Colleague of computing

Department of software engineering

Course Name: Fundamental of Big Data analysis and Business intelligence

Course code: SEng5112

Title: Building an End-to-End Data Pipeline

Individual Assignment

Name: Fikade Tibebe

Id: 1400042

Submitted to: Derbew F.

Submission date: 02/06/2017 EC.

Contents

1. Data Source Identification & Understanding

Large Dataset

A comprehensive e-commerce dataset was sourced from Kaggle. This dataset contains over 1 million rows, capturing various aspects of an e-commerce business, including sales transactions, customer information, order details, payment records, and product listings.

The dataset was downloaded as multiple CSV files, which are structured to represent different entities in the e-commerce ecosystem. These files include:

- **df_Customers.csv** – Contains customer details such as customer ID, location, and zip code prefix.
- **df_OrderItems.csv** – Stores itemized order details, linking products to specific orders.
- **df_Orders.csv** – Holds information on orders, including timestamps, customer ID references, and order status.
- **df_Payments.csv** – Includes payment transaction details, such as payment types and transaction values.
- **df_Products.csv** – Contains product information, including product categories and descriptions.

Understanding Dataset Relationships

Before moving forward with data extraction and transformation, an in-depth analysis of the dataset structure and relationships was conducted. The dataset follows a relational structure where:

- **Customers** place **Orders**, which consist of multiple **Order Items**.
- Each **Order Item** corresponds to a **Product**.
- **Payments** are linked to specific **Orders** with details on transaction methods and amounts.

Potential Use Cases

This dataset is valuable for various data analytics and business intelligence applications, including:

- **Sales Performance Analysis** – Understanding revenue trends, peak seasons, and growth opportunities.
- **Customer Behavior Insights** – Analyzing purchasing patterns and customer segmentation.

- **Product Demand Forecasting** – Identifying best-selling products and optimizing inventory management.
- **Payment Method Preferences** – Evaluating preferred transaction methods and optimizing payment systems.

By thoroughly examining the dataset, we ensure a structured approach to data processing, enabling effective extraction, transformation, and visualization in the later stages of the pipeline.

2. Data Extraction

Extracting E-Commerce Data

The dataset was downloaded in CSV format from Kaggle and stored in a dedicated folder named **csvData**. This folder contains multiple CSV files, each representing different aspects of the e-commerce dataset, such as customer information, order details, payments, and product listings.

To efficiently handle and process these datasets, the **pandas** library in Python was used. The CSV files were loaded into dataframes and then concatenated into a single dataframe for further processing.

Code Implementation

```
import pandas as pd

# Load CSV files
csv_files = [
    "csvData/df_Customers.csv",
    "csvData/df_OrderItems.csv",
    "csvData/df_Orders.csv",
    "csvData/df_Payments.csv",
    "csvData/df_Products.csv"
]

df_list = [pd.read_csv(file) for file in csv_files]
df = pd.concat(df_list, ignore_index=True)
```

By using pandas.read_csv(), each CSV file is read into a separate dataframe, and pd.concat() is used to merge them into a single dataframe, ensuring all data is in one place for further transformation.

3. Data Transformation

Cleaning and Preprocessing

Once the raw data was extracted, several data cleaning and preprocessing steps were applied to ensure accuracy and consistency before loading it into the database. The transformation steps included:

- **Handling Missing Values:** Missing values in categorical columns were replaced with the most frequent value (mode), while numerical columns were filled with the median value. This approach preserves data integrity and prevents bias from extreme values.
- **Removing Duplicates:** Duplicate records were identified and removed to maintain data consistency and avoid redundancy.
- **Formatting Data Types:** Columns containing date values were converted into the proper datetime format, and numeric fields were formatted appropriately to enhance computational efficiency and ensure compatibility with the database schema.

Code Implementation

```
# Handle missing values
df.dropna(how='all', inplace=True)

df.fillna({
    col: df[col].mode()[0] if df[col].dtype == 'object' else df[col].median()
    for col in df.columns
}, inplace=True)

# Convert date columns
date_columns = ['order_purchase_timestamp', 'order_approved_at']
for col in date_columns:
    if col in df.columns:
        df[col] = pd.to_datetime(df[col], errors='coerce')
```

This transformation ensures that the dataset is clean, structured, and ready for analysis and visualization.

4. Data Loading

Database Schema Design

After cleaning and transforming the dataset, it was necessary to store the structured data in a PostgreSQL database. The database schema was designed based on relational principles to optimize storage and retrieval. The key tables included:

- ✓ **Customers Table:** Stores details about customers, such as their unique ID, location, and state.
- ✓ **Orders Table:** Contains records of customer orders, including timestamps, order status, and customer references.
- ✓ **Products Table:** Holds product-related details such as product categories and descriptions.
- ✓ **Payments Table:** Stores payment transaction records, including payment type and transaction amount.
- ✓ **Order Items Table:** Represents the relationship between orders and products, storing item-level details.

Loading Data into PostgreSQL

The cleaned dataset was inserted into the PostgreSQL database using the psycopg2 library. The following code snippet demonstrates how customer data was inserted while ensuring no duplicate records were added using the ON CONFLICT DO NOTHING clause.

Code Implementation

```
import psycopg2

conn = psycopg2.connect(
   dbname="ecommrece_one", user="postgres", password="123123", host="localhost",
port="5432"
)
cursor = conn.cursor()

# Insert customers data
for _, row in df[['customer_id', 'customer_zip_code_prefix', 'customer_city',
'customer_state']].drop_duplicates().iterrows():
   cursor.execute("""
      INSERT INTO Customers (customer_id, customer_zip_code_prefix, customer_city,
customer_state)
      VALUES (%s, %s, %s, %s)
      ON CONFLICT (customer_id) DO NOTHING;
   """, (row['customer_id'], row['customer_zip_code_prefix'], row['customer_city'],
row['customer_state']))
```

This step ensures that the transformed data is efficiently loaded into the database, maintaining referential integrity and data consistency.

