Exam Questions Databricks-Certified-Professional-Data-Engineer

Databricks Certified Data Engineer Professional Exam

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

NEW QUESTION 1
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 2
A data ingestion task requires a one-TB JSON dataset to be written out to Parquet with a target part-file size of 512 MB. Because Parquet is being used instead of Delta Lake, built- in file-sizing features such as Auto-Optimize & Auto-Compaction cannot be used.
Which strategy will yield the best performance without shuffling data?

A. Set spark.sql.files.maxPartitionBytes to 512 MB, ingest the data, execute the narrow transformations, and then write to parquet.
B. Set spark.sql.shuffle.partitions to 2,048 partitions (1TB*1024*1024/512), ingest the data, execute the narrow transformations, optimize the data by sorting it (which automatically repartitions the data), and then write to parquet.
C. Set spark.sql.adaptive.advisoryPartitionSizeInBytes to 512 MB bytes, ingest the data, execute the narrow transformations, coalesce to 2,048 partitions (1TB*1024*1024/512), and then write to parquet.
D. Ingest the data, execute the narrow transformations, repartition to 2,048 partitions (1TB* 1024*1024/512), and then write to parquet.
E. Set spark.sql.shuffle.partitions to 512, ingest the data, execute the narrow transformations, and then write to parquet.

Answer: B

Explanation:
The key to efficiently converting a large JSON dataset to Parquet files of a specific size without shuffling data lies in controlling the size of the output files directly.
? Setting spark.sql.files.maxPartitionBytes to 512 MB configures Spark to process
data in chunks of 512 MB. This setting directly influences the size of the part-files in the output, aligning with the target file size.
? Narrow transformations (which do not involve shuffling data across partitions) can
then be applied to this data.
? Writing the data out to Parquet will result in files that are approximately the size specified by spark.sql.files.maxPartitionBytes, in this case, 512 MB.
? The other options involve unnecessary shuffles or repartitions (B, C, D) or an incorrect setting for this specific requirement (E).
References:
? Apache Spark Documentation: Configuration - spark.sql.files.maxPartitionBytes
? Databricks Documentation on Data Sources: Databricks Data Sources Guide

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
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 to power several production monitoring dashboards and a production model. At present, 45 of the 100 fields are being used in at least one of these applications.
The data engineer is trying to determine the best approach for dealing with schema declaration given the highly-nested structure of the data and the numerous fields.
Which of the following accurately presents information about Delta Lake and Databricks that may impact their decision-making process?

A. The 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.
B. Because Delta Lake uses Parquet for data storage, data types can be easily evolved by just modifying file footer information in place.
C. Human labor in writing code is the largest cost associated with data engineering workloads; as such, automating table declaration logic should be a priority in all migration workloads.
D. Because Databricks will infer schema using types that allow all observed data to be processed, setting types manually provides greater assurance of data quality enforcement.
E. Schema inference and evolution on .Databricks ensure that inferred types will always accurately match the data types used by downstream systems.

Answer: D

Explanation:
This is the correct answer because it accurately presents information about Delta Lake and Databricks that may impact the decision-making process of a junior data engineer who is trying to determine the best approach for dealing with schema declaration given the highly-nested structure of the data and the numerous fields. Delta Lake and Databricks support schema inference and evolution, which means that they can automatically infer the schema of a table from the source data and allow adding new columns or changing column types without affecting existing queries or pipelines. However, schema inference and evolution may not always be desirable or reliable, especially when dealing with complex or nested data structures or when enforcing data quality and consistency across different systems. Therefore, setting types manually can provide greater assurance of data quality enforcement and avoid potential errors or conflicts due to incompatible or unexpected data types. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Schema inference and partition of streaming DataFrames/Datasets” section.

NEW QUESTION 5
A data engineer needs to capture pipeline settings from an existing in the workspace, and use them to create and version a JSON file to create a new pipeline.
Which command should the data engineer enter in a web terminal configured with the Databricks CLI?

