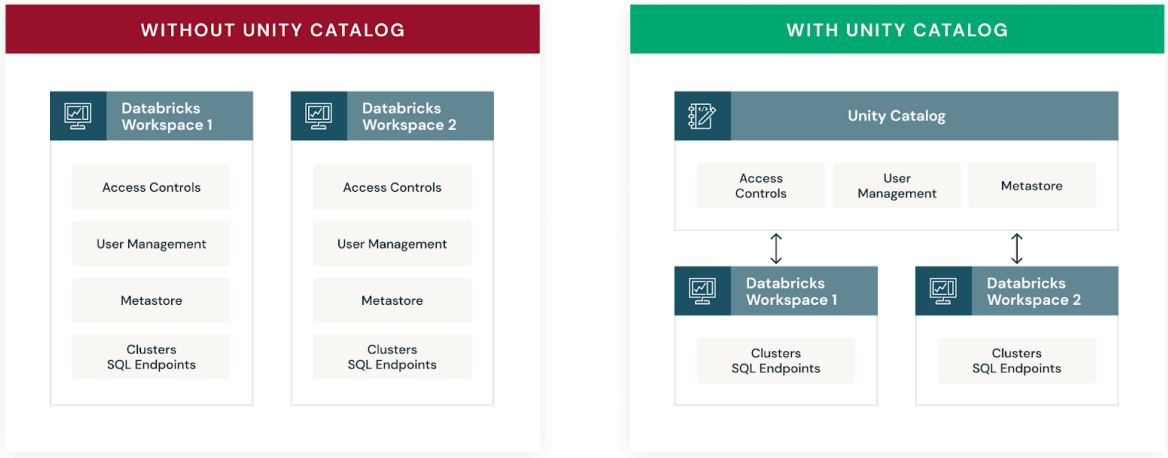
Coding Challenge

Unity Catalog

1. Introduction

Data plays a vital role in today's world, but without proper management, it can become messy and insecure. Data governance ensures that data is accurate, secure, and accessible only to the right users.

To address these needs, Databricks introduced Unity Catalog, a unified governance layer for data and AI assets across any cloud. Unity Catalog provides centralized security, access control, lineage tracking, and data discovery for the Databricks Lakehouse Platform.



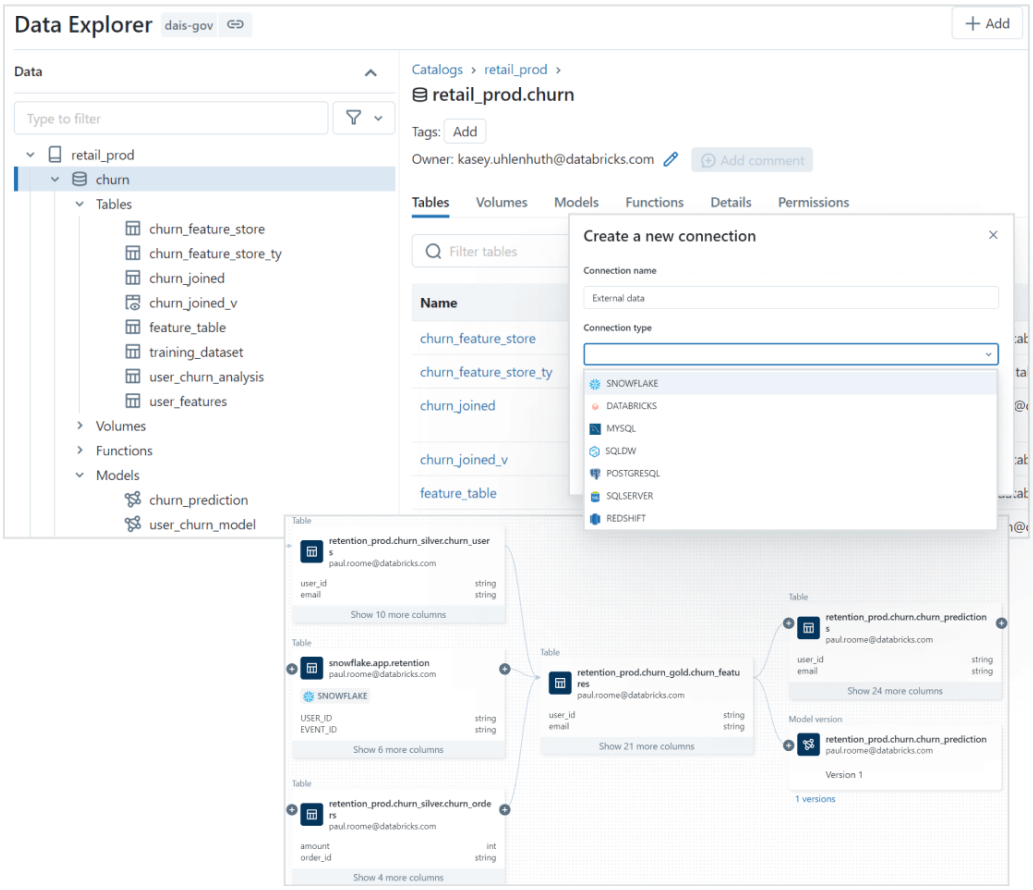
Key features and benefits of Databricks Unity Catalog:

1. Define once, secure everywhere
2. Standards-compliant security model
3. Built-in auditing and lineage
4. Data discovery
5. System tables (Public Preview)
6. Managed storage
7. External data access

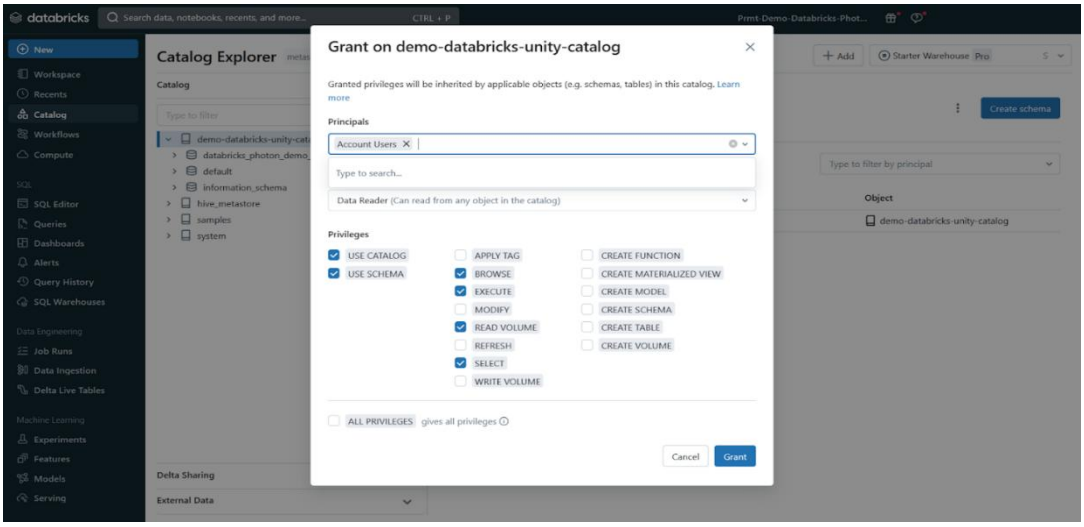
2. Databricks Unity Catalog Architecture

Unity Catalog is organized into several key layers:

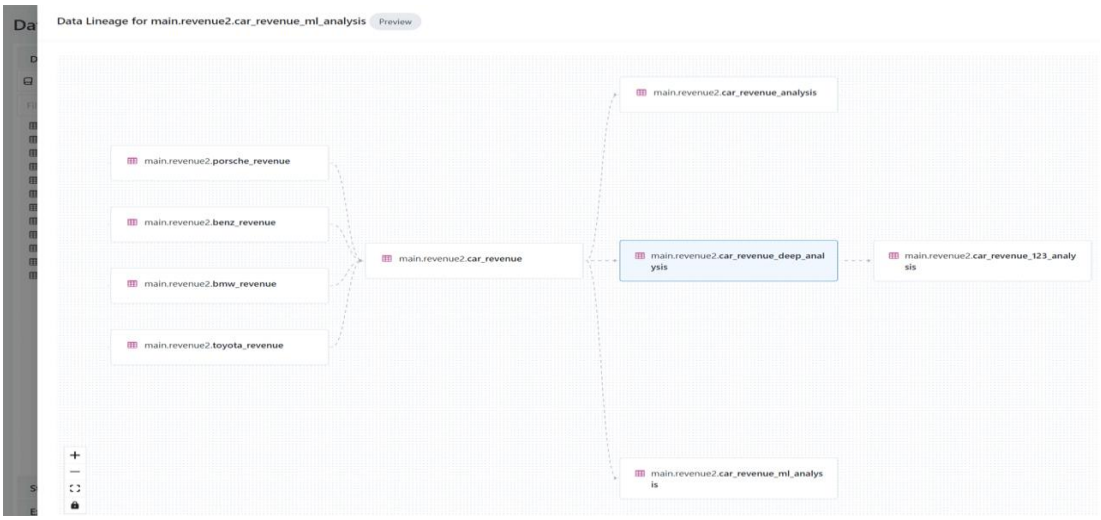
1. Unified Governance Layer – Provides central governance for all data/AI assets.



