
**DEBRE BERHAN UNIVERSITY**

**INSTITUTION Of TECHNOLOGY**

**COLLAGE Of COMPUTING**

**DEPARTMENT Of SOFTWARE ENGINEERING**

# FUNDAMENTAL OF BIG DATA ANALYTICS AND BUSINESS INTELLIGENCE (SEng5112)

**Prepared by: -**

**Name: Esmelealem Berhanu**

**Id: DBUR/1180/13**

**Submitted to: Mr. Derbew Felasman (MSc)**

**February, 2025 Debre Berhan, Ethiopia**

# Contents

**1. Overview** ..... 1

**2. Dataset Description** ..... 1

**3. Tools and Technologies Used** ..... 1

**4. ETL Pipeline Steps** ..... 2

1. **Extraction** ..... 2
2. **Transformation** ..... 3
3. **Loading** ..... 4

**5. Design Choices** ..... 6

**Data Schema** ..... 6

**6. Data Cleaning Processes** ..... 6

**7. Visualization and Insights** ..... 8

**8. Conclusion** ..... 11

**9. References** ..... 11

# 1. Overview

This document provides a comprehensive overview of the Extract, Transform, and Load (ETL) pipeline developed for processing the Retail Rocket e-commerce dataset. The pipeline is designed to extract raw data from multiple sources, transform it into a structured format suitable for analysis, and load it into a PostgreSQL database. The final step involves visualizing the data using Microsoft Power BI to extract meaningful insights. The code is well-commented to facilitate understanding and maintenance.

# 2. Dataset Description

The dataset used in this ETL pipeline is sourced from the Retail Rocket ecommerce dataset, available on Kaggle. (https://www.kaggle.com/datasets/retailrocket/ecommerce-dataset/data)

The dataset comprises four primary files:

1. **Events.csv**: Contains user interactions such as product views, add-to-cart actions, and transactions.
2. **Item_properties_part1.csv** and **Item_properties_part2.csv**: Detail various attributes of the products.
3. **Category_tree.csv**: Describes the hierarchical structure of product categories.

The dataset is rich in user interaction data, making it ideal for analyzing customer behavior, product performance, and sales trends.

# 3. Tools and Technologies Used

Used pandas to load the CSV file into a DataFrame:

- **Python**: The primary programming language used for scripting the ETL process.
    - **Pandas**: For data manipulation and cleaning.
    - **SQLAlchemy**: For connecting to the PostgreSQL database.
    - **psycopg2**: PostgreSQL database adapter for Python.
- **PostgreSQL**: The relational database used for storing the cleaned and transformed data.
- **Git**: Version control for tracking changes in the code base.

- **GitHub**: For hosting the project repository and sharing the code.
- **Microsoft Power BI**: Used for creating interactive visualizations and dashboards from the cleaned data.

# 4. ETL Pipeline Steps

### 1. Extraction

The extraction phase involves loading the raw data from the CSV files into Pandas DataFrames. The following code snippet demonstrates this process:

**Code:**

```
import pandas as pd

# Load the behavior data (what users did)
events_df = pd.read_csv('./data/events.csv')

# Load the two parts of item properties (info about the products)
item_properties_part1 = pd.read_csv('./data/item_properties_part1.csv')
item_properties_part2 = pd.read_csv('./data/item_properties_part2.csv')

# Combine the two item properties files
item_properties_df = pd.concat([item_properties_part1, item_properties_part2], ignore_index=True)

# Load the category tree (how items are grouped)
category_tree_df = pd.read_csv('./data/category_tree.csv')

# Print to check if the data is loaded correctly
print(events_df.head())
print(item_properties_df.head())
print(category_tree_df.head())
```

**Comments:**

- The events.csv file is loaded into a Data Frame named events_df.
- The item_properties_part1.csv and item_properties_part2.csv files are loaded and combined into item_properties_df.

- The category_tree.csv file is loaded into category_tree_df.

