data-Engineer/

NEW QUESTION 1
Review the following error traceback:
Which statement describes the error being raised?

A. The code executed was PvSoark 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 error1. 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 user new to Databricks is trying to troubleshoot long execution times for some pipeline logic they are working on. Presently, the user is executing code cell-by-cell, using display() calls to confirm code is producing the logically correct results as new transformations are added to an operation. To get a measure of average time to execute, the user is running each cell multiple times interactively.
Which of the following adjustments will get a more accurate measure of how code is likely to perform in production?

A. Scala is the only language that can be accurately tested using interactive notebooks; because the best performance is achieved by using Scala code compiled to JAR
B. all PySpark and Spark SQL logic should be refactored.
C. The only way to meaningfully troubleshoot code execution times in development notebooks Is to use production-sized data and production-sized clusters with Run All execution.
D. Production code development should only be done using an IDE; executing code against a local build of open source Spark and Delta Lake will provide the most accurate benchmarks for how code will perform in production.
E. Calling display () forces a job to trigger, while many transformations will only add to the logical query plan; because of caching, repeated execution of the same logic does not provide meaningful results.
F. The Jobs Ul should be leveraged to occasionally run the notebook as a job and track execution time during incremental code development because Photon can only be enabled on clusters launched for scheduled jobs.

Answer: D

Explanation:
In Databricks notebooks, using the display() function triggers an action that forces Spark to execute the code and produce a result. However, Spark operations are generally divided into transformations and actions. Transformations create a new dataset from an existing one and are lazy, meaning they are not computed immediately but added to a logical plan. Actions, like display(), trigger the execution of this logical plan. Repeatedly running the same code cell can lead to misleading performance measurements due to caching. When a dataset is used multiple times, Spark's optimization mechanism caches it in memory, making subsequent executions faster. This behavior does not accurately represent the first-time execution performance in a production environment where data might not be cached yet.
To get a more realistic measure of performance, it is recommended to:
? Clear the cache or restart the cluster to avoid the effects of caching.
? Test the entire workflow end-to-end rather than cell-by-cell to understand the cumulative performance.
? Consider using a representative sample of the production data, ensuring it includes various cases the code will encounter in production.
References:
? Databricks Documentation on Performance Optimization: Databricks Performance Tuning
? Apache Spark Documentation: RDD Programming Guide - Understanding transformations and actions

NEW QUESTION 4

The business intelligence team has a dashboard configured to track various summary metrics for retail stories. 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 persists the products_per_order table in memory to quickly the dashboard with each query.
B. Populate the dashboard by configuring a nightly batch job to save the required 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 5
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 6
The data architect has mandated that all tables in the Lakehouse should be configured as external Delta Lake tables.
Which approach will ensure that this requirement is met?

A. Whenever a database is being created, make sure that the location keyword is used
B. When configuring an external data warehouse for all table storag
C. leverage Databricks for all ELT.
D. Whenever a table is being created, make sure that the location keyword is used.
E. When tables are created, make sure that the external keyword is used in the create table statement.
F. When the workspace is being configured, make sure that external cloud object storage has been mounted.

Answer: C

Explanation:
This is the correct answer because it ensures that this requirement is met. The requirement is that all tables in the Lakehouse should be configured as external Delta Lake tables. An external table 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 by using the location keyword to specify the path to an existing directory in a cloud storage system, such as DBFS or S3. By creating external tables, the data engineering team can avoid losing data if they drop or overwrite the table, as well as leverage existing data without moving or copying it. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Create an external table” section.

NEW QUESTION 7
Which statement describes the default execution mode for Databricks Auto Loader?

A. New files are identified by listing the input directory; new files are incrementally and idempotently loaded into the target Delta Lake table.

B. Cloud vendor-specific queue storage and notification services are configured to track newly arriving files; new files are incrementally and impotently into the target Delta Laketable.
C. Webhook trigger Databricks job to run anytime new data arrives in a source directory; new data automatically merged into target tables using rules inferred from the data.
D. New files are identified by listing the input directory; the target table is materialized by directory querying all valid files in the source directory.

Answer: A

Explanation:
Databricks Auto Loader simplifies and automates the process of loading data into Delta Lake. The default execution mode of the Auto Loader identifies new files by listing the input directory. It incrementally and idempotently loads these new files into the target Delta Lake table. This approach ensures that files are not missed and are processed exactly once, avoiding data duplication. The other options describe different mechanisms or integrations that are not part of the default behavior of the Auto Loader.
References:
? Databricks Auto Loader Documentation: Auto Loader Guide
? Delta Lake and Auto Loader: Delta Lake Integration

NEW QUESTION 8
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 9
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.broadca st.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
The Databricks CLI is use to trigger a run of an existing job by passing the job_id parameter. The response that the job run request has been submitted successfully includes a filed run_id.
Which statement describes what the number alongside this field represents?

A. The job_id is returned in this field.
B. The job_id and number of times the job has been are concatenated and returned.
C. The number of times the job definition has been run in the workspace.
D. The globally unique ID of the newly triggered run.

Answer: D

Explanation:
When triggering a job run using the Databricks CLI, the run_id field in the response represents a globally unique identifier for that particular run of the job. This run_id is distinct from the job_id. While the job_id identifies the job definition and is constant across all runs of that job, the run_id is unique to each execution and is used to track and query the status of that specific job run within the Databricks environment. This distinction allows users to manage and reference individual executions of a job directly.

NEW QUESTION 10
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 13
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 18
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 ofdefault 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. Storinq 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 21
A Delta Lake table was created with the below query:
Consider the following query:
DROP TABLE prod.sales_by_store -
If this statement is executed by a workspace admin, which result will occur?

A. Nothing will occur until a COMMIT command is executed.
B. The table will be removed from the catalog but the data will remain in storage.
C. The table will be removed from the catalog and the data will be deleted.
D. An error will occur because Delta Lake prevents the deletion of production data.
E. Data will be marked as deleted but still recoverable with Time Travel.

Answer: C

Explanation:
When a table is dropped in Delta Lake, the table is removed from the catalog and the data is deleted. This is because Delta Lake is a transactional storage layer that provides ACID guarantees. When a table is dropped, the transaction log is updated to reflect the deletion of the table and the data is deleted from the underlying storage. References:
? https://docs.databricks.com/delta/quick-start.html#drop-a-table
? https://docs.databricks.com/delta/delta-batch.html#drop-table

NEW QUESTION 22
The business reporting tem 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 jo to execute the pipeline once and 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 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 25
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 29
A developer has successfully configured credential for Databricks Repos and cloned a remote Git repository. Hey don not have privileges to make changes to the main branch, which is the only branch currently visible in their workspace.

