Databricks

Exam Questions Databricks-Certified-Professional-Data-Engineer

Databricks Certified Data Engineer Professional Exam

About Exambible

Your Partner of IT Exam

Found in 1998

Exambible is a company specialized on providing high quality IT exam practice study materials, especially Cisco CCNA, CCDA, CCNP, CCIE, Checkpoint CCSE, CompTIA A+, Network+ certification practice exams and so on. We guarantee that the candidates will not only pass any IT exam at the first attempt but also get profound understanding about the certificates they have got. There are so many alike companies in this industry, however, Exambible has its unique advantages that other companies could not achieve.

Our Advances

* 99.9% Uptime

    All examinations will be up to date.

* 24/7 Quality Support

    We will provide service round the clock.

* 100% Pass Rate

    Our guarantee that you will pass the exam.

* Unique Gurantee

    If you do not pass the exam at the first time, we will not only arrange FULL REFUND for you, but also provide you another exam of your claim, ABSOLUTELY FREE!

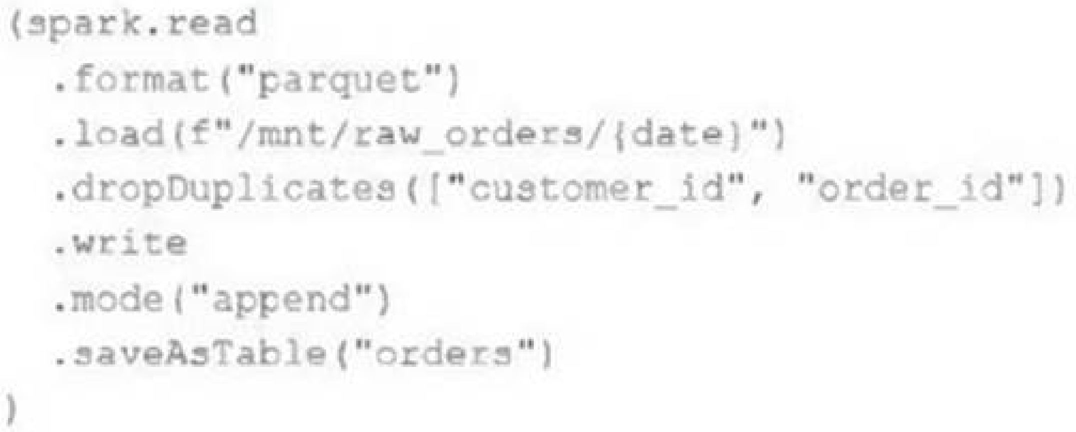
NEW QUESTION 1
Review the following error traceback:
Which statement describes the error being raised?

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
An upstream source writes Parquet data as hourly batches to directories named with the current date. A nightly batch job runs the following code to ingest all data from the previous day as indicated by the date variable:



```
(spark.read
   .format("parquet")
   .load(f"/mnt/raw_orders/{date}")
   .dropDuplicates(["customer_id", "order_id"])
   .write
   .mode("append")
   .saveAsTable("orders")
)
```

Assume that the fields customer_id and order_id serve as a composite key to uniquely identify each order.
If the upstream system is known to occasionally produce duplicate entries for a single order
hours apart, which statement is correct?

A. Each write to the orders table will only contain unique records, and only those records without duplicates in the target table will be written.
B. Each write to the orders table will only contain unique records, but newly written records may have duplicates already present in the target table.
C. Each write to the orders table will only contain unique records; if existing records with the same key are present in the target table, these records will be overwritten.
D. Each write to the orders table will only contain unique records; if existing records with the same key are present in the target table, the operation will tail.
E. Each write to the orders table will run deduplication over the union of new and existing records, ensuring no duplicate records are present.

Answer: B

Explanation:
This is the correct answer because the code uses the dropDuplicates method to remove any duplicate records within each batch of data before writing to the orders table. However, this method does not check for duplicates across different batches or in the target table, so it is possible that newly written records may have duplicates already present in the target table. To avoid this, a better approach would be to use Delta Lake and perform an upsert operation using mergeInto.
Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "DROP DUPLICATES" section.

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 data engineer team is configuring environment for development testing, and production before beginning migration on a new data pipeline. The team requires extensive testing on both the code and data resulting from code execution, and the team want to develop and test against similar production data as possible.
A junior data engineer suggests that production data can be mounted to the development testing environments, allowing pre production code to execute against production data. Because all users have
Admin privileges in the development environment, the junior data engineer has offered to configure permissions and mount this data for the team.
Which statement captures best practices for this situation?

