Leader: Carlo S. Gaballo

Member: Le Anne L. Durango

DOCUMENTATION

I. Extract, Transform, and Load

ETL stands for Extract, Transform, and Load. It is a process used in data integration to retrieve data from a source, clean and prepare it, and store it in a destination system for analysis and reporting.

The ETL process flow of this project is to prepare online retail transactional data for analysis by extracting the data from an Excel file, transforming it through cleaning and data enrichment, and loading the cleansed data into a new CSV file. So, first is to

Extract

The extraction phase retrieves raw data from the provided dataset and loads it into memory.

Steps:

1. Locate the Source File:
    - The source dataset is stored in an Excel file named Online_Retail.xlsx.
2. Load the Data into Memory:
    - The data is read into a pandas DataFrame using the following command:

```
import pandas as pd

# Load data from the original dataset
file_path = 'Online_Retail.xlsx'  # Update this with the correct file path
data = pd.read_excel(file_path)
```

    - This step enables the data to be accessible for further processing.

Outcome:

At the end of this phase, the raw data is successfully extracted and prepared for the transformation phase.

Transform

The transformation phase involves cleaning and enriching the data to make it usable for analysis. This ensures the dataset is free of inconsistencies and missing values.

Steps:

1. Link Transactions to Customers:
    - Rows without a valid CustomerID are removed, as they cannot be linked to a customer.

```
# Step 1: Drop rows with missing CustomerID
data = data.dropna(subset=['CustomerID'])
```

2. Handle Missing Product Descriptions:
    - Replace missing values in the Description column with "No Description."

```
# Step 2: Fill missing Description with 'No Description'
data['Description'] = data['Description'].fillna('No Description')
```

3. Eliminate Invalid Transactions:
    - Remove rows where the Quantity is less than or equal to zero.

```
# Step 3: Remove rows with non-positive Quantity values
data = data[data['Quantity'] > 0]
```

4. Standardize Dates:
    - Convert the InvoiceDate column into a uniform YYYY-MM-DD format for consistency in analysis.

```
# Step 4: Convert InvoiceDate to date format (YYYY-MM-DD)
data['InvoiceDate'] = pd.to_datetime(data['InvoiceDate']).dt.strftime('%Y-%m-%d')
```

5. Filter Out Extreme Values:
    - Outliers in Quantity and UnitPrice are filtered to retain only the data within the 1st and 99th percentiles.

```
# Step 5: Apply outlier filtering
quantity_thresholds = data['Quantity'].quantile([0.01, 0.99])
unitprice_thresholds = data['UnitPrice'].quantile([0.01, 0.99])

data = data[
    (data['Quantity'].between(quantity_thresholds.iloc[0], quantity_thresholds.iloc[1])) &
    (data['UnitPrice'].between(unitprice_thresholds.iloc[0], unitprice_thresholds.iloc[1]))
]
```

6. Add a Total Price Column:
    - Create a new column TotalPrice by multiplying Quantity and UnitPrice for each transaction.

```
# Step 6: Add TotalPrice column
data['TotalPrice'] = data['Quantity'] * data['UnitPrice']
```

Outcome:

The raw data is now cleaned, enriched, and structured in a way that makes it suitable for analysis and ready for the loading phase.

Load

The loading phase saves the transformed data into a new file, ensuring it is ready for analytical processes.

Steps:

1. Define the Output Destination:
    - Specify the name and format of the output file. In this case, the transformed data will be saved as a CSV file named Cleaned_Online_Retail_Corrected.csv.
2. Save the Data:
    - Use the pandas library to write the DataFrame to a CSV file:

```
# Save the cleaned data to a CSV file
output_file = 'Cleaned_Online_Retail_Corrected.csv'
data.to_csv(output_file, index=False)

print(f"Cleaned data saved to: {output_file}")
```

Outcome:

