Databricks

Exam Questions Databricks-Certified-Professional-Data-Engineer

Databricks Certified Data Engineer Professional Exam

NEW QUESTION 1
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 2
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 UI 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 3
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 4
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 5
A junior data engineer is migrating a workload from a relational database system to the Databricks Lakehouse. The source system uses a star schema, leveraging foreign key constrains and multi-table inserts to validate records on write.
Which consideration will impact the decisions made by the engineer while migrating this workload?

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

Answer: A

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

NEW QUESTION 6
In order to prevent accidental commits to production data, a senior data engineer has instituted a policy that all development work will reference clones of Delta Lake tables. After testing both deep and shallow clone, development tables are created using shallow clone.
A few weeks after initial table creation, the cloned versions of several tables implemented as Type 1 Slowly Changing Dimension (SCD) stop working. The transaction logs for the source tables show that vacuum was run the day before.
Why are the cloned tables no longer working?

A. The data files compacted by vacuum are not tracked by the cloned metadata; running refresh on the cloned table will pull in recent changes.
B. Because Type 1 changes overwrite existing records, Delta Lake cannot guarantee data consistency for cloned tables.
C. The metadata created by the clone operation is referencing data files that were purged as invalid by the vacuum command
D. Running vacuum automatically invalidates any shallow clones of a table; deep clone should always be used when a cloned table will be repeatedly queried.

Answer: C

Explanation:
In Delta Lake, a shallow clone creates a new table by copying the metadata of the source table without duplicating the data files. When the vacuum command is run on the source table, it removes old data files that are no longer needed to maintain the transactional log's integrity, potentially including files referenced by the shallow clone's metadata. If these files are purged, the shallow cloned tables will reference non-existent data files, causing them to stop working properly. This highlights the dependency of shallow clones on the source table's data files and the impact of data management operations like vacuum on these clones.References: Databricks documentation on Delta Lake, particularly the sections on cloning tables (shallow and deep cloning) and data retention with the vacuum command (https://docs.databricks.com/delta/index.html).

NEW QUESTION 7
An hourly batch job is configured to ingest data files from a cloud object storage container where each batch represent all records produced by the source system in a given hour. The batch job to process these records into the Lakehouse is sufficiently delayed to ensure no late-arriving data is missed. The user_id field represents a unique key for the data, which has the following schema:
user_id BIGINT, username STRING, user_utc STRING, user_region STRING, last_login BIGINT, auto_pay BOOLEAN, last_updated BIGINT
New records are all ingested into a table named account_history which maintains a full record of all data in the same schema as the source. The next table in the system is named account_current and is implemented as a Type 1 table representing the most recent value for each unique user_id.
Assuming there are millions of user accounts and tens of thousands of records processed hourly, which implementation can be used to efficiently update the described account_current table as part of each hourly batch job?

A. Use Auto Loader to subscribe to new files in the account history directory; configure a Structured Streaminq trigger once job to batch update newly detected files into the account current table.
B. Overwrite the account current table with each batch using the results of a query against the account history table grouping by user id and filtering for the max value of last updated.
C. Filter records in account history using the last updated field and the most recent hour processed, as well as the max last iogin by user id write a merge statement to update or insert the most recent value for each user id.
D. Use Delta Lake version history to get the difference between the latest version of account history and one version prior, then write these records to account current.
E. Filter records in account history using the last updated field and the most recent hour processed, making sure to deduplicate on username; write a merge statement to update orinsert themost recent value for each username.

Answer: C

Explanation:

This is the correct answer because it efficiently updates the account current table with only the most recent value for each user id. The code filters records in account history using the last updated field and the most recent hour processed, which means it will only process the latest batch of data. It also filters by the max last login by user id, which means it will only keep the most recent record for each user id within that batch. Then, it writes a merge statement to update or insert the most recent value for each user id into account current, which means it will perform an upsert operation based on the user id column. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Upsert into a table using merge” section.

NEW QUESTION 8
Which REST API call can be used to review the notebooks configured to run as tasks in a multi-task job?

