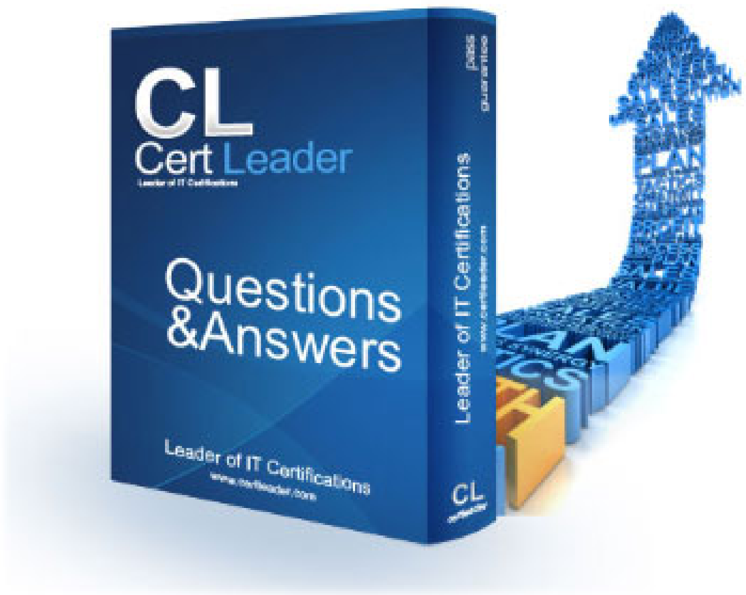
Databricks-Certified-Professional-Data-Engineer Dumps

Databricks Certified Data Engineer Professional Exam

https://www.certleader.com/Databricks-Certified-Professional-Data-Engineer-dumps.html



NEW QUESTION 1
A junior data engineer seeks to leverage Delta Lake's Change Data Feed functionality to create a Type 1 table representing all of the values that have ever been valid for all rows in a bronze table created with the property delta.enableChangeDataFeed = true. They plan to execute the following code as a daily job:
Which statement describes the execution and results of running the above query multiple times?

A. Each time the job is executed, newly updated records will be merged into the target table, overwriting previous values with the same primary keys.
B. Each time the job is executed, the entire available history of inserted or updated records will be appended to the target table, resulting in many duplicate entries.
C. Each time the job is executed, the target table will be overwritten using the entire history of inserted or updated records, giving the desired result.
D. Each time the job is executed, the differences between the original and current versions are calculated; this may result in duplicate entries for some records.
E. Each time the job is executed, only those records that have been inserted or updated since the last execution will be appended to the target table giving the desired result.

Answer: B

Explanation:
Reading table’s changes, captured by CDF, using spark.read means that you are reading them as a static source. So, each time you run the query, all table’s changes (starting from the specified startingVersion) will be read.

NEW QUESTION 2
A user new to Databricks is trying to troubleshoot long execution times for some pipeline logic they are working on. Presently, the user is executing code cell-by-cell, using display() calls to confirm code is producing the logically correct results as new transformations are added to an operation. To get a measure of average time to execute, the user is running each cell multiple times interactively.
Which of the following adjustments will get a more accurate measure of how code is likely to perform in production?

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

Answer: D

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

NEW QUESTION 3
The data engineer team is configuring environment for development testing, and production before beginning migration on a new data pipeline. The team requires extensive testing on both the code and data resulting from code execution, and the team want to develop and test against similar production data as possible.
A junior data engineer suggests that production data can be mounted to the development testing environments, allowing pre production code to execute against production data. Because all users have
Admin privileges in the development environment, the junior data engineer has offered to configure permissions and mount this data for the team.
Which statement captures best practices for this situation?

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

Answer: C

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

NEW QUESTION 4
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 5
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 6
Which statement characterizes the general programming model used by Spark Structured Streaming?