5. Data Visualization and Insights

Connecting to Power BI

After successfully loading the transformed data into the **PostgreSQL** database, the next step was to connect it to **Power BI**, a powerful business intelligence tool, to create interactive dashboards and gain meaningful insights from the dataset.

To establish the connection, Power BI's **PostgreSQL connector** was used. This allowed direct querying of the database, enabling real-time data exploration and visualization. Various charts and reports were built to analyze sales trends, customer behaviors, and transaction patterns.

The following key dashboards were created in Power BI:

- **Sales Trends Over Time:** This visualization provided a detailed view of how sales fluctuated over different periods, helping identify **peak sales seasons, high-demand months, and periods of low activity**.
- **Customer Segmentation:** Customers were categorized based on their purchase behavior, geographic location, and frequency of transactions. This helped in understanding which regions contributed the most to sales and identifying customer clusters with similar buying habits.
- **Payment Method Distribution:** The usage of different payment methods was analyzed to determine customer preferences. This insight was particularly useful for **businesses to optimize payment options** and enhance customer convenience.

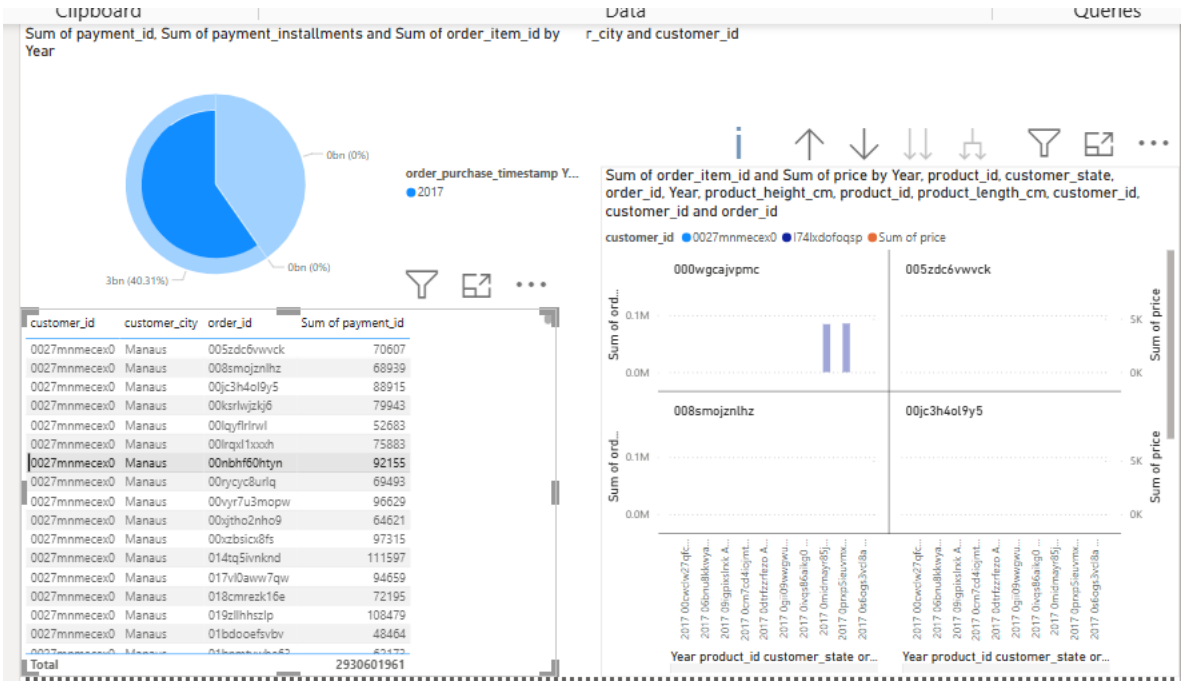

Figure 1 visulization

Key Insights from Data Visualization

Through Power BI visualizations, several important patterns were uncovered:

- **Sales Trends:** Seasonal fluctuations in sales were observed, with certain products performing better during specific months. This information is useful for **inventory planning and marketing strategies**.
- **Customer Segmentation:** The analysis revealed **regional differences in purchasing behaviors**, with some locations showing a higher concentration of frequent buyers. These insights help businesses tailor their marketing campaigns to different demographics.
- **Payment Preferences:** The data showed a preference for certain payment methods over others, which can assist businesses in **offering promotions on preferred payment platforms** or improving the checkout experience for customers.

The use of Power BI made it easy to interpret complex data through interactive visual representations, making data-driven decision-making more efficient.

Conclusion

This project successfully implemented a complete **end-to-end data pipeline** for processing and analyzing e-commerce data. The structured ETL (Extract, Transform, Load) workflow ensured seamless data handling from raw extraction to insightful business analytics.

The workflow involved:

1. **Extracting data** from CSV files sourced from Kaggle.
2. **Cleaning and transforming** the raw data by handling missing values, duplicates, and formatting issues.
3. **Loading the processed data** into a **PostgreSQL database** to maintain structured storage and easy retrieval.
4. **Connecting Power BI to PostgreSQL** and visualizing key insights through interactive dashboards.

Reference

https://www.kaggle.com/datasets/bytadit/ecommerce-order-dataset