A. /jobs/runs/list
B. /jobs/runs/get-output
C. /jobs/runs/get
D. /jobs/get
E. /jobs/list

Answer: D

Explanation:
This is the correct answer because it is the REST API call that can be used to review the notebooks configured to run as tasks in a multi-task job. The REST API is an interface that allows programmatically interacting with Databricks resources, such as clusters, jobs, notebooks, or tables. The REST API uses HTTP methods, such as GET, POST, PUT, or DELETE, to perform operations on these resources. The /jobs/get endpoint is a GET method that returns information about a job given its job ID. The information includes the job settings, such as the name, schedule, timeout, retries, email notifications, and tasks. The tasks are the units of work that a job executes. A task can be a notebook task, which runs a notebook with specified parameters; a jar task, which runs a JAR uploaded to DBFS with specified main class and arguments; or a python task, which runs a Python file uploaded to DBFS with specified parameters. A multi-task job is a job that has more than one task configured to run in a specific order or in parallel. By using the /jobs/get endpoint, one can review the notebooks configured to run as tasks in a multi-task job.
Verified References: [Databricks Certified Data Engineer Professional], under “Databricks Jobs” section; Databricks Documentation, under “Get” section; Databricks Documentation, under “JobSettings” section.

NEW QUESTION 9
A junior data engineer has manually configured a series of jobs using the Databricks Jobs UI. Upon reviewing their work, the engineer realizes that they are listed as the "Owner" for each job. They attempt to transfer "Owner" privileges to the "DevOps" group, but cannot successfully accomplish this task.
Which statement explains what is preventing this privilege transfer?

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

Answer: E

Explanation:
The reason why the junior data engineer cannot transfer “Owner” privileges to the “DevOps” group is that Databricks jobs must have exactly one owner, and the owner must be an individual user, not a group. A job cannot have more than one owner, and a job cannot have a group as an owner. The owner of a job is the user who created the job, or the user who was assigned the ownership by another user. The owner of a job has the highest level of permission on the job, and can grant or revoke permissions to other users or groups. However, the owner cannot transfer the ownership to a group, only to another user. Therefore, the junior data engineer’s attempt to transfer “Owner” privileges to the “DevOps” group is not possible. References:
? Jobs access control: https://docs.databricks.com/security/access-control/table-acls/index.html
? Job permissions: https://docs.databricks.com/security/access-control/table-acls/privileges.html#job-permissions

NEW QUESTION 10
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 10
What statement is true regarding the retention of job run history?

A. It is retained until you export or delete job run logs
B. It is retained for 30 days, during which time you can deliver job run logs to DBFS or S3
C. t 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 13
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 16
A junior data engineer is working to implement logic for a Lakehouse table named silver_device_recordings. The source data contains 100 unique fields in a highly nested JSON structure.
The silver_device_recordings table will be used downstream for highly selective joins on a number of fields, and will also be leveraged by the machine learning team to filter on a handful of relevant fields, in total, 15 fields have been identified that will often be used for filter and join logic.
The data engineer is trying to determine the best approach for dealing with these nested fields before declaring the table schema.
Which of the following accurately presents information about Delta Lake and Databricks that may Impact their decision-making process?

A. Because Delta Lake uses Parquet for data storage, Dremel encoding information for nesting can be directly referenced by the Delta transaction log.
B. Tungsten encoding used by Databricks is optimized for storing string data: newly-added native support for querying JSON strings means that string types are always most efficient.
C. Schema inference and evolution on Databricks ensure that inferred types will always accurately match the data types used by downstream systems.
D. By default Delta Lake collects statistics on the first 32 columns in a table; these statistics are leveraged for data skipping when executing selective queries.

Answer: D

Explanation:
Delta Lake, built on top of Parquet, enhances query performance through data skipping, which is based on the statistics collected for each file in a table. For tables with a large number of columns, Delta Lake by default collects and stores statistics only for the first 32 columns. These statistics include min/max values and null counts, which are used to optimize query execution by skipping irrelevant data files. When dealing with highly nested JSON structures, understanding this behavior is crucial for schema design, especially when determining which fields should be flattened or prioritized in the table structure to leverage data skipping efficiently for performance optimization.References: Databricks documentation on Delta Lake optimization techniques, including data skipping and statistics collection (https://docs.databricks.com/delta/optimizations/index.html).