A. Structured Streaming leverages the parallel processing of GPUs to achieve highly parallel data throughput.
B. Structured Streaming is implemented as a messaging bus and is derived from Apache Kafka.
C. Structured Streaming uses specialized hardware and I/O streams to achieve sub- second latency for data transfer.
D. Structured Streaming models new data arriving in a data stream as new rows appended to an unbounded table.
E. Structured Streaming relies on a distributed network of nodes that hold incremental state values for cached stages.

Answer: B

Explanation:
This is the correct answer because it characterizes the general programming model used by Spark Structured Streaming, which is to treat a live data stream as a table that is being continuously appended. This leads to a new stream processing model that is very similar to a batch processing model, where users can express their streaming computation using the same Dataset/DataFrame API as they would use for static data. The Spark SQL engine will take care of running the streaming query incrementally and continuously and updating the final result as streaming data continues to arrive. Verified References: [Databricks Certified Data Engineer Professional], under “Structured Streaming” section; Databricks Documentation, under “Overview” section.

NEW QUESTION 7
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 8
A data engineer is testing a collection of mathematical functions, one of which calculates the area under a curve as described by another function.
Which kind of the test does the above line exemplify?

A. Integration
B. Unit
C. Manual
D. functional

Answer: B

Explanation:
A unit test is designed to verify the correctness of a small, isolated piece of
code, typically a single function. Testing a mathematical function that calculates the area under a curve is an example of a unit test because it is testing a specific, individual function to ensure it operates as expected.
References:
? Software Testing Fundamentals: Unit Testing

NEW QUESTION 9
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 10
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 10
A Structured Streaming job deployed to production has been experiencing delays during peak hours of the day. At present, during normal execution, each microbatch of data is processed in less than 3 seconds. During peak hours of the day, execution time for each microbatch becomes very inconsistent, sometimes exceeding 30 seconds. The streaming write is currently configured with a trigger interval of 10 seconds.
Holding all other variables constant and assuming records need to be processed in less than 10 seconds, which adjustment will meet the requirement?

A. Decrease the trigger interval to 5 seconds; triggering batches more frequently allows idle executors to begin processing the next batch while longer running tasks from previous batches finish.
B. Increase the trigger interval to 30 seconds; setting the trigger interval near the maximum execution time observed for each batch is always best practice to ensure no records are dropped.

C. The trigger interval cannot be modified without modifying the checkpoint directory; to maintain the current stream state, increase the number of shuffle partitions to maximize parallelism.
D. Use the trigger once option and configure a Databricks job to execute the query every 10 seconds; this ensures all backlogged records are processed with each batch.
E. Decrease the trigger interval to 5 seconds; triggering batches more frequently may prevent records from backing up and large batches from causing spill.

Answer: E

Explanation:
The adjustment that will meet the requirement of processing records in less than 10 seconds is to decrease the trigger interval to 5 seconds. This is because triggering batches more frequently may prevent records from backing up and large batches from causing spill. Spill is a phenomenon where the data in memory exceeds the available capacity and has to be written to disk, which can slow down the processing and increase the execution time1. By reducing the trigger interval, the streaming query can process smaller batches of data more quickly and avoid spill. This can also improve the latency and throughput of the streaming job2.
The other options are not correct, because:
? Option A is incorrect because triggering batches more frequently does not allow idle executors to begin processing the next batch while longer running tasks from previous batches finish. In fact, the opposite is true. Triggering batches more frequently may cause concurrent batches to compete for the same resources and cause contention and backpressure2. This can degrade the performance and stability of the streaming job.
? Option B is incorrect because increasing the trigger interval to 30 seconds is not a good practice to ensure no records are dropped. Increasing the trigger interval means that the streaming query will process larger batches of data less frequently, which can increase the risk of spill, memory pressure, and timeouts12. This can also increase the latency and reduce the throughput of the streaming job.
? Option C is incorrect because the trigger interval can be modified without modifying the checkpoint directory. The checkpoint directory stores the metadata and state of the streaming query, such as the offsets, schema, and configuration3. Changing the trigger interval does not affect the state of the streaming query, and does not require a new checkpoint directory. However, changing the number of shuffle partitions may affect the state of the streaming query, and may require a new checkpoint directory4.
? Option D is incorrect because using the trigger once option and configuring a Databricks job to execute the query every 10 seconds does not ensure that all backlogged records are processed with each batch. The trigger once option means that the streaming query will process all the available data in the source and then stop5. However, this does not guarantee that the query will finish processing within 10 seconds, especially if there are a lot of records in the source. Moreover, configuring a Databricks job to execute the query every 10 seconds may cause overlapping or missed batches, depending on the execution time of the query.
References: Memory Management Overview, Structured Streaming Performance Tuning Guide, Checkpointing, Recovery Semantics after Changes in a Streaming Query, Triggers

