Question 1

You were asked to create a table that can store the below data, note that orderDate is the truncated date format of OrderTime, fill in the blank to complete the DDL.

transactionId	transactionDate	unitsSold
1	01-01-2021 09:10:24 AM	100
2	01-01-2022 10:30:30 AM	10

```
CREATE TABLE orders (
    orderId int,
    orderTime timestamp,
    orderdate date ___________________________________________ ,
    units int)
```

○ `AS DEFAULT (CAST(orderTime as DATE))`

○ `GENERATED ALWAYS AS (CAST(orderTime as DATE))`

○ `GENERATED DEFAULT AS (CAST(orderTime as DATE))`

○ `AS (CAST(orderTime as DATE))`

○ Delta lake does not support calculated columns, value should be inserted into the table as part of the ingestion process

Explanation
The answer is, `GENERATED ALWAYS AS (CAST(orderTime as DATE))`
https://docs.microsoft.com/en-us/azure/databricks/delta/delta-batch#--use-generated-columns
Delta Lake supports generated columns which are a special type of columns whose values are automatically generated based on a user-specified function over other columns in the Delta table. When you write to a table with generated columns and you do not explicitly provide values for them, Delta Lake automatically computes the values.
Note: Databricks also supports partitioning using generated column

Question 2:
The data engineering team noticed that one of the job fails randomly as a result of using spot instances, what feature in Jobs/Tasks can be used to address this issue so the job is more stable when using spot instances?

○ Use Databrick REST API to monitor and restart the job

○ Use Jobs runs, active runs UI section to monitor and restart the job

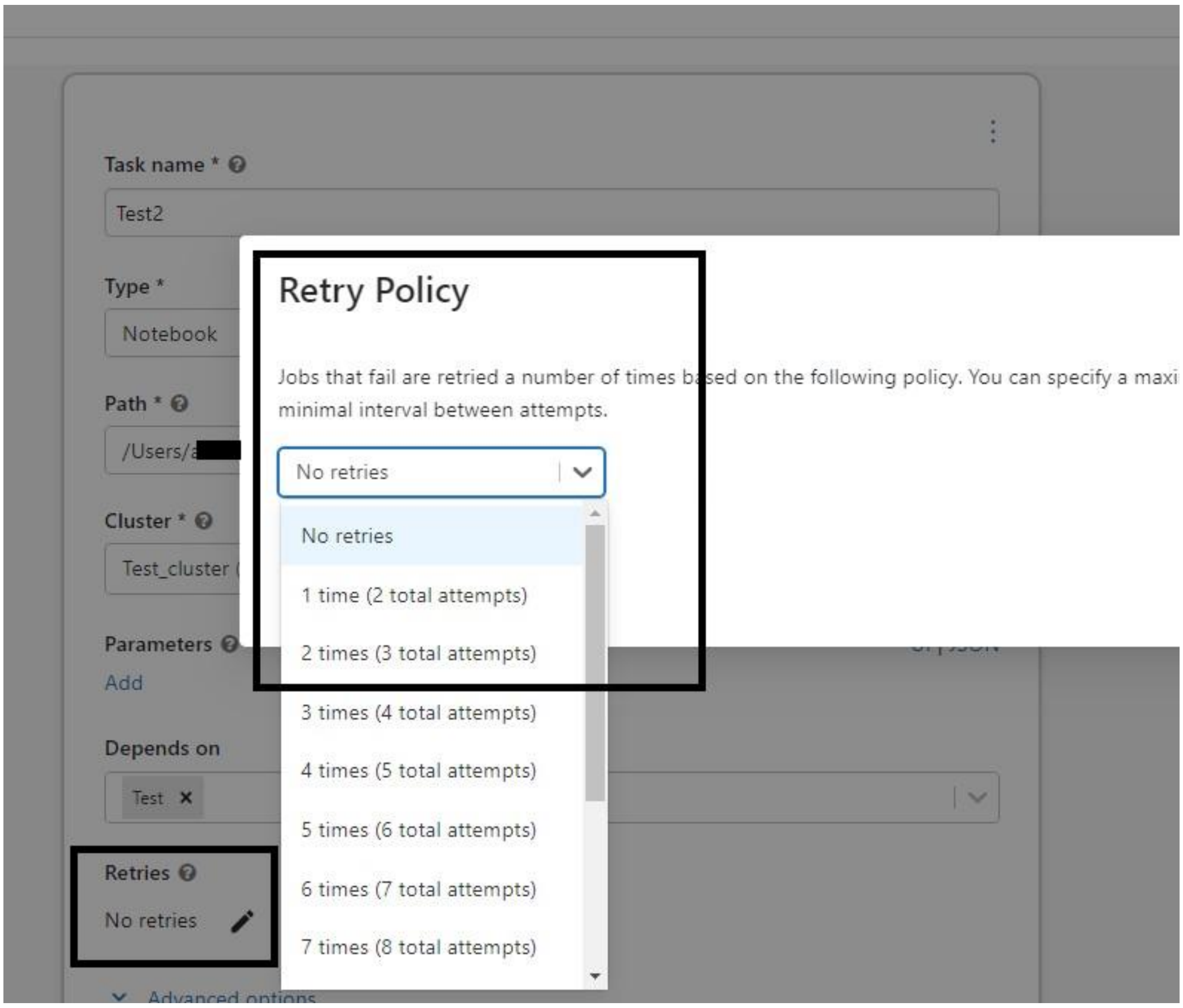
○ Add second task and add a check condition to rerun the first task if it fails

○ Restart the job cluster, job automatically restarts

○ Add a retry policy to the task

Explanation
The answer is, Add a retry policy to the task

Tasks in Jobs support Retry Policy, which can be used to retry a failed tasks, especially when using spot instance it is common to have failed executors or driver.


Task name * Test2

Type * Notebook

Path * /Users/[redacted]

Cluster * Test_cluster (

Parameters Add

Depends on Test

Retries No retries

Advanced options

Retry Policy

Jobs that fail are retried a number of times based on the following policy. You can specify a maxi minimal interval between attempts.

No retries

- No retries
- 1 time (2 total attempts)
- 2 times (3 total attempts)
- 3 times (4 total attempts)
- 4 times (5 total attempts)
- 5 times (6 total attempts)
- 6 times (7 total attempts)
- 7 times (8 total attempts)

Question 3:

What is the main difference between AUTO LOADER and COPY INTO?

○ COPY INTO supports schema evolution.

○ AUTO LOADER supports schema evolution.

○ COPY INTO supports file notification when performing incremental loads.

○ AUTO LOADER supports directory listing when performing incremental loads.

○ AUTO LOADER Supports file notification when performing incremental loads.

Explanation
Auto loader supports both directory listing and file notification but COPY INTO only supports directory listing.

Auto loader file notification will automatically set up a notification service and queue service that subscribe to file events from the input directory in cloud object storage like Azure blob storage or S3. File notification mode is more performant and scalable for large input directories or a high volume of files.

Here are some additional notes on when to use COPY INTO vs Auto Loader

When to use COPY INTO
https://docs.databricks.com/delta/delta-ingest.html#copy-into-sql-command
When to use Auto Loader
https://docs.databricks.com/delta/delta-ingest.html#auto-loader

Question 4:

Why does AUTO LOADER require schema location?