NEW QUESTION 21
A Delta Lake table representing metadata about content posts from users has the following schema:
user_id LONG, post_text STRING, post_id STRING, longitude FLOAT, latitude FLOAT, post_time TIMESTAMP, date DATE
This table is partitioned by the date column. A query is run with the following filter: longitude < 20 & longitude > -20
Which statement describes how data will be filtered?

A. Statistics in the Delta Log will be used to identify partitions that might Include files in the filtered range.
B. No file skipping will occur because the optimizer does not know the relationship between the partition column and the longitude.
C. The Delta Engine will use row-level statistics in the transaction log to identify the flies that meet the filter criteria.
D. Statistics in the Delta Log will be used to identify data files that might include records in the filtered range.
E. The Delta Engine will scan the parquet file footers to identify each row that meets the filter criteria.

Answer: D

Explanation:
This is the correct answer because it describes how data will be filtered when a query is run with the following filter: longitude < 20 & longitude > -20. The query is run on a Delta Lake table that has the following schema: user_id LONG, post_text STRING, post_id STRING, longitude FLOAT, latitude FLOAT, post_time TIMESTAMP, date DATE. This table is partitioned by the date column. When a query is run on a partitioned Delta Lake table, Delta Lake uses statistics in the Delta Log to identify data files that might include records in the filtered range. The statistics include information such as min and max values for each column in each data file. By using these statistics, Delta Lake can skip reading data files that do not match the filter condition, which can improve query performance and reduce I/O costs. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Data skipping” section.

NEW QUESTION 26

What is a method of installing a Python package scoped at the notebook level to all nodes in the currently active cluster?

A. Use &Pip install in a notebook cell
B. Run source env/bin/activate in a notebook setup script
C. Install libraries from PyPi using the cluster UI
D. Use &sh install in a notebook cell

Answer: C

Explanation:
Installing a Python package scoped at the notebook level to all nodes in the currently active cluster in Databricks can be achieved by using the Libraries tab in the cluster UI. This interface allows you to install libraries across all nodes in the cluster. While the %pip command in a notebook cell would only affect the driver node, using the cluster UI ensures that the package is installed on all nodes.
References:
? Databricks Documentation on Libraries: Libraries

NEW QUESTION 29
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 30
The data science team has created and logged a production model using MLflow. The following code correctly imports and applies the production model to output the predictions as a new DataFrame named preds with the schema "customer_id LONG, predictions DOUBLE, date DATE".

```
from pyspark.sql.functions import current_date
model = mlflow.pyfunc.spark_udf(spark, model_uri="models:/churn/prod")
df = spark.table("customers")
columns = ["account_age", "time_since_last_seen", "app_rating"]
preds = (df.select(
  "customer_id",
  model(*columns).alias("predictions"),
  current_date().alias("date")
  )
```

The data science team would like predictions saved to a Delta Lake table with the ability to compare all predictions across time. Churn predictions will be made at most once per day.
Which code block accomplishes this task while minimizing potential compute costs?
A) preds.write.mode("append").saveAsTable("churn_preds")
B) preds.write.format("delta").save("/preds/churn_preds")
C)

```
(preds.writeStream
  .outputMode("overwrite")
  .option("checkpointPath", "/_checkpoints/churn_preds")
  .start("/preds/churn_preds")
)
```

D)

```
(preds.write
   .format("delta")
   .mode("overwrite")
   .saveAsTable("churn_preds")
)
```

E)

```
(preds.writeStream
   .outputMode("append")
   .option("checkpointPath", "/_checkpoints/churn_preds")
   .table("churn_preds")
 )
```

A. Option A
B. Option B
C. Option C
D. Option D
E. Option E

Answer: A

NEW QUESTION 35
A distributed team of data analysts share computing resources on an interactive cluster with autoscaling configured. In order to better manage costs and query throughput, the workspace administrator is hoping to evaluate whether cluster upscaling is caused by many concurrent users or resource-intensive queries.
In which location can one review the timeline for cluster resizing events?