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

Answer: C

Explanation:
The best practice in such scenarios is to ensure that production data is handled securely and with proper access controls. By granting only read access to production data in development and testing environments, it mitigates the risk of unintended data modification. Additionally, maintaining isolated databases for different environments helps to avoid accidental impacts on production data and systems. References:
? Databricks best practices for securing data:
https://docs.databricks.com/security/index.html

NEW QUESTION 5
Which of the following technologies can be used to identify key areas of text when parsing Spark Driver log4j output?

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

Answer: A

Explanation:
Regex, or regular expressions, are a powerful way of matching patterns in text. They can be used to identify key areas of text when parsing Spark Driver log4j output, such as the log level, the timestamp, the thread name, the class name, the method name, and the message. Regex can be applied in various languages and frameworks, such as Scala, Python, Java, Spark SQL, and Databricks notebooks. References:
? https://docs.databricks.com/notebooks/notebooks-use.html#use-regular-expressions
? https://docs.databricks.com/spark/latest/spark-sql/udf-scala.html#using-regular- expressions-in-udfs
? https://docs.databricks.com/spark/latest/sparkr/functions/regexp_extract.html
? https://docs.databricks.com/spark/latest/sparkr/functions/regexp_replace.html

NEW QUESTION 6
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 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.

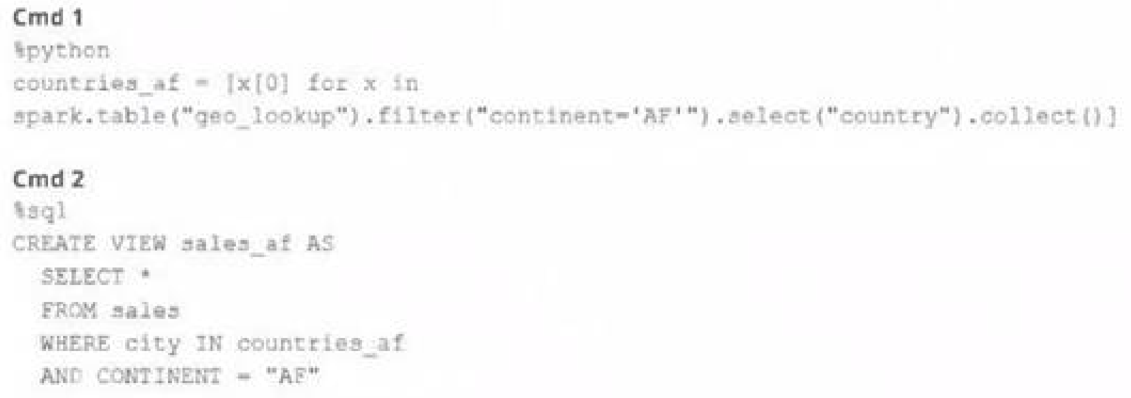
NEW QUESTION 8
The business reporting tem requires that data for their dashboards be updated every hour. The total processing time for the pipeline that extracts transforms and load the data for their pipeline runs in 10 minutes.
Assuming normal operating conditions, which configuration will meet their 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 9
A junior member of the data engineering team is exploring the language interoperability of Databricks notebooks. The intended outcome of the below code is to register a view of all sales that occurred in countries on the continent of Africa that appear in the geo_lookup table.
Before executing the code, running SHOW TABLES on the current database indicates the database contains only two tables: geo_lookup and sales.



Cmd 1
```
%python
countries_af = [x[0] for x in
spark.table("geo_lookup").filter("continent='AF'").select("country").collect()]
```

Cmd 2
```
%sql
CREATE VIEW sales_af AS
  SELECT *
  FROM sales
  WHERE city IN countries_af
  AND CONTINENT = "AF"
```

Which statement correctly describes the outcome of executing these command cells in order in an interactive notebook?

A. Both commands will succee
B. Executing show tables will show that countries at and sales at have been registered as views.
C. Cmd 1 will succee
D. Cmd 2 will search all accessible databases for a table or view named countries af: if this entity exists, Cmd 2 will succeed.
E. Cmd 1 will succeed and Cmd 2 will fail, countries at will be a Python variable representing a PySpark DataFrame.
F. Both commands will fai
G. No new variables, tables, or views will be created.
H. Cmd 1 will succeed and Cmd 2 will fail, countries at will be a Python variable containing a list of strings.