Use Response to pull changes from the remote Git repository commit and push changes to a branch that appeared as a changes were pulled.

A. Use Repos to merge all differences and make a pull request back to the remote repository.
B. Use repos to merge all difference and make a pull request back to the remote repository.
C. Use Repos to create a new branch commit all changes and push changes to the remote Git repertory.
D. Use repos to create a fork of the remote repository commit all changes and make a pull request on the source repository

Answer: C

Explanation:
In Databricks Repos, when a user does not have privileges to make changes directly to the main branch of a cloned remote Git repository, the recommended approach is to create a new branch within the Databricks workspace. The developer can then make changes in this new branch, commit those changes, and push the new branch to the remote Git repository. This workflow allows for isolated development without affecting the main branch, enabling the developer to propose changes via a pull request from the new branch to the main branch in the remote repository. This method adheres to common Git collaboration workflows, fostering code review and collaboration while ensuring the integrity of the main branch.
References:
? Databricks documentation on using Repos with Git: https://docs.databricks.com/repos.html

NEW QUESTION 30
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 34
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 parameter1. 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 parameters2.
? 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 request3.
? 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 notebook4.
? 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 API5.
References: Widgets, Spark Configuration, input(), sys.argv, Notebooks

NEW QUESTION 37
The data governance team has instituted a requirement that all tables containing Personal Identifiable Information (PH) must be clearly annotated. This includes adding column comments, table comments, and setting the custom table property "contains_pii" = true.
The following SQL DDL statement is executed to create a new table:
Which command allows manual confirmation that these three requirements have been met?

A. DESCRIBE EXTENDED dev.pii test
B. DESCRIBE DETAIL dev.pii test
C. SHOW TBLPROPERTIES dev.pii test
D. DESCRIBE HISTORY dev.pii test
E. SHOW TABLES dev

Answer: A

Explanation:
This is the correct answer because it allows manual confirmation that these three requirements have been met. The requirements are that all tables containing Personal Identifiable Information (PII) must be clearly annotated, which includes adding column comments, table comments, and setting the custom table property "contains_pii" = true. The DESCRIBE EXTENDED command is used to display detailed information about a table, such as its schema, location, properties, and comments. By using this command on the dev.pii_test table, one can verify that the table has been created with the correct column comments, table comment, and custom table property as specified in the SQL DDL statement. Verified References: [Databricks Certified Data Engineer Professional], under "Lakehouse" section; Databricks Documentation, under "DESCRIBE EXTENDED" section.

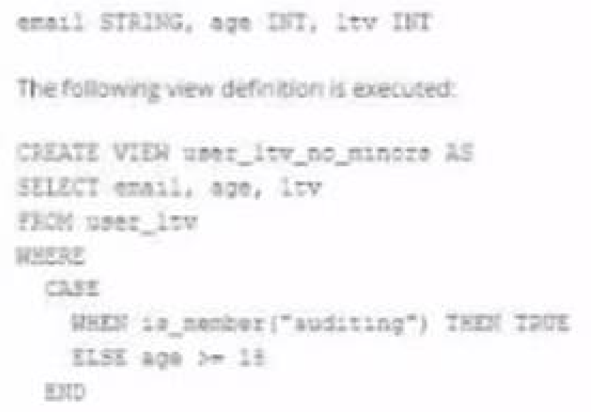
NEW QUESTION 39
A CHECK constraint has been successfully added to the Delta table named activity_details using the following logic:
A batch job is attempting to insert new records to the table, including a record where latitude = 45.50 and longitude = 212.67.
Which statement describes the outcome of 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 44
A table named user_ltv is being used to create a view that will be used by data analysis 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:

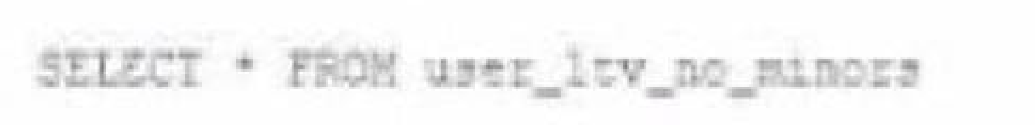


```
email STRING, age INT, ltv INT
```

The following view definition is executed:

```
CREATE VIEW user_ltv_no_minors AS
SELECT email, age, ltv
FROM user_ltv
WHERE
  CASE
    WHEN is_member("auditing") THEN TRUE
    ELSE age >= 18
  END
```

An analyze who is not a member of the auditing group executing the following query:



```
SELECT * FROM user_ltv_no_minors
```

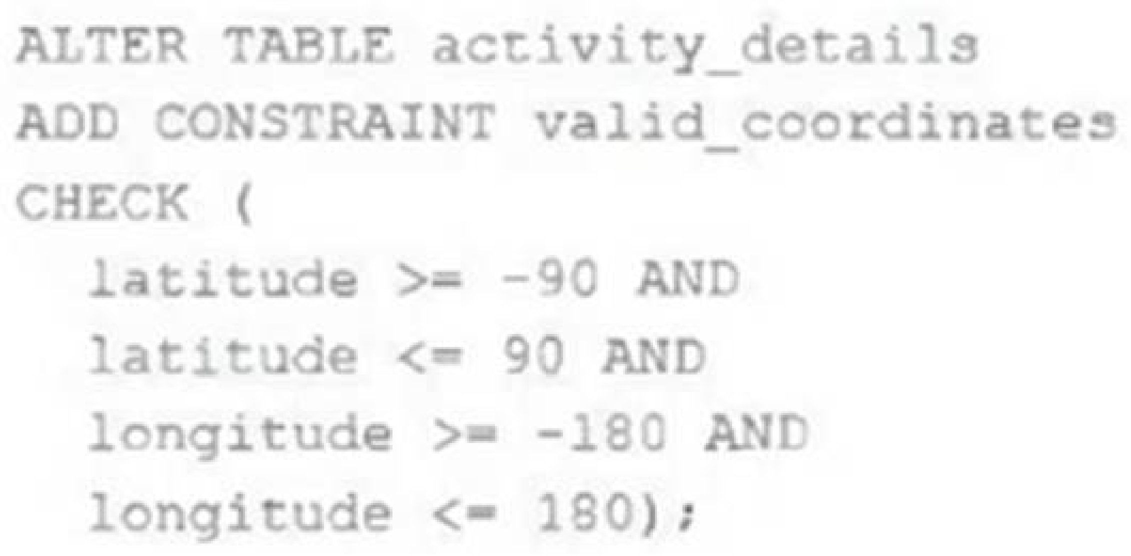
Which result will be returned by this query?