2. Access Control & Security – Role-based and attribute-based access control.

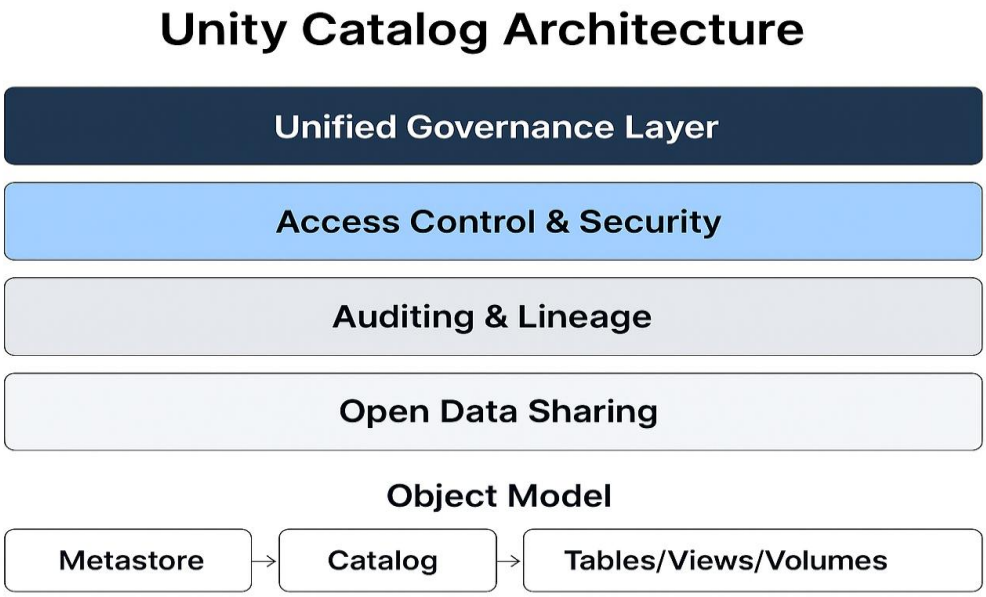


3. Auditing & Lineage – Tracks who accessed what data and how it was transformed.

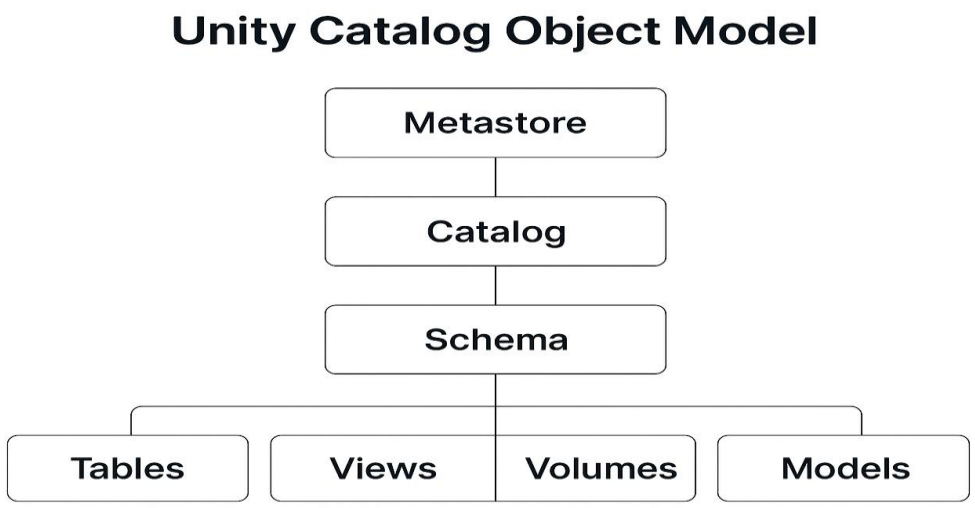


4. Open Data Sharing – Uses Delta Sharing for secure cross-cloud data exchange.

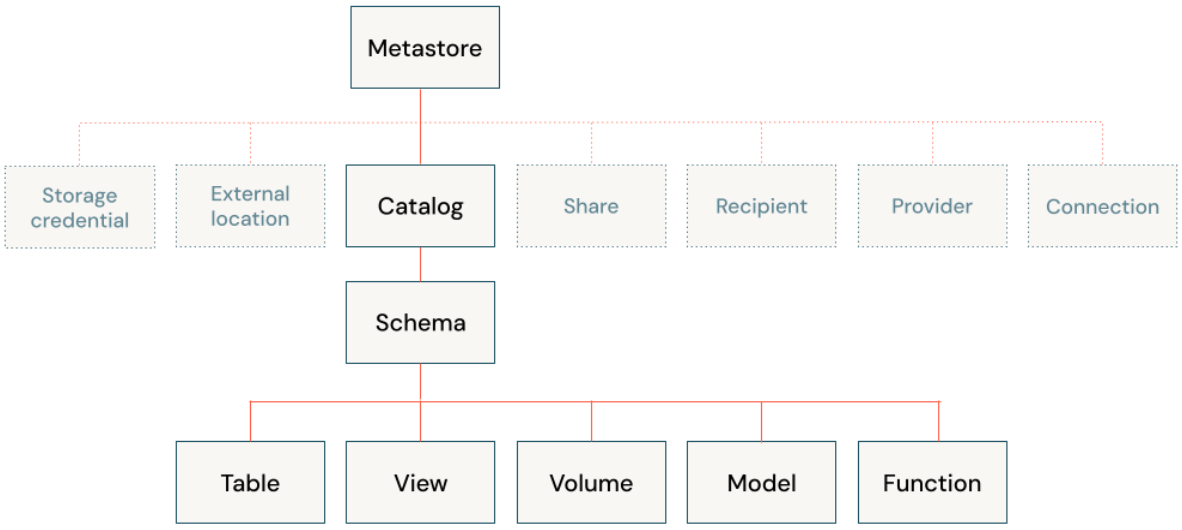
5. Object Model – Hierarchy:
    - Metastore → Catalog → Schema → Tables/Views/Volumes/Models



3. Unity Catalog Object Model



- Metastore: Top-level container.
- Catalog: Logical grouping of schemas.
- Schema: Similar to databases, contains tables/views.
- Tables: Structured datasets.
- Views: Virtual tables created via queries.
- Volumes: Containers for unstructured data (files).
- Models: ML models managed with MLflow.



4. Unity Catalog vs Hive Metastore

Databricks Unity Catalog	Hive Metastore
Databricks Unity Catalog is a centralized service for managing data governance and access control across workspaces in the Databricks	Hive Metastore is central repository for storing metadata about Hive databases, tables, partitions, and other objects in the Apache Hive data warehousing system
Databricks Unity Catalog supports a wide range of data sources, including Apache Spark tables, Delta Lake tables, AWS S3, Azure Blob Storage, HDFS, and more.	Hive Metastore is primarily designed for Hive tables and databases, but can also store metadata for external data sources like HDFS or cloud storage
Databricks Unity Catalog provides APIs and tools for managing and updating metadata, enabling automated metadata capture and synchronization with external metadata sources	Metadata management is primarily done through Hive commands or directly interacting with the underlying database
Databricks Unity Catalog offers fine-grained access control and data lineage tracking, allowing administrators to define and enforce policies for data access and modification	Access control is typically handled through Hadoop permissions or external tools like Apache Ranger
Databricks Unity Catalog is designed specifically for Databricks, offering seamless integration and collaboration within the platform	Hive Metastore is primarily designed for Hadoop-based environments, but can be used with other systems that support the Hive Metastore interface
Databricks Unity Catalog facilitates data sharing and collaboration by allowing users to grant and revoke access to data assets across different environments and teams	In Hive Metastore data sharing is typically achieved through Hadoop permissions or external tools like Apache Ranger
Databricks Unity Catalog is tightly integrated with the Databricks Unified Analytics Platform and other components of the Databricks ecosystem	Hive Metastore integrates with the Apache Hive ecosystem and can be used with other tools like Apache Spark, Apache Impala, and Apache Ranger

