#□□ **Tapovona Business Dataset**

Introduction:

Tapovona is a business that focuses on providing organic fruits, vegetables, millets, and grains. The dataset captures detailed records of products sold, seasonal trends, and sales channels, which can be used to analyze the impact of seasonality, pricing trends, and customer preferences. It can help predict demand, optimize inventory, and understand market fluctuations for agricultural goods.

Key Features:

- Product ID: A unique identifier for each product sold.
- List item
- List item
- Product Name: Name of the product (e.g., Millet, Organic Tomatoes, Organic Wheat).

*Category: Type of product (e.g., Fruits, Vegetables, Grains, Millets).

- Supplier: Name of the supplier or farm from which the product was sourced.
- Quantity Available: Quantity of the product available for sale (numerical value).
- Quantity Sold: Quantity of the product sold (numerical value).
- Price per Unit (BDT): Price for each unit of the product (numerical value).
- Total Sales (BDT): Total sales amount for the product (calculated from Quantity Sold x Price per Unit).
- Date of Sale: Date when the sale occurred.
- Customer Location: Geographic location of the customer (e.g., Dhaka, Chittagong, Sylhet).
- Sales Channel: Where the sale took place (e.g., Online, Farmers Market, Retail Store).
- Seasonality: The season in which the product is most in demand (e.g., Summer, Winter).
- Harvest Date: Date when the product was harvested (if applicable).
- Expiry Date: Date when the product's freshness expires (if applicable).
- Packaging Type: Type of packaging used (e.g., Loose, Packaged, Bulk).
- Organic Certification: Whether the product is certified organic (Yes, No).

- Sustainability Practices: Any sustainability efforts involved in production (e.g., Water conservation, Use of organic fertilizers).
- Customer Feedback: Customer reviews or ratings (numerical or categorical).
- Return Rate: Percentage of returned items (numerical value).
- Profit Margin: The profit margin for the product (numerical value).
- Total Stock Cost (BDT): The total cost of stock for the product (calculated from Quantity Available x Cost per Unit).
- Supply Chain Efficiency: Efficiency of the supply chain (measured via delivery times or other KPIs).

Model Implementation:

In this project, a machine learning regression model is used to predict the Total Sales (BDT) for Tapovona's organic products. The dataset includes features like product category, price per unit, quantity sold, supplier, and sales channel. The data is cleaned and preprocessed by handling missing values and encoding categorical variables. Then, the features and target variable are separated, and a regression model (such as Random Forest) is trained to learn how different factors influence the total sales of the products.

Model Evaluation:

The model's performance is evaluated using metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R^2). These metrics help measure how accurate the sales predictions are and how well the model captures the relationship between input features and total sales.

- Mean Absolute Error (MAE): Measures the average magnitude of errors in a set of predictions, without considering their direction. It's useful for understanding how far off the model's predictions are in general.
- Root Mean Squared Error (RMSE): Provides a measure of the average magnitude of error in the predictions, but gives more weight to larger errors. This metric is sensitive to outliers.
- R-squared (R^2): Indicates the proportion of the variance in the target variable (Total Sales) that is explained by the model. A higher R^2 value means a better fit of the model to the data.

Visualization and Prediction:

Visualizations like feature importance charts help understand which factors most affect Total Sales (BDT). These charts provide insight into which features, such as Product Category, Price per Unit, or Quantity Sold, are the most influential in predicting sales.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

#Load the file
df=pd.read_csv('/content/Tapovona_CustomerData_10000Rows_SequentialID.csv')

df
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 10000,\n  \"fields\":
[\n    {\n      \"column\": \"Customer_ID\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 2886,\n
\"min\": 1,\n        \"max\": 10000,\n        \"num_unique_values\":
10000,\n        \"samples\": [\n          6253,\n          4685,\n
1732\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Customer_Name\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 9384,\n
\"samples\": [\n          \"Teresa Guerra\",\n          \"Melissa
Thomas\",\n          \"Kathryn Horton\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 13,\n        \"min\": 18,\n
\"max\": 65,\n        \"num_unique_values\": 48,\n        \"samples\":
[\n          36,\n          30,\n          57\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Location\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
7648,\n        \"samples\": [\n          \"East Tonyamouth\",\n
\"Allisonstad\",\n          \"Daltonhaven\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Product_Category\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n
\"Grain\",\n          \"Millet\",\n          \"Fruit\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Product_Name\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 20,\n        \"samples\": [\n
\"Banana\",\n          \"Rice\",\n          \"Mango\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
```