○ Schema location is used to store user provided schema

○ Schema location is used to identify the schema of target table

○ AUTO LOADER does not require schema location, because its supports Schema evolution

○ Schema location is used to store schema inferred by AUTO LOADER

○ Schema location is used to identify the schema of target table and source table

Explanation
The answer is, Schema location is used to store schema inferred by AUTO LOADER

Auto Loader samples the first 50 GB or 1000 files that it discovers, whichever limit is crossed first. To avoid incurring this inference cost at every stream start up, and to be able to provide a stable schema across stream restarts, you must set the option `cloudFiles.schemaLocation`. Auto Loader creates a hidden directory `_schemas` at this location to track schema changes to the input data over time.

The below link contains detailed documentation on different options

Auto Loader options | Databricks on AWS

Question 5:

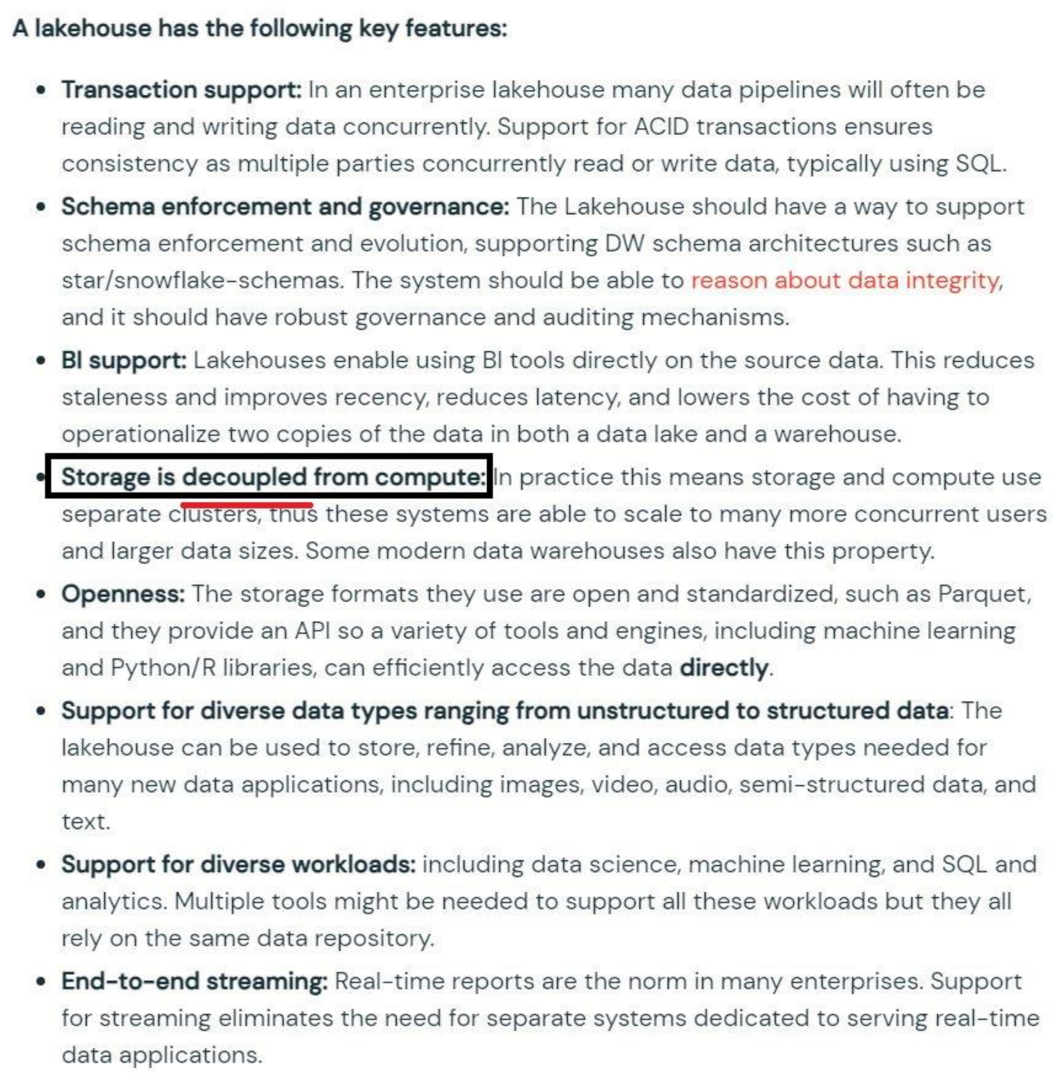
Which of the following statements are incorrect about the lakehouse

- Support end-to-end streaming and batch workloads
- Supports ACID
- Support for diverse data types that can store both structured and unstructured
- Supports BI and Machine learning
- [✓ Storage is coupled with Compute]

Explanation
The answer is, Storage is coupled with Compute.

The Question was asking what is the incorrect option, in Lakehouse Storage is decoupled with compute so both can scale independently.

[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.
- $\boxed{\textbf{Storage is decoupled from compute:}}$ 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 6:
You are designing a data model that works for both machine learning using images and Batch ETL/ELT workloads. Which of the following features of data lakehouse can help you meet the needs of both workloads?

○ Data lakehouse requires very little data modeling.

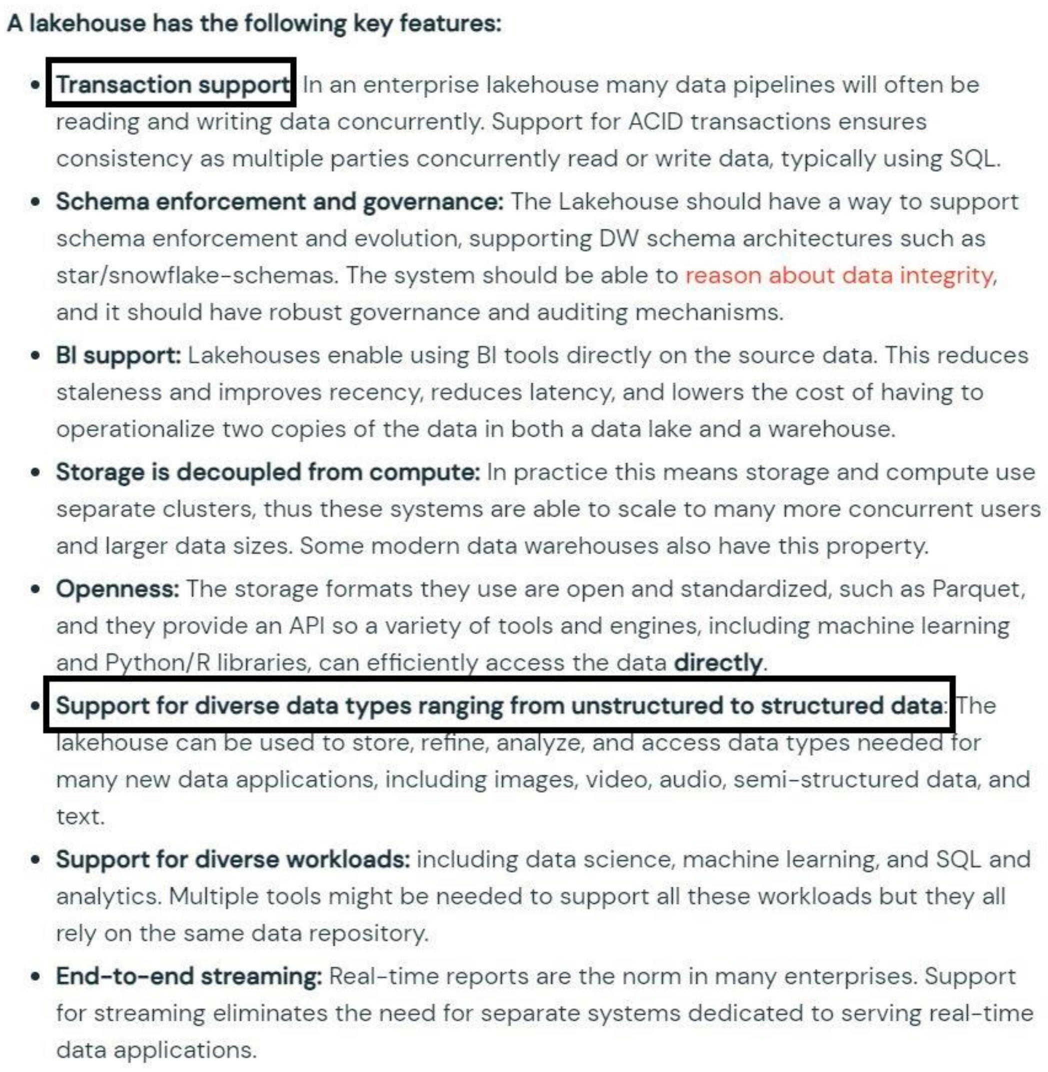
○ Data lakehouse combines compute and storage for simple governance.

