Coding Challenge

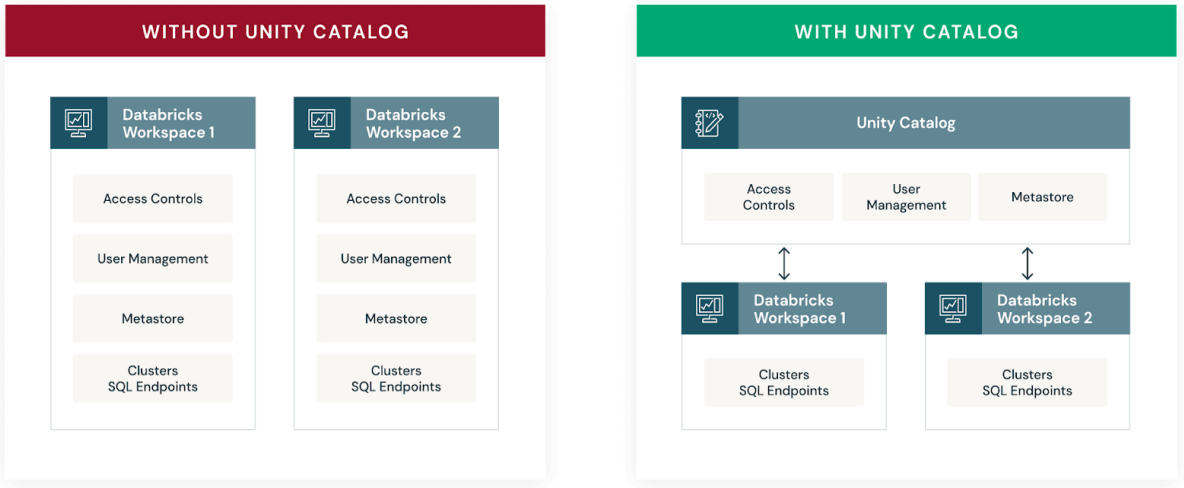
Bavatharani S

Introduction

Data has become one of the most valuable assets in the modern world, driving innovation and decision-making across industries. However, without proper management, data often becomes inconsistent, insecure, and difficult to use.

To address this challenge, data governance ensures that information remains accurate, secure, and accessible only to authorized users. Recognizing the need for a unified governance solution, Databricks introduced Unity Catalog, a centralized layer for managing data and AI assets across any cloud platform.

Unity Catalog provides fine-grained access control, auditing, lineage tracking, and data discovery, all within the Databricks Lakehouse Platform. By consolidating governance into a single framework, it simplifies administration while improving compliance and collaboration. This project explores Unity Catalog's architecture, features, and implementation through a practical use case.



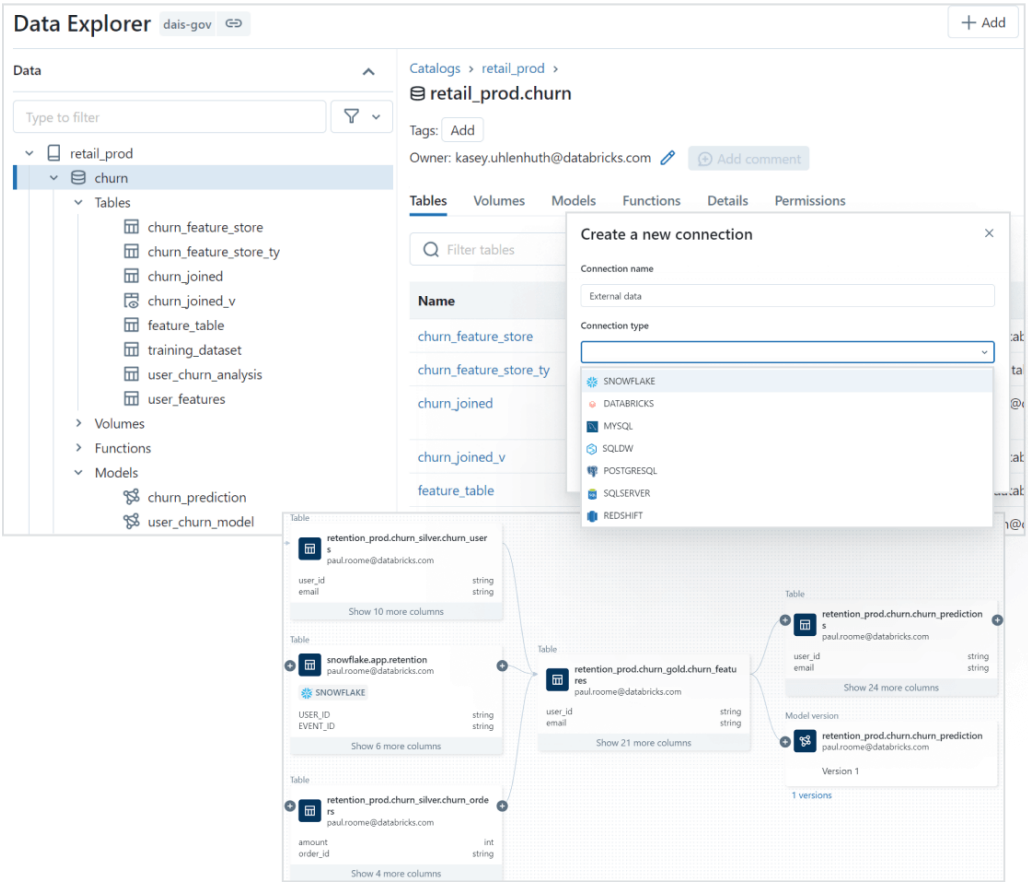
Key features and benefits of Databricks Unity Catalog:

- Define once, secure everywhere
- Standards-compliant security model
- Built-in auditing and lineage
- Data discovery
- System tables (Public Preview)
- Managed storage
- External data access

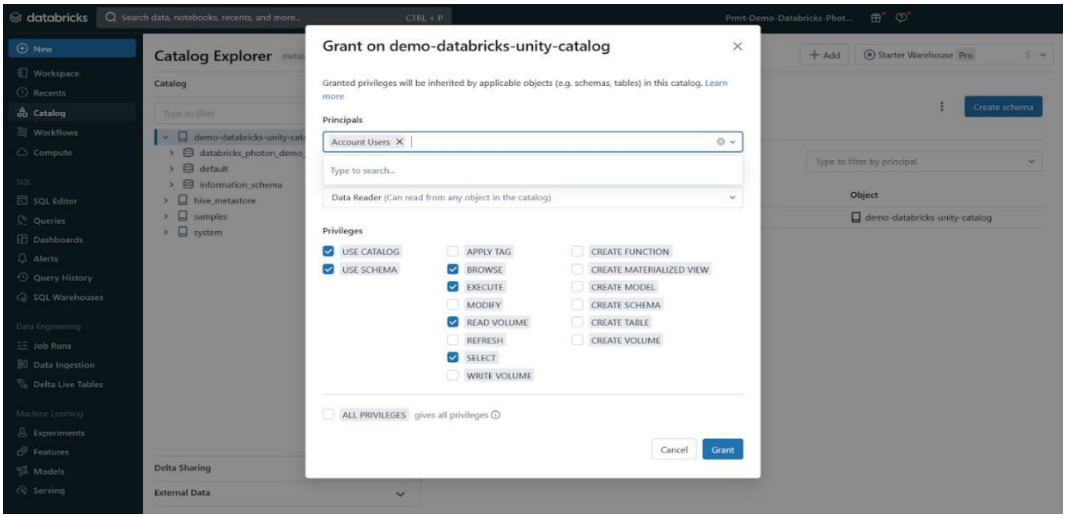
1. Databricks Unity Catalog Architecture

Unity Catalog is organized into several key layers:

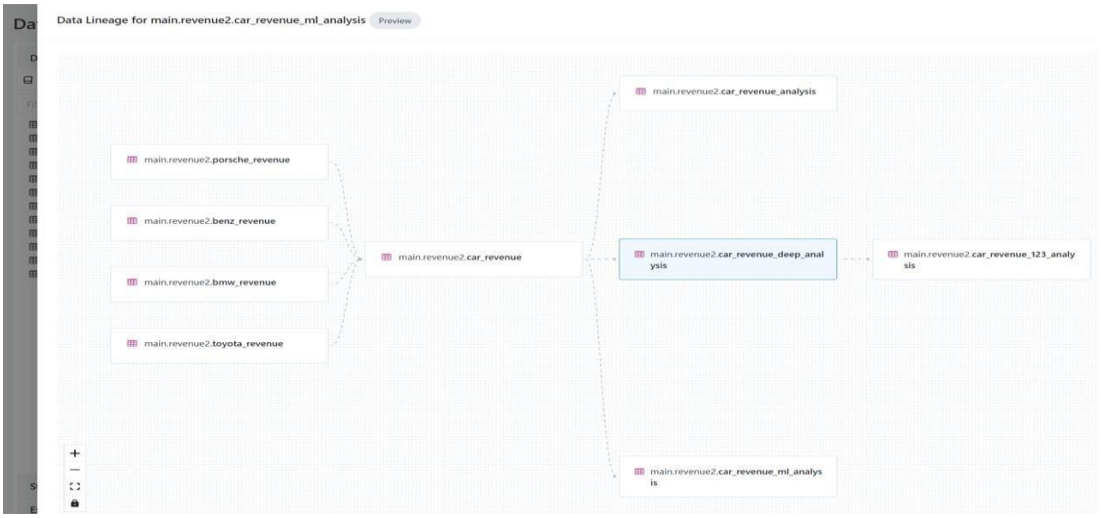
- Unified Governance Layer – Provides central governance for all data/AI assets.



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

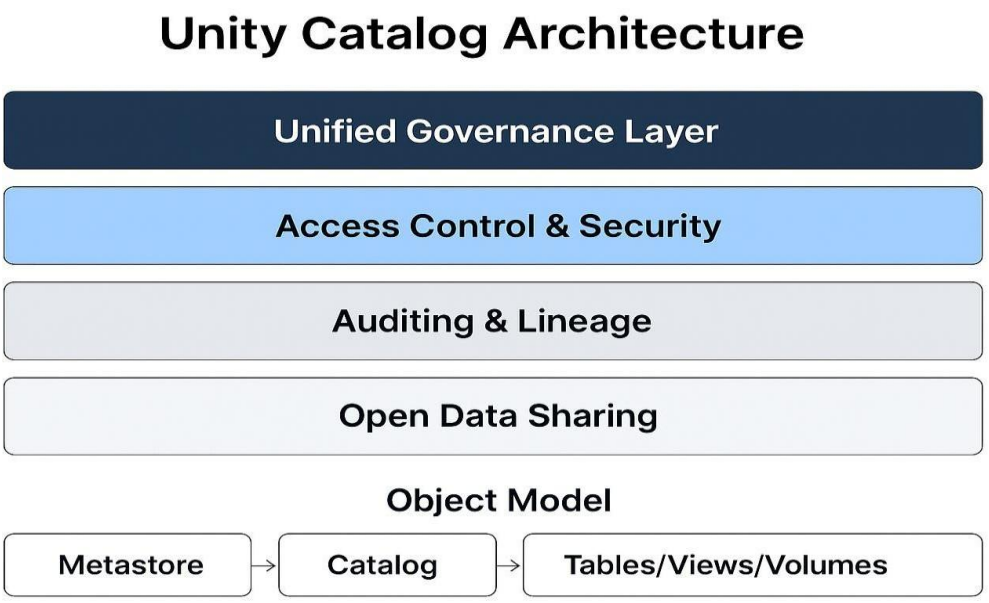


- Auditing & Lineage – Tracks who accessed what data and how it was transformed.

