

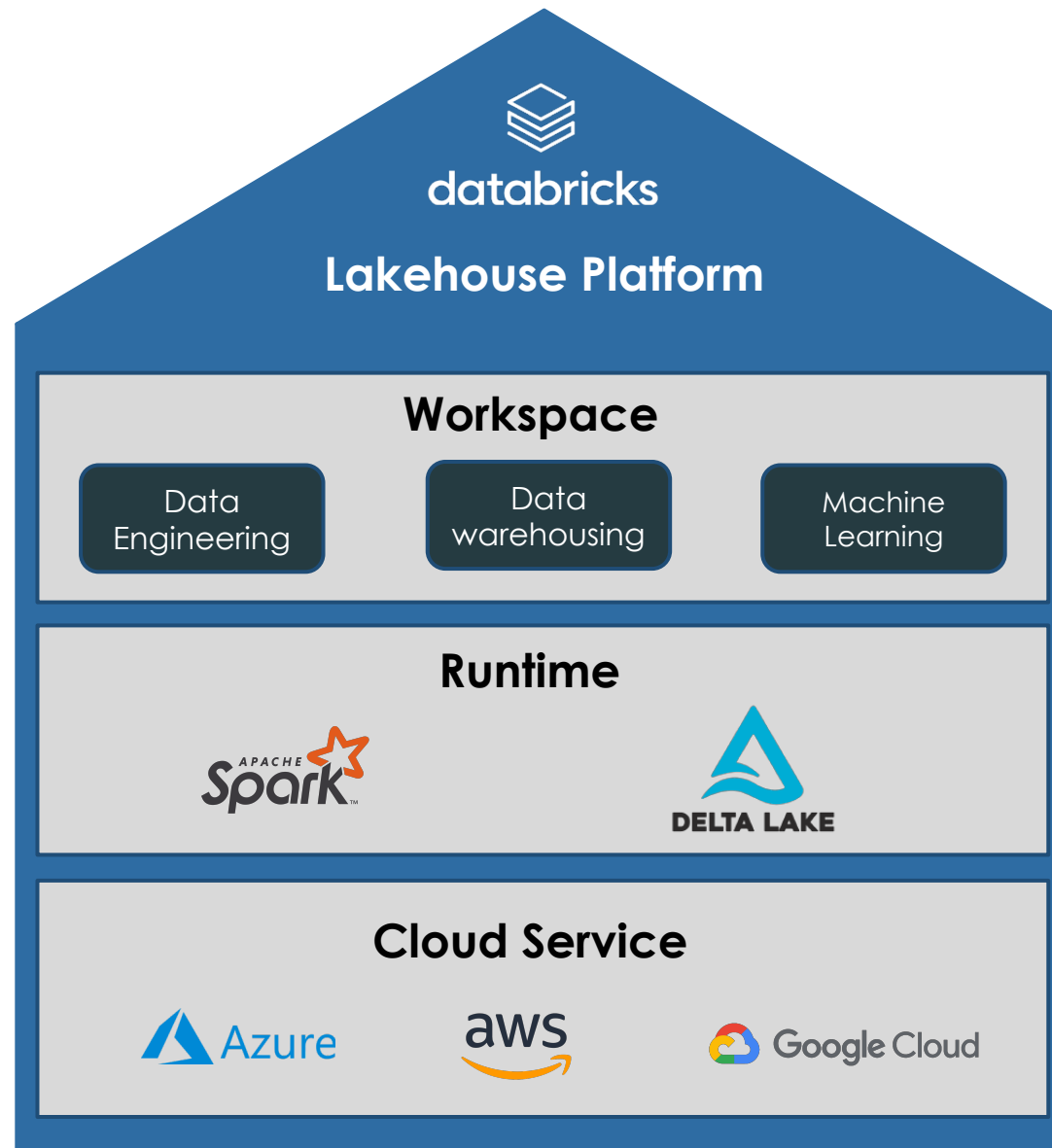
What is Databricks

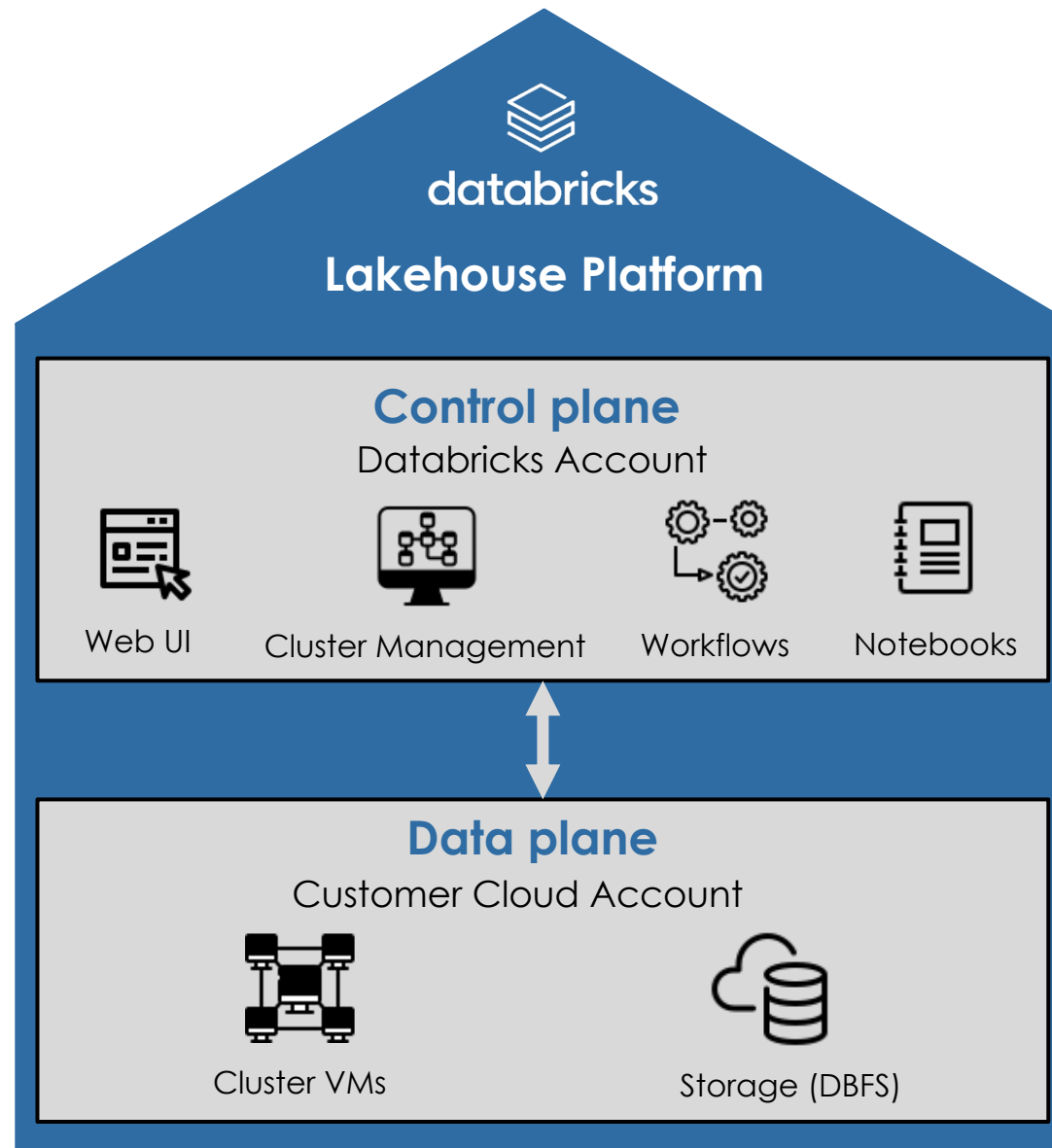
Databricks

► Multi-cloud Lakehouse Platform
based on Apache Spark

Lakehouse







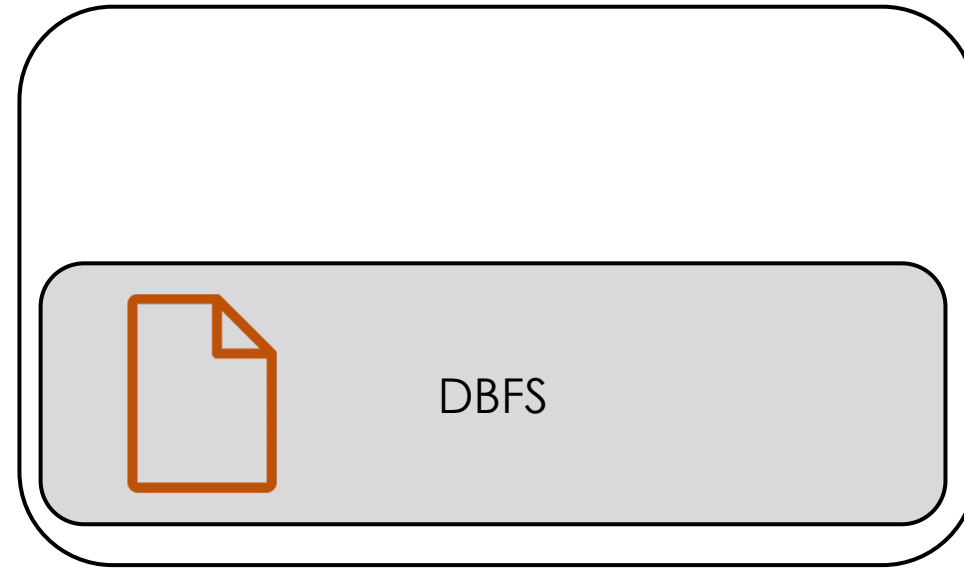
Spark on Databricks

- ▶ In-memory, distributed data processing
- ▶ All languages support
 - ▶ Scala, Python, SQL, R, & Java
- ▶ Batch processing & stream processing
- ▶ Structured, semi structured, & unstructured data

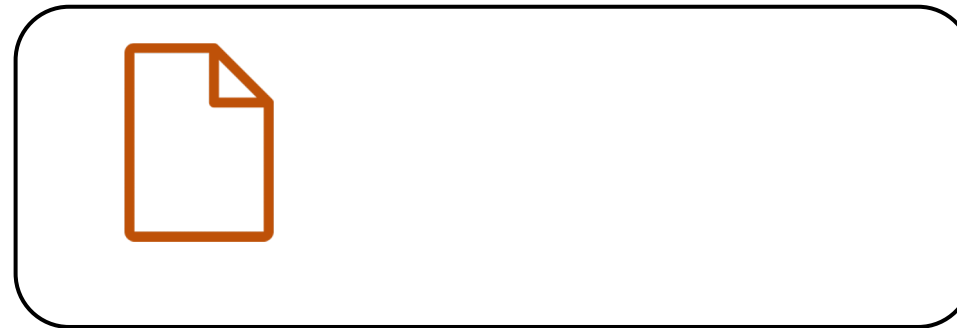
Databricks File System (DBFS)

- ▶ Distributed file system
- ▶ Preinstalled in Databricks clusters
- ▶ Abstraction layer
 - ▶ data persisted to the underlying cloud storage

Cluster



**Cloud
Storage**



Delta Lake

Delta Lake

- ▶ Open-source storage framework that brings reliability to data lakes

Delta Lake is/is not

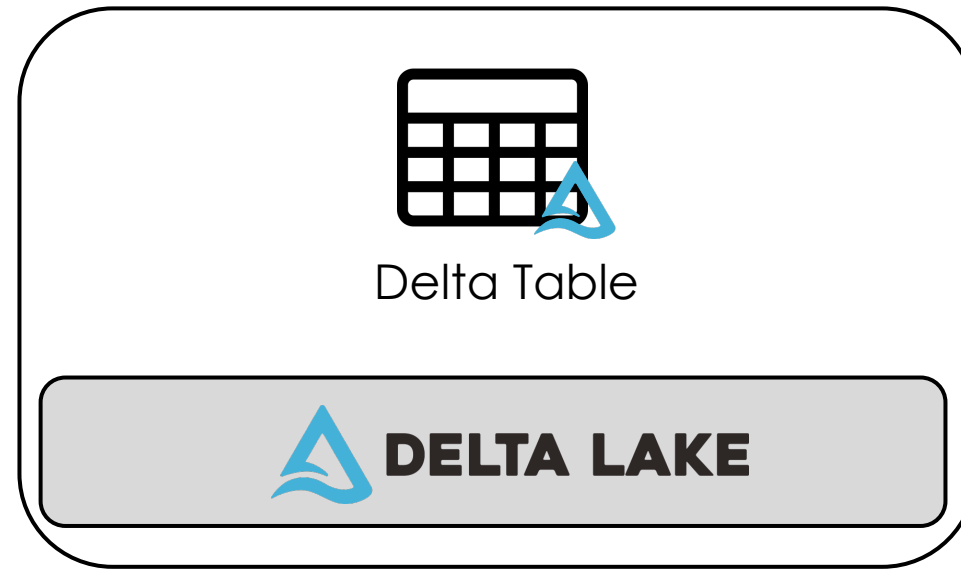
Is

- ▶ Open-source technology
- ▶ Storage framework/layer
- ▶ Enabling building Lakehouse

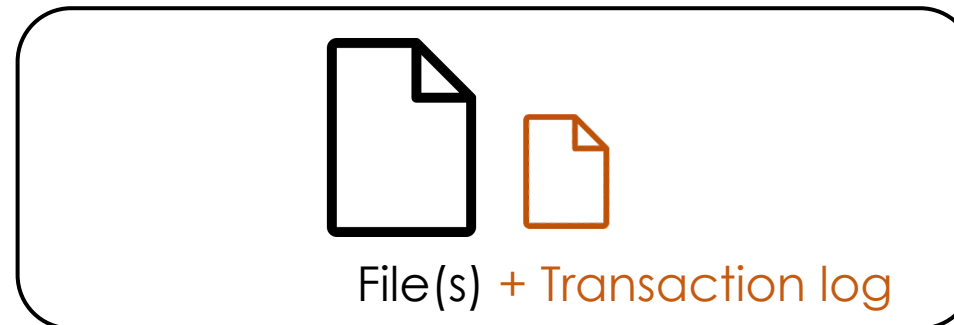
Is Not

- ▶ Proprietary technology
- ▶ Storage format/medium
- ▶ Data warehouse/Database service

Cluster



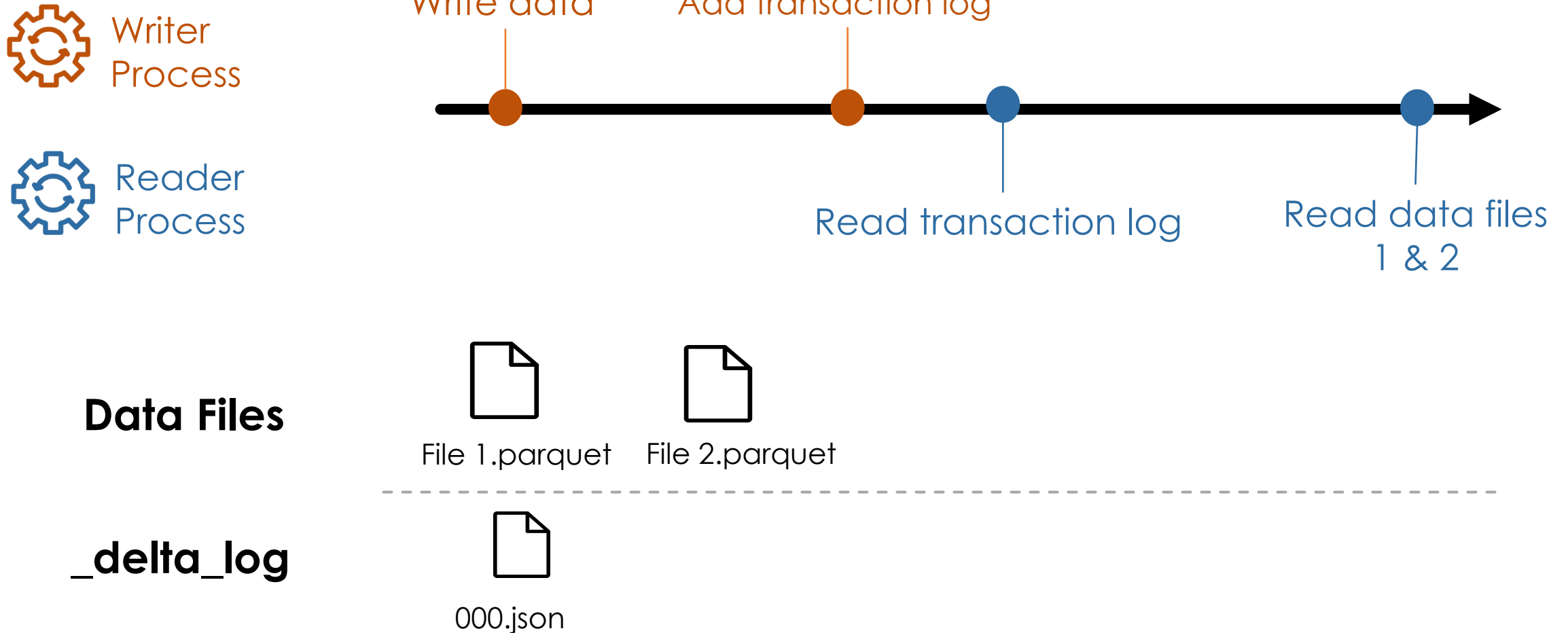
Storage



Transaction log (Delta log)

