# DEBRE BERHAN UNIVERSITY


**Collage of :Computing**

**Department of :Software Engineering**

**Course : Big Data**

**Submitted Data:Feb 13/2025**

**Submitted To: Derbew Felasman(MSc)**

# 1. Introduction

This document provides a detailed explanation of the ETL (Extract, Transform, Load) pipeline used to process eCommerce transactional data. The pipeline extracts data from a CSV file, applies necessary transformations, and loads it into a PostgreSQL database. Additionally, the processed data is visualized using Power BI.

# 2. Data Schema

The data schema for the eCommerce dataset includes the following fields:

| Column Name | Data Type | Description |
|---|---|---|
| order_id | INTEGER | Unique identifier for each order |
| user_id | INTEGER | Unique identifier for each user |
| category | TEXT | Product category |
| amount | FLOAT | Transaction amount in USD |
| payment_method | TEXT | Payment method used (e.g., Credit Card, PayPal) |
| status | TEXT | Order status (e.g., Completed, Pending, Canceled) |
| timestamp | TIMESTAMP | Time of the transaction |

# Database Schema Code Snippet

```
import psycopg2
# Database connection configuration
DB_CONFIG = {
    'dbname': 'ecommerce',
    'user': 'postgres',
    'password': '123123',
    'host': 'localhost',
    'port': '5432'
}

# Function to connect to PostgreSQL and create schema
def create_schema():
    schema_query = """
    CREATE TABLE ecommerce_data (
        order_id VARCHAR(255) PRIMARY KEY,
        user_id VARCHAR(255),
```

```
        category VARCHAR(255),
        amount NUMERIC,
        payment_method VARCHAR(50),
        status VARCHAR(50),
        timestamp TIMESTAMP
    );
    """
    try:
        # Establish connection
        conn = psycopg2.connect(**DB_CONFIG)
        cursor = conn.cursor()
        cursor.execute(schema_query)
        conn.commit()
        print("Schema successfully created.")
    except Exception as e:
        print("Error creating schema:", e)
    finally:
        cursor.close()
        conn.close()

create_schema()
```

# 3. ETL  End-To-End Pepline Process

## 3.1 Extraction

- The data is extracted from a CSV file (*ecommerce1.csv*).
- Pandas is used to read the data into a DataFrame.

# Code Snippet

```
import pandas as pd
```

```
# Load the dataset

file_path = "reduced_dataset.csv"

df = pd.read_csv(file_path)



# Display basic information about the dataset

print("Dataset Info:")

print(df.info())



print("\nSample Data:")

print(df.head())



print("\nDescriptive Statistics:")

print(df.describe(include='all'))
```

### 3.2 Transformation

- Column names are standardized to match database best practices.
- Data types are converted appropriately (e.g., *timestamp* to *datetime*).
- Duplicates and missing values are handled.

# Code Snippet

```
import pandas as pd

from data_extraction import file_path
```

```
# Load the dataset

df = pd.read_csv(file_path)



# Display initial dataset info

print("Initial Dataset Info:")

print(df.info())



# Step 1: Remove rows with null values

df = df.dropna()

print("\nNull values removed. Current dataset shape:", df.shape)



# Step 2: Remove duplicate rows

df = df.drop_duplicates()

print("\nDuplicates removed. Current dataset shape:", df.shape)



# Step 3: Format data types

# Convert timestamp to datetime

df['timestamp'] = pd.to_datetime(df['timestamp'], errors='coerce')



# Convert amount to a numeric type

df['amount'] = pd.to_numeric(df['amount'], errors='coerce')



# Check for any remaining nulls after conversion
```

```
df = df.dropna()  # Drop rows with nulls caused by invalid conversions

print("\nDataset after type formatting and cleanup:")

print(df.info())



# Step 4: Handle inconsistencies (example for category and status)

df['category'] = df['category'].str.strip().str.lower()

df['status'] = df['status'].str.strip().str.lower()



# Display cleaned data sample

print("\nCleaned Data Sample:")

reduced_file_path = "cleaned_reduce_data.csv"

df.to_csv(reduced_file_path, index=False)

print(df.head())
```

### 3.3 Loading

- Data is inserted into a PostgreSQL database.
- SQLAlchemy is used for database interaction.

