IS-107

Semestral Performance Task
Building a Business Intelligence Solution for a Retail Store

Members:
John Carlo E. Ladia
Nimfa Amistoso
Dhemple Mae Naigal
Vincent Lee Fuego
Carl Lexter Buron

I. Project Overview

This project involves building, designing, and implementing Business Intelligence (BI) solution for an online retail store. This project is required to perform end-to-end tasks, from data extraction, transformation, and loading (ETL) to designing a data warehouse, conducting data mining, and developing a visual analytics dashboard that will be deployed as a web application.

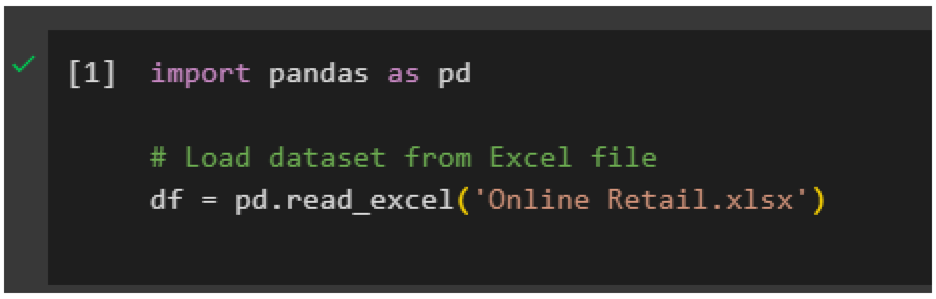
II. Project Deliverables

1. ETL Documentation

This documentation outlines the steps taken to clean, transform, and load the Online Retail dataset using Python and the pandas library. The dataset contains transaction details, product descriptions, customer IDs, sales data, and transaction dates. The goal of this process is to prepare the data for further analysis by addressing missing values, inconsistencies, and outliers.

Extract

The dataset was provided as an Excel file named Online Retail.xlsx. I used the pd.read_excel() function from the pandas library to load the data into a DataFrame.



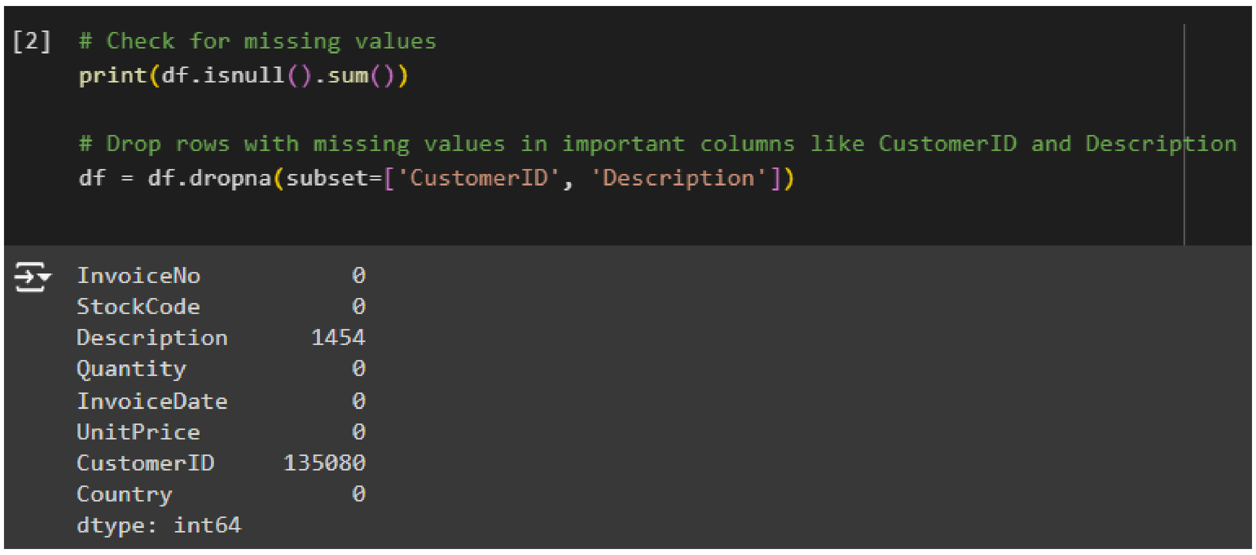
```
[1]  import pandas as pd

     # Load dataset from Excel file
     df = pd.read_excel('Online Retail.xlsx')
```

Transform

a. Handling Missing Values

I chose to drop rows with missing values in these two columns, as they are essential for analysis (customer identification and product descriptions).



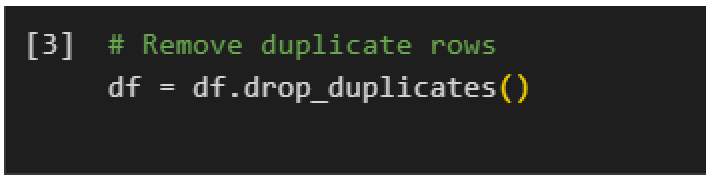
```
[2] # Check for missing values
    print(df.isnull().sum())

    # Drop rows with missing values in important columns like CustomerID and Description
    df = df.dropna(subset=['CustomerID', 'Description'])
```

```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

b. Removing Duplicates

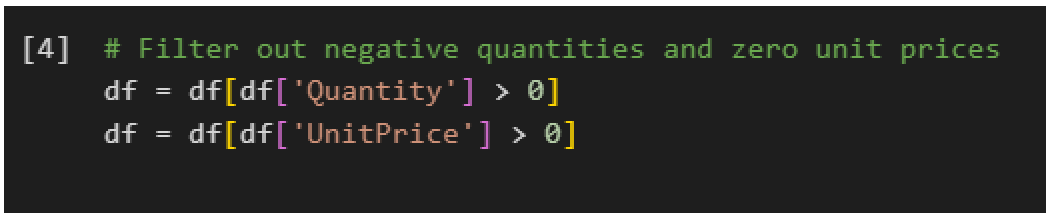
Duplicate records in transactional data can distort analysis. I removed any duplicate rows to maintain the integrity of the dataset.