Answer: E

Explanation:

This is the correct answer because Cmd 1 is written in Python and uses a list comprehension to extract the country names from the geo_lookup table and store them in a Python variable named countries af. This variable will contain a list of strings, not a PySpark DataFrame or a SQL view. Cmd 2 is written in SQL and tries to create a view named sales af by selecting from the sales table where city is in countries af. However, this command will fail because countries af is not a valid SQL entity and cannot be used in a SQL query. To fix this, a better approach would be to use spark.sql() to execute a SQL query in Python and pass the countries af variable as a parameter. Verified References: [Databricks Certified Data Engineer Professional], under “Language Interoperability” section; Databricks Documentation, under “Mix languages” section.

NEW QUESTION 10
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 10
A data architect has designed a system in which two Structured Streaming jobs will concurrently write to a single bronze Delta table. Each job is subscribing to a different topic from an Apache Kafka source, but they will write data with the same schema. To keep the directory structure simple, a data engineer has decided to nest a checkpoint directory to be shared by both streams.
The proposed directory structure is displayed below:
Which statement describes whether this checkpoint directory structure is valid for the given scenario and why?

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

Answer: E

Explanation:
This is the correct answer because checkpointing is a critical feature of Structured Streaming that provides fault tolerance and recovery in case of failures. Checkpointing stores the current state and progress of a streaming query in a reliable storage system, such as DBFS or S3. Each streaming query must have its own checkpoint directory that is unique and exclusive to that query. If two streaming queries share the same checkpoint directory, they will interfere with each other and cause unexpected errors or data loss. Verified References: [Databricks Certified Data Engineer Professional], under
“Structured Streaming” section; Databricks Documentation, under “Checkpointing” section.

NEW QUESTION 11
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 13

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

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

Answer: D

Explanation:

The configuration that automatically recovers from query failures and keeps costs low is to use a new job cluster, set retries to unlimited, and set maximum concurrent runs to 1. This configuration has the following advantages:
? A new job cluster is a cluster that is created and terminated for each job run. This means that the cluster resources are only used when the job is running, and no idle costs are incurred. This also ensures that the cluster is always in a clean state and has the latest configuration and libraries for the job1.
? Setting retries to unlimited means that the job will automatically restart the query in case of any failure, such as network issues, node failures, or transient errors. This improves the reliability and availability of the streaming job, and avoids data loss or inconsistency2.
? Setting maximum concurrent runs to 1 means that only one instance of the job can run at a time. This prevents multiple queries from competing for the same resources or writing to the same output location, which can cause performance degradation or data corruption3.
Therefore, this configuration is the best practice for scheduling Structured Streaming jobs for production, as it ensures that the job is resilient, efficient, and consistent.
References: Job clusters, Job retries, Maximum concurrent runs

NEW QUESTION 14

The data engineering team maintains a table of aggregate statistics through batch nightly updates. This includes total sales for the previous day alongside totals and averages for a variety of time periods including the 7 previous days, year-to-date, and quarter-to-date. This table is named store_saies_summary and the schema is as follows:
The table daily_store_sales contains all the information needed to update store_sales_summary. The schema for this table is: store_id INT, sales_date DATE, total_sales FLOAT If daily_store_sales is implemented as a Type 1 table and the total_sales column might be adjusted after manual data auditing, which approach is the safest to generate accurate reports in the store_sales_summary table?

A. Implement the appropriate aggregate logic as a batch read against the daily_store_salestable and overwrite the store_sales_summary table with each Update.
B. Implement the appropriate aggregate logic as a batch read against the daily_store_sales table and append new rows nightly to the store_sales_summary table.
C. Implement the appropriate aggregate logic as a batch read against the daily_store_sales table and use upsert logic to update results in the store_sales_summary table.
D. Implement the appropriate aggregate logic as a Structured Streaming read against the daily_store_sales table and use upsert logic to update results in the store_sales_summary table.
E. Use Structured Streaming to subscribe to the change data feed for daily_store_sales and apply changes to the aggregates in the store_sales_summary table with each update.