A. Workspace audit logs
B. Driver's log file
C. Ganglia
D. Cluster Event Log
E. Executor's log file

Answer: C

NEW QUESTION 37
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 39
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 40
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 compactionC Bloom filler indices calculated on the table are preventing file compaction
C. Databricks has autotuned to a smaller target file size based on the overall size of data in the table
D. 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 44
The following table consists of items found in user carts within an e-commerce website.

```
Carts (id LONG, items ARRAY<STRUCT<id: LONG, count: INT>>)
id  items                                          email
1001[{id: "DESK65", count: 1}]                     "u1@domain.com"
1002[{id: "KYBD45", count: 1}, {id: "M27", count: 2}]"u2@domain.com"
1003[{id: "M27", count: 1}]                        "u3@domain.com"
```

The following MERGE statement is used to update this table using an updates view, with schema evaluation enabled on this table.

```
MERGE INTO carts c
USING updates u
ON c.id = u.id
WHEN MATCHED
  THEN UPDATE SET *
```

How would the following update be handled?

```
(new nested field, missing existing column)
id  items
1001[{id: "DESK65", count: 2, coupon: "BOGO50"}]
```

How would the following update be handled?

A. The update is moved to separate ''restored'' column because it is missing a column expected in the target schema.
B. The new restored field is added to the target schema, and dynamically read as NULL for existing unmatched records.
C. The update throws an error because changes to existing columns in the target schema are not supported.
D. The new nested field is added to the target schema, and files underlying existing records are updated to include NULL values for the new field.

Answer: D

Explanation:
With schema evolution enabled in Databricks Delta tables, when a new field is added to a record through a MERGE operation, Databricks automatically modifies the table schema to include the new field. In existing records where this new field is not present, Databricks will insert NULL values for that field. This ensures that the schema remains consistent across all records in the table, with the new field being present in every record, even if it is NULL for records that did not originally include it.
References:
? Databricks documentation on schema evolution in Delta Lake: https://docs.databricks.com/delta/delta-batch.html#schema-evolution

NEW QUESTION 48
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 51
Spill occurs as a result of executing various wide transformations. However, diagnosing spill requires one to proactively look for key indicators.
Where in the Spark UI are two of the primary indicators that a partition is spilling to disk?

A. Stage’s detail screen and Executor’s files
B. Stage’s detail screen and Query’s detail screen
C. Driver’s and Executor’s log files
D. Executor’s detail screen and Executor’s log files

Answer: B

Explanation:
In Apache Spark's UI, indicators of data spilling to disk during the execution of wide transformations can be found in the Stage’s detail screen and the Query’s detail screen. These screens provide detailed metrics about each stage of a Spark job, including information about memory usage and spill data. If a task is spilling data to disk, it indicates that the data being processed exceeds the available memory, causing Spark to spill data to disk to free up memory. This is an important performance metric as excessive spill can significantly slow down the processing.
References:
? Apache Spark Monitoring and Instrumentation: Spark Monitoring Guide
? Spark UI Explained: Spark UI Documentation

NEW QUESTION 55
A data engineer wants to reflector the following DLT code, which includes multiple definition with very similar code:

```
@dlt.table(name=f"t1_dataset")
def t1_dataset():
    return spark.read.table(t1)

@dlt.table(name=f"t2_dataset")
def t2_dataset():
    return spark.read.table(t2)

@dlt.table(name=f"t3_dataset")
def t3_dataset():
    return spark.read.table(t3)

...
```

In an attempt to programmatically create these tables using a parameterized table definition, the data engineer writes the following code.

```
tables = ["t1", "t2", "t3"]

for t in tables:
    @dlt.table(name=f"{t}_dataset")
    def new_table():
```

The pipeline runs an update with this refactored code, but generates a different DAG showing incorrect configuration values for tables.
How can the data engineer fix this?

A. Convert the list of configuration values to a dictionary of table settings, using table names as keys.
B. Convert the list of configuration values to a dictionary of table settings, using different input the for loop.
C. Load the configuration values for these tables from a separate file, located at a path provided by a pipeline parameter.
D. Wrap the loop inside another table definition, using generalized names and properties to replace with those from the inner table

Answer: A

Explanation:
The issue with the refactored code is that it tries to use string interpolation to dynamically create table names within the dlc.table decorator, which will not correctly interpret the table names. Instead, by using a dictionary with table names as keys and their configurations as values, the data engineer can iterate over the dictionary items and use the keys (table names) to properly configure the table settings. This way, the decorator can correctly recognize each table name, and the corresponding configuration settings can be applied appropriately.

