

# Quiz -- Day 3

**Modules:** M07 (Medallion & Lakeflow), M08 (Orchestration), M09 (Governance & Security)

**Format:** 20 questions, single correct answer (A-D)

**Time:** ~15-20 min

---

Write your answers in the table at the end, then check against the Answer Key at the bottom.

## Q1

In the Medallion Architecture, which layer contains raw, unvalidated data ingested from source systems?

- A. Silver
- B. Gold
- C. Bronze
- D. Platinum

## Q2

In a Lakeflow pipeline, what is the difference between a STREAMING TABLE and a MATERIALIZED VIEW?

- A. STREAMING TABLE supports batch only; MATERIALIZED VIEW supports streaming only
- B. STREAMING TABLE processes data incrementally (append-only); MATERIALIZED VIEW recomputes the full result set
- C. They are identical; the names are interchangeable
- D. STREAMING TABLE stores views; MATERIALIZED VIEW stores tables

## Q3

In Lakeflow SQL, what does the **LIVE.** prefix reference?

- A. A live-streaming external source
- B. Another table or view within the same pipeline
- C. A table in the default catalog
- D. A real-time dashboard

## Q4

What does this Lakeflow declaration do?

**CONSTRAINT** valid\_id EXPECT (id **IS NOT NULL**) **ON** VIOLATION **DROP ROW**

- A. Logs a warning for null IDs but keeps all rows
- B. Drops the entire pipeline run if any null ID is found
- C. Silently removes rows where `id` is NULL from the output
- D. Replaces NULL IDs with a default value

## Q5

What happens when a Lakeflow expectation uses **ON VIOLATION FAIL UPDATE** ?

- A. The failing rows are dropped and the pipeline continues
- B. The pipeline update fails immediately, and no data is written for that table
- C. A warning is logged but processing continues
- D. The failing rows are quarantined to an error table

## Q6

What happens when an expectation is declared without an **ON VIOLATION** clause?

**CONSTRAINT** positive\_amount EXPECT (amount > 0)

- A. Failing rows are dropped
- B. The pipeline fails
- C. All rows are kept; a warning is logged and metrics are recorded
- D. An error -- **ON VIOLATION** is required

## Q7

Which Lakeflow SQL keyword is used to read from an upstream streaming table within the same pipeline?

A.

**SELECT \* FROM** STREAM(LIVE.bronze\_orders)

B.

**SELECT \* FROM** LIVE.bronze\_orders

C.

**SELECT \* FROM** STREAMING(bronze\_orders)

D.

```
SELECT * FROM READ_STREAM(bronze_orders)
```

## Q8

Where can you view data quality metrics (pass/fail counts) for Lakeflow expectations?

A.

```
SELECT * FROM event_log(TABLE(catalog.schema.silver_table))  
WHERE details:flow_progress:data_quality IS NOT NULL
```

B.

```
DESCRIBE EXPECTATIONS silver_table
```

C.

```
SELECT * FROM INFORMATION_SCHEMA.EXPECTATIONS
```

D.

```
SHOW QUALITY METRICS FOR silver_table
```

## Q9

In Databricks Workflows, what is a "Task" within a Job?

- A. A separate cluster configuration
- B. A single unit of work (notebook, SQL, Python script, or pipeline) that the Job orchestrates
- C. A permission setting
- D. A notification channel

## Q10

What does `dbutils.jobs.taskValues.set()` do?

Task 1:

```
dbutils.jobs.taskValues.set(key="row_count", value=1000)
```

Task 2 (downstream):

```
count = dbutils.jobs.taskValues.get(taskKey="Task_1", key="row_count")
```

- A. Sets a job-level environment variable
- B. Passes a value from one task to a downstream task within the same job run
- C. Configures cluster parameters
- D. Sets the task retry count

## Q11

What is a "Repair Run" in Databricks Workflows?

- A. A full re-run of the entire job from scratch
- B. Re-runs only the failed tasks (and their downstream dependents) without re-running successful ones
- C. A manual fix applied to the job configuration
- D. Rolling back the job to a previous version

## Q12

Which of the following is a valid task type in a Databricks Workflow Job?

- A. Notebook, SQL, Python script, Lakeflow pipeline, JAR
- B. Only notebooks
- C. Only SQL and Python
- D. Only Lakeflow pipelines

## Q13

Which Unity Catalog feature allows you to restrict which rows a user can see when querying a table?

- A. Column Mask
- B. Row Filter
- C. Dynamic View
- D. Data Profile

## Q14

How do you apply a Column Mask to a table in Unity Catalog?

A.

```
GRANT MASK ON COLUMN email TO mask_func
```

B.

```
CREATE MASK ON TABLE t COLUMN email
```

C.

```
ALTER TABLE t ALTER COLUMN email SET MASK catalog.schema.mask_func
```

D.

```
UPDATE TABLE t SET COLUMN email MASKED
```

## Q15

Which INFORMATION\_SCHEMA view would you query to list all tables and views in a Unity Catalog catalog?