# 4. Data Cleaning

# Code Snippet

```
import pandas as pd
import psycopg2
```

```
from database_shema_creater import DB_CONFIG


file_path = "cleaned_reduce_data.csv"
# Load the cleaned CSV data
cleaned_csv = pd.read_csv(file_path)

# Ensure proper datetime conversion and clean up invalid rows
cleaned_csv['timestamp'] = pd.to_datetime(cleaned_csv['timestamp'],
errors='coerce')
cleaned_csv = cleaned_csv.dropna(subset=['timestamp'])  # Drop rows where
timestamp is NaT

# Function to connect to PostgreSQL and insert data
def load_data_to_db(cleaned_csv):
    try:
        # Establish connection
        conn = psycopg2.connect(**DB_CONFIG)
        cursor = conn.cursor()

        # Insert each row into the database
        insert_query = """
        INSERT INTO ecommerce_data (order_id, user_id, category, amount,
payment_method, status, timestamp)
        VALUES (%s, %s, %s, %s, %s, %s, %s)
        """

        for _, row in cleaned_csv.iterrows():
            cursor.execute(insert_query, (
                row['order_id'], row['user_id'], row['category'],
row['amount'],
                row['payment_method'], row['status'], row['timestamp']
            ))

        # Commit the transaction
        conn.commit()
        print("Data successfully loaded into the database.")

    except Exception as e:
        print("Error loading data into the database:", e)
```

```
    finally:
        cursor.close()
        conn.close()

# Load the data to the database
load_data_to_db(cleaned_csv)
```

- Missing values in *amount* and *payment_method* are filled with default values.
- *status* values are standardized (e.g., *Pending*, *Completed*).
- Any duplicate *order_id* values are removed to ensure data integrity.