```
[3] # Remove duplicate rows
    df = df.drop_duplicates()
```

c. Handling Outliers

Negative or zero values in Quantity and UnitPrice fields are likely due to refunds or data entry errors. I filtered out transactions with negative or zero quantities and unit prices, as they do not represent valid sales transactions.

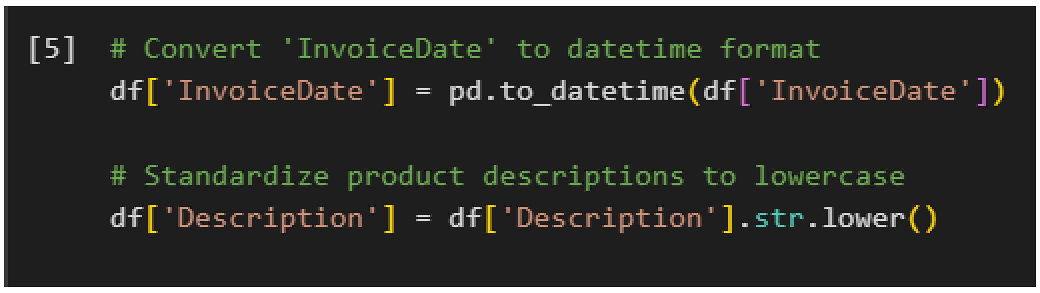


```
[4] # Filter out negative quantities and zero unit prices
    df = df[df['Quantity'] > 0]
    df = df[df['UnitPrice'] > 0]
```

d. Standardizing Data

Date Conversion: The InvoiceDate column was converted to a datetime format to allow for time-based analysis.

Text Standardization: Product descriptions were standardized by converting all text to lowercase.

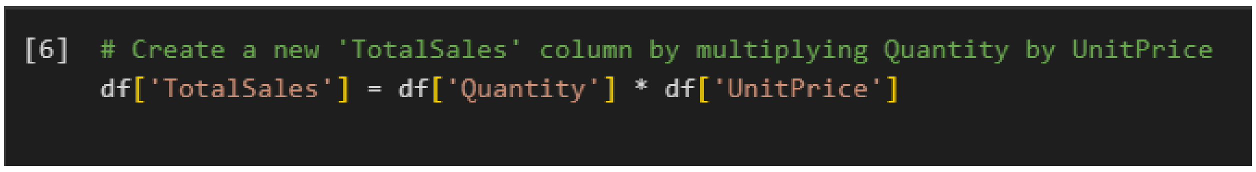


```
[5] # Convert 'InvoiceDate' to datetime format
    df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])

    # Standardize product descriptions to lowercase
    df['Description'] = df['Description'].str.lower()
```

e. Creating New Features

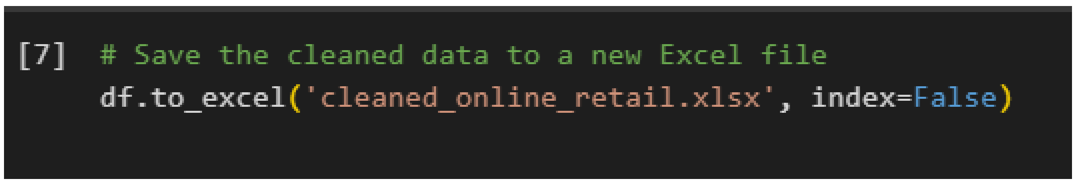
To aid in future analysis, I created a new column called TotalSales, which represents the total sales value for each transaction by multiplying Quantity by UnitPrice.



```
[6]  # Create a new 'TotalSales' column by multiplying Quantity by UnitPrice
     df['TotalSales'] = df['Quantity'] * df['UnitPrice']
```

Load

After transforming the data, the final step was to save the cleaned dataset to a new Excel file for future use.



```
[7]  # Save the cleaned data to a new Excel file
     df.to_excel('cleaned_online_retail.xlsx', index=False)
```

2. DATA WAREHOUSING

This documentation outlines the steps takin to create tables using star schema design that stores the cleaned data. The schema includes fact table, and dimension tables for products, customers, and date-time and steps takin to loading cleaned data into the data warehouse.

SQL Script

- **CREATING TABLES**

```
-- Sales Fact Table
CREATE TABLE sales_fact (
   sales_id SERIAL PRIMARY KEY,
   invoice_no INT,
   quantity INT,
   unit_price NUMERIC,
   total_sales NUMERIC,
   productDIM_id SERIAL REFERENCES product_dimension(productDIM_id),
   customerDim_id SERIAL REFERENCES customer_dimension(customerDim_id),
   dateDim_id SERIAL REFERENCES date_dimension(dateDim_id)
);

-- Product Dimension Table
CREATE TABLE product_dimension (
   productDim_id SERIAL PRIMARY KEY,
   stock_code VARCHAR(20),
   description VARCHAR(255)
);

-- Customer Dimension Table
CREATE TABLE customer_dimension (
   customerDim_id SERIAL PRIMARY KEY,
   customer_id INT,
   country VARCHAR(100)
);