A. Use the get command to capture the settings for the existing pipeline; remove the pipeline_id and rename the pipeline; use this in a create command
B. Stop the existing pipeline; use the returned settings in a reset command
C. Use the alone command to create a copy of an existing pipeline; use the get JSON command to get the pipeline definition; save this to git
D. Use list pipelines to get the specs for all pipelines; get the pipeline spec from the return results parse and use this to create a pipeline

Answer: A

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

NEW QUESTION 6
The data architect has mandated that all tables in the Lakehouse should be configured as external Delta Lake tables.
Which approach will ensure that this requirement is met?

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

Answer: C

Explanation:
This is the correct answer because it ensures that this requirement is met. The requirement is that all tables in the Lakehouse should be configured as external Delta Lake tables. An external table is a table that is stored outside of the default warehouse directory and whose metadata is not managed by Databricks. An external table can be created by using the location keyword to specify the path to an existing directory in a cloud storage system, such as DBFS or S3. By creating external tables, the data engineering team can avoid losing data if they drop or overwrite the table, as well as leverage existing data without moving or copying it. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Create an external table” section.

NEW QUESTION 7
Which 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 8
A junior data engineer has configured a workload that posts the following JSON to the Databricks REST API endpoint 2.0/jobs/create.

```
{
  "name": "Ingest new data",
  "existing_cluster_id": "6015-954420-peace720",
  "notebook_task": {
    "notebook_path": "/Prod/ingest.py"
  }
}
```

Assuming that all configurations and referenced resources are available, which statement describes the result of executing this workload three times?

A. Three new jobs named "Ingest new data" will be defined in the workspace, and they will each run once daily.
B. The logic defined in the referenced notebook will be executed three times on new clusters with the configurations of the provided cluster ID.
C. Three new jobs named "Ingest new data" will be defined in the workspace, but no jobs will be executed.
D. One new job named "Ingest new data" will be defined in the workspace, but it will not be executed.
E. The logic defined in the referenced notebook will be executed three times on the referenced existing all purpose cluster.

Answer: E

Explanation:
This is the correct answer because the JSON posted to the Databricks REST API endpoint 2.0/jobs/create defines a new job with a name, an existing cluster id, and a notebook task. However, it does not specify any schedule or trigger for the job execution. Therefore, three new jobs with the same name and configuration will be created in the workspace, but none of them will be executed until they are manually triggered or scheduled. Verified References: [Databricks Certified Data Engineer Professional], under “Monitoring & Logging” section; [Databricks Documentation], under “Jobs API - Create” section.

NEW QUESTION 9
Which statement describes the correct use of pyspark.sql.functions.broadcast?

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

Answer: D

Explanation:
https://spark.apache.org/docs/3.1.3/api/python/reference/api/pyspark.sql.functions.broadca st.html
The broadcast function in PySpark is used in the context of joins. When you mark a DataFrame with broadcast, Spark tries to send this DataFrame to all worker nodes so that it can be joined with another DataFrame without shuffling the larger DataFrame across the nodes. This is particularly beneficial when the DataFrame is small enough to fit into the memory of each node. It helps to optimize the join process by reducing the amount of data that needs to be shuffled across the cluster, which can be a very expensive operation in terms of computation and time.
The pyspark.sql.functions.broadcast function in PySpark is used to hint to Spark that a DataFrame is small enough to be broadcast to all worker nodes in the cluster. When this hint is applied, Spark can perform a broadcast join, where the smaller DataFrame is sent to each executor only once and joined with the larger DataFrame on each executor. This can significantly reduce the amount of data shuffled across the network and can improve the performance of the join operation.
In a broadcast join, the entire smaller DataFrame is sent to each executor, not just a specific column or a cached version on attached storage. This function is particularly useful when one of the DataFrames in a join operation is much smaller than the other, and can fit comfortably in the memory of each executor node.

References:
? Databricks Documentation on Broadcast Joins: Databricks Broadcast Join Guide
? PySpark API Reference: pyspark.sql.functions.broadcast

NEW QUESTION 10
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 10
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 15
The Databricks CLI is use to trigger a run of an existing job by passing the job_id parameter. The response that the job run request has been submitted successfully includes a filed run_id.
Which statement describes what the number alongside this field represents?

