Understanding Customer Behavior through Transactional Data Analysis

Introduction:

Understanding customer behavior through transactional data analysis is a crucial aspect of modern business strategy. By leveraging transactional data, businesses can gain valuable insights into customer preferences, buying habits, and trends, allowing them to tailor their products and services to meet customer needs effectively.

Objectives:

The objective of analyzing transactional data is to gain a comprehensive understanding of customer behavior. This includes studying buying habits, social trends, and background factors that influence purchasing decisions. By doing so, businesses can create more enticing products and service offers, improve customer experiences, and ultimately boost profitability. Additionally, the objective is to utilize data analytics to gain complete visibility into the buyer journey, analyze browsing time, transaction data, and purchase histories, and create specific customer segments for targeted marketing and personalized communication.

Data Set Description:

Hackathon_Ideal_Data: Contains brand level data for 10 stores for the last 3 months.

Hackathon_Working_Data: Includes data for selected stores with missing and/or incomplete information.

Hackathon_Mapping_File: Provides column names and descriptions for better understanding of the dataset.

Hackathon_Validation_Data: Data stores and product groups for which total value prediction is required.

Sample Submission: Represents the format for uploading output, including columns and values required.

Methodology:

- **Data Preprocessing:** Clean the dataset, handle missing values, and transform variables if necessary.
- **Exploratory Data Analysis (EDA):** Explore the dataset through summary statistics, visualizations, and correlation analysis to understand customer behavior.
- **Customer Segmentation:** Utilize clustering algorithms to identify distinct customer segments based on their purchasing behavior.
- **Predictive Modeling:** Develop predictive models using regression or time series analysis to forecast customer spending or predict total sales value for validation data.
- **Interpretation and Insights:** Interpret the results of analysis and modeling to derive actionable insights for business stakeholders.

Importing necessary libraries:

```
[1] #Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

Importing datasets:

```
#Importing datasets
ideal_data=pd.read_csv("/content/Hackathon_Ideal_Data.csv")
mapping_file=pd.read_csv("/content/Hackathon_Mapping_File.csv")
validation_data=pd.read_csv("/content/Hackathon_Validation_Data.csv")
working_data=pd.read_csv("/content/Hackathon_Working_Data.csv")
sample_submission=pd.read_csv("/content/Sample Submission.csv")
print(ideal_data.head())
print(mapping_file.head())
print(working_data.head())
print(sample_submission.head())
```

Display the dataset:

```
   MONTH STORECODE  QTY  VALUE                 GRP                SGRP  \
0     M1        P1   25     83  HAIR CONDITIONERS  HAIR CONDITIONERS
1     M1        P1    6     22  HAIR CONDITIONERS  HAIR CONDITIONERS
2     M1        P1    4     15  HAIR CONDITIONERS  HAIR CONDITIONERS
3     M1        P1   15     60  HAIR CONDITIONERS  HAIR CONDITIONERS
4     M1        P2    0      0  HAIR CONDITIONERS  HAIR CONDITIONERS

               SSGRP                         CMP          MBRD  \
0  HAIR CONDITIONERS  HINDUSTAN UNILEVER LIMITED          DOVE
1  HAIR CONDITIONERS  HINDUSTAN UNILEVER LIMITED          DOVE
2  HAIR CONDITIONERS  HINDUSTAN UNILEVER LIMITED          DOVE
3  HAIR CONDITIONERS                L'OREAL INDIA       GARNIER
4  HAIR CONDITIONERS  HINDUSTAN UNILEVER LIMITED  CLINIC PLUS

                     BRD
0  DOVE HAIR FALL RESCUE
1    DOVE INTENSE REPAIR
2   DOVE OXYGEN MOISTURE
3                FRUCTIS
4            CLINIC PLUS
           File Name Column Name            Column Description
0  Hackathon_Ideal_Data       MONTH         Month ID (M1, M2, M3)
1                   NaN   STORECODE  STORE CODE (P1, P2, …, P10)
2                   NaN         QTY                    Sales Unit
3                   NaN       VALUE                   Sales Value
4                   NaN         GRP                      Category
   MONTH STORECODE  DAY BILL_ID  BILL_AMT  QTY  VALUE  PRICE  \
0     M1        N1    4    T375     225.0  1.0  225.0  225.0
1     M1        N1    4    T379      95.0  1.0   95.0   95.0
2     M1        N1    4    T381      10.0  1.0   10.0   10.0
3     M1        N1    4    T382     108.0  1.0  108.0  108.0
4     M1        N1    4    T384      19.0  1.0   19.0   19.0

                        GRP                      SGRP                     SSGRP  \
0      BUTTER MARGR  (4/94)                    BUTTER                    SALTED
1  CONFECTIONERY - ECLAIRS  CONFECTIONERY - ECLAIRS  CONFECTIONERY - ECLAIRS
2                 CHOCOLATE          CHOCOLATE PANNED          CHOCOLATE PANNED
3              PACKAGED TEA               MAIN PACKS                MAIN PACKS
4         ALL IODISED SALT            POWDERED SALT             POWDERED SALT

                        CMP            MBRD                  BRD
0                 G C M M F            AMUL                 AMUL
1               PARLE PRODS          MELODY     MELODY CHOCOLATY
2  MONDELEZ INTERNATIONAL  CADBURY SHOTS        CADBURY SHOTS
3       GUJ TEA PROCESSORS      WAGH BAKRI   WAGH BAKRI INSTANT
4                 TATA CHEM            TATA            TATA SALT
        ID  TOTALVALUE
0  1112535           0
1  1112539           1
2  1112543           2
3  1112547           3
4  1112551           4
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_as
  and should_run_async(code)
```

Handling Missing Values:

To identify missing values and removing the missing values:

```
#Handling missing data
print(ideal_data.isnull().sum())
is_any_missing_data=ideal_data.isna().any().any()
print("Are there any missing values?")
print(is_any_missing_data)
ideal_data.dropna(axis=0, inplace=True)
ideal_data.dropna(axis=1, inplace=True)
```

```
MONTH        0
STORECODE    0
QTY          0
VALUE        0
GRP          0
SGRP         0
SSGRP        0
CMP          0
MBRD         0
BRD          0
dtype: int64
Are there any missing values?
False
```

Removing duplicates values:

```
ideal_data.drop_duplicates(inplace=True)
ideal_data.duplicated().sum()
```

```
0
```

```
working_data.drop_duplicates(inplace=True)
working_data.duplicated().sum()
```

```
0
```

DATA PROCESSING AND FEATURE ENGINEERING

```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="mean")
ideal_data_filled = imputer.fit_transform(ideal_data.select_dtypes(include=['int', 'float']))
ideal_data_filled = pd.DataFrame(ideal_data_filled, columns=ideal_data.select_dtypes(include=['int', 'float']).columns)
```

Data cleaning,normalization and standardization:

```
[17] # Data cleaning, normalization, and standardization
scaler = StandardScaler()
working_data_scaled = scaler.fit_transform(working_data.select_dtypes(include=['int', 'float']))
working_data_scaled = pd.DataFrame(working_data_scaled, columns=working_data.select_dtypes(include=['int', 'float'])
```

```
encoder = OneHotEncoder()
ideal_data_encoded = pd.get_dummies(ideal_data.select_dtypes(include=['object']))

# Concatenate numerical and encoded categorical features
ideal_data_preprocessed = pd.concat([ideal_data_filled, ideal_data_encoded], axis=1)

