Question 1)

Which of the statements are incorrect when choosing between lakehouse and Datawarehouse?

○

Lakehouse can have special indexes and caching which are optimized for Machine learning

○

Lakehouse cannot serve low query latency with high reliability for BI workloads, only suitable for batch workloads.

○

Lakehouse can be accessed through various API's including but not limited to Python/R/SQL

○

Traditional Data warehouses have storage and compute are coupled.

○

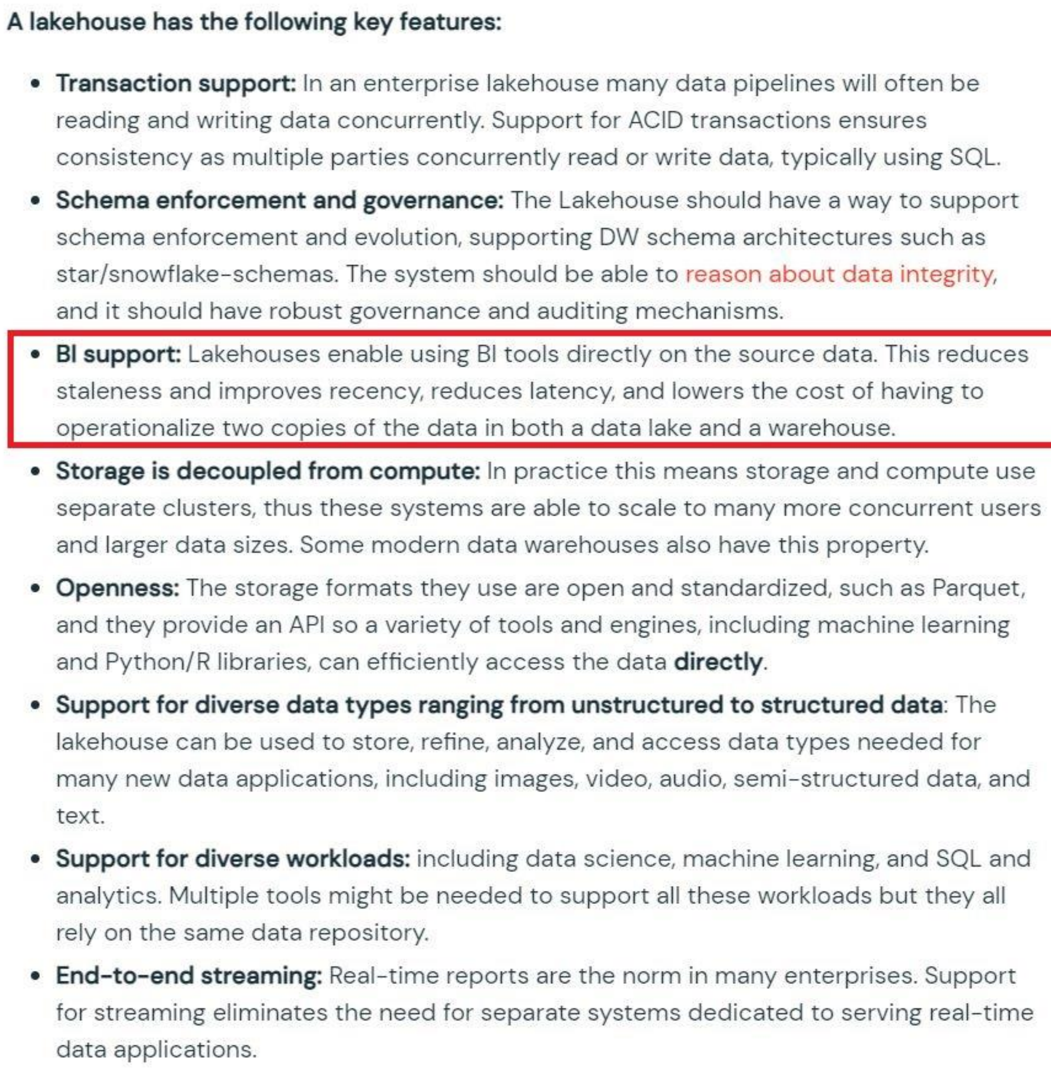
Lakehouse uses standard data formats like Parquet.

Explanation
Answer is B:
The answer is Lakehouse cannot serve low query latency with high reliability for BI workloads, only suitable for batch workloads.
Lakehouse can replace traditional warehouses by leveraging storage and compute optimizations like caching to serve them with low query latency with high reliability.

Focus on comparisons between Spark Cache vs Delta Cache.

https://docs.databricks.com/delta/optimizations/delta-cache.html

What Is a Lakehouse? - The Databricks Blog



A lakehouse has the following key features:

- **Transaction support:** In an enterprise lakehouse many data pipelines will often be reading and writing data concurrently. Support for ACID transactions ensures consistency as multiple parties concurrently read or write data, typically using SQL.
- **Schema enforcement and governance:** The Lakehouse should have a way to support schema enforcement and evolution, supporting DW schema architectures such as star/snowflake-schemas. The system should be able to reason about data integrity, and it should have robust governance and auditing mechanisms.
- **BI support:** Lakehouses enable using BI tools directly on the source data. This reduces staleness and improves recency, reduces latency, and lowers the cost of having to operationalize two copies of the data in both a data lake and a warehouse.
- **Storage is decoupled from compute:** In practice this means storage and compute use separate clusters, thus these systems are able to scale to many more concurrent users and larger data sizes. Some modern data warehouses also have this property.
- **Openness:** The storage formats they use are open and standardized, such as Parquet, and they provide an API so a variety of tools and engines, including machine learning and Python/R libraries, can efficiently access the data **directly**.
- **Support for diverse data types ranging from unstructured to structured data**: The lakehouse can be used to store, refine, analyze, and access data types needed for many new data applications, including images, video, audio, semi-structured data, and text.
- **Support for diverse workloads:** including data science, machine learning, and SQL and analytics. Multiple tools might be needed to support all these workloads but they all rely on the same data repository.
- **End-to-end streaming:** Real-time reports are the norm in many enterprises. Support for streaming eliminates the need for separate systems dedicated to serving real-time data applications.

Question 2)

Which of the statements are correct about lakehouse?

○ Lakehouse only supports Machine learning workloads and Data warehouses support BI workloads

○ Lakkehouse only supports end-to-end streaming workloads and Data warehouses support Batch workloads

○ Lakehouse does not support ACID

○ In Lakehouse Storage and compute are coupled

○ Lakehouse supports schema enforcement and evolution

[✓ ●]

Explanation

The answer is Lakehouse supports schema enforcement and evolution,

Lakehouse using Delta lake can not only enforce a schema on write which is contrary to traditional big data systems that can only enforce a schema on read, it also supports evolving schema over time with the ability to control the evolution.

For example below is the Dataframe writer API and it supports three modes of enforcement and evolution,

Default: Only enforcement, no changes are allowed and any schema drift/evolution will result in failure.

Merge: Flexible, supports enforcement and evolution

New columns are added

Evolves nested columns

Supports evolving data types, like Byte to Short to Integer to Bigint

How to enable:

```
DF.write.format("delta").option("mergeSchema", "true").saveAsTable("table_name")
                                                   or
spark.databricks.delta.schema.autoMerge = True  ## Spark session
```

Overwrite: No enforcement

Dropping columns

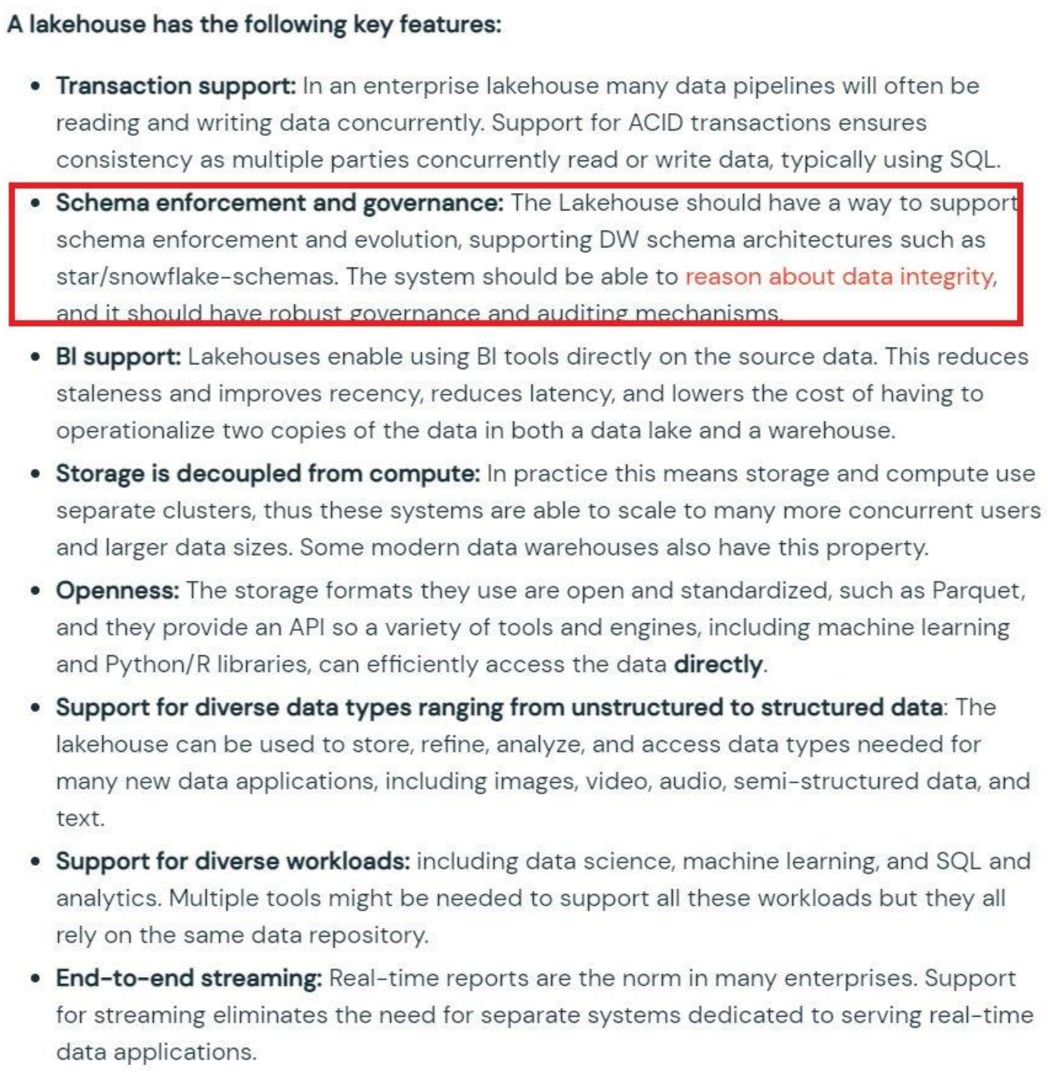
Change string to integer

Rename columns

How to enable:

```
DF.write.format("delta").option("overwriteSchema",
"True").saveAsTable("table_name")
```