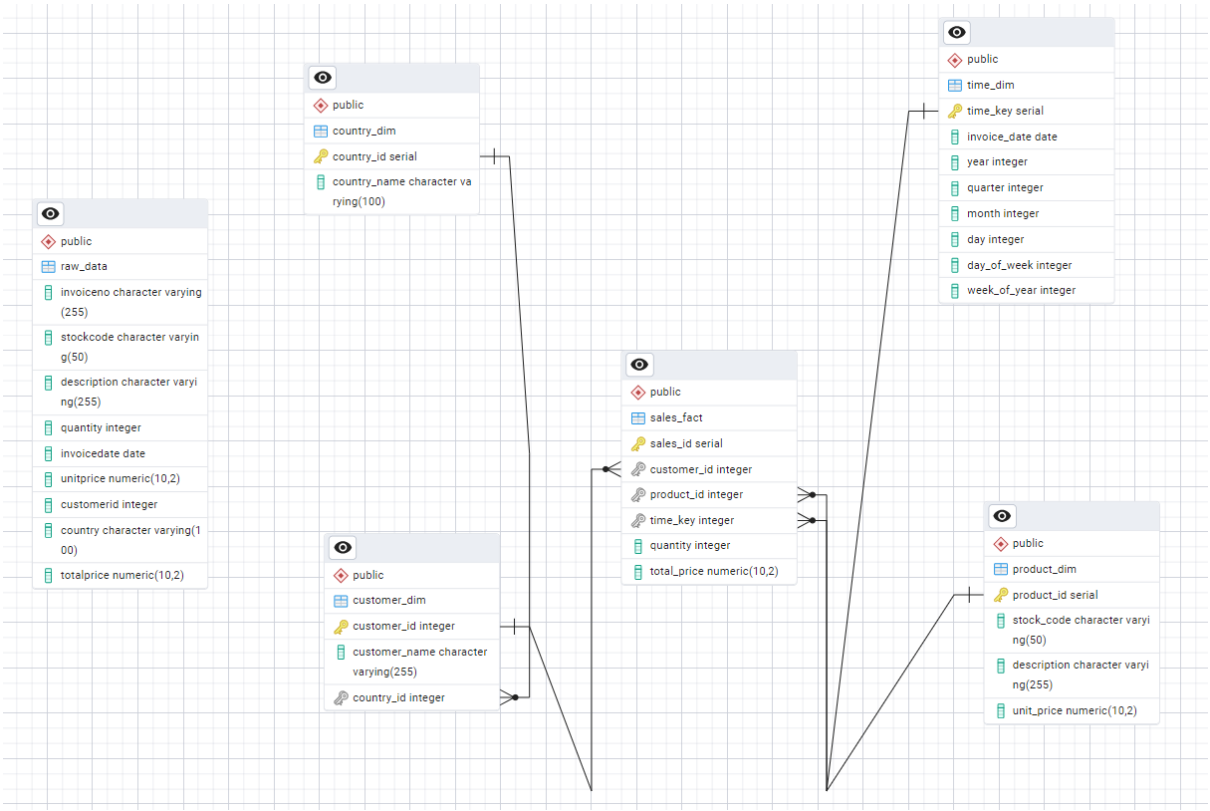
The cleaned and transformed data is successfully stored in a CSV file, ready for use in subsequent analytical tasks or reporting.

II. Data Warehousing

Data warehousing is the design and implementation of a centralized repository for aggregating, transforming, and analyzing data from multiple sources. This repository enables efficient query performance for analytical purposes and supports business intelligence initiatives. The document details the SQL scripts and the Entity-Relationship Diagram (ERD) used to design a star schema-based data warehouse. The star schema centralizes a fact table surrounded by dimension tables, optimizing performance for analytical queries. The data that has undergone the Extract, Transform, and Load (ETL) process is stored and managed using PGAdmin. The project employs the star schema design to ensure scalability and efficiency.

Star Schema

The star schema is a simple yet powerful data warehouse design that organizes data into a fact table and dimension tables. The Raw Table Holds the original unprocessed data from the ETL process. This table has no direct connection to the star schema but serves as the source for data extraction and transformation. Next, the fact table stores measurable, quantitative data, such as sales transactions, while the dimension tables provide descriptive context, such as product details or customer information.