```
n    },\n    {\n      \"column\": \"Quantity_Purchased (kg)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2.7363610224201875,\n        \"min\": 0.5,\n        \"max\": 10.0,\n
\"num_unique_values\": 951,\n        \"samples\": [\n          3.01,\n
6.57,\n          1.69\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Price_per_kg\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 29.170263680420092,\n        \"min\":
20.03,\n        \"max\": 119.98,\n        \"num_unique_values\":
6297,\n        \"samples\": [\n          87.04,\n          54.79,\n
25.49\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total_Amount\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 257.3159288975785,\n        \"min\":
11.45,\n        \"max\": 1175.76,\n        \"num_unique_values\":
9354,\n        \"samples\": [\n          162.96,\n          183.64,\n
530.87\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Purchase_Date\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"num_unique_values\": 366,\n        \"samples\":
[\n          \"2024-07-06\",\n          \"2025-01-15\",\n
\"2024-05-16\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Payment_Method\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          \"UPI\",\n          \"Card\",\n          \"Online\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Delivery_Type\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Pickup\",\n          \"On-spot\",\n          \"Home Delivery\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Feedback_Rating\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1,\n        \"min\": 1,\n        \"max\": 5,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          3,\n
2,\n          1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Returning_Customer\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Yes\",\n          \"No\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```


```
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 10000,\n  \"fields\":
[\n    {\n      \"column\": \"Customer_ID\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 2886,\n
```

```
\"min\": 1,\n        \"max\": 10000,\n        \"num_unique_values\":
10000,\n        \"samples\": [\n          6253,\n          4685,\n
1732\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Customer_Name\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 9384,\n
\"samples\": [\n          \"Teresa Guerra\",\n          \"Melissa
Thomas\",\n          \"Kathryn Horton\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 13,\n        \"min\": 18,\n
\"max\": 65,\n        \"num_unique_values\": 48,\n        \"samples\":
[\n          36,\n          30,\n          57\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Location\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
7648,\n        \"samples\": [\n          \"East Tonyamouth\",\n
\"Allisonstad\",\n          \"Daltonhaven\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Product_Category\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n
\"Grain\",\n          \"Millet\",\n          \"Fruit\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Product_Name\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 20,\n        \"samples\": [\n
\"Banana\",\n          \"Rice\",\n          \"Mango\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Quantity_Purchased (kg)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2.7363610224201875,\n        \"min\": 0.5,\n        \"max\": 10.0,\n
\"num_unique_values\": 951,\n        \"samples\": [\n          3.01,\n
6.57,\n          1.69\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Price_per_kg\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 29.170263680420092,\n        \"min\":
20.03,\n        \"max\": 119.98,\n        \"num_unique_values\":
6297,\n        \"samples\": [\n          87.04,\n          54.79,\n
25.49\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total_Amount\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 257.3159288975785,\n        \"min\":
11.45,\n        \"max\": 1175.76,\n        \"num_unique_values\":
9354,\n        \"samples\": [\n          162.96,\n          183.64,\n
530.87\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Purchase_Date\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"num_unique_values\": 366,\n        \"samples\":
```

```
[\n          \"2024-07-06\",\n          \"2025-01-15\",\n
\"2024-05-16\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Payment_Method\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          \"UPI\",\n          \"Card\",\n          \"Online\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Delivery_Type\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Pickup\",\n          \"On-spot\",\n          \"Home Delivery\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Feedback_Rating\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1,\n        \"min\": 1,\n        \"max\": 5,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          3,\n
2,\n          1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Returning_Customer\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Yes\",\n          \"No\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Customer_ID              10000 non-null  int64
 1   Customer_Name            10000 non-null  object
 2   Age                      10000 non-null  int64
 3   Location                 10000 non-null  object
 4   Product_Category         10000 non-null  object
 5   Product_Name             10000 non-null  object
 6   Quantity_Purchased (kg)  10000 non-null  float64
 7   Price_per_kg             10000 non-null  float64
 8   Total_Amount             10000 non-null  float64
 9   Purchase_Date            10000 non-null  object
 10  Payment_Method           10000 non-null  object
 11  Delivery_Type            10000 non-null  object
 12  Feedback_Rating          10000 non-null  int64
 13  Returning_Customer       10000 non-null  object
dtypes: float64(3), int64(3), object(8)
memory usage: 1.1+ MB
```

```
df.describe()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 8,\n \"fields\": [\n{\n   \"column\": \"Customer_ID\",\n   \"properties\": {\n\"dtype\": \"number\",\n   \"std\": 3603.743586536124,\n\"min\": 1.0,\n   \"max\": 10000.0,\n\"num_unique_values\": 6,\n   \"samples\": [\n   10000.0,\n   5000.5,\n   7500.25\n   ],\n\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n\"dtype\": \"number\",\n   \"std\": 3522.330732276702,\n\"min\": 13.840298957303048,\n   \"max\": 10000.0,\n\"num_unique_values\": 8,\n   \"samples\": [\n   41.3996,\n   41.0,\n   10000.0\n   ],\n\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Quantity_Purchased (kg)\",\n\"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 3533.8077436772023,\n   \"min\": 0.5,\n   \"max\": 10000.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n5.244487999999999,\n   5.25,\n   10000.0\n   ],\n\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Price_per_kg\",\n\"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 3512.9896531510963,\n   \"min\": 20.03,\n   \"max\": 10000.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n70.04649099999999,\n   70.295,\n   10000.0\n   ],\n\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Total_Amount\",\n\"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 3411.6564122988907,\n   \"min\": 11.45,\n   \"max\": 10000.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n367.20065,\n   306.845,\n   10000.0\n   ],\n\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Feedback_Rating\",\n\"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 3534.554089739651,\n   \"min\": 1.0,\n   \"max\": 10000.0,\n\"num_unique_values\": 8,\n   \"samples\": [\n   2.9848,\n3.0,\n   10000.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
df.isnull().sum()
```

```
Customer_ID                0
Customer_Name              0
Age                        0
Location                   0
Product_Category           0
Product_Name               0
Quantity_Purchased (kg)    0
Price_per_kg               0
Total_Amount               0
```

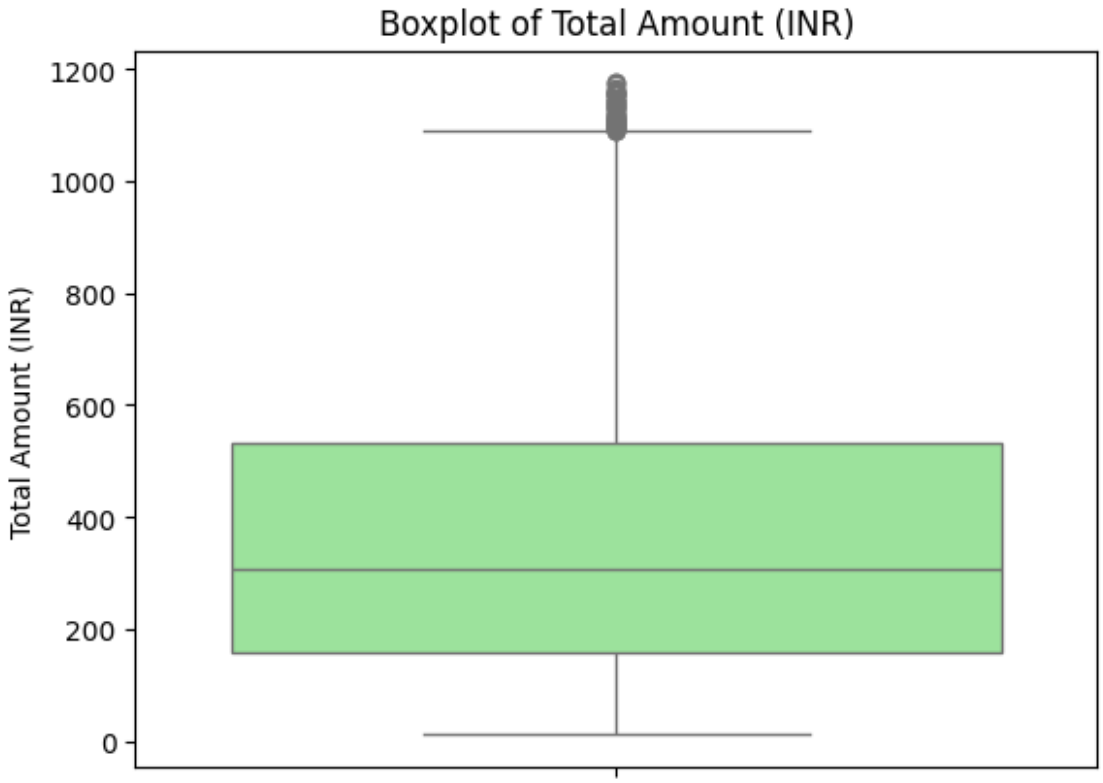
```
Purchase_Date              0
Payment_Method             0
Delivery_Type              0
Feedback_Rating            0
Returning_Customer         0
dtype: int64
```

```
#Fill missing values in categorical columns with mode
df['Location'] = df['Location'].fillna(df['Location'].mode()[0])
df['Product_Category'] =
df['Product_Category'].fillna(df['Product_Category'].mode()[0])
df['Product_Name'] =
df['Product_Name'].fillna(df['Product_Name'].mode()[0])
df['Payment_Method'] =
df['Payment_Method'].fillna(df['Payment_Method'].mode()[0])
df['Delivery_Type'] =
df['Delivery_Type'].fillna(df['Delivery_Type'].mode()[0])
df['Returning_Customer'] =
df['Returning_Customer'].fillna(df['Returning_Customer'].mode()[0])

# Filling missing values in categorical columns with their mode
df['Location']           = df['Location'].fillna(df['Location'].mode()
[0])
df['Product_Category']   =
df['Product_Category'].fillna(df['Product_Category'].mode()[0])
df['Product_Name']       =
df['Product_Name'].fillna(df['Product_Name'].mode()[0])
df['Payment_Method']     =
df['Payment_Method'].fillna(df['Payment_Method'].mode()[0])
df['Delivery_Type']      =
df['Delivery_Type'].fillna(df['Delivery_Type'].mode()[0])
df['Returning_Customer'] =
df['Returning_Customer'].fillna(df['Returning_Customer'].mode()[0])

df.isnull().sum()
```