What Is a Lakehouse? - The Databricks Blog



A lakehouse has the following key features:

- **Transaction support:** In an enterprise lakehouse many data pipelines will often be reading and writing data concurrently. Support for ACID transactions ensures consistency as multiple parties concurrently read or write data, typically using SQL.
- **Schema enforcement and governance:** The Lakehouse should have a way to support schema enforcement and evolution, supporting DW schema architectures such as star/snowflake-schemas. The system should be able to reason about data integrity, and it should have robust governance and auditing mechanisms.
- **BI support:** Lakehouses enable using BI tools directly on the source data. This reduces staleness and improves recency, reduces latency, and lowers the cost of having to operationalize two copies of the data in both a data lake and a warehouse.
- **Storage is decoupled from compute:** In practice this means storage and compute use separate clusters, thus these systems are able to scale to many more concurrent users and larger data sizes. Some modern data warehouses also have this property.
- **Openness:** The storage formats they use are open and standardized, such as Parquet, and they provide an API so a variety of tools and engines, including machine learning and Python/R libraries, can efficiently access the data **directly**.
- **Support for diverse data types ranging from unstructured to structured data**: The lakehouse can be used to store, refine, analyze, and access data types needed for many new data applications, including images, video, audio, semi-structured data, and text.
- **Support for diverse workloads:** including data science, machine learning, and SQL and analytics. Multiple tools might be needed to support all these workloads but they all rely on the same data repository.
- **End-to-end streaming:** Real-time reports are the norm in many enterprises. Support for streaming eliminates the need for separate systems dedicated to serving real-time data applications.

Question 3:
Which of the following are stored in the control pane of Databricks Architecture?

○ Job Clusters

○ All Purpose Clusters

○ Databricks Filesystem

○ Databricks Web Application

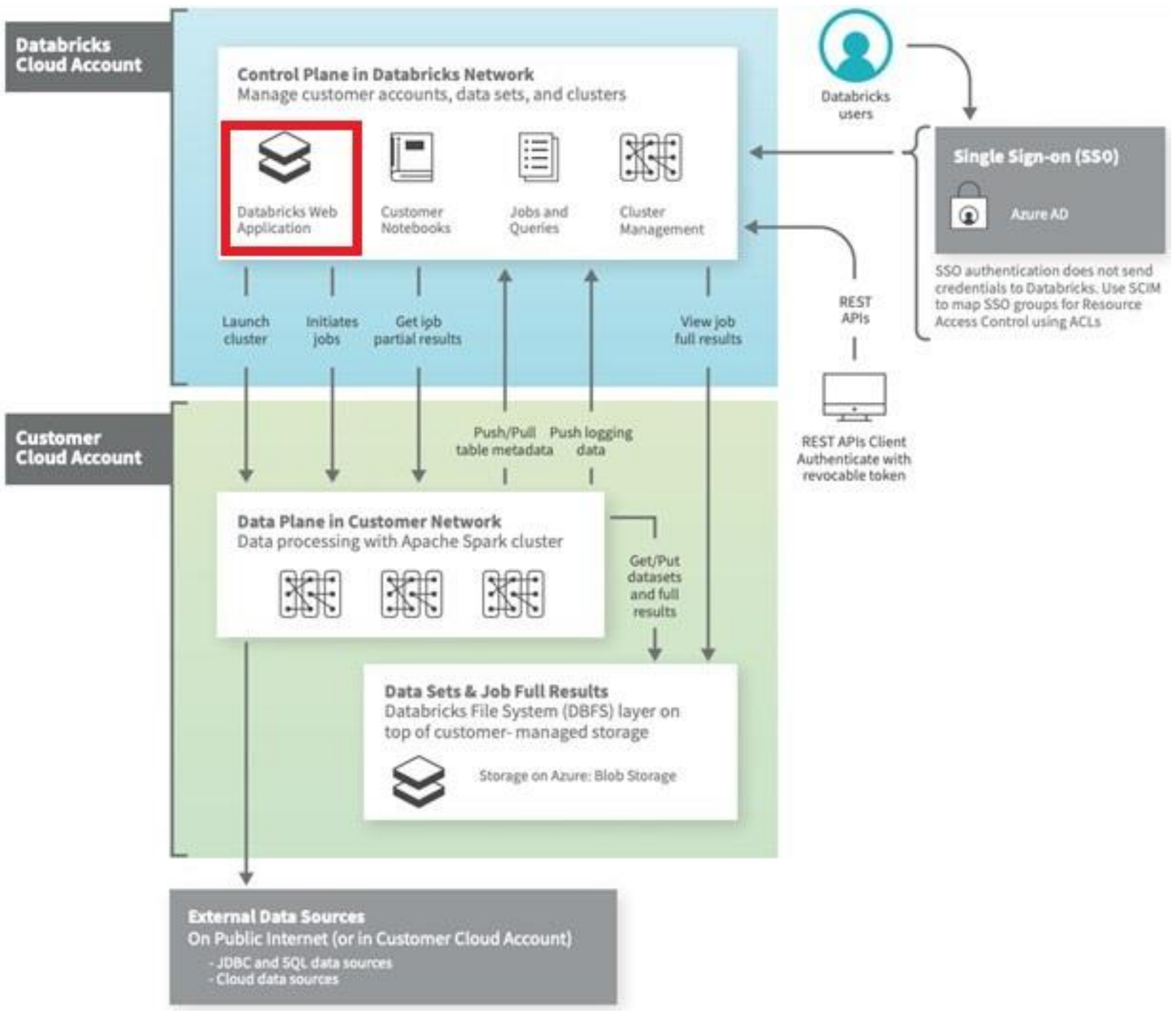
○ Delta tables

Explanation
The answer is Databricks Web Application
[Azure Databricks architecture overview - Azure Databricks | Microsoft Docs]
Databricks operates most of its services out of a control plane and a data plane, *please note serverless features like SQL Endpoint and DLT compute use shared compute in Control pane.*

Control Plane: Stored in Databricks Cloud Account

The control plane includes the backend services that Databricks manages in its own Azure account. Notebook commands and many other workspace configurations are stored in the control plane and encrypted at rest.
Data Plane: Stored in Customer Cloud Account

The data plane is managed by your Azure account and is where your data resides. This is also where data is processed. You can use Azure Databricks connectors so that your clusters can connect to external data sources outside of your Azure account to ingest data or for storage.


Databricks Cloud Account
Control Plane in Databricks Network
Manage customer accounts, data sets, and clusters
Databricks Web Application
Customer Notebooks
Jobs and Queries
Cluster Management
Databricks users
Single Sign-on (SSO)
Azure AD
SSO authentication does not send credentials to Databricks. Use SCIM to map SSO groups for Resource Access Control using ACLs
REST APIs
REST APIs Client Authenticate with revocable token
Launch cluster
Initiates jobs
Get job partial results
View job full results
Push/Pull table metadata
Push logging data
Customer Cloud Account
Data Plane in Customer Network
Data processing with Apache Spark cluster
Get/Put datasets and full results
Data Sets & Job Full Results
Databricks File System (DBFS) layer on top of customer-managed storage
Storage on Azure: Blob Storage
External Data Sources
On Public Internet (or in Customer Cloud Account)
- JDBC and SQL data sources
- Cloud data sources

Question 4:
You have written a notebook to generate a summary data set for reporting, Notebook was scheduled using the job cluster, but you realized it takes 8 minutes to start the cluster, what feature can be to start the cluster in a timely fashion

○ Setup an additional job to run ahead of the actual job so the cluster is running second job starts

○ Use the Databricks cluster pools feature to reduce the startup time

○ Use Databricks Premium edition instead of Databricks standard edition

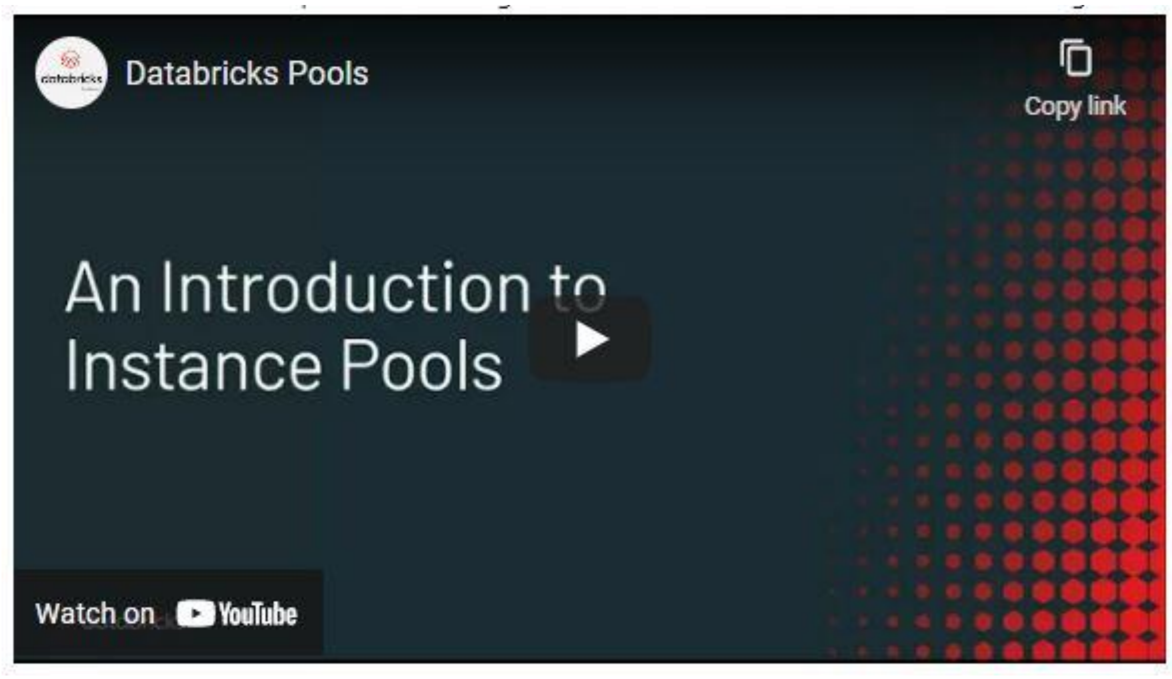
○ Pin the cluster in the cluster UI page so it is always available to the jobs

○ Disable auto termination so the cluster is always running

Explanation

Cluster pools allow us to reserve VM's ahead of time, when a new job cluster is created VM are grabbed from the pool. Note: when the VM's are waiting to be used by the cluster only cost incurred is Azure. Databricks run time cost is only billed once VM is allocated to a cluster.