SQL scripts for data warehousing are used to create and manage the structure of the data warehouse, load data, and perform transformations and queries. Below are the SQL scripts utilized in this project.

The first script is to set Up the Raw Data Table and Connect to your PostgreSQL database and create a table to hold the raw data:

```
CREATE TABLE raw_data (
  InvoiceNo VARCHAR(50),
  StockCode VARCHAR(50),
```

```
  Description VARCHAR(255),
  Quantity INT,
  InvoiceDate DATE,
  UnitPrice DECIMAL(10, 2),
  CustomerID INT,
  Country VARCHAR(100),
  TotalPrice DECIMAL(10, 2)
);
```

Then, Import the CSV File into PostgreSQL by transferring the CSV file to your server or local machine where PostgreSQL is running

-Using psql:

\copy raw_data FROM '/path/to/Cleaned_Online_Retail_Corrected.csv' WITH CSV HEADER;

-Using pgAdmin graphical import option:

Steps:

1. *In pgAdmin, right-click on the table where you want to import the data.*
2. *Select Import/Export from the context menu.*
3. *In the dialog box:*
    - *Choose Import as the action.*
    - *Select your CSV file in the Filename field.*
    - *Ensure the Delimiter matches your CSV format (e.g., , for commas).*
    - *Check Header if your CSV includes column headers.*
    - *Click OK to import the file.*

Next is Create Dimension and Fact Tables:

```
-- Time Dimension Table
CREATE TABLE time_dim (
  time_key SERIAL PRIMARY KEY,
  invoice_date DATE NOT NULL,
```

```
    year INT NOT NULL,
    quarter INT NOT NULL,
    month INT NOT NULL,
    day INT NOT NULL,
    day_of_week INT NOT NULL,
    week_of_year INT NOT NULL
);

-- Customer Dimension Table
CREATE TABLE customer_dim (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255),
    country_id INT,
    FOREIGN KEY (country_id) REFERENCES country_dim(country_id)
);

-- Product Dimension Table
CREATE TABLE product_dim (
    product_id SERIAL PRIMARY KEY,
    stock_code VARCHAR(50),
    description VARCHAR(255),
    unit_price DECIMAL(10, 2)
);

-- Country Dimension Table
CREATE TABLE country_dim (
    country_id SERIAL PRIMARY KEY,
    country_name VARCHAR(100) NOT NULL
);
```

-- Sales Fact Table

```
CREATE TABLE sales_fact (
  sales_id SERIAL PRIMARY KEY,
  customer_id INT,
  product_id INT,
  time_key INT,
  quantity INT,
  total_price DECIMAL(10, 2),
  FOREIGN KEY (customer_id) REFERENCES customer_dim(customer_id),
  FOREIGN KEY (product_id) REFERENCES product_dim(product_id),
  FOREIGN KEY (time_key) REFERENCES time_dim(time_key)
);
```

After creating the tables, the next step involves writing and executing queries to populate the dimension tables and fact table with relevant data. These queries are designed to extract data from the source, transform it as needed, and insert it into the respective dimension tables and fact table, ensuring the data warehouse is prepared for efficient analysis.

```
INSERT INTO time_dim (invoice_date, year, quarter, month, day, day_of_week, week_of_year)
SELECT DISTINCT
  InvoiceDate,
  EXTRACT(YEAR FROM InvoiceDate),
  EXTRACT(QUARTER FROM InvoiceDate),
  EXTRACT(MONTH FROM InvoiceDate),
  EXTRACT(DAY FROM InvoiceDate),
  EXTRACT(DOW FROM InvoiceDate),
  EXTRACT(WEEK FROM InvoiceDate)
FROM raw_data;

INSERT INTO country_dim (country_name)
SELECT DISTINCT Country FROM raw_data WHERE Country IS NOT NULL;
```

```
INSERT INTO customer_dim (customer_id, country_id)
SELECT DISTINCT
  CustomerID,
  (SELECT country_id FROM country_dim WHERE country_name = raw_data.Country LIMIT 1)
FROM raw_data
WHERE CustomerID IS NOT NULL
ON CONFLICT (customer_id) DO NOTHING;

INSERT INTO product_dim (stock_code, description, unit_price)
SELECT DISTINCT StockCode, Description, UnitPrice
FROM raw_data;
```