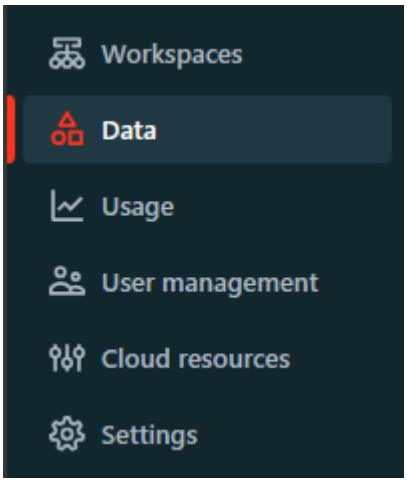
5.How to Create Unity Catalog Metastore (AWS)

Creating a Metastore (AWS Example)

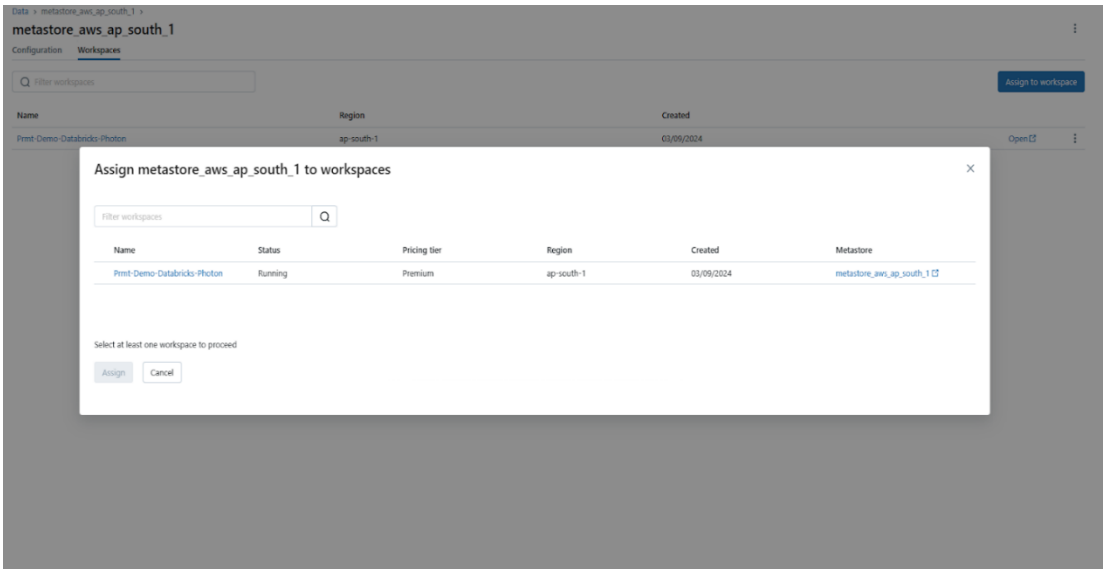
1. Configure storage (S3 bucket).
2. Create an IAM Role with access to storage.
3. Create the Metastore in Databricks and assign it to a workspace.