-- Date Dimension Table
CREATE TABLE date_dimension (
   dateDim_id SERIAL PRIMARY KEY,
   invoice_date TIMESTAMP WITH TIMEZONE
);
```

- **LOADING DATA INTO DATA WAREHOUSING**

Table sales fact:

Step 1: Create a .csv file that stores the data of invoice_no, quantity, unit_price, and total_sales from the cleaned dataset.

Step 2: Create a temporary table to store data

```
CREATE TABLE stage_sales (
   invoice_no INT,
   quantity INT,
   unit_price NUMERIC,
   total_sales NUMERIC
);
```

Step 3: Using PSQL tool (pgAdmin terminal) copy the csv data into stage_sales table

```
\copy sales_fact(invoice_no, quantity, unit_price, total_sales) FROM
'C:\Users\ACER\Dropbox\PC\Documents\4th year nako bruh\IS 107\csv\SALES_FACT.csv' WITH
CSV HEADER;
```

Step 4: Insert the data from stage_sales table to sales_fact table

```
INSERT INTO stage_sales(invoice_no, quantity, unit_price, total_sales)
SELECT
        invoice_no,
        quantity,
        unit_price,
        total_sales
FROM stage_sales
```

Step 5: Drop the temporary table, stage_sales

```
DROP TABLE stage_sales;
```

Table product_dimension:

Step 1: Create a .csv file that stores the data of stock_code and description from the cleaned dataset.

Step 2: Create a temporary table to store data

```
CREATE TABLE stage_product(
        stock_code VARCHAR(20),
        description VARCHAR(200)
);
```

Step 3: Using PSQL tool (pgAdmin terminal) copy the csv data into stage_product table

```
\copy stage_product(stock_code, description) FROM
'C:\Users\ACER\Dropbox\PC\Documents\4th year nako bruh\IS 107\StockCode_Description.csv'
WITH CSV HEADER;
```

Step 4: Insert the data from stage_product table to product_dimension table

```
INSERT INTO product_dimension(stock_code, description)
SELECT
        stock_code,
        description
FROM stage_product;
```

Step 5: Drop the temporary table, stage_product

```
DROP TABLE stage_product;
```

Table customer_dimension:

Step 1: Create a .csv file that stores the data of customer_id and country from the cleaned dataset.

Step 2: Create a temporary table to store data

```
CREATE TABLE stage_customer(
        customer_id INT,
        country VARCHAR(100),
);
```

Step 3: Using PSQL tool (pgAdmin terminal) copy the csv data into stage_customer table

```
\copy stage_customer(customer_id, country) FROM 'C:\Users\ACER\Dropbox\PC\Documents\4th
year nako bruh\IS 107\csv\Customer.csv' WITH CSV HEADER;
```

Step 4: Insert the data from stage_customer table to customer_dimension table

```
INSERT INTO customer_dimension(customer_id, country)
SELECT
        customer_id,
```

country
FROM stage_customer;

Step 5: Drop the temporary table, stage_customer

DROP TABLE stage_customer;

Table date_dimension:

Step 1: Create a .csv file that stores the data of invoice_date from the cleaned dataset.

Step 2: Create a temporary table to store data

```
CREATE TABLE stage_date(
        invoice_date TIMESTAMP WITH TIME ZONE
);
```

Step 3: Using PSQL tool (pgAdmin terminal) copy the csv data into stage_date table

```
\copy stage_date(invoice_date) FROM 'C:\Users\ACER\Dropbox\PC\Documents\4th year nako bruh\IS 107\csv\InvoiceDate.csv' WITH CSV HEADER;
```

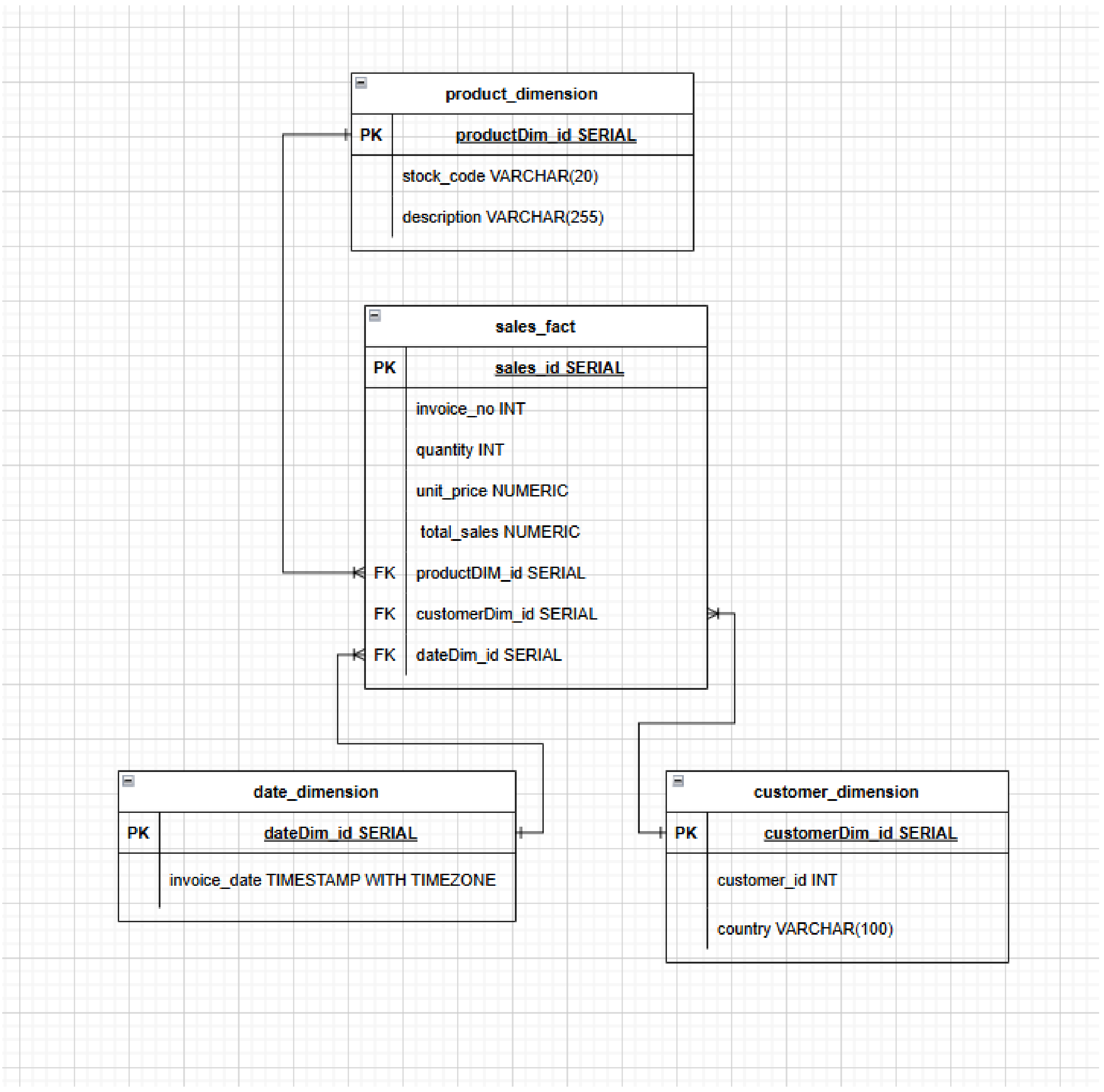
Step 4: Insert the data from stage_date table to date_dimension table