## 2. Transformation

The transformation phase involves cleaning, merging, and preparing the data for analysis. The following steps were performed:

**Code:**

```
import pandas as pd

# Load the dataset
final_df = pd.read_csv('./data/cleaned_final_data.csv')

# Debugging: Check column names and dataset structure
print("Columns in final_df:", final_df.columns)
print("First few rows of final_df:\n", final_df.head())

# Step 1: Handle timestamp column
# Convert the 'timestamp' column to datetime format
final_df['timestamp'] = pd.to_datetime(final_df['timestamp'], errors='coerce')

# Drop rows with invalid timestamps (if any)
final_df = final_df.dropna(subset=['timestamp'])

# Step 2: Add 'day_of_week' column (as numbers)
# Extract the day of the week as an integer (Monday=0, Sunday=6)
final_df['day_of_week'] = final_df['timestamp'].dt.dayofweek

# Step 3: Clean the 'property' column
# Filter out rows where 'property' is not a numerical value
final_df = final_df[final_df['property'].astype(str).str.isnumeric()]

# Convert the 'property' column to integers (since it should now contain only numbers)
final_df['property'] = final_df['property'].astype(int)

# Step 4: Clean the 'value' column
# Standardize numerical values in the 'value' column
# Remove prefixes like 'n' and suffixes like '.000' and convert to numerical values
final_df['value'] = final_df['value'].astype(str).str.replace('n', '').str.replace('.000', '').str.strip()
final_df['value'] = pd.to_numeric(final_df['value'], errors='coerce')  # Convert to numeric, set invalid values to
NaN
```

```
# Handle zero values in the 'value' column
# Replace zeros with NaN if they represent missing data
final_df['value'] = final_df['value'].replace(0, pd.NA)

# Step 5: Drop the 'status' column (if it exists)
if 'status' in final_df.columns:
    final_df.drop(columns=['status'], inplace=True)

# Step 6: Handle missing values
# Drop rows with missing values in essential columns
final_df.dropna(subset=['visitor_id', 'action', 'item_id', 'property', 'value'], inplace=True)

# Step 7: Remove duplicates
final_df.drop_duplicates(inplace=True)

# Step 8: Ensure all columns have consistent and meaningful values
# Replace any remaining 'NaN' or 'None' values in the 'value' column with 0 (if appropriate)
final_df['value'].fillna(0, inplace=True)

# Debugging: Check final dataset structure
print("Columns after cleaning:", final_df.columns)
print("First few rows after cleaning:\n", final_df.head())

# Save the further cleaned data to a new CSV file
final_df.to_csv('./data/cleaned_final_data.csv', index=False)

print("Data cleaning completed!")
```

**Comments:**

The **cleaned_final_data.csv** file is loaded into a DataFrame named final_df for further processing.

- **Step 1**: The timestamp column is converted to datetime format, and any invalid timestamps are removed.
- **Step 2**: A new column, day_of_week, is added to indicate the day of the week (Monday=0, Sunday=6).
- **Step 3**: The property column is filtered to ensure it contains only numeric values and is converted to integers.
- **Step 4**: The value column is cleaned by:
    - Removing unnecessary prefixes/suffixes (like 'n' or '.000').
    - Converting it to numeric format.
    - Handling zero values (replacing them with NaN where necessary).

- **Step 5**: If a status column exists, it is removed.
- **Step 6**: Any rows with missing essential values (visitor_id, action, item_id, property, value) are dropped.
- **Step 7**: Duplicate rows are removed to ensure data consistency.
- **Step 8**: Remaining NaN or None values in the value column are replaced with 0 (if appropriate).

A final **debugging check** is performed to confirm that columns are cleaned correctly.

The cleaned dataset is saved back to **cleaned_final_data.csv** for use in the next stage of the pipeline.

### 3. Loading

The loading phase involves storing the transformed data into a PostgreSQL database. The following code snippet demonstrates this process:

**Code:**

```
from sqlalchemy import create_engine
import pandas as pd

# Load the cleaned data
final_df = pd.read_csv('./data/cleaned_final_data.csv')

# Connect to PostgreSQL
engine = create_engine('postgresql://postgres:2011eb3465@localhost:5432/ecommerce1_db')

# Load data into the database
final_df.to_sql('ecommerce_data', engine, if_exists='replace', index=False)

print("Data loaded into PostgreSQL!")
```

**Comments:**

- The transformed DataFrame final_df is loaded into a PostgreSQL table named ecommerce_data.
- The if_exists='replace' argument ensures that any existing table with the same name is replaced.