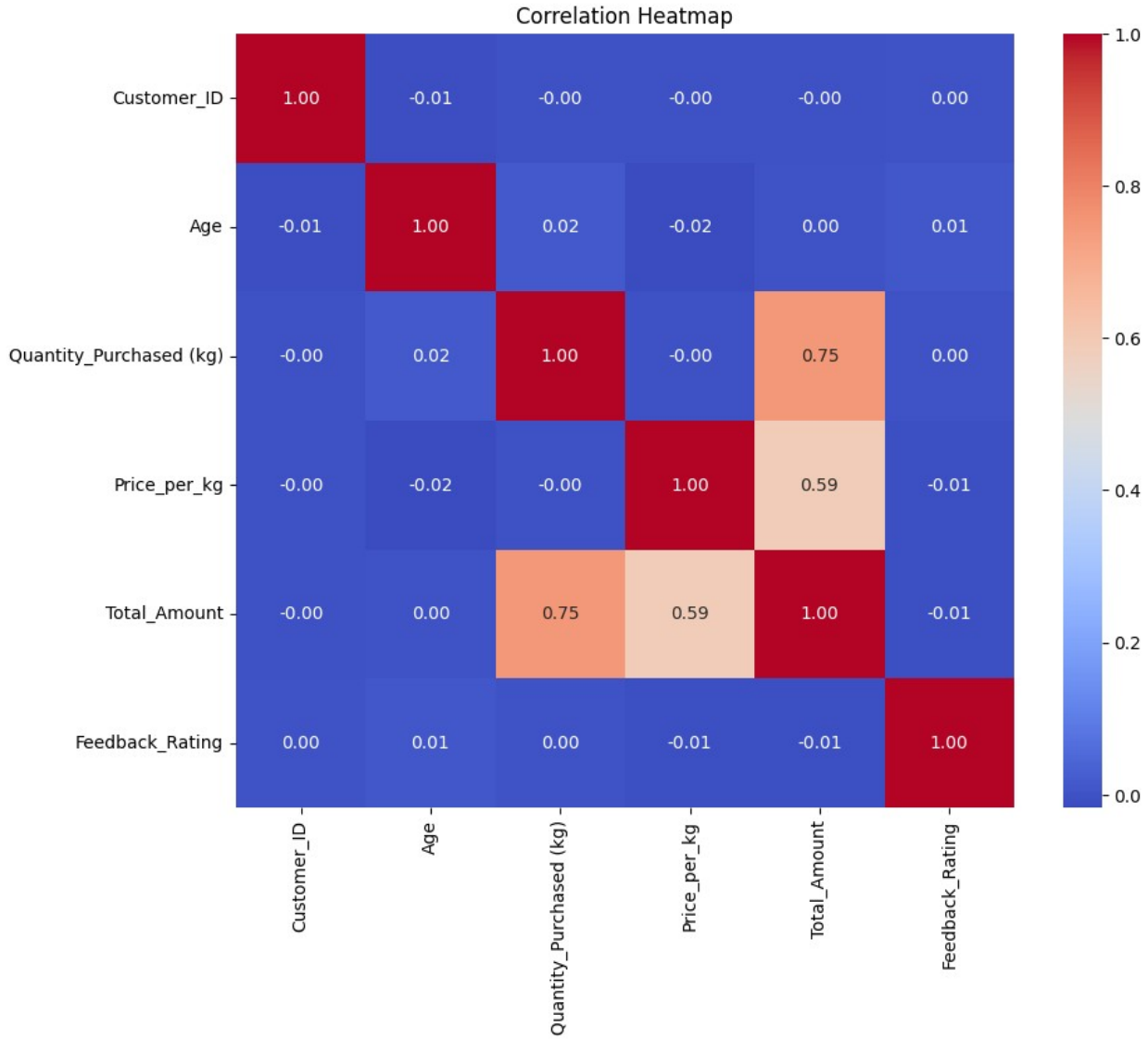
```
Customer_ID                0
Customer_Name              0
Age                        0
Location                   0
Product_Category           0
Product_Name               0
Quantity_Purchased (kg)    0
Price_per_kg               0
Total_Amount               0
Purchase_Date              0
Payment_Method             0
Delivery_Type              0
Feedback_Rating            0
```

```
Returning_Customer          0
dtype: int64
```