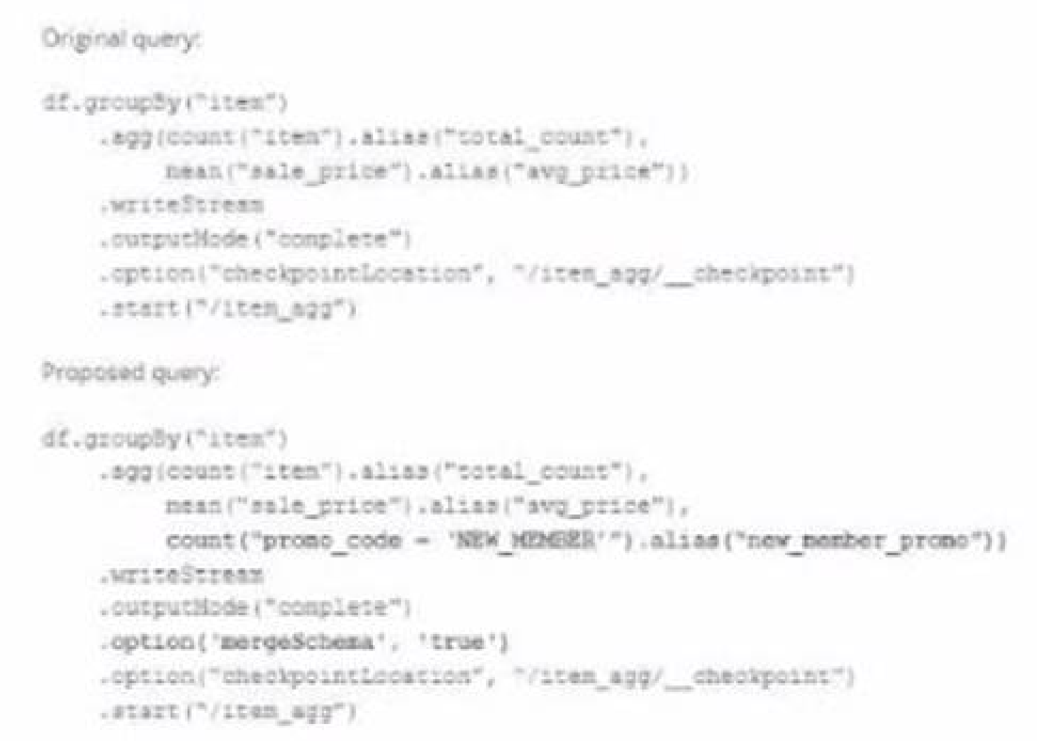
Answer: E

Explanation:

The daily_store_sales table contains all the information needed to update store_sales_summary. The schema of the table is:
store_id INT, sales_date DATE, total_sales FLOAT
The daily_store_sales table is implemented as a Type 1 table, which means that old values are overwritten by new values and no history is maintained. The total_sales column might be adjusted after manual data auditing, which means that the data in the table may change over time.
The safest approach to generate accurate reports in the store_sales_summary table is to use Structured Streaming to subscribe to the change data feed for daily_store_sales and apply changes to the aggregates in the store_sales_summary table with each update. Structured Streaming is a scalable and fault-tolerant stream processing engine built on Spark SQL. Structured Streaming allows processing data streams as if they were tables or DataFrames, using familiar operations such as select, filter, groupBy, or join. Structured Streaming also supports output modes that specify how to write the results of a streaming query to a sink, such as append, update, or complete. Structured Streaming can handle both streaming and batch data sources in a unified manner.
The change data feed is a feature of Delta Lake that provides structured streaming sources that can subscribe to changes made to a Delta Lake table. The change data feed captures both data changes and schema changes as ordered events that can be processed by downstream applications or services. The change data feed can be configured with different options, such as starting from a specific version or timestamp, filtering by operation type or partition values, or excluding no-op changes.
By using Structured Streaming to subscribe to the change data feed for daily_store_sales, one can capture and process any changes made to the total_sales column due to manual data auditing. By applying these changes to the aggregates in the store_sales_summary table with each update, one can ensure that the reports are always consistent and accurate with the latest data. Verified References: [Databricks Certified Data Engineer Professional], under “Spark Core” section; Databricks Documentation, under “Structured Streaming” section; Databricks Documentation, under “Delta Change Data Feed” section.

