Databricks

Exam Questions Databricks-Certified-Professional-Data-Engineer

Databricks Certified Data Engineer Professional Exam

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
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 4
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 5
A data engineer is configuring a pipeline that will potentially see late-arriving, duplicate records.
In addition to de-duplicating records within the batch, which of the following approaches allows the data engineer to deduplicate data against previously processed records as it is inserted into a Delta table?

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

Answer: C

Explanation:
To deduplicate data against previously processed records as it is inserted into a Delta table, you can use the merge operation with an insert-only clause. This allows you to insert new records that do not match any existing records based on a unique key, while ignoring duplicate records that match existing records. For example, you can use the following syntax:
MERGE INTO target_table USING source_table ON target_table.unique_key = source_table.unique_key WHEN NOT MATCHED THEN INSERT *
This will insert only the records from the source table that have a unique key that is not present in the target table, and skip the records that have a matching key. This way, you can avoid inserting duplicate records into the Delta table.
References:
? https://docs.databricks.com/delta/delta-update.html#upsert-into-a-table-using- merge
? https://docs.databricks.com/delta/delta-update.html#insert-only-merge

NEW QUESTION 6
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 7
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 8
In order to prevent accidental commits to production data, a senior data engineer has instituted a policy that all development work will reference clones of Delta Lake tables. After testing both deep and shallow clone, development tables are created using shallow clone.
A few weeks after initial table creation, the cloned versions of several tables implemented as Type 1 Slowly Changing Dimension (SCD) stop working. The transaction logs for the source tables show that vacuum was run the day before.
Why are the cloned tables no longer working?

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

Answer: C

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

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

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

Answer: D

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

NEW QUESTION 10
A junior data engineer has manually configured a series of jobs using the Databricks Jobs UI. Upon reviewing their work, the engineer realizes that they are listed as the "Owner" for each job. They attempt to transfer "Owner" privileges to the "DevOps" group, but cannot successfully accomplish this task.
Which statement explains what is preventing this privilege transfer?

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

Answer: E

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

NEW QUESTION 10
The data engineering team is migrating an enterprise system with thousands of tables and views into the Lakehouse. They plan to implement the target architecture using a series of bronze, silver, and gold tables. Bronze tables will almost exclusively be used by production data engineering workloads, while silver tables will be used to support both data engineering and machine learning workloads. Gold tables will largely serve business intelligence and reporting purposes. While personal identifying information (PII) exists in all tiers of data, pseudonymization and anonymization rules are in place for all data at the silver and gold levels.
The organization is interested in reducing security concerns while maximizing the ability to collaborate across diverse teams.
Which statement exemplifies best practices for implementing this system?

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

Answer: A

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

NEW QUESTION 14
A junior data engineer is working to implement logic for a Lakehouse table named silver_device_recordings. The source data contains 100 unique fields in a highly nested JSON structure.
The silver_device_recordings table will be used downstream for highly selective joins on a number of fields, and will also be leveraged by the machine learning team to filter on a handful of relevant fields, in total, 15 fields have been identified that will often be used for filter and join logic.
The data engineer is trying to determine the best approach for dealing with these nested fields before declaring the table schema.
Which of the following accurately presents information about Delta Lake and Databricks that may Impact their decision-making process?

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

Answer: D

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

NEW QUESTION 16
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 19
A developer has successfully configured credential for Databricks Repos and cloned a remote Git repository. Hey don not have privileges to make changes to the main branch, which is the only branch currently visible in their workspace.
Use Response to pull changes from the remote Git repository commit and push changes to a branch that appeared as a changes were pulled.

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

Answer: C

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

NEW QUESTION 23
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 29
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 31
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 34
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 39
The Databricks workspace administrator has configured interactive clusters for each of the data engineering groups. To control costs, clusters are set to terminate after 30 minutes of inactivity. Each user should be able to execute workloads against their assigned clusters at any time of the day.
Assuming users have been added to a workspace but not granted any permissions, which of the following describes the minimal permissions a user would need to start and attach to an already configured cluster.