- A. INFORMATION\_SCHEMA.SCHEMATA
- B. INFORMATION\_SCHEMA.COLUMNS
- C. INFORMATION\_SCHEMA.TABLES
- D. INFORMATION\_SCHEMA.TABLE\_PRIVILEGES

## Q16

Which SQL function is used inside Row Filters and Column Masks to check if the querying user belongs to a specific group?

A.

`is_account_group_member('admins')`

B.

`user_in_group('admins')`

C.

`has_role('admins')`

D.

`check_group('admins')`

## Q17

What is the purpose of Delta Sharing?

- A. Sharing compute clusters between workspaces
- B. An open protocol for secure, cross-organization data sharing without copying data
- C. Sharing notebooks between users
- D. Synchronizing Git repositories

## Q18

Which statement correctly grants SELECT access on all tables in the silver schema to the analysts group?

A.

`GRANT SELECT ON SCHEMA my_catalog.silver TO `analysts``

B.

**GRANT READ ON DATABASE** silver **TO** analysts

C.

**GRANT ALL ON SCHEMA** my\_catalog.silver **TO** `analysts`

D.

**ALLOW SELECT ON** my\_catalog.silver **FOR** analysts

## Q19

How do you remove a Row Filter from a table?

- A. **DROP FILTER ON TABLE t**
- B. **ALTER TABLE t DROP ROW FILTER**
- C. **REVOKE ROW FILTER FROM t**
- D. **DELETE ROW FILTER ON t**

## Q20

What does Unity Catalog's lineage feature provide?

- A. Version history of table schema changes
- B. Automatic tracking of data flow from source to downstream tables, showing column-level dependencies
- C. A list of users who accessed a table
- D. Storage cost breakdown per table

---

## Your Answers

**Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 Q9 Q10**

---

**Q11 Q12 Q13 Q14 Q15 Q16 Q17 Q18 Q19 Q20**

---

**Score: \_\_\_\_ / 20**

---

---

---

*Scroll down for Answer Key*

## Answer Key -- Day 3

#	Ans	Explanation
Q1	C	Bronze is the raw/landing layer. Silver is cleaned/validated. Gold is business-level aggregations.
Q2	B	STREAMING TABLE processes data incrementally (append-only). MATERIALIZED VIEW recomputes the full result when updated.
Q3	B	<code>LIVE.table_name</code> references other datasets within the same Lakeflow pipeline. It is pipeline-scoped.
Q4	C	<code>ON VIOLATION DROP ROW</code> silently filters out rows that fail the expectation.
Q5	B	<code>FAIL UPDATE</code> aborts the pipeline immediately, preventing bad data from being written.
Q6	C	Without <code>ON VIOLATION</code> , the expectation is in "warn" mode: all rows pass through, but violations are logged as metrics.
Q7	A	<code>STREAM(LIVE.table)</code> reads from an upstream dataset as a streaming source. <code>LIVE.table</code> (without <code>STREAM</code> ) reads as batch.
Q8	A	<code>event_log(TABLE(...))</code> is the function to query pipeline event logs, including data quality metrics.
Q9	B	A Task is the smallest unit of work in a Job. Jobs can have multiple tasks with dependencies forming a DAG.
Q10	B	<code>taskValues.set(key, value)</code> stores a value that downstream tasks retrieve with <code>taskValues.get()</code> .
Q11	B	Repair Run re-executes only failed tasks and their downstream dependents, saving time and compute.

#	Ans	Explanation
Q12	A	Valid task types include: Notebook, SQL, Python script, Lakeflow pipeline, JAR, dbt, and more.
Q13	B	Row Filters are SQL UDFs attached to tables that filter rows based on the querying user's identity or group.
Q14	C	Column Masks are applied via <code>ALTER TABLE ... ALTER COLUMN ... SET MASK &lt;function&gt; .</code>
Q15	C	<code>INFORMATION_SCHEMA.TABLES</code> lists all tables and views with their schema, name, and type.
Q16	A	<code>is_account_group_member('group_name')</code> returns true if the current user belongs to the specified group.
Q17	B	Delta Sharing is an open protocol for cross-org data sharing without copying data.
Q18	A	<code>GRANT SELECT ON SCHEMA catalog.schema TO principal</code> grants SELECT on all current and future tables in the schema.
Q19	B	<code>ALTER TABLE t DROP ROW FILTER</code> removes the row filter. Similarly, <code>ALTER TABLE t ALTER COLUMN c DROP MASK</code> removes a mask.
Q20	B	Lineage automatically tracks data flow between tables, including column-level dependencies, in the Unity Catalog UI.