# Print preprocessed data
print("Ideal Data after preprocessing:")
print(ideal_data_preprocessed.head())
print("\nWorking Data after scaling:")
print(working_data_scaled.head())
```

✓ 0s completed at 7:04 PM

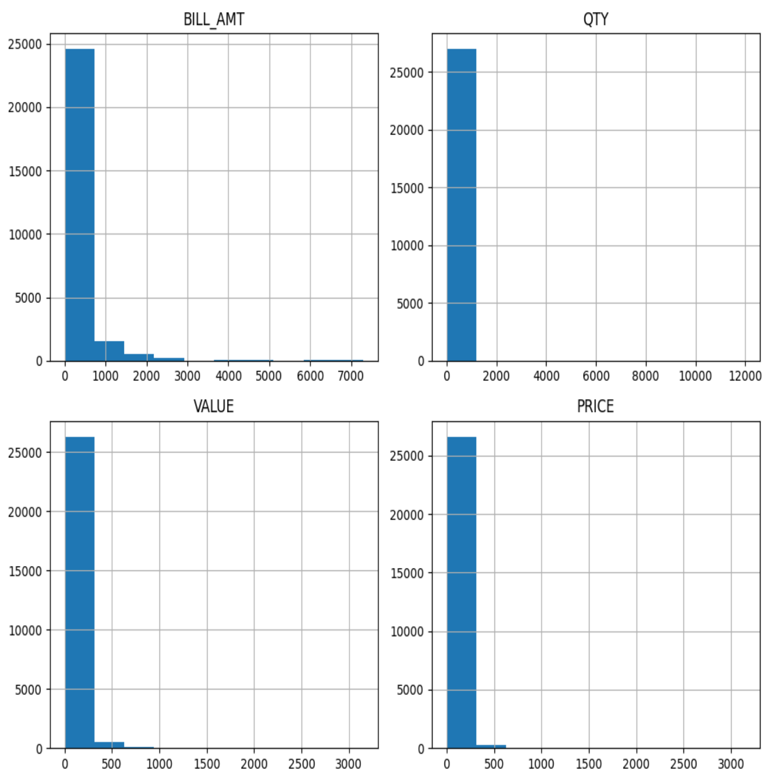
```
Ideal Data after preprocessing:
    QTY  VALUE  MONTH_M1  MONTH_M2  MONTH_M3  STORECODE_P1  STORECODE_P10  \
0  25.0   83.0      True     False     False          True          False
1   6.0   22.0      True     False     False          True          False
2   4.0   15.0      True     False     False          True          False
3  15.0   60.0      True     False     False          True          False
4   0.0    0.0      True     False     False         False          False

   STORECODE_P2  STORECODE_P3  STORECODE_P4  ...  BRD_ZANDU  BRD_ZANDU GEL  \
0         False         False         False  ...      False          False
1         False         False         False  ...      False          False
2         False         False         False  ...      False          False
3         False         False         False  ...      False          False
4          True         False         False  ...      False          False

   BRD_ZANDU ULTRA POWER  BRD_ZATPAT  BRD_ZED BLACK  \
0                  False       False          False
1                  False       False          False
2                  False       False          False
3                  False       False          False
4                  False       False          False

   BRD_ZED BLACK  DEEP MOGRA  BRD_ZED BLACK DEEP GULAB  BRD_ZED BLACK MANTHAN  \
0                      False                     False                  False
1                      False                     False                  False
2                      False                     False                  False
3                      False                     False                  False
4                      False                     False                  False

   BRD_ZED BLACK PANCHDEEP  BRD_ZOOPY
0                    False      False
1                    False      False
2                    False      False
3                    False      False
4                    False      False

[5 rows x 3457 columns]

Working Data after scaling:
        DAY  BILL_AMT       QTY     VALUE     PRICE
0 -1.245983 -0.099349 -0.032506  1.330044  2.023710
1 -1.245983 -0.339200 -0.032506  0.229519  0.495384
2 -1.245983 -0.496026 -0.032506 -0.490054 -0.503906
3 -1.245983 -0.315215 -0.032506  0.339572  0.648217
4 -1.245983 -0.479421 -0.032506 -0.413864 -0.398098
```

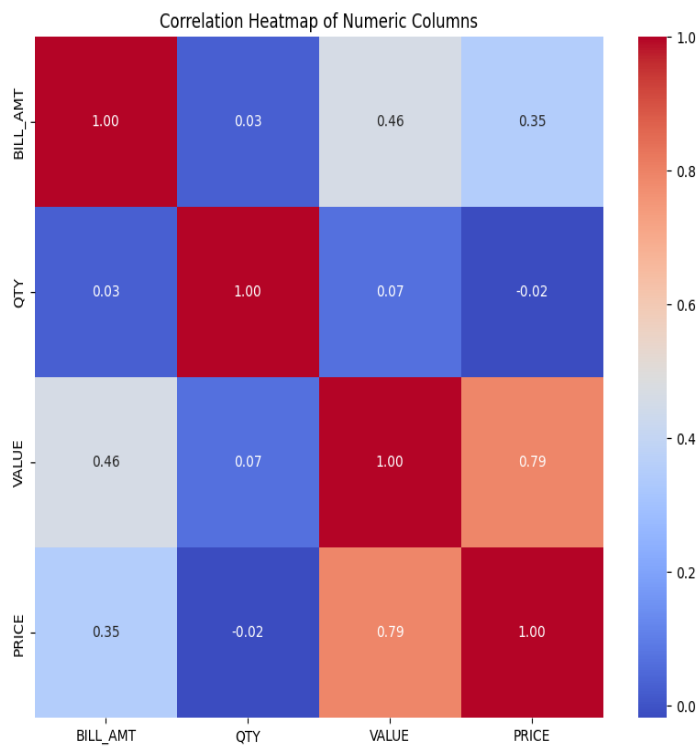
Ideal Data after preprocessing:

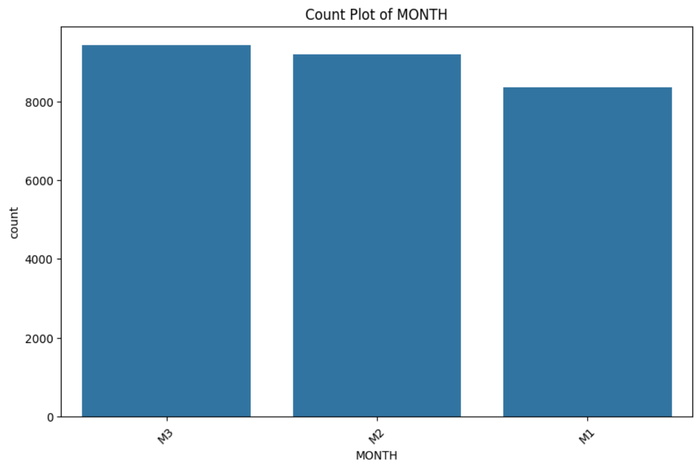
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Read the CSV file into a DataFrame
working_data = pd.read_csv("/content/Hackathon_Working_Data.csv")

# Plot the distribution of numeric columns
numeric_cols = ['BILL_AMT', 'QTY', 'VALUE', 'PRICE']
working_data[numeric_cols].hist(figsize=(10, 8))
plt.tight_layout()
plt.show()

# Plot the correlation heatmap of numeric columns
plt.figure(figsize=(10, 8))
sns.heatmap(working_data[numeric_cols].corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Numeric Columns')
plt.show()

# Plot count plots for categorical columns
categorical_cols = ['MONTH', 'STORECODE', 'GRP', 'SGRP']
for col in categorical_cols:
    plt.figure(figsize=(10, 6))
    sns.countplot(data=working_data, x=col, order=working_data[col].value_counts().index)
    plt.title(f'Count Plot of {col}')
    plt.xticks(rotation=45)
    plt.show()
```