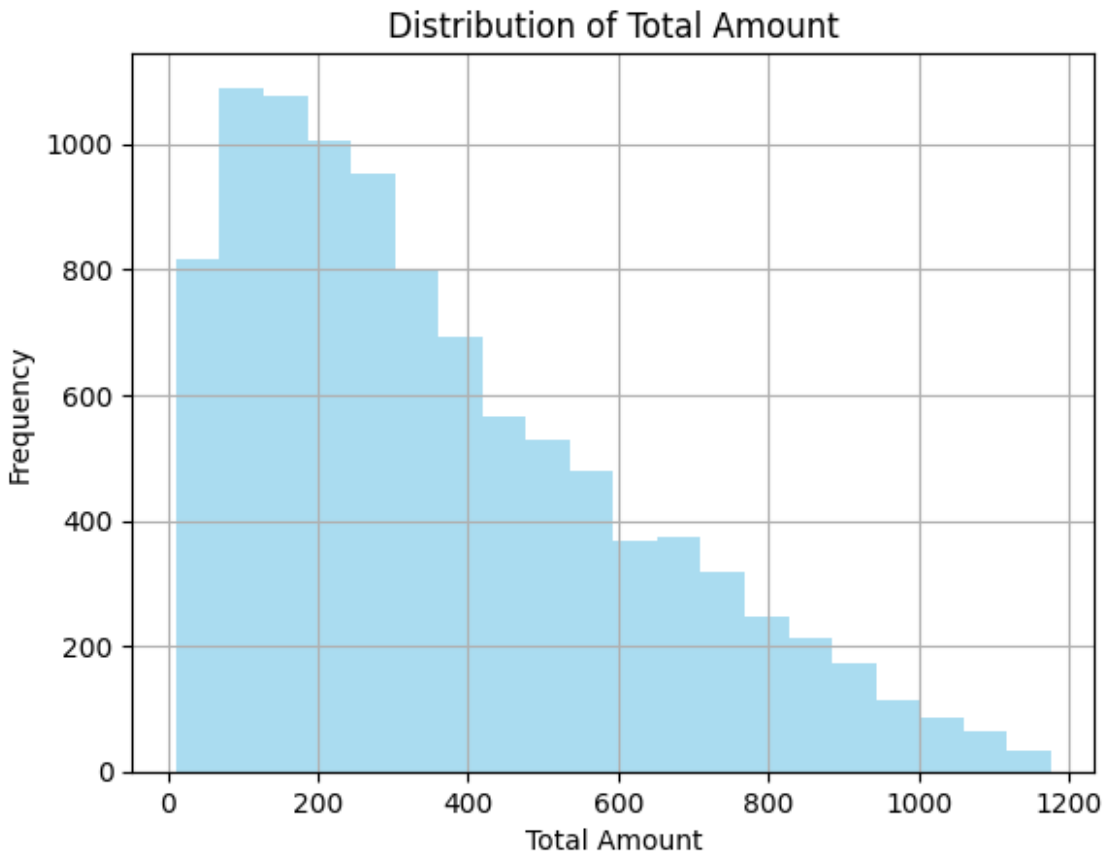
```
# Then plot the boxplot
sns.boxplot(y=df['Total_Amount'], color='lightgreen')
plt.title('Boxplot of Total Amount (INR)')
plt.ylabel('Total Amount (INR)')
plt.show()
```



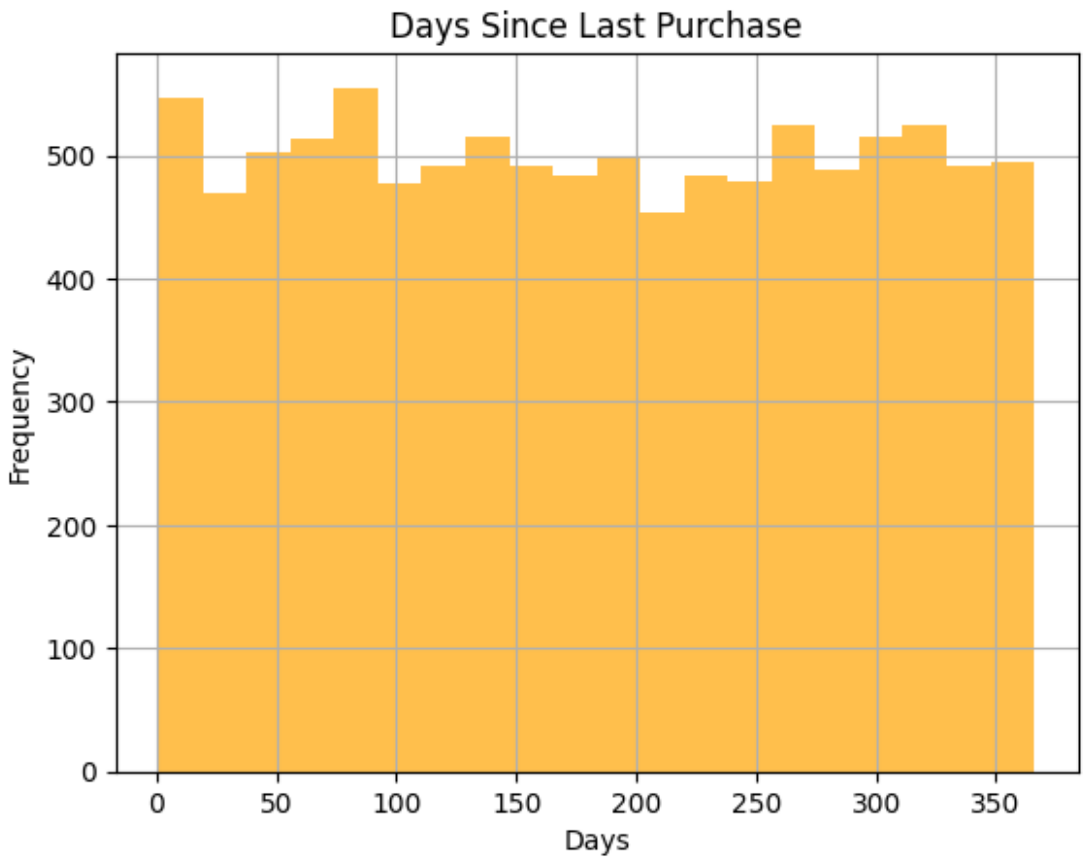

```
#Select only Numericals coloumns
Numerical_cols = df.select_dtypes(include=['float64',
'int64']).columns
#caluculate the correlation matrix
corr_matrix = df[Numerical_cols].corr()
#plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```



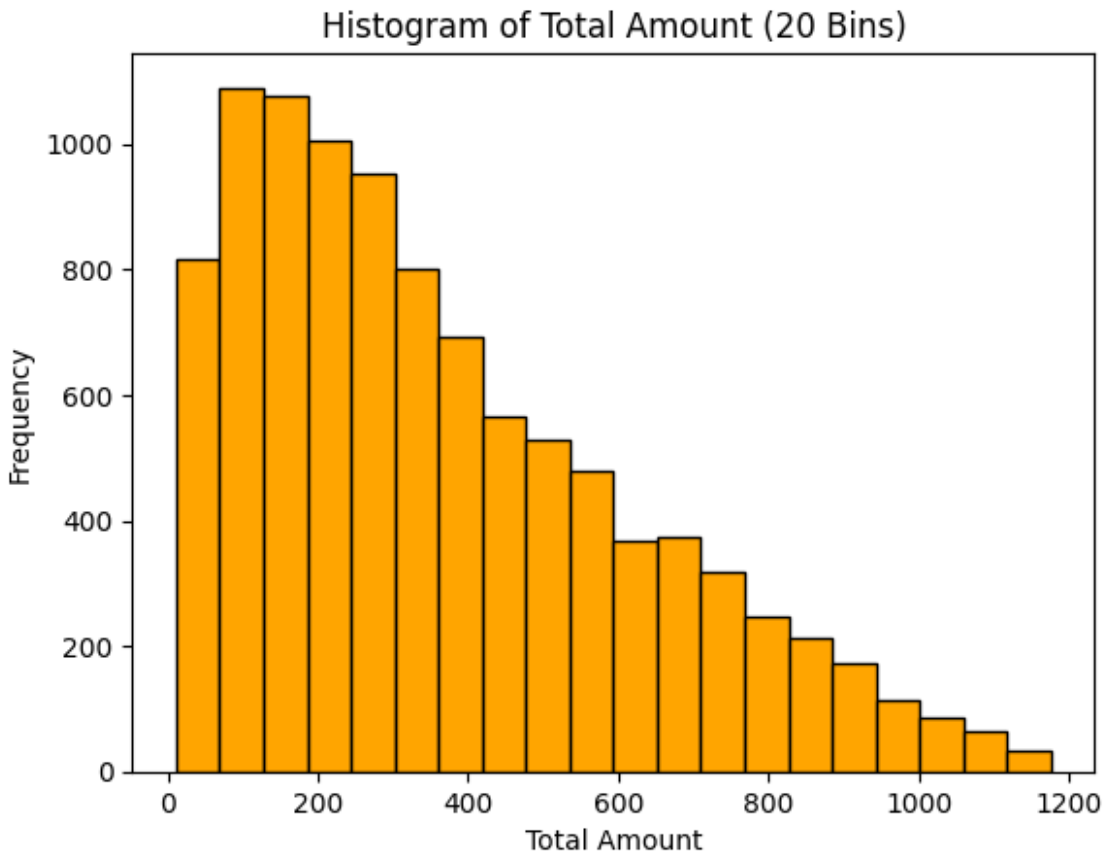