Here is a demo of how to setup a pool and follow some best practices,



Question 5:
Which of the following developer operations in the CI/CD can only be implemented through a GIT provider when using Databricks Repos.

○ Trigger Databricks Repos pull API to update the latest version

○ Commit and push code

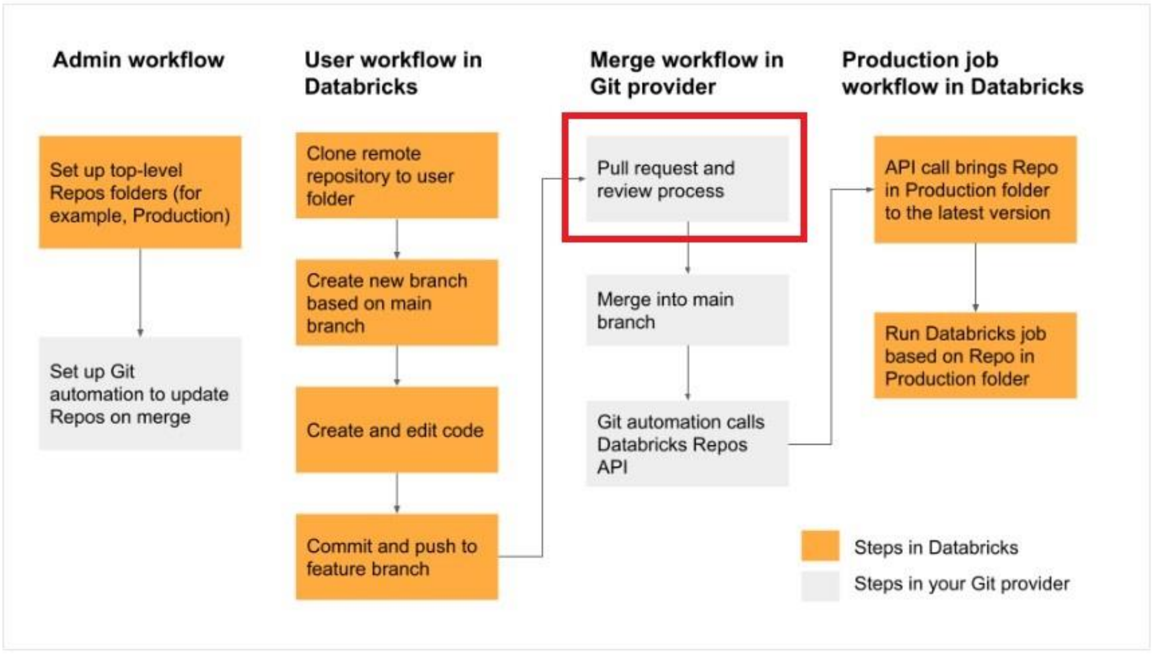
○ Create and edit code

○ Create a new branch

○ Pull request and review process

Explanation
The answer is Pull request and review process, please note: the question is asking for steps that are being implemented in GIT provider not Databricks Repos.

See below diagram to understand the role of Databricks Repos and Git provider plays when building a CI/CD workdlow.
All the steps highlighted in yellow can be done Databricks Repo, all the steps highlighted in Gray are done in a git provider like Github or Azure Devops.


Admin workflow
Set up top-level Repos folders (for example, Production)
Set up Git automation to update Repos on merge
User workflow in Databricks
Clone remote repository to user folder
Create new branch based on main branch
Create and edit code
Commit and push to feature branch
Merge workflow in Git provider
Pull request and review process
Merge into main branch
Git automation calls Databricks Repos API
Production job workflow in Databricks
API call brings Repo in Production folder to the latest version
Run Databricks job based on Repo in Production folder
Steps in Databricks
Steps in your Git provider

Question 6:
You have noticed the Data scientist team is using the notebook versioning feature with git integration, you have recommended them to switch to using Databricks Repos, which of the below reasons could be the reason the why the team needs to switch to Databricks Repos.

○ Databricks Repos allows multiple users to make changes

○ Databricks Repos allows merge and conflict resolution

○ Databricks Repos has a built-in version control system

○ Databricks Repos automatically saves changes

○ Databricks Repos allow you to add comments and select the changes you want to commit.

Explanation
The answer is Databricks Repos allow you to add comments and select the changes you want to commit.

Question 7:
Data science team members are using a single cluster to perform data analysis, although cluster size was chosen to handle multiple users and auto-scaling was enabled, the team realized queries are still running slow, what would be the suggested fix for this?

○ Setup multiple clusters so each team member has their own cluster

○ Disable the auto-scaling feature

○ Use High concurrency mode instead of the standard mode

○ Increase the size of the driver node

Explanation
The answer is Use High concurrency mode instead of the standard mode,
https://docs.databricks.com/clusters/cluster-config-best-practices.html#cluster-mode

High Concurrency clusters are ideal for groups of users who need to share resources or run ad-hoc jobs. Administrators usually create High Concurrency clusters. Databricks recommends enabling autoscaling for High Concurrency clusters.

Question 8:
Which of the following SQL commands are used to append rows to an existing delta table?

○ APPEND INTO DELTA table_name

○ APPEND INTO table_name

○ COPY DELTA INTO table_name

○ INSERT INTO table_name

○ UPDATE table_name

Explanation
The answer is INSERT INTO table_name

Insert adds rows to existing table

Question 9:
How are Delt tables stored?

A Directory where parquet data files are stored, a sub directory _delta_log where meta data, and the transaction log is stored as JSON files.

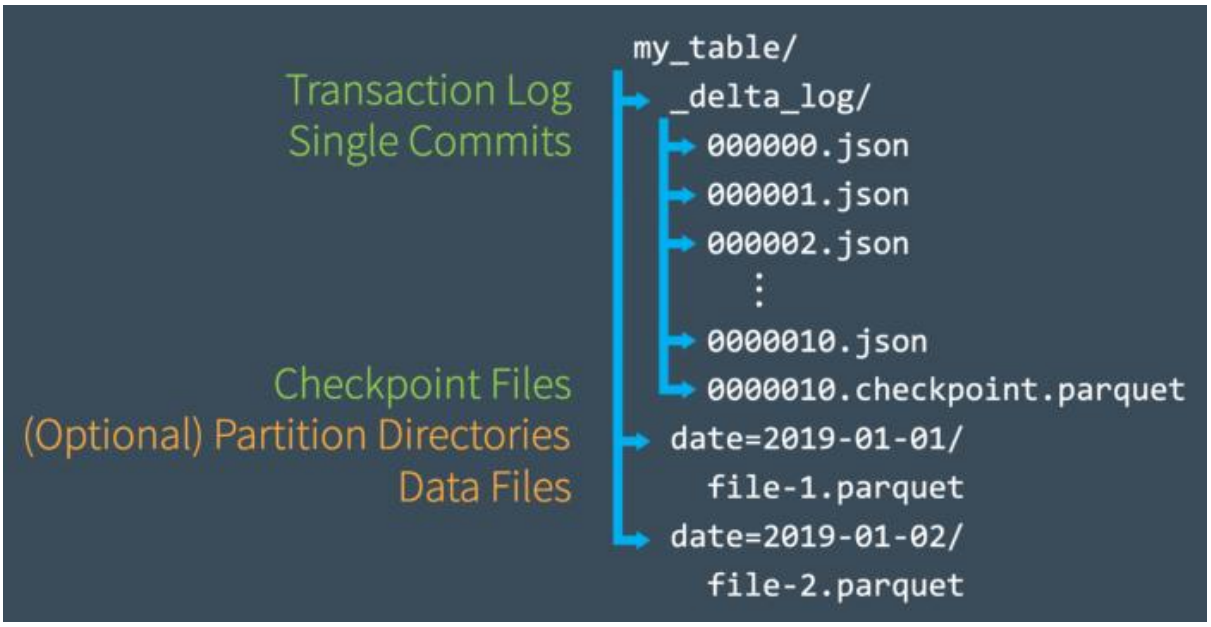
A Directory where parquet data files are stored, all of the meta data is stored in memory

A Directory where parquet data files are stored in Data plane, a sub directory _delta_log where meta data, history and log is stored in control pane.

A Directory where parquet data files are stored, all of the metadata is stored in parquet files

Data is stored in Data plane and Metadata and delta log are stored in control pane

Explanation
The answer is A Directory where parquet data files are stored, a sub directory _delta_log where meta data, and the transaction log is stored as JSON files.



Question 10:
While investigating a data issue in a Delta table, you wanted to review when and who updated the table, what is the best way to review this data?