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

Answer: D

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

NEW QUESTION 20
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 21
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 24
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 27
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 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 36
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 38
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 42
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 47
A junior developer complains that the code in their notebook isn't producing the correct results in the development environment. A shared screenshot reveals that while they're using a notebook versioned with Databricks Repos, they're using a personal branch that contains old logic. The desired branch named dev-2.3.9 is not available from the branch selection dropdown.
Which approach will allow this developer to review the current logic for this notebook?

A. Use Repos to make a pull request use the Databricks REST API to update the current branch to dev-2.3.9
B. Use Repos to pull changes from the remote Git repository and select the dev-2.3.9 branch.
C. Use Repos to checkout the dev-2.3.9 branch and auto-resolve conflicts with the current branch
D. Merge all changes back to the main branch in the remote Git repository and clone the repo again
E. Use Repos to merge the current branch and the dev-2.3.9 branch, then make a pull request to sync with the remote repository

Answer: B

Explanation:

This is the correct answer because it will allow the developer to update their local repository with the latest changes from the remote repository and switch to the desired branch. Pulling changes will not affect the current branch or create any conflicts, as it will only fetch the changes and not merge them. Selecting the dev-2.3.9 branch from the dropdown will checkout that branch and display its contents in the notebook. Verified References: [Databricks Certified Data Engineer Professional], under “Databricks Tooling” section; Databricks Documentation, under “Pull changes from a remote repository” section.

NEW QUESTION 49
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 50
A Spark job is taking longer than expected. Using the Spark UI, a data engineer notes that the Min, Median, and Max Durations for tasks in a particular stage show the minimum and median time to complete a task as roughly the same, but the max duration for a task to be roughly 100 times as long as the minimum.
Which situation is causing increased duration of the overall job?

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

Answer: D

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

NEW QUESTION 55
The DevOps team has configured a production workload as a collection of notebooks scheduled to run daily using the Jobs UI. 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 58
What is the first of a Databricks Python notebook when viewed in a text editor?

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

Answer: B

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

NEW QUESTION 63
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 66
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 70
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 75
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 76
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 78
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 83
A table is registered with the following code:
Both users and orders are Delta Lake tables. Which statement describes the results of querying recent_orders?

A. All logic will execute at query time and return the result of joining the valid versions of the source tables at the time the query finishes.
B. All logic will execute when the table is defined and store the result of joining tables to the DBFS; this stored data will be returned when the table is queried.
C. Results will be computed and cached when the table is defined; these cached results will incrementally update as new records are inserted into source tables.
D. All logic will execute at query time and return the result of joining the valid versions of the source tables at the time the query began.
E. The versions of each source table will be stored in the table transaction log; query results will be saved to DBFS with each query.

Answer: B

NEW QUESTION 87
The DevOps team has configured a production workload as a collection of notebooks scheduled to run daily using the Jobs UI. 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. No permissions
D. Can Read
E. Can Run

Answer: C

Explanation:
This is the correct answer because it is the maximum notebook permissions that can be granted to the user without allowing accidental changes to production code or data. Notebook permissions are used to control access to notebooks in Databricks workspaces. There are four types of notebook permissions: Can Manage, Can Edit, Can Run, and Can Read. Can Manage allows full control over the notebook, including editing, running, deleting, exporting, and changing permissions. Can Edit allows modifying and running the notebook, but not changing permissions or deleting it. Can Run allows executing commands in an existing cluster attached to the notebook, but not modifying or exporting it. Can Read allows viewing the notebook content, but not running or modifying it. In this case, granting Can Read permission to the user will allow them to review the
production logic in the notebook without allowing them to make any changes to it or run any commands that may affect production data. Verified References: [Databricks Certified Data Engineer Professional], under “Databricks Workspace” section; Databricks Documentation, under “Notebook permissions” section.