```
#Histogram for Total_Amount
df['Total_Amount'].hist(bins=20, color='skyblue', alpha=0.7)
plt.title('Distribution of Total Amount')
plt.xlabel('Total Amount')
plt.ylabel('Frequency')
plt.show()
```




```
#Histogram for days_since_purchase
import pandas as pd
from datetime import datetime
df['Purchase_Date'] = pd.to_datetime(df['Purchase_Date'])
current_date = datetime.now()
df['days_since_purchase'] = (current_date -
df['Purchase_Date']).dt.days

# Create histogram
df['days_since_purchase'].hist(bins=20, color='orange', alpha=0.7)
plt.title('Days Since Last Purchase')
plt.xlabel('Days')
plt.ylabel('Frequency')
plt.show()
```




```
df.to_csv('cleaned_tapovona_data.csv', index=False)

#plot histogram
plt.hist(df['Total_Amount'], bins=20, edgecolor='black',
color='orange')
plt.xlabel('Total Amount')
plt.ylabel('Frequency')
plt.title('Histogram of Total Amount (20 Bins)')
plt.show()
```



Univariate Analysis:

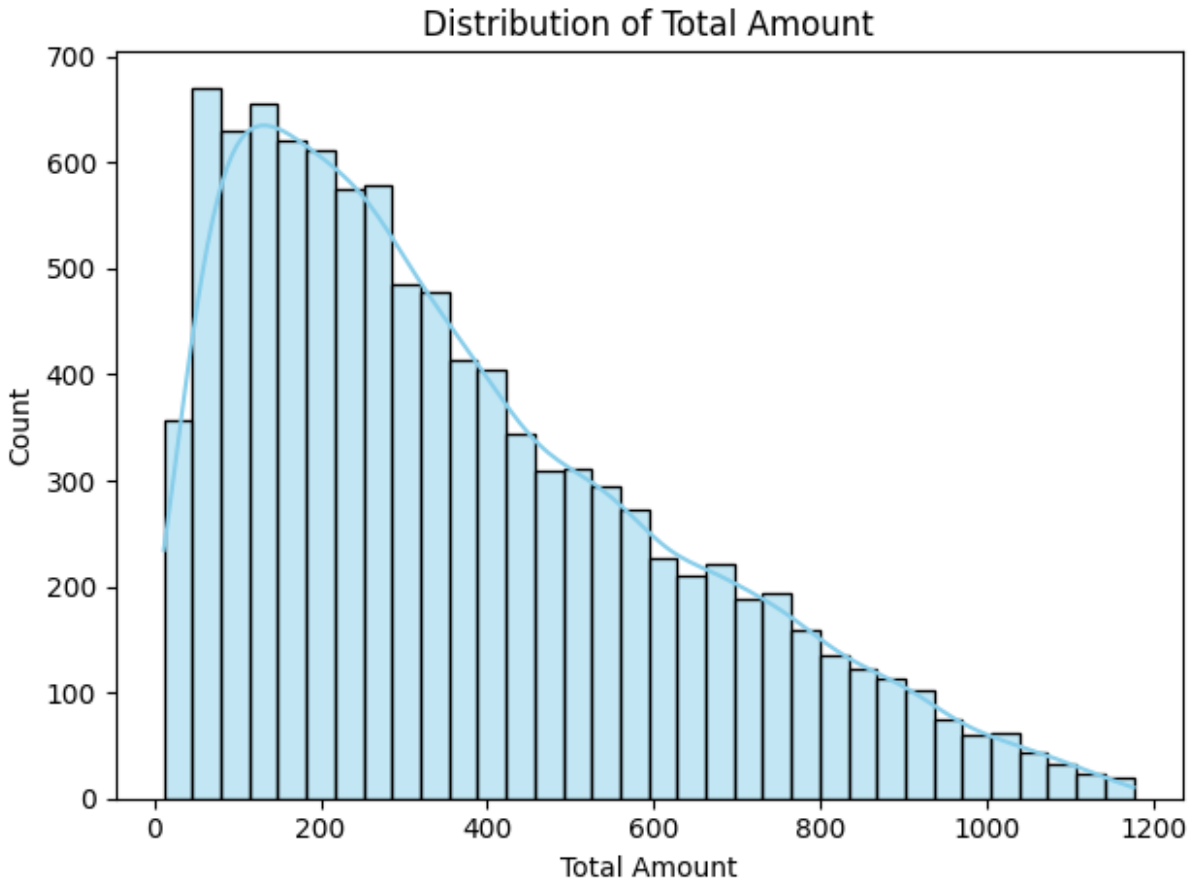
Univariate Analysis is a type of data visualization where we visualize only a single variable at a time. Univariate Analysis helps us to analyze the distribution of the variable present in the data so that we can perform further analysis. For instance, by visualizing variables like Price per Unit, Quantity Sold, or Total Sales (BDT), we can gain insights into their distribution, detect potential outliers, and understand the central tendency.

This analysis allows us to make informed decisions on data preprocessing, such as handling missing values, scaling, or transforming variables. Visualization techniques like histograms, box plots, and bar plots are commonly used to perform univariate analysis.

Histogram

```
sns.histplot(df['Total_Amount'], kde=True, color='skyblue',
edgecolor='black')
plt.xlabel('Total Amount')  # Replace with currency if known (e.g.,
"Total Amount (USD)")
plt.title('Distribution of Total Amount')
```