A. "Can Manage" privileges on the required cluster
B. Workspace Admin privileges, cluster creation allowe
C. "Can Attach To" privileges on the required cluster
D. Cluster creation allowe
E. "Can Attach To" privileges on the required cluster
F. "Can Restart" privileges on the required cluster
G. Cluster creation allowe
H. "Can Restart" privileges on the required cluster

Answer: D

Explanation:
https://learn.microsoft.com/en-us/azure/databricks/security/auth-authz/access-control/cluster-acl
https://docs.databricks.com/en/security/auth-authz/access-control/cluster-acl.html

NEW QUESTION 44
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 49
The data engineering team maintains the following code:

```
import pyspark.sql.functions as F

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

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

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

Answer: C

Explanation:
This code is using the pyspark.sql.functions library to group the silver_customer_sales table by customer_id and then aggregate the data using the minimum sale date, maximum sale total, and sum of distinct order ids. The resulting aggregated data is then written to the gold_customer_lifetime_sales_summary table,

overwriting any existing data in that table. This is a batch job that does not use any incremental or streaming logic, and does not perform any merge or update operations. Therefore, the code will overwrite the gold table with the aggregated values from the silver table every time it is executed. References:
? https://docs.databricks.com/spark/latest/dataframes-datasets/introduction-to-dataframes-python.html
? https://docs.databricks.com/spark/latest/dataframes-datasets/transforming-data- with-dataframes.html
? https://docs.databricks.com/spark/latest/dataframes-datasets/aggregating-data- with-dataframes.html

NEW QUESTION 53
A team of data engineer are adding tables to a DLT pipeline that contain repetitive expectations for many of the same data quality checks.
One member of the team suggests reusing these data quality rules across all tables defined for this pipeline.
What approach would allow them to do this?

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

Answer: A

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

NEW QUESTION 54
A Spark job is taking longer than expected. Using the Spark UI, a data engineer notes that the Min, Median, and Max Durations for tasks in a particular stage show the minimum and median time to complete a task as roughly the same, but the max duration for a task to be roughly 100 times as long as the minimum.
Which situation is causing increased duration of the overall job?

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

Answer: D

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

NEW QUESTION 56
The data engineer team has been tasked with configured connections to an external database that does not have a supported native connector with Databricks. The external database already has data security configured by group membership. These groups map directly to user group already created in Databricks that represent various teams within the company.
A new login credential has been created for each group in the external database. The Databricks Utilities Secrets module will be used to make these credentials available to Databricks users.
Assuming that all the credentials are configured correctly on the external database and group membership is properly configured on Databricks, which statement describes how teams can be granted the minimum necessary access to using these credentials?

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

Answer: C

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

NEW QUESTION 57
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 60
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 63
Which statement regarding spark configuration on the Databricks platform is true?

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

Answer: A

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

NEW QUESTION 66
An upstream system is emitting change data capture (CDC) logs that are being written to a cloud object storage directory. Each record in the log indicates the change type (insert, update, or delete) and the values for each field after the change. The source table has a primary key identified by the field pk_id.
For auditing purposes, the data governance team wishes to maintain a full record of all values that have ever been valid in the source system. For analytical purposes, only the most recent value for each record needs to be recorded. The Databricks job to ingest these records occurs once per hour, but each individual record may have changed multiple times over the course of an hour.
Which solution meets these requirements?

A. Create a separate history table for each pk_id resolve the current state of the table by running a union all filtering the history tables for the most recent state.
B. Use merge into to insert, update, or delete the most recent entry for each pk_id into abronze table, then propagate all changes throughout the system.
C. Iterate through an ordered set of changes to the table, applying each in turn; rely on Delta Lake's versioning ability to create an audit log.
D. Use Delta Lake's change data feed to automatically process CDC data from an external system, propagating all changes to all dependent tables in the Lakehouse.
E. Ingest all log information into a bronze table; use merge into to insert, update, or delete the most recent entry for each pk_id into a silver table to recreate the current table state.

Answer: B

Explanation:
This is the correct answer because it meets the requirements of maintaining a full record of all values that have ever been valid in the source system and recreating the current table state with only the most recent value for each record. The code ingests all log information into a bronze table, which preserves the raw CDC data as it is. Then, it uses merge into to perform an upsert operation on a silver table, which means it will insert new records or update or delete existing records based on the change type and the pk_id columns. This way, the silver table will always reflect the current state of the source table, while the bronze table will keep the history of all changes. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Upsert into a table using merge” section.

NEW QUESTION 71
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 72
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 75
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 76
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 80
The security team is exploring whether or not the Databricks secrets module can be leveraged for connecting to an external database.
After testing the code with all Python variables being defined with strings, they upload the password to the secrets module and configure the correct permissions for the currently active user. They then modify their code to the following (leaving all other variables unchanged).

