Unity Catalog Mini-Project - Summary

1. Purpose & Importance

Unity Catalog in Databricks provides a unified governance solution for data and AI assets across clouds and platforms. It ensures secure, consistent, and fine-grained access control while enabling centralized metadata management, data lineage tracking, and secure data sharing. It eliminates the complexity of using third-party governance tools by being natively integrated into Databricks.

2. Key Features

- Centralized governance for all data and AI assets.
- Fine-grained access control at the column, row, and object levels.
- Data lineage tracking for visibility into data flow and transformations.
- Built-in auditing to monitor data access and changes.
- Delta Sharing for secure, cross-platform data sharing.
- Support for managed and external data locations.
- Search and discovery capabilities through tagging and documentation.

3. Architecture & Object Model

Unity Catalog follows a hierarchical model:

• Metastore – Top-level metadata store.
• Catalog – Logical container for schemas and assets.
• Schema – Contains tables, views, and volumes.
• Tables/Views/Volumes – Data assets, structured or unstructured.

Assets are referenced using a three-part naming convention:
<catalog>.<schema>.<object>.

4. Step-by-Step Setup

1. Ensure you have Databricks Premium plan or above with admin privileges.
2. Create or identify cloud storage (ADLS/S3/GCS) for managed data.
3. Enable hierarchical namespace in the storage account.
4. Create a Unity Catalog Metastore in the Databricks Account Console.

5. Assign the metastore to one or more workspaces.
6. Configure cluster access modes (Shared or Single User) for Unity Catalog compatibility.
7. Create catalogs, schemas, and tables as needed.
8. Grant privileges to users and groups for data access.

5. Best Practices

- Use catalogs as isolation units (e.g., by environment or data sensitivity).
- Apply storage isolation at the catalog/schema level when needed.
- Grant permissions to groups, not individual users.
- Leverage privilege inheritance to simplify access management.
- Use dynamic views for row/column-level security.
- Audit and monitor data access regularly.
- Use Delta Sharing for secure collaboration across domains.

6. Limitations

- Requires Databricks Runtime 11.3 LTS or later.
- No support for bucketing or certain custom partition schemes.
- Dynamic views for row/column-level security not supported in R workloads.
- Some UDF types are limited depending on Databricks Runtime version.
- Cannot use workspace-local groups in GRANT statements (must use account-level groups).
- Certain naming restrictions for objects and columns.
- Multi-region writes can cause consistency issues.