A. All columns will be displayed normally for those records that have an age greater than 18; records not meeting this condition will be omitted.
B. All columns will be displayed normally for those records that have an age greater than 17; records not meeting this condition will be omitted.
C. All age values less than 18 will be returned as null values all other columns will be returned with the values in user_ltv.
D. All records from all columns will be displayed with the values in user_ltv.

Answer: A

Explanation:
Given the CASE statement in the view definition, the result set for a user not in the auditing group would be constrained by the ELSE condition, which filters out records based on age. Therefore, the view will return all columns normally for records with an age greater than 18, as users who are not in the auditing group will not satisfy the is_member('auditing') condition. Records not meeting the age > 18 condition will not be displayed.

NEW QUESTION 45
The downstream consumers of a Delta Lake table have been complaining about data quality issues impacting performance in their applications. Specifically, they have complained that invalid latitude and longitude values in the activity_details table have been breaking their ability to use other geolocation processes.
A junior engineer has written the following code to add CHECK constraints to the Delta Lake table:



```
ALTER TABLE activity_details
ADD CONSTRAINT valid_coordinates
CHECK (
   latitude >= -90 AND
   latitude <= 90 AND
   longitude >= -180 AND
   longitude <= 180);
```

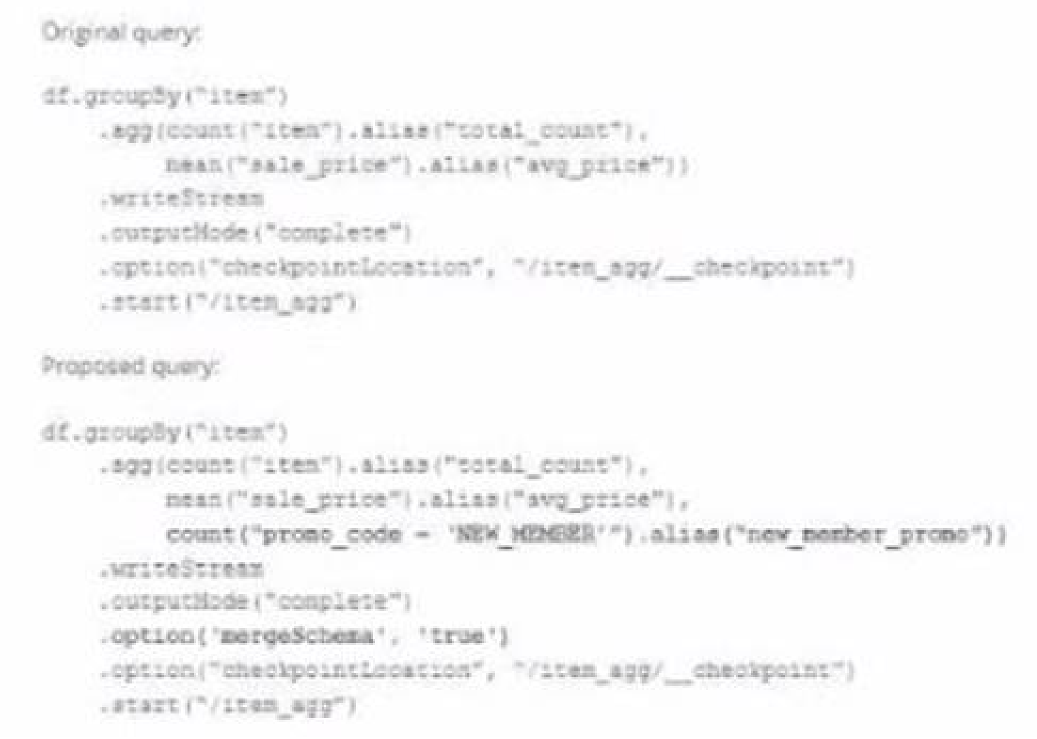
A senior engineer has confirmed the above logic is correct and the valid ranges for latitude and longitude are provided, but the code fails when executed.
Which statement explains the cause of this failure?

A. Because another team uses this table to support a frequently running application, two- phase locking is preventing the operation from committing.
B. The activity details table already exists; CHECK constraints can only be added during initial table creation.
C. The activity details table already contains records that violate the constraints; all existing data must pass CHECK constraints in order to add them to an existing table.
D. The activity details table already contains records; CHECK constraints can only be added prior to inserting values into a table.
E. The current table schema does not contain the field valid coordinates; schema evolution will need to be enabled before altering the table to add a constraint.

Answer: C

Explanation:
The failure is that the code to add CHECK constraints to the Delta Lake table fails when executed. The code uses ALTER TABLE ADD CONSTRAINT commands to add two CHECK constraints to a table named activity_details. The first constraint checks if the latitude value is between -90 and 90, and the second constraint checks if the longitude value is between -180 and 180. The cause of this failure is that the activity_details table already contains records that violate these constraints, meaning that they have invalid latitude or longitude values outside of these ranges. When adding CHECK constraints to an existing table, Delta Lake verifies that all existing data satisfies the constraints before adding them to the table. If any record violates the constraints, Delta Lake throws an exception and aborts the operation. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Add a CHECK constraint to an existing table” section. https://docs.databricks.com/en/sql/language-manual/sql-ref-syntax-ddl-alter-table.html#add-constraint

NEW QUESTION 49
A data team's Structured Streaming job is configured to calculate running aggregates for item sales to update a downstream marketing dashboard. The marketing team has introduced a new field to track the number of times this promotion code is used for each item. A junior data engineer suggests updating the existing query as follows: Note that proposed changes are in bold.



Original query:

```
df.groupBy("item")
    .agg(count("item").alias("total_count"),
         mean("sale_price").alias("avg_price"))
    .writeStream
    .outputMode("complete")
    .option("checkpointLocation", "/item_agg/__checkpoint")
    .start("/item_agg")
```

Proposed query:

```
df.groupBy("item")
    .agg(count("item").alias("total_count"),
         mean("sale_price").alias("avg_price"),
         count("promo_code = 'NEW_MEMBER'").alias("new_member_promo"))
    .writeStream
    .outputMode("complete")
    .option('mergeSchema', 'true')
    .option("checkpointLocation", "/item_agg/__checkpoint")
    .start("/item_agg")
```

Which step must also be completed to put the proposed query into production?