○ Review event logs in the Workspace

○ Run SQL SHOW HISTORY table_name

○ Check Databricks SQL Audit logs

○ Run SQL command DESCRIBE HISTORY table_name

○ Review workspace audit logs

Explanation

The answer is Run SQL command DESCRIBE HISTORY table_name.
here is the sample data of how DESCRIBE HISTORY table_name looks

```
+-------+-------------------+------+--------+---------+--------------------+----+-
-------+---------+-----------+-------------+-------------+--------------------+
|version|          timestamp|userId|userName|operation| operationParameters|
job|notebook|clusterId|readVersion|isolationLevel|isBlindAppend|
operationMetrics|
+-------+-------------------+------+--------+---------+--------------------+----+-
-------+---------+-----------+-------------+-------------+--------------------+
|      5|2019-07-29 14:07:47|  null|    null|   DELETE|[predicate -> ["(...|null|
null|     null|          4|  Serializable|        false|[numTotalRows -> ...|
|      4|2019-07-29 14:07:41|  null|    null|   UPDATE|[predicate -> (id...|null|
null|     null|          3|  Serializable|        false|[numTotalRows -> ...|
|      3|2019-07-29 14:07:29|  null|    null|   DELETE|[predicate -> ["(...|null|
null|     null|          2|  Serializable|        false|[numTotalRows -> ...|
|      2|2019-07-29 14:06:56|  null|    null|   UPDATE|[predicate -> (id...|null|
null|     null|          1|  Serializable|        false|[numTotalRows -> ...|
|      1|2019-07-29 14:04:31|  null|    null|   DELETE|[predicate -> ["(...|null|
null|     null|          0|  Serializable|        false|[numTotalRows -> ...|
|      0|2019-07-29 14:01:40|  null|    null|    WRITE|[mode -> ErrorIfE...|null|
null|     null|       null|  Serializable|         true|[numFiles -> 2, n...|
+-------+-------------------+------+--------+---------+--------------------+----+-
-------+---------+-----------+-------------+-------------+--------------------+
```

Question 11:
While investigating a performance issue, you realized that you have too many small files for a given table, which command are you going to run to fix this issue

○ COMPACT table_name

○ VACUUM table_name

○ MERGE table_name

○ SHRINK table_name

○ OPTIMIZE table_name

Explanation
The answer is OPTIMIZE table_name,

Optimize compacts small parquet files into a bigger file, by default the size of the files are determined based on the table size at the time of OPTIMIZE, the file size can also be set manually or adjusted based on the workload.

https://docs.databricks.com/delta/optimizations/file-mgmt.html

Tune file size based on Table size

To minimize the need for manual tuning, Databricks automatically tunes the file size of Delta tables based on the size of the table. Databricks will use smaller file sizes for smaller tables and larger file sizes for larger tables so that the number of files in the table does not grow too large.

Table size	Target file size	Approximate number of files in table
10 GB	256 MB	40
1 TB	256 MB	4096
2.56 TB	256 MB	10240
3 TB	307 MB	12108
5 TB	512 MB	17339
7 TB	716 MB	20784
10 TB	1 GB	24437
20 TB	1 GB	34437
50 TB	1 GB	64437
100 TB	1 GB	114437

Question 12:
Create a sales database using the DBFS location `'dbfs:/mnt/delta/databases/sales.db/'`

○
`CREATE DATABASE sales FORMAT DELTA LOCATION 'dbfs:/mnt/delta/databases/sales.db/'’`

○
`CREATE DATABASE sales USING LOCATION 'dbfs:/mnt/delta/databases/sales.db/'`

○
`CREATE DATABASE sales LOCATION 'dbfs:/mnt/delta/databases/sales.db/'`

○
The sales database can only be created in Delta lake

○
`CREATE DELTA DATABASE sales LOCATION 'dbfs:/mnt/delta/databases/sales.db/'`

Explanation
The answer is
`CREATE DATABASE sales LOCATION 'dbfs:/mnt/delta/databases/sales.db/'`
Note: with the introduction of the Unity catalog and three-layer namespace usage of `SCHEMA` and `DATABASE` is interchangeable

Question 13:
What is the type of table created when you issue SQL DDL command `CREATE TABLE sales (id int, units int)`

○ Query fails due to missing location

○ Query fails due to missing format

○ Managed Delta table

○ External Table

○ Managed Parquet table

Explanation
Answer is Managed Delta table
Anytime a table is created without the Location keyword it is considered a managed table, by default all managed tables DELTA tables
Syntax

```
CREATE TABLE table_name ( column column_data_type…)
```

Question 14:
How to determine if a table is a managed table vs external table?

○

Run `IS_MANAGED(‘table_name’)` function

○

All external tables are stored in data lake, managed tables are stored in DELTA lake

○

All managed tables are stored in unity catalog

○

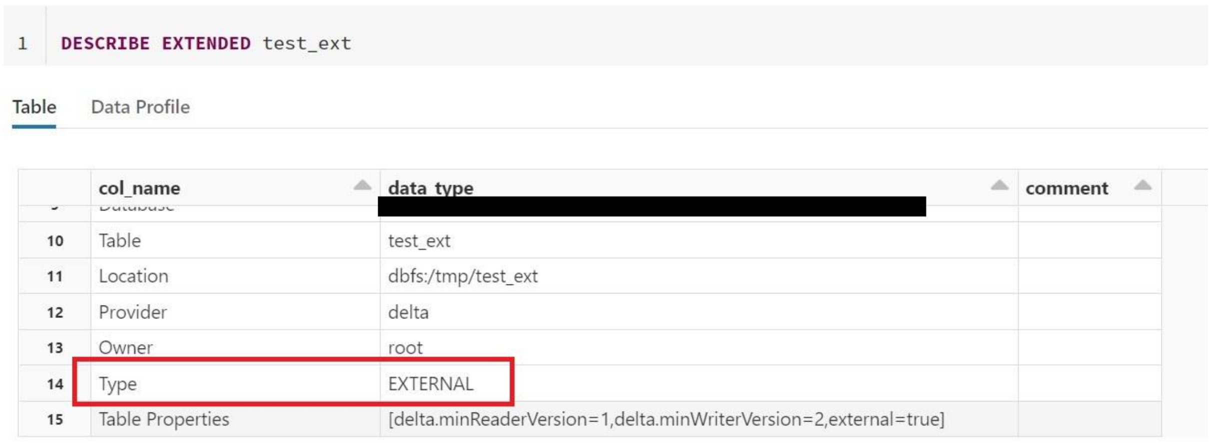
Run SQL command `DESCRIBE EXTENDED table_name` and check type

○

Run SQL command `SHOW TABLES` to see the type of the table

Explanation
The answer is Run SQL command `DESCRIBE EXTENDED table_name` and check type

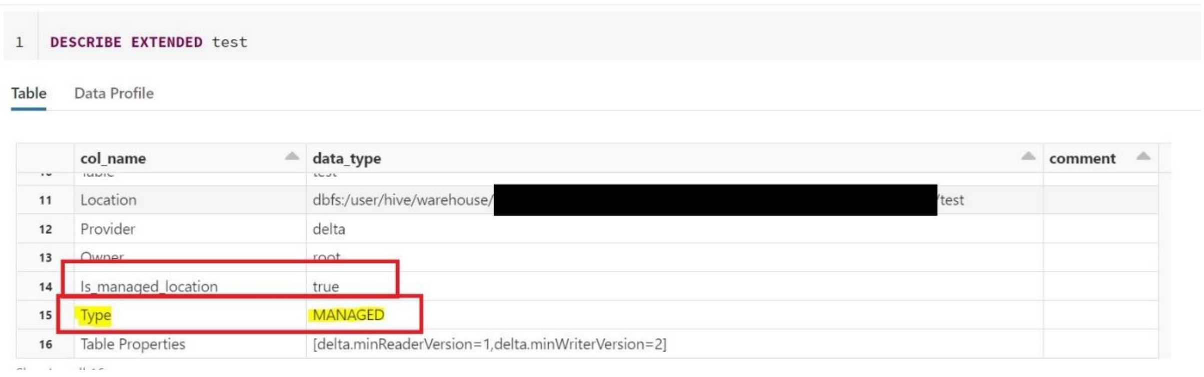
Example of External table



```sql
DESCRIBE EXTENDED test_ext
```

	col_name	data_type	comment
10	Table	test_ext	
11	Location	dbfs:/tmp/test_ext	
12	Provider	delta	
13	Owner	root	
14	Type	EXTERNAL	
15	Table Properties	[delta.minReaderVersion=1,delta.minWriterVersion=2,external=true]	

Example of managed table



```sql
DESCRIBE EXTENDED test
```

	col_name	data_type	comment
11	Location	dbfs:/user/hive/warehouse/[redacted]/test	
12	Provider	delta	
13	Owner	root	
14	Is_managed_location	true	
15	Type	MANAGED	
16	Table Properties	[delta.minReaderVersion=1,delta.minWriterVersion=2]	

Question 15:
Which of the below SQL commands creates a session scoped temporary view?

○
```
CREATE OR REPLACE TEMPORARY VIEW view_name
AS SELECT * FROM table_name
```

○
```
CREATE OR REPLACE LOCAL TEMPORARY VIEW view_name
AS SELECT * FROM table_name
```

○
```
CREATE OR REPLACE GLOBAL TEMPORARY VIEW view_name
AS SELECT * FROM table_name
```

○
```
CREATE OR REPLACE VIEW view_name
AS SELECT * FROM table_name
```

○
```
CREATE OR REPLACE LOCAL VIEW view_name
AS SELECT * FROM table_name
```

Explanation

The answer is

```
CREATE OR REPLACE TEMPORARY VIEW view_name
AS SELECT * FROM table_name
```

The default temporary view is session scoped, as soon as the session ends or if a notebook is detached session scoped temporary view is dropped.

Question 16:
Drop the customers database and associated tables and data, all of the tables inside the database are managed tables.

○ `DROP DATABASE customers FORCE`

○ `DROP DATABASE customers CASCADE`

○ `DROP DATABASE customers  INCLUDE`

○ All the tables must be dropped first before dropping database

○ `DROP DELTA DATABSE customers`

Explanation

The answer is `DROP DATABASE customers CASCADE`

Drop database with cascade option drops all the tables, since all of the tables inside the database are managed tables we do not need to perform any additional steps to clean the data in the storage.