- ▶ Ordered records of every transaction performed on the table
- ▶ Single Source of Truth
- ▶ JSON file contains commit information:
 - ▶ Operation performed + Predicates used
 - ▶ data files affected (added/removed)

Writes/Reads



Updates



Writer
Process



Reader
Process

Update data

Add transaction log

Read transaction log

Read data files
2 & 3

copy & update



File 1.parquet



File 2.parquet



File 3.parquet

Data Files

_delta_log

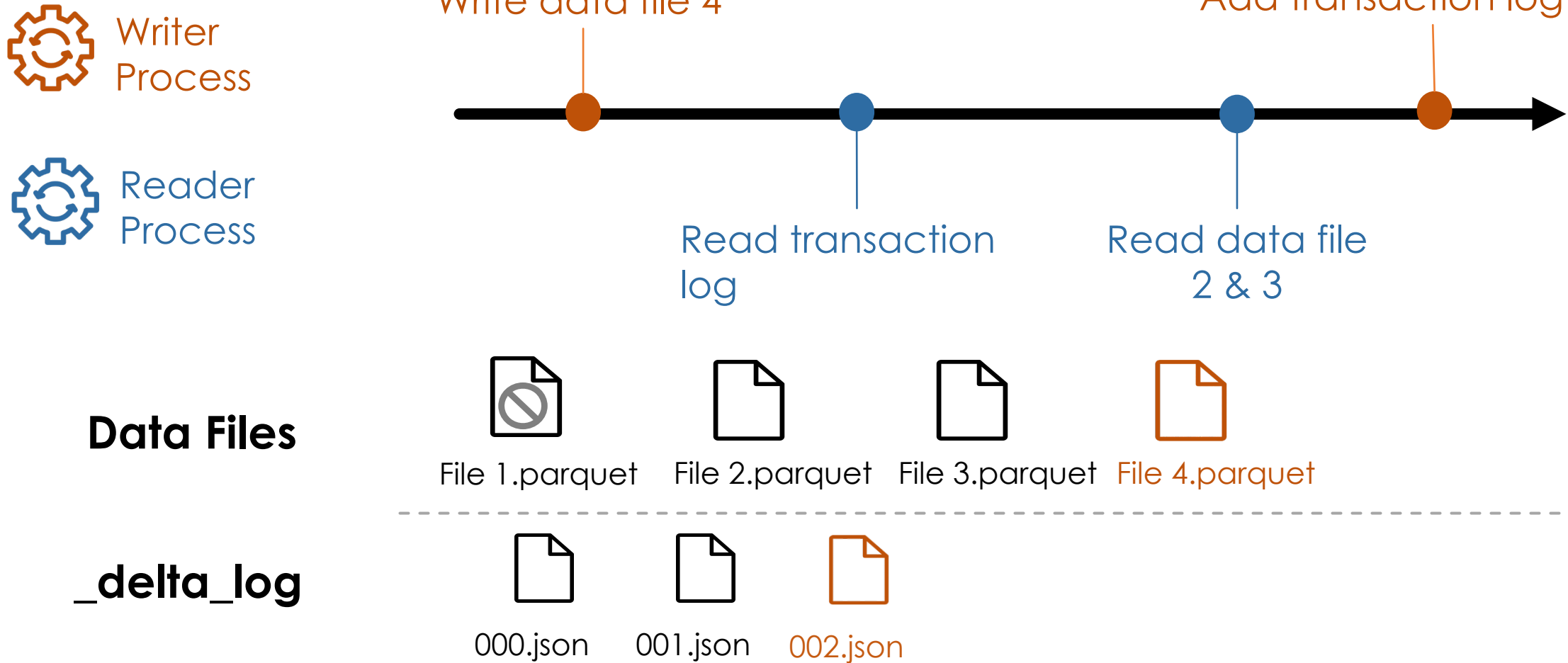


000.json



001.json

Simultaneous Writes/Reads



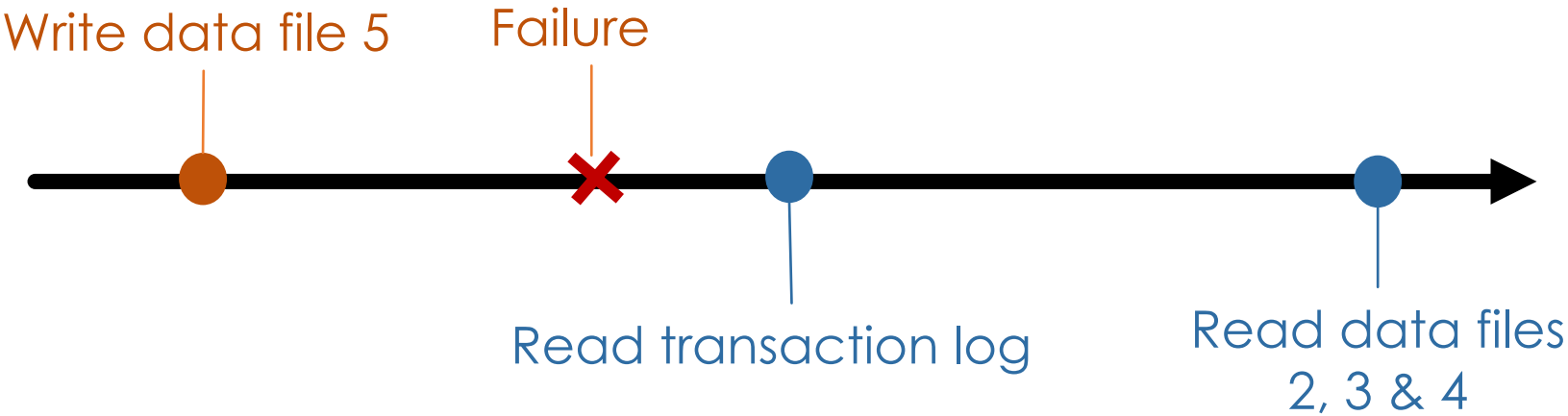
Failed Writes



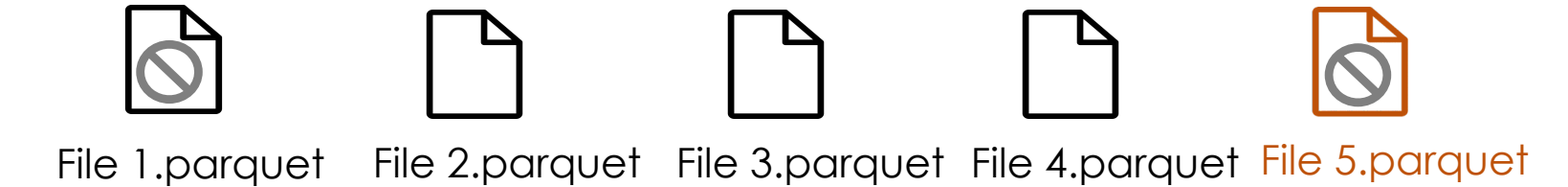
Writer
Process



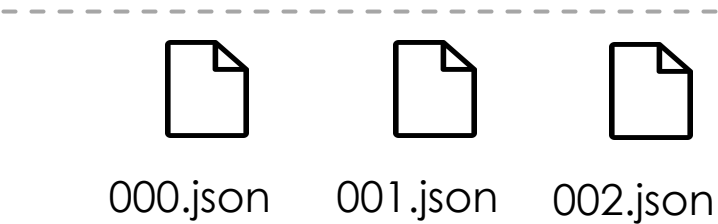
Reader
Process



Data Files



_delta_log



Delta Lake Advantages

- ▶ Brings ACID transactions to object storage
- ▶ Handle scalable metadata
- ▶ Full audit trail of all changes
- ▶ Builds upon standard data formats: Parquet + Json

Advanced Delta Lake Features

Learning Objectives

- ▶ Time travel
- ▶ Compacting Small Files and Indexing
- ▶ Vacuum

Time travel

- ▶ Audit data changes
- ▶ **DESCRIBE HISTORY** command

Time travel

- ▶ Query older versions of the data
- ▶ Using a timestamp
 - ▶ `SELECT * FROM my_table TIMESTAMP AS OF "2019-01-01"`
- ▶ Using a version number
 - ▶ `SELECT * FROM my_table VERSION AS OF 36`
 - ▶ `SELECT * FROM my_table@v36`

Time travel

- ▶ Rollback Versions

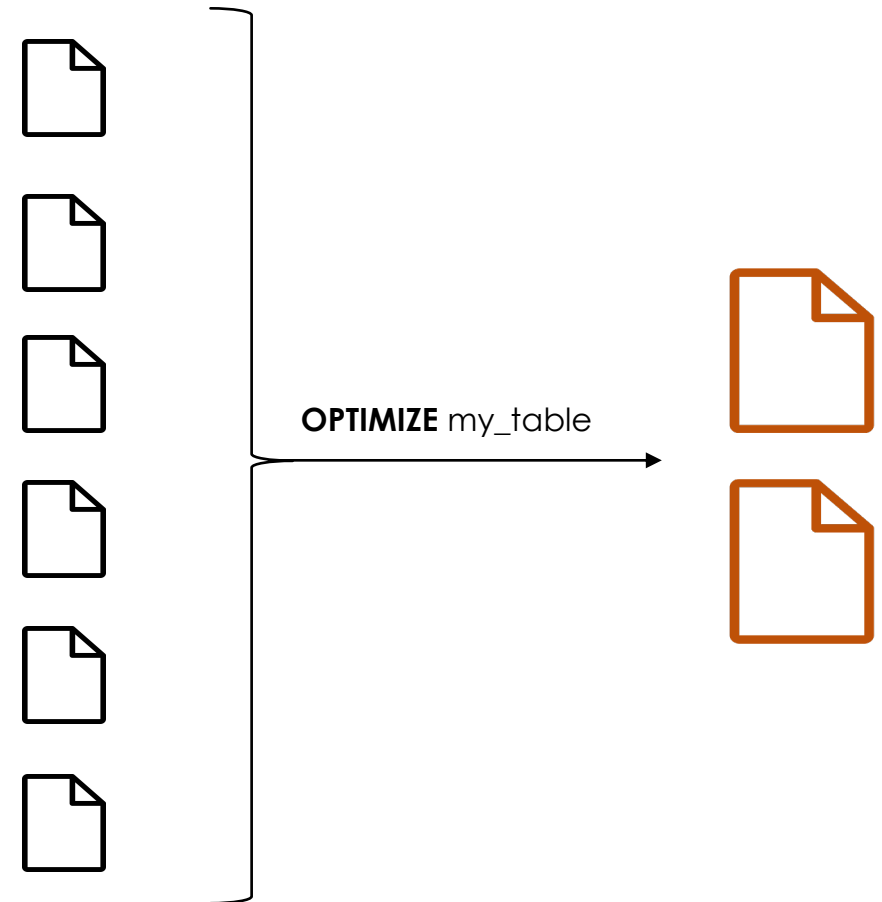
- ▶ **RESTORE TABLE** command:

- ▶ **RESTORE TABLE** my_table **TO TIMESTAMP AS OF** "2019-01-01"

- ▶ **RESTORE TABLE** my_table **TO VERSION AS OF** 36

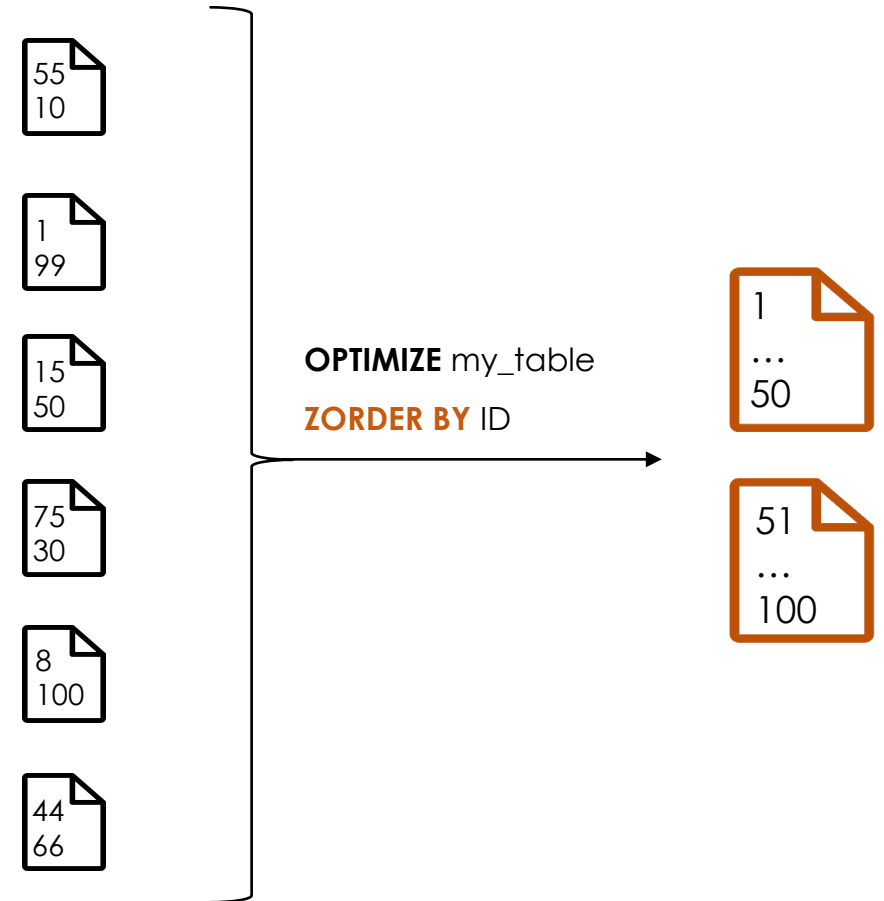
Compaction

- ▶ Compacting Small Files
- ▶ **OPTIMIZE** command:
 - ▶ **OPTIMIZE** my_table



Indexing

- ▶ Co-locate column information
- ▶ **OPTIMIZE** command:
 - ▶ OPTIMIZE my_table
ZORDER BY column_name



Vacuum a Delta table

- ▶ Cleaning up unused data files
 - ▶ uncommitted files
 - ▶ files that are no longer in in latest table state
- ▶ **VACUUM** command
 - ▶ **VACUUM** table_name [retention period]
 - ▶ Default retention period: 7 days
- ▶ **Note: Vacuum = no time travel**

Relational Entities on Databricks

Learning Objectives

- ▶ Databases
- ▶ Tables
- ▶ The impact of the LOCATION keyword

Database

- ▶ Databases = Schemas in Hive metastore
- ▶ CREATE DATABASE db_name
- ▶ CREATE SCHEMA db_name

Hive metastore

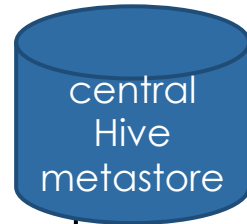
- ▶ repository of metadata
 - ▶ Databases
 - ▶ Tables
 - ▶ ...

```
CREATE TABLE table1;
```

```
CREATE TABLE table2;
```

```
...
```

Workspace



- **default**
- table_1
- table_2
- ...