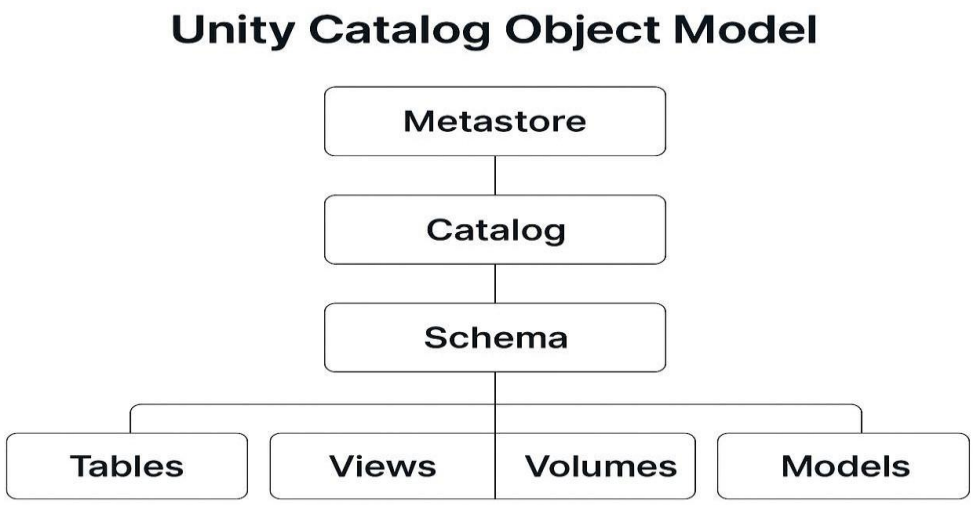


- Open Data Sharing – Uses Delta Sharing for secure cross-cloud data exchange.

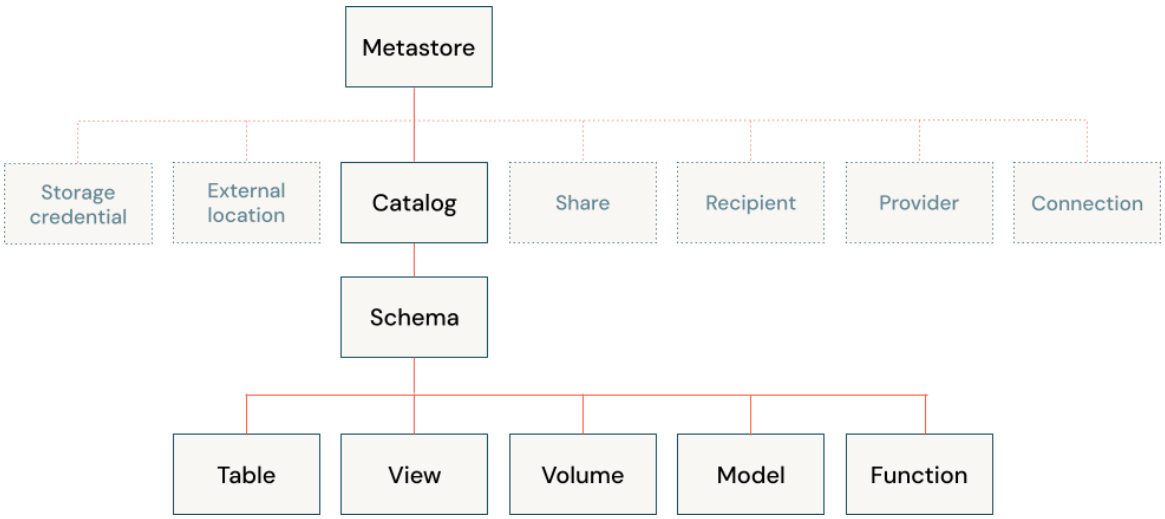
- Object Model – Hierarchy:
    - Metastore → Catalog → Schema → Tables/Views/Volumes/Models



➢ 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.



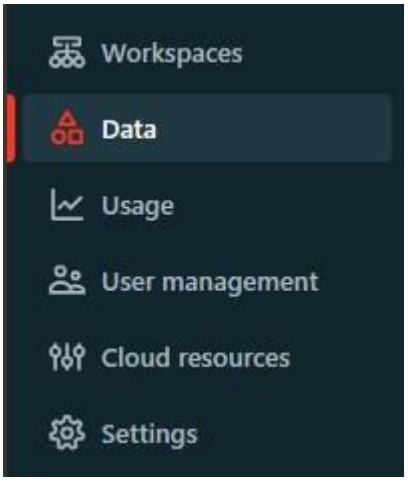
- Unity Catalog vs Hive Metastore

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.