○ Data lakehouse provides autoscaling for compute clusters.

○ Data lakehouse can store unstructured data and support ACID transactions.

○ Data lakehouse fully exists in the cloud.

Explanation
The answer is A data lakehouse stores unstructured data and is ACID-compliant,



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 7

Which of the following locations in Databricks product architecture hosts jobs/pipelines and queries?

○ Data plane

○ Control plane

○ Databricks Filesystem

○ JDBC data source

○ Databricks web application

Explanation
The answer is Control Plane,

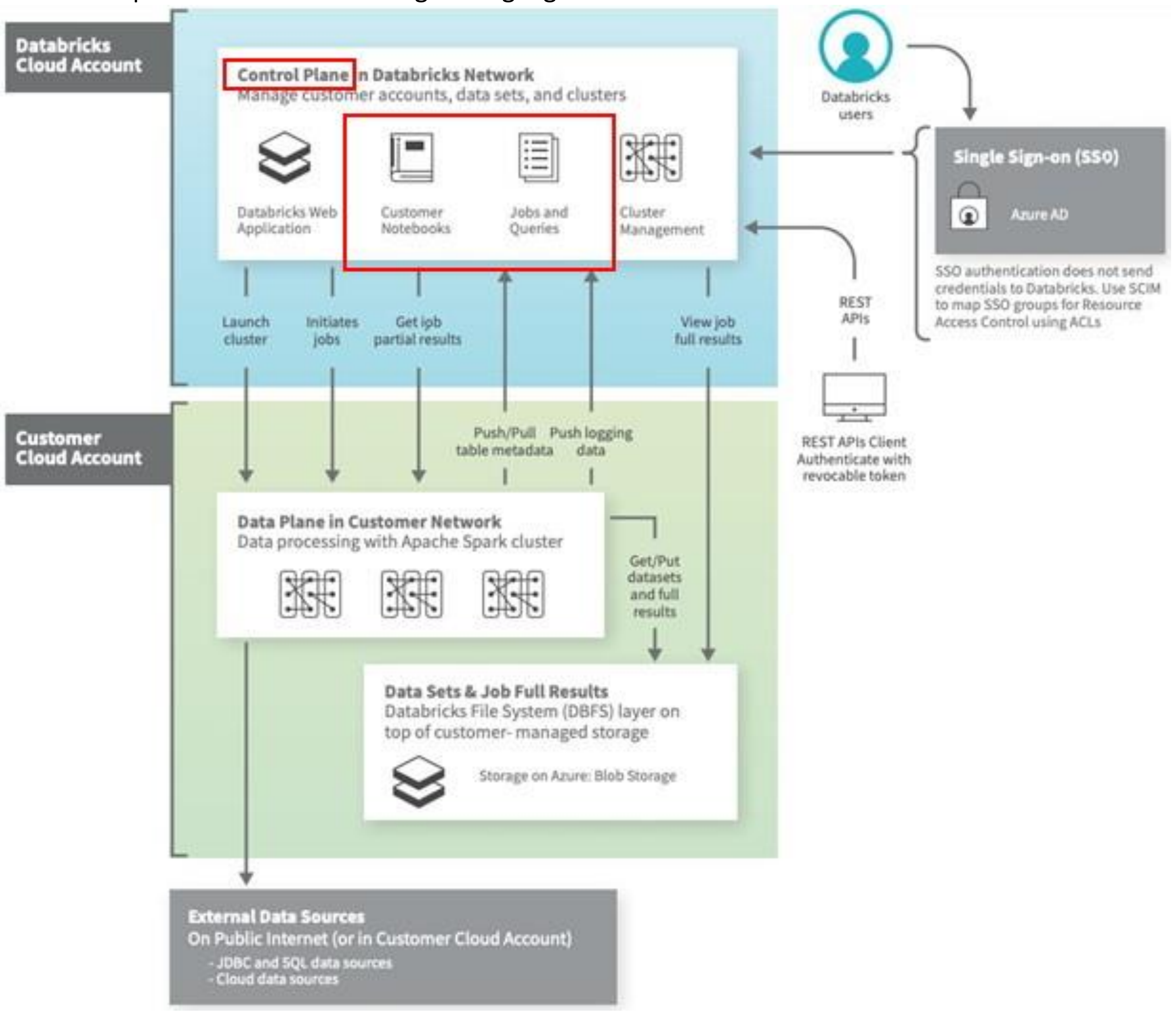
Databricks operates most of its services out of a control plane and a data plane, *please note serverless features like SQL Endpoint and DLT compute use shared compute in Control pane.*

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.

Here is the product architecture diagram highlighted where


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
Customer Cloud Account
Push/Pull table metadata
Push logging data
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 8:

You are currently working on a notebook that will populate a reporting table for downstream process consumption, this process needs to run on a schedule every hour. what type of cluster are you going to use to set up this job?

○

Since it's just a single job and we need to run every hour, we can use an all-purpose cluster

○

The job cluster is best suited for this purpose.

○

Use Azure VM to read and write delta tables in Python

○

Use delta live table pipeline to run in continuous mode

Explanation
The answer is, The Job cluster is best suited for this purpose.

Since you don't need to interact with the notebook during the execution especially when it's a scheduled job, job cluster makes sense. Using an all-purpose cluster can be twice as expensive as a job cluster.
FYI,
When you run a job scheduler with option of creating a new cluster when the job is complete it terminates the cluster. You cannot restart a job cluster.

Question 9

Which of the following developer operations in CI/CD flow can be implemented in Databricks Repos?