Storage



dbfs:/user/hive/warehouse



table_1

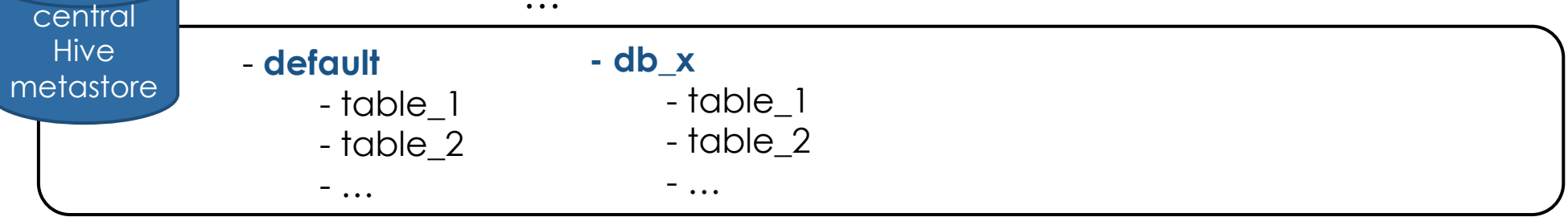
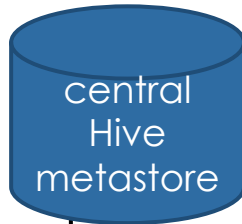


table_2

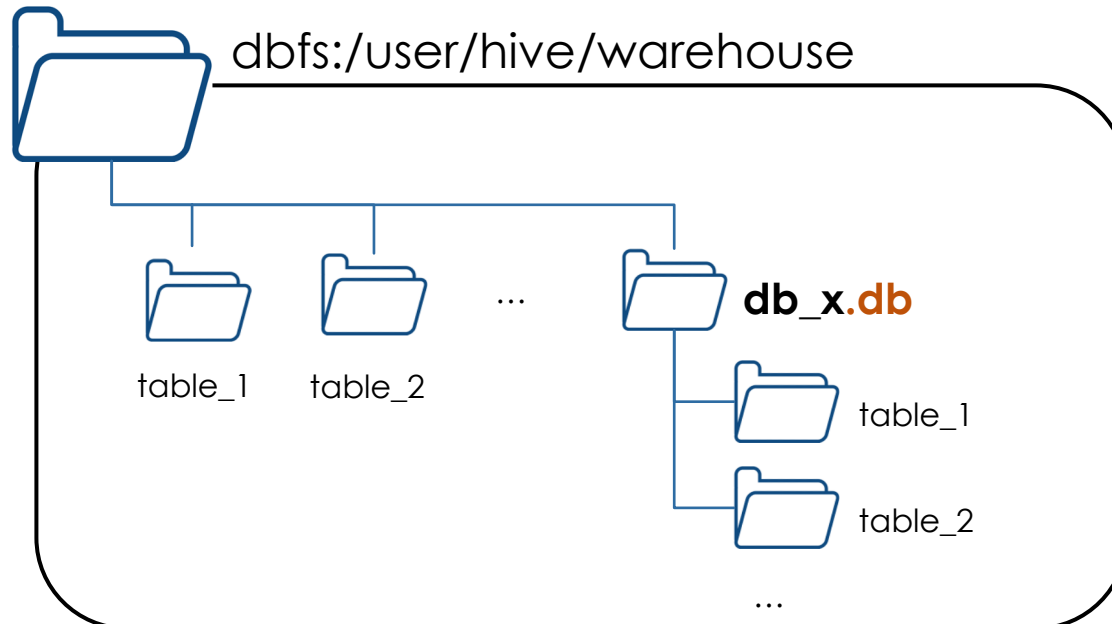
...

```
CREATE SCHEMA db_x
```

```
USE db_x;  
CREATE TABLE table1;  
CREATE TABLE table2;  
...
```



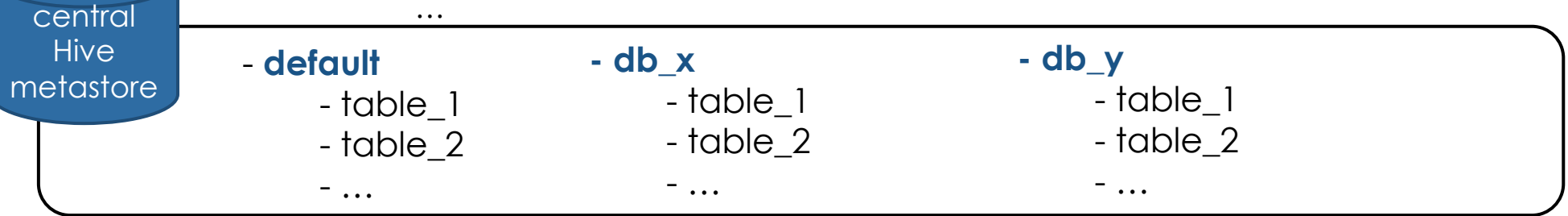
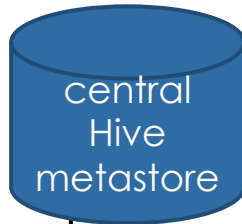
Workspace



Storage


```
CREATE SCHEMA db_y  
LOCATION 'dbfs:/custom/path/db_y.db'
```

```
USE db_y;  
CREATE TABLE table1;  
CREATE TABLE table2;  
...
```



Workspace



dbfs:/user/hive/warehouse



table_1



table_2

...



db_x.db



table_1



table_2

...



dbfs:/custom/path



db_y.db



table_1



table_2

...

Storage

Tables

Manged tables

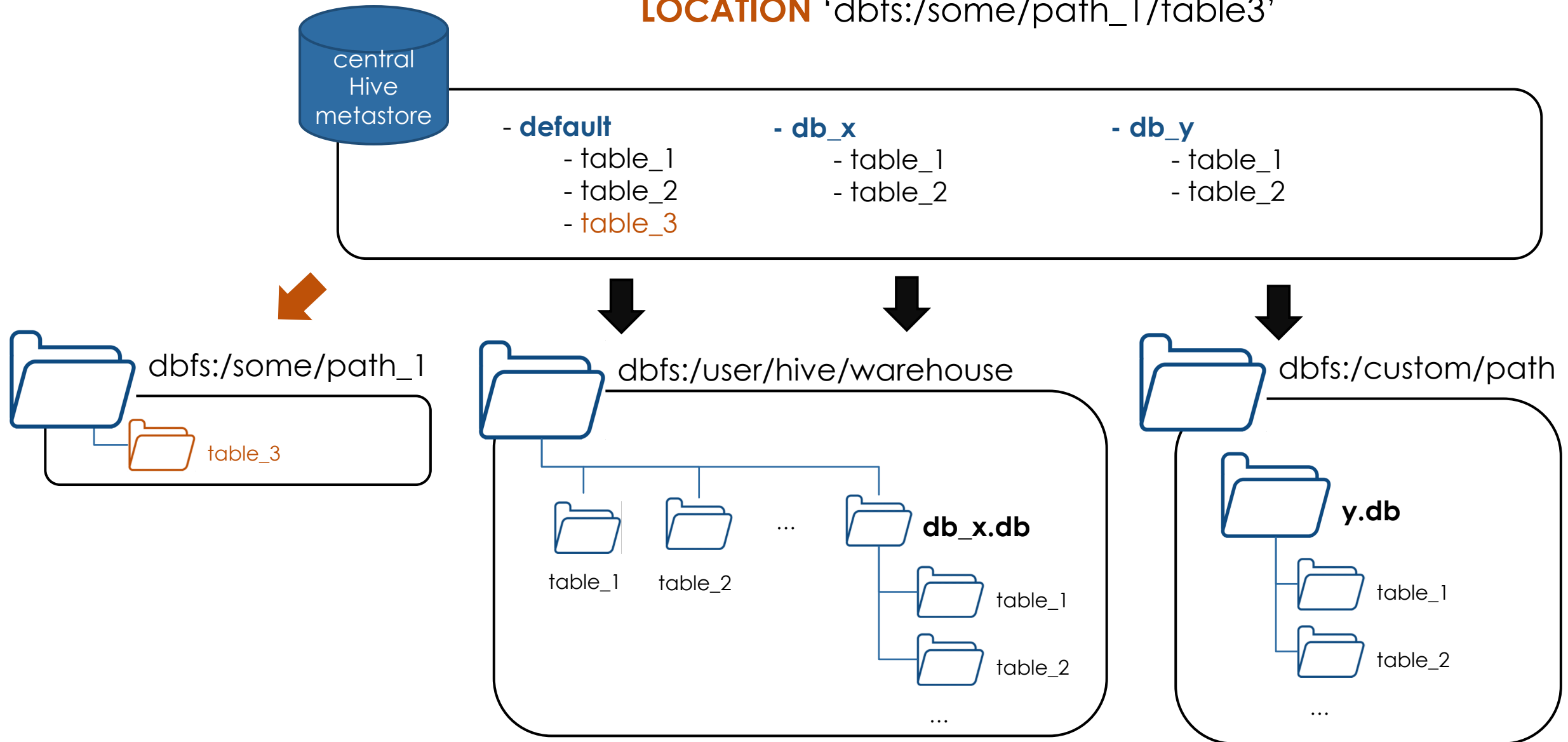
- ▶ Created under the database directory
 - ▶ CREATE TABLE table_name
- ▶ Dropping the table, delete the underlying data files

External tables

- ▶ Created outside the database directory
 - ▶ CREATE TABLE table_name
LOCATION 'path'
- ▶ Dropping the table, will **Not** delete the underlying data files

CREATE TABLE table3

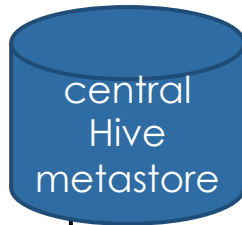
LOCATION 'dbfs:/some/path_1/table3'



USE db_x;

CREATE TABLE table3

LOCATION 'dbfs:/some/path_2/x_table3';



- **default**

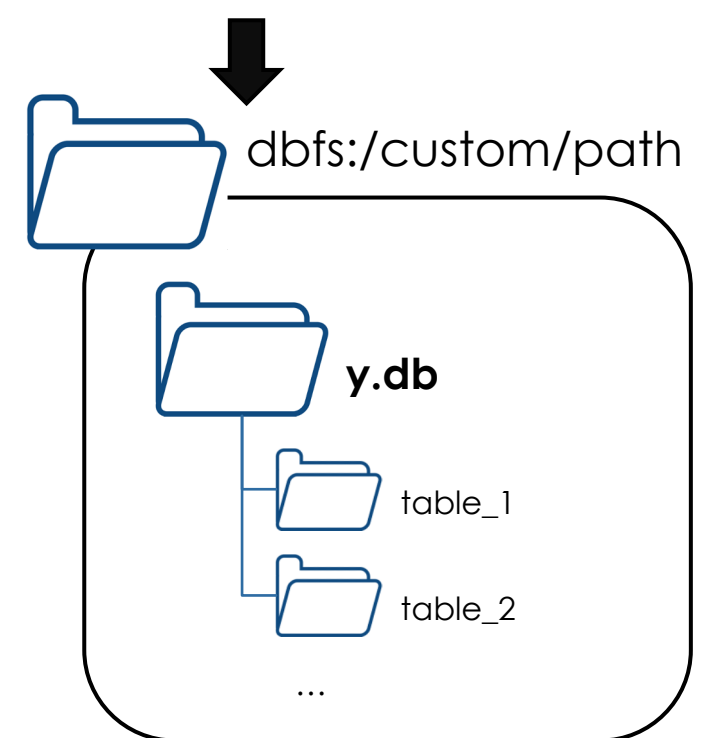
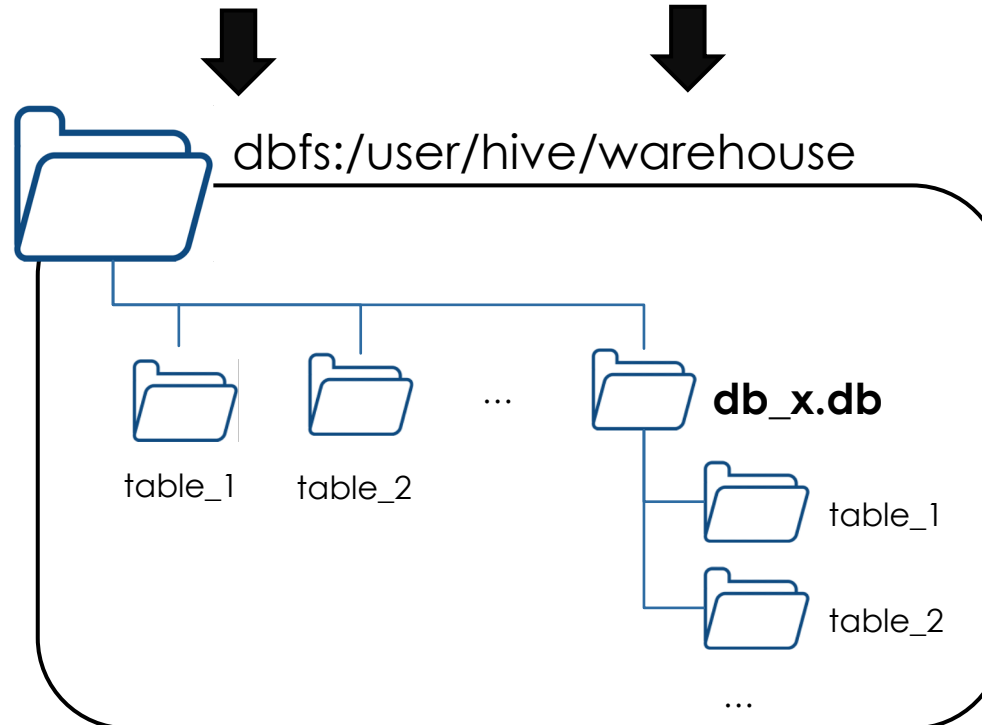
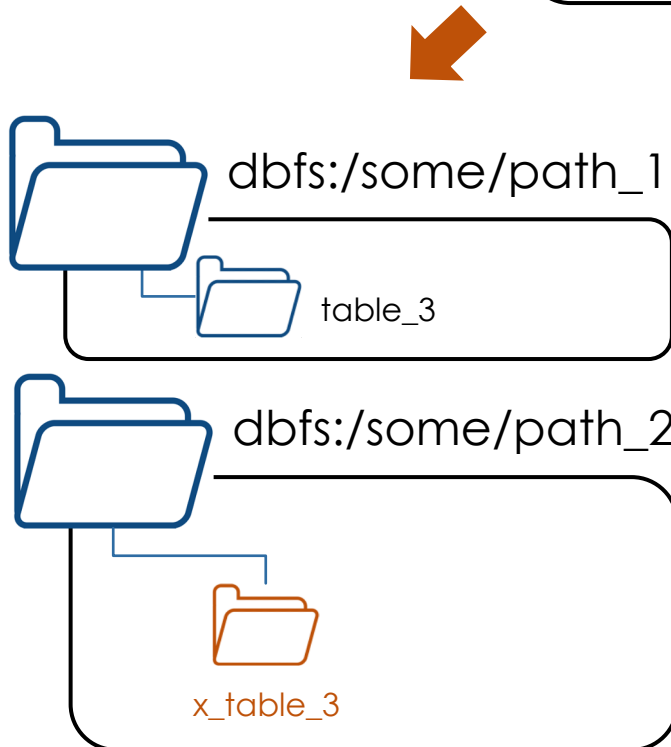
- table_1
- table_2
- **table_3**

- **db_x**

- table_1
- table_2
- **table_3**

- **db_y**

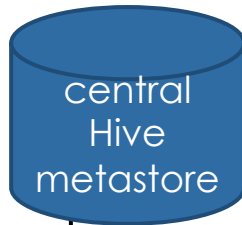
- table_1
- table_2



USE db_y;

CREATE TABLE table3

LOCATION 'dbfs:/some/path_2/y_table3';



- **default**

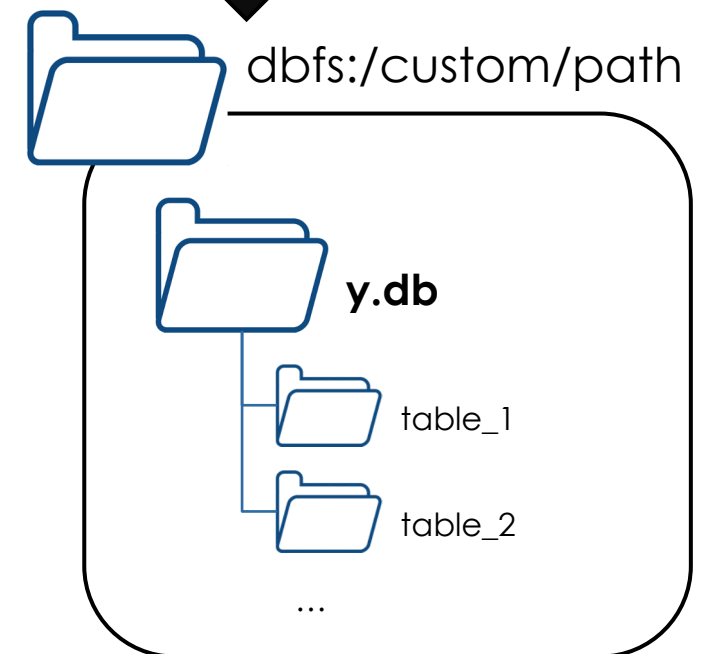
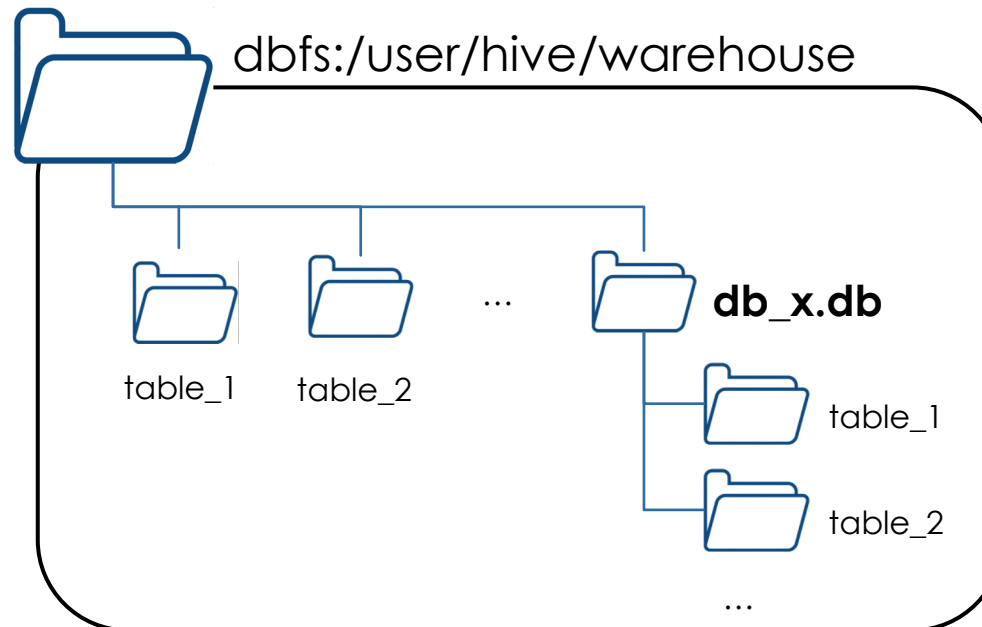
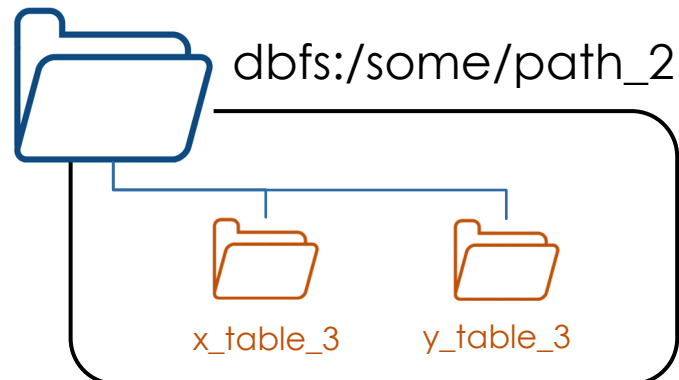
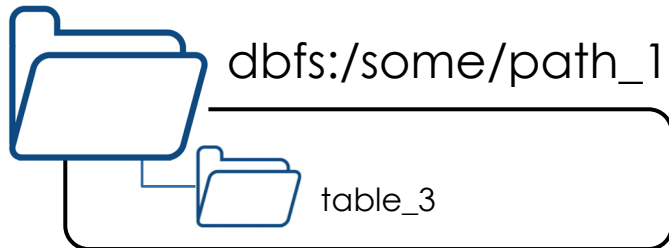
- table_1
- table_2
- **table_3**