NEW QUESTION 59
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 62
The data architect has decided that once data has been ingested from external sources into the
Databricks Lakehouse, table access controls will be leveraged to manage permissions for all production tables and views.
The following logic was executed to grant privileges for interactive queries on a production database to the core engineering group.
GRANT USAGE ON DATABASE prod TO eng; GRANT SELECT ON DATABASE prod TO eng;
Assuming these are the only privileges that have been granted to the eng group and that these users are not workspace administrators, which statement describes their privileges?

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

Answer: D

Explanation:
The GRANT USAGE ON DATABASE prod TO eng command grants the eng group the permission to use the prod database, which means they can list and access the tables and views in the database. The GRANT SELECT ON DATABASE prod TO eng command grants the eng group the permission to select data from the tables and views in the prod database, which means they can query the data using SQL or DataFrame API. However, these commands do not grant the eng group any other permissions, such as creating, modifying, or deleting tables and views, or defining custom functions. Therefore, the eng group members are able to query all tables and views in the prod database, but cannot create or edit anything in the database. References:
? Grant privileges on a database: https://docs.databricks.com/en/security/auth-authz/table-acls/grant-privileges-database.html
? Privileges you can grant on Hive metastore objects: https://docs.databricks.com/en/security/auth-authz/table-acls/privileges.html

NEW QUESTION 65
The data science team has requested assistance in accelerating queries on free form text from user reviews. The data is currently stored in Parquet with the below schema:
item_id INT, user_id INT, review_id INT, rating FLOAT, review STRING
The review column contains the full text of the review left by the user. Specifically, the data science team is looking to identify if any of 30 key words exist in this field.
A junior data engineer suggests converting this data to Delta Lake will improve query performance.
Which response to the junior data engineer s suggestion is correct?

A. Delta Lake statistics are not optimized for free text fields with high cardinality.
B. Text data cannot be stored with Delta Lake.
C. ZORDER ON review will need to be run to see performance gains.
D. The Delta log creates a term matrix for free text fields to support selective filtering.
E. Delta Lake statistics are only collected on the first 4 columns in a table.

Answer: A

Explanation:
Converting the data to Delta Lake may not improve query performance on free text fields with high cardinality, such as the review column. This is because Delta Lake collects statistics on the minimum and maximum values of each column, which are not very useful for filtering or skipping data on free text fields. Moreover, Delta Lake collects statistics on the first 32 columns by default, which may not include the review column if the table has more columns. Therefore, the junior data engineer’s suggestion is not correct. A better approach would be to use a full-text search engine, such as Elasticsearch, to index and query the review column. Alternatively, you can use natural language processing techniques, such as tokenization, stemming, and lemmatization, to preprocess the review column and create a new column with normalized terms that can be used for filtering or skipping data. References:
? Optimizations: https://docs.delta.io/latest/optimizations-oss.html
? Full-text search with Elasticsearch: https://docs.databricks.com/data/data- sources/elasticsearch.html
? Natural language processing: https://docs.databricks.com/applications/nlp/index.html

NEW QUESTION 68
A Delta Lake table representing metadata about content from user has the following schema:
Based on the above schema, which column is a good candidate for partitioning the Delta Table?

A. Date
B. Post_id
C. User_id
D. Post_time

Answer: A

Explanation:
Partitioning a Delta Lake table improves query performance by organizing data into partitions based on the values of a column. In the given schema, the date column is a good candidate for partitioning for several reasons:
? Time-Based Queries: If queries frequently filter or group by date, partitioning by the date column can significantly improve performance by limiting the amount of data scanned.
? Granularity: The date column likely has a granularity that leads to a reasonable number of partitions (not too many and not too few). This balance is important for optimizing both read and write performance.
? Data Skew: Other columns like post_id or user_id might lead to uneven partition sizes (data skew), which can negatively impact performance.
Partitioning by post_time could also be considered, but typically date is preferred due to its more manageable granularity.
References:
? Delta Lake Documentation on Table Partitioning: Optimizing Layout with Partitioning

NEW QUESTION 69
A production cluster has 3 executor nodes and uses the same virtual machine type for the driver and executor.
When evaluating the Ganglia Metrics for this cluster, which indicator would signal a bottleneck caused by code executing on the driver?