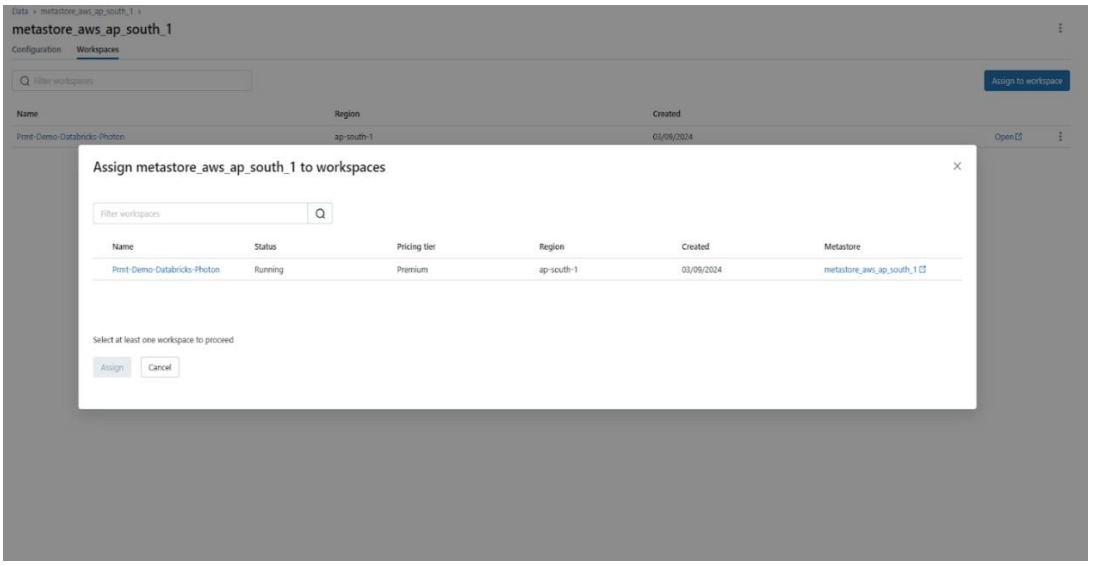
➢ How to Create Unity Catalog Metastore (AWS)

Creating a Metastore (AWS Example)

- Configure storage (S3 bucket).
- Create an IAM Role with access to storage.
- Create the Metastore in Databricks and assign it to a workspace. Enabling Unity Catalog for Workspace
  - Log in as account admin.
  - Navigate to Data → Metastore.

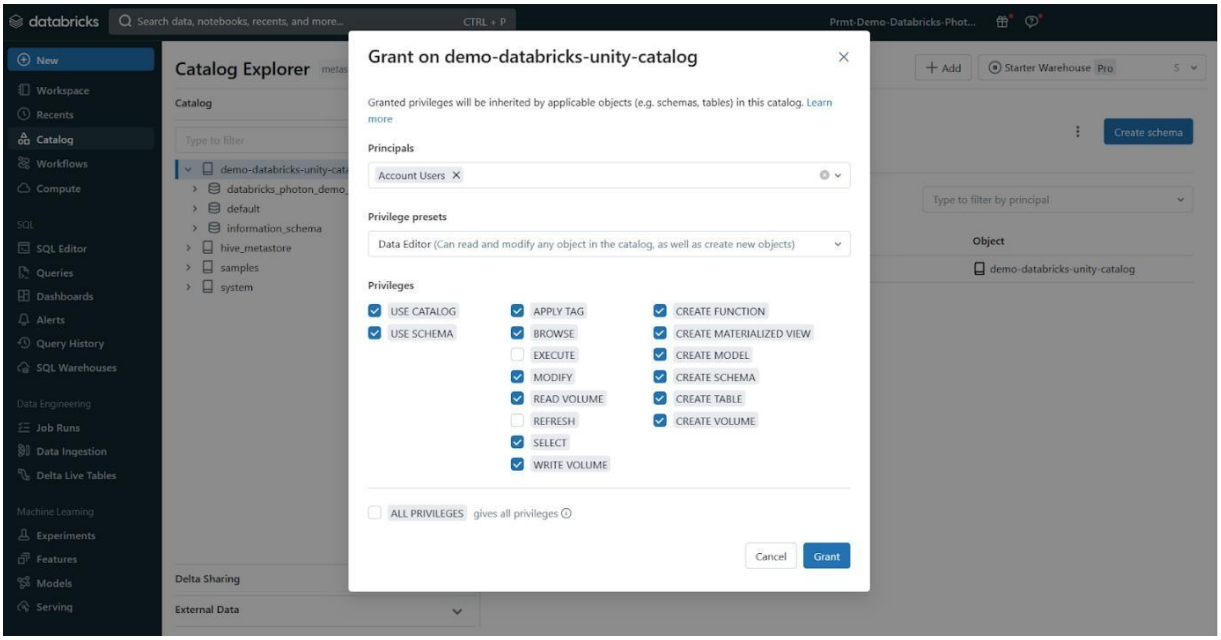


  - Assign the metastore to workspaces.