○ Merge when code is committed

○ Pull request and review process

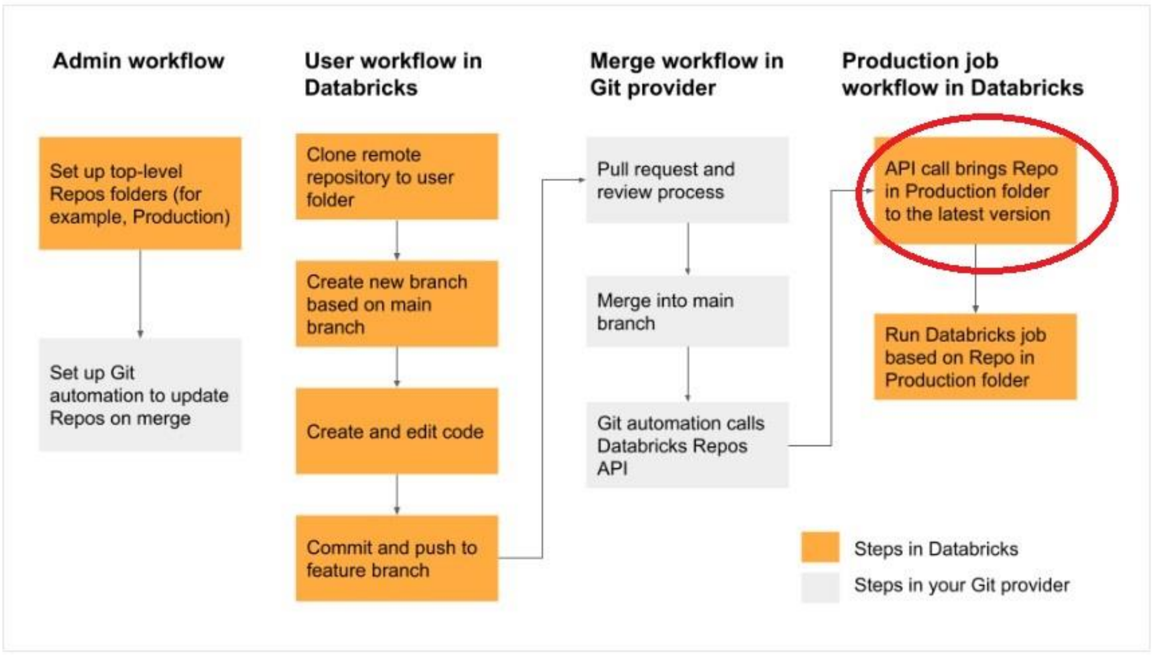
○ Trigger Databricks Repos pull API to update the latest version

○ Create and edit code.

○ Delete a branch

Explanation

See the below diagram to understand the role Databricks Repos and Git provider plays when building a CI/CD workflow.
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 10

You are currently working with the second team and both teams are looking to modify the same notebook, you noticed that the second member is copying the notebooks to the personal folder to edit and replace the collaboration notebook, which notebook feature do you recommend to make the process easier to collaborate.

○ Databricks notebooks should be copied to a local machine and setup source control locally to version the notebooks

○ Databricks notebooks support automatic change tracking and versioning

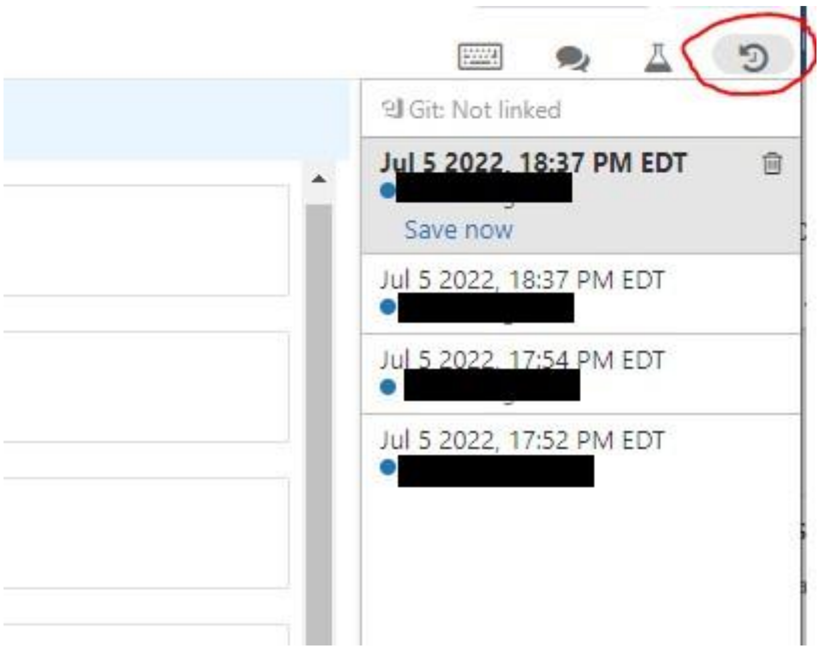
○ Databricks Notebooks support real-time coauthoring on a single notebook

○ Databricks notebooks can be exported into dbc archive files and stored in data lake

○ Databricks notebook can be exported as HTML and imported at a later time

Explanation
Answer is Databricks Notebooks support real-time coauthoring on a single notebook

Every change is saved, and a notebook can be changed my multiple users.


Git: Not linked

Jul 5 2022, 18:37 PM EDT
[redacted]
Save now

Jul 5 2022, 18:37 PM EDT
[redacted]

Jul 5 2022, 17:54 PM EDT
[redacted]

Jul 5 2022, 17:52 PM EDT
[redacted]

Question 11:
You are currently working on a project that requires the use of SQL and Python in a given notebook, what would be your approach

○

Create two separate notebooks, one for SQL and the second for Python

○

A single notebook can support multiple languages, use the magic command to switch between the two.

○

Use an All-purpose cluster for python, SQL endpoint for SQL

○

Use job cluster to run python and SQL Endpoint for SQL

Explanation
The answer is, A single notebook can support multiple languages, use the magic command to switch between the two.

Use %sql and %python magic commands within the same notebook.

Question 12:

Which of the following statements are correct on how Delta Lake implements a lake house?

○ Delta lake uses a proprietary format to write data, optimized for cloud storage

○ Using Apache Hadoop on cloud object storage

○ Delta lake always stores meta data in memory vs storage

○ Delta lake uses open source, open format, optimized cloud storage and scalable meta data

○ Delta lake stores data and meta data in computes memory

Explanation

Delta lake is
Open source
Builds up on standard data format
Optimized for cloud object storage
Built for scalable metadata handling
Delta lake is not
Proprietary technology
Storage format
Storage medium
Database service or data warehouse

Question 13:

You were asked to create or overwrite an existing delta table to store below transaction data.

transactionId	transactionDate	unitsSold
1	01-01-2021 09:10:24 AM	100
2	01-01-2022 10:30:30 AM	10

○
```
CREATE OR REPLACE DELTA TABLE transactions (
transactionId int,
transactionDate timestamp,
unitsSold int)
```
○
```
CREATE OR REPLACE TABLE IF EXISTS transactions (
transactionId int,
transactionDate timestamp,
unitsSold int)
FORMAT DELTA
```
○
```
CREATE IF EXSITS REPLACE TABLE transactions (
transactionId int,
transactionDate timestamp,
unitsSold int)
```
○
```
CREATE OR REPLACE TABLE transactions (
transactionId int,
transactionDate timestamp,
unitsSold int)
```

Explanation

The answer is

```
CREATE OR REPLACE TABLE transactions (
transactionId int,
transactionDate timestamp,
unitsSold int)
```

When creating a table in Databricks by default the table is stored in DELTA format.

Question 14

if you run the command `VACUUM transactions retain 0 hours`? What is the outcome of this command?