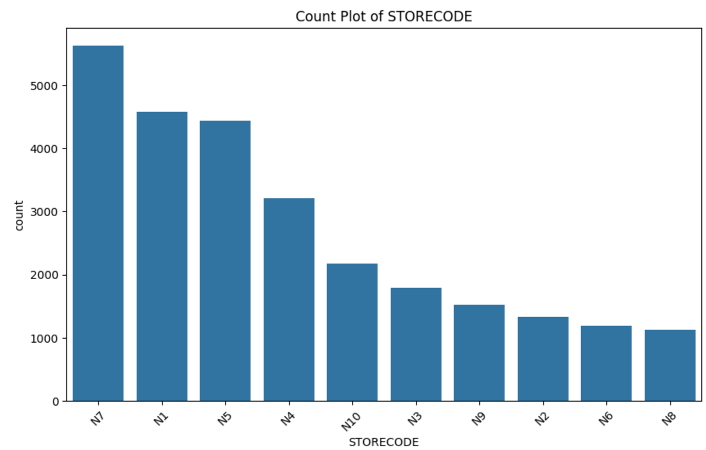
EXPLORATORY DATA ANALYSIS (EDA)

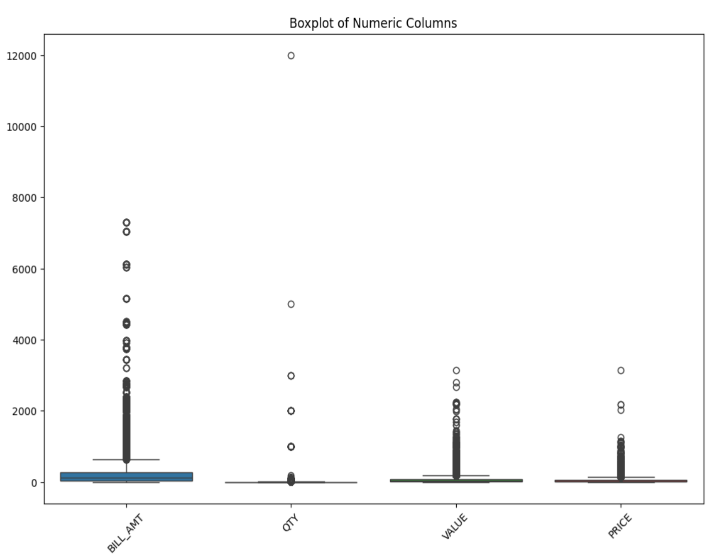
```
# Plot boxplots to identify outliers
plt.figure(figsize=(12, 8))
sns.boxplot(data=working_data[numeric_cols])
plt.title('Boxplot of Numeric Columns')
plt.xticks(rotation=45)
plt.show()

# Plot scatter plots for pairwise relationships between numeric columns
sns.pairplot(working_data[numeric_cols], diag_kind='kde')
plt.suptitle('Pairwise Relationships between Numeric Columns')
plt.show()
```

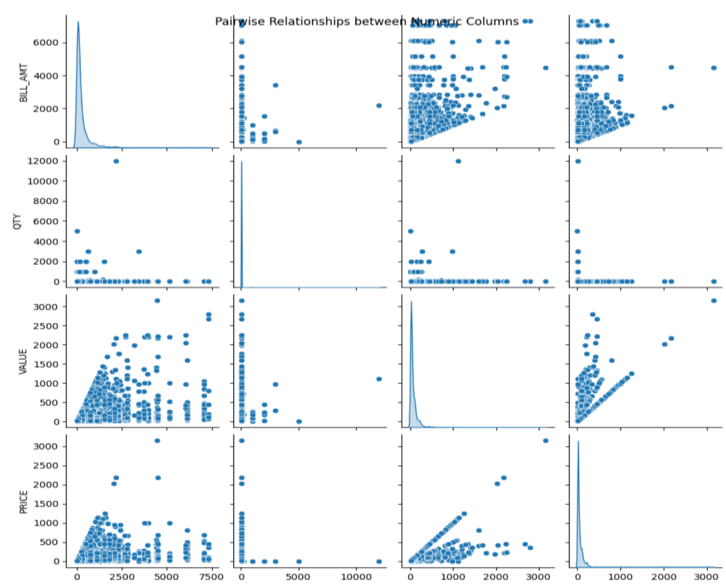





Count Plot of MONTH
count
MONTH
M3
M2
M1


Count Plot of STORECODE
count
STORECODE
N7
N1
N5
N4
N10
N3
N9
N2
N6
N8

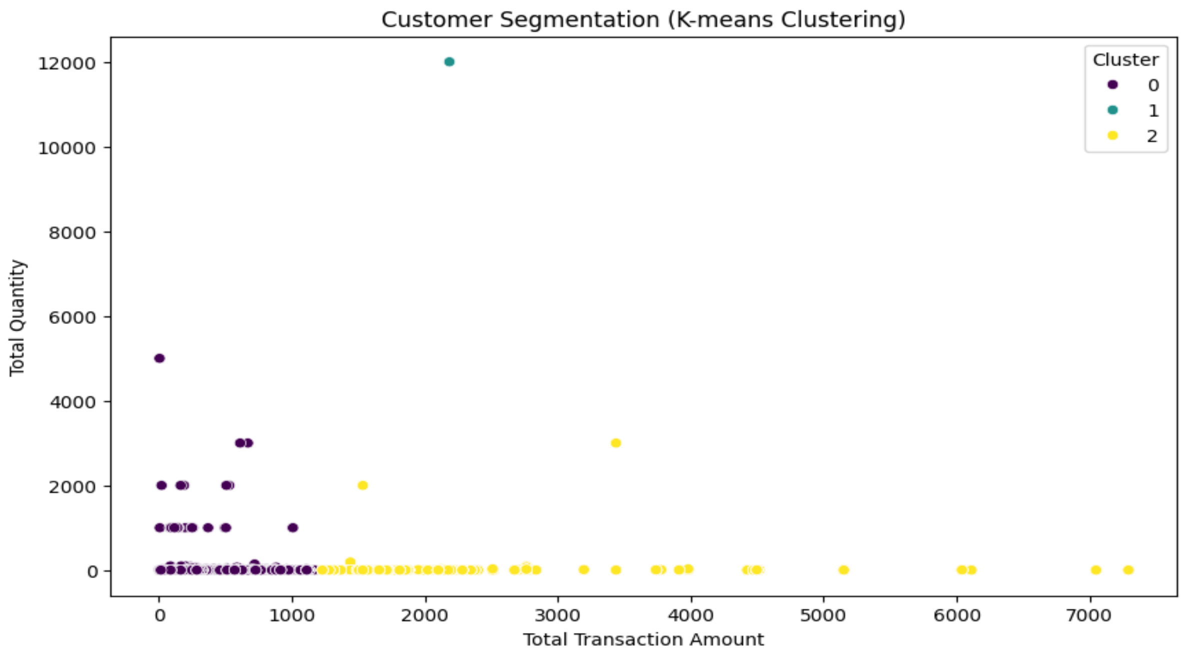

Boxplot of Numeric Columns
BILL_AMT
QTY
VALUE
PRICE


Pairwise Relationships between Numeric Columns
BILL_AMT
QTY
VALUE
PRICE

CUSTOMER SEGMENTATION WITH K-MEANS CLUSTERING

```
# Apply K-means clustering
kmeans = KMeans(n_clusters=n_clusters, init='k-means++', random_state=42)
working_data['Cluster'] = kmeans.fit_predict(X_scaled)

# Visualize clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(data=working_data, x='BILL_AMT', y='QTY', hue='Cluster', palette='viridis')
plt.title('Customer Segmentation (K-means Clustering)')
plt.xlabel('Total Transaction Amount')
plt.ylabel('Total Quantity')
plt.legend(title='Cluster')
plt.show()
```

OUTPUT:

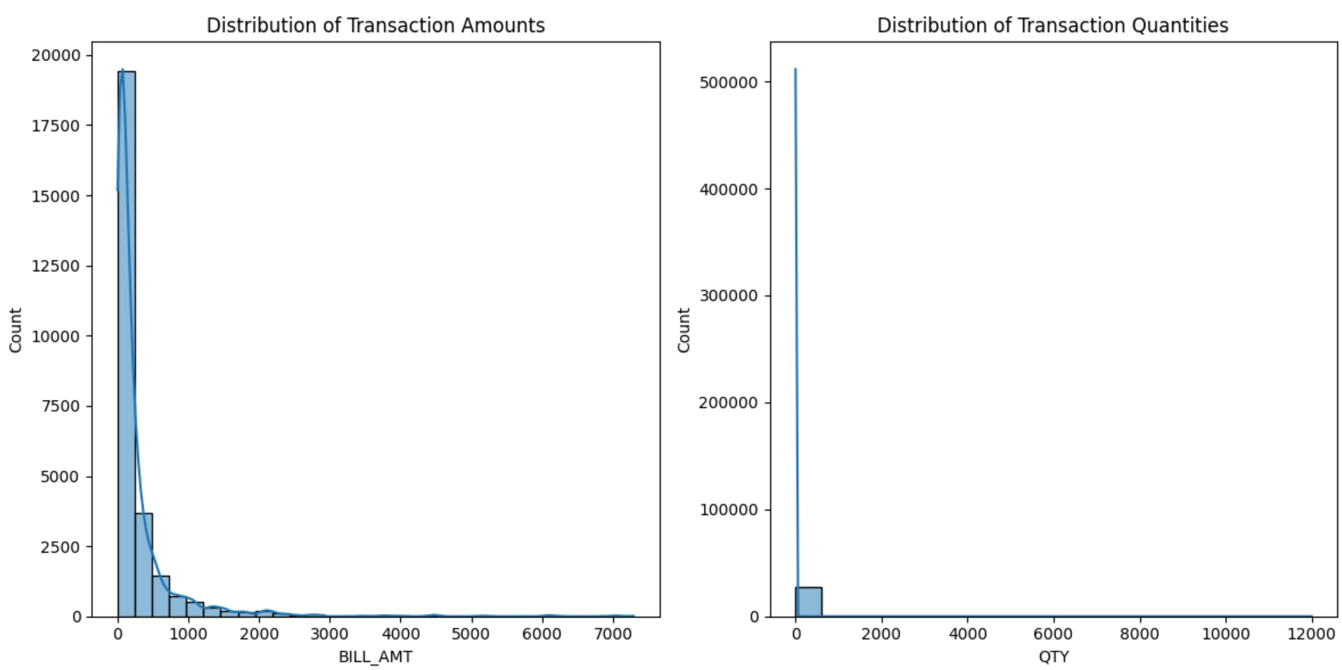


PREDICTIVE MODELING

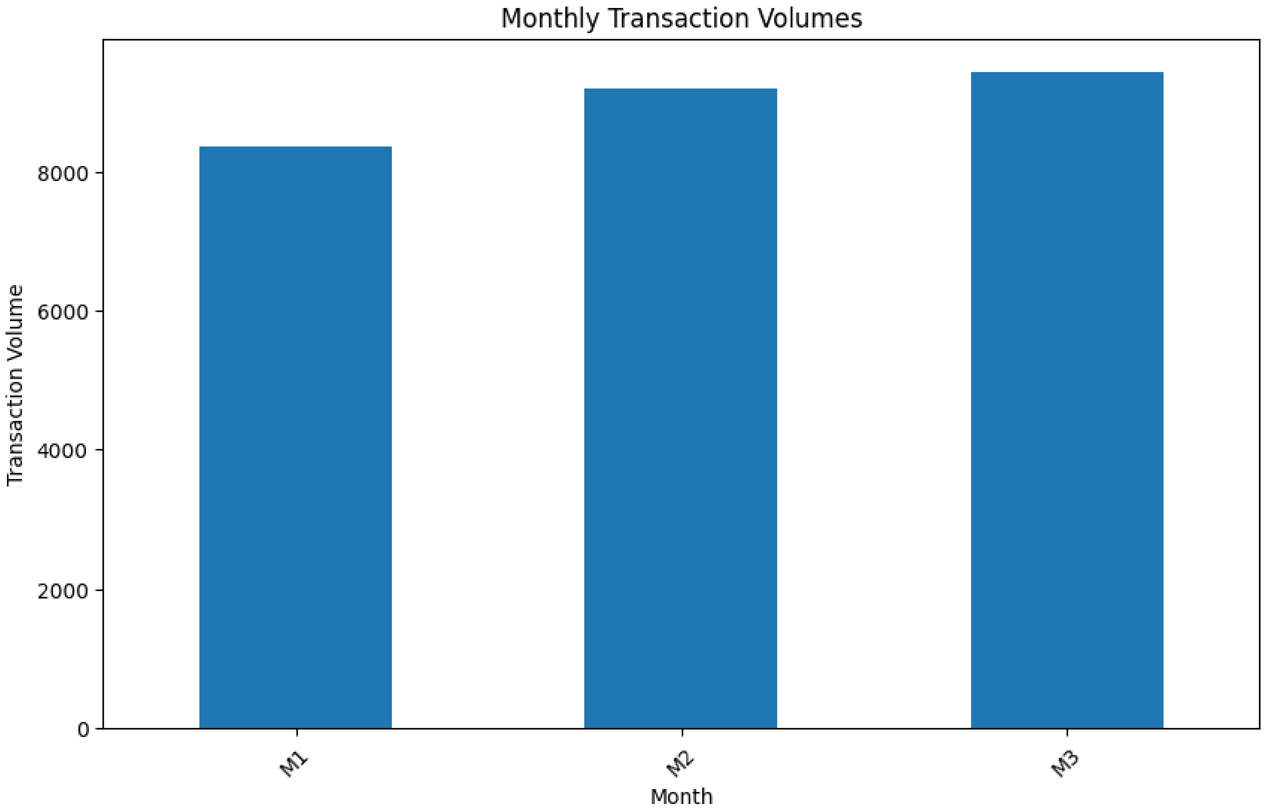
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
data = pd.read_csv("Hackathon_Working_Data.csv")
# Create histograms for transaction amounts and quantities
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(data['BILL_AMT'], bins=30, kde=True)
plt.title('Distribution of Transaction Amounts')
```

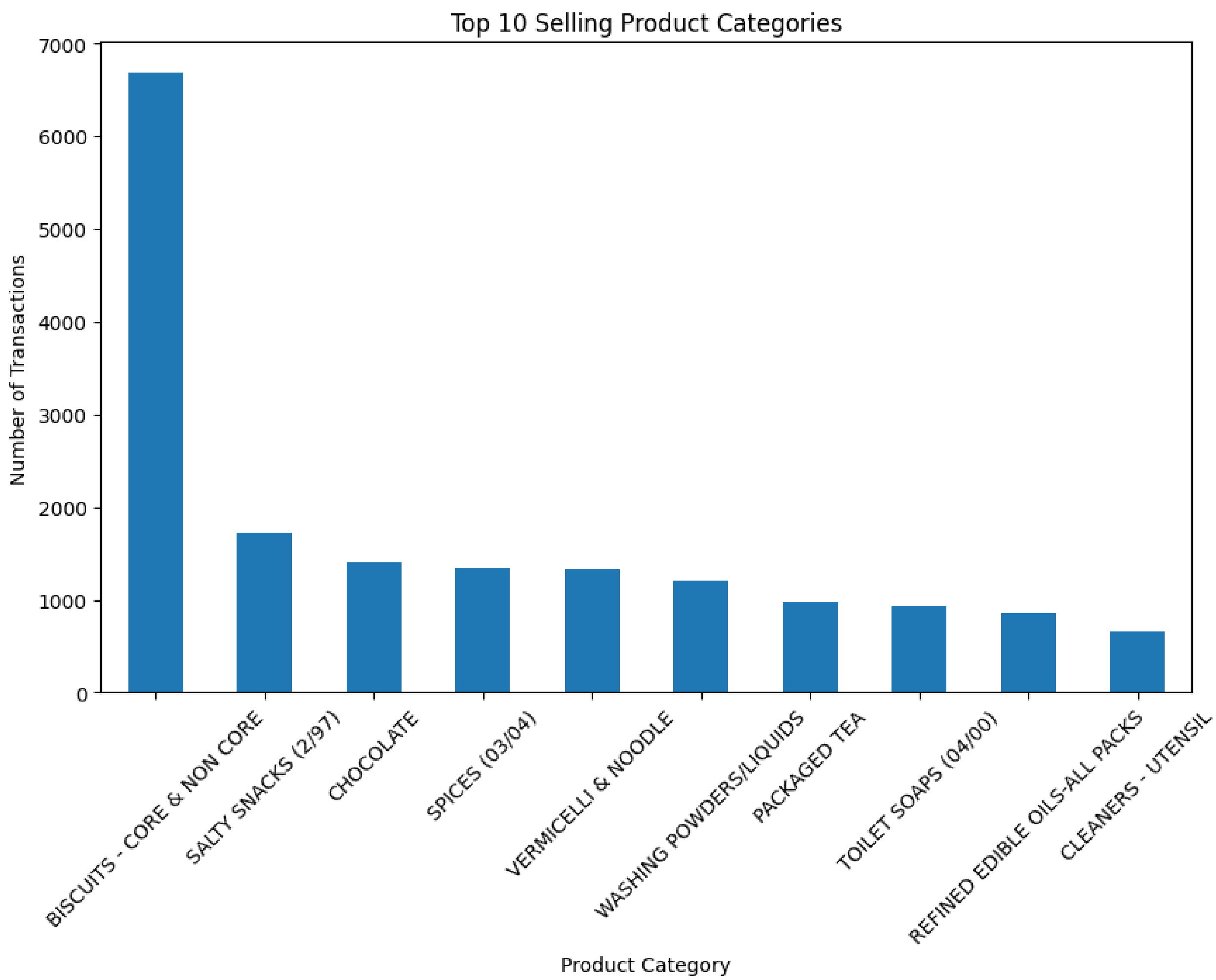
```python
plt.subplot(1, 2, 2)
sns.histplot(data['QTY'], bins=20, kde=True)
plt.title('Distribution of Transaction Quantities')