```
INSERT INTO date_dimension(invoice_date)
SELECT
        invoice_date
FROM stage_date;
```

Step 5: Drop the temporary table, stage_date

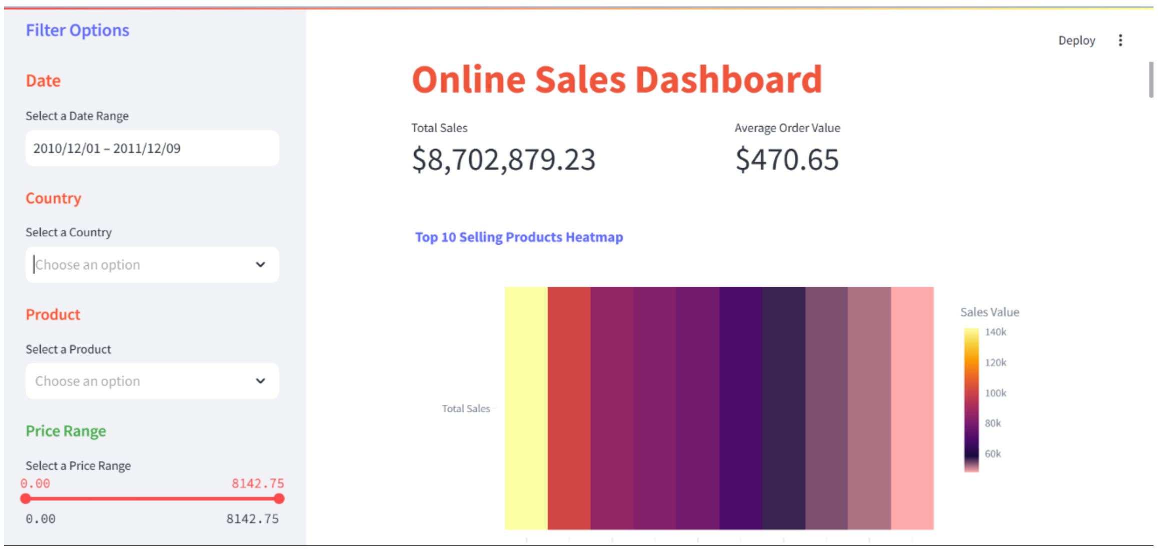
```
DROP TABLE stage_date;
```

- **ENTITY RELATIONSHIP DIAGRAM**



3. DATA VISUALIZATION

- **DASHBOARD**



Filter Options

Date

Select a Date Range: 2010/12/01 – 2011/12/09

Country

Select a Country: Choose an option

Product

Select a Product: Choose an option

Price Range

Select a Price Range: 0.00 – 8142.75

Online Sales Dashboard

Total Sales: $8,702,879.23

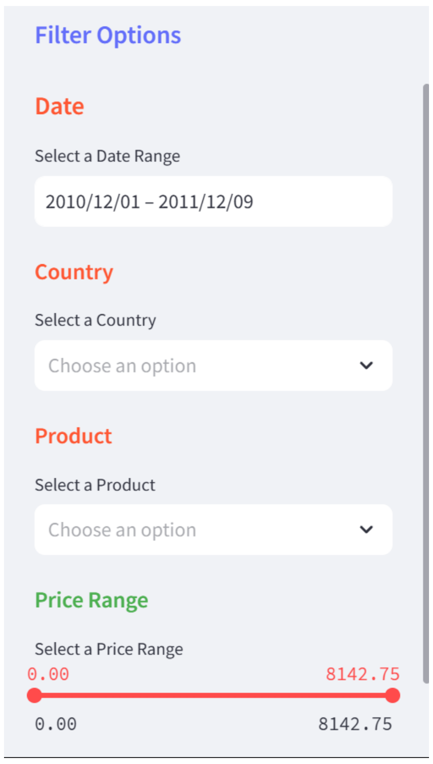
Average Order Value: $470.65

This is the overview of the online retail store dashboard

- **FILTER OPTIONS**

The **Filter Options** allows the user to manipulate the data by changing the date, country, product, or price range. The data shown in Total Sales, Average Order Value, Monthly Sales Growth Rate, Monthly Sales Trend, and Sales Forecasting changes according to the selected filter options.

The filter option aids in decision making by filtering out irrelevant data, analyze trends over specific periods, and assesses performance in specific regions.



Filter Options

Date

Select a Date Range

2010/12/01 – 2011/12/09

Country

Select a Country

Choose an option

Product

Select a Product

Choose an option

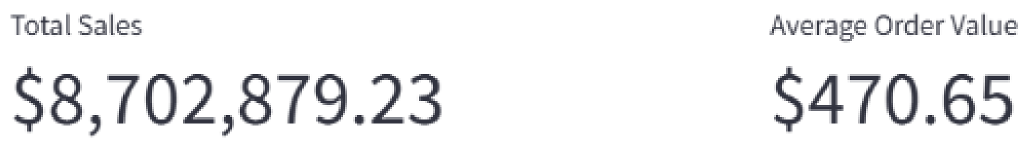
Price Range

Select a Price Range