Then, to obtain the foreign keys for sales_fact, you must join it with the relevant dimension tables and insert it:

```
INSERT INTO sales_fact (customer_id, product_id, time_key, quantity, total_price)
SELECT
  cd.customer_id,
  pd.product_id,
  td.time_key,
  rd.Quantity,
  rd.TotalPrice
FROM raw_data rd
INNER JOIN customer_dim cd ON rd.CustomerID = cd.customer_id
INNER JOIN product_dim pd ON rd.StockCode = pd.stock_code
INNER JOIN time_dim td ON rd.InvoiceDate = td.invoice_date;
```

The querying the data warehouse for analysis can start once the tables are set up,

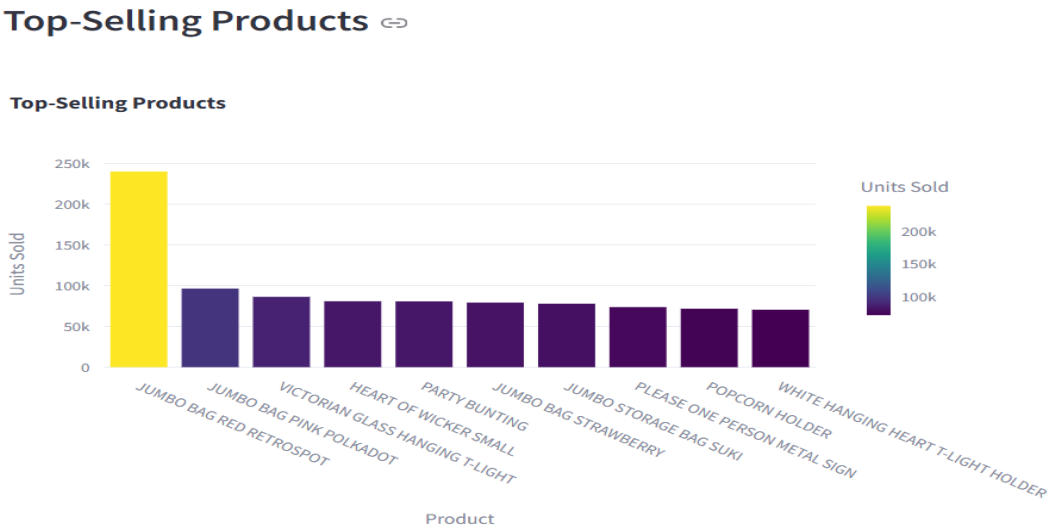
Example of the query for analysis is the query for getting the total sales by country:

```
SELECT c.country_name, SUM(sf.total_price) AS total_sales
FROM sales_fact sf
JOIN customer_dim cd ON sf.customer_id = cd.customer_id
JOIN country_dim c ON cd.country_id = c.country_id
GROUP BY c.country_name
ORDER BY total_sales DESC;
```

III. Data Visualization and Data Mining

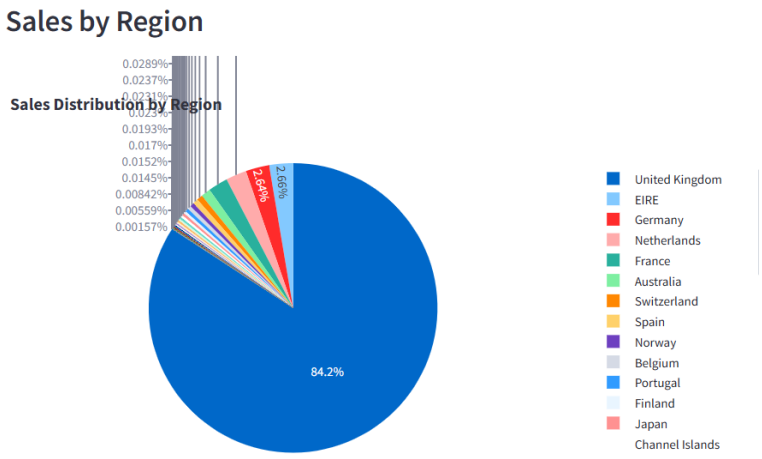
This project integrates data visualization and data mining techniques to provide actionable insights into sales and customer behaviors. **Data visualization** transforms raw data into meaningful visual representations, enabling users to identify trends, patterns, and anomalies quickly. It helps stakeholders make informed decisions through charts and graphs such as bar charts, pie charts, and line graphs. **Data mining**, on the other hand, applies statistical and machine learning techniques to discover hidden patterns and relationships in data. In this project, data mining includes customer segmentation using clustering algorithms and sales forecasting through predictive modeling. Together, these components create a comprehensive business intelligence solution that aids in monitoring performance, planning strategies, and improving customer targeting.