# 5. Design Choices

**Data Schema**

The transformed data schema includes the following fields:

- timestamp: Datetime of the event.
- visitorid: Unique identifier for the visitor.
- itemid: Unique identifier for the item.
- event: Type of event ('view' or 'transaction').
- property: Attribute of the item.
- value: Value corresponding to the item attribute.

This schema was designed to facilitate easy querying and analysis of user interactions and product attributes.

# 6. Data Cleaning Processes

The data cleaning process involved several steps to ensure the dataset was ready for analysis. Below is a detailed breakdown of the cleaning steps performed:

### 6.1 Timestamp Handling

- Original Dataset Issue: The timestamp column was in a format that needed to be converted to a human-readable datetime format.
- Cleaning Process:
    - The timestamp column was converted to a datetime format using pd.to_datetime().
    - Rows with invalid timestamps (e.g., NaT values) were dropped to ensure data integrity.
- Impact: This step ensures that the timestamp column is properly formatted for time-based analysis, such as identifying trends over time or analyzing user behavior by day of the week.

### 6.2 Adding 'day_of_week' Column

- Original Dataset Issue: The dataset did not include a column indicating the day of the week, which is useful for analyzing weekly trends.

- Cleaning Process: A new column, day_of_week, was added to the dataset. This column extracts the day of the week as an integer (Monday=0, Sunday=6) from the timestamp column.
- Impact: This addition allows for easy analysis of user behavior by day of the week, such as identifying peak activity days.

### 6.3 Cleaning the 'property' Column

- Original Dataset Issue: The property column contained non-numerical values, which could cause issues during analysis.
- Cleaning Process:
    - Rows where the property column contained non-numerical values were filtered out.
    - The property column was converted to integers to ensure consistency.
- Impact: This step ensures that the property column contains only numerical values, making it suitable for quantitative analysis.

### 6.4 Cleaning the 'value' Column

- Original Dataset Issue: The value column contained inconsistent formatting, such as prefixes (e.g., 'n') and suffixes (e.g., '.000'), which needed to be standardized.
- Cleaning Process:
    - Prefixes like 'n' and suffixes like '.000' were removed from the value column.
    - The value column was converted to numerical values using pd.to_numeric(). Invalid values were set to NaN.
- Zero values in the value column were replaced with NaN if they represented missing data.
- Impact: This step ensures that the value column contains clean, standardized numerical data, which is essential for accurate analysis.

### 6.5 Dropping the 'status' Column

- Original Dataset Issue: The status column (if it existed) was not relevant to the analysis.
- Cleaning Process: The status column was dropped from the dataset if it existed.
- Impact: Removing irrelevant columns simplifies the dataset and reduces unnecessary clutter.

### 6.6 Handling Missing Values

- Original Dataset Issue: The dataset contained missing values in essential columns, which could lead to incomplete or inaccurate analysis.
- Cleaning Process: Rows with missing values in essential columns (visitor_id, action, item_id, property, value) were dropped.

- Impact: This step ensures that the dataset contains only complete records, improving the reliability of the analysis.

**6.7 Removing Duplicates**

- Original Dataset Issue: The dataset contained duplicate rows, which could skew analysis results.
- Cleaning Process: Duplicate rows were removed using the drop_duplicates() function.
- Impact: Removing duplicates ensures that each record in the dataset is unique, leading to more accurate analysis.

**6.8 Ensuring Consistent Values**

- Original Dataset Issue: Some columns, such as value, still contained NaN or None values after initial cleaning.
- Cleaning Process: Remaining NaN or None values in the value column were replaced with 0 (if appropriate).
- Impact: This step ensures that all columns have consistent and meaningful values, making the dataset ready for analysis.

**7. Changes from the Original Dataset**

After the cleaning process, the dataset has undergone significant improvements:

1. **Timestamp Format**: The timestamp column is now in a proper datetime format, enabling time-based analysis.
2. **New Column**: The day_of_week column has been added, allowing for analysis of user behavior by day of the week.
3. **Numerical Consistency**: The property and value columns now contain only numerical values, ensuring consistency and accuracy in quantitative analysis.
4. **Removed Irrelevant Data**: The status column (if it existed) has been removed, simplifying the dataset.
5. **Missing Values**: Rows with missing values in essential columns have been dropped, ensuring the dataset contains only complete records.
6. **Duplicates Removed**: Duplicate rows have been removed, ensuring each record is unique.
7. **Standardized Values**: The value column has been standardized, with inconsistent formatting removed and missing values handled appropriately.