plt.tight_layout()
plt.show()

# Box plot for transaction amounts
plt.figure(figsize=(8, 6))
sns.boxplot(y=data['BILL_AMT'])
plt.title('Boxplot of Transaction Amounts')
plt.show()
# Assuming 'MONTH' column represents the month
monthly_sales = data.groupby('MONTH').size()
```

OUTPUT:

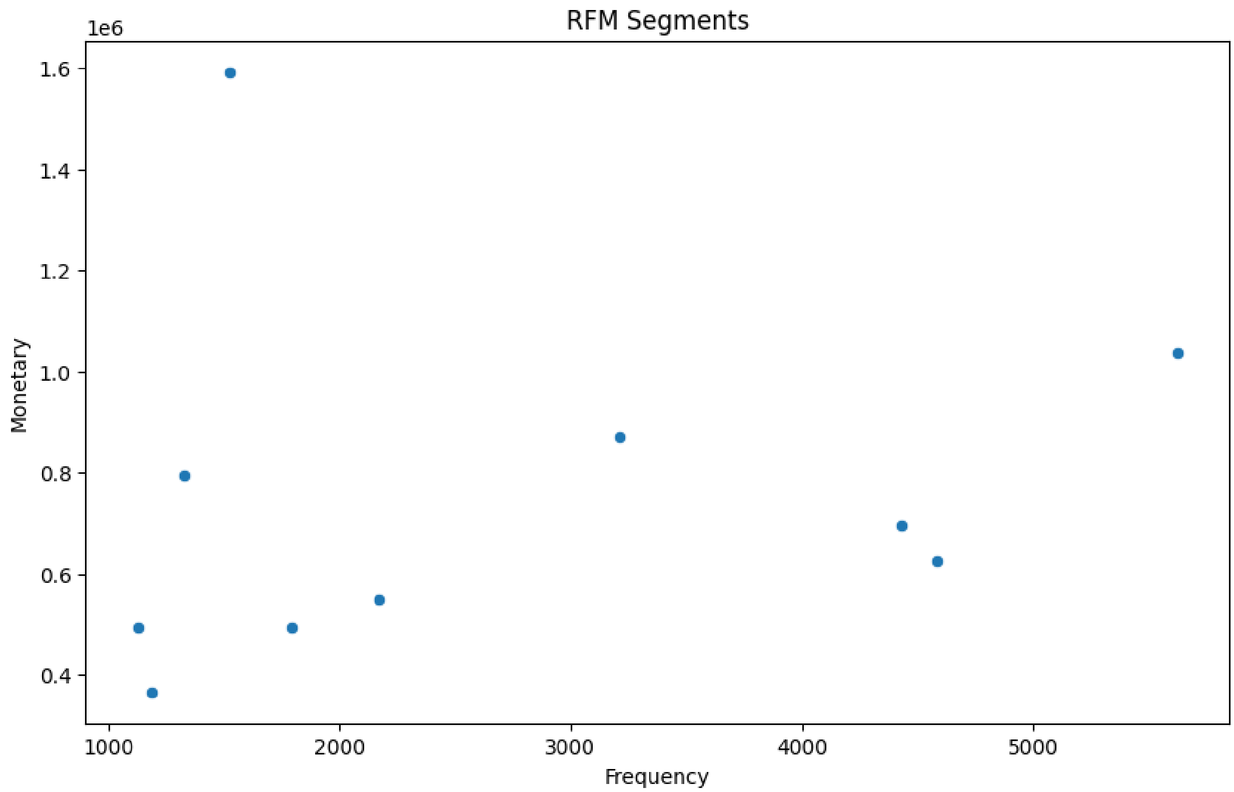


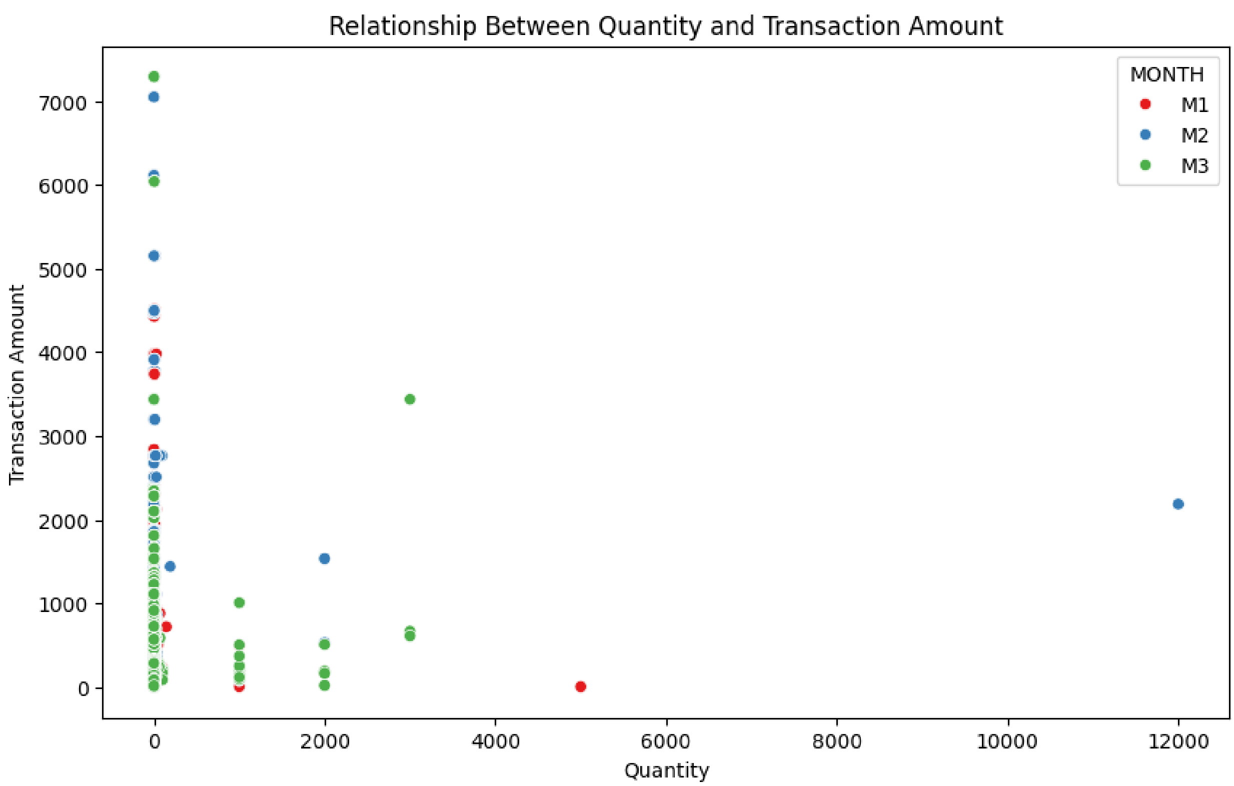