➢ Managing Users & Permissions

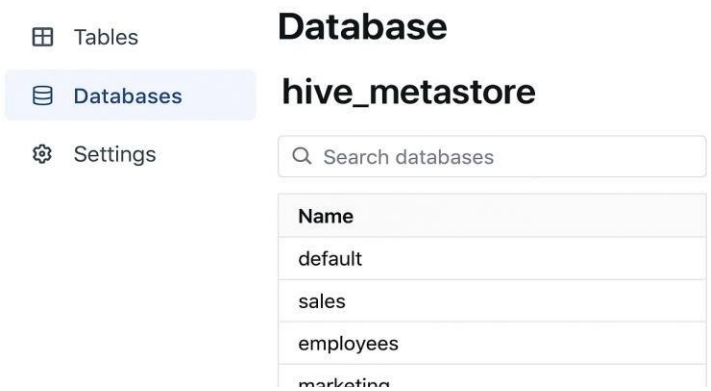
1. Add users, groups, and roles.
2. Assign privileges like CREATE, SELECT, ALTER, DROP.



➢ Creating and Using Catalogs & Schemas

CREATE CATALOG IF NOT EXISTS project_catalog; USE CATALOG project_catalog;

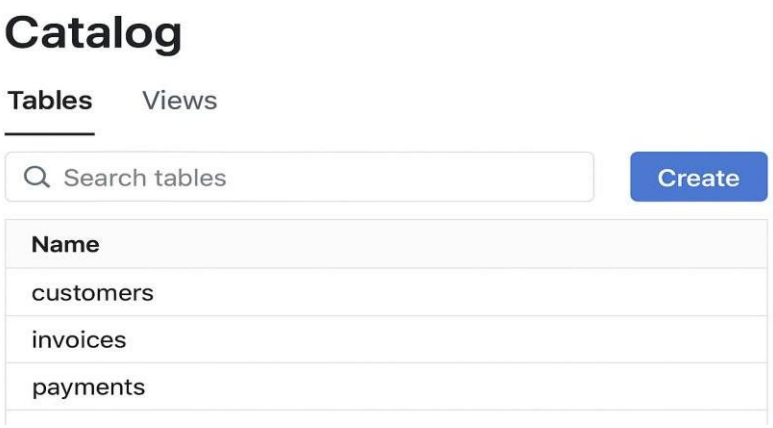
CREATE SCHEMA IF NOT EXISTS project_schema; USE project_schema;




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

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

SELECT * FROM sample_table;
```



➢ Best Practices

- Organize data by creating separate catalogs for different environments such as development, testing, and production.
- Assign ownership to groups rather than individual users to simplify management.
- Apply row-level and column-level security using dynamic views to protect sensitive data.
- Enable and regularly review audit logs to track user activities and ensure compliance.
- Follow the principle of least privilege, granting only the minimum access required.
- Leverage Delta Sharing for controlled and secure data exchange across platforms or external partners.

➢ Limitations

- Unity Catalog requires Databricks Runtime 11.3 LTS or higher for full functionality.

- Table bucketing is currently not supported in Unity Catalog.
- R language workloads have limited compatibility with row and column-level security.
- Custom partitioning schemes cannot be applied to Unity Catalog tables.
- Naming restrictions exist, with object names limited to 255 characters and stored only in lowercase.

➢ Conclusion

Unity Catalog introduces a unified and scalable solution for managing governance across data and AI assets in Databricks. By offering advanced capabilities such as detailed access control, auditing, lineage tracking, and secure data sharing, it goes beyond the traditional Hive Metastore. The adoption of Unity Catalog allows organizations to strengthen compliance, improve data collaboration, and enhance efficiency in cloud-based data projects. Its role in simplifying governance makes it a key enabler for organizations moving toward modern data-driven ecosystems.