```
plt.tight_layout()  # Prevents label cutoff
plt.show()
```



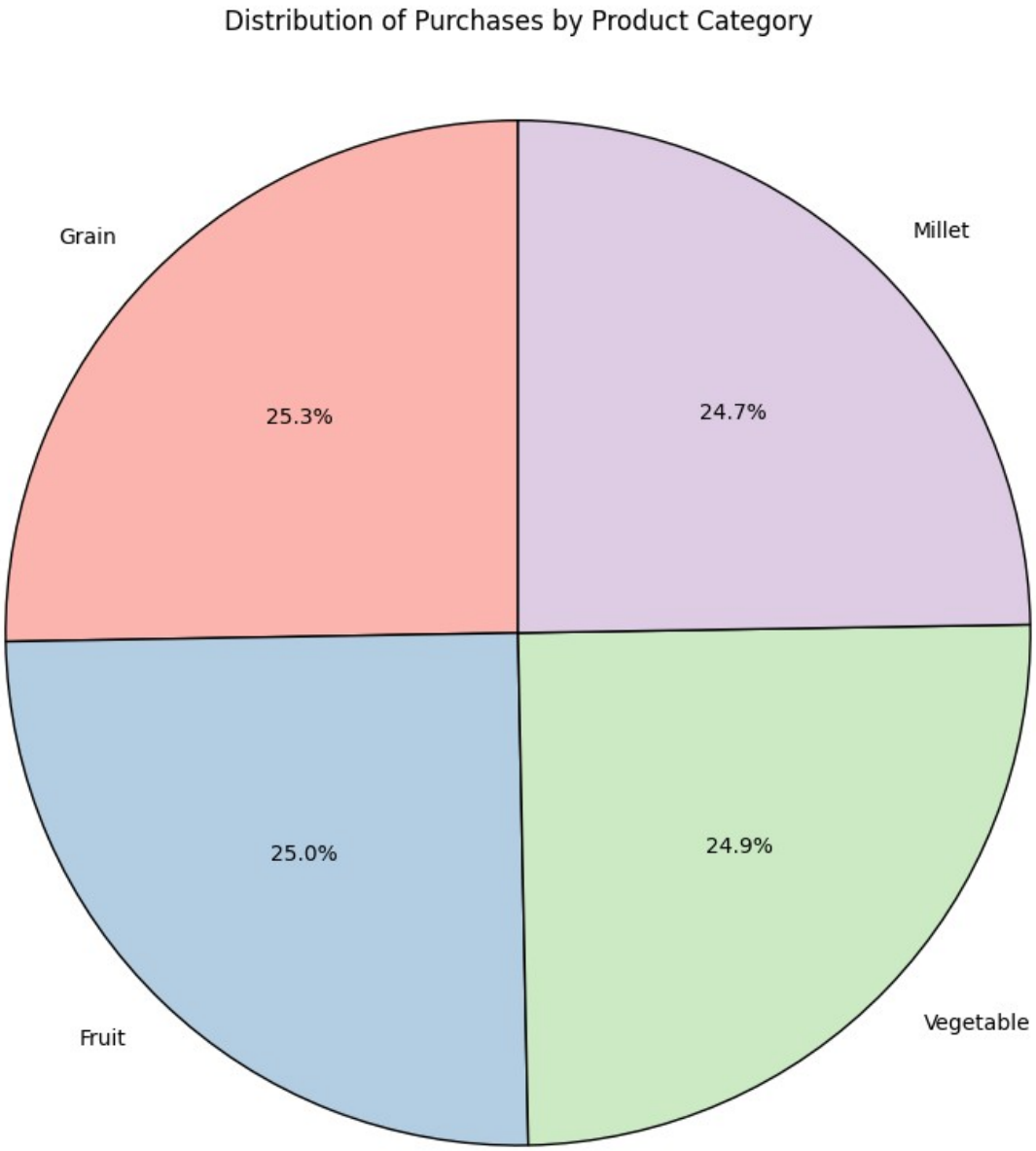
Bar Chart

```
sns.countplot(x='Product_Category', data=df, palette='viridis')
plt.title('Count of Purchases by Product Category')
plt.xticks(rotation=45)  # Rotate labels to avoid overlap
plt.tight_layout()
plt.show()
```

```
<ipython-input-21-8fa9bffc8acf>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(x='Product_Category', data=df, palette='viridis')
```

Pie Chart

```
x = df['Product_Category'].value_counts()  # Calculate the frequency
of each category

plt.figure(figsize=(8, 8))
plt.pie(x.values, labels=x.index, autopct='%1.1f%%', startangle=90,
        colors=plt.cm.Pastel1.colors, wedgeprops={'edgecolor':
'black'})
plt.title('Distribution of Purchases by Product Category', pad=20)
plt.axis('equal')  # Ensures pie is circular
plt.tight_layout()
plt.show()
```



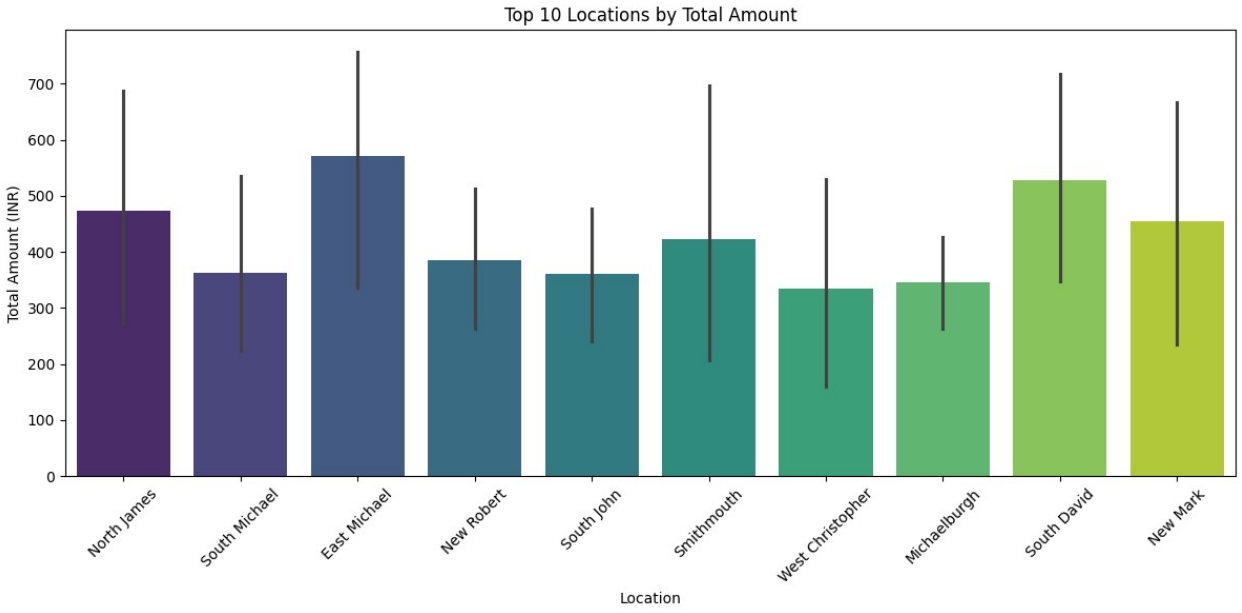
Bivariate Analysis:

Bivariate Analysis is the simultaneous analysis of two variables. It explores the relationship between two variables, examining whether there is an association and the strength of that association, or whether there are differences between the variables and the significance of those differences. Bivariate analysis helps us understand how two variables interact with each other and how they may impact each other.

Categorical vs Numerical