```python
plt.figure(figsize=(10, 6))
monthly_sales.plot(kind='bar')
plt.title('Monthly Transaction Volumes')
plt.xlabel('Month')
plt.ylabel('Transaction Volume')
plt.xticks(rotation=45)
plt.show()
# Assuming 'GRP' column represents product categories
top_categories = data['GRP'].value_counts().nlargest(10)

plt.figure(figsize=(10, 6))
top_categories.plot(kind='bar')
plt.title('Top 10 Selling Product Categories')
plt.xlabel('Product Category')
plt.ylabel('Number of Transactions')
plt.xticks(rotation=45)
plt.show()
# Conduct RFM analysis or clustering techniques
# Here, we'll use RFM analysis as an example
# Assuming 'STORECODE' represents unique customers
rfm_data = data.groupby('STORECODE').agg({
    'BILL_ID': 'count',            # Frequency
    'BILL_AMT': 'sum',             # Monetary
}).rename(columns={'BILL_ID': 'Frequency', 'BILL_AMT': 'Monetary'})
```


Monthly Transaction Volumes
Transaction Volume
0
2000
4000
6000
8000
M1
M2
M3
Month


Top 10 Selling Product Categories
Number of Transactions
0
1000
2000
3000
4000
5000
6000
7000
BISCUITS - CORE & NON CORE
SALTY SNACKS (2/97)
CHOCOLATE
SPICES (03/04)
VERMICELLI & NOODLE
WASHING POWDERS/LIQUIDS
PACKAGED TEA
TOILET SOAPS (04/00)
REFINED EDIBLE OILS-ALL PACKS
CLEANERS - UTENSIL
Product Category

```python
# Visualize RFM segments
plt.figure(figsize=(10, 6))
sns.scatterplot(data=rfm_data, x='Frequency', y='Monetary', palette='viridis')
plt.title('RFM Segments')
plt.xlabel('Frequency')
plt.ylabel('Monetary')
plt.show()