A. Increase the shuffle partitions to account for additional aggregates
B. Specify a new checkpointlocation
C. Run REFRESH TABLE delta, /item_agg'
D. Remove .option (mergeSchema', true') from the streaming write

Answer: B

Explanation:
When introducing a new aggregation or a change in the logic of a Structured Streaming query, it is generally necessary to specify a new checkpoint location. This is because the checkpoint directory contains metadata about the offsets and the state of the aggregations of a streaming query. If the logic of the query changes, such as including a new aggregation field, the state information saved in the current checkpoint would not be compatible with the new logic, potentially leading to incorrect results or failures. Therefore, to accommodate the new field and ensure the streaming job has the correct starting point and state information for aggregations, a new checkpoint location should be specified. References:
? Databricks documentation on Structured Streaming:
https://docs.databricks.com/spark/latest/structured-streaming/index.html
? Databricks documentation on streaming checkpoints: https://docs.databricks.com/spark/latest/structured- streaming/production.html#checkpointing

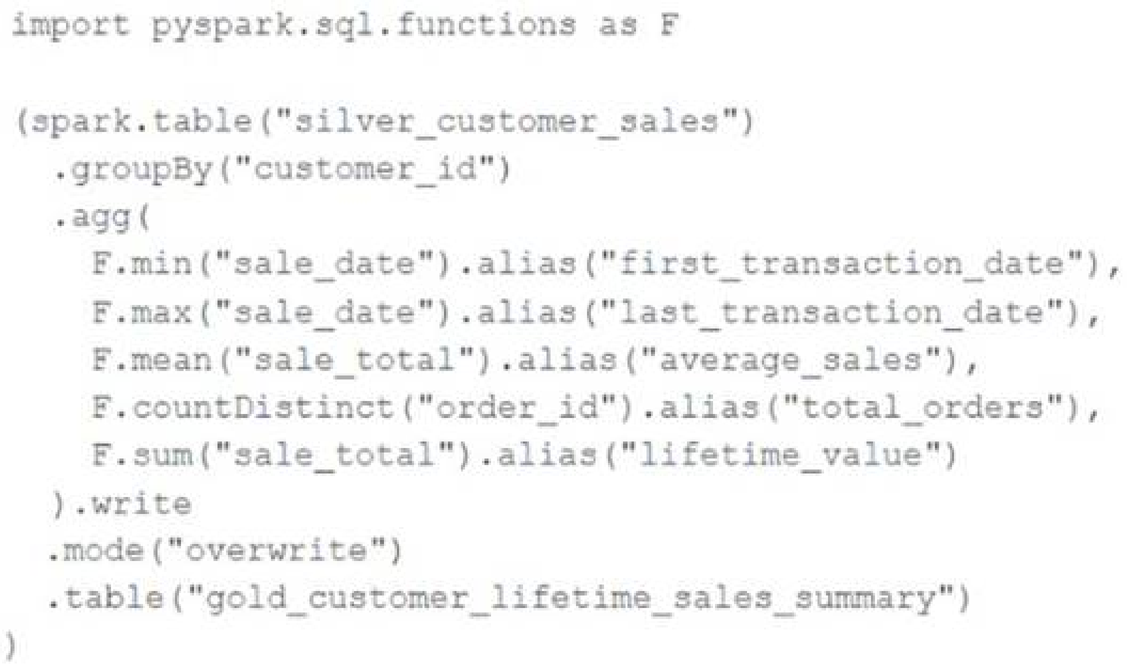
NEW QUESTION 52
The Databricks workspace administrator has configured interactive clusters for each of the data engineering groups. To control costs, clusters are set to terminate after 30 minutes of inactivity. Each user should be able to execute workloads against their assigned clusters at any time 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 57
The data engineering team maintains the following code:



```
import pyspark.sql.functions as F

(spark.table("silver_customer_sales")
  .groupBy("customer_id")
  .agg(
    F.min("sale_date").alias("first_transaction_date"),
    F.max("sale_date").alias("last_transaction_date"),
    F.mean("sale_total").alias("average_sales"),
    F.countDistinct("order_id").alias("total_orders"),
    F.sum("sale_total").alias("lifetime_value")
  ).write
  .mode("overwrite")
  .table("gold_customer_lifetime_sales_summary")
)
```

Assuming that this code produces logically correct results and the data in the source table has been de-duplicated and validated, which statement describes what will occur when this code is executed?

A. The silver_customer_sales table will be overwritten by aggregated values calculated from all records in the gold_customer_lifetime_sales_summary table as a batch job.
B. A batch job will update the gold_customer_lifetime_sales_summary table, replacing only those rows that have different values than the current version of the table, using customer_id as the primary key.
C. The gold_customer_lifetime_sales_summary table will be overwritten by aggregated values calculated from all records in the silver_customer_sales table as a batch job.
D. An incremental job will leverage running information in the state store to update aggregate values in the gold_customer_lifetime_sales_summary table.
E. An incremental job will detect if new rows have been written to the silver_customer_sales table; if new rows are detected, all aggregates will be recalculated and used to overwrite the gold_customer_lifetime_sales_summary table.

Answer: C

Explanation:
This code is using the pyspark.sql.functions library to group the silver_customer_sales table by customer_id and then aggregate the data using the minimum sale date, maximum sale total, and sum of distinct order ids. The resulting aggregated data is then written to the gold_customer_lifetime_sales_summary table, overwriting any existing data in that table. This is a batch job that does not use any incremental or streaming logic, and does not perform any merge or update operations. Therefore, the code will overwrite the gold table with the aggregated values from the silver table every time it is executed. References:
? https://docs.databricks.com/spark/latest/dataframes-datasets/introduction-to-dataframes-python.html
? https://docs.databricks.com/spark/latest/dataframes-datasets/transforming-data- with-dataframes.html
? https://docs.databricks.com/spark/latest/dataframes-datasets/aggregating-data- with-dataframes.html

NEW QUESTION 60
A team of data engineer are adding tables to a DLT pipeline that contain repetitive expectations for many of the same data quality checks.
One member of the team suggests reusing these data quality rules across all tables defined for this pipeline.
What approach would allow them to do this?

A. Maintain data quality rules in a Delta table outside of this pipeline's target schema, providing the schema name as a pipeline parameter.
B. Use global Python variables to make expectations visible across DLT notebooks included in the same pipeline.
C. Add data quality constraints to tables in this pipeline using an external job with access to pipeline configuration files.
D. Maintain data quality rules in a separate Databricks notebook that each DLT notebook of file.

