DEBRE BERHAN UNIVERSITY

COLLEGE OF COMPUTING

DEPARTMENT OF SOFTWARE ENGINEERING

PREPARED BY : NATNAEL HABTE

ID : 0593/13

ASSIGNMENT TITLE : BUILDING AN END-TO-END DATA PIPELINE

SUBMITTED TO : DERBEW FELASMAN(MSc)

SUBMITION DATE : FEBRUARY 13 /2025

DEBRE BERHAN ,ETHIOPIA

1 Overview

This ETL pipeline takes raw e-commerce transaction data in a CSV file, cleans and transforms it, and loads it into a PostgreSQL database. The data is then used for analysis and visualization in Power BI.

- Data Source: kz.csv (E-commerce transactions)
- Processing: Python (pandas, sqlalchemy, psycopg2)
- Database: PostgreSQL
- Visualization: Power BI

2 Extract: Loading Raw Data

2.1 Source Data

The raw data comes from a CSV file (kz.csv) containing e-commerce transactions.

2.2 Code: Loading CSV File

```python
import pandas as pd

# File paths
file_path = "kz.csv"
output_file = "transformed_kz.csv"

try:
    # Load raw data
    df = pd.read_csv(file_path, low_memory=False)
    print(f"Successfully loaded {file_path}")
except Exception as e:
    print(f"Error loading {file_path}: {e}")
    exit()

```

Findings:

- The dataset contains duplicates and missing values.
- Some fields like price contain invalid values (non-numeric data).
- event_time is not properly formatted.

3 Transform: Data Cleaning & Processing

3.1 Design Choices & Cleaning Steps

Cleaning Step	Purpose
Remove Duplicates	Ensure unique orders and users
Handle Missing Values	Prevent data inconsistencies
Convert Price to Numeric	Standardize financial data
Standardize Text Columns	Ensure consistency in categorical data
Format DateTime	Ensure proper timestamp handling

3.2 Code: Cleaning and Transforming Data

```python
# Remove duplicate orders
df.drop_duplicates(subset=["order_id"], keep="first", inplace=True)

# Ensure user_id is unique
df.drop_duplicates(subset=["user_id"], keep="first", inplace=True)

# Handle missing values
df["price"] = pd.to_numeric(df["price"], errors='coerce').fillna(0.0)  # Replace invalid prices with 0.0
df["category_id"] = df["category_id"].fillna("unknown")
df["category_code"] = df["category_code"].fillna("unknown")
df["brand"] = df["brand"].fillna("unknown")
df["event_time"] = pd.to_datetime(df["event_time"], errors='coerce').fillna(pd.Timestamp("1970-01-01"))

# Standardize text columns
df["category_code"] = df["category_code"].astype(str).str.lower()
df["brand"] = df["brand"].astype(str).str.lower()

# Save the transformed data
df.to_csv(output_file, index=False)
print(f"\nTransformed data saved to {output_file}")
```

Findings:

- Missing values were handled appropriately.
- The dataset contained duplicate user_id values, which were removed to maintain uniqueness.

4 Load: Storing Data in PostgreSQL

4.1 Database Schema

```
CREATE TABLE IF NOT EXISTS ecommerce_transactions (
        user_id VARCHAR(50) PRIMARY KEY,
        order_id VARCHAR(50),
        product_id VARCHAR(50),
        category_id TEXT,
        category_code TEXT,
        brand TEXT,
        price NUMERIC,
        event_time TIMESTAMP
);

```

Design Choice:

- order_id is the primary key to ensure each user appears only once in the database.