- **db_x**

- table_1
- table_2
- **table_3**

- **db_y**

- table_1
- table_2
- **table_3**



Set Up Delta Tables

Learning Objectives

- ▶ CTAS statements
- ▶ Table Constraints
- ▶ Cloning Delta Lake Tables

CTAS

- ▶ CREATE TABLE _ AS SELECT statement
- ▶ **CREATE TABLE** table_1
AS SELECT * FROM table_2
- ▶ Automatically infer schema information from query results
 - ▶ do not support manual schema declaration

CTAS: Filtering and Renaming Columns

► **CREATE TABLE** table_1

AS SELECT col_1, col_3 AS new_col_3 FROM table_2

CTAS: Additional Options

► CREATE TABLE new_table
 COMMENT "Contains PII"
 PARTITIONED BY (city, birth_date)
 LOCATION '/some/path'
 AS SELECT id, name, email, birth_date, city FROM users

CREATE TABLE vs. CTAS

CREATE TABLE

```
CREATE TABLE table_1  
(col1 INT, col2 STRING, col3 DOUBLE)
```

- ▶ Manual schema declaration
- ▶ Create empty table
 - ▶ Need **INSERT INTO** statement

CTAS

```
CREATE TABLE table_1  
AS SELECT col1, col2, col3 FROM table_2
```

- ▶ Do **Not** support manual schema declaration
 - ▶ Automatically infer schema
- ▶ Table created with data

Table Constraints

- ▶ NOT NULL constraints
- ▶ CHECK constraints
- ▶ **ALTER TABLE** table_name **ADD CONSTRAINT** constraint_name constraint_details
- ▶ **ALTER TABLE** orders **ADD CONSTRAINT** valid_date **CHECK** (date > '2020-01-01');

Cloning Delta Lake Tables

- ▶ DEEP CLONE
- ▶ SHALLOW CLONE

Deep Cloning

- ▶ Fully copies data + metadata from a source table to a target
- ▶ **CREATE TABLE** table_clone
DEEP CLONE source_table
- ▶ Can sync changes
- ▶ Take quite a while for large datasets

Shallow Cloning

- ▶ Quickly create a copy of a table
 - ▶ Just copy the Delta transaction logs
- ▶ **CREATE TABLE** table_clone
SHALLOW CLONE source_table

Cloning Delta Lake Tables

- ▶ Useful to set up tables for testing in development.
- ▶ In either case, data modifications will not affect the source

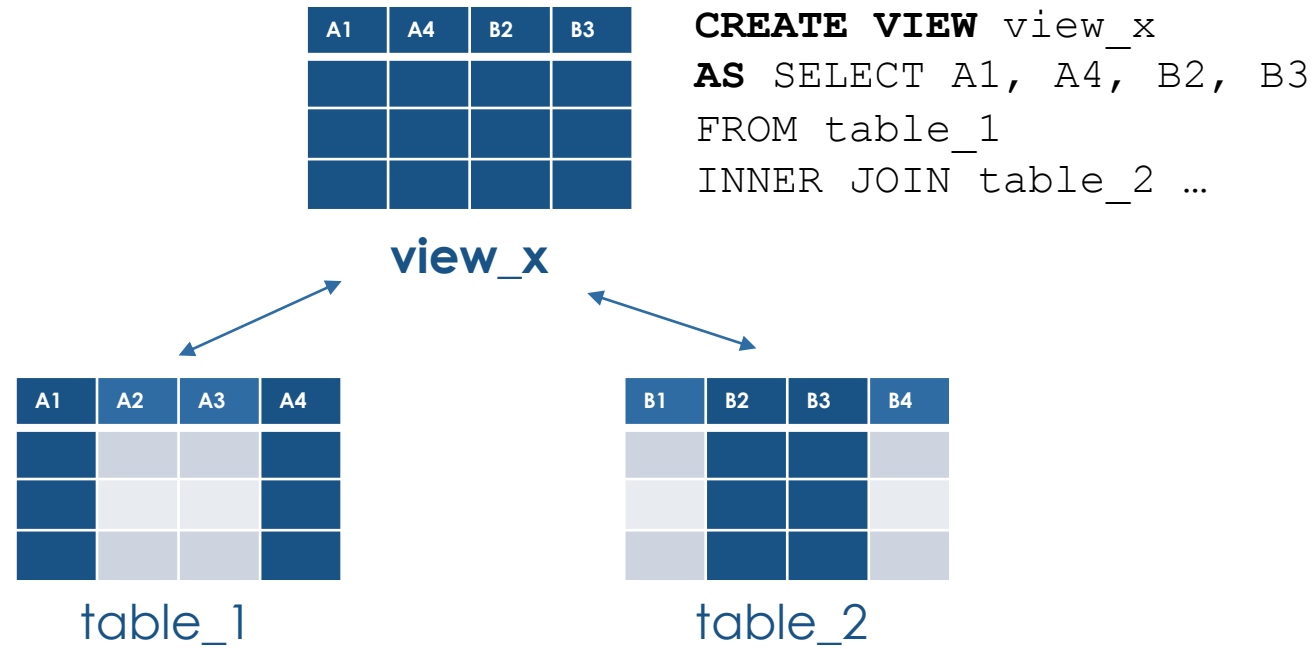
Views

Learning Objectives

- ▶ Views
- ▶ Types of views

View

► Logical query against source tables



Types of views

1. (Stored) Views
2. Temporary views
3. Global Temporary views

1- (Stored) Views

- ▶ Persisted objects
- ▶ **CREATE VIEW** view_name
AS query

2- Temporary view

- ▶ Session-scoped view
- ▶ **CREATE TEMP VIEW** view_name
AS query

Spark Session

- ▶ Opening a new notebook
- ▶ Detaching and reattaching to a cluster
- ▶ Installing a python package
- ▶ Restarting a cluster

3- Global Temporary views

- ▶ Cluster-scoped view
- ▶ **CREATE GLOBAL TEMP VIEW** view_name
AS query
- ▶ **SELECT * FROM global_temp.view_name**

Views comparison

(Stored) Views

- ▶ Persisted in DB
- ▶ Dropped only by **DROP VIEW**
- ▶ CREATE VIEW

Temp views

- ▶ Session-scoped
- ▶ dropped when session ends
- ▶ CREATE **TEMP** VIEW

Global Temp views

- ▶ Cluster-scoped
- ▶ dropped when cluster restarted
- ▶ CREATE **GLOBAL TEMP** VIEW

Querying Files

Learning Objectives

- ▶ Querying data files directly
- ▶ Extract files as raw contents
- ▶ Configure options of external sources
- ▶ Use CTAS statements to create Delta Lake tables

Querying Files Directly

```
SELECT * FROM file format. /path/to/file
```

Self-describing formats

- json
- parquet
- ...

Non self-describing Formats

- CSV
- TSV
- ...

Single file
file_2022.json

Multiple files
file_*.json

complete directory
/path/dir

Example: JSON

```
SELECT * FROM json.`/path/file_name.json`
```

Raw data

- ▶ Extract text files as raw strings
 - ▶ Text-based files (JSON, CSV, TSV, and TXT formats)
 - ▶ `SELECT * FROM text.`/path/to/file``
- ▶ Extract files as raw bytes
 - ▶ Images or unstructured data
 - ▶ `SELECT * FROM binaryFile.`/path/to/file``

CTAS: Registering Tables from Files

- ▶ **CREATE TABLE** table_name
AS SELECT * FROM file_format.`/path/to/file`
- ▶ Automatically infer schema information from query results
 - ▶ Do **Not** support manual schema declaration.
 - ▶ Useful for external data ingestion with well-defined schema
- ▶ Do **Not** support file options

Registering Tables on External Data Sources

- ▶ **CREATE TABLE** table_name
(col_name1 col_type1, ...)
USING data_source
OPTIONS (key1 = val1, key2 = val2, ...)
LOCATION = path
- ▶ External table
- ▶ Non-Delta table!

Example: CSV

```
► CREATE TABLE table_name  
  (col_name1 col_type1, ...)  
  USING CSV  
  OPTIONS (header = "true",  
            delimiter = ";")  
  LOCATION = path
```

Example: Database

► **CREATE TABLE** table_name
 (col_name1 col_type1, ...)
USING JDBC
OPTIONS (url = "jdbc:sqlite://hostname:port",
 dbtable = "database.table",
 user = "username",
 password = "pwd")

Limitation

- ▶ It's Not Delta table!
- ▶ We can not expect the performance guarantees associated with Delta Lake and Lakehouse
- ▶ Having a huge database table

Solution

- ▶ **CREATE TEMP VIEW** temp_view_name (col_name1 col_type1, ...)
USING data_source
OPTIONS (key1 = "val1", key2 = "val2", ..., path = "/path/to/files")
- ▶ **CREATE TABLE** table_name
AS SELECT * FROM temp_view_name

Structured Streaming

Learning Objectives

- ▶ Process streaming data
- ▶ DataStreamReader
- ▶ DataStreamWriter

Data Stream

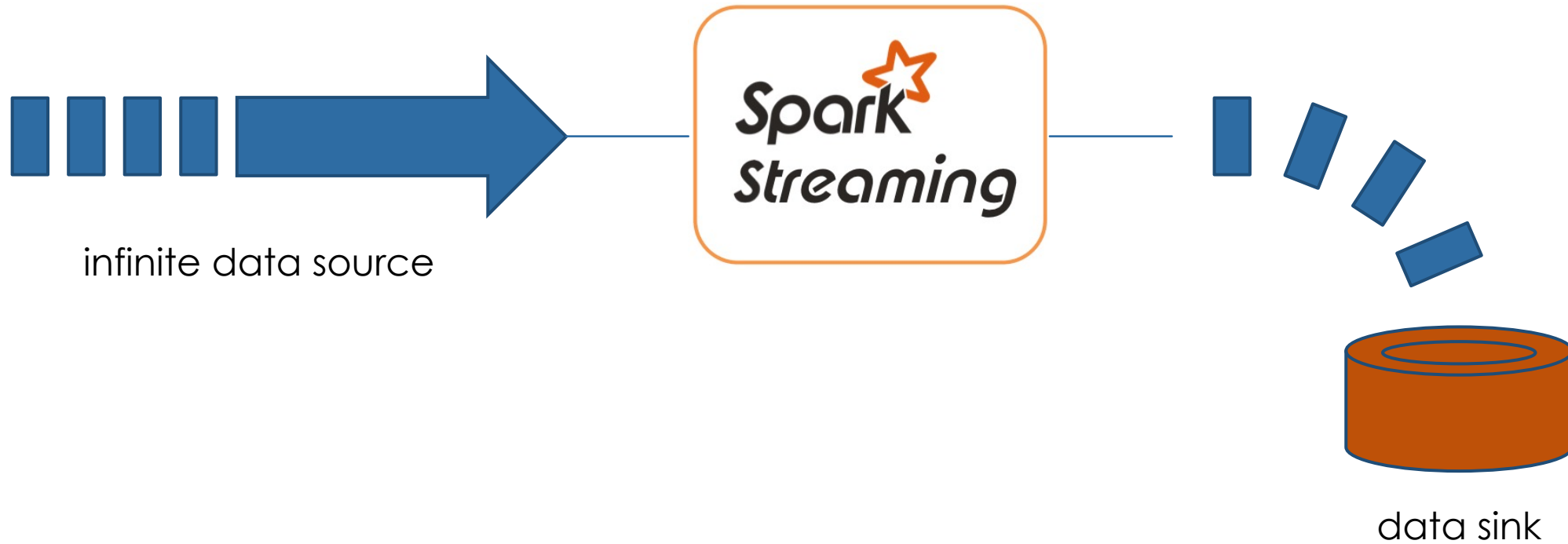
- ▶ Any data source that grows over time
- ▶ New files landing in cloud storage
- ▶ Updates to a database captured in a CDC feed
- ▶ Events queued in a pub/sub messaging feed

Processing Data Stream

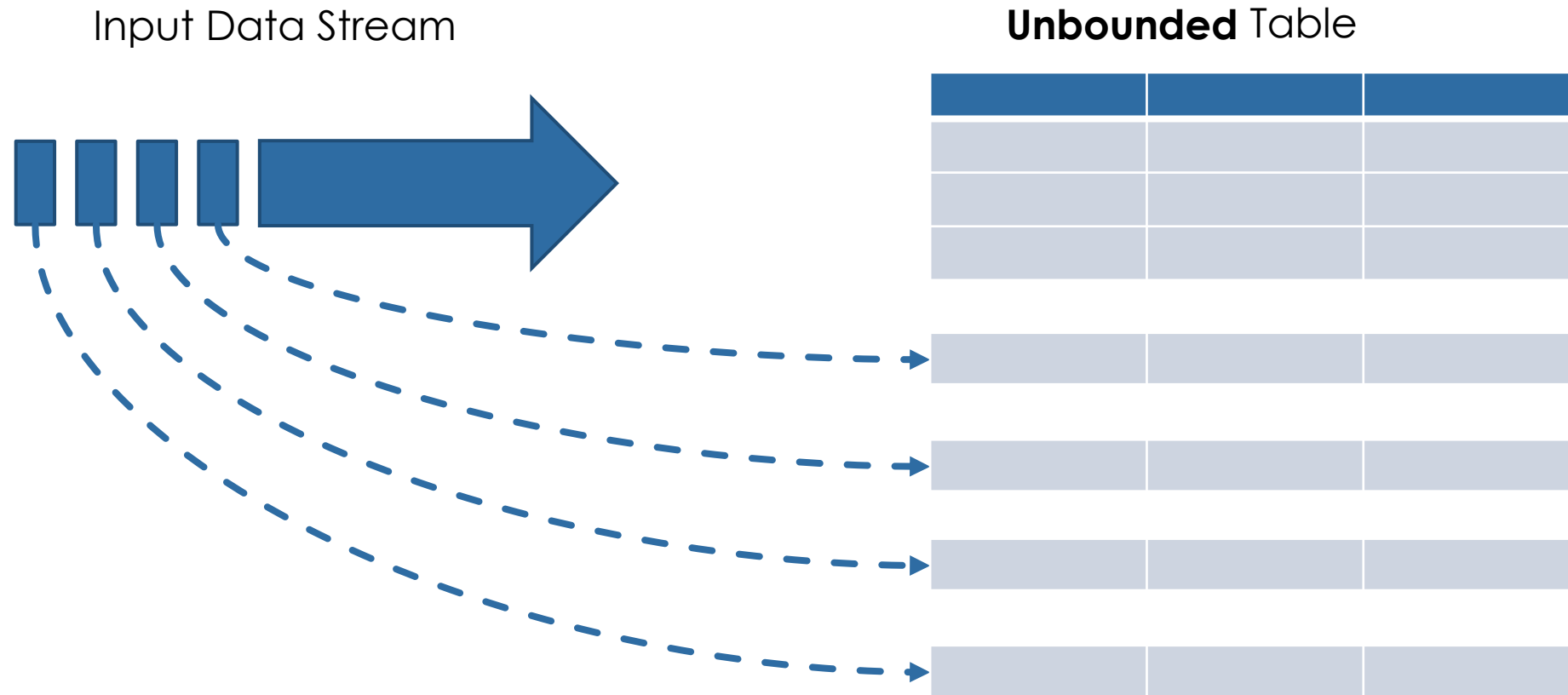
► 2 approaches:

1. Reprocess the entire source dataset each time
2. Only process those new data added since last update
 - Structured Streaming

Spark Structured Streaming



Treating Infinite Data as a Table



Input Streaming Table

Input_Table

Output_Table



```
streamDF = spark.readStream  
            .table("Input_Table")
```

```
streamDF.writeStream  
            .trigger(processingTime="2 minutes")  
            .outputMode("append")  
            .option("checkpointLocation", "/path")  
            .table("Output_Table")
```

Trigger Intervals

```
streamDF.writeStream  
  .trigger(processingTime="2 minutes")  
  .outputMode("append")  
  .option("checkpointLocation", "/path")  
  .table("Output_Table")
```

Trigger	Method call	Behavior
Unspecified		Default: processingTime="500ms"
Fixed interval	<code>.trigger(processingTime="5 minutes")</code>	Process data in micro-batches at the user-specified intervals
Triggered batch	<code>.trigger(once=True)</code>	Process all available data in a single batch, then stop
Triggered micro-batches	<code>.trigger(availableNow=True)</code>	Process all available data in multiple micro-batches, then stop

Output Modes

```
streamDF.writeStream  
    .trigger(processingTime="2 minutes")  
    .outputMode("append")  
    .option("checkpointLocation", "/path")  
    .table("Output_Table")
```

Mode	Method call	Behavior
Append (Default)	<code>.outputMode("append")</code>	Only newly appended rows are incrementally appended to the target table with each batch
Complete	<code>.outputMode("complete")</code>	The target table is overwritten with each batch

Checkpointing

```
streamDF.writeStream  
    .trigger(processingTime="2 minutes")  
    .outputMode("append")  
    .option("checkpointLocation", "/path")  
    .table("Output_Table")
```

- ▶ Store stream state
- ▶ Track the progress of your stream processing
- ▶ Can **Not** be shared between separate streams

Guarantees

1. Fault Tolerance

- ▶ Checkpointing + Write-ahead logs
 - ▶ record the offset range of data being processed during each trigger interval.