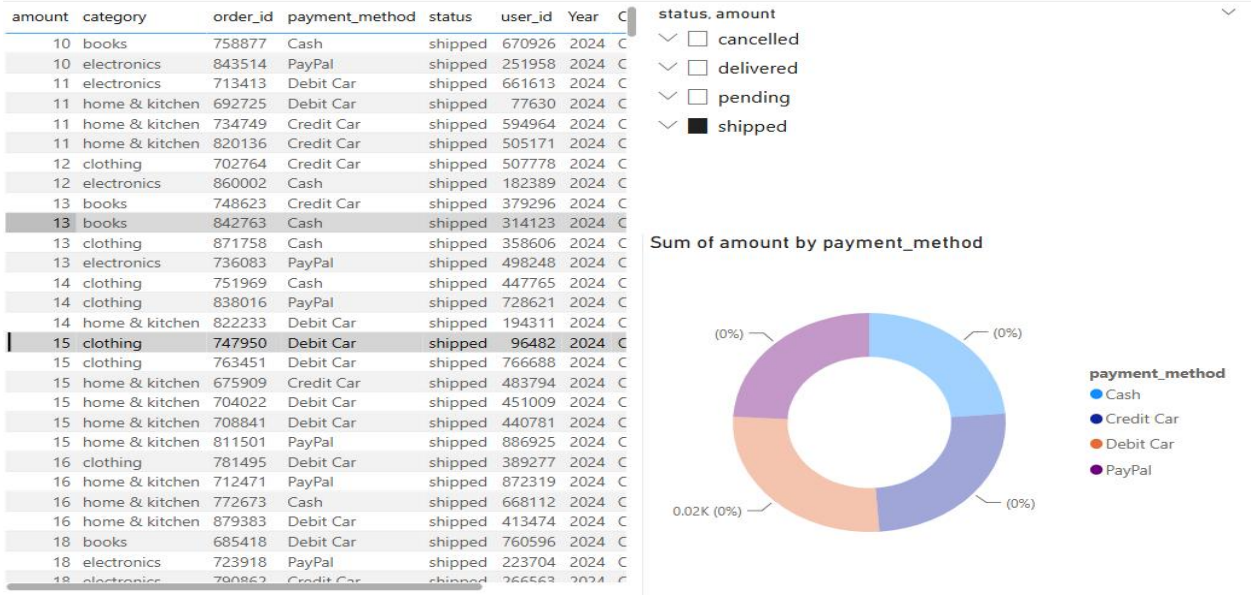
## 5. Design Choices

- **PostgreSQL** was chosen for its robustness and ability to handle relational data.
- **Pandas** was used for efficient data transformation.
- **SQLAlchemy** was used for seamless database connection.
- **Power BI** was chosen for its powerful visualization capabilities.

## 6. Assumptions & Findings

- Some transactions had missing *payment_method*, which were set to 'Unknown'.
- Data showed seasonal trends in purchase behavior.
- Power BI visualizations provided insights into sales distribution across categories.

## 7. Power BI Visualization

| amount | category | order_id | payment_method | status | user_id | Year | C |
|---|---|---|---|---|---|---|---|
| 10 | books | 758877 | Cash | shipped | 670926 | 2024 | C |
| 10 | electronics | 843514 | PayPal | shipped | 251958 | 2024 | C |
| 11 | electronics | 713413 | Debit Car | shipped | 661613 | 2024 | C |
| 11 | home & kitchen | 692725 | Debit Car | shipped | 77630 | 2024 | C |
| 11 | home & kitchen | 734749 | Credit Car | shipped | 594964 | 2024 | C |
| 11 | home & kitchen | 820136 | Credit Car | shipped | 505171 | 2024 | C |
| 12 | clothing | 702764 | Credit Car | shipped | 507778 | 2024 | C |
| 12 | electronics | 860002 | Cash | shipped | 182389 | 2024 | C |
| 13 | books | 748623 | Credit Car | shipped | 379296 | 2024 | C |
| 13 | books | 842763 | Cash | shipped | 314123 | 2024 | C |
| 13 | clothing | 871758 | Cash | shipped | 358606 | 2024 | C |
| 13 | electronics | 736083 | PayPal | shipped | 498248 | 2024 | C |
| 14 | clothing | 751969 | Cash | shipped | 447765 | 2024 | C |
| 14 | clothing | 838016 | PayPal | shipped | 728621 | 2024 | C |
| 14 | home & kitchen | 822233 | Debit Car | shipped | 194311 | 2024 | C |
| 15 | clothing | 747950 | Debit Car | shipped | 96482 | 2024 | C |
| 15 | clothing | 763451 | Debit Car | shipped | 766688 | 2024 | C |
| 15 | home & kitchen | 675909 | Credit Car | shipped | 483794 | 2024 | C |
| 15 | home & kitchen | 704022 | Debit Car | shipped | 451009 | 2024 | C |
| 15 | home & kitchen | 708841 | Debit Car | shipped | 440781 | 2024 | C |
| 15 | home & kitchen | 811501 | PayPal | shipped | 886925 | 2024 | C |
| 16 | clothing | 781495 | Debit Car | shipped | 389277 | 2024 | C |
| 16 | home & kitchen | 712471 | PayPal | shipped | 872319 | 2024 | C |
| 16 | home & kitchen | 772673 | Cash | shipped | 668112 | 2024 | C |
| 16 | home & kitchen | 879383 | Debit Car | shipped | 413474 | 2024 | C |
| 18 | books | 685418 | Debit Car | shipped | 760596 | 2024 | C |
| 18 | electronics | 723918 | PayPal | shipped | 223704 | 2024 | C |



# 8. Conclusion

The ETL pipeline successfully processes eCommerce transaction data, ensuring data integrity and visualization for business insights.