Enabling Unity Catalog for Workspace

1. Log in as account admin.
2. Navigate to Data → Metastore.

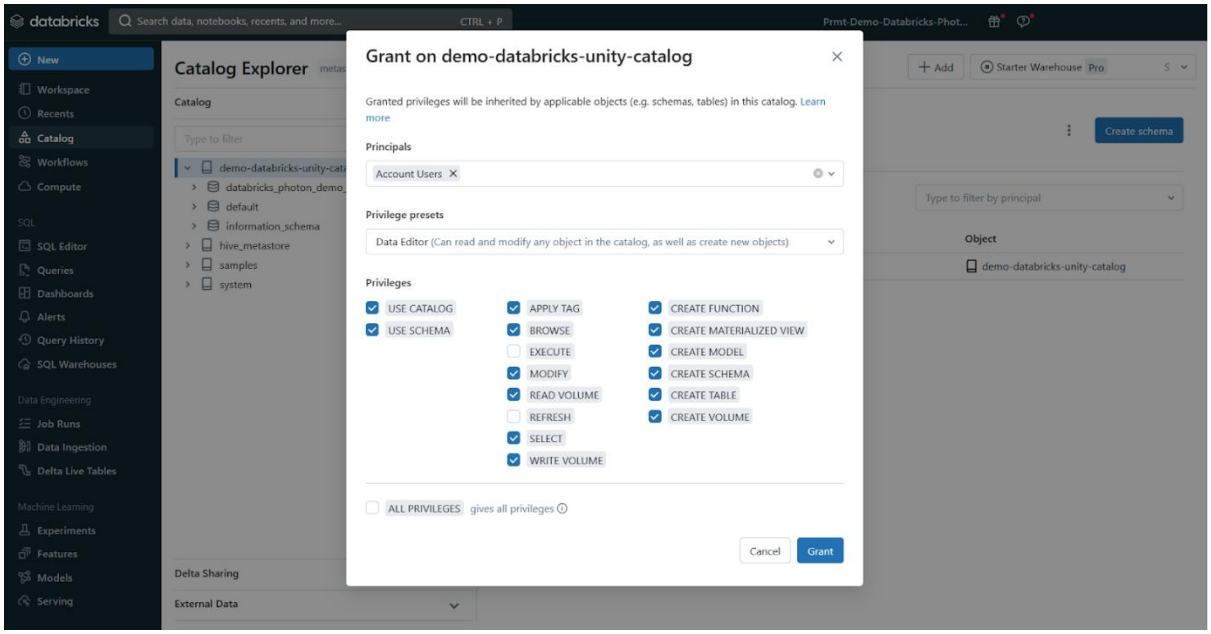


3. Assign the metastore to workspaces.



Managing Users & Permissions

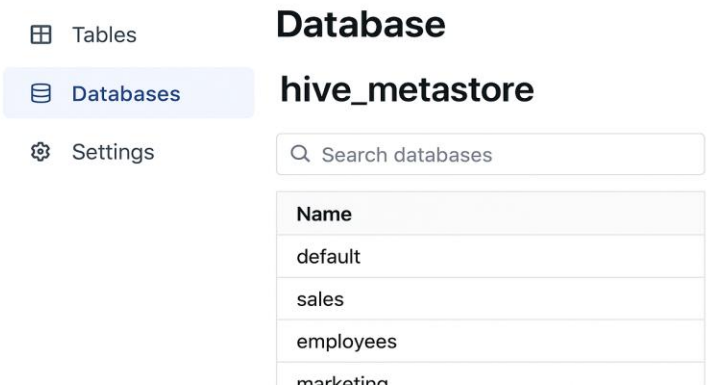
1. Add users, groups, and roles.
2. Assign privileges like CREATE, SELECT, ALTER, DROP.



6. Creating and Using Catalogs & Schemas

```
CREATE CATALOG IF NOT EXISTS project_catalog;
USE CATALOG project_catalog;
```

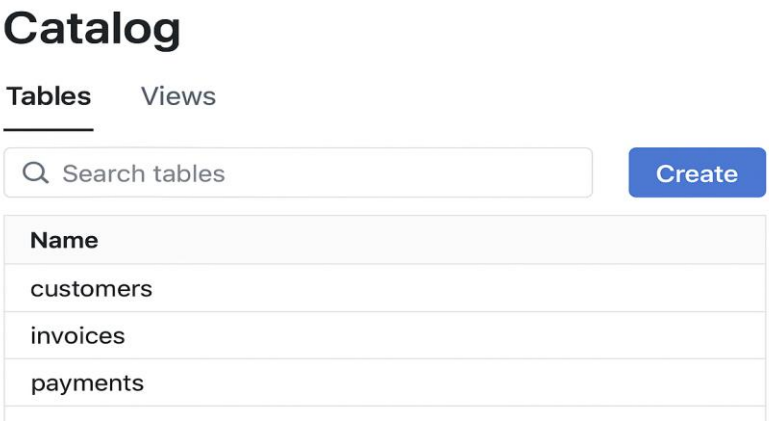
```
CREATE SCHEMA IF NOT EXISTS project_schema;
USE project_schema;
```




```
CREATE TABLE IF NOT EXISTS sample_table (
  id INT, name STRING
);

INSERT INTO sample_table VALUES (1, 'Unity'), (2,
'Catalog');

SELECT * FROM sample_table;
```



6. Best Practices

- Use catalogs as isolation units (e.g., dev, test, prod).
- Apply group ownership instead of individuals.
- Implement row/column security via dynamic views.
- Use audit logs for monitoring.
- Enforce least privilege access.
- Use Delta Sharing for secure external collaboration.

7. Limitations

- Requires Databricks Runtime 11.3 LTS or later.
- No support for bucketing in tables.
- Limited support for R workloads.
- No custom partition schemes.
- Object names limited to 255 chars and stored in lowercase.

8. Conclusion

Unity Catalog provides a centralized, secure, and scalable way to manage data governance in Databricks. With its powerful features like fine-grained access control, lineage tracking, and Delta Sharing, it is a big step forward compared to legacy Hive Metastore.

Adopting Unity Catalog helps organizations achieve better compliance, collaboration, and efficiency in their data and AI projects.