2. Exactly-once guarantee

- ▶ Idempotent sinks

Unsupported Operations

- ▶ Some operations are not supported by streaming DataFrame
 - ▶ Sorting
 - ▶ Deduplication
- ▶ Advanced methods
 - ▶ Windowing
 - ▶ Watermarking

Incremental Data Ingestion from Files

Learning Objectives

- ▶ What is incremental data Ingestion from file
- ▶ COPY INTO
- ▶ Auto Loader

Incremental Data Ingestion

- ▶ Loading new data files encountered since the last ingestion
- ▶ Reduces redundant processing
- ▶ 2 mechanisms:
 - ▶ COPY INTO
 - ▶ Auto loader

COPY INTO

- ▶ SQL command
- ▶ Idempotently and incrementally load new data files
 - ▶ Files that have already been loaded are skipped.

COPY INTO

```
► COPY INTO my_table  
  FROM '/path/to/files'  
  FILEFORMAT = <format>  
  FORMAT_OPTIONS (<format options>  
  COPY_OPTIONS (<copy options>);
```

Example

```
► COPY INTO my_table  
FROM '/path/to/files'  
FILEFORMAT = CSV  
FORMAT_OPTIONS ('delimiter' = '|',  
                  'header' = 'true')  
COPY_OPTIONS ('mergeSchema' = 'true')
```

Auto loader

- ▶ Structured Streaming
- ▶ Can process billions of files
- ▶ Support near real-time ingestion of millions of files per hour.

Auto loader Checkpointing

- ▶ Store metadata of the discovered files
- ▶ Exactly-once guarantees
- ▶ Fault tolerance

Auto Loader in PySpark API

spark.readStream

.format("cloudFiles")

.option("cloudFiles.format", <source_format>)

.load('/path/to/files')

.writeStream

.option("checkpointLocation", <checkpoint_directory>)

.table(<table_name>)

Auto Loader + Schema

spark.readStream

.format("cloudFiles")

.option("cloudFiles.format", <source_format>)

.option("cloudFiles.schemaLocation", <schema_directory>)

.load('/path/to/files')

.writeStream

.option("checkpointLocation", <checkpoint_directory>)

.option("mergeSchema", "true")

.table(<table_name>)

COPY INTO vs. Auto Loader

COPY INTO

- ▶ Thousands of files
- ▶ Less efficient at scale

Auto Loader

- ▶ Millions of files
- ▶ Efficient at scale

Multi-Hop Architecture

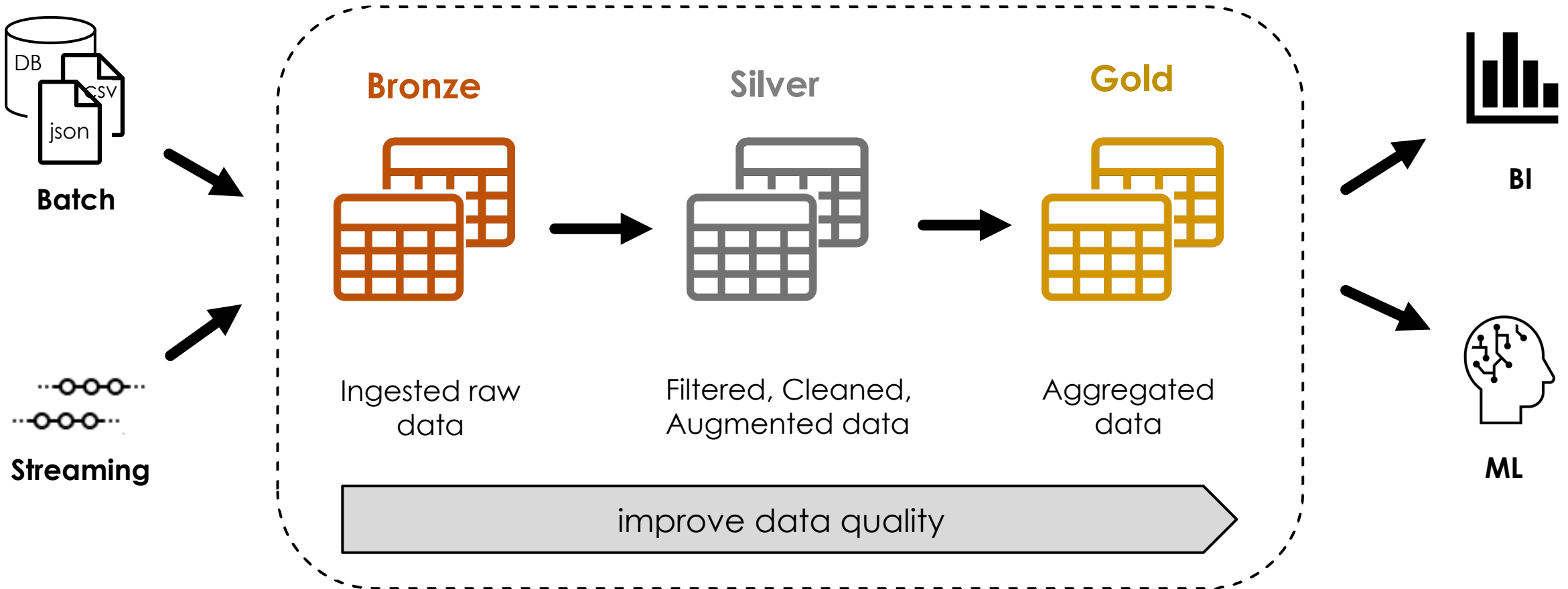
Learning Objectives

- ▶ Incremental Multi-Hop pipeline
- ▶ Describe Bronze, Silver, and Gold tables

Multi-Hop Architecture

- ▶ Medallion Architecture
- ▶ Organize data in multi-layered approach
- ▶ Incrementally improving the structure and quality of data as it flows through each layer

Multi-Hop Architecture



Benefits

- ▶ Simple data model
- ▶ Enables incremental ETL
- ▶ Combine streaming and batch workloads in unified pipeline
- ▶ Can recreate your tables from raw data at any time

Change Data Capture

Learning Objectives

- ▶ Definition of Change data capture
- ▶ Processing Change data capture with DLT

Change Data Capture

- ▶ CDC: Process of identifying changes made to data in the source and delivering those changes to the target



Row-Level changes

- ▶ Inserting new records
- ▶ Updating existing records
- ▶ Deleteing existing records

CDC Feed

► Row data + metadata

Country ID	Country	Vaccination Rate	sequence	operation
FR	France	0,74	2022-11-01 07:00	UPDATE
FR	France	0.75	2022-11-01 08:00	UPDATE
CA			2022-11-01 00:00	DELETE
US	USA	0,5	2022-11-01 00:00	INSERT
IN	India	0.66	2022-11-01 00:00	INSERT

CDC Feed



Country ID	Country	Vaccination Rate
FR	France	0,6
CA	Canada	0,71

Target table

CDC Feed

► Row data + metadata

Country ID	Country	Vaccination Rate	sequence	operation
FR	France	0.74	2022-11-01 07:00	UPDATE
FR	France	0.75	2022-11-01 08:00	UPDATE
CA			2022-11-01 00:00	DELETE
US	USA	0.5	2022-11-01 00:00	INSERT
IN	India	0.66	2022-11-01 00:00	INSERT

CDC Feed



Country ID	Country	Vaccination Rate
FR	France	0.75
US	USA	0.5
IN	India	0.66

Target table

CDC with DLT

► APPLY CHANGES INTO command

```
APPLY CHANGES INTO LIVE.target_table  
FROM STREAM(LIVE.cdc_feed_table)  
KEYS (key_field)  
APPLY AS DELETE WHEN operation_field = "DELETE"  
SEQUENCE BY sequence_field  
COLUMNS *
```

APPLY CHANGES INTO: **Pros**

- ▶ Orders late-arriving records using the sequencing key
- ▶ Default assumption is that rows will contain inserts and updates
- ▶ Can optionally apply deletes (APPLY AS DELETE WHEN condition)
- ▶ Specify one or many fields as the primary key for a table
- ▶ Specify columns to ignore with the EXCEPT keyword
- ▶ Support applying changes as SCD Type 1 (default) or SCD Type 2

APPLY CHANGES INTO: **Cons**

- ▶ Breaks the append-only requirements for streaming table sources
 - ▶ Can Not perform streaming queries against the table

Data object privileges

Learning Objectives

- ▶ Data governance model
- ▶ Managing Permissions for Data objects

Data governance model

- ▶ Programmatically grant, deny, and revoke access to data objects

GRANT Privilege **ON** Object <object-name> **TO** <user or group>

- ▶ **GRANT** **SELECT** **ON** **TABLE** my_table **TO** user_1@company.com

Data objects

GRANT Privilege **ON** Object <object-name> **TO** <user or group>

Object	Scope
CATALOG	controls access to the entire data catalog.
SCHEMA	controls access to a database.
TABLE	controls access to a managed or external table.
VIEW	controls access to SQL views.
FUNCTION	controls access to a named function.
ANY FILE	controls access to the underlying filesystem.

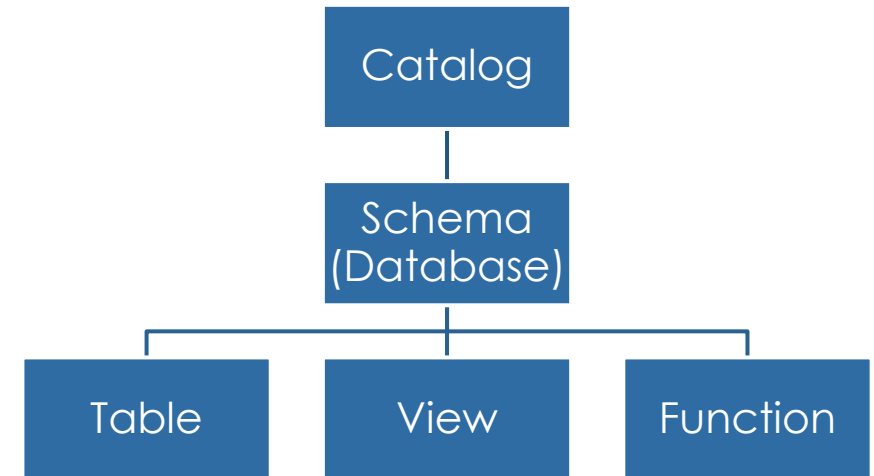
Privileges

GRANT **Privilege** **ON** **Object** <object-name> **TO** <user or group>

Privilege	Ability
SELECT	read access to an object.
MODIFY	add, delete, and modify data to or from an object.
CREATE	create an object
READ_METADATA	view an object and its metadata.
USAGE	No effect! required to perform any action on a database object.
ALL PRIVILEGES	gives all privileges

Granting Privileges by Role

Role	Can grant access privileges for
Databricks administrator	All objects in the catalog and the underlying filesystem.
Catalog owner	All objects in the catalog.
Database owner	All objects in the database.
Table owner	Only the table
...	...



More operations

- ▶ Grant
- ▶ DENY
- ▶ REVOKE

- ▶ SHOW GRANTS

Unity Catalog

Learning Objectives

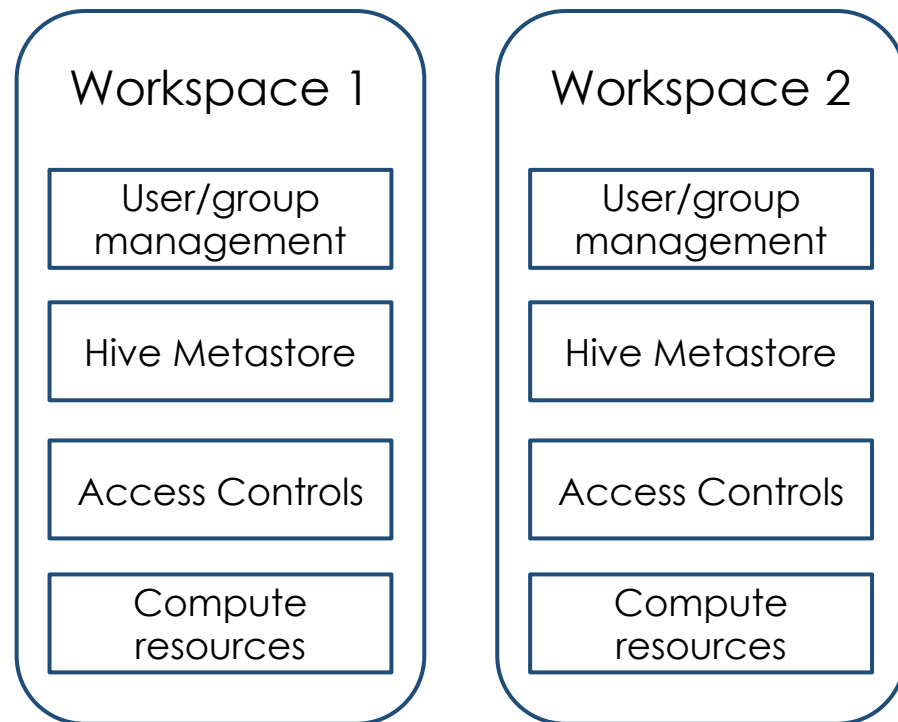
- ▶ What is Unity Catalog
- ▶ 3-level namespace
- ▶ Security model for governing data objects

Unity Catalog

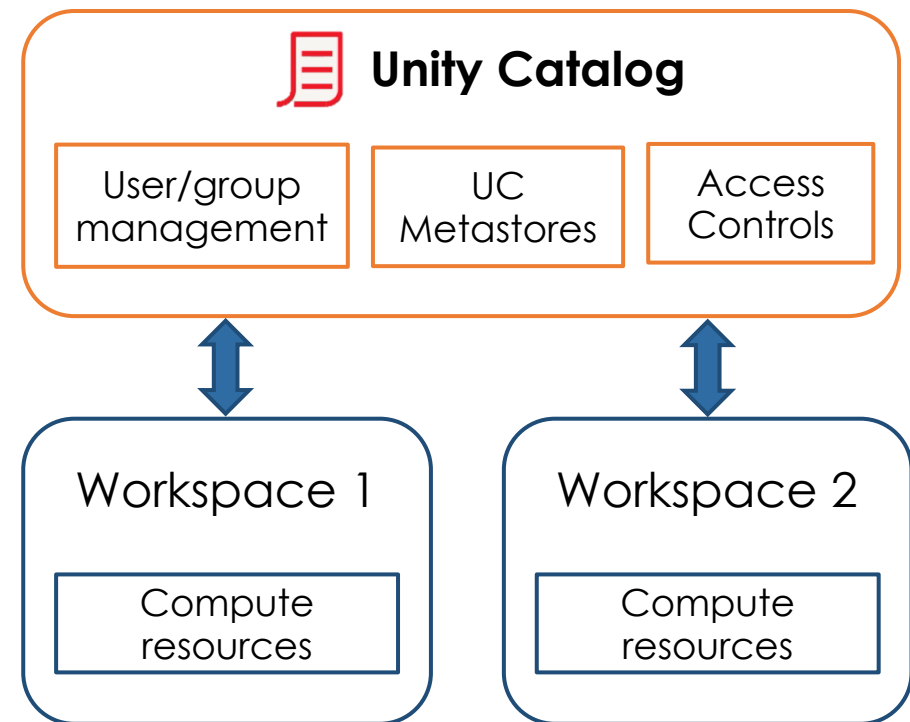
- ▶ Centralized governance solution across all your workspaces on any cloud.
- ▶ Unify governance for all data and AI assets
 - ▶ files, tables, machine learning models and dashboards
 - ▶ based on SQL

Architecture

Before Unity Catalog



With Unity Catalog



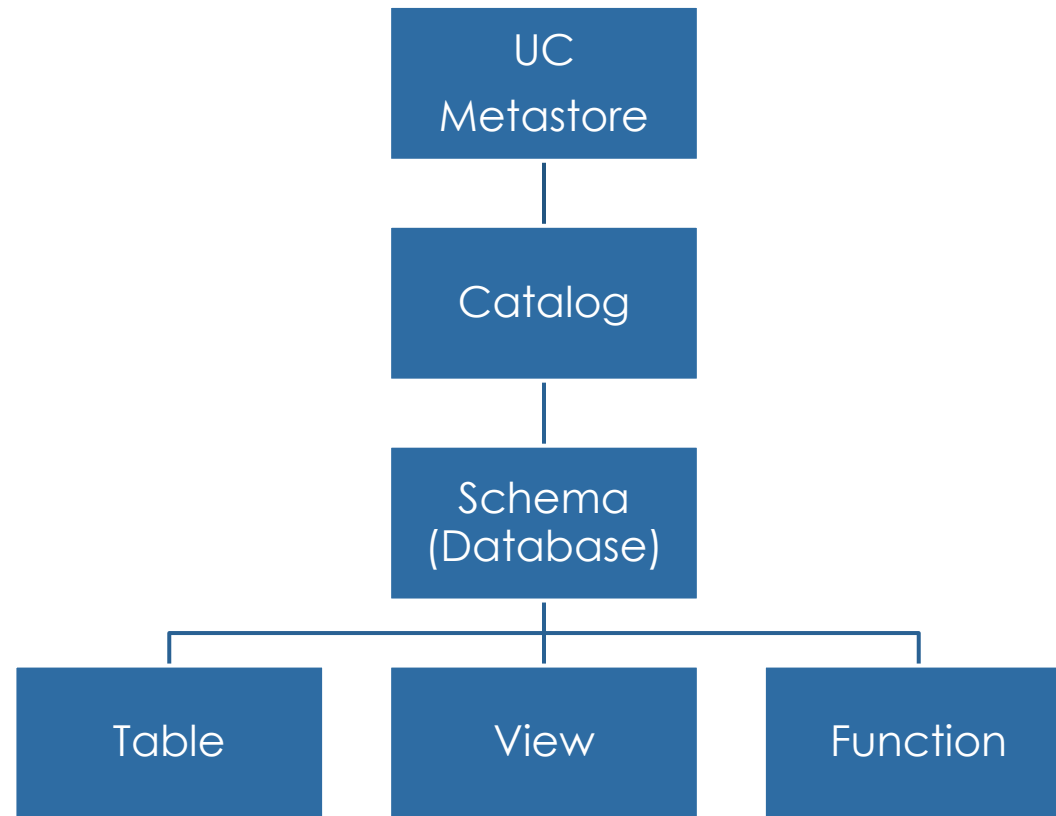
UC 3-level namespace

```
SELECT * FROM schema.table
```