```
password = dbutils.secrets.get(scope="db_creds", key="jdbc_password")

print(password)

df = (spark
  .read
  .format("jdbc")
  .option("url", connection)
  .option("dbtable", tablename)
  .option("user", username)
  .option("password", password)
  )
```

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

A. The connection to the external table will fail; the string "redacted" will be printed.
B. An interactive input box will appear in the notebook; if the right password is provided, the connection will succeed and the encoded password will be saved to DBFS.
C. An interactive input box will appear in the notebook; if the right password is provided, the connection will succeed and the password will be printed in plain text.
D. The connection to the external table will succeed; the string value of password will be printed in plain text.
E. The connection to the external table will succeed; the string "redacted" will be printed.

Answer: E

Explanation:
This is the correct answer because the code is using the dbutils.secrets.get method to retrieve the password from the secrets module and store it in a variable. The secrets module allows users to securely store and access sensitive information such as passwords, tokens, or API keys. The connection to the external table will succeed because the password variable will contain the actual password value. However, when printing the password variable, the string “redacted” will be displayed instead of the plain text password, as a security measure to prevent exposing sensitive information in notebooks. Verified References: [Databricks Certified Data Engineer Professional], under “Security & Governance” section; Databricks Documentation, under “Secrets” section.

NEW QUESTION 81
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 82
A Delta table of weather records is partitioned by date and has the below schema: date DATE, device_id INT, temp FLOAT, latitude FLOAT, longitude FLOAT
To find all the records from within the Arctic Circle, you execute a query with the below filter:
latitude > 66.3
Which statement describes how the Delta engine identifies which files to load?

A. All records are cached to an operational database and then the filter is applied
B. The Parquet file footers are scanned for min and max statistics for the latitude column
C. All records are cached to attached storage and then the filter is applied
D. The Delta log is scanned for min and max statistics for the latitude column
E. The Hive metastore is scanned for min and max statistics for the latitude column

Answer: D

Explanation:
This is the correct answer because Delta Lake uses a transaction log to store metadata about each table, including min and max statistics for each column in each data file. The Delta engine can use this information to quickly identify which files to load based on a filter condition, without scanning the entire table or the file footers. This is called data skipping and it can improve query performance significantly. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; [Databricks Documentation], under “Optimizations - Data Skipping” section.
In the Transaction log, Delta Lake captures statistics for each data file of the table. These statistics indicate per file:
- Total number of records
- Minimum value in each column of the first 32 columns of the table
- Maximum value in each column of the first 32 columns of the table
- Null value counts for in each column of the first 32 columns of the table
When a query with a selective filter is executed against the table, the query optimizer uses these statistics to generate the query result. it leverages them to identify data files that may contain records matching the conditional filter.
For the SELECT query in the question, The transaction log is scanned for min and max statistics for the price column

NEW QUESTION 86
The view updates represents an incremental batch of all newly ingested data to be inserted or updated in the customers table.
The following logic is used to process these records.
Which statement describes this implementation?

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

Answer: A

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

NEW QUESTION 91
Two of the most common data locations on Databricks are the DBFS root storage and external object storage mounted with dbutils.fs.mount().
Which of the following statements is correct?

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

Answer: A

Explanation:
DBFS is a file system protocol that allows users to interact with files stored in object storage using syntax and guarantees similar to Unix file systems1. DBFS is not a physical file system, but a layer over the object storage that provides a unified view of data across different data sources1. By default, the DBFS root is accessible to all users in the workspace, and the access to mounted data sources depends on the permissions of the storage account or container2. Mounted