A. The five Minute Load Average remains consistent/flat
B. Bytes Received never exceeds 80 million bytes per second
C. Total Disk Space remains constant
D. Network I/O never spikes
E. Overall cluster CPU utilization is around 25%

Answer: E

Explanation:
This is the correct answer because it indicates a bottleneck caused by code executing on the driver. A bottleneck is a situation where the performance or capacity of a system is limited by a single component or resource. A bottleneck can cause slow execution, high latency, or low throughput. A production cluster has 3 executor nodes and uses the same virtual machine type for the driver and executor. When evaluating the Ganglia Metrics for this cluster, one can look for indicators that show how the cluster resources are being utilized, such as CPU, memory, disk, or network. If the overall cluster CPU utilization is around 25%, it means that only one out of the four nodes (driver + 3 executors) is using its full CPU capacity, while the other three nodes are idle or underutilized. This suggests

that the code executing on the driver is taking too long or consuming too much CPU resources, preventing the executors from receiving tasks or data to process. This can happen when the code has driver-side operations that are not parallelized or distributed, such as collecting large amounts of data to the driver, performing complex calculations on the driver, or using non-Spark libraries on the driver. Verified References: [Databricks Certified Data Engineer Professional], under “Spark Core” section; Databricks Documentation, under “View cluster status and event logs - Ganglia metrics” section; Databricks Documentation, under “Avoid collecting large RDDs” section.
In a Spark cluster, the driver node is responsible for managing the execution of the Spark application, including scheduling tasks, managing the execution plan, and interacting with the cluster manager. If the overall cluster CPU utilization is low (e.g., around 25%), it may indicate that the driver node is not utilizing the available resources effectively and might be a bottleneck.

NEW QUESTION 71
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 74
A user wants to use DLT expectations to validate that a derived table report contains all records from the source, included in the table validation_copy.
The user attempts and fails to accomplish this by adding an expectation to the report table definition.
Which approach would allow using DLT expectations to validate all expected records are present in this table?

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

Answer: D

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

NEW QUESTION 77
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 80
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.
Which statement describes this implementation?

A. The customers table is implemented as a Type 3 table; old values are maintained as a new column alongside the current value.
B. 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.
C. The customers table is implemented as a Type 0 table; all writes are append only with no changes to existing values.
D. The customers table is implemented as a Type 1 table; old values are overwritten by new values and no history is maintained.
E. The customers table is implemented as a Type 2 table; old values are overwritten and new customers are appended.

Answer: A

Explanation:
The logic uses the MERGE INTO command to merge new records from the view updates into the table customers. The MERGE INTO command takes two arguments: a target table and a source table or view. The command also specifies a condition to match records between the target and the source, and a set of actions to perform when there is a match or not. In this case, the condition is to match records by customer_id, which is the primary key of the customers table. The actions are to update the existing record in the target with the new values from the source, and set the current_flag to false to indicate that the record is no longer current; and to insert a new record in the target with the new values from the source, and set the current_flag to true to indicate that the record is current. This means that old values are maintained but marked as no longer current and new values are inserted, which is the definition of a Type 2 table. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Merge Into (Delta Lake on Databricks)” section.

NEW QUESTION 85
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 88
The data engineering team has configured a Databricks SQL query and alert to monitor the values in a Delta Lake table. The recent_sensor_recordings table contains an identifying sensor_id alongside the timestamp and temperature for the most recent 5 minutes of recordings.
The below query is used to create the alert:

```
SELECT MEAN(temperature), MAX(temperature), MIN(temperature)
FROM recent_sensor_recordings
GROUP BY sensor_id
```

The query is set to refresh each minute and always completes in less than 10 seconds. The alert is set to trigger when mean (temperature) > 120. Notifications are triggered to be sent at most every 1 minute.
If this alert raises notifications for 3 consecutive minutes and then stops, which statement must be true?