Base on the data that extract, transformed, load and stored on the database the visualization of the analysis data is the following:



This bar chart highlights the top 10 products with the highest units sold. Each bar represents a product, with its height corresponding to the number of units sold. The vibrant colors make it easy

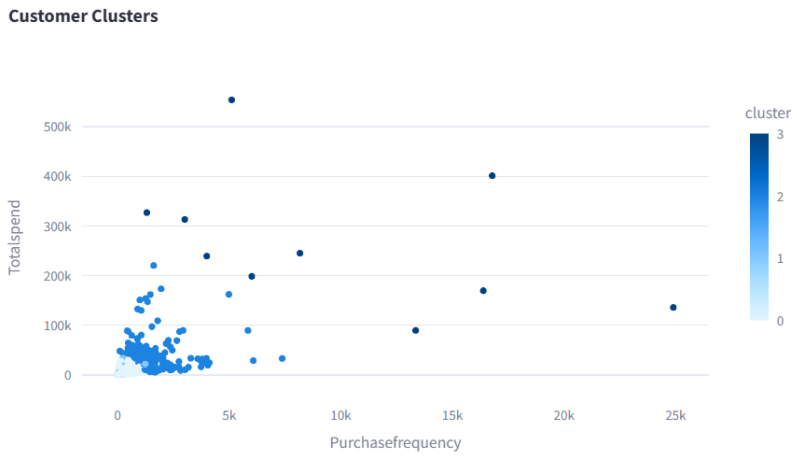
to differentiate between products. This visualization provides insight into which items are driving the most revenue, aiding businesses in identifying key products to prioritize in inventory and marketing strategies. For example, the chart clearly shows a dominant product outperforming others, suggesting its role in driving overall sales. By hovering over each bar, users can view precise product details and their sales metrics.



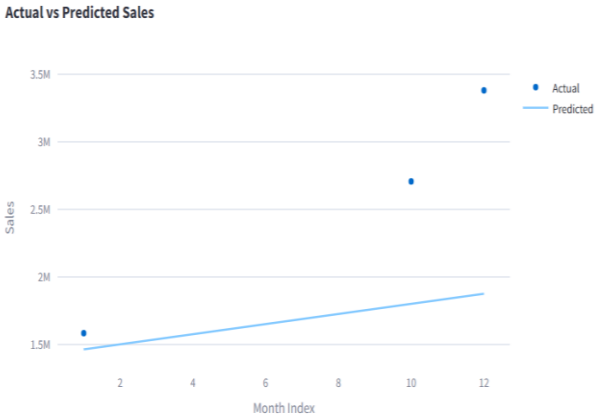
The pie chart illustrates the sales distribution across various regions, with each slice representing the proportion of sales from a specific country. The chart uses interactive tooltips to display region names and their respective sales percentages. This visualization is crucial for identifying high-performing markets and regions that may require additional focus. For example, the chart may reveal that the United Kingdom contributes over 80% of total sales, emphasizing its importance as the primary market. Smaller slices for other regions highlight potential areas for expansion or targeted marketing efforts.