NEW QUESTION 18

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

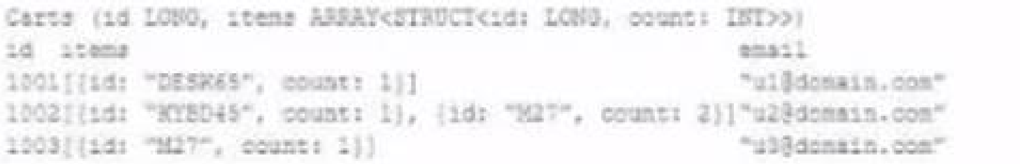
NEW QUESTION 21
A Data engineer wants to run unit's tests using common Python testing frameworks on python functions defined across several Databricks notebooks currently used in production.
How can the data engineer run unit tests against function that work with data in production?

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

Answer: A

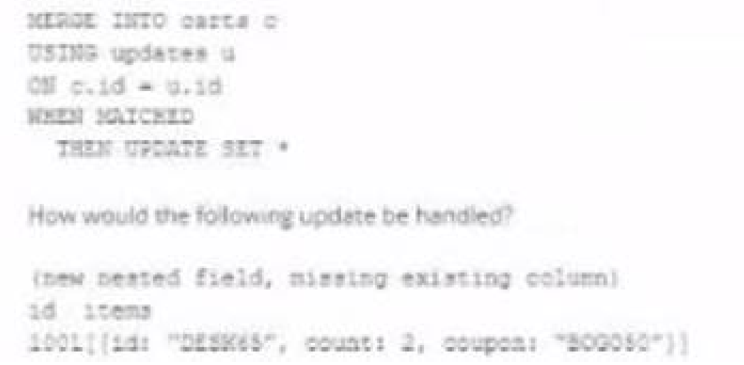
Explanation:
The best practice for running unit tests on functions that interact with data is to use a dataset that closely mirrors the production data. This approach allows data engineers to validate the logic of their functions without the risk of affecting the actual production data. It's important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

NEW QUESTION 22
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"}]
```

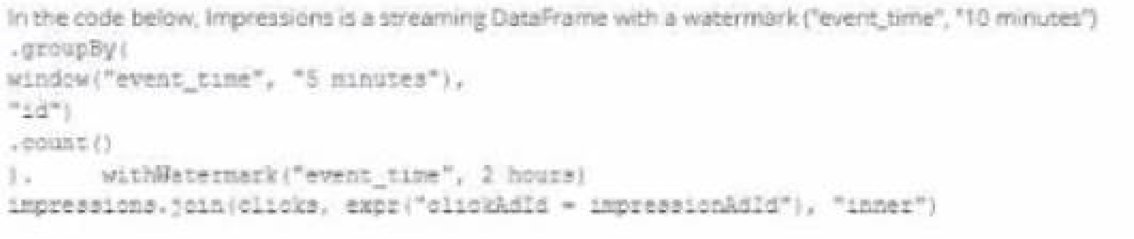
How would the following update be handled?

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 24
A data engineer wants to join a stream of advertisement impressions (when an ad was shown) with another stream of user clicks on advertisements to correlate when impression led to monitizable clicks.



In the code below, Impressions is a streaming DataFrame with a watermark ("event_time", "10 minutes")

```
.groupBy(
window("event_time", "5 minutes"),
"id")
.count()
).     withWatermark("event_time", 2 hours)
impressions.join(clicks, expr("clickAdId = impressionAdId"), "inner")
```

Which solution would improve the performance?
A)

```
Joining on event time constraint: clickTime == impressionTime using a leftOuter join
```

B)

```
Joining on event time constraint: clickTime >= impressionTime - interval 3 hours and removing
watermarks
```

C)

```
Joining on event time constraint: clickTime + 3 hours < impressionTime - 2 hours
```

D)