# Scatter plot to visualize relationship between quantity and transaction amount
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='QTY', y='BILL_AMT', hue='MONTH', palette='Set1')
plt.title('Relationship Between Quantity and Transaction Amount')
plt.xlabel('Quantity')
plt.ylabel('Transaction Amount')
plt.show()
```



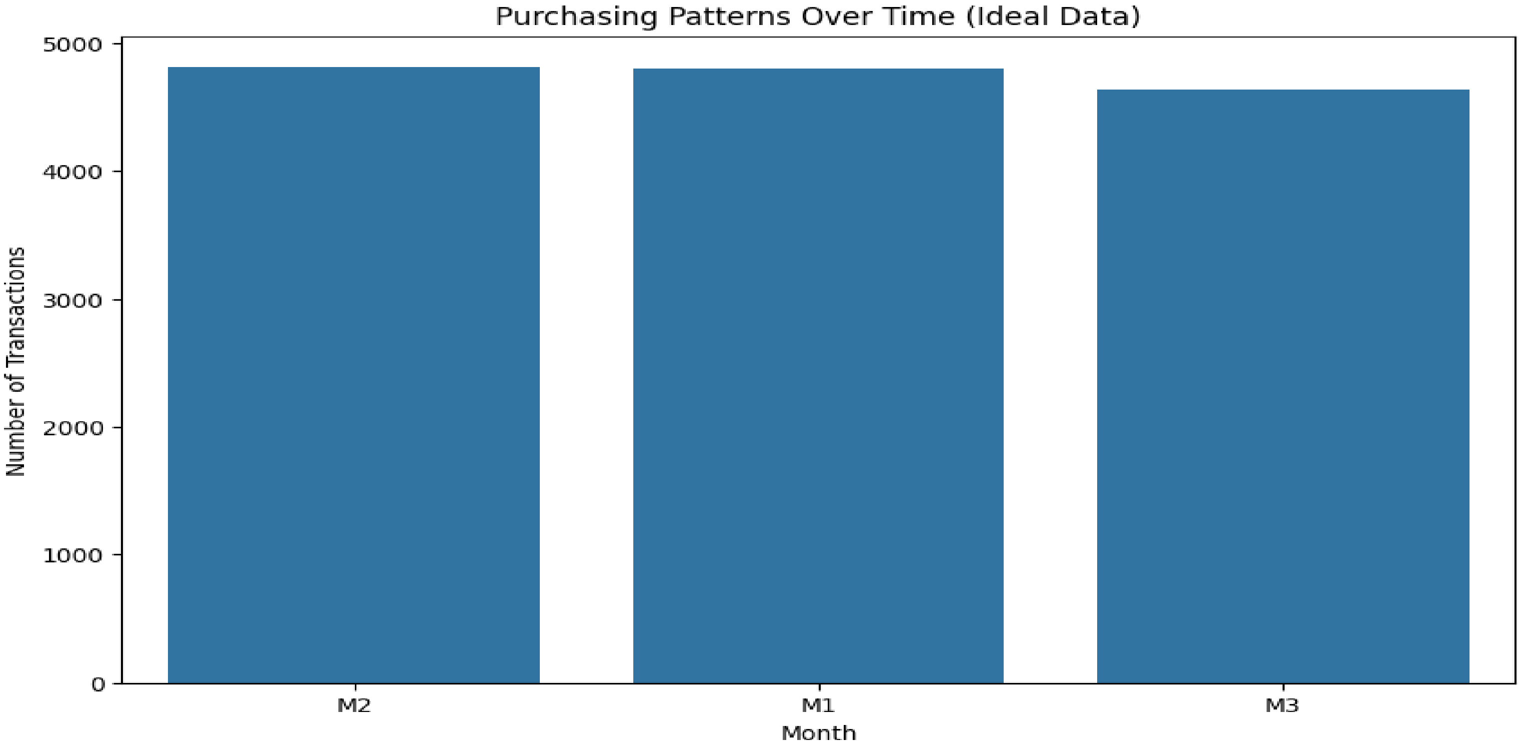


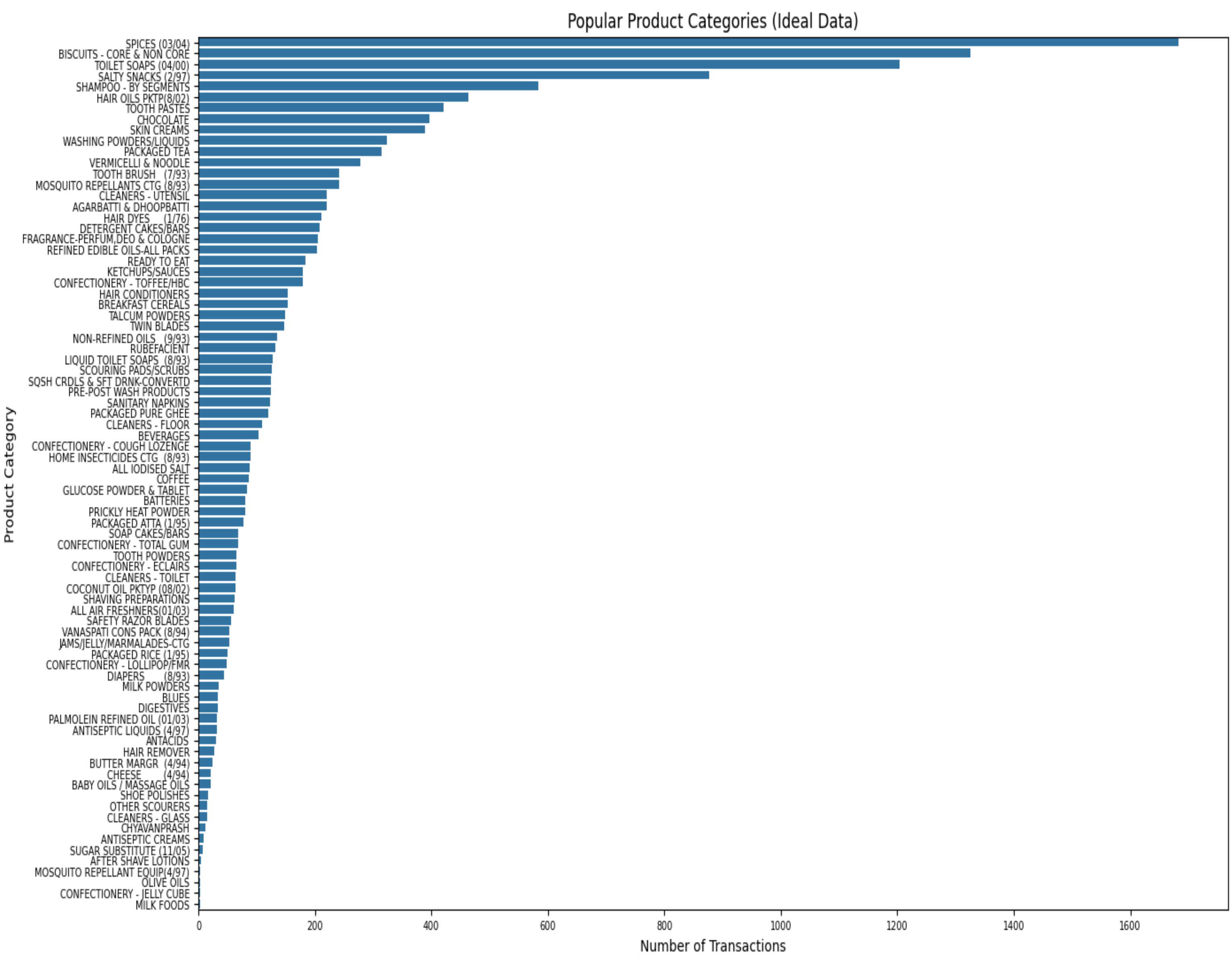
CUSTOMER BEHAVIOR ANALYSIS:

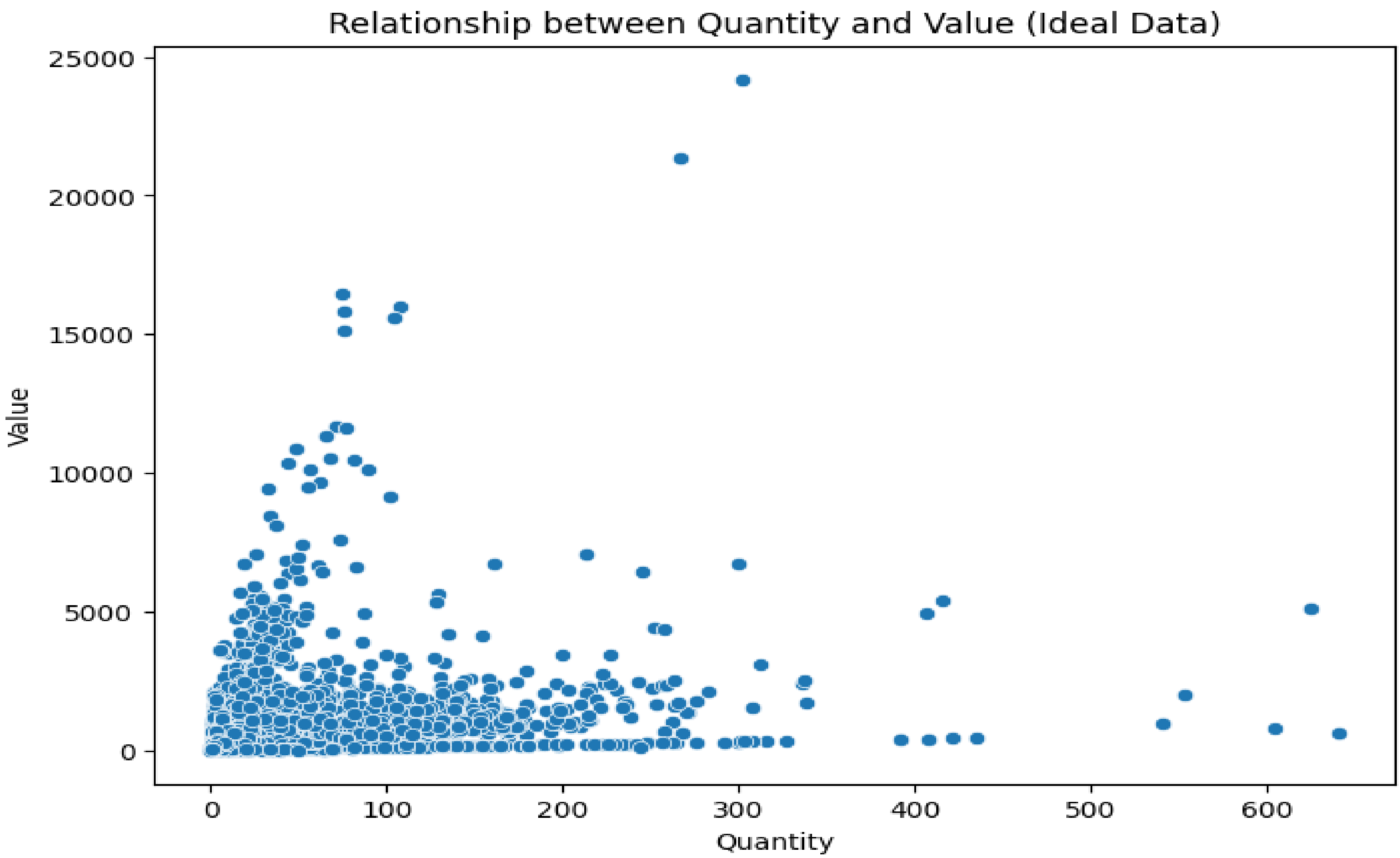
For ideal_data:

+ Code + Text

```
# For ideal_data
# Analyze purchasing patterns over time
plt.figure(figsize=(10, 6))
sns.countplot(data=ideal_data, x='MONTH', order=ideal_data['MONTH'].value_counts().index)
plt.title('Purchasing Patterns Over Time (Ideal Data)')
plt.xlabel('Month')
plt.ylabel('Number of Transactions')
plt.show()

# Identify popular product categories
plt.figure(figsize=(15, 10))
sns.countplot(data=ideal_data, y='GRP', order=ideal_data['GRP'].value_counts().index)
plt.title('Popular Product Categories (Ideal Data)')
plt.xlabel('Number of Transactions')
plt.ylabel('Product Category')
plt.xticks(fontsize=7)
plt.yticks(fontsize=7)
plt.show()