Answer: A

Explanation:

Maintaining data quality rules in a centralized Delta table allows for the reuse of these rules across multiple DLT (Delta Live Tables) pipelines. By storing these rules outside the pipeline's target schema and referencing the schema name as a pipeline parameter, the team can apply the same set of data quality checks to different tables within the pipeline. This approach ensures consistency in data quality validations and reduces redundancy in code by not having to replicate the same rules in each DLT notebook or file. References:
? Databricks Documentation on Delta Live Tables: Delta Live Tables Guide

NEW QUESTION 65
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 69
The data engineer is using Spark's MEMORY_ONLY storage level.
Which indicators should the data engineer look for in the spark UI's Storage tab to signal that a cached table is not performing optimally?

A. Size on Disk is> 0
B. The number of Cached Partitions> the number of Spark Partitions
C. The RDD Block Name included the '' annotation signaling failure to cache
D. On Heap Memory Usage is within 75% of off Heap Memory usage

Answer: C

Explanation:
In the Spark UI's Storage tab, an indicator that a cached table is not performing optimally would be the presence of the _disk annotation in the RDD Block Name. This annotation indicates that some partitions of the cached data have been spilled to disk because there wasn't enough memory to hold them. This is suboptimal because accessing data from disk is much slower than from memory. The goal of caching is to keep data in memory for fast access, and a spill to disk means that this goal is not fully achieved.

NEW QUESTION 72
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_iot 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_iot

Answer: D

Explanation:
In Databricks Lakehouse, to retain manually deleted or updated records in the raw_iot 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_iot 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 76
Assuming that the Databricks CLI has been installed and configured correctly, which Databricks CLI command can be used to upload a custom Python Wheel to object storage mounted with the DBFS for use with a production job?

A. configure
B. fs
C. jobs
D. libraries
E. workspace

Answer: B

Explanation:
The libraries command group allows you to install, uninstall, and list libraries on Databricks clusters. You can use the libraries install command to install a custom

Python Wheel on a cluster by specifying the --whl option and the path to the wheel file. For example, you can use the following command to install a custom Python Wheel named mylib-0.1-py3-none-any.whl on a cluster with the id 1234-567890-abcde123:
databricks libraries install --cluster-id1234-567890-abcde123--whldbfs:/mnt/mylib/mylib-0.1-py3-none-any.whl
This will upload the custom Python Wheel to the cluster and make it available for use with a production job. You can also use the libraries uninstall command to uninstall a library from a cluster, and the libraries list command to list the libraries installed on a cluster. References:
? Libraries CLI (legacy): https://docs.databricks.com/en/archive/dev-tools/cli/libraries-cli.html
? Library operations: https://docs.databricks.com/en/dev- tools/cli/commands.html#library-operations
? Install or update the Databricks CLI: https://docs.databricks.com/en/dev- tools/cli/install.html

NEW QUESTION 78
What is the first of a Databricks Python notebook when viewed in a text editor?

A. %python
B. % Databricks notebook source
C. -- Databricks notebook source
D. //Databricks notebook source

Answer: B

Explanation:
When viewing a Databricks Python notebook in a text editor, the first line indicates the format and source type of the notebook. The correct option is % Databricks notebook source, which is a magic command that specifies the start of a Databricks notebook source file.

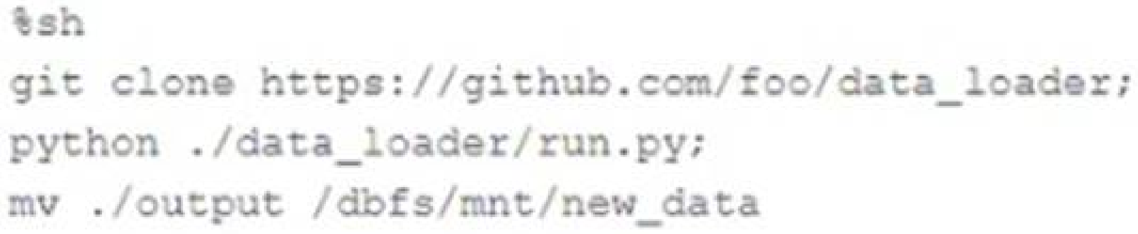
NEW QUESTION 79
An upstream system is emitting change data capture (CDC) logs that are being written to a cloud object storage directory. Each record in the log indicates the change type (insert, update, or delete) and the values for each field after the change. The source table has a primary key identified 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 abronze 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 84
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
```

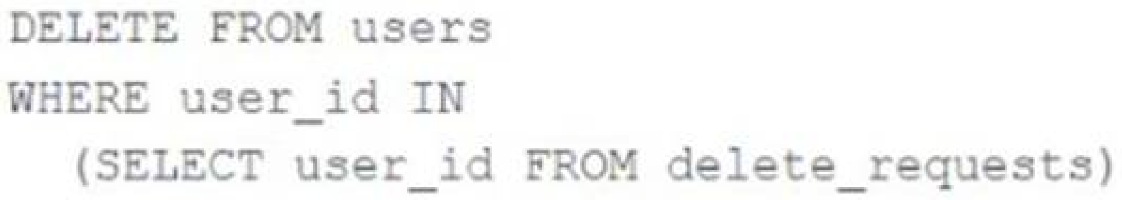
The code executes successfully and provides the logically correct results, however, it takes over 20 minutes to extract and load around 1 GB of data.
Which statement is a possible explanation for this behavior?

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 88
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)
```

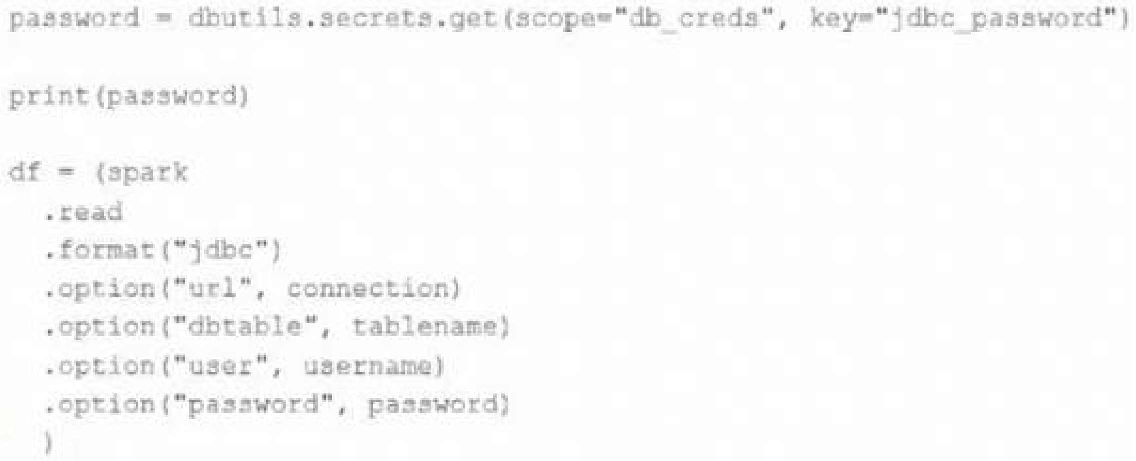
Assuming that user_id is a unique identifying key and that delete_requests contains all users that have requested deletion, which statement describes whether successfully executing the above logic guarantees that the records to be deleted are no longer accessible and why?

