Data Engineering Batch 1 **Date: 21/02/2024**

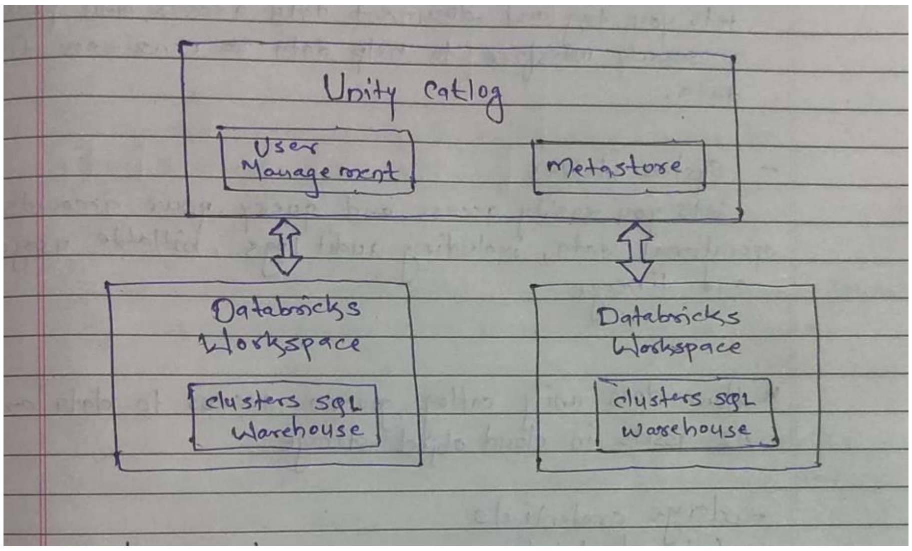
Name: Pradip Bochare

Azure Databricks Coding Assessment Question 2

Explain Overview of 3 level namespace and creating Unity Catalog objects.

❖ Unity Catalog

Unity Catalog provides centralized access control, auditing, lineage, and data discovery capabilities across Azure Databricks workspaces.



❖ Key features of Unity Catalog include:

- Define once, secure everywhere: Unity Catalog offers a single place to administer data access policies that apply across all workspaces.
- Standards-compliant security model: Unity Catalog's security model is based on standard ANSI SQL and allows administrators to grant permissions in their existing data lake using familiar syntax, at the level of catalogs, databases, tables, and views.

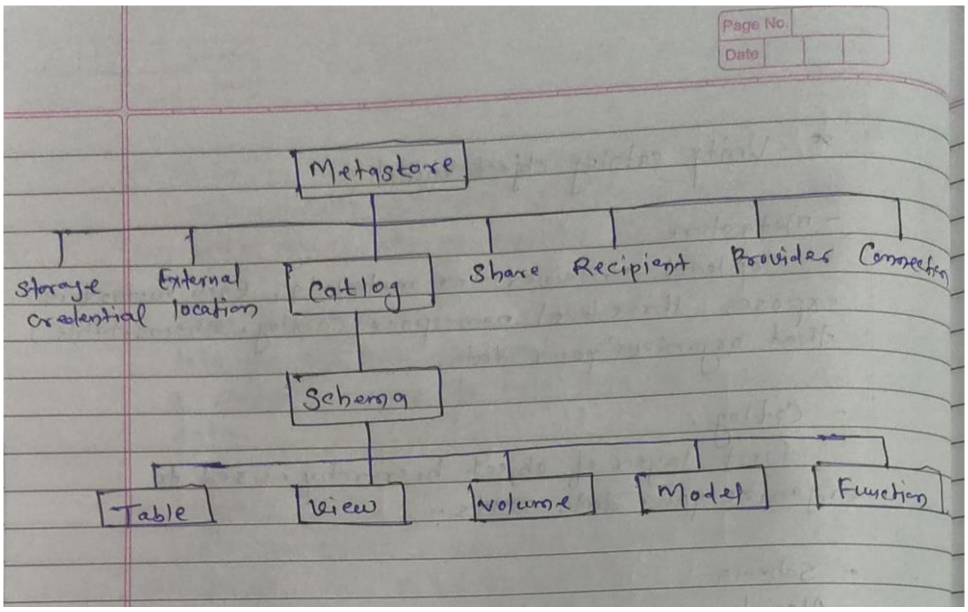
- Built-in auditing and lineage: Unity Catalog automatically captures user-level audit logs that record access to your data. Unity Catalog also captures lineage data that tracks how data assets are created and used across all languages.
- Data discovery: Unity Catalog lets you tag and document data assets, and provides a search interface to help data consumers find data.
- System tables: Unity Catalog lets you easily access and query your account's operational data, including audit logs, billable usage, and lineage.

❖ Concepts to manage relationships between data in Azure Databricks and cloud object storage

- Storage credentials encapsulate a long-term cloud credential that provides access to cloud storage.
- External locations contain a reference to a storage credential and a cloud storage path.
- Managed storage locations associate a storage location in an Azure Data Lake Storage Gen2 container in your Azure account with a metastore, catalog, or schema.
- Volumes provide access to non-tabular data stored in cloud object storage.
- Tables provide access to tabular data stored in cloud object storage.

❖ Unity Catalog object model

- Metastore: The top-level container for metadata. Each metastore exposes a three-level namespace (catalog.schema.table) that organizes your data.
- Catalog: The first layer of the object hierarchy, used to organize your data assets.
- Schema: Also known as databases, schemas are the second layer of the object hierarchy and contain tables and views.
- Tables, views, and volumes: At the lowest level in the data object hierarchy are tables, views, and volumes. Volumes provide governance for non-tabular data.
- Models: Although they are not, strictly speaking, data assets, registered models can also be managed in Unity Catalog and reside at the lowest level in the object hierarchy.



❖ Three-Level Namespace

- In Unity, the three-level namespace refers to the hierarchical structure used for organizing game objects, scripts, assets, and other elements within a Unity project.
- The three levels typically consist of the following:
    - Project: The top-level namespace, representing the entire Unity project. It encompasses all assets, scenes, scripts, and settings associated with the project.
    - Folder: The second-level namespace, used for organizing assets within the project. Folders can contain subfolders and assets of various types.
    - Asset: The lowest level, representing individual assets such as textures, models, scripts, audio files, etc.
- This hierarchical structure helps developers to organize and manage their project assets efficiently, improving readability, maintenance, and collaboration within the development team.

- ❖ Creating Unity Catalog Objects:

- In Unity, catalog objects refer to reusable assets or prefabs that can be instantiated and used within the game scenes.
- To create a catalog object in Unity:
  - Prepare the Asset: Create or import the asset (e.g., a 3D model, a script, a particle effect) that you want to use as a catalog object.
  - Prefab Creation: If the asset is meant to be reused multiple times, convert it into a Prefab. A Prefab is a reusable game object stored as an asset in Unity.
  - Catalog Organization: Organize the Prefabs within the Unity project structure, typically within folders corresponding to their type or purpose.
  - Instantiate in Scenes: Drag-and-drop the Prefabs from the Project window into the scenes where you want to use them. Alternatively, instantiate them programmatically via scripts.
- By creating catalog objects, developers can efficiently reuse assets across multiple scenes, promoting modularity, consistency, and productivity in game development workflows.