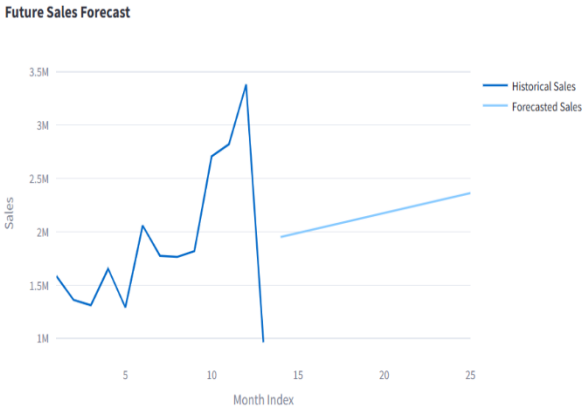
This line graph tracks the total sales value over months, providing a clear view of seasonal trends and anomalies. Each point on the line represents the total sales for a particular month. For instance, the chart might show a noticeable spike in sales during November, likely due to seasonal shopping events. Conversely, dips in certain months may indicate periods of lower

consumer activity. By analyzing the ups and downs of the sales trends, businesses can prepare for peak periods and mitigate dips in performance. This chart is particularly useful for long-term strategic planning and understanding the impact of external factors on sales.



The scatter plot visualizes customer clusters based on K-Means clustering, using features such as total spend, purchase frequency, and recency. Each point represents a customer, and colors distinguish the different clusters. For instance, the chart shows distinct clusters of high-value customers who frequently purchase and spend more, contrasted with occasional buyers with low spend. This segmentation helps businesses identify opportunities for targeted campaigns. For example, high-spending clusters can be targeted with loyalty programs, while lower-spending groups might benefit from promotional offers to increase their activity.



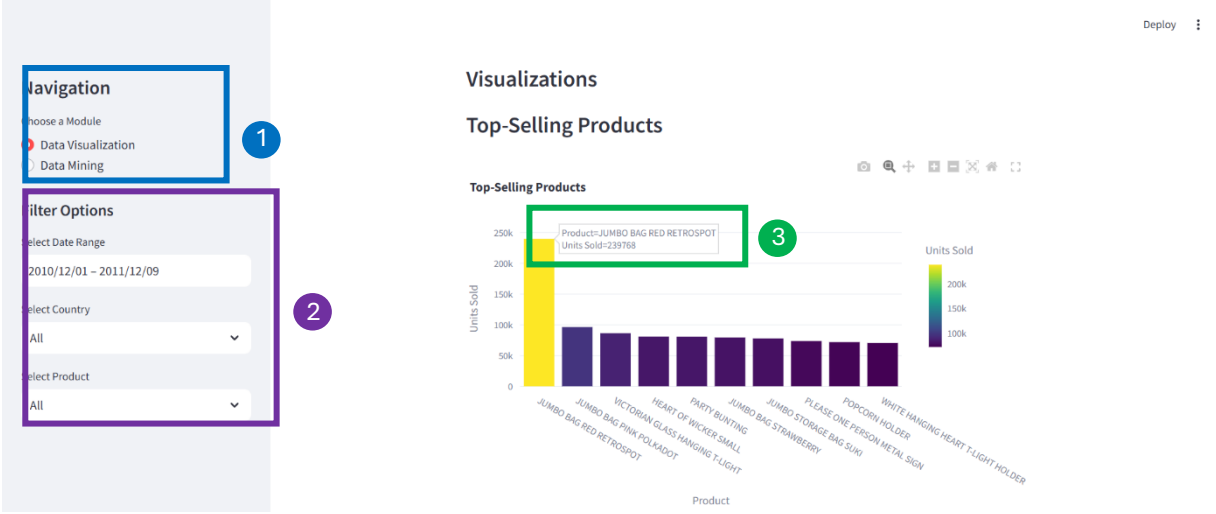


The "Actual vs Predicted Sales" line chart compares the model's predictions against actual sales data. The line for predicted sales helps in visualizing how well the linear regression model captures trends. The results may show close alignment between actual and predicted values, indicating a robust forecasting model. An additional chart projects future sales trends, showing steady growth or potential declines based on historical patterns. These insights are invaluable for financial forecasting and inventory planning, enabling businesses to anticipate demand and

optimize stock levels accordingly. For instance, the future sales forecast can guide businesses in stockpiling inventory ahead of anticipated peaks.