○ Command will be successful, but no data is removed

○ Command will fail if you have an active transaction running

○ Command will fail, you cannot run the command with retentionDurationcheck enabled

○ Command will be successful, but historical data will be removed

○ Command runs successful and compacts all of the data in the table

Explanation
The answer is,
Command will fail, you cannot run the command with retentionDurationcheck enabled.

```
VACUUM [ [db_name.]table_name | path] [RETAIN num HOURS] [DRY RUN]
```

Recursively vacuum directories associated with the Delta table and remove data files that are no longer in the latest state of the transaction log for the table and are older than a retention threshold. Default is 7 Days.
The reason this check is enabled is because, DELTA is trying to prevent unintentional deletion of history, and also one important thing to point out is with 0 hours of retention there is a possibility of data loss(see below kb)
Documentation in VACUUM https://docs.delta.io/latest/delta-utility.html
https://kb.databricks.com/delta/data-missing-vacuum-parallel-write.html

Question 15:

You noticed a colleague is manually copying the data to the backup folder prior to running an update command, incase if the update command did not provide the expected outcome so he can use the backup copy to replace table, which Delta Lake feature would you recommend simplifying the process?

○ Use time travel feature to refer old data instead of manually copying

○ Use DEEP CLONE to clone the table prior to update to make a backup copy

○ Use SHADOW copy of the table as preferred backup choice

○ Cloud object storage retains previous version of the file

○ Cloud object storage automatically backups the data

Explanation

The answer is, Use time travel feature to refer old data instead of manually copying.

https://databricks.com/blog/2019/02/04/introducing-delta-time-travel-for-large-scale-data-lakes.html

```
SELECT count(*) FROM my_table TIMESTAMP AS OF "2019-01-01"
SELECT count(*) FROM my_table TIMESTAMP AS OF date_sub(current_date(), 1)
SELECT count(*) FROM my_table TIMESTAMP AS OF "2019-01-01 01:30:00.000"
```

Question 16:

Which one of the following is not a Databricks lakehouse object?

○ Tables

○ Views

○ Database/Schemas

○ Catalog

○ Functions

○ Stored Procedures

Explanation
The answer is, Stored Procedures.
Databricks lakehouse does not support stored procedures.

Question 17:

What type of table is created when you create delta table with below command?

```
CREATE TABLE transactions USING DELTA LOCATION
"DBFS:/mnt/bronze/transactions"
```

○ Managed delta table

○ External table

○ Managed table

○ Temp table

○ Delta Lake table

Explanation

Anytime a table is created using the `LOCATION` keyword it is considered an external table, below is the current syntax.

Syntax

```
CREATE TABLE table_name ( column column_data_type…) USING format LOCATION "dbfs:/"
```

format -> DELTA, JSON, CSV, PARQUET, TEXT

Question 18

Which of the following command can be used to drop a managed delta table and the underlying files in the storage?

○ `DROP TABLE table_name CASCADE`

○ `DROP TABLE table_name`

○ Use `DROP TABLE table_name` command and manually delete files using command `dbutils.fs.rm("/path",True)`

○ `DROP TABLE table_name INCLUDE_FILES`

○ `DROP TABLE table` and run `VACUUM` command

Explanation
The answer is DROP TABLE table_name,

When a managed table is dropped, the table definition is dropped from metastore and everything including data, metadata, and history are also dropped from storage.

Question 19:

Which of the following is the correct statement for a session scoped temporary view?

○ Temporary views are lost once the notebook is detached and re-attached

○ Temporary views stored in memory

○ Temporary views can be still accessed even if the notebook is detached and attached

○ Temporary views can be still accessed even if cluster is restarted

○ Temporary views are created in local_temp database

Explanation
The answer is Temporary views are lost once the notebook is detached and attached

There are two types of temporary views that can be created, Session scoped and Global

A local/session scoped temporary view is only available with a spark session, so another notebook in the same cluster can not access it. if a notebook is detached and reattached local temporary view is lost.
A global temporary view is available to all the notebooks in the cluster, if a cluster restarts global temporary view is lost.

Question 20

Which of the following is correct for the global temporary view?

○

global temporary views cannot be accessed once the notebook is detached and attached

○

global temporary views can be accessed across many clusters

○

global temporary views can be still accessed even if the notebook is detached and attached

○

global temporary views can be still accessed even if the cluster is restarted

○

global temporary views are created in a database called `temp` database

Explanation
The answer is global temporary views can be still accessed even if the notebook is detached and attached

There are two types of temporary views that can be created Local and Global
A local temporary view is only available with a spark session, so another notebook in the same cluster can not access it. if a notebook is detached and reattached local temporary view is lost.
A global temporary view is available to all the notebooks in the cluster, even if the notebook is detached and reattached it can still be accessible but if a cluster is restarted the global temporary view is lost.

Question 21:

You are currently working on reloading customer_sales tables using the below query

```
INSERT OVERWRITE customer_sales
SELECT * FROM customers c
INNER JOIN sales_monthly s on s.customer_id = c.customer_id
```

After you ran the above command, the Marketing team quickly wanted to review the old data that was in the table. How does INSERT OVERWRITE impact the data in the `customer_sales` table if you want to see the previous version of the data prior to running the above statement?

○ Overwrites the data in the table, all historical versions of the data, you can not time travel to previous versions

○ Overwrites the data in the table but preserves all historical versions of the data, you can time travel to previous versions

○ Overwrites the current version of the data but clears all historical versions of the data, so you can not time travel to previous versions.

○ Appends the data to the current version, you can time travel to previous versions

○ By default, overwrites the data and schema, you cannot perform time travel

Explanation

The answer is, `INSERT OVERWRITE` Overwrites the current version of the data but preserves all historical versions of the data, you can time travel to previous versions.

```
INSERT OVERWRITE customer_sales
SELECT * FROM customers c
INNER JOIN sales s on s.customer_id = c.customer_id
```

Let's just assume that this is the second time you are running the above statement, you can still query the prior version of the data using time travel, and any DML/DDL except DROP TABLE creates new PARQUET files so you can still access the previous versions of data.

SQL Syntax for Time travel

`SELECT * FROM table_name as of [version number]`

with customer_sales example

`SELECT * FROM customer_sales as of 1 -- previous version`

`SELECT * FROM customer_sales as of 2 -- current version`

You see all historical changes on the table using DESCRIBE HISTORY table_name

Note: the main difference between `INSERT OVERWRITE` and `CREATE OR REPLACE TABLE(CRAS)` is that CRAS can modify the schema of the table, i.e it can add new columns or change data types of existing columns. By default `INSERT OVERWRITE` only overwrites the data.

`INSERT OVERWRITE` can also be used to update the schema when `spark.databricks.delta.schema.autoMerge.enabled` is set `true` if this option is not enabled and if there is a schema mismatch command `INSERT OVERWRITE` will fail.

Any DML/DDL operation(except DROP TABLE) on the Delta table preserves the historical version of the data.

Question 22

Which of the following SQL statements can be used to identify duplicate rows from a table?

◯

```
SELECT DISTINCT * FROM table_name
```

◯

```
SELECT DISTINCT * FROM table_name HAVING count(*) > 1
```

◯

```
SELECT DISTINCT_ROWS (*) FROM table_name
```

◯

```
SELECT * FROM table_name GROUP BY * HAVING COUN(*) < 1
```

◯

```
SELECT * FROM table_name GROUP BY * HAVING COUN(*) > 1
```

Explanation
The answer is SELECT DISTINCT * FROM table_name

Question 23:

Which of the below SQL Statements can be used to create a SQL UDF to convert Celsius to Fahrenheit and vice versa, you need to pass two parameters to this function one, actual temperature, and the second that identifies if its needs to be converted to Fahrenheit or Celcius with a one-word letter F or C?