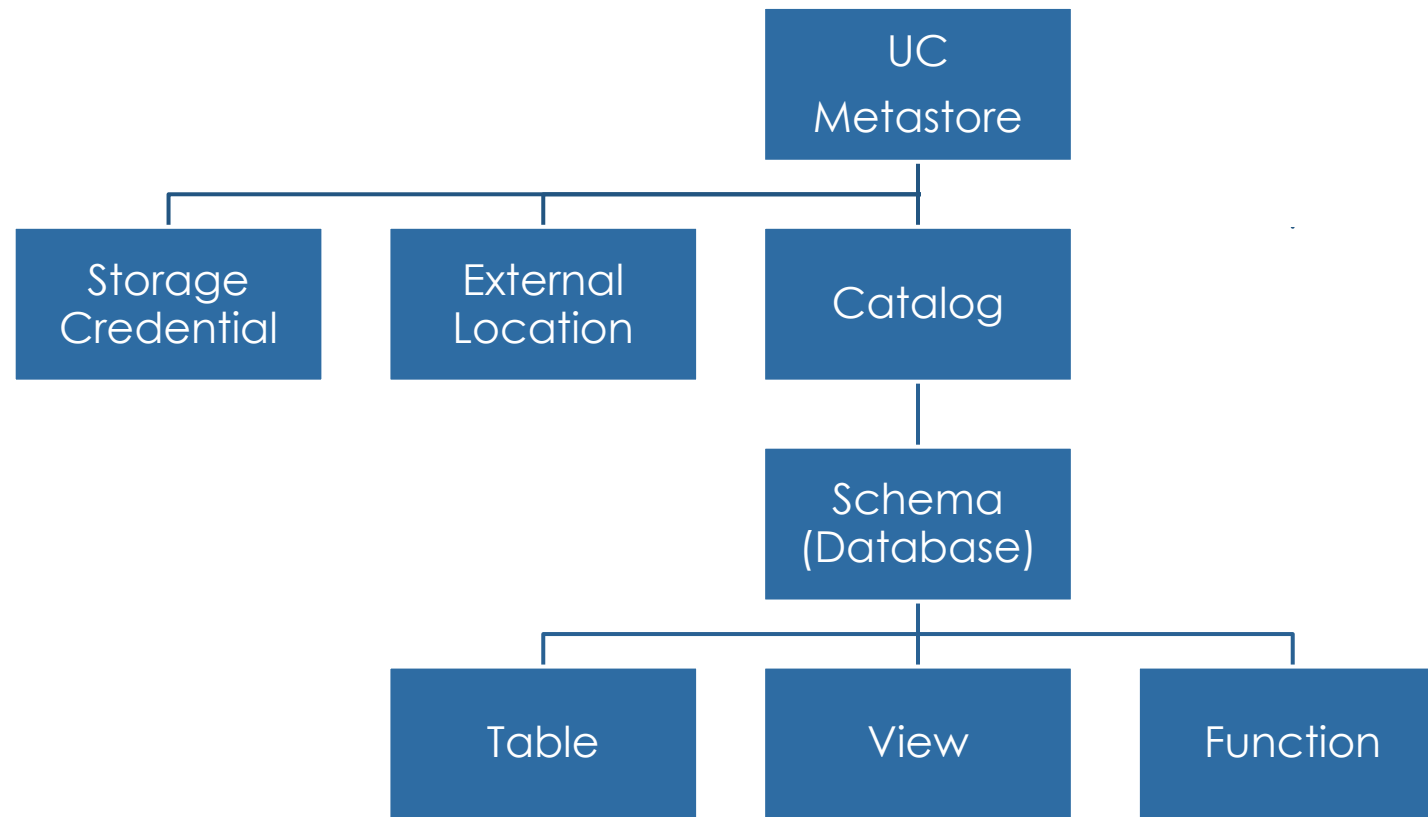


```
SELECT * FROM catalog.schema.table
```

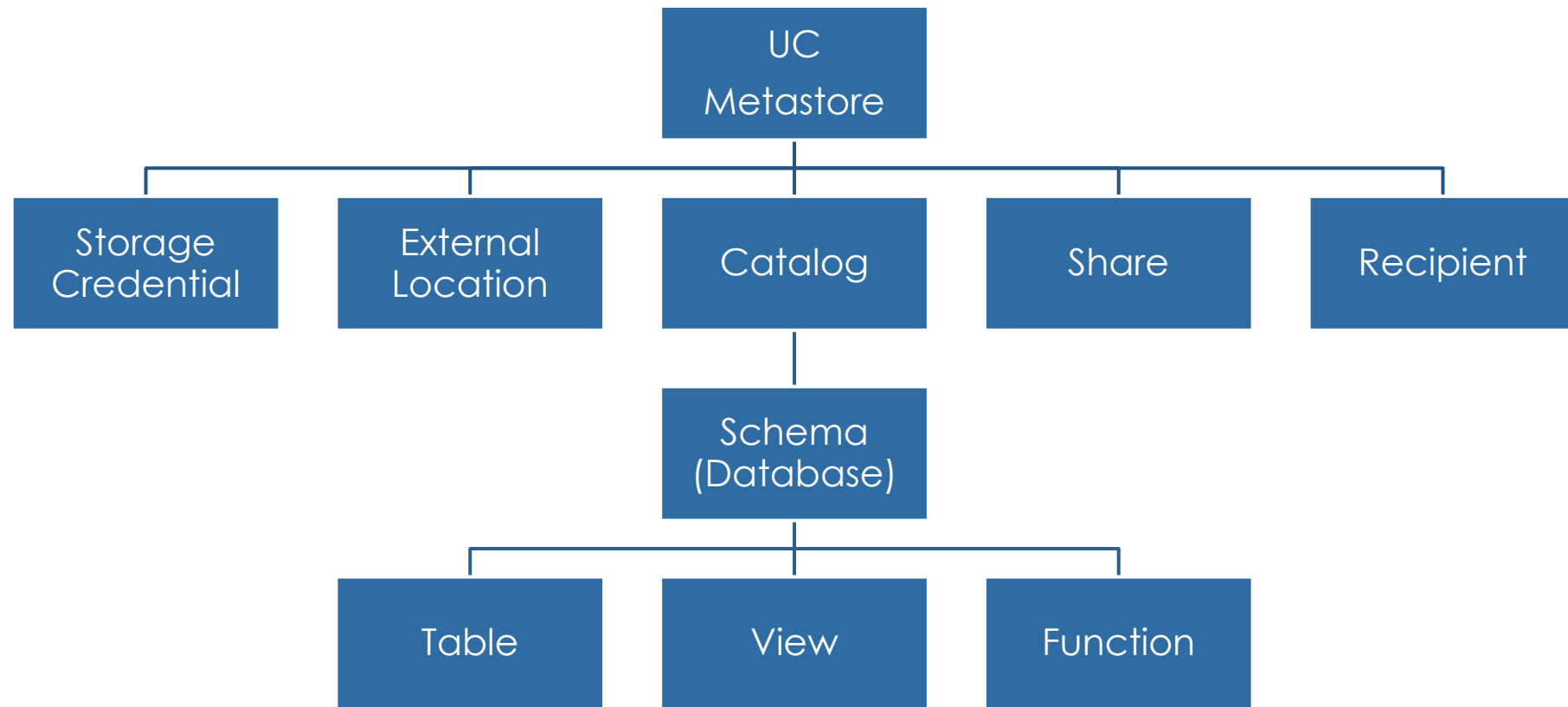
UC hierarchy



UC hierarchy



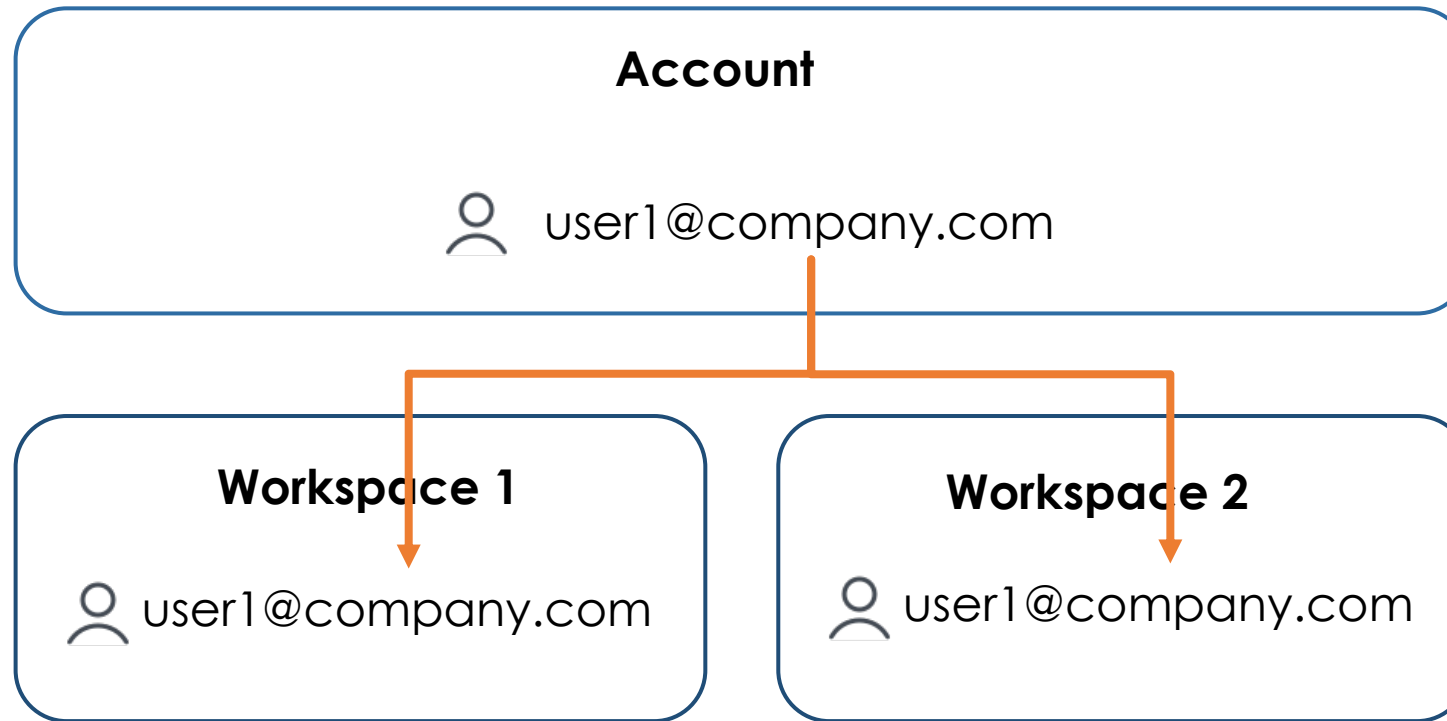
UC hierarchy



Identities

- ▶ **Users:** identified by e-mail addresses
 - ▶ Account administrator
- ▶ **Service Principles:** identified by Application IDs
 - ▶ Service Principles with administrative privilege
- ▶ **Groups:** grouping Users and Service Principles
 - ▶ Nested groups

Identity Federation



Privileges

- ▶ CREATE
- ▶ USAGE
- ▶ SELECT
- ▶ MODIFY
- ▶ READ FILES
- ▶ WRITE FILES
- ▶ EXECUTE

Security model

GRANT **Privilege** ON **Securable_Object** TO **Principal**

Privileges

CREATE

USAGE

SELECT

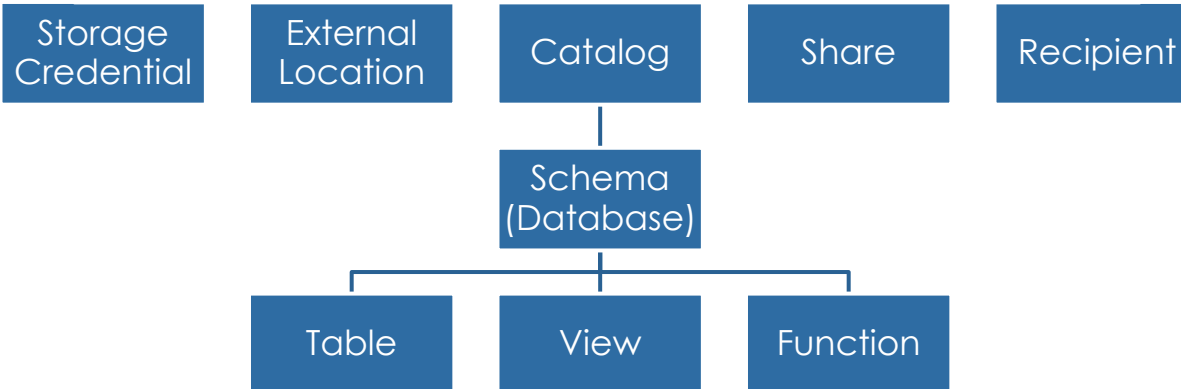
MODIFY

READ FILES

WRITE FILES

EXECUTE

Securable objects



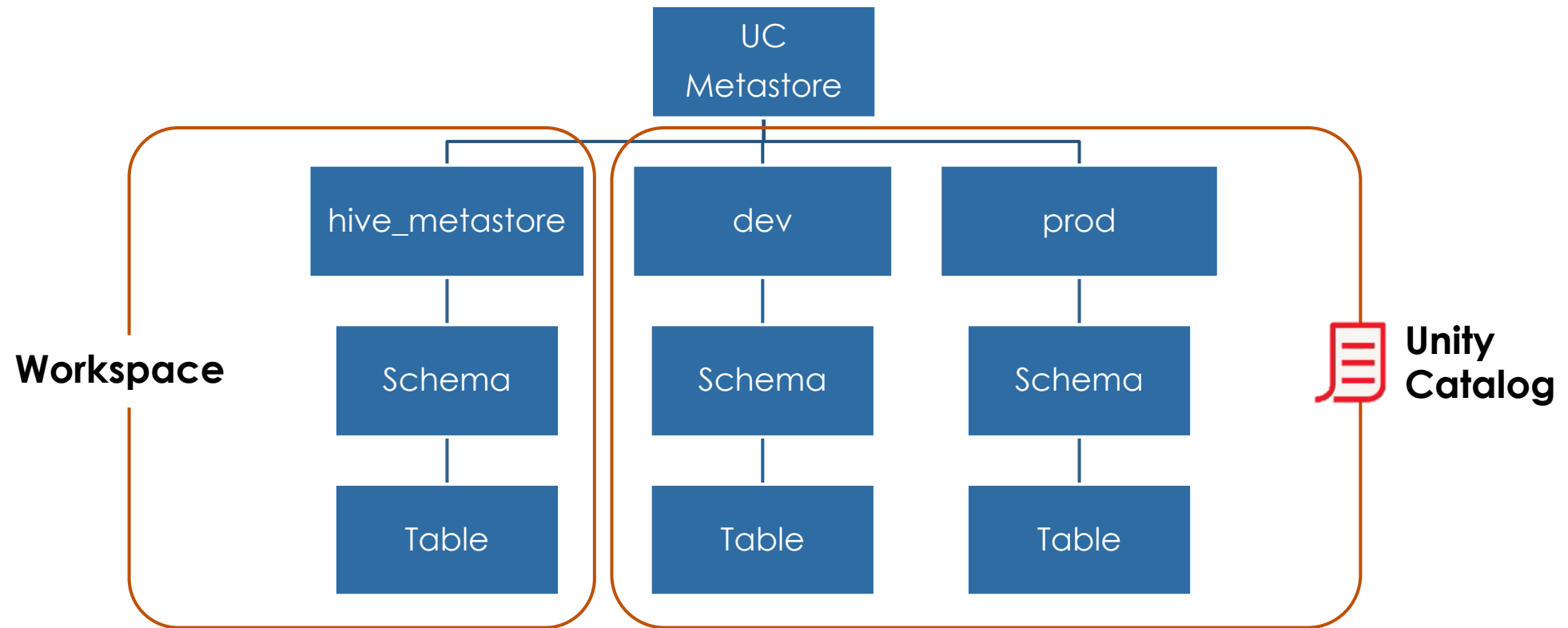
Principles

User

Service Principles

group

Accessing legacy Hive metastore



Features

- ▶ Centralized governance for data and AI
- ▶ Built-in data search and discovery
- ▶ Automated lineage
- ▶ No hard migration required

