1. **Project Overview**

This project builds an end-to-end scalable data pipeline using Apache Spark on Databricks for processing raw e-commerce data from disparate sources like CSV, Excel and JSON thereby generating enriched datasets in bronze, silver and gold layer.

1. **Data Sources**

* **Customer Data** : Excel file (.xlsx)
* **Order Data** : JSON file (.json)
* **Product Data** : CSV file (.csv)

1. **Architecture Layers**
2. **Bronze Layer :**

* **Format** : Delta
* **Tables** : bronze\_customer, bronze\_orders, bronze\_products
* **Logic** :
  + - Predefined schema definitions
    - File-type based loading
    - Standardising column names
    - Stored as Delta tables in hive metastore

1. **Silver Layer :**

* **Tables** : silver\_customer, silver\_products
* **Customer Logic** :
  + - Cleaned names , phone checks, address parsing
    - Validated Column names
    - Introduced surrogate\_key : Customer\_PK
* **Product Logic** :
  + - Validated Column names
    - Introduced surrogate\_key : Product\_PK

1. **Gold Layer :**

* **Tables** : gold\_orders, gold\_profit\_agg
* **Orders Logic** :
  + - Cleaned date format
    - Rounded Profit to 2 decimals
    - Joined with enriched customer and product data
* **Profit\_Agg Logic** :
  + - Grouped dataset on Year, Customer\_Name, Category, Sub\_Category
    - Aggregated Profit
    - Partitioned by Year

**4. Validation & Testing**

* **Framework :** PyTest
* **Test Cases**:
  + - Customers with Profit less than 10%
    - Column schema validation
    - Data Assertions like name and phone cleanup
    - Address parsing correctness
    - Filtering bad data

**5. Optimisation Techniques**

* **Delta Lake Partitioning**: Partitioning the gold orders table by Year keeping in mind future scope of data volume upgrades
* **Adaptive Query Language(AQE)**: Enabled to dynamically optimise query plans at runtime like switch join strategies or coalesce partitions
* **Predicate PushDown:**  Filters applied early to minimise the I/O and shuffle using the Catalyst optimiser
* **Selective Columns :** Only required columns selected during transformation to reduce data shuffling
* **Explicit Schema Definition:** To avoid schema overheads
* **Caching Dataframes :** Caching intermediate dataframes to reduce recomputation overheads

**6. Tools & TechStack**

* **Distribution Compute :** Databricks Community Edition(DBR 13.3 LTS)
* **Language :** PySpark (Python 3.10)
* **Storage :** Delta Lake
* **Orchestration :**Databricks Job ( Notebook-based)
* **Git Integration :** GitHub

**7. Error Handling**

* Try-Except in File Readers

**8. Project Structure**

src/  
 |--- bronze/  
 |--- silver/  
 |--- gold/  
 |--- transform/  
 |--- test/  
 |--- resources/  
 |--- sql\_analytics/  
 |--- docs/  
 main.py  
 README.md

**8. Improvements & TODOs**

* Add main.py to orchestrate and resuse the modules
* Improve the TDD approach
* Add environment variables and runtime parameters like source and target tables , filepath
* Segregate gold table notebooks and add into Gold folder
* Move from Databricks Community to Enterprise edition leveraging functionalities
* Refine and functionalise modular classes and incorporate it in the current methodology
* Apply incremental logic and SCDs using Change Data Feed feature for Data loads
* Integrate with Unity Catalog for Data Governance
* Integrate with CI/CD using poetry + Jenkins
* Schedule workflows using Dolphin Scheduler or TIDAL

**9. Conclusion**

This project establishes a robust, efficient and scalable foundation in retail e-commerce analytics.It ensures data quality, modular transformation, performance optimization and aggregated business driven gold layer for Presentation layer.