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 92
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)
  )
```

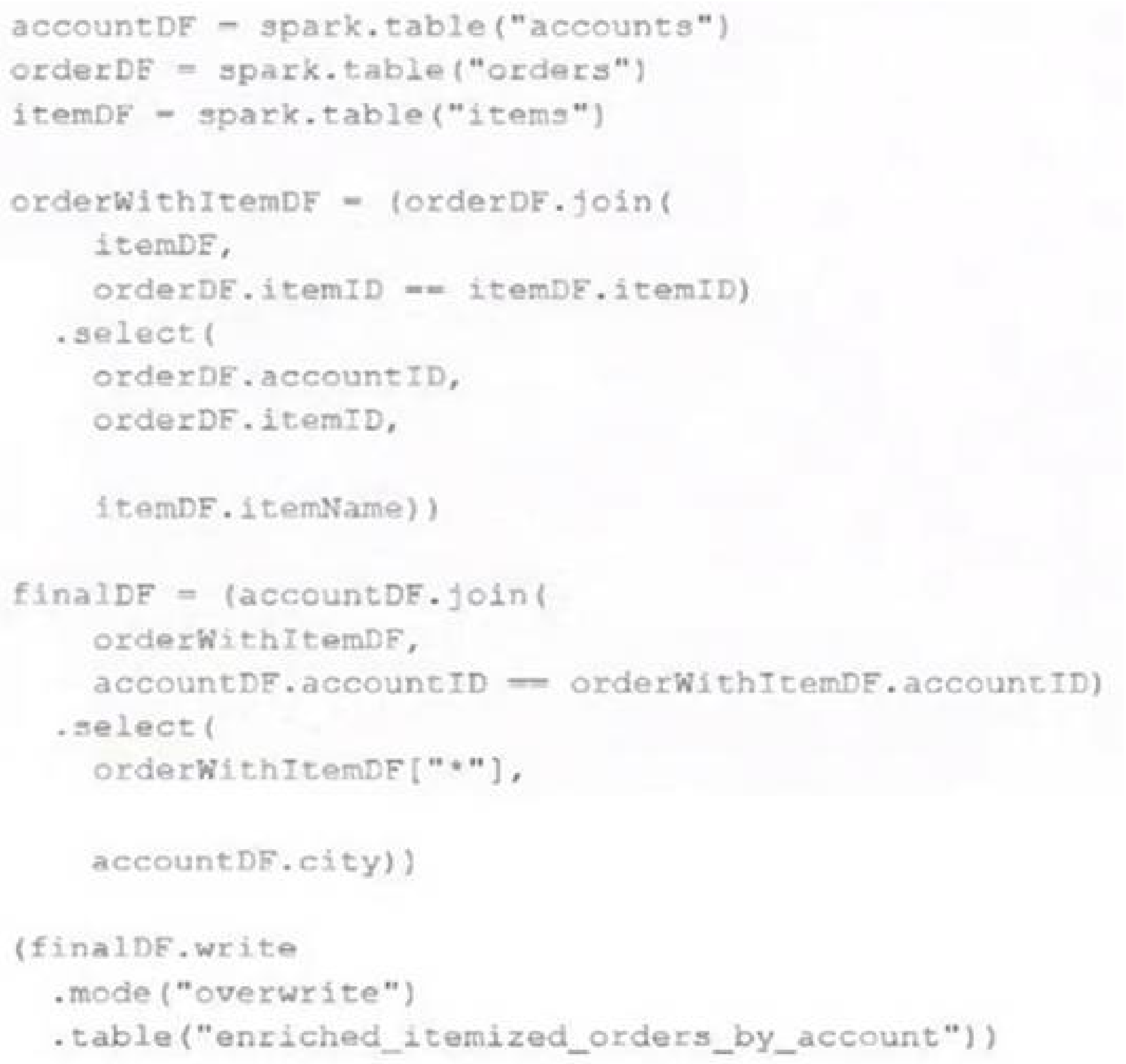
Which statement describes what will happen when the above code is executed?

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 94
The data engineering team maintains the following code:



```
accountDF = spark.table("accounts")
orderDF = spark.table("orders")
itemDF = spark.table("items")

orderWithItemDF = (orderDF.join(
    itemDF,
    orderDF.itemID == itemDF.itemID)
  .select(
    orderDF.accountID,
    orderDF.itemID,

    itemDF.itemName))

finalDF = (accountDF.join(
    orderWithItemDF,
    accountDF.accountID == orderWithItemDF.accountID)
  .select(
    orderWithItemDF["*"],

    accountDF.city))