`select udf_convert(60,'C')` will result in 15.5

`select udf_convert(10,'F')` will result in 50

○
```
 CREATE UDF FUNCTION udf_convert(temp DOUBLE, measure STRING)
    RETURNS DOUBLE
    RETURN CASE WHEN measure == 'F' then (temp * 9/5) + 32
            ELSE (temp – 33 ) * 5/9
           END
```

○
```
CREATE UDF FUNCTION udf_convert(temp DOUBLE, measure STRING)
    RETURN CASE WHEN measure == 'F' then (temp * 9/5) + 32
            ELSE (temp – 33 ) * 5/9
           END
```

○
```
CREATE FUNCTION udf_convert(temp DOUBLE, measure STRING)
RETURN CASE WHEN measure == 'F' then (temp * 9/5) + 32
          ELSE (temp – 33 ) * 5/9
       END
```

○
```
CREATE FUNCTION udf_convert(temp DOUBLE, measure STRING)
RETURNS DOUBLE
RETURN CASE WHEN measure == 'F' then (temp * 9/5) + 32
        ELSE (temp – 33 ) * 5/9
       END
```

○
```
CREATE USER FUNCTION udf_convert(temp DOUBLE, measure STRING)
RETURNS DOUBLE
RETURN CASE WHEN measure == 'F' then (temp * 9/5) + 32
          ELSE (temp – 33 ) * 5/9
      END
```

Explanation

The answer is

```
CREATE FUNCTION udf_convert(temp DOUBLE, measure STRING)
RETURNS DOUBLE
RETURN CASE WHEN measure == ‘F’ then (temp * 9/5) + 32
        ELSE (temp – 33 ) * 5/9
       END
```

Question 24
You are trying to parse a complex JSON object below, and calculate total sales made by all the employees, how would you approach this in SQL

`id INT, performance ARRAY<employeeId:INT, sales double>, insertDate TIMESTAMP`

Sample data of `performance` column

Sample data of performance column

```
[
{ "employeeId":1234
"sales" : 10000.00 },

{ "employeeId":3232
"sales" : 30000.00 }
]
```

○ WITH CTE as (SELECT EXPLODE (performance) FROM table_name)
SELECT SUM (performance.sales) FROM CTE

○ WITH CTE as (SELECT FLATTEN (performance) FROM table_name)
SELECT SUM (sales) FROM CTE

○ SELECT SUM (performance.sales) FROM employee

○ SELECT SUM (SLICE (performance, sales)) FROM employee

○ SELECT SUM (ELEMENT_AT (performance, sales)) FROM employee

Explanation

The answer is `SELECT SUM (performance.sales) FROM employee`

Nested JSON can be queried using the . notation `performance.sales` will give you access to all the sales values in the JSON structure.

Question 25

Which of the following statements can be used to test the functionality of code to test number of rows in the table equal to 10 in python?

```
row_count = spark.sql("select count(*) from table").collect()[0][0]
```

- ◯ `assert (row_count = 10, "Row count did not match")`
- ◯ `assert if (row_count = 10, "Row count did not match")`
- ◯ `assert row_count == 10, "Row count did not match"`
- ◯ `assert if row_count == 10, "Row count did not match"`
- ◯ `assert row_count = 10, "Row count did not match"`

Explanation

The answer is `assert row_count == 10, "Row count did not match"`

Review below documentation

[Assert Python]

Question 26

How do you handle failures gracefully when writing code in Pyspark, fill in the blanks to complete the below statement

```
_____

Spark.read.table("table_name").select("column").write.mode("append").SaveAsTable("
new_table_name")

_____

    print(f"query failed")
```

○ try: failure:

○ try: catch:

○ try: except:

○ try: fail:

○ try: error:

Explanation

The answer is try: and except:

Question 27:
You are working on a process to query the table based on batch date, and batch date is an input parameter and expected to change every time the program runs, what is the best way to we can parameterize the query to run without manually changing the batch date?

○ Create a notebook parameter for batch date and assign the value to a python variable and use a spark data frame to filter the data based on the python variable

○ Create a dynamic view that can calculate the batch date automatically and use the view to query the data

○ There is no way we can combine python variable and spark code

○ Manually edit code every time to change the batch date

○ Store the batch date in the spark configuration and use a spark data frame to filter the data based on the spark configuration.

Explanation

The answer is, Create a notebook parameter for batch date and assign the value to a python variable and use a spark data frame to filter the data based on the python variable

Question 28:
Which of the following commands results in the successful creation of a view on top of the streaming data frame?

○
```
Spark.read.table("sales").createOrReplaceTempView("streaming_vw")
```
○
```
Spark.readStream.table("sales").createOrReplaceTempView("streaming_vw")
```

○
```
Spark.read.table("sales").mode("stream").createOrReplaceTempView("streaming_vw")
```
○
```
Spark.read.table("sales").trigger("stream").createOrReplaceTempView("streaming_vw")
```
○
```
Spark.read.stream("sales").createOrReplaceTempView("streaming_vw")
```

Explanation

The answer is

```
Spark.readStream.table("sales").createOrReplaceTempView("streaming_vw")
```

Question 29

Which of the following techniques structured streaming uses to create an end-to-end fault tolerance?

○ Checkpointing and Water marking

○ Write ahead logging and water marking

○ Checkpointing and idempotent sinks

○ Write ahead logging and idempotent sinks

○ Stream will failover to available nodes in the cluste

Explanation
The answer is Checkpointing and idempotent sinks

How does structured streaming achieves end to end fault tolerance:
First, Structured Streaming uses checkpointing and write-ahead logs to record the offset range of data being processed during each trigger interval.
Next, the streaming sinks are designed to be _idempotent_—that is, multiple writes of the same data (as identified by the offset) do *not* result in duplicates being written to the sink.
Taken together, replayable data sources and idempotent sinks allow Structured Streaming to ensure end-to-end, exactly-once semantics under any failure condition.

Question 30:

Which of the following two options are supported in identifying the arrival of new files, and incremental data from Cloud object storage using Auto Loader?

○

Directory listing, File notification

○

Checking pointing, watermarking

○

Writing ahead logging, read head logging

○

File hashing, Dynamic file lookup

○

Checkpointing and Write ahead logging

Explanation
The answer is A, Directory listing, File notifications

Directory listing: Auto Loader identifies new files by listing the input directory.
File notification: Auto Loader can automatically set up a notification service and queue service that subscribe to file events from the input directory.

Choosing between file notification and directory listing modes | Databricks on AWS

Question 31:

Which of the following data workloads will utilize a Bronze table as its destination?

○ A job that aggregates cleaned data to create standard summary statistics

○ A job that queries aggregated data to publish key insights into a dashboard

○ A job that ingests raw data from a streaming source into the Lakehouse

○ A job that develops a feature set for a machine learning application

○ A job that enriches data by parsing its timestamps into a human-readable format

Explanation
The answer is A job that ingests raw data from a streaming source into the Lakehouse.

The ingested data from the raw streaming data source like Kafka is first stored in the Bronze layer as first destination before it is further optimized and stored in Silver.