NEW QUESTION 13
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 18
A Delta Lake table representing metadata about content posts from users has the following schema:
user_id LONG, post_text STRING, post_id STRING, longitude FLOAT, latitude FLOAT, post_time TIMESTAMP, date DATE
This table is partitioned by the date column. A query is run with the following filter: longitude < 20 & longitude > -20
Which statement describes how data will be filtered?

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

Answer: D

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

NEW QUESTION 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 28
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 32
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
The following table consists of items found in user carts within an e-commerce website.

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

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

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

How would the following update be handled?

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

How would the following update be handled?

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

Answer: D

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

NEW QUESTION 39
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 40
The DevOps team has configured a production workload as a collection of notebooks scheduled to run daily using the Jobs Ul. A new data engineering hire is onboarding to the team and has requested access to one of these notebooks to review the production logic.
What are the maximum notebook permissions that can be granted to the user without allowing accidental changes to production code or data?

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

Answer: D

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

NEW QUESTION 43
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 45
A member of the data engineering team has submitted a short notebook that they wish to schedule as part of a larger data pipeline. Assume that the commands provided below produce the logically correct results when run as presented.

```
Cmd 1

rawDF = spark.table("raw_data")

Cmd 2

rawDF.printSchema()

Cmd 3

flattenedDF = rawDF.select("*", "values.*")

Cmd 4

finalDF = flattenedDF.drop("values")

Cmd 5

display(finalDF)

Cmd 6

finalDF.write.mode("append").saveAsTable("flat_data")
```

Which command should be removed from the notebook before scheduling it as a job?

A. Cmd 2
B. Cmd 3
C. Cmd 4
D. Cmd 5
E. Cmd 6

Answer: E

Explanation:
Cmd 6 is the command that should be removed from the notebook before scheduling it as a job. This command is selecting all the columns from the finalDF dataframe and displaying them in the notebook. This is not necessary for the job, as the finalDF dataframe is already written to a table in Cmd 7. Displaying the dataframe in the notebook will only consume resources and time, and it will not affect the output of the job. Therefore, Cmd 6 is redundant and should be removed.
The other commands are essential for the job, as they perform the following tasks:
? Cmd 1: Reads the raw_data table into a Spark dataframe called rawDF.
? Cmd 2: Prints the schema of the rawDF dataframe, which is useful for debugging and understanding the data structure.
? Cmd 3: Selects all the columns from the rawDF dataframe, as well as the nested columns from the values struct column, and creates a new dataframe called flattenedDF.
? Cmd 4: Drops the values column from the flattenedDF dataframe, as it is no longer needed after flattening, and creates a new dataframe called finalDF.
? Cmd 5: Explains the physical plan of the finalDF dataframe, which is useful for optimizing and tuning the performance of the job.
? Cmd 7: Writes the finalDF dataframe to a table called flat_data, using the append mode to add new data to the existing table.

NEW QUESTION 50
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 53
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 55
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 57
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 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. 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 65
Which statement describes integration testing?