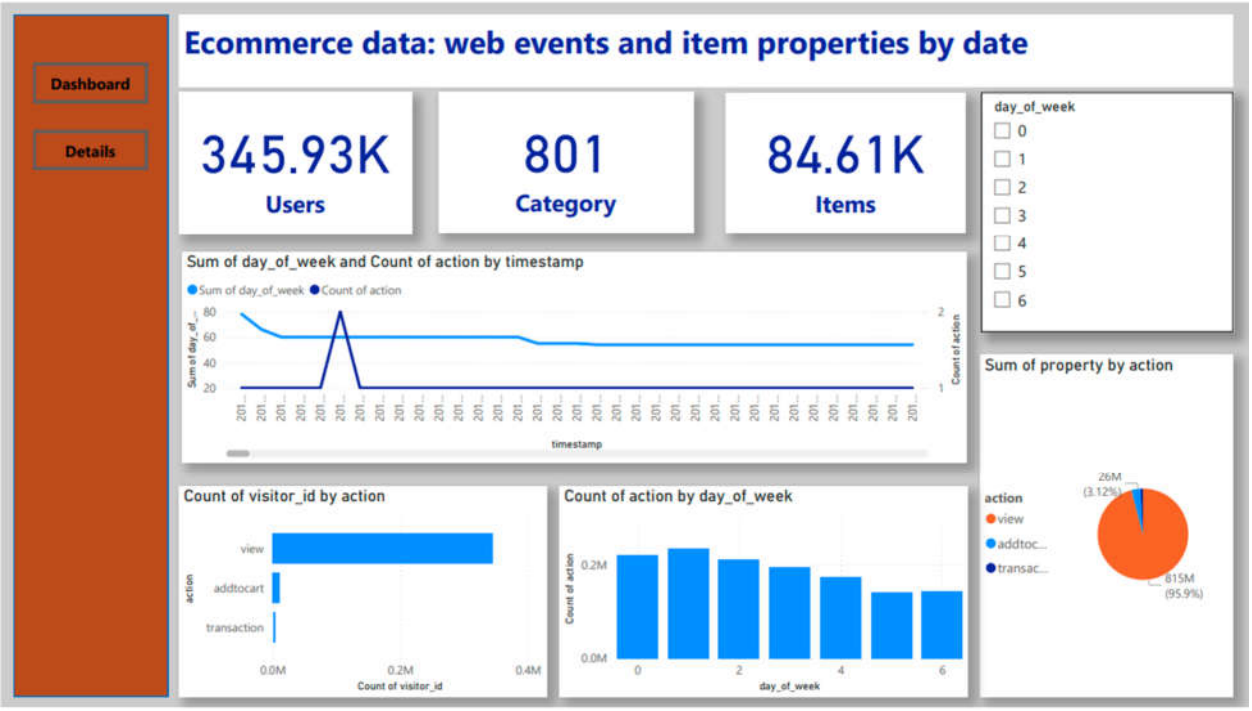
# 7. Visualization and Insights

**Microsoft Power BI** was used to create interactive dashboards and visualizations. Power BI is a powerful business analytics tool that allows users to connect to various data sources, transform