Medallion Architecture – Databricks

Bronze Layer:
Raw copy of ingested data
Replaces traditional data lake
Provides efficient storage and querying of full, unprocessed history of data
No schema is applied at this layer

Question 32

Which of the following data workloads will utilize a silver table as its source?

○

A job that enriches data by parsing its timestamps into a human-readable format

○

A job that queries aggregated data that already feeds into a dashboard

○

A job that ingests raw data from a streaming source into the Lakehouse

○

A job that aggregates cleaned data to create standard summary statistics

○

A job that cleans data by removing malformatted records

Explanation
The answer is, A job that aggregates cleaned data to create standard summary statistics

Silver zone maintains the grain of the original data, in this scenario a job is taking data from the silver zone as the source and aggregating and storing them in the gold zone.

Medallion Architecture – Databricks
Silver Layer:

Reduces data storage complexity, latency, and redundency
Optimizes ETL throughput and analytic query performance
Preserves grain of original data (without aggregation)
Eliminates duplicate records
production schema enforced
Data quality checks, quarantine corrupt data

Question 33

Which of the following data workloads will utilize a gold table as its source?

○ A job that enriches data by parsing its timestamps into a human-readable format

○ A job that queries aggregated data that already feeds into a dashboard

○ A job that ingests raw data from a streaming source into the Lakehouse

○ A job that aggregates cleaned data to create standard summary statistics

○ A job that cleans data by removing malformatted records

Explanation
The answer is, A job that queries aggregated data that already feeds into a dashboard

The gold layer is used to store aggregated data, which are typically used for dashboards and reporting.

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 34:

You are currently asked to work on building a data pipeline, you have noticed that you are currently working with data source with a lot of data quality issues and you need to monitor data quality and enforce it as part of the data ingestion process, which of the following tools can be used to address this problem?

○ AUTO LOADER

○ DELTA LIVE TABLES

○ JOBS and TASKS

○ UNITY Catalog and Data Governance

○ STRUCTURED STREAMING with MULTI HOP

Explanation
The answer is, DELTA LIVE TABLES

Delta live tables expectations can be used to identify and quarantine bad data, all of the data quality metrics are stored in the event logs which can be used to later analyze and monitor.

DELTA LIVE Tables expectations

Below are three types of expectations, make sure to pay attention differences between these three.

Retain invalid records:
Use the `expect` operator when you want to keep records that violate the expectation. Records that violate the expectation are added to the target dataset along with valid records:

Python
```
@dlt.expect("valid timestamp", "col(“timestamp”) > '2012-01-01'")
```
SQL
```
CONSTRAINT valid_timestamp EXPECT (timestamp > '2012-01-01')
```
Drop invalid records:
Use the `expect or drop` operator to prevent the processing of invalid records. Records that violate the expectation are dropped from the target dataset:

Python
```
@dlt.expect_or_drop("valid_current_page", "current_page_id IS NOT NULL AND
current_page_title IS NOT NULL")
```
SQL
```
CONSTRAINT valid_current_page EXPECT (current_page_id IS NOT NULL and
current_page_title IS NOT NULL) ON VIOLATION DROP ROW
```
Fail on invalid records:
When invalid records are unacceptable, use the `expect or fail` operator to halt execution immediately when a record fails validation. If the operation is a table update, the system atomically rolls back the transaction:

Python
```
@dlt.expect_or_fail("valid_count", "count > 0")
```
SQL
```
CONSTRAINT valid_count EXPECT (count > 0) ON VIOLATION FAIL UPDATE
```

Question 35

What is the main difference between CREATE STREAMING LIVE TABLE vs CREATE LIVE TABLE?

○ CREATE STREAMING LIVE table is used in MULTI HOP Architecture

○ CREATE LIVE TABLE is used when working with Streaming data sources and Incremental data

○ CREATE STREAMING LIVE TABLE is used when working with Streaming data sources and Incremental data

○ There is no difference both are the same, CREATE STRAMING LIVE will be deprecated soon

○ CREATE LIVE TABLE is used in DELTA LIVE TABLES, CREATE STREAMING LIVE can only used in Structured Streaming applications

Explanation
The answer is, CREATE STREAMING LIVE TABLE is used when working with Streaming data sources and Incremental data

Question 36

A particular job seems to be performing slower and slower over time, the team thinks this started to happen when a recent production change was implemented, you were asked to take look at the job history and see if we can identify trends and root cause, where in the workspace UI can you perform this analysis?

○ Under jobs UI select the job you are interested, under runs we can see current active runs and last 60 days historical run

○ Under jobs UI select the job cluster, under spark UI select the application job logs, then you can access last 60 day historical runs

○ Under Workspace logs, select job logs and select the job you want to monitor to view the last 60 day historical runs

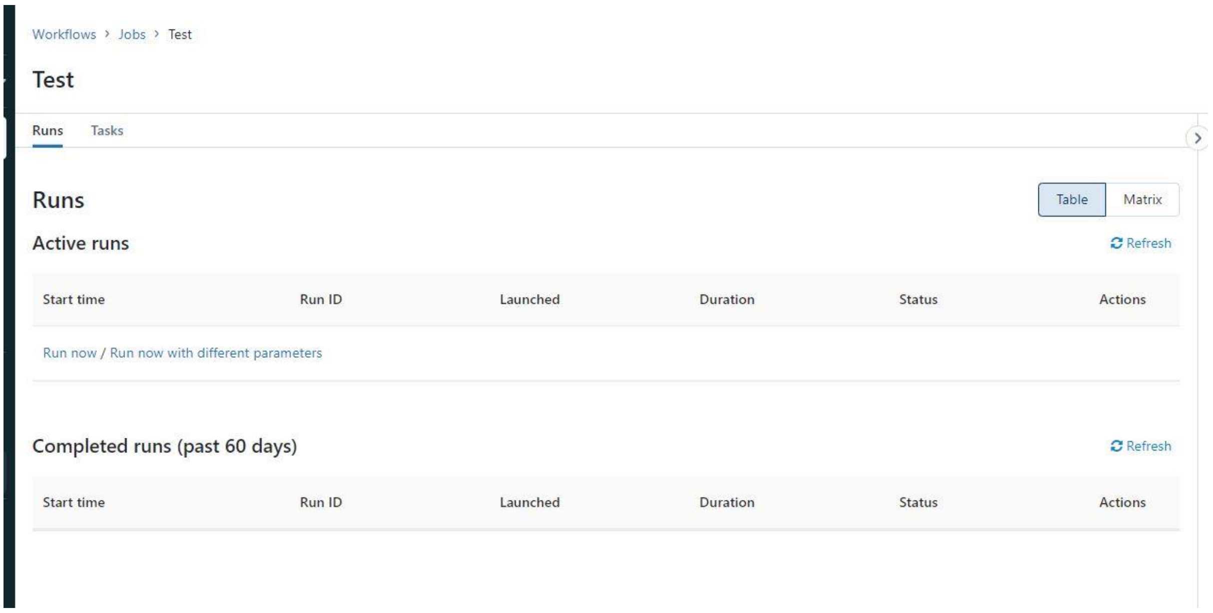
○ Under Compute UI, select Job cluster and select the job cluster to see last 60 day historical runs

○ Historical job runs can only be accessed by REST API

Explanation

The answer is,

Under jobs UI select the job you are interested, under runs we can see current active runs and last 60 days historical run



Workflows › Jobs › Test

Test

Runs

Active runs

Start time	Run ID	Launched	Duration	Status	Actions
Run now / Run now with different parameters					

Completed runs (past 60 days)

Start time	Run ID	Launched	Duration	Status	Actions

Question 37

What are the different ways you can schedule a job in Databricks workspace?