Account Console

- ▶ Log in as account administrator
- ▶ <https://accounts.cloud.databricks.com>

Certification Overview

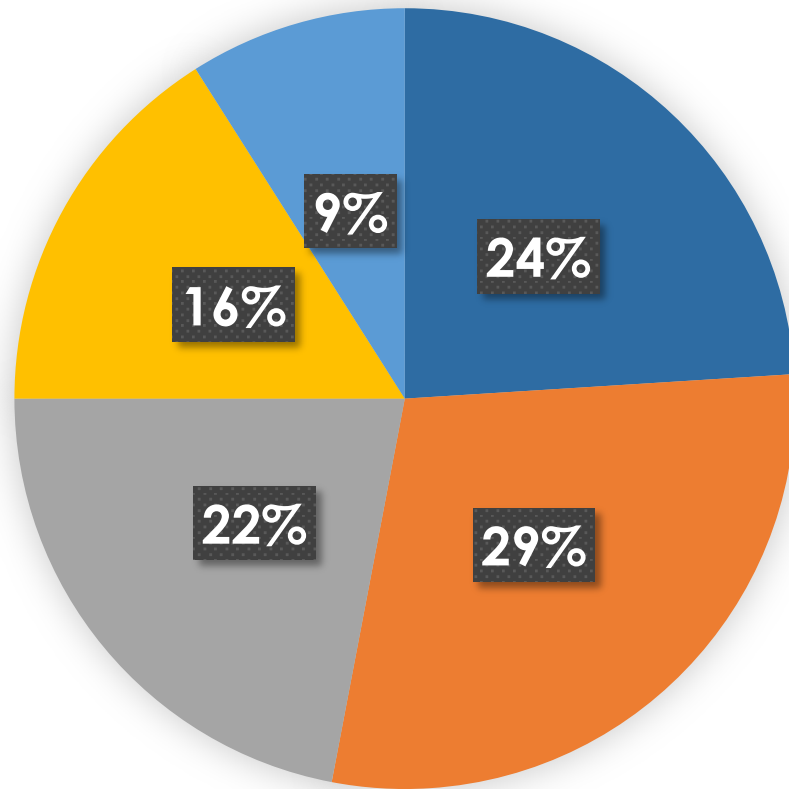
Learning Objectives

- ▶ Format and structure of the exam.
- ▶ The topics covered in the exam.
- ▶ Types of questions provided on the exam

Exam Details

- ▶ Time allotted to complete exam = 1.5 hours
- ▶ Number of Questions = 45
- ▶ Passing scores = At least 70% (32/45)
- ▶ Exam fee = \$200
- ▶ Exam retake fee = \$200

Exam Questions



- Databricks Lakehouse Platform (11/45)
- ELT with Spark SQL and Python (13/45)
- Incremental Data Processing (10/45)
- Production Pipelines (7/45)
- Data Governance (4/45)

Out-of-scope

- ▶ Apache Spark internals
- ▶ Databricks CLI and REST API
- ▶ Change Data Capture CDC/CDF
- ▶ Data modeling concepts
- ▶ GDPR/CCPA
- ▶ Monitoring and logging production jobs
- ▶ Dependency management
- ▶ Testing


Code Examples

- ▶ Code will be in mainly in SQL
- ▶ Otherwise, Python

Exam Platform

<https://www.webassessor.com/databricks>

[Home](#) | [Login](#) | [Forgot Password](#) | [Create New Account](#)

 **databricks**

[Company](#) [Contact](#) [Register for an Exam](#)

Databricks Certification

Welcome to Databricks online assessment registration.

Follow these steps to schedule an assessment:

1. Create a New Account by clicking [here](#).
2. Log into your newly created account.
3. Click on 'Register for an Assessment' and follow the remaining steps to complete scheduling.

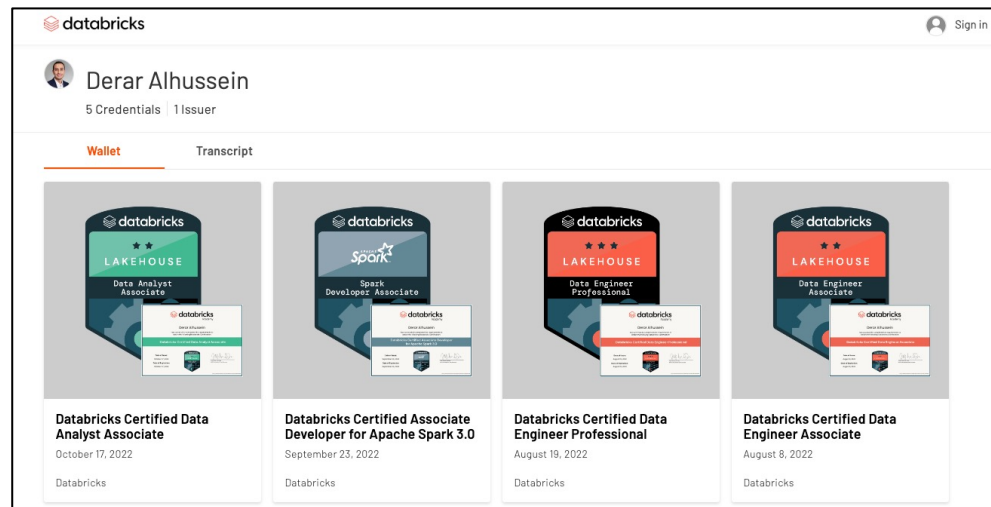
Exam Proctoring

- ▶ Monitored via webcam by a Webassessor proctor.
- ▶ Asked to provide valid, photo-based identification.
- ▶ Proctor does not provide assistance on the content of the exam
- ▶ No test aids will be available during the exam

Exam Result

- ▶ Certification exams are automatically graded.
- ▶ 24 hours to receive the badge and Certificate

<https://credentials.databricks.com>



Questions Types

- ▶ Multiple-choice questions – Only one correct answer
- ▶ Questions Types:
 - ▶ Conceptual Questions
 - ▶ Code-Based Questions

Conceptual Questions

- ▶ Which part of the Databricks Lakehouse Platform can data engineers use to orchestrate jobs ?
 - A. Repos
 - B. Workflows
 - C. Data Explorer
 - D. Databricks SQL
 - E. Cluster

Conceptual Questions

- ▶ Which part of the Databricks Lakehouse Platform can data engineers use to orchestrate jobs ?
 - A. Repos
 - B. Workflows**
 - C. Data Explorer
 - D. Databricks SQL
 - E. Cluster

Code-Based Questions

```
spark.table("sales")  
  .writeStream  
  .option("checkpointLocation", checkpointPath)  
  . _____  
  .table("new_sales")
```

If you want the query to execute a single micro-batch to process all of the available data, which of the following lines of code should you use to fill in the blank ?

- A. `trigger(once=True)`
- B. `trigger(continuous="once")`
- C. `processingTime("once")`
- D. `trigger(processingTime="once")`
- E. `processingTime(1)`

Code-Based Questions

```
spark.table("sales")  
  .writeStream  
  .option("checkpointLocation", checkpointPath)  
  .  
  .table("new_sales")
```

If you want the query to execute a single micro-batch to process all of the available data, which of the following lines of code should you use to fill in the blank ?

- A. **trigger(once=True)**
- B. trigger(continuous="once")
- C. processingTime("once")
- D. trigger(processingTime="once")
- E. processingTime(1)

Practice Exam

- ▶ Databricks official Practice test
- ▶ PDF file



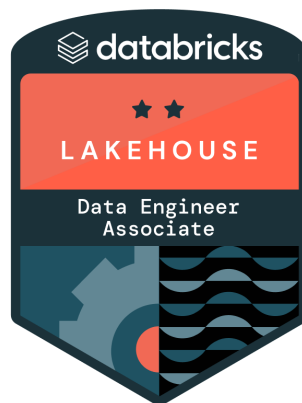
Derar Alhussein

www.linkedin.com/in/DerarAlhussein



Practice Exam

Databricks Certified Data Engineer Associate



Overview

This is a practice exam for the [Databricks Certified Data Engineer Associate](#) exam. The questions here are retired questions from the actual exam that are representative of the questions one will receive while taking the actual exam. After taking this practice exam, one should know what to expect while taking the actual Data Engineer Associate exam.

Just like the actual exam, it contains 45 multiple-choice questions. Each of these questions has one correct answer. The correct answer for each question is listed at the bottom in the **Correct Answers** section.

There are a few more things to be aware of:

1. There is a 90-minute time limit to take the actual exam.
2. In order to pass the actual exam, testers will need to correctly answer at least 32 of the 45 questions.
3. Testers will not have access to any documentation or Databricks environments during the exam.
4. These questions are representative of questions that are on the actual exam, but they are no longer on the actual exam.

If you have more questions, please review the [Databricks Academy Certification FAQ](#).

Once you've completed the practice exam, evaluate your score using the correct answers at the bottom of this document. If you're ready to take the exam, head to the [Databricks Certification portal](#) to register.

Questions

Question 1

Which of the following describes a benefit of a data lakehouse that is unavailable in a traditional data warehouse?

- A. A data lakehouse provides a relational system of data management.
- B. A data lakehouse captures snapshots of data for version control purposes.
- C. A data lakehouse couples storage and compute for complete control.
- D. A data lakehouse utilizes proprietary storage formats for data.
- E. A data lakehouse enables both batch and streaming analytics.

Question 2

Which of the following locations hosts the driver and worker nodes of a Databricks-managed cluster?

- A. Data plane
- B. Control plane
- C. Databricks Filesystem
- D. JDBC data source
- E. Databricks web application

Question 3

A data architect is designing a data model that works for both video-based machine learning workloads and highly audited batch ETL/ELT workloads.

Which of the following describes how using a data lakehouse can help the data architect meet the needs of both workloads?

- A. A data lakehouse requires very little data modeling.
- B. A data lakehouse combines compute and storage for simple governance.
- C. A data lakehouse provides autoscaling for compute clusters.
- D. A data lakehouse stores unstructured data and is ACID-compliant.
- E. A data lakehouse fully exists in the cloud.

Question 4

Which of the following describes a scenario in which a data engineer will want to use a Job cluster instead of an all-purpose cluster?

- A. An ad-hoc analytics report needs to be developed while minimizing compute costs.
- B. A data team needs to collaborate on the development of a machine learning model.
- C. An automated workflow needs to be run every 30 minutes.
- D. A Databricks SQL query needs to be scheduled for upward reporting.
- E. A data engineer needs to manually investigate a production error.

Question 5

A data engineer has created a Delta table as part of a data pipeline. Downstream data analysts now need SELECT permission on the Delta table.

Assuming the data engineer is the Delta table owner, which part of the Databricks Lakehouse Platform can the data engineer use to grant the data analysts the appropriate access?

- A. Repos
- B. Jobs
- C. Data Explorer
- D. Databricks Filesystem
- E. Dashboards

Question 6

Two junior data engineers are authoring separate parts of a single data pipeline notebook. They are working on separate Git branches so they can pair program on the same notebook simultaneously. A senior data engineer experienced in Databricks suggests there is a better alternative for this type of collaboration.

Which of the following supports the senior data engineer's claim?

- A. Databricks Notebooks support automatic change-tracking and versioning
- B. Databricks Notebooks support real-time coauthoring on a single notebook
- C. Databricks Notebooks support commenting and notification comments
- D. Databricks Notebooks support the use of multiple languages in the same notebook
- E. Databricks Notebooks support the creation of interactive data visualizations

Question 7

Which of the following describes how Databricks Repos can help facilitate CI/CD workflows on the Databricks Lakehouse Platform?

- A. Databricks Repos can facilitate the pull request, review, and approval process before merging branches
- B. Databricks Repos can merge changes from a secondary Git branch into a main Git branch
- C. Databricks Repos can be used to design, develop, and trigger Git automation pipelines
- D. Databricks Repos can store the single-source-of-truth Git repository
- E. Databricks Repos can commit or push code changes to trigger a CI/CD process

Question 8

Which of the following statements describes Delta Lake?

- A. Delta Lake is an open source analytics engine used for big data workloads.
- B. Delta Lake is an open format storage layer that delivers reliability, security, and performance.
- C. Delta Lake is an open source platform to help manage the complete machine learning lifecycle.
- D. Delta Lake is an open source data storage format for distributed data.
- E. Delta Lake is an open format storage layer that processes data.

Question 9

A data architect has determined that a table of the following format is necessary:

id	birthDate	avgRating
a1	1990-01-06	5.5
a2	1974-11-21	7.1
...

Which of the following code blocks uses SQL DDL commands to create an empty Delta table in the above format regardless of whether a table already exists with this name?

- A.

```
CREATE OR REPLACE TABLE table_name AS
SELECT
    id STRING,
    birthDate DATE,
    avgRating FLOAT
USING DELTA
```
- B.

```
CREATE OR REPLACE TABLE table_name (
    id STRING,
    birthDate DATE,
    avgRating FLOAT
)
```
- C.

```
CREATE TABLE IF NOT EXISTS table_name (
    id STRING,
    birthDate DATE,
    avgRating FLOAT
)
```
- D.

```
CREATE TABLE table_name AS
SELECT
    id STRING,
    birthDate DATE,
    avgRating FLOAT
```
- E.

```
CREATE OR REPLACE TABLE table_name WITH COLUMNS (
    id STRING,
    birthDate DATE,
    avgRating FLOAT
) USING DELTA
```

Question 10

Which of the following SQL keywords can be used to append new rows to an existing Delta table?

- A. UPDATE
- B. COPY
- C. INSERT INTO
- D. DELETE
- E. UNION

Question 11

A data engineering team needs to query a Delta table to extract rows that all meet the same condition. However, the team has noticed that the query is running slowly. The team has already tuned the size of the data files. Upon investigating, the team has concluded that the rows meeting the condition are sparsely located throughout each of the data files.

Based on the scenario, which of the following optimization techniques could speed up the query?

- A. Data skipping
- B. Z-Ordering
- C. Bin-packing
- D. Write as a Parquet file
- E. Tuning the file size

Question 12

A data engineer needs to create a database called **customer360** at the location **/customer/customer360**. The data engineer is unsure if one of their colleagues has already created the database.

Which of the following commands should the data engineer run to complete this task?

- A. `CREATE DATABASE customer360 LOCATION '/customer/customer360';`
- B. `CREATE DATABASE IF NOT EXISTS customer360;`
- C. `CREATE DATABASE IF NOT EXISTS customer360 LOCATION '/customer/customer360';`
- D. `CREATE DATABASE IF NOT EXISTS customer360 DELTA LOCATION '/customer/customer360';`
- E. `CREATE DATABASE customer360 DELTA LOCATION '/customer/customer360';`

Question 13

A junior data engineer needs to create a Spark SQL table **my_table** for which Spark manages both the data and the metadata. The metadata and data should also be stored in the Databricks Filesystem (DBFS).

Which of the following commands should a senior data engineer share with the junior data engineer to complete this task?

- A. `CREATE TABLE my_table (id STRING, value STRING) USING org.apache.spark.sql.parquet OPTIONS (PATH "storage-path");`
- B. `CREATE MANAGED TABLE my_table (id STRING, value STRING) USING org.apache.spark.sql.parquet OPTIONS (PATH "storage-path");`
- C. `CREATE MANAGED TABLE my_table (id STRING, value STRING);`
- D. `CREATE TABLE my_table (id STRING, value STRING) USING DBFS;`
- E. `CREATE TABLE my_table (id STRING, value STRING);`

Question 14

A data engineer wants to create a relational object by pulling data from two tables. The relational object must be used by other data engineers in other sessions. In order to save on storage costs, the data engineer wants to avoid copying and storing physical data.

Which of the following relational objects should the data engineer create?

- A. View
- B. Temporary view
- C. Delta Table
- D. Database
- E. Spark SQL Table

Question 15

A data engineering team has created a series of tables using Parquet data stored in an external system. The team is noticing that after appending new rows to the data in the external system, their queries within Databricks are not returning the new rows. They identify the caching of the previous data as the cause of this issue.

Which of the following approaches will ensure that the data returned by queries is always up-to-date?

- A. The tables should be converted to the Delta format
- B. The tables should be stored in a cloud-based external system
- C. The tables should be refreshed in the writing cluster before the next query is run
- D. The tables should be altered to include metadata to not cache
- E. The tables should be updated before the next query is run

Question 16

A table **customerLocations** exists with the following schema:

```
id STRING,  
date STRING,  
city STRING,  
country STRING
```

A senior data engineer wants to create a new table from this table using the following command:

```
CREATE TABLE customersPerCountry AS  
SELECT country,  
       COUNT(*) AS customers  
FROM customerLocations  
GROUP BY country;
```

A junior data engineer asks why the schema is not being declared for the new table.

Which of the following responses explains why declaring the schema is not necessary?

- A. CREATE TABLE AS SELECT statements adopt schema details from the source table and query.
- B. CREATE TABLE AS SELECT statements infer the schema by scanning the data.
- C. CREATE TABLE AS SELECT statements result in tables where schemas are optional.
- D. CREATE TABLE AS SELECT statements assign all columns the type STRING.
- E. CREATE TABLE AS SELECT statements result in tables that do not support schemas.

Question 17

A data engineer is overwriting data in a table by deleting the table and recreating the table. Another data engineer suggests that this is inefficient and the table should simply be overwritten instead.

Which of the following reasons to overwrite the table instead of deleting and recreating the table is incorrect?