Question 17:
Define an external SQL table by connecting to a local instance of an SQLite database using JDBC

○
```
CREATE TABLE users_jdbc
USING SQLITE
OPTIONS (
    url = "jdbc:/sqmple_db",
    dbtable = "users"
)
```
○
```
CREATE TABLE users_jdbc
USING SQL
URL = {server:"jdbc:/sqmple_db",dbtable: “users”}
```
○
```
CREATE TABLE users_jdbc
USING SQL
OPTIONS (
    url = "jdbc:sqlite:/sqmple_db",
    dbtable = "users"
)
```
○
```
CREATE TABLE users_jdbc
USING org.apache.spark.sql.jdbc.sqlite
OPTIONS (
    url = "jdbc:/sqmple_db",
    dbtable = "users"
)
```

○
```
CREATE TABLE users_jdbc
USING org.apache.spark.sql.jdbc
OPTIONS (
    url = "jdbc:sqlite:/sqmple_db",
    dbtable = "users"
)
```

The answer is,
```
CREATE TABLE users_jdbc
USING org.apache.spark.sql.jdbc
OPTIONS (
    url = "jdbc:sqlite:/sqmple_db",
    dbtable = "users"
)
```

Databricks runtime currently supports connecting to a few flavors of SQL Database including SQL Server, My SQL, SQL Lite and Snowflake using JDBC.

```
CREATE TABLE <jdbcTable>
USING org.apache.spark.sql.jdbc or JDBC
OPTIONS (
    url = "jdbc:<databaseServerType>://<jdbcHostname>:<jdbcPort>",
    dbtable " = <jdbcDatabase>.atable",
    user = "<jdbcUsername>",
    password = "<jdbcPassword>"
)
```

More detailed documentation

SQL databases using JDBC - Azure Databricks | Microsoft Docs

Question 18:
When defining external tables using formats CSV, JSON, TEXT, BINARY any query on the external tables caches the data and location for performance reasons, so within a given spark session any new files that may have arrived will not be available after the initial query. How can we address this limitation?

○ `UNCACHE TABLE table_name`

○ `CACHE TABLE table_name`

○ `REFRESH TABLE table_name`

○ `BROADCAST TABLE table_name`

○ `CLEAR CACH table_name`

Explanation

The answer is `REFRESH TABLE table_name`

`REFRESH TABLE table_name` will force Spark to refresh the availability of external files and any changes.

Question 19:
Which of the following table constraints are supported in Delta tables?

○ Primary key, foreign key, Not Null, Check Constraints

○ Primary key, Not Null, Check Constraints

○ Default, Not Null, Check Constraints

○ Not Null, Check Constraints

○ Unique, Not Null, Check Constraints

Explanation
The answer is Not Null, Check Constraints
https://docs.microsoft.com/en-us/azure/databricks/delta/delta-constraints

```
CREATE TABLE events( id LONG,
                     date STRING,
                    location STRING,
                    description STRING
                    ) USING DELTA;
ALTER TABLE events CHANGE COLUMN id SET NOT NULL;
ALTER TABLE events ADD CONSTRAINT dateWithinRange CHECK (date > '1900-01-
01');
```

Question 20:
The data engineering team is looking to add a new column to the table, but the QA team would like to test the change which of the below options allow you to quickly copy the table from Prod to QA environment, modify and run the tests

○ DEEP CLONE

○ SHADOW CLONE

○ ZERO COPY CLONE

○ SHALLOW CLONE

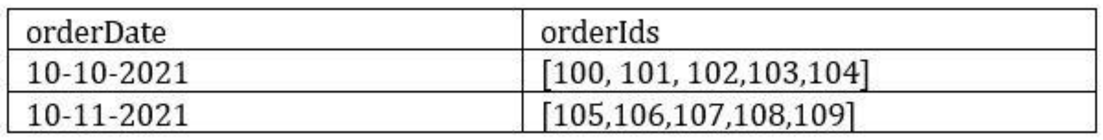
○ METADATA CLONE

Explanation
The answer is SHALLOW CLONE
`SHALLOW CLONE` If you wish to create a copy of a table quickly to test out applying changes without the risk of modifying the current table, `SHALLOW CLONE` can be a good option. Shallow clones just copy the Delta transaction logs, meaning that the data doesn't move so it can be very quick.
`DEEP CLONE` fully copies data and metadata from a source table to a target. This copy occurs incrementally, so executing this command again can sync changes from the source to the target location. It copies all of the data and transaction logs this can take a long time based on the size of the table.

Question 21:
Sales team is looking to get a report on a measure number of units sold by date, below is the schema. Fill in the blank with the appropriate array function.

Table orders: `orderDate DATE, orderIds ARRAY<INT>`

Table orderDetail: `orderId INT, unitsSold INT, salesAmt DOUBLE`



orderDate	orderIds
10-10-2021	[100, 101, 102,103,104]
10-11-2021	[105,106,107,108,109]

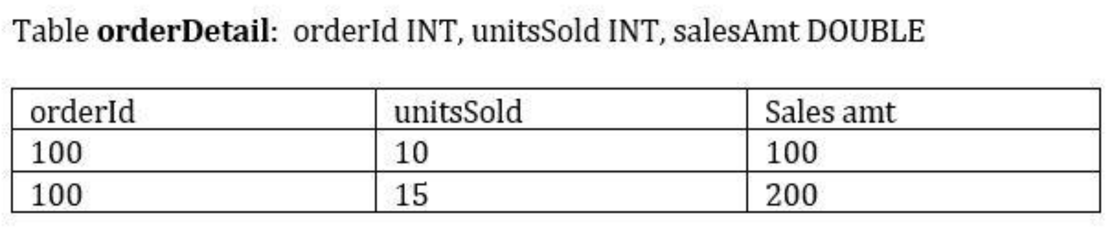


Table **orderDetail:** orderId INT, unitsSold INT, salesAmt DOUBLE

orderId	unitsSold	Sales amt
100	10	100
100	15	200

```
SELECT orderDate, SUM(unitsSold)
      FROM orderDetail od
JOIN (select orderDate, ___________(orderIds) as orderId FROM orders) o
    ON o.orderId = od.orderId
GROUP BY orderDate
```

○ FLATTEN

○ EXTEND

○ EXPLODE

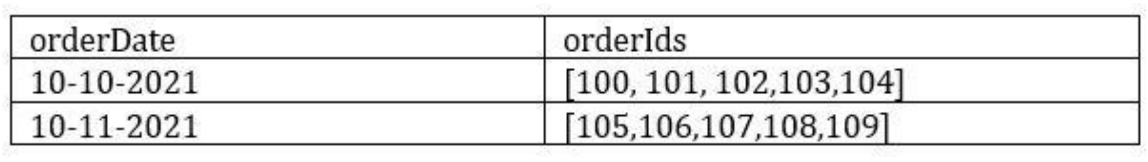
○ EXTRACT

○ ARRAY_FLATTEN

Explanation
The answer is EXPLODE,

explode is table-valued function, takes an array or map and returns a row for each element in the array.

For below table



orderDate	orderIds
10-10-2021	[100, 101, 102,103,104]
10-11-2021	[105,106,107,108,109]

```
select explode(orderIds) orderId, orderdate from orders
```

This above would return a result something like below, creates a row for each element in the array

OrderDate	orderId
10-10-2021	100
10-10-2021	101
10-10-2021	102

…
..

Question 22:
You are asked to write a python function that can read data from a delta table and return the DataFrame, which of the following is correct?

○ Python function cannot return a DataFrame

○ Write SQL UDF to return a DataFrame

○ Write SQL UDF that can return tabular data

○ Python function will result in out of memory error due to data volume

○ Python function can return a DataFrame

Explanation
The answer is Python function can return a DataFrame
The function would something like this,

```
get_source_dataframe(tablename):
 df = spark.read.table(tablename)
return df
```

df = get_source_dataframe('test_table')
since there is no action spark returns a Dataframe and assigns to df python variable

Question 23:
What is the output of the below function when executed with input parameters 1, 3 :

```
def check_input(x,y):
    if x < y:
        x= x+1
        if x<y:
            x= x+1
            if x <y:
                x = x+1
     return x
```

check_input(1,3)

○ 1

○ 2

○ 3

○ 4

○ 5

Explanation
The answer is 3

Question 24:
Which of the following SQL statements can replace a python variable, when the notebook is set in SQL mode

```
table_name = "sales"
schema_name = "bronze"
```

○
```
spark.sql(f"SELECT * FROM f{schema_name.table_name}")
```
○
```
spark.sql(f"SELECT * FROM {schem_name.table_name}")
```
○
```
spark.sql(f"SELECT * FROM ${schema_name}.${table_name}")
```

○
```
spark.sql(f"SELECT * FROM {schema_name}.{table_name}")
```

○
```
spark.sql("SELECT * FROM schema_name.table_name")
```

Explanation

The answer is `spark.sql(f"SELECT * FROM {schema_name}.{table_name}")`

Question 25:
When writing streaming data, Spark’s structured stream supports the below write modes

○ Append, Delta, Complete

○ Delta, Complete, Continuous

○ Append, Complete, Update

○ Complete, Incremental, Update

○ Append, overwrite, Continuous

Explanation
The answer is Append, Complete, Update

Append mode (default) - This is the default mode, where only the new rows added to the Result Table since the last trigger will be outputted to the sink. This is supported for only those queries where rows added to the Result Table is never going to change. Hence, this mode guarantees that each row will be output only once (assuming fault-tolerant sink). For example, queries with only `select`, `where`, `map`, `flatMap`, `filter`, `join`, etc. will support Append mode.
Complete mode - The whole Result Table will be outputted to the sink after every trigger. This is supported for aggregation queries.
Update mode - (*Available since Spark 2.1.1*) Only the rows in the Result Table that were updated since the last trigger will be outputted to the sink. More information to be added in future releases.

Question 26:
When using the complete mode to write stream data, how does it impact the target table?