A. The total average temperature across all sensors exceeded 120 on three consecutive executions of the query
B. The recent_sensor_recordingstable was unresponsive for three consecutive runs of the query
C. The source query failed to update properly for three consecutive minutes and then restarted
D. The maximum temperature recording for at least one sensor exceeded 120 on three consecutive executions of the query
E. The average temperature recordings for at least one sensor exceeded 120 on three consecutive executions of the query

Answer: E

Explanation:
This is the correct answer because the query is using a GROUP BY clause on the sensor_id column, which means it will calculate the mean temperature for each sensor separately. The alert will trigger when the mean temperature for any sensor is greater than 120, which means at least one sensor had an average temperature above 120 for three consecutive minutes. The alert will stop when the mean temperature for all sensors drops below 120. Verified References: [Databricks Certified Data Engineer Professional], under “SQL Analytics” section; Databricks Documentation, under “Alerts” section.

NEW QUESTION 89
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 92
Which distribution does Databricks support for installing custom Python code packages?

A. sbt
B. CRAN
C. CRAM
D. nom
E. Wheels
F. jars

Answer: D

NEW QUESTION 93
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"),
                input_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 98
Although the Databricks Utilities Secrets module provides tools to store sensitive credentials and avoid accidentally displaying them in plain text users should still be careful with which credentials are stored here and which users have access to using these secrets.
Which statement describes a limitation of Databricks Secrets?

A. Because the SHA256 hash is used to obfuscate stored secrets, reversing this hash will display the value in plain text.
B. Account administrators can see all secrets in plain text by logging on to the Databricks Accounts console.
C. Secrets are stored in an administrators-only table within the Hive Metastore; database administrators have permission to query this table by default.
D. Iterating through a stored secret and printing each character will display secret contents in plain text.
E. The Databricks REST API can be used to list secrets in plain text if the personal access token has proper credentials.

Answer: E

Explanation:
This is the correct answer because it describes a limitation of Databricks Secrets. Databricks Secrets is a module that provides tools to store sensitive credentials and avoid accidentally displaying them in plain text. Databricks Secrets allows creating secret scopes, which are collections of secrets that can be accessed by users or groups. Databricks Secrets also allows creating and managing secrets using the Databricks CLI or the Databricks REST API. However, a limitation of Databricks Secrets is that the Databricks REST API can be used to list secrets in plain text if the personal access token has proper credentials. Therefore, users should still be careful with which credentials are stored in Databricks Secrets and which users have access to using these secrets. Verified References: [Databricks Certified Data Engineer Professional], under “Databricks Workspace” section; Databricks Documentation, under “List secrets” section.

NEW QUESTION 100
A junior data engineer on your team has implemented the following code block.

```
MERGE INTO events
USING new_events
ON events.event_id = new_events.event_id
WHEN NOT MATCHED
  INSERT *
```

The view new_events contains a batch of records with the same schema as the events Delta table. The event_id field serves as a unique key for this table.
When this query is executed, what will happen with new records that have the same event_id as an existing record?

A. They are merged.
B. They are ignored.
C. They are updated.
D. They are inserted.
E. They are deleted.

Answer: B

Explanation:
This is the correct answer because it describes what will happen with new records that have the same event_id as an existing record when the query is executed. The query uses the INSERT INTO command to append new records from the view new_events to the table events. However, the INSERT INTO command does not check for duplicate values in the primary key column (event_id) and does not perform any update or delete operations on existing records. Therefore, if there are new records that have the same event_id as an existing record, they will be ignored and not inserted into the table events. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Append data using INSERT INTO” section.
"If none of the WHEN MATCHED conditions evaluate to true for a source and target row pair that matches the merge_condition, then the target row is left unchanged." https://docs.databricks.com/en/sql/language-manual/delta-merge-into.html#:~:text=If%20none%20of%20the%20WHEN%20MATCHED%20conditions%20evaluate%20to%20true%20for%20a%20source%20and%20target%20row%20pair%20that%20matches%20the%20merge_condition%2C%20then%20the%20target%20row%20is%20l eft%20unchanged.

NEW QUESTION 103
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 108
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 BACUM command is used to remove invalidated data files.
B. Yes: Delta Lake ACID guarantees provide assurance that the DELETE command successed 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 112
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 116
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 118
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 119
.....

Thank You for Trying Our Product

We offer two products:

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

2nd - Questons 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