0.00 8142.75

0.00 8142.75

- **TOTAL SALES & AVG ORDER VALUE**



Total Sales

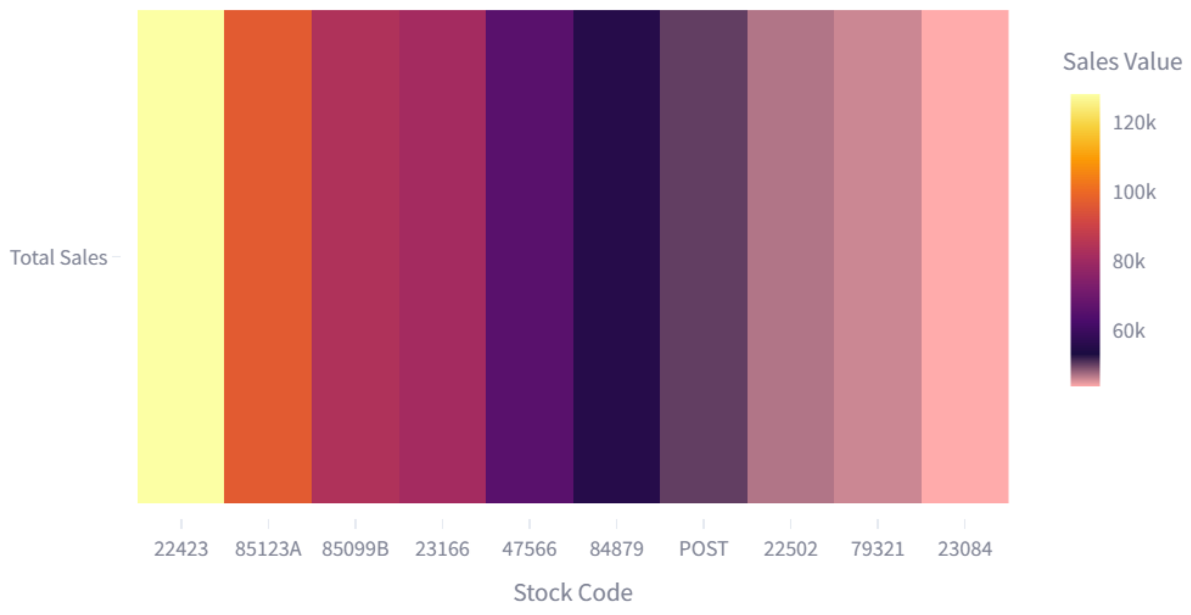
$8,702,879.23

Average Order Value

$470.65

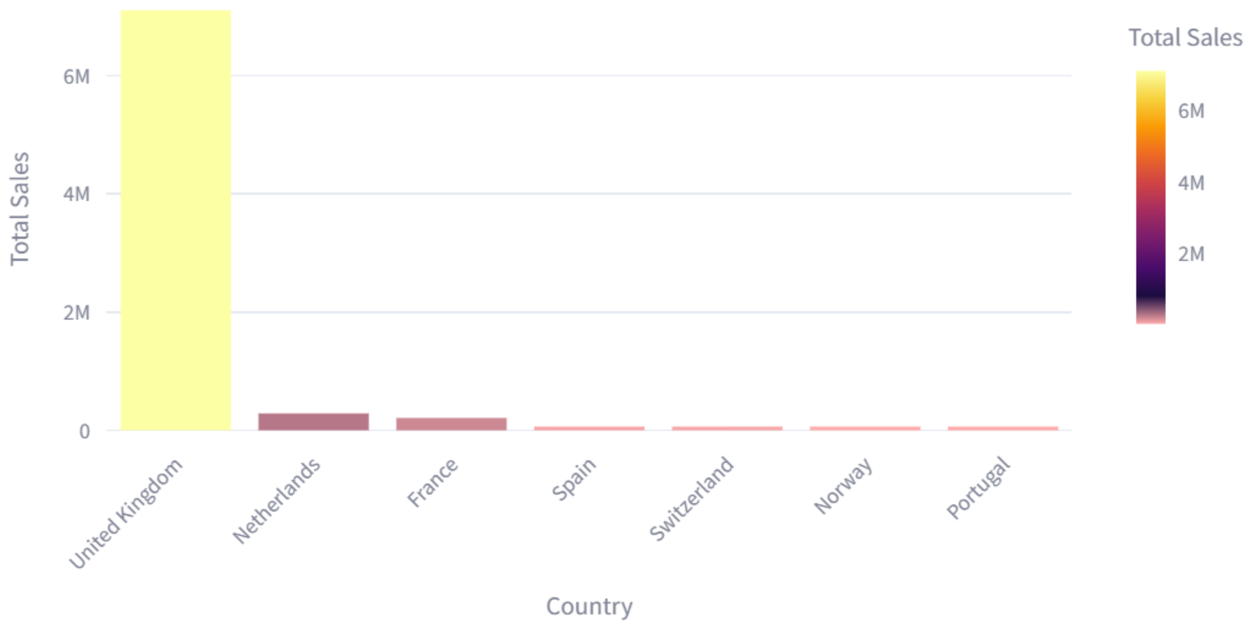
The data shown in **Total Sales** represents the total revenue of a country or countries within a specific period, based on the selected country in filter options. The **Average Order Value** measures the average amount spent by a customer per transaction. The total sales and average order value aids in decision making by tracking the overall business performance and determines whether the goals are met.

- **TOP 10 SELLING PRODUCTS**



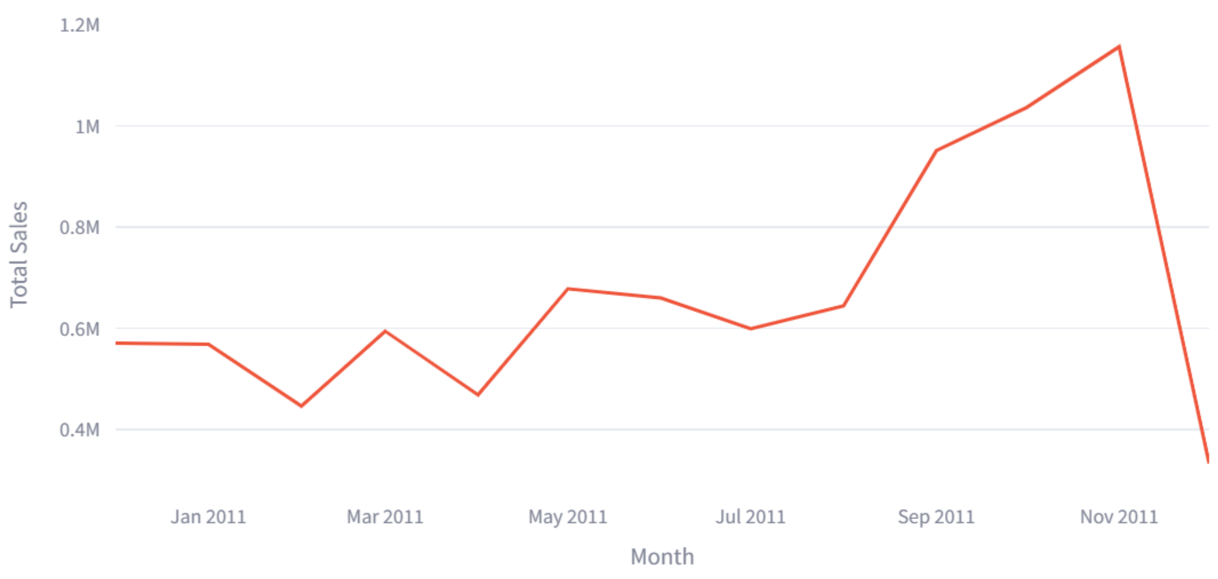
This heatmap shows the **top 10 selling products** along with their sales values. Knowing the top 10 selling products aids in decision making by ensuring popular products are always in stock to meet customer demands. Additionally, it helps in creating advertising and marketing campaigns to further boost the products' popularity to attract more customer.

- **TOP SALES BY COUNTRY**



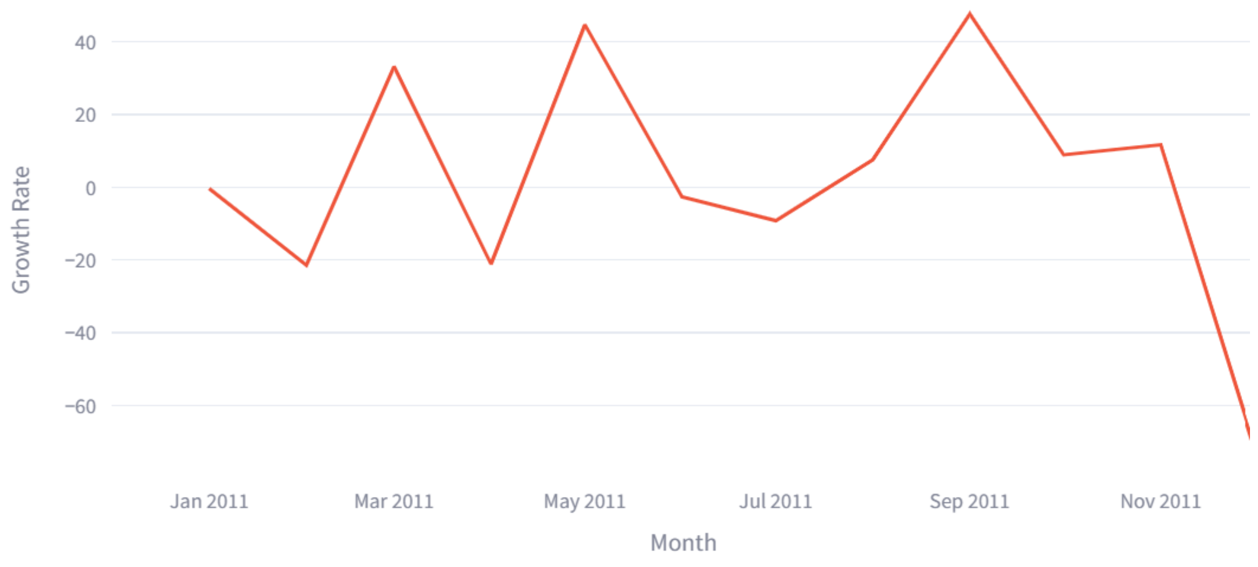
This graph shows the **Top sales by country** a performance metric that identifies the country generating the highest revenue over a specific period**.** The top sales by country aids in decision making by knowing which countries are the most profitable, helping allocate resources, inventory, and marketing efforts effectively.