```
Joining on event time constraint: clickTime >= impressionTime AND clickTime <= impressionTime +
interval 1 hour
```

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

Answer: A

Explanation:
When joining a stream of advertisement impressions with a stream of user clicks, you want to minimize the state that you need to maintain for the join. Option A suggests using a left outer join with the condition that clickTime == impressionTime, which is suitable for correlating events that occur at the exact same time. However, in a real-world scenario, you would likely need some leeway to account for the delay between an impression and a possible click. It's important to design the join condition and the window of time considered to optimize performance while still capturing the relevant user interactions. In this case, having the watermark can help with state management and avoid state growing unbounded by discarding old state data that's unlikely to match with new data.

NEW QUESTION 29
A data engineer, User A, has promoted a new pipeline to production by using the REST API to programmatically create several jobs. A DevOps engineer, User B, has configured an external orchestration tool to trigger job runs through the REST API. Both users authorized the REST API calls using their personal access tokens.
Which statement describes the contents of the workspace audit logs concerning these events?

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

Answer: C

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

NEW QUESTION 30
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 31
The marketing team is looking to share data in an aggregate table with the sales organization, but the field names used by the teams do not match, and a number of marketing specific fields have not been approval for the sales org.
Which of the following solutions addresses the situation while emphasizing simplicity?

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

Answer: A

Explanation:
Creating a view is a straightforward solution that can address the need for field name standardization and selective field sharing between departments. A view allows for presenting a transformed version of the underlying data without duplicating it. In this scenario, the view would only include the approved fields for the sales team and rename any fields as per their naming conventions.
References:
? Databricks documentation on using SQL views in Delta Lake: https://docs.databricks.com/delta/quick-start.html#sql-views

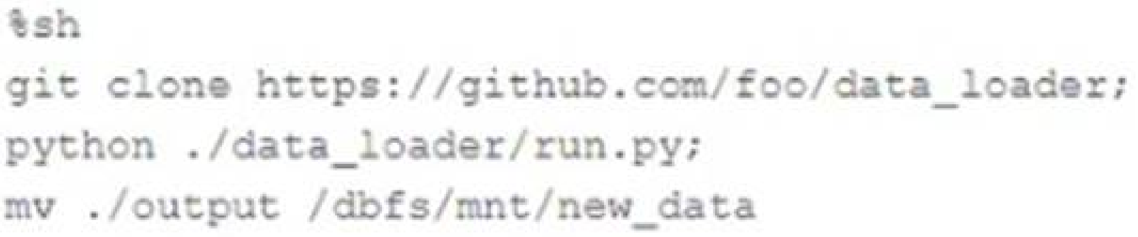
NEW QUESTION 34
The DevOps team has configured a production workload as a collection of notebooks scheduled to run daily using the Jobs Ul. A new data engineering hire is onboarding to the team and has requested access to one of these notebooks to review the production logic.
What are the maximum notebook permissions that can be granted to the user without allowing accidental changes to production code or data?

A. Can manage
B. Can edit
C. Can run
D. Can Read

Answer: D

Explanation:
Granting a user 'Can Read' permissions on a notebook within Databricks allows them to view the notebook's content without the ability to execute or edit it. This level of permission ensures that the new team member can review the production logic for learning or auditing purposes without the risk of altering the notebook's code or affecting production data and workflows. This approach aligns with best practices for maintaining security and integrity in production environments, where strict access controls are essential to prevent unintended modifications.References: Databricks documentation on access control and permissions for notebooks within the workspace (https://docs.databricks.com/security/access-control/workspace-acl.html).

NEW QUESTION 38
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 40