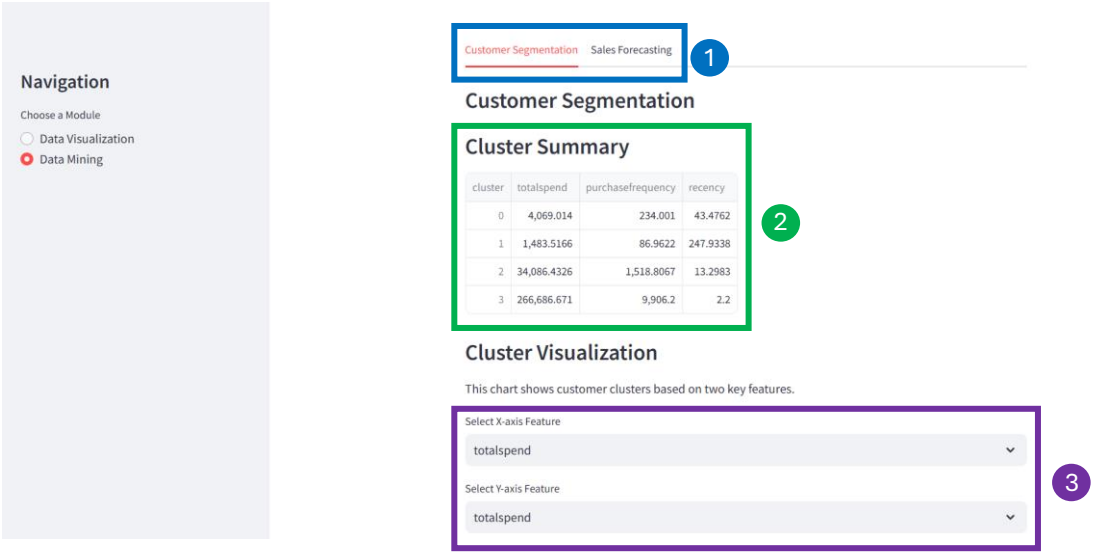
IV. USER GUIDE

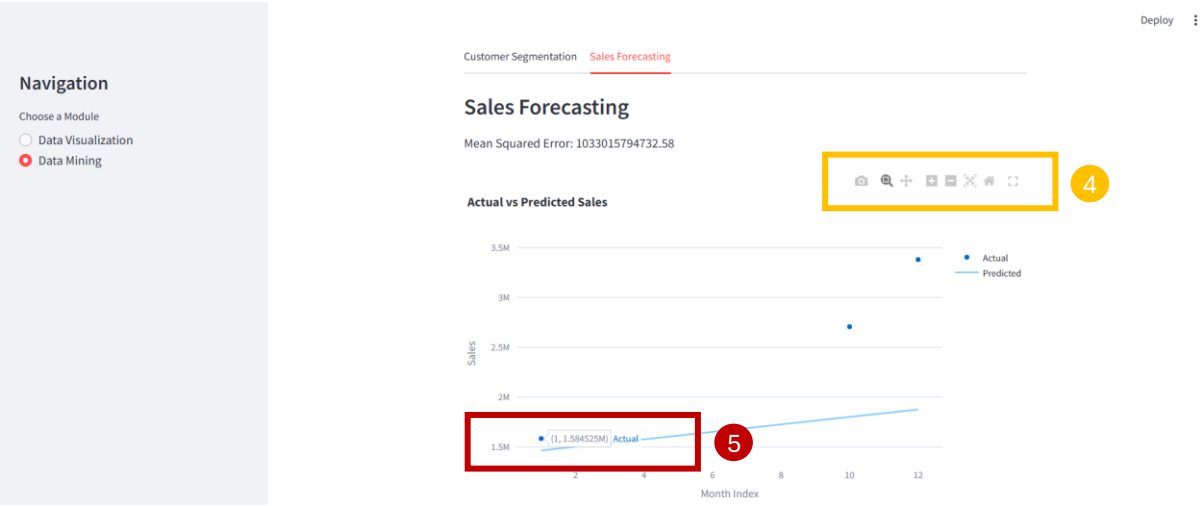
How to Use the Visualizations



1. Navigation:
    - Use the navigation sidebar to switch between "Data Visualization" and "Data Mining" modules.
2. Filters:
    - Apply date range, country, and product filters to customize the displayed data.
3. Charts:
    - Hover over charts for detailed tooltips.
    - Use the interactive capabilities to focus on specific data points.

How to Use the Data Mining Insights





1. Click which of the two you want to navigate.
2. View the table summarizing customer clusters based on total spend, purchase frequency, and recency.
3. Use the dropdown menus to select features for the X and Y axes (e.g., Total Spend vs. Purchase Frequency).
    - Total Spend: The cumulative revenue generated by customers.
    - Purchase Frequency: The number of transactions made by each customer.
    - Recency: The time since the last purchase.
4. Use chart toolbar for interacting with the chart or plot, such as downloading it as a PNG, zooming, panning, resetting views, and other related functionalities.
5. Use the tooltips to examine specific data points.

V. Challenges and Solutions

Here is a summary of challenges and their corresponding solutions for each phase of the project:

ETL (Extract, Transform, Load)

Challenges:

1. **Handling Missing or Incomplete Data**: Many records had missing CustomerID or Description fields, leading to potential inaccuracies in analysis.
    - **Solution**: Implemented data cleaning steps to drop rows with missing CustomerID and fill missing Description values with 'No Description'.
2. **Outlier Detection**: Extreme values in Quantity and UnitPrice skewed the analysis.
    - **Solution**: Filtered out rows with values outside the 1st and 99th percentiles to ensure data consistency.

3. **Standardizing Date Formats**: Dates were inconsistent, making it difficult to group data for analysis.
    - **Solution**: Converted all dates to a standardized YYYY-MM-DD format using Python.

Data Warehousing

Challenges:

1. **Schema Design**: Designing an effective schema that optimizes query performance while maintaining simplicity.