NEW QUESTION 88
The data governance team is reviewing code used for deleting records for compliance with GDPR. They note the following logic is used to delete records from the Delta Lake table named users.

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

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

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

Answer: E

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

NEW QUESTION 93
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 95
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 97
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 102
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 106
A Databricks SQL dashboard has been configured to monitor the total number of records present in a collection of Delta Lake tables using the following query pattern:
SELECT COUNT (*) FROM table -
Which of the following describes how results are generated each time the dashboard is updated?

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

Answer: C

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

NEW QUESTION 108
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 113
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 117
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 119
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 123
A new data engineer notices that a critical field was omitted from an application that writes its Kafka source to Delta Lake. This happened even though the critical field was in the Kafka source. That field was further missing from data written to dependent, long-term storage. The retention threshold on the Kafka service is seven days. The pipeline has been in production for three months.
Which describes how Delta Lake can help to avoid data loss of this nature in the future?

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

Answer: E

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

NEW QUESTION 127
Each configuration below is identical to the extent that each cluster has 400 GB total of RAM, 160 total cores and only one Executor per VM.
Given a job with at least one wide transformation, which of the following cluster configurations will result in maximum performance?

A. • Total VMs; 1• 400 GB per Executor• 160 Cores / Executor
B. • Total VMs: 8• 50 GB per Executor• 20 Cores / Executor
C. • Total VMs: 4• 100 GB per Executor• 40 Cores/Executor
D. • Total VMs:2• 200 GB per Executor• 80 Cores / Executor

Answer: B

Explanation:
This is the correct answer because it is the cluster configuration that will result in maximum performance for a job with at least one wide transformation. A wide transformation is a type of transformation that requires shuffling data across partitions, such as join, groupBy, or orderBy. Shuffling can be expensive and time-consuming, especially if there are too many or too few partitions. Therefore, it is important to choose a cluster configuration that can balance the trade-off between parallelism and network overhead. In this case, having 8 VMs with 50 GB per executor and 20 cores per executor will create 8 partitions, each with enough memory and CPU resources to handle the shuffling efficiently. Having fewer VMs with more memory and cores per executor will create fewer partitions, which will reduce parallelism and increase the size of each shuffle block. Having more VMs with less memory and cores per executor will create more partitions, which will increase parallelism but also increase the network overhead and the number of shuffle files. Verified References: [Databricks Certified Data Engineer Professional], under “Performance Tuning” section; Databricks Documentation, under “Cluster configurations” section.

NEW QUESTION 129
All records from an Apache Kafka producer are being ingested into a single Delta Lake table with the following schema:
key BINARY, value BINARY, topic STRING, partition LONG, offset LONG, timestamp LONG
There are 5 unique topics being ingested. Only the "registration" topic contains Personal Identifiable Information (PII). The company wishes to restrict access to PII. The company also wishes to only retain records containing PII in this table for 14 days after initial ingestion. However, for non-PII information, it would like to retain these records indefinitely.
Which of the following solutions meets the requirements?

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

Answer: B

Explanation:
Partitioning the data by the topic field allows the company to apply different access control policies and retention policies for different topics. For example, the company can use the Table Access Control feature to grant or revoke permissions to the registration topic based on user roles or groups. The company can also use the DELETE command to remove records from the registration topic that are older than 14 days, while keeping the records from other topics indefinitely. Partitioning by the topic field also improves the performance of queries that filter by the topic field, as they can skip reading irrelevant partitions. References:
? Table Access Control: https://docs.databricks.com/security/access-control/table-acls/index.html
? DELETE: https://docs.databricks.com/delta/delta-update.html#delete-from-a-table

NEW QUESTION 133
.....

THANKS FOR TRYING THE DEMO OF OUR PRODUCT

Visit Our Site to Purchase the Full Set of Actual Databricks-Certified-Professional-Data-Engineer Exam Questions With Answers.

We Also Provide Practice Exam Software That Simulates Real Exam Environment And Has Many Self-Assessment Features. Order the Databricks-Certified-Professional-Data-Engineer Product From:

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

Money Back Guarantee

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