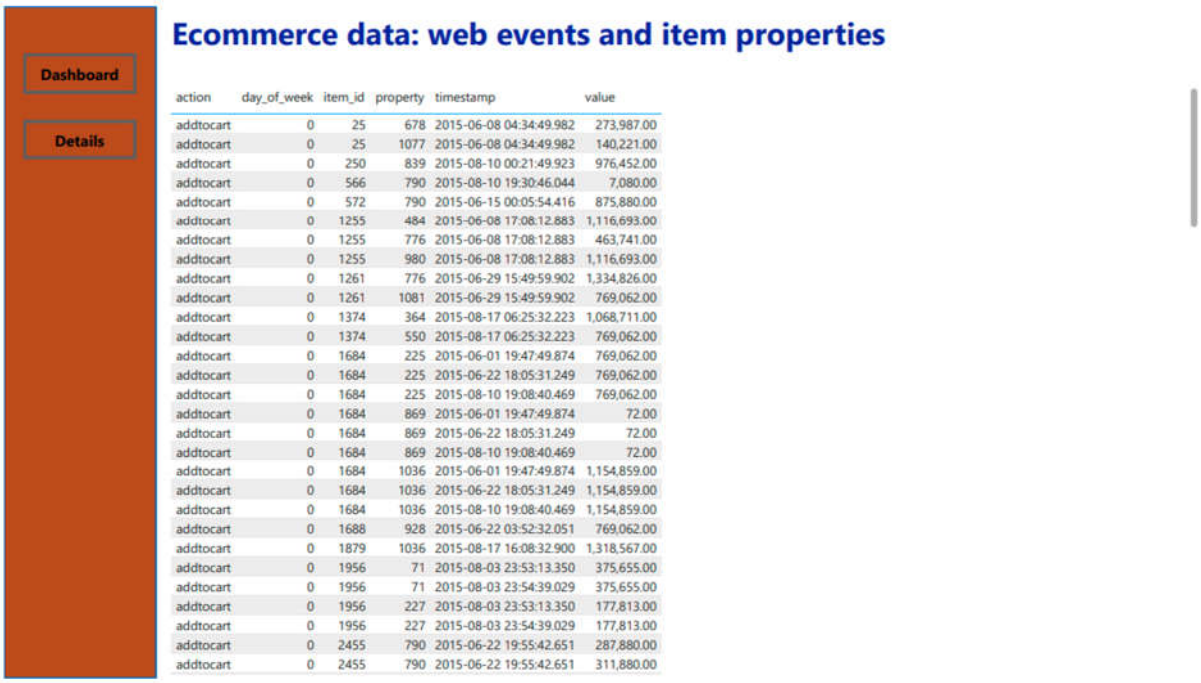
data, and create visually appealing reports and dashboards. The following steps were taken to visualize the data:

1. **Connecting to PostgreSQL**: Power BI was connected to the PostgreSQL database using the PostgreSQL connector.
2. **Data Modeling**: The data was modeled in Power BI to create relationships between tables and define measures for analysis.
3. **Creating Visualizations**: Various visualizations such as bar charts, line charts, and pie charts were created to analyze sales trends, customer segmentation, and product performance.

**Visualization from Dashboard that shows based on a day.**



**Visualization from Details**




| action | day_of_week | item_id | property | timestamp | value |
|---|---|---|---|---|---|
| addtocart | 0 | 25 | 678 | 2015-06-08 04:34:49.982 | 273,987.00 |
| addtocart | 0 | 25 | 1077 | 2015-06-08 04:34:49.982 | 140,221.00 |
| addtocart | 0 | 250 | 839 | 2015-08-10 00:21:49.923 | 976,452.00 |
| addtocart | 0 | 566 | 790 | 2015-08-10 19:30:46.044 | 7,080.00 |
| addtocart | 0 | 572 | 790 | 2015-06-15 00:05:54.416 | 875,880.00 |
| addtocart | 0 | 1255 | 484 | 2015-06-08 17:08:12.883 | 1,116,693.00 |
| addtocart | 0 | 1255 | 776 | 2015-06-08 17:08:12.883 | 463,741.00 |
| addtocart | 0 | 1255 | 980 | 2015-06-08 17:08:12.883 | 1,116,693.00 |
| addtocart | 0 | 1261 | 776 | 2015-06-29 15:49:59.902 | 1,334,826.00 |
| addtocart | 0 | 1261 | 1081 | 2015-06-29 15:49:59.902 | 769,062.00 |
| addtocart | 0 | 1374 | 364 | 2015-08-17 06:25:32.223 | 1,068,711.00 |
| addtocart | 0 | 1374 | 550 | 2015-08-17 06:25:32.223 | 769,062.00 |
| addtocart | 0 | 1684 | 225 | 2015-06-01 19:47:49.874 | 769,062.00 |
| addtocart | 0 | 1684 | 225 | 2015-06-22 18:05:31.249 | 769,062.00 |
| addtocart | 0 | 1684 | 225 | 2015-08-10 19:08:40.469 | 769,062.00 |
| addtocart | 0 | 1684 | 869 | 2015-06-01 19:47:49.874 | 72.00 |
| addtocart | 0 | 1684 | 869 | 2015-06-22 18:05:31.249 | 72.00 |
| addtocart | 0 | 1684 | 869 | 2015-08-10 19:08:40.469 | 72.00 |
| addtocart | 0 | 1684 | 1036 | 2015-06-01 19:47:49.874 | 1,154,859.00 |
| addtocart | 0 | 1684 | 1036 | 2015-06-22 18:05:31.249 | 1,154,859.00 |
| addtocart | 0 | 1684 | 1036 | 2015-08-10 19:08:40.469 | 1,154,859.00 |
| addtocart | 0 | 1688 | 928 | 2015-06-22 03:52:32.051 | 769,062.00 |
| addtocart | 0 | 1879 | 1036 | 2015-08-17 16:08:32.900 | 1,318,567.00 |
| addtocart | 0 | 1956 | 71 | 2015-08-03 23:53:13.350 | 375,655.00 |
| addtocart | 0 | 1956 | 71 | 2015-08-03 23:54:39.029 | 375,655.00 |
| addtocart | 0 | 1956 | 227 | 2015-08-03 23:53:13.350 | 177,813.00 |
| addtocart | 0 | 1956 | 227 | 2015-08-03 23:54:39.029 | 177,813.00 |
| addtocart | 0 | 2455 | 790 | 2015-06-22 19:55:42.651 | 287,880.00 |
| addtocart | 0 | 2455 | 790 | 2015-06-22 19:55:42.651 | 311,880.00 |