(finalDF.write
  .mode("overwrite")
  .table("enriched_itemized_orders_by_account"))
```

Assuming that this code produces logically correct results and the data in the source tables has been de-duplicated and validated, which statement describes what will occur when this code is executed?

A. A batch job will update the enriched_itemized_orders_by_account table, replacing only those rows that have different values than the current version of the table, using accountID as the primary key.
B. The enriched_itemized_orders_by_account table will be overwritten using the current valid version of data in each of the three tables referenced in the join logic.
C. An incremental job will leverage information in the state store to identify unjoined rows in the source tables and write these rows to the enriched_iteinized_orders_by_account table.
D. An incremental job will detect if new rows have been written to any of the source tables; if new rows are detected, all results will be recalculated and used to overwrite the enriched_itemized_orders_by_account table.
E. No computation will occur until enriched_itemized_orders_by_account is queried; upon query materialization, results will be calculated using the current valid version of data in each of the three tables referenced in the join logic.

Answer: B

Explanation:
This is the correct answer because it describes what will occur when this code is executed. The code uses three Delta Lake tables as input sources: accounts, orders, and order_items. These tables are joined together using SQL queries to create a view called new_enriched_itemized_orders_by_account, which contains information about each order item and its associated account details. Then, the code uses write.format("delta").mode("overwrite") to overwrite a target table called enriched_itemized_orders_by_account using the data from the view. This means that every time this code is executed, it will replace all existing data in the target table with new data based on the current valid version of data in each of the three input tables. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Write to Delta tables" section.

NEW QUESTION 96
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 100
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 systems1. DBFS is not a physical file system, but a layer over the object storage that provides a unified view of data across different data sources1. 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 container2. 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 mounting3. Both the DBFS root and mounted storage can be accessed when using %sh in a Databricks notebook, as long as the cluster has FUSE enabled4. The DBFS root does not store files in ephemeral block volumes attached to the driver, but in the object storage associated with the workspace1. Mounted directories will persist saved data to external storage between sessions, unless they are unmounted or deleted3. References: DBFS, Work with files on Azure Databricks, Mounting cloud object storage on Azure Databricks, Access DBFS with FUSE

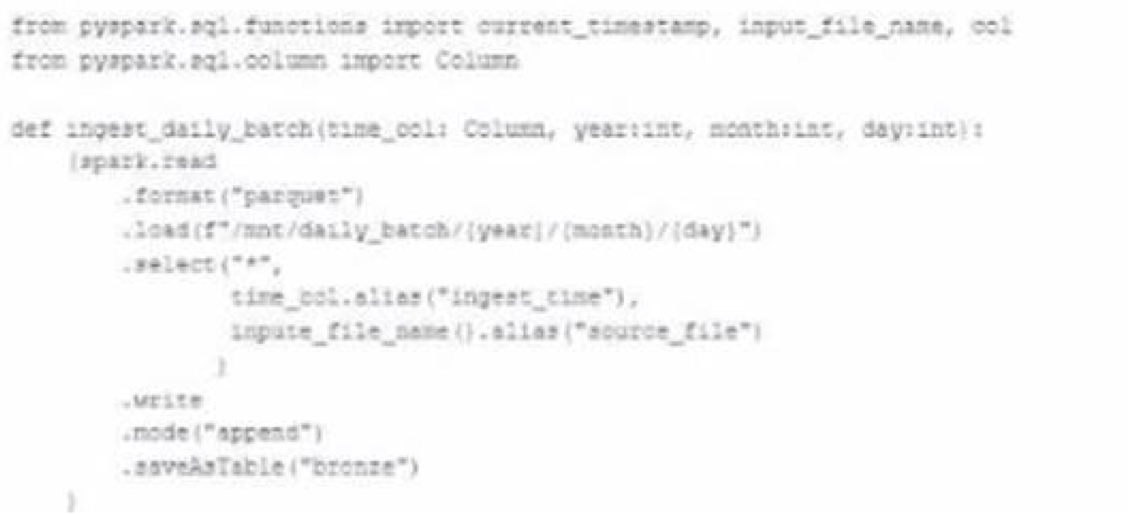
NEW QUESTION 103
Which statement describes Delta Lake optimized writes?

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 107
A nightly job ingests data into a Delta Lake table using the following code:

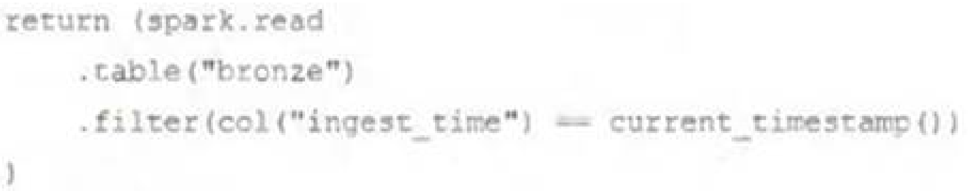


```
from pyspark.sql.functions import current_timestamp, input_file_name, col
from pyspark.sql.column import Column

def ingest_daily_batch(time_col: Column, year:int, month:int, day:int):
    (spark.read
        .format("parquet")
        .load(f"/mnt/daily_batch/{year}/{month}/{day}")
        .select("*",
                time_col.alias("ingest_time"),
                inpute_file_name().alias("source_file")
               )
        .write
        .mode("append")
        .saveAsTable("bronze")
    )
```

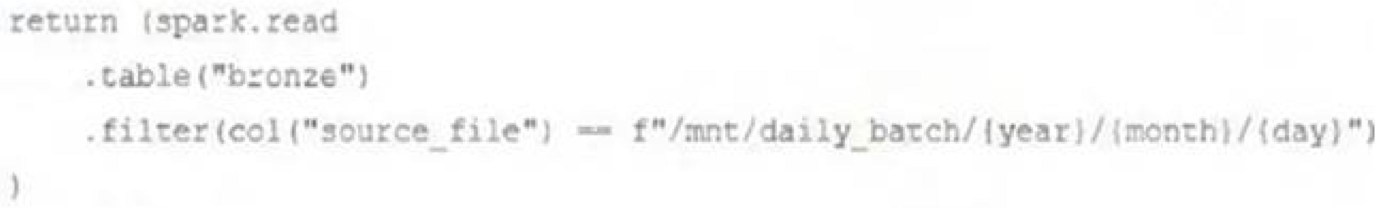
The next step in the pipeline requires a function that returns an object that can be used to manipulate new records that have not yet been processed to the next table in the pipeline.
Which code snippet completes this function definition? def new_records():

A. return spark.readStream.table("bronze")
B. return spark.readStream.load("bronze")C.



```
return (spark.read
    .table("bronze")
    .filter(col("ingest_time") == current_timestamp())
)
```

D.return spark.read.option("readChangeFeed", "true").table ("bronze")
C.