storage volumes do not need to have full public read and write permissions, but they do require a valid connection string or access key to be provided when mounting3. Both the DBFS root and mounted storage can be accessed when using %sh in a Databricks notebook, as long as the cluster has FUSE enabled4. The DBFS root does not store files in ephemeral block volumes attached to the driver, but in the object storage associated with the workspace1. Mounted directories will persist saved data to external storage between sessions, unless they are unmounted or deleted3. References: DBFS, Work with files on Azure Databricks, Mounting cloud object storage on Azure Databricks, Access DBFS with FUSE

NEW QUESTION 93
Which configuration parameter directly affects the size of a spark-partition upon ingestion of data into Spark?

A. spark.sql.files.maxPartitionBytes
B. spark.sql.autoBroadcastJoinThreshold
C. spark.sql.files.openCostInBytes
D. spark.sql.adaptive.coalescePartitions.minPartitionNum
E. spark.sql.adaptive.advisoryPartitionSizeInBytes

Answer: A

Explanation:
This is the correct answer because spark.sql.files.maxPartitionBytes is a configuration parameter that directly affects the size of a spark-partition upon ingestion of data into Spark. This parameter configures the maximum number of bytes to pack into a single partition when reading files from file-based sources such as Parquet, JSON and ORC. The default value is 128 MB, which means each partition will be roughly 128 MB in size, unless there are too many small files or only one large file. Verified References: [Databricks Certified Data Engineer Professional], under “Spark Configuration”
section; Databricks Documentation, under “Available Properties - spark.sql.files.maxPartitionBytes” section.

NEW QUESTION 94
A data architect has heard about lake's built-in versioning and time travel capabilities. For auditing purposes they have a requirement to maintain a full of all valid street addresses as they appear in the customers table.
The architect is interested in implementing a Type 1 table, overwriting existing records with new values and relying on Delta Lake time travel to support long-term auditing. A data engineer on the project feels that a Type 2 table will provide better performance and scalability.
Which piece of information is critical to this decision?

A. Delta Lake time travel does not scale well in cost or latency to provide a long-term versioning solution.
B. Delta Lake time travel cannot be used to query previous versions of these tables because Type 1 changes modify data files in place.
C. Shallow clones can be combined with Type 1 tables to accelerate historic queries for long-term versioning.
D. Data corruption can occur if a query fails in a partially completed state because Type 2 tables requiresSetting multiple fields in a single update.

Answer: A

Explanation:
Delta Lake's time travel feature allows users to access previous versions of a table, providing a powerful tool for auditing and versioning. However, using time travel as a long-term versioning solution for auditing purposes can be less optimal in terms of cost and performance, especially as the volume of data and the number of versions grow. For maintaining a full history of valid street addresses as they appear in a customers table, using a Type 2 table (where each update creates a new record with versioning) might provide better scalability and performance by avoiding the overhead associated with accessing older versions of a large table. While Type 1 tables, where existing records are overwritten with new values, seem simpler and can leverage time travel for auditing, the critical piece of information is that time travel might not scale well in cost or latency for long- term versioning needs, making a Type 2 approach more viable for performance and scalability.References:
? Databricks Documentation on Delta Lake's Time Travel: Delta Lake Time Travel
? Databricks Blog on Managing Slowly Changing Dimensions in Delta Lake: Managing SCDs in Delta Lake

NEW QUESTION 96
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 99
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 101
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 106
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 109

.....

Thank You for Trying Our Product

We offer two products:

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

2nd - Questons and Answers in PDF Format

Databricks-Certified-Professional-Data-Engineer Practice Exam Features:

* Databricks-Certified-Professional-Data-Engineer Questions and Answers Updated Frequently

* Databricks-Certified-Professional-Data-Engineer Practice Questions Verified by Expert Senior Certified Staff

* Databricks-Certified-Professional-Data-Engineer Most Realistic Questions that Guarantee you a Pass on Your FirstTry

* Databricks-Certified-Professional-Data-Engineer Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year