- **MONTHLY SALES TREND**



This line chart shows the **Monthly sales trend** and how the sales revenue changes from month to month. The monthly sales trend aids in decision making by identifying trends and patterns, performance evaluation, problem detection, and strategic planning.

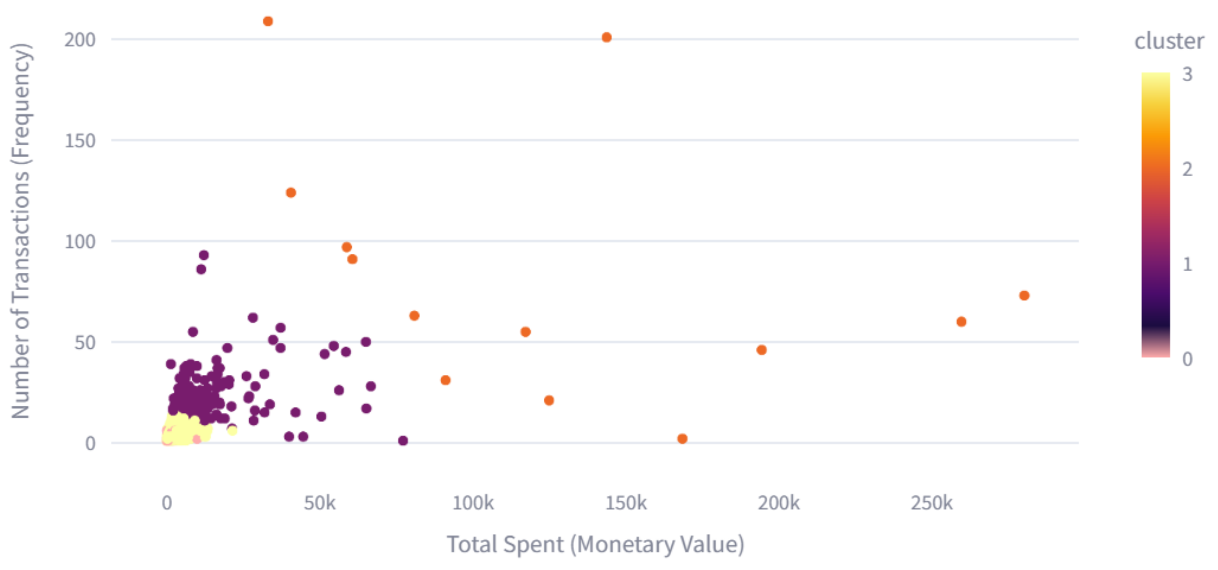
- MONTHLY SALES GROWTH RATE



This chart shows the **Monthly sales growth rate,** it measures the percentage change in sales revenue from one month to next. The monthly sales growth rate aids in decision making by tracking month to month changes in sales, helping in evaluating growth, stability, or decline in sales.

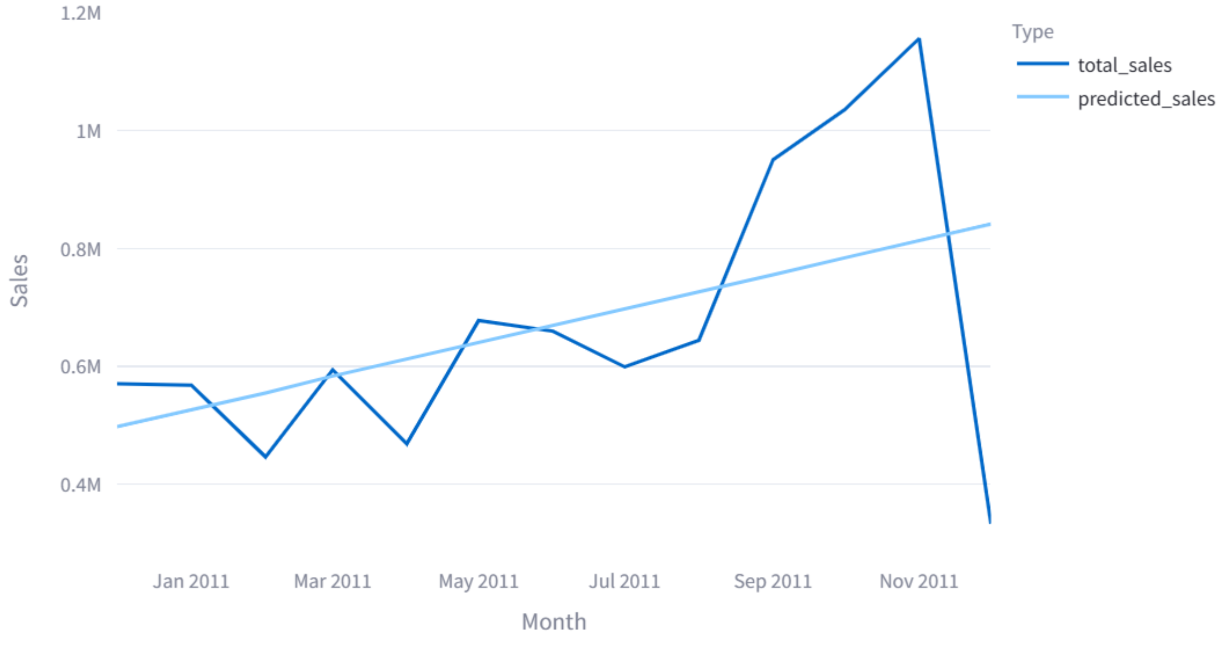
- **CUSTOMER SEGMENTATION**

Frequency vs. Monetary Value



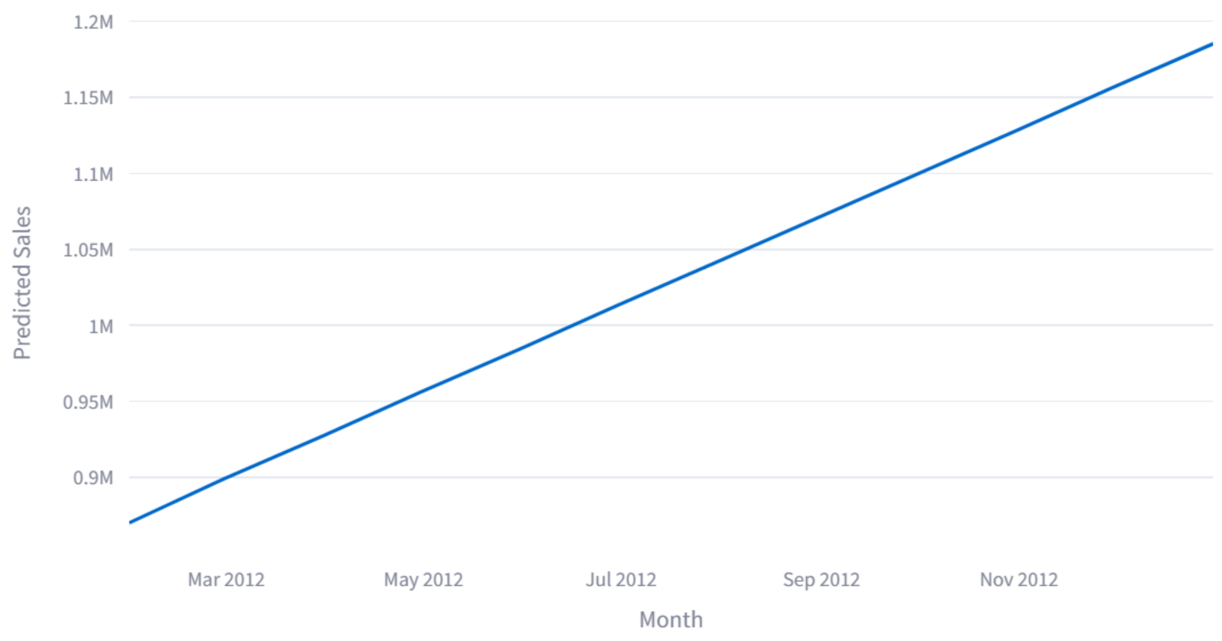
The chart uses a scatter plot and K-means clustering to visualize the number of transactions made by each customer and their total spending on purchases. **Customer segmentation** based on the customers number of transaction and total spending aids in decision making by providing a clear view of customer behavior and purchasing patterns personalizing marketing campaigns offering discounts to high spenders, send promotions or incentives to encourage low spenders to spend more.

- **SALES FORECASTING**