○ Entire stream waits for complete data to write

○ Stream must complete to write the data

○ Target table cannot be updated while stream is pending

○ Target table is overwritten for each batch

○ Delta commits transaction once the stream is stopped

Explanation
The answer is Target table is overwritten for each batch

Complete mode - The whole Result Table will be outputted to the sink after every trigger. This is supported for aggregation queries

Question 27:
At the end of the inventory process a file gets uploaded to the cloud object storage, you are asked to build a process to ingest data which of the following method can be used to ingest the data incrementally, the schema of the file is expected to change overtime ingestion process should be able to handle these changes automatically. Below is the auto loader command to load the data, fill in the blanks for successful execution of the below code.

```
spark.readStream
.format("cloudfiles")
.option("cloudfiles.format",”csv)
.option("_______", ‘dbfs:/location/checkpoint/’)
.load(data_source)
.writeStream
.option("_______",’ dbfs:/location/checkpoint/’)
.option("mergeSchema", "true")
.table(table_name))
```

○ checkpointlocation, schemalocation

○ checkpointlocation, cloudfiles.schemalocation

○ schemalocation, checkpointlocation

○ cloudfiles.schemalocation, checkpointlocation

○ cloudfiles.schemalocation, cloudfiles.checkpointlocation

Explanation

The answer is `cloudfiles.schemalocation, checkpointlocation`

When reading the data `cloudfiles.schemalocation` is used to store the inferred schema of the incoming data.

When writing a stream to recover from failures `checkpointlocation` is used to store the offset of the byte that was most recently processed.

Question 28:
When working with AUTO LOADER you noticed that most of the columns that were inferred as part of loading are string data types including columns that were supposed to be integers, how can we fix this?

○ Provide the schema of the source table in the cloudfiles.schemalocation

○ Provide the schema of the target table in the cloudfiles.schemalocation

○ Provide schema hints

○ Update the checkpoint location

○ Correct the incoming data by explicitly casting the data types

Explanation

The answer is, Provide schema hints.

```
spark.readStream \
  .format("cloudFiles") \
  .option("cloudFiles.format", "csv") \
  .option("header", "true") \
  .option("cloudFiles.schemaLocation", schema_location) \
  .option("cloudFiles.schemaHints", "id int, description string")
  .load(raw_data_location)
  .writeStream \
  .option("checkpointLocation", checkpoint_location) \
  .start(target_delta_table_location)
```

.option("cloudFiles.schemaHints", "id int, description string")
# Here we are providing a hint that id column is int and the description is a string
When cloudfiles.schemalocation is used to store the output of the schema inference during the load process, with schema hints you can enforce data types for known columns ahead of time.

Question 29:
You have configured AUTO LOADER to process incoming IOT data from cloud object storage every 15 mins, recently a change was made to the notebook code to update the processing logic but the team later realized that the notebook was failing for the last 24 hours, what steps team needs to take to reprocess the data that was not loaded after the notebook was corrected?

○

Move the files that were not processed to another location and manually copy the files into the ingestion path to reprocess them

○

Enable back_fill = TRUE to reprocess the data

○

Delete the checkpoint folder and run the autoloader again

○

Autoloader automatically re-processes data that was not loaded

○

Manually re-load the data

Explanation

The answer is,

Autoloader automatically re-processes data that was not loaded using the checkpoint.

Question 30:
Which of the following Structured Streaming queries is performing a hop from a bronze table to a Silver table?

○

```
(spark.table("sales").groupBy("store")
.agg(sum("sales")).writeStream
.option("checkpointLocation",checkpointPath)
.outputMode("complete")
.table("aggregatedSales"))
```

○

```
(spark.table("sales").agg(sum("sales"),sum("units"))
.writeStream
.option("checkpointLocation",checkpointPath)
.outputMode("complete")
.table("aggregatedSales"))
```

○

```
(spark.table("sales")
.withColumn("avgPrice", col("sales") / col("units"))
.writeStream
.option("checkpointLocation", checkpointPath)
.outputMode("append")
.table("cleanedSales"))
```

○

```
(spark.readStream.load(rawSalesLocation)
.writeStream
.option("checkpointLocation", checkpointPath)
.outputMode("append")
.table("uncleanedSales") )
```

○

```
(spark.read.load(rawSalesLocation)
.writeStream
.option("checkpointLocation", checkpointPath)
.outputMode("append")
.table("uncleanedSales") )
```

Explanation

The question is asking to identify a structured streaming command that is moving data from bronze to silver.

The answer is

```
(spark.table("sales")
.withColumn("avgPrice", col("sales") / col("units"))
.writeStream
.option("checkpointLocation", checkpointPath)
.outputMode("append")
.table("cleanedSales"))
```

We are preserving the grain of incoming data and enriching the data by adding avg price, the other options listed use aggregations which are mostly performed on top of the silver to move data to Gold.

Question 31:
Which of the following Structured Streaming queries successfully performs a hop from a Silver to Gold table?

○
```
(spark.table("sales")
.groupBy("store")
.agg(sum("sales"))
.writeStream
.option("checkpointLocation", checkpointPath)
.outputMode("complete")
.table("aggregatedSales") )
```

○
```
(spark.table("sales")
.writeStream
.option("checkpointLocation", checkpointPath)
.outputMode("complete")
.table("sales") )
```

○
```
(spark.table("sales")
.withColumn("avgPrice", col("sales") / col("units"))
.writeStream
.option("checkpointLocation", checkpointPath)
.outputMode("append")
.table("cleanedSales") )
```

○
```
(spark.readStream.load(rawSalesLocation)
.writeStream
.option("checkpointLocation", checkpointPath)
.outputMode("append")
.table("uncleanedSales") )
```

○
```
(spark.read.load(rawSalesLocation)
    .writeStream
    .option("checkpointLocation", checkpointPath)
    .outputMode("append")
    .table("uncleanedSales") )
```

Explanation

The answer is

```
(spark.table("sales")
.groupBy("store")
.agg(sum("sales"))
.writeStream
.option("checkpointLocation", checkpointPath)
.outputMode("complete")
.table("aggregatedSales") )
```

The gold layer is normally used to store aggregated data

Review the below link for more info,

Medallion Architecture – Databricks

Gold Layer:

Powers Ml applications, reporting, dashboards, ad hoc analytics

Refined views of data, typically with aggregations

Reduces strain on production systems

Optimizes query performance for business-critical data

Question 32:
Which of the following Auto loader structured streaming commands successfully performs a hop from the landing area into Bronze?

◯
```
spark\
.readStream\
.format("csv")\
.option("cloudFiles.schemaLocation", checkpoint_directory)\
.load("landing")\
.writeStream.option("checkpointLocation", checkpoint_directory)\
.table(raw)
```

◯
```
spark\
.readStream\
.format("cloudFiles")\
.option("cloudFiles.format","csv")\
.option("cloudFiles.schemaLocation", checkpoint_directory)\
.load("landing")\
.writeStream.option("checkpointLocation", checkpoint_directory)\
.table(raw)
```

◯
```
spark\
.read\
.format("cloudFiles")\
.option("cloudFiles.format",”csv”)\
.option("cloudFiles.schemaLocation", checkpoint_directory)\
.load("landing")\
.writeStream.option("checkpointLocation", checkpoint_directory)\
.table(raw)
```

◯
```
spark\
.readStream\
.load(rawSalesLocation)\
.writeStream \
.option("checkpointLocation", checkpointPath).outputMode("append")\
.table("uncleanedSales")
```

◯
```
spark\
.read\
.load(rawSalesLocation) \
.writeStream\
.option("checkpointLocation", checkpointPath) \
.outputMode("append")\
.table("uncleanedSales")
```

Explanation
The answer is

```
spark\
.readStream\
.format("cloudFiles") \# use Auto loader
.option("cloudFiles.format","csv") \ # csv format files
.option("cloudFiles.schemaLocation", checkpoint_directory)\
.load('landing')\
.writeStream.option("checkpointLocation", checkpoint_directory)\
.table(raw)
```

Note: if you chose the below option which is incorrect because it does not have readStream

```
spark.read.format("cloudFiles")
.option("cloudFiles.format",”csv”)
...
..
..
```

Question 33:
What are two different modes of DELTA LIVE TABLE Pipelines

○ Triggered, Incremental

○ Once, Continuous

○ Triggered, Continuous

○ Once, Incremental

○ Continuous, Incremental

Explanation
The answer is Triggered, Continuous
https://docs.microsoft.com/en-us/azure/databricks/data-engineering/delta-live-tables/delta-live-tables-concepts#--continuous-and-triggered-pipelines

Triggered pipelines update each table with whatever data is currently available and then stop the cluster running the pipeline. Delta Live Tables automatically analyzes the dependencies between your tables and starts by computing those that read from external sources. Tables within the pipeline are updated after their dependent data sources have been updated.
Continuous pipelines update tables continuously as input data changes. Once an update is started, it continues to run until manually stopped. Continuous pipelines require an always-running cluster but ensure that downstream consumers have the most up-to-date data.

Question 34:
Your team member is trying to set up a delta pipeline and build a second gold table to the same pipeline with aggregated metrics based on an existing Delta Live table called sales_orders_cleaned but he is facing a problem in starting the pipeline, the pipeline is failing to state it cannot find the table sales_orders_cleaned, you are asked to identify and fix the problem.

```
CREATE LIVE TABLE sales_order_in_chicago
AS
SELECT order_date, city, sum(price) as sales,
FROM sales_orders_cleaned
WHERE city = 'Chicago')
GROUP BY order_date, city
```

○ Use STREAMING LIVE instead of LIVE table

○ Delta live table can be used in a group by clause

○ Delta live tables pipeline can only have one table

○ Sales_orders_cleaned table is missing schema name LIVE

○ The pipeline needs to be deployed so the first table is created before we add a second table

Explanation
The answer is, Sales_orders_cleaned table is missing schema name LIVE
Every Delta live table should have schema LIVE
Here is the correct syntax,

```
CREATE LIVE TABLE sales_order_in_chicago
AS
SELECT order_date, city, sum(price) as sales,
FROM LIVE.sales_orders_cleaned
WHERE city = 'Chicago')
GROUP BY order_date, city
```

Question 35: Incorrect
Which of the following type of tasks cannot setup through a job?

○ Notebook

○ DELTA LIVE PIPELINE

○
Spark Submit
○
Python
○
Databricks SQL Dashboard refresh

Question 36:
Which of the following approaches can the data engineer use to obtain a version-controllable configuration of the Job's schedule and configuration?

○ They can link the Job to notebooks that are a part of a Databricks Repo.

○ They can submit the Job once on a Job cluster.

○ They can download the JSON equivalent of the job from the Job's page.