```
top_locations = df.groupby('Location')
['Total_Amount'].sum().nlargest(10).index

# Create the bar plot
plt.figure(figsize=(12, 6))
sns.barplot(x='Location', y='Total_Amount',
data=df[df['Location'].isin(top_locations)], palette='viridis')
plt.title('Top 10 Locations by Total Amount')
plt.xlabel('Location')
plt.ylabel('Total Amount (INR)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
<ipython-input-26-bf0eceeae111>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Location', y='Total_Amount',
data=df[df['Location'].isin(top_locations)], palette='viridis')
```

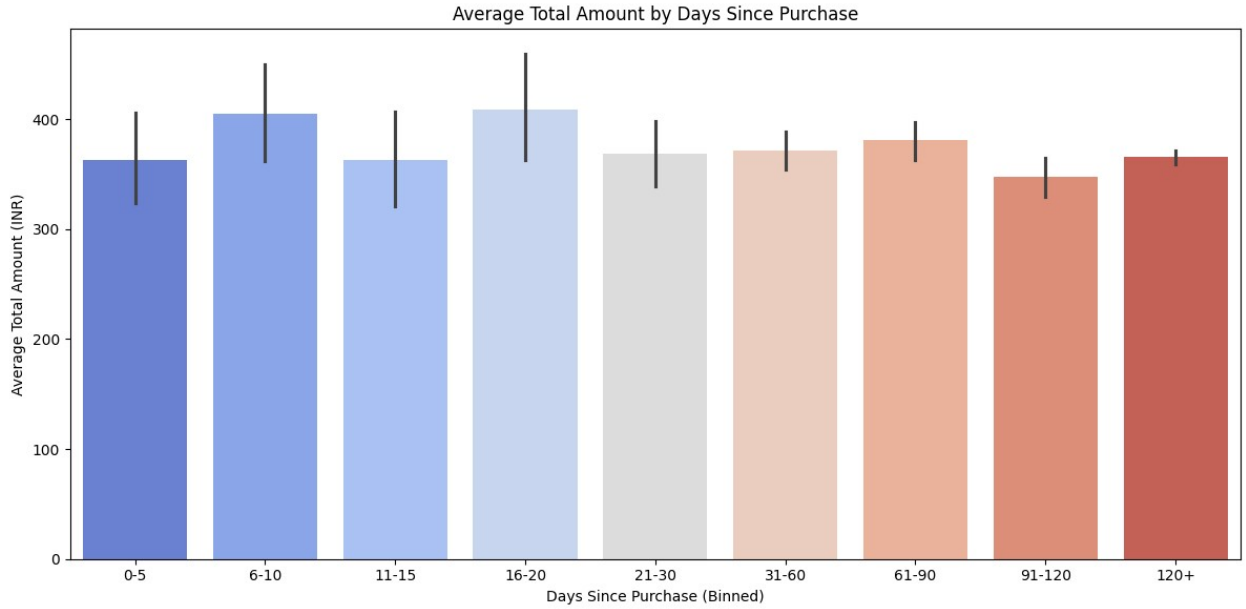


Numerical vs Numerical

```
# Create bins and labels
# Assuming 'days_since_purchase' is the relevant column for departure
calculation
bins = [0, 5, 10, 15, 20, 30, 60, 90, 120,
df['days_since_purchase'].max()]
labels = ['0-5', '6-10', '11-15', '16-20', '21-30', '31-60', '61-90',
```

```
'91-120', '120+']

# Create new binned column
df['departure_bin'] = pd.cut(df['days_since_purchase'], bins=bins,
labels=labels, include_lowest=True) # Changed to 'days_since_purchase'

# Group by binned column and plot
plt.figure(figsize=(12, 6))
sns.barplot(x='departure_bin', y='Total_Amount', data=df,
estimator=np.mean, palette='coolwarm') # Changed y to 'Total_Amount'
plt.title('Average Total Amount by Days Since Purchase') # Changed
title
plt.xlabel('Days Since Purchase (Binned)') # Changed x-axis label
plt.ylabel('Average Total Amount (INR)') # Changed y-axis label
plt.tight_layout()
plt.show()
```

```
<ipython-input-29-57e9d501a8a4>:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='departure_bin', y='Total_Amount', data=df,
estimator=np.mean, palette='coolwarm') # Changed y to 'Total_Amount'
```

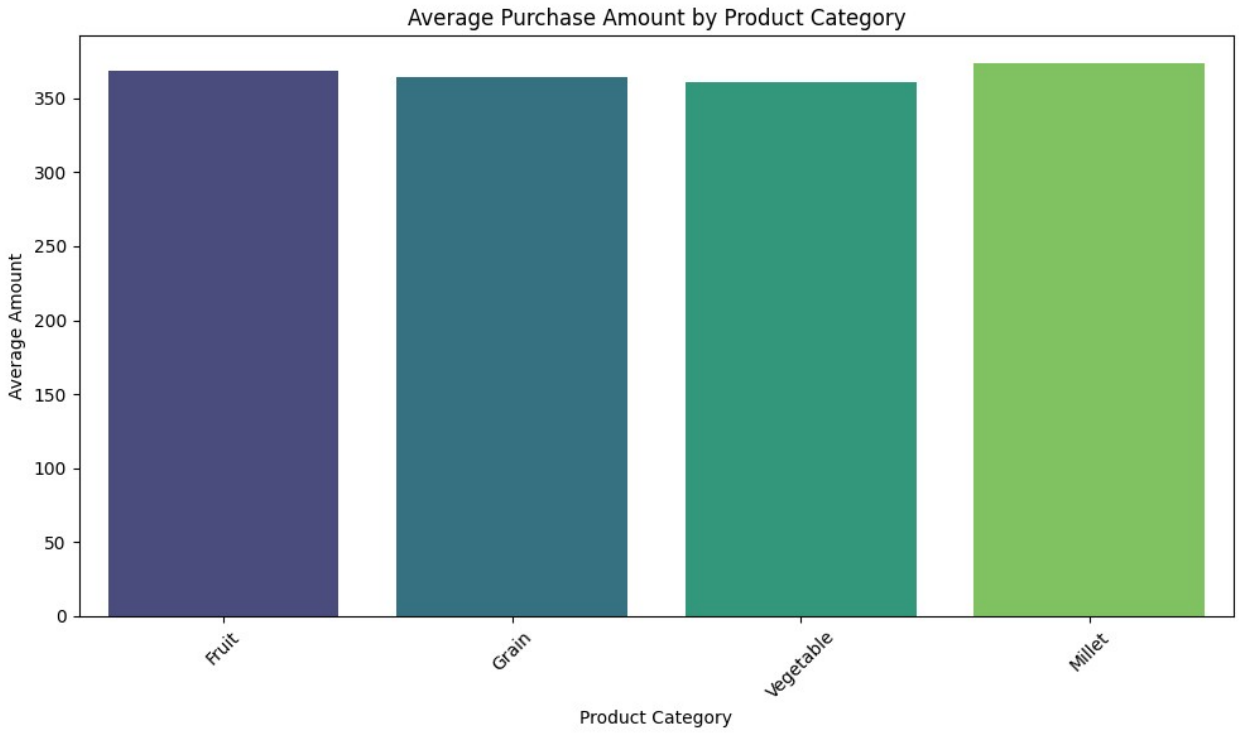


Categorical vs Categorical

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Product_Category', y='Total_Amount', data=df,
            palette='viridis', estimator='mean', ci=None)
```

```
plt.title('Average Purchase Amount by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Average Amount')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
<ipython-input-30-94b58c6b2d69>:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same
effect.

  sns.barplot(x='Product_Category', y='Total_Amount', data=df,
<ipython-input-30-94b58c6b2d69>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Product_Category', y='Total_Amount', data=df,
```