○ Continuous, Incremental

○ On-Demand runs, File notification from Cloud object storage

○ Cron, On Demand runs

○ Cron, File notification from Cloud object storage

○ Once, Continuous

Explanation
The answer is, Cron, On-Demand runs

Supports running job immediately or using can be scheduled using CRON syntax

Jobs in Databricks

Question 38

You have noticed that Databricks SQL queries are running slow, you are asked to look reason why queries are running slow and identify steps to improve the performance, when you looked at the issue you noticed all the queries are running in parallel and using a SQL endpoint cluster. Which of the following steps can be taken to resolve the issue?

○ They can turn on the Serverless feature for the SQL endpoint.

○ They can increase the maximum bound of the SQL endpoint's scaling range.

○ They can increase the cluster size of the SQL endpoint.

○ They can turn on the Auto Stop feature for the SQL endpoint.

○ They can turn on the Serverless feature for the SQL endpoint and change the Spot Instance Policy to "Reliability Optimized."

Explanation
The answer is, They can increase the maximum bound of the SQL endpoint’s scaling range.

SQL endpoint scales horizontally(scale-out) and vertical (scale-up), you have to understand when to use what.
Scale-out -> to add more clusters for a SQL endpoint, change max number of clusters
If you are trying to improve the throughput, being able to run as many queries as possible then having an additional cluster(s) will improve the performance.
Scale-up-> Increase the size of the SQL endpoint, change cluster size from x-small to small, to medium, X Large….
If you are trying to improve the performance of a single query having additional memory, additional nodes and cpu in the cluster will improve the performance.

Question 39

You currently working with marketing team to setup a dashboard on ad campaign analysis, since the team is not sure how often the dashboard should refreshed they have decided to do a manual refresh on as needed basis. Which of the following steps can be taken to reduce the overall cost of the compute?

○

They can turn on the Serverless feature for the SQL endpoint.

○

They can decrease the maximum bound of the SQL endpoint's scaling range.

○

They can decrease the cluster size of the SQL endpoint.

○

They can turn on the Auto Stop feature for the SQL endpoint.

○

They can turn on the Serverless feature for the SQL endpoint and change the Spot Instance Policy from "Reliability Optimized" to "Cost optimized"

Explanation
The answer is, They can turn on the Auto Stop feature for the SQL endpoint.

Use auto stop to automatically terminate the cluster when you are not using it.

Question 40
You had worked with the Data analysts team to set up a SQL Endpoint point so they can easily query and analze data in the gold layer, but once they started consuming the SQL Endpoint you noticed that during the peak hours as the number of users increase you are seeing a degradation in the query performance, which of the following steps can be taken to resolve the issue?

○ They can turn on the Serverless feature for the SQL endpoint.

○ They can increase the maximum bound of the SQL endpoint's scaling range.

○ They can increase the cluster size of the SQL endpoint.

○ They can turn on the Auto Stop feature for the SQL endpoint.

○ They can turn on the Serverless feature for the SQL endpoint and change the Spot Instance Policy from "Cost optimized" to "Reliability Optimized."

Explanation
the answer is,
They can increase the maximum bound of the SQL endpoint's scaling range.

SQL endpoint scales horizontally(scale-out) and vertical (scale-up), you have to understand when to use what.
Scale-out -> to add more clusters for a SQL endpoint, change max number of clusters
If you are trying to improve the throughput, being able to run as many queries as possible then having an additional cluster(s) will improve the performance.
Scale-up-> Increase the size of the SQL endpoint, change cluster size from x-small to small, to medium, X Large.
If you are trying to improve the performance of a single query having additional memory, additional nodes, and cpu in the cluster will improve the performance

Question 41

The research team has put together a funnel analysis query to monitor the customer traffic on the e-commerce platform, the query takes about 30 mins to run on a small SQL endpoint cluster with max scaling set to 1 cluster.

○ They can turn on the Serverless feature for the SQL endpoint.

○ They can increase the maximum bound of the SQL endpoint's scaling range anywhere from between 1 to 100 to review the performance and select the size that meets the required SLA.

○ They can increase the cluster size anywhere from X small to 3XL to review the performance and select the size that meets the required SLA.

○ They can turn off the Auto Stop feature for the SQL endpoint to more than 30 mins.

○ They can turn on the Serverless feature for the SQL endpoint and change the Spot Instance Policy from "Cost optimized" to "Reliability Optimized."

Explanation
The answer is, They can increase the cluster size anywhere from small to 3XL to review the performance and select the size that meets your SLA.

SQL endpoint scales horizontally(scale-out) and vertical (scale-up), you have to understand when to use what.
Scale-out -> to add more clusters for a SQL endpoint, change max number of clusters
If you are trying to improve the throughput, being able to run as many queries as possible then having an additional cluster(s) will improve the performance.
Scale-up-> Increase the size of the SQL endpoint, change cluster size from x-small to small, to medium, X Large....
If you are trying to improve the performance of a single query having additional memory, additional nodes and cpu in the cluster will improve the performance.

Question 42:
Unity catalog simplifies managing multiple workspaces, by storing and managing permissions and ACL at _______ level

○ Workspace

○ Account

○ Storage

○ Data pane

○ Control pane

Explanation
The answer is, Account Level
The classic access control list (tables, workspace, cluster) is at the workspace level, Unity catalog is at the account level and can manage all the workspaces in an Account.

Question 43:
Which of the following section in the UI can be used to manage permissions and grants to tables?

○ User Settings

○ Admin UI

○ Workspace admin settings

○ User access control lists

○ Data Explorer

Explanation
The answer is Data Explorer

Data explorer

Question 44:
Which of the following is not a privilege in the Unity catalog?

○ SELECT

○ MODIFY

○ EXECUTE

○ CREATE

○ USAGE

Explanation
The Answer is EXECUTE,

Here are the list of all privileges
Privileges
`SELECT`: gives read access to an object.
`CREATE`: gives ability to create an object (for example, a table in a schema).
`MODIFY`: gives ability to add, delete, and modify data to or from an object.
`USAGE`: does not give any abilities, but is an additional requirement to perform any action on a schema object.
`READ_METADATA`: gives ability to view an object and its metadata.
`CREATE_NAMED_FUNCTION`: gives ability to create a named UDF in an existing catalog or schema.
`MODIFY_CLASSPATH`: gives ability to add files to the Spark class path.
`ALL PRIVILEGES`: gives all privileges (is translated into all the above privileges).

Privileges

Question 45

A team member is leaving the team and he/she is currently the owner of the few tables, instead of transfering the ownership to a user you have decided to transfer the ownership to a group so in the future anyone in the group can manage the permissions rather than a single individual, which of the following commands help you accomplish this?

- ○ `ALTER TABLE table_name OWNER to 'group'`
- ○ `TRANSFER OWNER table_name to 'group'`
- ○ `GRANT OWNER table_name to 'group'`
- ○ `ALTER OWNER ON table_name to 'group'`
- ○ `GRANT OWNER On table_name to 'group'`

Explanation
The answer is ALTER TABLE table_name OWNER to ‘group’
Assign owner to object