The Linear Regression is applied here as data mining technique for predictive modelling of **Sales Forecasting**. The chart shows how sales started low in its first month of the year but gradually grew over time. The sales forecasting aids in decision making by providing actionable actions into future performance, sales forecasting enables businesses to make data driven decisions, reduce uncertainty, and stay competitive in the market.

- **FUTURE SALES PREDICTION**



The **Future Sales Prediction** chart also uses Linear Regression as the data mining technique for predictive modelling to forecast sales for the next 12 months. The chart shows how the predicted sales grew over time. The future sales prediction aids in decision making in setting long term goals and expand into new markets or product lines.

4. User Guide for the Web Application

1. Start with the Filter options, play with it a little and familiarize yourself with the interface.
2. To see the data of each chart based on specific date, go to the **Filter Options,** and select your preferred date in the Date filter option. The interface should automatically display all the new information.
3. To see the data of each chart and graphs based on country, go to the **Filter Options,** inside the filter options you can select a country by clicking the rectangular shape under the label **Country**. The interface should automatically display all new information.
4. If you want to see the data of each chart based on product, go to the Filter Options, inside the filter options you will find the **Product** label and select a product by clicking the rectangular shape. The interface should automatically display all new information.
5. If you want to see the data of each chart based on price range of a product, go to the Filter Options, inside the filter options you will find the **Price Range** label and drag the slider to your preferred price range. The interface should automatically display all new information.

5. Challenges Faced

Missing Values: The dataset contained a significant number of missing CustomerID values. Instead of imputing these values, I decided to remove the affected rows, considering the importance of customer identification in analyzing sales patterns.

Handling Outliers: Several transactions had negative or zero Quantity values, likely representing returns or incorrect entries. These were filtered out, as they would skew revenue calculations.

Loading data into the data warehouse: Using PSQL tool you cannot insert the cleaned data directly into the tables. So we decided to create temporary tables to store the cleaned data and then transfer it into the main tables.

Customer segmentation: We struggled a bit in choosing the metrics to use for customer segmentation but ultimately found the right one.