○ They can submit the Job once on an all-purpose cluster.

○ They can download the XML description of the Job from the Job's page

Question 37:
Skipped
what steps need to be taken to set up a DELTA LIVE PIPELINE as a job?

○ DELTA LIVE TABLES do not support job cluster

○ Select Workflows UI and Delta live tables tab, under task type select Delta live tables pipeline and select the notebook

○ Select Workflows UI and Delta live tables tab, under task type select Delta live tables pipeline and select the pipeline JSON file

○ Use Pipeline creation UI, select a new pipeline and job cluster

The answer is,
Select Workflows UI and Delta live tables tab, under task type select Delta live tables pipeline and select the notebook.

Create a pipeline
To create a new pipeline using the Delta Live Tables notebook:
Click Workflows in the sidebar, click the Delta Live Tables tab, and click Create Pipeline.
Give the pipeline a name and click to select a notebook.
Optionally enter a storage location for output data from the pipeline. The system uses a default location if you leave Storage Location empty.
Select Triggered for Pipeline Mode.
Click Create.
The system displays the Pipeline Details page after you click Create. You can also access your pipeline by clicking the pipeline name in the Delta Live Tables tab.
Question 38:
Data engineering team has provided 10 queries and asked Data Analyst team to build a dashboard and refresh the data every day at 8 AM, identify the best approach to set up data refresh for this dashaboard?

○ Each query requires a separate task and setup 10 tasks under a single job to run at 8 AM to refresh the dashboard

○ The entire dashboard with 10 queries can be refreshed at once, single schedule needs to be set up to refresh at 8 AM.

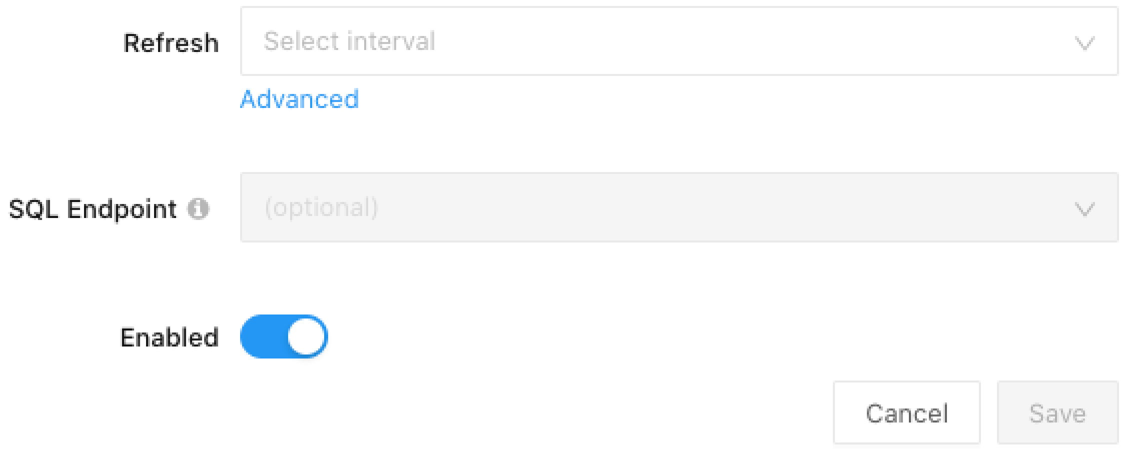
○ Setup JOB with linear dependency to all load all 10 queries into a table so the dashboard can be refreshed at once.

○ A dashboard can only refresh one query at a time, 10 schedules to set up the refresh.

○ Use Incremental refresh to run at 8 AM every day.

The answer is,
The entire dashboard with 10 queries can be refreshed at once, single schedule needs to be set up to refresh at 8 AM.

Automatically refresh a dashboard
A dashboard's owner and users with the Can Edit permission can configure a dashboard to automatically refresh on a schedule. To automatically refresh a dashboard:
Click the Schedule button at the top right of the dashboard. The scheduling dialog appears.



In the Refresh every drop-down, select a period.
In the SQL Warehouse drop-down, optionally select a SQL warehouse to use for all the queries. If you don't select a warehouse, the queries execute on the last used SQL warehouse.
Next to Subscribers, optionally enter a list of email addresses to notify when the dashboard is automatically updated.
Each email address you enter must be associated with a Azure Databricks account or configured as an alert destination.
Click Save. The Schedule button label changes to Scheduled.

Question 39:
The data engineering team is using a SQL query to review data completeness every day to monitor the ETL job, and query output is being used in multiple dashboards which of the following approaches can be used to set up a schedule and automate this process?

○ They can schedule the query to run every day from the Jobs UI.

○ They can schedule the query to refresh every day from the query's page in Databricks SQL

○ They can schedule the query to run every 12 hours from the Jobs UI.

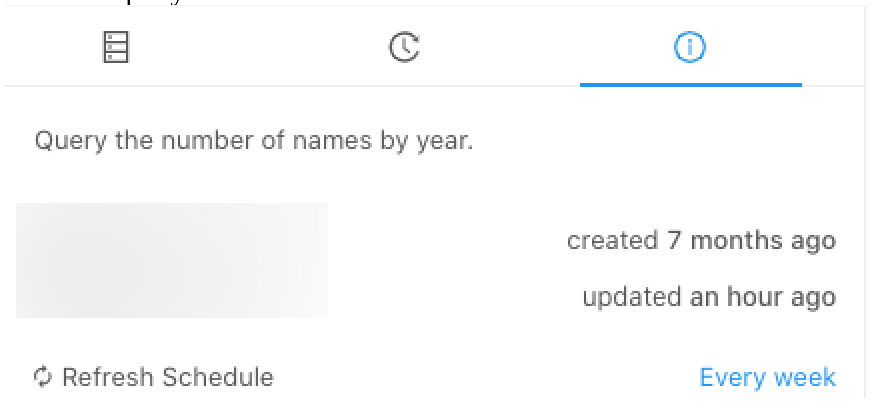
○ They can schedule the query to refresh every day from the SQL endpoint's page in Databricks SQL.
(Incorrect)

○ They can schedule the query to refresh every 12 hours from the SQL endpoint's page in Databricks SQL

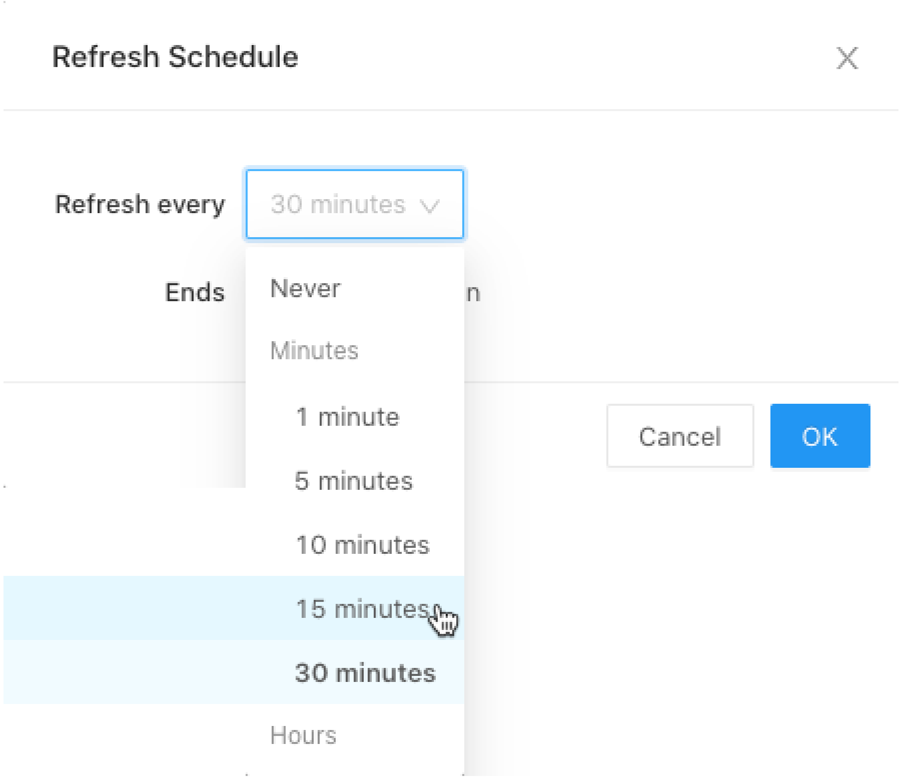
Explanation
The answer is They can schedule the query to refresh every 12 hours from the SQL endpoint's page in Databricks SQL,

The query pane view in Databricks SQL workspace provides the ability to add or edit and schedule individual queries to run.

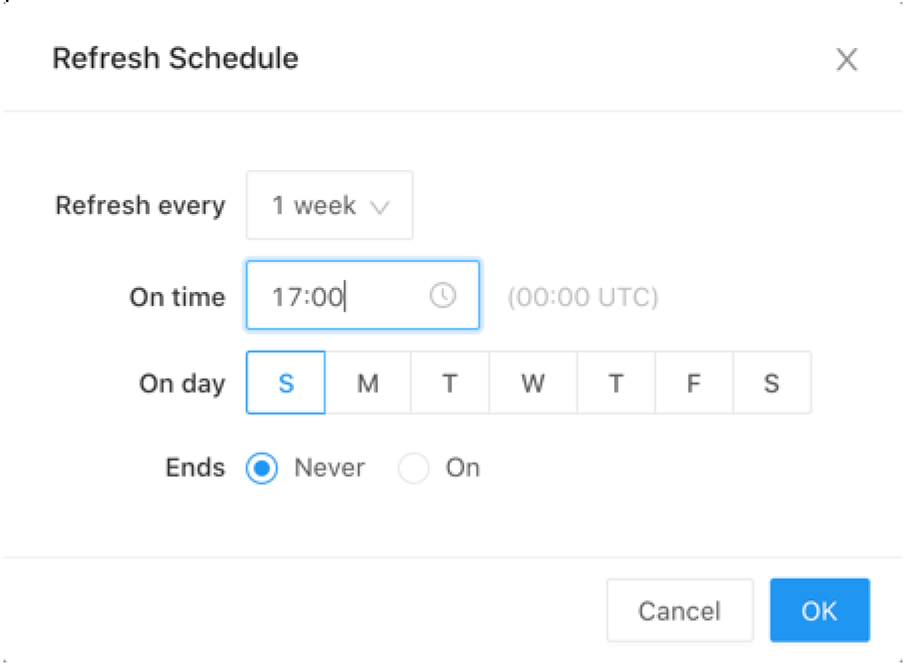
You can use scheduled query executions to keep your dashboards updated or to enable routine alerts. By default, your queries do not have a schedule.
Note
If your query is used by an alert, the alert runs on its own refresh schedule and does not use the query schedule.
To set the schedule:
Click the query info tab.