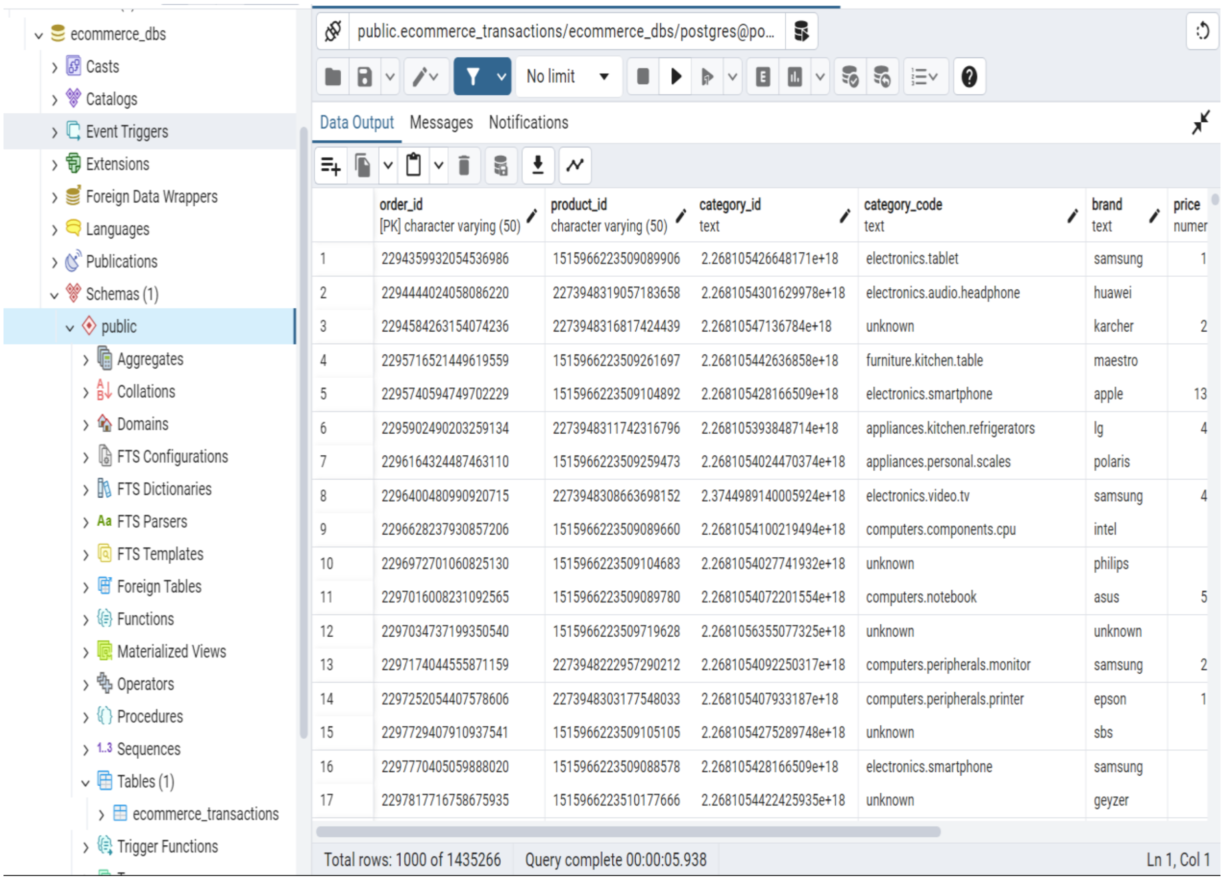
4.2 Code: Loading Data into PostgreSQL

```
from sqlalchemy import create_engine
import psycopg2

# Database connection details
DB_USER = "postgres"
DB_PASS = "nathab"
DB_HOST = "localhost"
DB_PORT = "3000"
DB_NAME = "ecommerce_db"

try:
    # Connect to PostgreSQL
    conn = psycopg2.connect(
        dbname=DB_NAME, user=DB_USER, password=DB_PASS, host=DB_HOST, port=DB_PORT
    )
    cursor = conn.cursor()
    print("\nConnected to PostgreSQL successfully!")

```

```
    # Create table
    create_table_query = """
    CREATE TABLE IF NOT EXISTS ecommerce_transactions (
        user_id VARCHAR(50) PRIMARY KEY,
        order_id VARCHAR(50),
        product_id VARCHAR(50),
        category_id TEXT,
        category_code TEXT,
        brand TEXT,
        price NUMERIC,
        event_time TIMESTAMP
    );
    """
    cursor.execute(create_table_query)
    conn.commit()
    print("Table checked/created successfully!")

    # Load data
    engine = create_engine(f"postgresql://{DB_USER}:{DB_PASS}@{DB_HOST}:{DB_PORT}/{DB_NAME}")
    df.to_sql("ecommerce_transactions", con=engine, if_exists="append", index=False)
    print("Data successfully loaded into PostgreSQL")

except Exception as e:
    print(f"Error loading data into PostgreSQL: {e}")

finally:
    cursor.close()
    conn.close()
    print("Database connection closed.")
```

Findings:

- Unique constraint errors were encountered due to duplicate user_id values.
- These were resolved by ensuring user_id is unique before insertion.



5 Power BI Integration

5.1 Connecting PostgreSQL to Power BI

Steps:

1. Go to Home → Get Data → Database → PostgreSQL
2. Enter connection details:

- Server: localhost
- Database: ecommerce_db
- Username: postgres
- Password: *****

3. Click Load or Transform Data if preprocessing is needed.

6 Data Visualization in Power BI

6.1 Creating Scatter Plot (Similar to Python)

To replicate the Python scatter plot in Power BI:

Steps:

1. Drag price to the X-axis

2. Drag qty_ordered to the Y-axis

3. Drag category_name_1 to the legend (color differentiation)

4. Set Scatter Chart as the visualization type.

7 Summary & Key Learnings

7.1 Project Achievements

- Extracted raw data from CSV.
- Cleaned & transformed data (handled missing values, duplicates, standardization).
- Stored data in PostgreSQL, ensuring user_id is unique.
- Connected PostgreSQL to Power BI for analysis.

7.2 Future Improvements

- Optimize performance by indexing frequently queried columns.
- Enhance error handling for database operations.
- Implement automated ETL pipeline for continuous data updates.