- A. Overwriting a table is efficient because no files need to be deleted.
- B. Overwriting a table results in a clean table history for logging and audit purposes.
- C. Overwriting a table maintains the old version of the table for Time Travel.
- D. Overwriting a table is an atomic operation and will not leave the table in an unfinished state.
- E. Overwriting a table allows for concurrent queries to be completed while in progress.

Question 18

Which of the following commands will return records from an existing Delta table `my_table` where duplicates have been removed?

- A. `DROP DUPLICATES FROM my_table;`
- B. `SELECT * FROM my_table WHERE duplicate = False;`
- C. `SELECT DISTINCT * FROM my_table;`
- D. `MERGE INTO my_table a USING new_records b ON a.id = b.id WHEN NOT MATCHED THEN INSERT *;`
- E. `MERGE INTO my_table a USING new_records b;`

Question 19

A data engineer wants to horizontally combine two tables as a part of a query. They want to use a shared column as a key column, and they only want the query result to contain rows whose value in the key column is present in both tables.

Which of the following SQL commands can they use to accomplish this task?

- A. `INNER JOIN`
- B. `OUTER JOIN`
- C. `LEFT JOIN`
- D. `MERGE`
- E. `UNION`

Question 20

A junior data engineer has ingested a JSON file into a table **raw_table** with the following schema:

```
cart_id STRING,  
items ARRAY<item_id:STRING>
```

The junior data engineer would like to unnest the **items** column in **raw_table** to result in a new table with the following schema:

```
cart_id STRING,  
item_id STRING
```

Which of the following commands should the junior data engineer run to complete this task?

- A. `SELECT cart_id, filter(items) AS item_id FROM raw_table;`
- B. `SELECT cart_id, flatten(items) AS item_id FROM raw_table;`
- C. `SELECT cart_id, reduce(items) AS item_id FROM raw_table;`
- D. `SELECT cart_id, explode(items) AS item_id FROM raw_table;`
- E. `SELECT cart_id, slice(items) AS item_id FROM raw_table;`

Question 21

A data engineer has ingested a JSON file into a table **raw_table** with the following schema:

```
transaction_id STRING,  
payload ARRAY<customer_id:STRING, date:TIMESTAMP, store_id:STRING>
```

The data engineer wants to efficiently extract the date of each transaction into a table with the following schema:

```
transaction_id STRING,  
date TIMESTAMP
```

Which of the following commands should the data engineer run to complete this task?

- A. `SELECT transaction_id, explode(payload) FROM raw_table;`
- B. `SELECT transaction_id, payload.date FROM raw_table;`
- C. `SELECT transaction_id, date FROM raw_table;`

- D. `SELECT transaction_id, payload[date] FROM raw_table;`
- E. `SELECT transaction_id, date from payload FROM raw_table;`

Question 22

A data analyst has provided a data engineering team with the following Spark SQL query:

```
SELECT district,
       avg(sales)
FROM store_sales_20220101
GROUP BY district;
```

The data analyst would like the data engineering team to run this query every day. The date at the end of the table name (20220101) should automatically be replaced with the current date each time the query is run.

Which of the following approaches could be used by the data engineering team to efficiently automate this process?

- A. They could wrap the query using PySpark and use Python's string variable system to automatically update the table name.
- B. They could manually replace the date within the table name with the current day's date.
- C. They could request that the data analyst rewrites the query to be run less frequently.
- D. They could replace the string-formatted date in the table with a timestamp-formatted date.
- E. They could pass the table into PySpark and develop a robustly tested module on the existing query.

Question 23

A data engineer has ingested data from an external source into a PySpark DataFrame `raw_df`. They need to briefly make this data available in SQL for a data analyst to perform a quality assurance check on the data.

Which of the following commands should the data engineer run to make this data available in SQL for only the remainder of the Spark session?

- A. `raw_df.createOrReplaceTempView("raw_df")`
- B. `raw_df.createTable("raw_df")`
- C. `raw_df.write.save("raw_df")`
- D. `raw_df.saveAsTable("raw_df")`

- E. There is no way to share data between PySpark and SQL.

Question 24

A data engineer needs to dynamically create a table name string using three Python variables: **region**, **store**, and **year**. An example of a table name is below when **region** = "nyc", **store** = "100", and **year** = "2021":

nyc100_sales_2021

Which of the following commands should the data engineer use to construct the table name in Python?

- A. "{region}+{store}+_sales_{year}"
- B. f"{region}+{store}+_sales_{year}"
- C. "{region}{store}_sales_{year}"
- D. f"{region}{store}_sales_{year}"
- E. {region}+{store}+"_sales_" + {year}

Question 25

A data engineer has developed a code block to perform a streaming read on a data source. The code block is below:

```
(spark
    .read
    .schema(schema)
    .format("cloudFiles")
    .option("cloudFiles.format", "json")
    .load(dataSource)
)
```

The code block is returning an error.

Which of the following changes should be made to the code block to configure the block to successfully perform a streaming read?

- A. The **.read** line should be replaced with **.readStream**.
- B. A new **.stream** line should be added after the **.read** line.
- C. The **.format("cloudFiles")** line should be replaced with **.format("stream")**.
- D. A new **.stream** line should be added after the **spark** line.
- E. A new **.stream** line should be added after the **.load(dataSource)** line.

Question 26

A data engineer has configured a Structured Streaming job to read from a table, manipulate the data, and then perform a streaming write into a new table.

The code block used by the data engineer is below:

```
(spark.table("sales")
  .withColumn("avg_price", col("sales") / col("units"))
  .writeStream
  .option("checkpointLocation", checkpointPath)
  .outputMode("complete")
  ._____
  .table("new_sales")
)
```

If the data engineer only wants the query to execute a single micro-batch to process all of the available data, which of the following lines of code should the data engineer use to fill in the blank?

- A. `trigger(once=True)`
- B. `trigger(continuous="once")`
- C. `processingTime("once")`
- D. `trigger(processingTime="once")`
- E. `processingTime(1)`

Question 27

A data engineer is designing a data pipeline. The source system generates files in a shared directory that is also used by other processes. As a result, the files should be kept as is and will accumulate in the directory. The data engineer needs to identify which files are new since the previous run in the pipeline, and set up the pipeline to only ingest those new files with each run.

Which of the following tools can the data engineer use to solve this problem?

- A. Databricks SQL
- B. Delta Lake
- C. Unity Catalog
- D. Data Explorer
- E. Auto Loader

Question 28

A data engineering team is in the process of converting their existing data pipeline to utilize Auto Loader for incremental processing in the ingestion of JSON files. One data engineer comes across the following code block in the Auto Loader documentation:

```
(streaming_df = spark.readStream.format("cloudFiles")
    .option("cloudFiles.format", "json")
    .option("cloudFiles.schemaLocation", schemaLocation)
    .load(sourcePath))
```

Assuming that **schemaLocation** and **sourcePath** have been set correctly, which of the following changes does the data engineer need to make to convert this code block to use Auto Loader to ingest the data?

- A. The data engineer needs to change the **format("cloudFiles")** line to **format("autoLoader")**.
- B. There is no change required. Databricks automatically uses Auto Loader for streaming reads.
- C. There is no change required. The inclusion of **format("cloudFiles")** enables the use of Auto Loader.
- D. The data engineer needs to add the **.autoLoader** line before the **.load(sourcePath)** line.
- E. There is no change required. The data engineer needs to ask their administrator to turn on Auto Loader.

Question 29

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

- A. A job that aggregates cleaned data to create standard summary statistics
- B. A job that queries aggregated data to publish key insights into a dashboard
- C. A job that ingests raw data from a streaming source into the Lakehouse
- D. A job that develops a feature set for a machine learning application
- E. A job that enriches data by parsing its timestamps into a human-readable format

Question 30

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

- A. A job that enriches data by parsing its timestamps into a human-readable format
- B. A job that queries aggregated data that already feeds into a dashboard
- C. A job that ingests raw data from a streaming source into the Lakehouse
- D. A job that aggregates cleaned data to create standard summary statistics
- E. A job that cleans data by removing malformed records

Question 31

Which of the following Structured Streaming queries is performing a hop from a Bronze table to a Silver table?

- A.

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

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

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

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

```
(spark.read.load(rawSalesLocation)
  .writeStream
```

```
        .option("checkpointLocation", checkpointPath)
        .outputMode("append")
        .table("uncleanedSales")
    )
```

Question 32

Which of the following benefits does Delta Live Tables provide for ELT pipelines over standard data pipelines that utilize Spark and Delta Lake on Databricks?

- A. The ability to declare and maintain data table dependencies
- B. The ability to write pipelines in Python and/or SQL
- C. The ability to access previous versions of data tables
- D. The ability to automatically scale compute resources
- E. The ability to perform batch and streaming queries

Question 33

A data engineer has three notebooks in an ELT pipeline. The notebooks need to be executed in a specific order for the pipeline to complete successfully. The data engineer would like to use Delta Live Tables to manage this process.

Which of the following steps must the data engineer take as part of implementing this pipeline using Delta Live Tables?

- A. They need to create a Delta Live Tables pipeline from the Data page.
- B. They need to create a Delta Live Tables pipeline from the Jobs page.
- C. They need to create a Delta Live tables pipeline from the Compute page.
- D. They need to refactor their notebook to use Python and the dlt library.
- E. They need to refactor their notebook to use SQL and **CREATE LIVE TABLE** keyword.

Question 34

A data engineer has written the following query:

```
SELECT *
FROM json.`/path/to/json/file.json`;
```

The data engineer asks a colleague for help to convert this query for use in a Delta Live Tables (DLT) pipeline. The query should create the first table in the DLT pipeline.

Which of the following describes the change the colleague needs to make to the query?

- A. They need to add a **COMMENT** line at the beginning of the query.
- B. They need to add a **CREATE LIVE TABLE table_name AS** line at the beginning of the query.
- C. They need to add a **live.** prefix prior to **json.** in the **FROM** line.
- D. They need to add a **CREATE DELTA LIVE TABLE table_name AS** line at the beginning of the query.
- E. They need to add the **cloud_files(...)** wrapper to the JSON file path.

Question 35

A dataset has been defined using Delta Live Tables and includes an expectations clause:

```
CONSTRAINT valid_timestamp EXPECT (timestamp > '2020-01-01')
```

What is the expected behavior when a batch of data containing data that violates these constraints is processed?

- A. Records that violate the expectation are added to the target dataset and recorded as invalid in the event log.
- B. Records that violate the expectation are dropped from the target dataset and recorded as invalid in the event log.
- C. Records that violate the expectation cause the job to fail.
- D. Records that violate the expectation are added to the target dataset and flagged as invalid in a field added to the target dataset.
- E. Records that violate the expectation are dropped from the target dataset and loaded into a quarantine table.

Question 36

A Delta Live Table pipeline includes two datasets defined using **STREAMING LIVE TABLE**. Three datasets are defined against Delta Lake table sources using **LIVE TABLE**.

The table is configured to run in Development mode using the Triggered Pipeline Mode.

Assuming previously unprocessed data exists and all definitions are valid, what is the expected outcome after clicking Start to update the pipeline?

- A. All datasets will be updated once and the pipeline will shut down. The compute resources will be terminated.
- B. All datasets will be updated at set intervals until the pipeline is shut down. The compute resources will be deployed for the update and terminated when the pipeline is stopped.
- C. All datasets will be updated at set intervals until the pipeline is shut down. The compute resources will persist after the pipeline is stopped to allow for additional testing.
- D. All datasets will be updated once and the pipeline will shut down. The compute resources will persist to allow for additional testing.
- E. All datasets will be updated continuously and the pipeline will not shut down. The compute resources will persist with the pipeline.

Question 37

A data engineer has a Job with multiple tasks that runs nightly. One of the tasks unexpectedly fails during 10 percent of the runs.

Which of the following actions can the data engineer perform to ensure the Job completes each night while minimizing compute costs?

- A. They can institute a retry policy for the entire Job
- B. They can observe the task as it runs to try and determine why it is failing
- C. They can set up the Job to run multiple times ensuring that at least one will complete
- D. They can institute a retry policy for the task that periodically fails
- E. They can utilize a Jobs cluster for each of the tasks in the Job

Question 38

A data engineer has set up two Jobs that each run nightly. The first Job starts at 12:00 AM, and it usually completes in about 20 minutes. The second Job depends on the first Job, and it starts at 12:30 AM. Sometimes, the second Job fails when the first Job does not complete by 12:30 AM.

Which of the following approaches can the data engineer use to avoid this problem?

- A. They can utilize multiple tasks in a single job with a linear dependency
- B. They can use cluster pools to help the Jobs run more efficiently
- C. They can set up a retry policy on the first Job to help it run more quickly
- D. They can limit the size of the output in the second Job so that it will not fail as easily
- E. They can set up the data to stream from the first Job to the second Job

Question 39

A data engineer has set up a notebook to automatically process using a Job. The data engineer's manager wants to version control the schedule due to its complexity.

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

- A. They can link the Job to notebooks that are a part of a Databricks Repo.
- B. They can submit the Job once on a Job cluster.
- C. They can download the JSON description of the Job from the Job's page.
- D. They can submit the Job once on an all-purpose cluster.
- E. They can download the XML description of the Job from the Job's page.

Question 40

A data analyst has noticed that their Databricks SQL queries are running too slowly. They claim that this issue is affecting all of their sequentially run queries. They ask the data engineering team for help. The data engineering team notices that each of the queries uses the same SQL endpoint, but the SQL endpoint is not used by any other user.

Which of the following approaches can the data engineering team use to improve the latency of the data analyst's queries?

- A. They can turn on the Serverless feature for the SQL endpoint.
- B. They can increase the maximum bound of the SQL endpoint's scaling range.
- C. They can increase the cluster size of the SQL endpoint.
- D. They can turn on the Auto Stop feature for the SQL endpoint.
- E. They can turn on the Serverless feature for the SQL endpoint and change the Spot Instance Policy to "Reliability Optimized."

Question 41

An engineering manager uses a Databricks SQL query to monitor their team's progress on fixes related to customer-reported bugs. The manager checks the results of the query every day, but they are manually rerunning the query each day and waiting for the results.

Which of the following approaches can the manager use to ensure the results of the query are updated each day?

- A. They can schedule the query to run every 1 day from the Jobs UI.
- B. They can schedule the query to refresh every 1 day from the query's page in Databricks SQL.
- C. They can schedule the query to run every 12 hours from the Jobs UI.
- D. They can schedule the query to refresh every 1 day from the SQL endpoint's page in Databricks SQL.
- E. They can schedule the query to refresh every 12 hours from the SQL endpoint's page in Databricks SQL.

Question 42

A data engineering team has been 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?

- A. They can set up an Alert for the accompanying dashboard to notify them if the returned value is greater than 60.
- B. They can set up an Alert for the query to notify when the ELT job fails.
- C. They can set up an Alert for the accompanying dashboard to notify when it has not refreshed in 60 minutes.
- D. They can set up an Alert for the query to notify them if the returned value is greater than 60.
- E. This type of alerting is not possible in Databricks.

Question 43

A data engineering manager has noticed that each of the queries in a Databricks SQL dashboard takes a few minutes to update when they manually click the "Refresh" button. They are curious why this might be occurring, so a team member provides a variety of reasons on why the delay might be occurring.

Which of the following reasons *fails* to explain why the dashboard might be taking a few minutes to update?

- A. The SQL endpoint being used by each of the queries might need a few minutes to start up.
- B. The queries attached to the dashboard might take a few minutes to run under normal circumstances.

- C. The queries attached to the dashboard might first be checking to determine if new data is available.
- D. The Job associated with updating the dashboard might be using a non-pooled endpoint.
- E. The queries attached to the dashboard might all be connected to their own, unstarted Databricks clusters.

Question 44

A new data engineer has started at a company. The data engineer has recently been added to the company's Databricks workspace as **new.engineer@company.com**. The data engineer needs to be able to query the table **sales** in the database **retail**. The new data engineer already has been granted **USAGE** on the database **retail**.

Which of the following commands can be used to grant the appropriate permissions to the new data engineer?

- A. `GRANT USAGE ON TABLE sales TO new.engineer@company.com;`
- B. `GRANT CREATE ON TABLE sales TO new.engineer@company.com;`
- C. `GRANT SELECT ON TABLE sales TO new.engineer@company.com;`
- D. `GRANT USAGE ON TABLE new.engineer@company.com TO sales;`
- E. `GRANT SELECT ON TABLE new.engineer@company.com TO sales;`

Question 45

A new data engineer **new.engineer@company.com** has been assigned to an ELT project. The new data engineer will need full privileges on the table **sales** to fully manage the project.

Which of the following commands can be used to grant full permissions on the table to the new data engineer?

- A. `GRANT ALL PRIVILEGES ON TABLE sales TO new.engineer@company.com;`
- B. `GRANT USAGE ON TABLE sales TO new.engineer@company.com;`
- C. `GRANT ALL PRIVILEGES ON TABLE new.engineer@company.com TO sales;`
- D. `GRANT SELECT ON TABLE sales TO new.engineer@company.com;`
- E. `GRANT SELECT CREATE MODIFY ON TABLE sales TO new.engineer@company.com;`

Correct Answers

1. E
2. A
3. D
4. C
5. C
6. B
7. E
8. B
9. B
10. C
11. B
12. C
13. E
14. A
15. A
16. A
17. B
18. C
19. A
20. D
21. B
22. A
23. A
24. D
25. A
26. A
27. E
28. C
29. E
30. D
31. C
32. A
33. B
34. B
35. A
36. D
37. D
38. A
39. C
40. C

- 41. B
- 42. D
- 43. D
- 44. C
- 45. A