```
return (spark.read
    .table("bronze")
    .filter(col("source_file") == f"/mnt/daily_batch/{year}/{month}/{day}")
)
```

Answer: E

Explanation:
https://docs.databricks.com/en/delta/delta-change-data-feed.html

NEW QUESTION 109
A Databricks SQL dashboard has been configured to monitor the total number of records present in a collection of Delta Lake tables using the following query pattern:
SELECT COUNT (*) FROM table -

Which of the following describes how results are generated each time the dashboard is updated?

A. The total count of rows is calculated by scanning all data files
B. The total count of rows will be returned from cached results unless REFRESH is run
C. The total count of records is calculated from the Delta transaction logs
D. The total count of records is calculated from the parquet file metadata
E. The total count of records is calculated from the Hive metastore

Answer: C

Explanation:
https://delta.io/blog/2023-04-19-faster-aggregations-metadata/#:~:text=You%20can%20get%20the%20number,a%20given%20Delta%20table%20version.

NEW QUESTION 114
Where in the Spark UI can one diagnose a performance problem induced by not leveraging predicate push-down?

A. In the Executor's log file, by gripping for "predicate push-down"
B. In the Stage's Detail screen, in the Completed Stages table, by noting the size of data read from the Input column
C. In the Storage Detail screen, by noting which RDDs are not stored on disk
D. In the Delta Lake transaction lo
E. by noting the column statistics
F. In the Query Detail screen, by interpreting the Physical Plan

Answer: E

Explanation:
This is the correct answer because it is where in the Spark UI one can diagnose a performance problem induced by not leveraging predicate push-down. Predicate push-down is an optimization technique that allows filtering data at the source before loading it into memory or processing it further. This can improve performance and reduce I/O costs by avoiding reading unnecessary data. To leverage predicate push-down, one should use supported data sources and formats, such as Delta Lake, Parquet, or JDBC, and use filter expressions that can be pushed down to the source. To diagnose a performance problem induced by not leveraging predicate push-down, one can use the Spark UI to access the Query Detail screen, which shows information about a SQL query executed on a Spark cluster. The Query Detail screen includes the Physical Plan, which is the actual plan executed by Spark to perform the query. The Physical Plan shows the physical operators used by Spark, such as Scan, Filter, Project, or Aggregate, and their input and output statistics, such as rows and bytes. By interpreting the Physical Plan, one can see if the filter expressions are pushed down to the source or not, and how much data is read or processed by each operator. Verified References: [Databricks Certified Data Engineer Professional], under “Spark Core” section; Databricks Documentation, under “Predicate pushdown” section; Databricks Documentation, under “Query detail page” section.

NEW QUESTION 115
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 120
The view updates represents an incremental batch of all newly ingested data to be inserted or updated in the customers table.
The following logic is used to process these records.
MERGE INTO customers USING (
SELECT updates.customer_id as merge_ey, updates .* FROM updates
UNION ALL
SELECT NULL as merge_key, updates .* FROM updates JOIN customers
ON updates.customer_id = customers.customer_id
WHERE customers.current = true AND updates.address <> customers.address
) staged_updates
ON customers.customer_id = mergekey
WHEN MATCHED AND customers. current = true AND customers.address <> staged_updates.address THEN
UPDATE SET current = false, end_date = staged_updates.effective_date WHEN NOT MATCHED THEN
INSERT (customer_id, address, current, effective_date, end_date)
VALUES (staged_updates.customer_id, staged_updates.address, true, staged_updates.effective_date, null)
Which statement describes this implementation?

A. The customers table is implemented as a Type 2 table; old values are overwritten and new customers are appended.
B. The customers table is implemented as a Type 1 table; old values are overwritten by new values and no history is maintained.
C. The customers table is implemented as a Type 2 table; old values are maintained but marked as no longer current and new values are inserted.
D. The customers table is implemented as a Type 0 table; all writes are append only with no changes to existing values.

Answer: C

Explanation:
The provided MERGE statement is a classic implementation of a Type 2 SCD in a data warehousing context. In this approach, historical data is preserved by

keeping old records (marking them as not current) and adding new records for changes. Specifically, when a match is found and there's a change in the address, the existing record in the customers table is updated to mark it as no longer current (current = false), and an end date is assigned (end_date = staged_updates.effective_date). A new record for the customer is then inserted with the updated information, marked as current. This method ensures that the full history of changes to customer information is maintained in the table, allowing for time-based analysis of customer data.References: Databricks documentation on implementing SCDs using Delta Lake and the MERGE statement (https://docs.databricks.com/delta/delta-update.html#upsert-into-a-table-using-merge).

NEW QUESTION 122
Which statement describes Delta Lake Auto Compaction?

A. An asynchronous job runs after the write completes to detect if files could be further compacted; if yes, an optimize job is executed toward a default of 1 GB.
B. Before a Jobs cluster terminates, optimize is executed on all tables modified during the most recent job.
C. Optimized writes use logical partitions instead of directory partitions; because partition boundaries are only represented in metadata, fewer small files are written.
D. Data is queued in a messaging bus instead of committing data directly to memory; all data is committed from the messaging bus in one batch once the job is complete.
E. An asynchronous job runs after the write completes to detect if files could be further compacted; if yes, an optimize job is executed toward a default of 128 MB.

Answer: E

Explanation:
This is the correct answer because it describes the behavior of Delta Lake Auto Compaction, which is a feature that automatically optimizes the layout of Delta Lake tables by coalescing small files into larger ones. Auto Compaction runs as an asynchronous job after a write to a table has succeeded and checks if files within a partition can be further compacted. If yes, it runs an optimize job with a default target file size of 128 MB. Auto Compaction only compacts files that have not been compacted previously. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Auto Compaction for Delta Lake on Databricks” section.
"Auto compaction occurs after a write to a table has succeeded and runs synchronously on the cluster that has performed the write. Auto compaction only compacts files that haven’t been compacted previously."
https://learn.microsoft.com/en-us/azure/databricks/delta/tune-file-size

NEW QUESTION 123
A new data engineer notices that a critical field was omitted from an application that writes its Kafka source to Delta Lake. This happened even though the critical field was in the Kafka source. That field was further missing from data written to dependent, long-term storage. The retention threshold on the Kafka service is seven days. The pipeline has been in production for three months.
Which describes how Delta Lake can help to avoid data loss of this nature in the future?

A. The Delta log and Structured Streaming checkpoints record the full history of the Kafkaproducer.
B. Delta Lake schema evolution can retroactively calculate the correct value for newly added fields, as long as the data was in the original source.
C. Delta Lake automatically checks that all fields present in the source data are included in the ingestion layer.
D. Data can never be permanently dropped or deleted from Delta Lake, so data loss is not possible under any circumstance.
E. Ingestine all raw data and metadata from Kafka to a bronze Delta table creates a permanent, replayable history of the data state.

Answer: E

Explanation:
This is the correct answer because it describes how Delta Lake can help to avoid data loss of this nature in the future. By ingesting all raw data and metadata from Kafka to a bronze Delta table, Delta Lake creates a permanent, replayable history of the data state that can be used for recovery or reprocessing in case of errors or omissions in downstream applications or pipelines. Delta Lake also supports schema evolution, which allows adding new columns to existing tables without affecting existing queries or pipelines. Therefore, if a critical field was omitted from an application that writes its Kafka source to Delta Lake, it can be easily added later and the data can be reprocessed from the bronze table without losing any information. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Delta Lake core features” section.

NEW QUESTION 126
......