Azure Databricks End-to-End Real-Time Data Engineering Project

✅ Objective

To design and implement a real-world, enterprise-level data engineering project using **Azure Databricks**, covering the complete data pipeline lifecycle: **data ingestion, transformation, modeling, and analytics**, leveraging the **Medallion architecture** (bronze, silver, and gold layers). The goal is to simulate how data engineering is done in production environments using modern tools like **Delta Live Tables**, **Unity Catalog**, and **Azure Data Lake**.

🧱 Architecture Overview

Cloud Platform & Tools:

- **Cloud Provider**: Microsoft Azure
- **Data Lake**: Azure Data Lake Gen2 (with hierarchical namespace)
- **Processing Engine**: Apache Spark on Azure Databricks
- **Data Governance**: Unity Catalog
- **ETL Orchestration**: Databricks Workflows and Delta Live Tables
- **Data Format**: Parquet (optimized columnar storage)
- **Visualization**: Power BI (connected through Databricks SQL Warehouse)

Architecture Layers:

- **Bronze Layer**: Raw data ingestion
- **Silver Layer**: Data cleansing, transformation, and enrichment
- **Gold Layer**: Business-ready analytical models (e.g., star schema)

📂 Data Used

- **Domain**: Retail/E-commerce
- **Files**: Orders, Customers, Products, Regions
- **Source Format**: Parquet (columnar format; schema stored in the footer)
- **Why Parquet?**
    - Efficient for big data processing
    - Schema evolution support
    - Optimized for read-heavy workloads

Why These Datasets?

These are intuitive and familiar datasets (orders, customers, products) which make it easier to focus on **engineering and pipeline logic** rather than interpreting obscure data.

🛠️ Implementation Steps

1. Azure Setup

- Created an **Azure free trial account**.
- Created a **Resource Group**, which is like a folder to organize Azure services.
- Provisioned a **Storage Account**, configured for **Data Lake Gen2**.
- Created containers for:
    - `source` (initial raw files)
    - `bronze`, `silver`, `gold` (Medallion architecture layers)
    - `metastore` (for Unity Catalog managed tables)

2. Databricks Workspace Setup

- Created a **Databricks workspace** linked with Azure.
- Configured a **Databricks Access Connector** to enable secure communication between Databricks and Azure Data Lake.
- Assigned permissions to allow Databricks to read/write from the storage account.

3. Unity Catalog Configuration

- **Why Unity Catalog?**
    - Centralized governance, security, and schema management.
- Created a **Unity Metastore** to manage schemas and tables.
- Assigned metastore to the workspace.
- Created **external locations** (for each container: `bronze`, `silver`, `gold`, `source`) for access control.
- Created **catalog** and **schemas** inside the Unity Catalog.

4. Data Ingestion (Bronze Layer)

- Ingested data from Azure Data Lake using **Databricks Data Ingestion (No-code)** tool.
- Performed **incremental data loading** using **Spark Structured Streaming**.
- Applied **idempotency** to ensure no duplicate records are processed.
- Data saved in Delta format for better versioning and performance.

5. Data Transformation (Silver Layer)

- Used **PySpark DataFrame APIs** for transformations.

- Applied **Python OOP principles** to create reusable, modular code.
- Created **functions** within Unity Catalog for shared business logic (e.g., custom logic for formatting, null handling).
- Stored transformed data in **Delta format** within the `silver` schema.

6. Business Modeling (Gold Layer)

- Created **Star Schema**:
  - **Dimension Tables**: customers, products, regions
  - **Fact Table**: orders
- Implemented **Slowly Changing Dimensions (SCD)**:
  - **Type 1**: Manually using PySpark (overwrites old data)
  - **Type 2**: Using **Delta Live Tables** for tracking historical changes
- Built **ETL Workflows** using Databricks Workflows feature to automate transformation pipelines.

7. Data Serving & BI Integration

- Enabled **SQL Warehouse** (serverless compute) to query gold layer tables.
- Established **Power BI connection** via Databricks SQL endpoint.
- Demonstrated dashboard reporting capability for analysts.

🚀 Key Features Demonstrated

Feature	Description
Medallion Architecture	Designed bronze, silver, and gold layers with clear data lineage
Delta Lake	Used Delta format for reliability, time travel, and performance
Incremental Loading	Processed new data using Spark Structured Streaming
Data Governance	Set up Unity Catalog for access control and table management
ETL Orchestration	Created pipelines using Delta Live Tables and Databricks Workflows
Python + PySpark	Combined object-oriented Python with PySpark for modular ETL logic
Real-Time Concepts	Idempotency, schema evolution, audit control
SCD Type 1 & 2	Modeled historical tracking using code and Delta Live Tables
SQL Warehouse	Optimized serving layer for BI tools like Power BI