### 7.2 Insights

The following insights were derived from the visualizations:

1. Sales Trends Over Time

- ✓ Visualization: A line chart showing sales trends over time.
- ✓ Insight: The dashboard reveals a clear pattern of sales trends over time. There are noticeable peaks and troughs in sales, indicating seasonal trends or promotional periods.

2. Customer Segmentation

- ✓ Visualization: A bar chart or pie chart showing customer segmentation based on purchasing behavior.
- ✓ Insight: The dashboard segments customers into distinct groups based on their purchasing behavior. For example, some customers might be frequent buyers, while others might make one-time purchases.

This segmentation allows businesses to tailor their marketing strategies to different customer groups. For instance, frequent buyers could be targeted with loyalty programs, while one-time buyers might be encouraged to make repeat purchases through discounts or personalized offers.

3. Product Performance

- ✓ Visualization: A bar chart or table showing the top-performing products.
- ✓ Insight: The dashboard highlights the top-performing products in terms of sales or user interactions (e.g., views, add-to-cart actions).

This insight helps businesses identify which products are most popular and which ones might need improvement. For example, if a product has a high number of views but low sales, it might indicate that the product description or pricing needs to be adjusted.

4. User Interaction Analysis

- ✓ Visualization: A stacked bar chart or heatmap showing user interactions (e.g., views, add-to-cart actions, transactions) over time or by product category.
- ✓ Insight: The dashboard provides a breakdown of user interactions, such as product views, add-to-cart actions, and transactions. This helps businesses understand how users are engaging with their products.

For example, if a product has a high number of views but a low add-to-cart rate, it might indicate that the product is attracting interest but failing to convert users into buyers. This could prompt a review of the product's pricing, description, or images.

5. Day-of-Week Analysis

- ✓ Visualization: A bar chart showing user activity or sales by day of the week.
- ✓ Insight: The dashboard reveals patterns in user activity or sales based on the day of the week. For example, there might be higher activity on weekends compared to weekdays.

This insight can help businesses optimize their marketing efforts by targeting users on days when they are most active. For instance, promotional emails or ads could be scheduled for weekends to maximize engagement.

# 8. Conclusion

This project successfully implemented an end-to-end ETL pipeline, from data extraction to visualization. The use of Python, PostgreSQL, and Power BI provided a robust framework for processing and analyzing large datasets. The insights gained from the visualizations can help businesses make data-driven decisions to improve sales and customer satisfaction.

# 9. References


- Retail Rocket e-commerce dataset: Kaggle
- Python Documentation: https://www.python.org/
- PostgreSQL Documentation: https://www.postgresql.org/

- Microsoft Power BI Documentation: https://powerbi.microsoft.com/