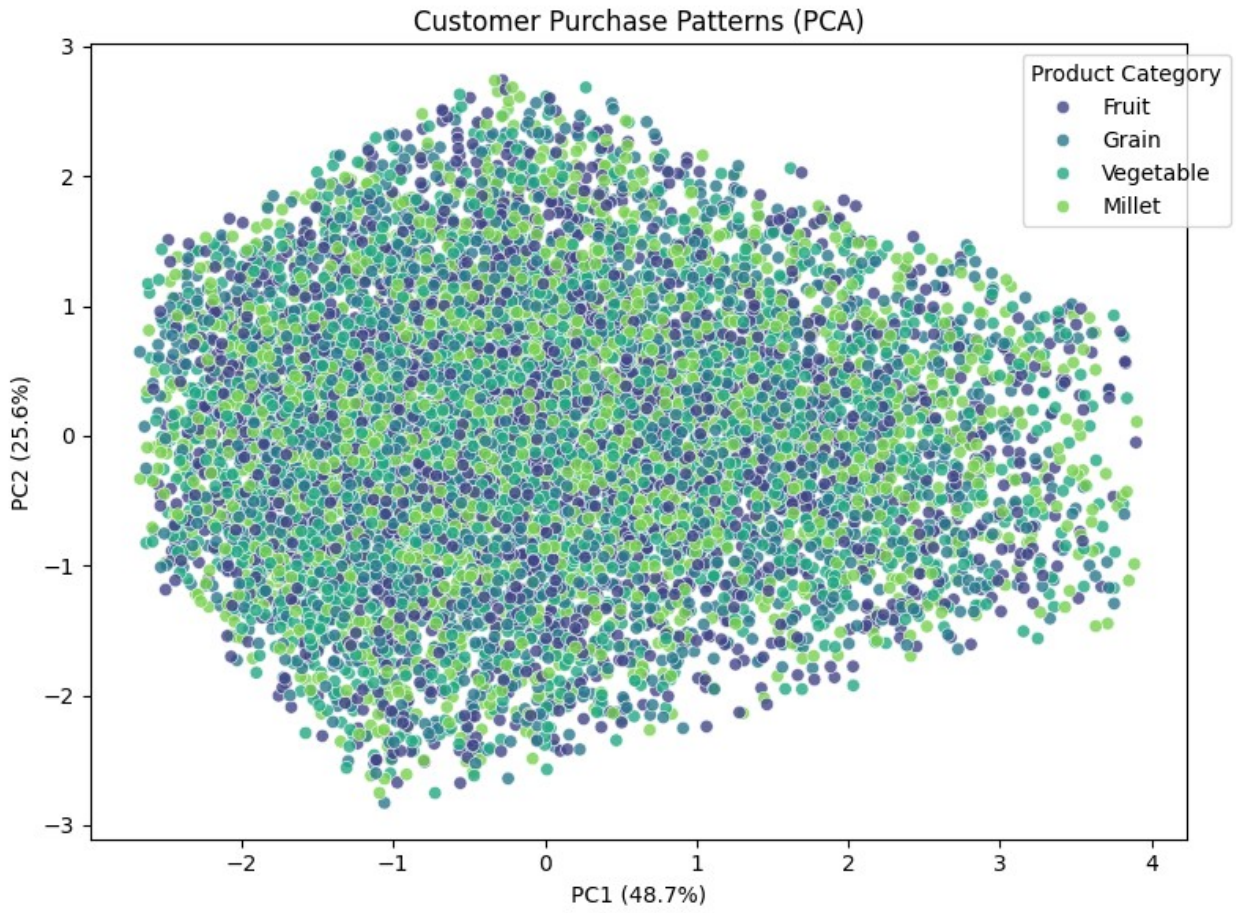
Multivariate Analysis:

Multivariate Analysis is an extension of bivariate analysis, which means it involves multiple variables at the same time to find correlations between them. It is a set of statistical models that examine patterns in multidimensional data by considering several data variables simultaneously.

PCA (Principal Component Analysis) is a dimensionality reduction technique used in multivariate analysis. It reduces the number of variables while keeping the most important information.

```
!pip install scikit-learn
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA # Import PCA class

numerical_cols = ['Age', 'Quantity_Purchased (kg)', 'Price_per_kg',
'Total_Amount']
X = df[numerical_cols].dropna()

# Standardize and apply PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform((X - X.mean()) / X.std())  # Standardized

# Visualize with hue=Product_Category or Payment_Method
plt.figure(figsize=(8,6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1],
                hue=df.loc[X.index, 'Product_Category'],  # Color by
category
                palette='viridis',
                alpha=0.8)
plt.title('Customer Purchase Patterns (PCA)')
plt.xlabel(f'PC1 ({pca.explained_variance_ratio_[0]:.1%})')
plt.ylabel(f'PC2 ({pca.explained_variance_ratio_[1]:.1%})')
plt.legend(bbox_to_anchor=(1.05, 1), title='Product Category')
plt.tight_layout()
plt.show()
```

```
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
```


Customer Purchase Patterns (PCA)
Product Category
Fruit
Grain
Vegetable
Millet
PC2 (25.6%)
PC1 (48.7%)
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
4