Query the number of names by year.
created 7 months ago
updated an hour ago
Refresh Schedule Every week

Click the link to the right of Refresh Schedule to open a picker with schedule intervals.



Set the schedule.
The picker scrolls and allows you to choose:
An interval: 1-30 minutes, 1-12 hours, 1 or 30 days, 1 or 2 weeks
A time. The time selector displays in the picker only when the interval is greater than 1 day and the day selection is greater than 1 week. When you schedule a specific time, Databricks SQL takes input in your computer's timezone and converts it to UTC. If you want a query to run at a certain time in UTC, you must adjust the picker by your local offset. For example, if you want a query to execute at `00:00` UTC each day, but your current timezone is PDT (UTC-7), you should select `17:00` in the picker:



Refresh every: 1 week
On time: 17:00 (00:00 UTC)
On day: [✓ S] M T W T F S
Ends: [✓ Never] On

Question 40:
A data engineer is using a Databricks SQL query to monitor the performance of an ELT job. The ELT job is triggered by a specific number of input records being ready to process. The Databricks SQL query returns the number of minutes since the job's most recent runtime. Which of the following approaches can enable the data engineering team to be notified if the ELT job has not been run in an hour?

○ They can set up an Alert for the accompanying dashboard to notify them if the returned value is greater than 60.

○ They can set up an Alert for the query to notify when the ELT job fails.

○ They can set up an Alert for the accompanying dashboard to notify when it has not refreshed in 60 minutes.

○ They can set up an Alert for the query to notify them if the returned value is greater than 60.

○ This type of alert is not possible in Databricks

Explanation
The answer is, They can set up an Alert for the query to notify them if the returned value is greater than 60.

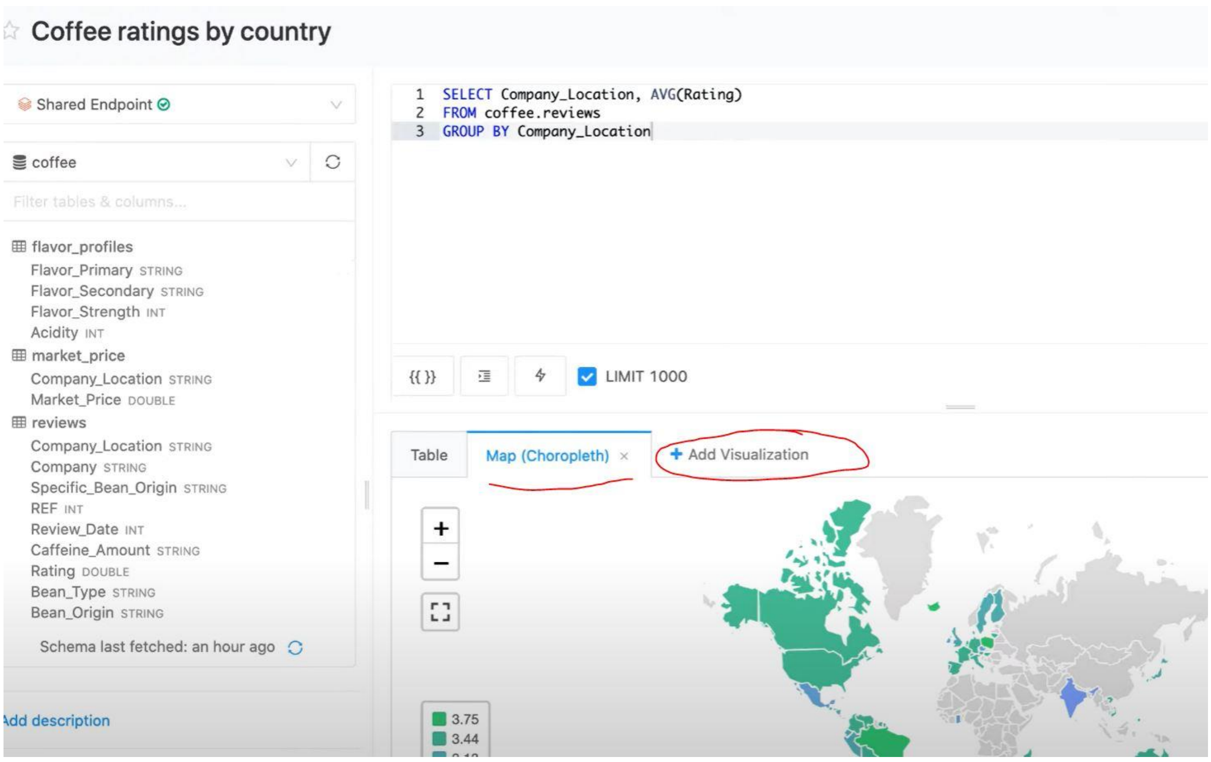
The important thing to note here is that alert can only be setup on query not on the dashboard, query can return a value, which is used if alert can be triggered.

Question 41:
Which of the following is true, when building a Databricks SQL dashboard?

- A dashboard can only use results from one query
- Only one visualization can be developed with one query result
- A dashboard can only connect to one schema/Database
- More than one visualization can be developed using a single query result
- A dashboard can only have one refresh schedule

Explanation
the answer is, More than one visualization can be developed using a single query result.

In the query editor pane + Add visualization tab can be used for many visualizations for a single query result.



Coffee ratings by country

```
SELECT Company_Location, AVG(Rating)
FROM coffee.reviews
GROUP BY Company_Location
```

Question 42:
A newly joined team member John Smith in the Marketing team currently has access read access to sales tables but does not have access to update the table, which of the following commands help you accomplish this?

○
```
GRANT UPDATE ON TABLE table_name TO john.smith@marketing.com
```
○
```
GRANT USAGE ON TABLE table_name TO john.smith@marketing.com
```
○
```
GRANT MODIFY ON TABLE table_name TO john.smith@marketing.com
```

○
```
GRANT UPDATE TO TABLE table_name ON john.smith@marketing.com
```
○
```
GRANT MODIFY TO TABLE table_name ON john.smith@marketing.com
```
Explanation
The answer is GRANT MODIFY ON TABLE table_name TO john.smith@marketing.com
https://docs.microsoft.com/en-us/azure/databricks/security/access-control/table-acls/object-privileges#privileges

Question 43
A new user who currently does not have access to the catalog or schema is requesting access to the customer table in sales schema, but the customer table contains sensitive information, so you have decided to create view on the table excluding columns that are sensitive and granted access to the view GRANT SELECT ON view_name to user@company.com but when the user tries to query the view, gets the error view does not exist. What is the issue preventing user to access the view and how to fix it?

○ User requires SELECT on the underlying table

○ User requires to be put in a special group that has access to PII data

○ User has to be the owner of the view

○ User requires USAGE privilege on Sales schema

○ User needs ADMIN privilege on the view

Explanation
The answer is User requires USAGE privilege on Sales schema,

[Data object privileges - Azure Databricks | Microsoft Docs]()

```
GRANT USAGE ON SCHEMA sales TO user@company.com;
```

`USAGE`: does not give any abilities, but is an additional requirement to perform any action on a schema object.

Question 44

How do you access or use tables in the unity catalog?

○ `schema_name.table_name`

○ `schema_name.catalog_name.table_name`

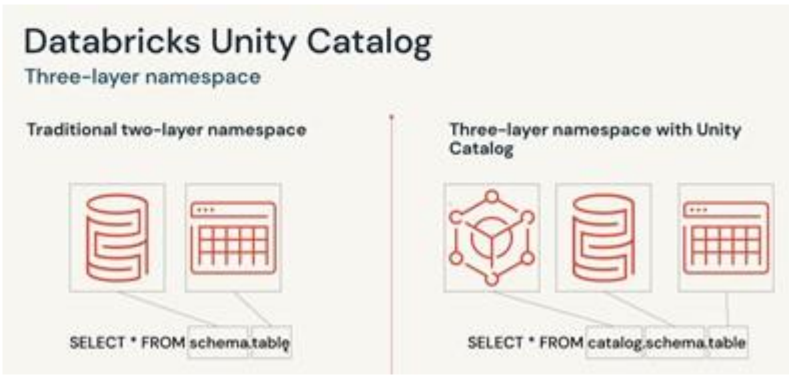
○ `catalog_name.table_name`

○ `catalog_name.database_name.schema_name.table_name`

○ `catalog_name.schema_name.table_name`

Explanation

The answer is catalog_name.schema_name.table_name


Databricks Unity Catalog
Three-layer namespace
Traditional two-layer namespace
SELECT * FROM schema.table
Three-layer namespace with Unity Catalog
SELECT * FROM catalog.schema.table

note: Database and Schema are analogous they are interchangeably used in the Unity catalog.

Question 45:

How do you upgrade an existing workspace managed table to a unity catalog table?

○ `ALTER TABLE table_name SET UNITY_CATALOG = TRUE`

○ `Create table catalog_name.schema_name.table_name`
`as select * from hive_metastore.old_schema.old_table`

○ `Create table table_name as select * from`
`hive_metastore.old_schema.old_table`

○ `Create table table_name format = UNITY as select * from old_table_name`

○ `Create or replace table_name format = UNITY using deep clone`
`old_table_name`

Explanation

The answer is Create table catalog_name.schema_name.table_name as select * from hive_metastore.old_schema.old_table

Basically, we are moving the data from an internal hive metastore to a metastore and catalog that is registered in Unity catalog.

note: if it is a managed table the data is copied to a different storage account, for a large tables this can take a lot of time. For an external table the process is different.
Managed table: Upgrade a managed to Unity Catalog
External table: Upgrade an external table to Unity Catalog
s