A data pipeline uses Structured Streaming to ingest data from kafka to Delta Lake. Data is being stored in a bronze table, and includes the Kafka_generated timesamp, key, and value. Three months after the pipeline is deployed the data engineering team has noticed some latency issued during certain times of the day.
A senior data engineer updates the Delta Table's schema and ingestion logic to include the current timestamp (as recoded by Apache Spark) as well the Kafka topic and partition. The team plans to use the additional metadata fields to diagnose the transient processing delays:
Which limitation will the team face while diagnosing this problem?

A. New fields not be computed for historic records.
B. Updating the table schema will invalidate the Delta transaction log metadata.
C. Updating the table schema requires a default value provided for each file added.
D. Spark cannot capture the topic partition fields from the kafka source.

Answer: A

Explanation:
When adding new fields to a Delta table's schema, these fields will not be retrospectively applied to historical records that were ingested before the schema change. Consequently, while the team can use the new metadata fields to investigate transient processing delays moving forward, they will be unable to apply this diagnostic approach to past data that lacks these fields.
References:
? Databricks documentation on Delta Lake schema management: https://docs.databricks.com/delta/delta-batch.html#schema-management

NEW QUESTION 44
A table in the Lakehouse named customer_churn_params 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.
The churn prediction model used by the ML team is fairly stable in production. The team is only interested in making predictions on records that have changed in the past 24 hours.
Which approach would simplify the identification of these changed records?

A. Apply the churn model to all rows in the customer_churn_params table, but implement logic to perform an upsert into the predictions table that ignores rows where predictions have not changed.
B. Convert the batch job to a Structured Streaming job using the complete output mode; configure a Structured Streaming job to read from the customer_churn_params table and incrementally predict against the churn model.
C. Calculate the difference between the previous model predictions and the current customer_churn_params on a key identifying unique customers before making new predictions; only make predictions on those customers not in the previous predictions.
D. Modify the overwrite logic to include a field populated by calling spark.sql.functions.current_timestamp() as data are being written; use this field to identify records written on a particular date.
E. Replace the current overwrite logic with a merge statement to modify only those records that have changed; write logic to make predictions on the changed records identified by the change data feed.

Answer: E

Explanation:
The approach that would simplify the identification of the changed records is to replace the current overwrite logic with a merge statement to modify only those records that have changed, and write logic to make predictions on the changed records identified by the change data feed. This approach leverages the Delta Lake features of merge and change data feed, which are designed to handle upserts and track row-level changes in a Delta table12. By using merge, the data engineering team can avoid overwriting the entire table every night, and only update or insert the records that have changed in the source data. By using change data feed, the ML team can easily access the change events that have occurred in the customer_churn_params table, and filter them by operation type (update or insert) and timestamp. This way, they can only make predictions on the records that have changed in the past 24 hours, and avoid re-processing the unchanged records. The other options are not as simple or efficient as the proposed approach, because:
? Option A would require applying the churn model to all rows in the customer_churn_params table, which would be wasteful and redundant. It would also require implementing logic to perform an upsert into the predictions table, which would be more complex than using the merge statement.
? Option B would require converting the batch job to a Structured Streaming job, which would involve changing the data ingestion and processing logic. It would also require using the complete output mode, which would output the entire result table every time there is a change in the source data, which would be inefficient and costly.
? Option C would require calculating the difference between the previous model predictions and the current customer_churn_params on a key identifying unique customers, which would be computationally expensive and prone to errors. It would also require storing and accessing the previous predictions, which would add extra storage and I/O costs.
? Option D would require modifying the overwrite logic to include a field populated by calling spark.sql.functions.current_timestamp() as data are being written, which would add extra complexity and overhead to the data engineering job. It would also require using this field to identify records written on a particular date, which would be less accurate and reliable than using the change data feed.
References: Merge, Change data feed

NEW QUESTION 45
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 48
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

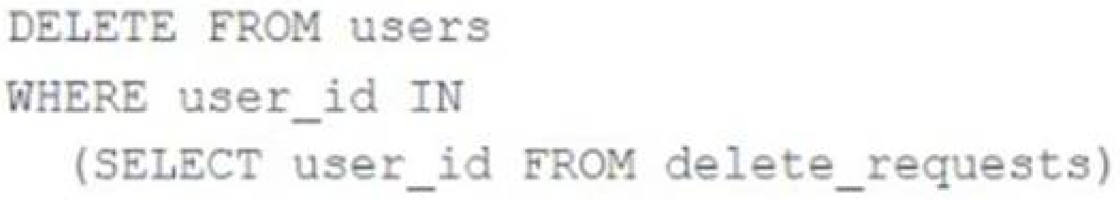
NEW QUESTION 52
A production cluster has 3 executor nodes and uses the same virtual machine type for the driver and executor.
When evaluating the Ganglia Metrics for this cluster, which indicator would signal a bottleneck caused by code executing on the driver?

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 54
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 58
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

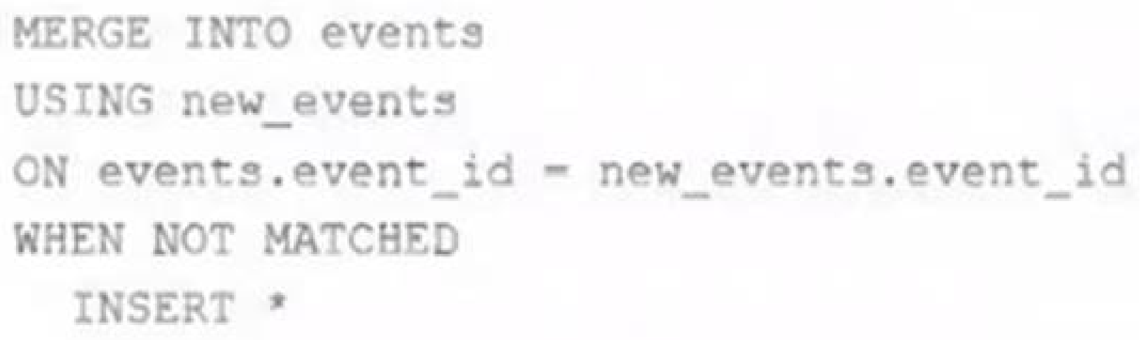
Explanation:
https://learn.microsoft.com/en-us/azure/databricks/delta/vacuum

NEW QUESTION 60
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 64
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.

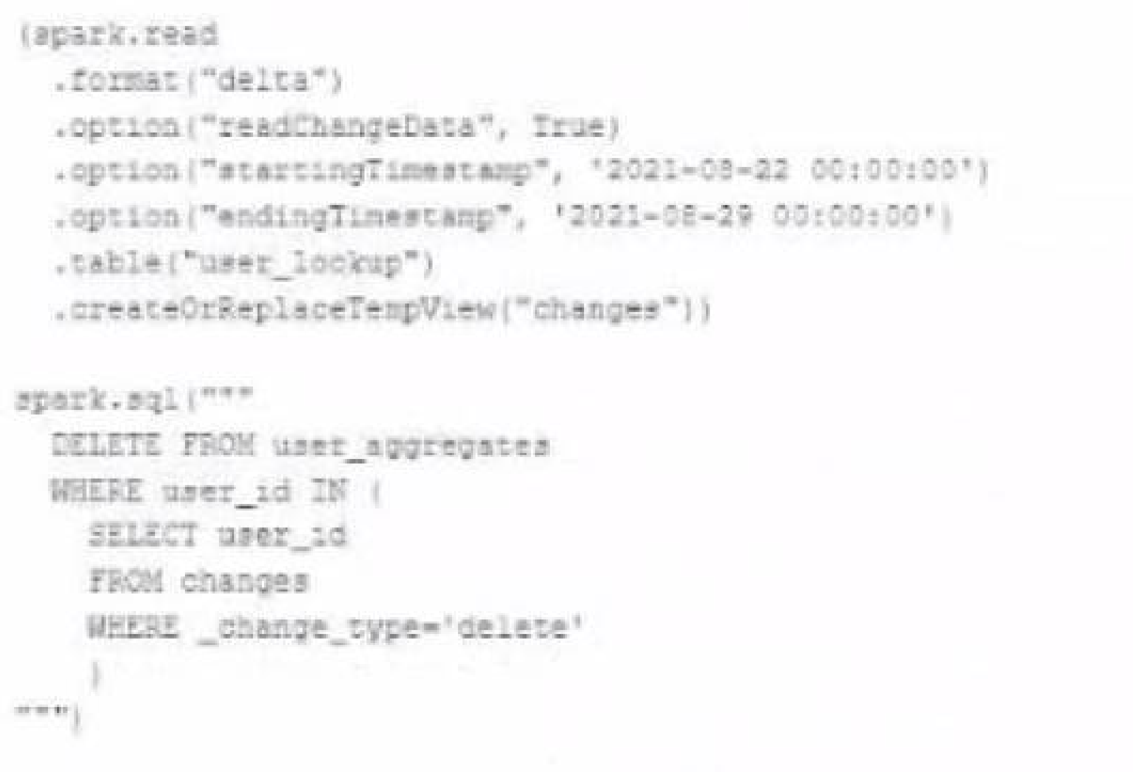
NEW QUESTION 68
Which is a key benefit of an end-to-end test?

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 71
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 74
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 79
A small company based in the United States has recently contracted a consulting firm in India to implement several new data engineering pipelines to power artificial intelligence applications. All the company's data is stored in regional cloud storage in the United States.
The workspace administrator at the company is uncertain about where the Databricks workspace used by the contractors should be deployed.
Assuming that all data governance considerations are accounted for, which statement accurately informs this decision?

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

Answer: C

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

NEW QUESTION 83
An external object storage container has been mounted to the location /mnt/finance_eda_bucket.
The following logic was executed to create a database for the finance team:
After the database was successfully created and permissions configured, a member of the finance team runs the following code:
If all users on the finance team are members of the finance group, which statement describes how the tx_sales table will be created?

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

Answer: A

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

NEW QUESTION 86
......

Relate Links

Contact us