Approach and Optimizations

The ETL pipeline was designed for efficiency, scalability, and data integrity. Since an open-source tool was required, I used **Pandas in Jupyter Notebook** to develop the solution. Two notebooks were created:

1. **SibongileChiwandireAssessment.ipynb** – A basic ETL pipeline that extracts data from the API, transforms it, and loads it into the MySQL table **Customer.Transactions** row by row, without optimizations or data validation.
2. **Optimized.ipynb** – An improved version incorporating **performance optimizations and data quality checks**, loading data into **Customer.TransactionsOptimized** in batches to improve efficiency.

Summary of Optimizations

Performance Optimization:

- Implemented **parallel processing** using Python's **ThreadPoolExecutor** to reduce database transactions.
- Used **batch inserts** instead of row-by-row loading to optimize MySQL performance.
- Added **indexes** on frequently queried columns to improve query speed.

Data Quality Checks:

- Ensured no missing categorizations (**product_category, spend_category**).
- Prevented negative transaction amounts.
- Logged any discrepancies, including missing or invalid data.

Key Considerations

- **Handling Missing Product Categories:** The dataset included an unlisted column, **spend_category**, which contained similar data to **product_category**. Many rows had missing values in **product_category**, while **spend_category** had valid data. To ensure data completeness, the script replaced **NULL** values in **product_category** with values from **spend_category**.
- **Monthly Spend Trends Limitation:** I extracted only **one month's data** from the API as per the assessment instructions, making it impossible to generate **yearly spend trends**. However, I provided the necessary SQL script to perform this analysis when more data is available.
- **Preventing Duplicates in Continuous Loads:** Since the pipeline is expected to run continuously, I implemented a **Unique Constraint** (**customer_id, product_id, transaction_date**) in **Customer.TransactionsOptimized** to prevent duplicate records. In the **Customer.Transactions** table, I used **INSERT IGNORE** to skip duplicates during continuous loads.

This optimized approach balances **performance, scalability, and data integrity**, ensuring an efficient and reliable ETL pipeline.