    - **Solution**: Adopted a star schema, placing fact tables at the center and dimension tables around it for clear relationships and fast analytical queries.
2. **Data Loading Issues**: Errors occurred when loading data into PostgreSQL due to incorrect data types and duplicate entries.
    - **Solution**: Validated data types before loading and implemented constraints to handle duplicates (e.g., using ON CONFLICT for dimension tables).
3. **Query Optimization**: Complex queries took longer to execute due to large datasets.
    - **Solution**: Indexed key columns and used aggregate functions to reduce computation during queries.

Data Visualization

Challenges:

1. **Interactive Filtering**: Providing an intuitive way for users to filter by date, country, and product.
    - **Solution**: Built a responsive sidebar in Streamlit with drop-down menus and date range pickers for real-time filtering.
2. **Presenting Complex Data**: Displaying large amounts of information in a clear and concise way.
    - **Solution**: Used bar charts, pie charts, and line graphs to focus on specific metrics and trends. Added tooltips for additional insights.
3. **Rendering Performance**: Visualizations with large datasets caused slow rendering.
    - **Solution**: Cached data loading and applied filtering at the database query level to reduce the amount of data passed to the app.

Data Mining

Challenges:

1. **Determining Features for Clustering**: Identifying meaningful features (e.g., total spend, purchase frequency, recency) for effective segmentation.
    - **Solution**: Used domain knowledge to choose relevant features and scaled them for clustering with the K-Means algorithm.
2. **Cluster Interpretability**: Explaining cluster characteristics to stakeholders.
    - **Solution**: Created visualizations (scatter plots and cluster summaries) to present cluster attributes clearly and added textual explanations.
3. **Sales Forecasting Accuracy**: The linear regression model initially produced high errors due to unclean data.
    - **Solution**: Cleaned and normalized data, retrained the model, and evaluated performance using Mean Squared Error (MSE) to ensure accuracy.