Flask API project using Databricks and Unity Catalog

Business Overview:

The project's primary objective is to create a robust data processing and analysis platform using a combination of Flask-based APIs, Docker for containerization, Databricks for data processing, data encryption for security, and Unity Catalog for data cataloging and lineage tracking. The dataset being utilized is related to climate data.

Services and Benefits:

- **Flask-Based AP**I: Flask serves as the framework for creating the API. Users can submit parameters through the API, requesting specific data based on criteria such as city ID, date range, etc.
- **Docker**: The Flask API is containerized using Docker, facilitating seamless deployment and scalability. Docker's microservices architecture ensures efficient resource utilization.
- **Databricks**: Databricks is used for backend data processing. It receives user parameters from the Flask API, applies transformations, and stores the processed data in a Delta format within Azure Blob Storage. This setup allows for scalable and efficient data processing.
- **Data Encryption**: The project incorporates data encryption techniques to protect sensitive information like email addresses, phone numbers, and salaries. Cryptography modules are employed to encrypt this data, enhancing data security and confidentiality.
- **Unity Catalog**: Unity Catalog within Databricks is pivotal in data sharing and lineage tracking. It enables data sharing within and outside the organization while providing lineage information. In the future, Unity Catalog may also support data encryption against Personally Identifiable Information (PII) flags.

The project's impact on businesses is significant. It streamlines data processing, enhances data security, and provides users with a seamless access and analysis experience. By using Unity Catalog, organizations can gain better visibility into data lineage, improving data governance and compliance. Overall, this project empowers businesses to make data-driven decisions efficiently and securely, which is crucial in today's data-centric world.

Aim:

To build a dockerized Flask API application that fetches user data as a date range, then queries, analyses, and stores the hourly weather data in Databricks and Azure Storage. Unity Catalog is integrated to track and understand data lineage and handling of Personally Identifiable Information (PII) flags.

Dataset Description

For this project, we will be using an hourly weather dataset with the following fields:

- City_id
- Date
- Hour (in 2400 format)
- Temp
- Wind_speed
- Description
- Precipitation
- Humidity

The demonstration of PII flags will be explained using an Employee dataset with the following fields:

- Employee_id
- Name
- Email
- Phone
- Hire_date
- Salary

Finally, SQL DDL commands will demonstrate the Unity Catalog data lineage by creating two tables, Menu and Dinner.

Tech Stack:
Language: Python, SQL
Framework: Flask
Services: Azure Resource Group, Azure Storage Account, Databricks, Unity Catalog, Docker

Databricks:
Databricks is a cloud-based big data analytics platform that simplifies the processing and analysis of large datasets. It combines data engineering, data science, and machine learning into a unified workspace, providing data ingestion, transformation, exploration, and collaborative data-driven decision-making tools. Databricks leverages Apache Spark under the hood, offering scalable and efficient data processing capabilities. Its collaborative features enable teams to work on data projects together. Databricks is widely used for data engineering, data warehousing, and machine learning tasks. It is a powerful platform for organizations seeking to extract valuable insights from their data in a scalable and collaborative environment.

Docker:
Docker is a containerization platform that simplifies the deployment and management of software applications. It packages applications and their dependencies into portable containers, ensuring consistency across different environments. These containers are isolated, lightweight, and can run on any system that supports Docker, enabling developers to build, ship, and run applications consistently from development to production. Docker's efficiency and scalability make it popular in microservices architectures and DevOps practices, as it streamlines the development and deployment process, reduces compatibility issues, and enhances resource utilization by allowing multiple containers to run on a single host efficiently.

Unity Catalog:
Unity Catalog in Databricks is a centralized hub for access control, auditing, lineage tracking, and data discovery across Databricks workspaces. Key features include unified data access policies, ANSI SQL-based security, automatic audit logs, and data lineage tracking. Users can tag and document data assets, making data discovery efficient. The object hierarchy includes Metastore, Catalog, Schema (databases), Volume (governance for non-tabular data), and Table (tabular data). Unity Catalog streamlines data governance, security, and accessibility, enhancing the management and utilization of data assets in Databricks workspaces.

Key Learning Takeaways:
- Understanding the Project Overview and Architecture
- Understanding the datasets used
- Create a Flask API and send requests via the application
- Explanation of backend databricks code
- Setting up resources in Azure to configure Databricks and Unity Catalog
- Deploying Flask app to Docker
- Working with Data Encryption and Anonymity
- Introduction to Unity Catalog
- Access Connectivity and Metastore creation
- Setting up Unity Catalog in Databricks Workspace
- Creating Data Lineage in Databricks
- Exploring Delta Sharing in Unity Catalog
- Adding Security Feature to the Flask API