A. Validates interactions between subsystems of your application
B. Requires an automated testing framework
C. Requires manual intervention
D. Validates an application use case
E. Validates behavior of individual elements of your application

Answer: D

Explanation:
This is the correct answer because it describes integration testing. Integration testing is a type of testing that validates interactions between subsystems of your application, such as modules, components, or services. Integration testing ensures that the subsystems work together as expected and produce the correct outputs or results. Integration testing can be done at different levels of granularity, such as component integration testing, system integration testing, or end-to-end testing. Integration testing can help detect errors or bugs that may not be found by unit testing, which only validates behavior of individual elements of your application. Verified References: [Databricks Certified Data Engineer Professional], under “Testing” section; Databricks Documentation, under “Integration testing” section.

NEW QUESTION 67
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 71
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 72
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 74
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 76
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 80
A nightly job ingests data into a Delta Lake table using the following code:

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

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

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

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

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

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

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

Answer: E

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

NEW QUESTION 85
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 86
Which is a key benefit of an end-to-end test?

A. It closely simulates real world usage of your application.
B. It pinpoint errors in the building blocks of your application.
C. It provides testing coverage for all code paths and branches.
D. It makes it easier to automate your test suite

Answer: A

Explanation:
End-to-end testing is a methodology used to test whether the flow of an application, from start to finish, behaves as expected. The key benefit of an end-to-end test is that it closely simulates real-world, user behavior, ensuring that the system as a whole operates correctly.
References:
? Software Testing: End-to-End Testing

NEW QUESTION 90
Which of the following is true of Delta Lake and the Lakehouse?

A. Because Parquet compresses data row by ro
B. strings will only be compressed when a character is repeated multiple times.
C. Delta Lake automatically collects statistics on the first 32 columns of each table which are leveraged in data skipping based on query filters.
D. Views in the Lakehouse maintain a valid cache of the most recent versions of source tables at all times.
E. Primary and foreign key constraints can be leveraged to ensure duplicate values are never entered into a dimension table.
F. Z-order can only be applied to numeric values stored in Delta Lake tables

Answer: B

Explanation:
https://docs.delta.io/2.0.0/table-properties.html
Delta Lake automatically collects statistics on the first 32 columns of each table, which are leveraged in data skipping based on query filters1. Data skipping is a performance optimization technique that aims to avoid reading irrelevant data from the storage layer1. By collecting statistics such as min/max values, null counts, and bloom filters, Delta Lake can efficiently prune unnecessary files or partitions from the query plan1. This can significantly improve the query performance and reduce the I/O cost.
The other options are false because:
? Parquet compresses data column by column, not row by row2. This allows for better compression ratios, especially for repeated or similar values within a column2.
? Views in the Lakehouse do not maintain a valid cache of the most recent versions of source tables at all times3. Views are logical constructs that are defined by a SQL query on one or more base tables3. Views are not materialized by default, which means they do not store any data, but only the query definition3. Therefore, views always reflect the latest state of the source tables when queried3. However, views can be cached manually using the CACHE TABLE or CREATE TABLE AS SELECT commands.
? Primary and foreign key constraints can not be leveraged to ensure duplicate values are never entered into a dimension table. Delta Lake does not support enforcing primary and foreign key constraints on tables. Constraints are logical rules that define the integrity and validity of the data in a table. Delta Lake relies on

the application logic or the user to ensure the data quality and consistency.
? Z-order can be applied to any values stored in Delta Lake tables, not only numeric values. Z-order is a technique to optimize the layout of the data files by sorting them on one or more columns. Z-order can improve the query performance by clustering related values together and enabling more efficient data skipping. Z-order can be applied to any column that has a defined ordering, such as numeric, string, date, or boolean values.
References: Data Skipping, Parquet Format, Views, [Caching], [Constraints], [Z-Ordering]

NEW QUESTION 91
Where in the Spark UI can one diagnose a performance problem induced by not leveraging predicate push-down?

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

Answer: E

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

NEW QUESTION 92
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 97
......