# Explore the relationship between quantity and value
plt.figure(figsize=(8, 6))
sns.scatterplot(data=ideal_data, x='QTY', y='VALUE')
plt.title('Relationship between Quantity and Value (Ideal Data)')
plt.xlabel('Quantity')
plt.ylabel('Value')
plt.show()

# Analyze the distribution of purchases across different stores
plt.figure(figsize=(10, 6))
sns.countplot(data=ideal_data, x='STORECODE', order=ideal_data['STORECODE'].value_counts().index)
plt.title('Distribution of Purchases Across Stores (Ideal Data)')
plt.xlabel('Store Code')
plt.ylabel('Number of Transactions')
plt.show()
```


Purchasing Patterns Over Time (Ideal Data)
Number of Transactions
5000
4000
3000
2000
1000
0
M2
M1
M3
Month


Popular Product Categories (Ideal Data)
Product Category
SPICES (03/04)
BISCUITS - CORE & NON CORE
TOILET SOAPS (04/00)
SALTY SNACKS (2/97)
SHAMPOO - BY SEGMENTS
HAIR OILS PKTP(8/02)
TOOTH PASTES
CHOCOLATE
SKIN CREAMS
WASHING POWDERS/LIQUIDS
PACKAGED TEA
VERMICELLI & NOODLE
TOOTH BRUSH (7/93)
MOSQUITO REPELLANTS CTG (8/93)
CLEANERS - UTENSIL
AGARBATTI & DHOOPBATTI
HAIR DYES (1/76)
DETERGENT CAKES/BARS
FRAGRANCE-PERFUM,DEO & COLOGNE
REFINED EDIBLE OILS-ALL PACKS
READY TO EAT
KETCHUPS/SAUCES
CONFECTIONERY - TOFFEE/HBC
HAIR CONDITIONERS
BREAKFAST CEREALS
TALCUM POWDERS
TWIN BLADES
NON-REFINED OILS (9/93)
RUBEFACIENT
LIQUID TOILET SOAPS (8/93)
SCOURING PADS/SCRUBS
SQSH CRDLS & SFT DRNK-CONVERTD
PRE-POST WASH PRODUCTS
SANITARY NAPKINS
PACKAGED PURE GHEE
CLEANERS - FLOOR
BEVERAGES
CONFECTIONERY - COUGH LOZENGE
HOME INSECTICIDES CTG (8/93)
ALL IODISED SALT
COFFEE
GLUCOSE POWDER & TABLET
BATTERIES
PRICKLY HEAT POWDER
PACKAGED ATTA (1/95)
SOAP CAKES/BARS
CONFECTIONERY - TOTAL GUM
TOOTH POWDERS
CONFECTIONERY - ECLAIRS
CLEANERS - TOILET
COCONUT OIL PKTYP (08/02)
SHAVING PREPARATIONS
ALL AIR FRESHNERS(01/03)
SAFETY RAZOR BLADES
VANASPATI CONS PACK (8/94)
JAMS/JELLY/MARMALADES-CTG
PACKAGED RICE (1/95)
CONFECTIONERY - LOLLIPOP/FMR
DIAPERS (8/93)
MILK POWDERS
BLUES
DIGESTIVES
PALMOLEIN REFINED OIL (01/03)
ANTISEPTIC LIQUIDS (4/97)
ANTACIDS
HAIR REMOVER
BUTTER MARGR (4/94)
CHEESE (4/94)
BABY OILS / MASSAGE OILS
SHOE POLISHES
OTHER SCOURERS
CLEANERS - GLASS
CHYAVANPRASH
ANTISEPTIC CREAMS
SUGAR SUBSTITUTE (11/05)
AFTER SHAVE LOTIONS
MOSQUITO REPELLANT EQUIP(4/97)
OLIVE OILS
CONFECTIONERY - JELLY CUBE
MILK FOODS
0
200
400
600
800
1000
1200
1400
1600
Number of Transactions

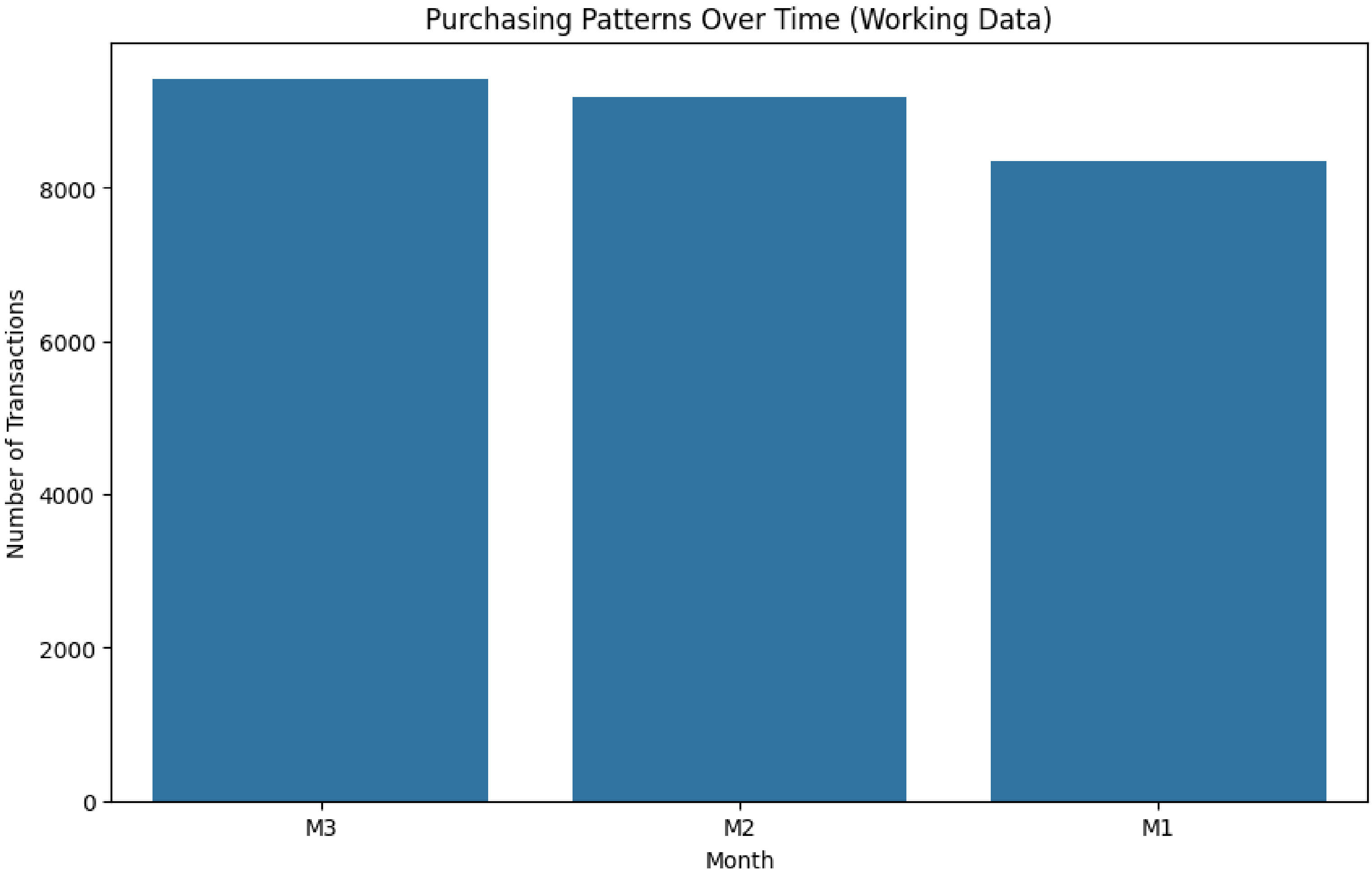


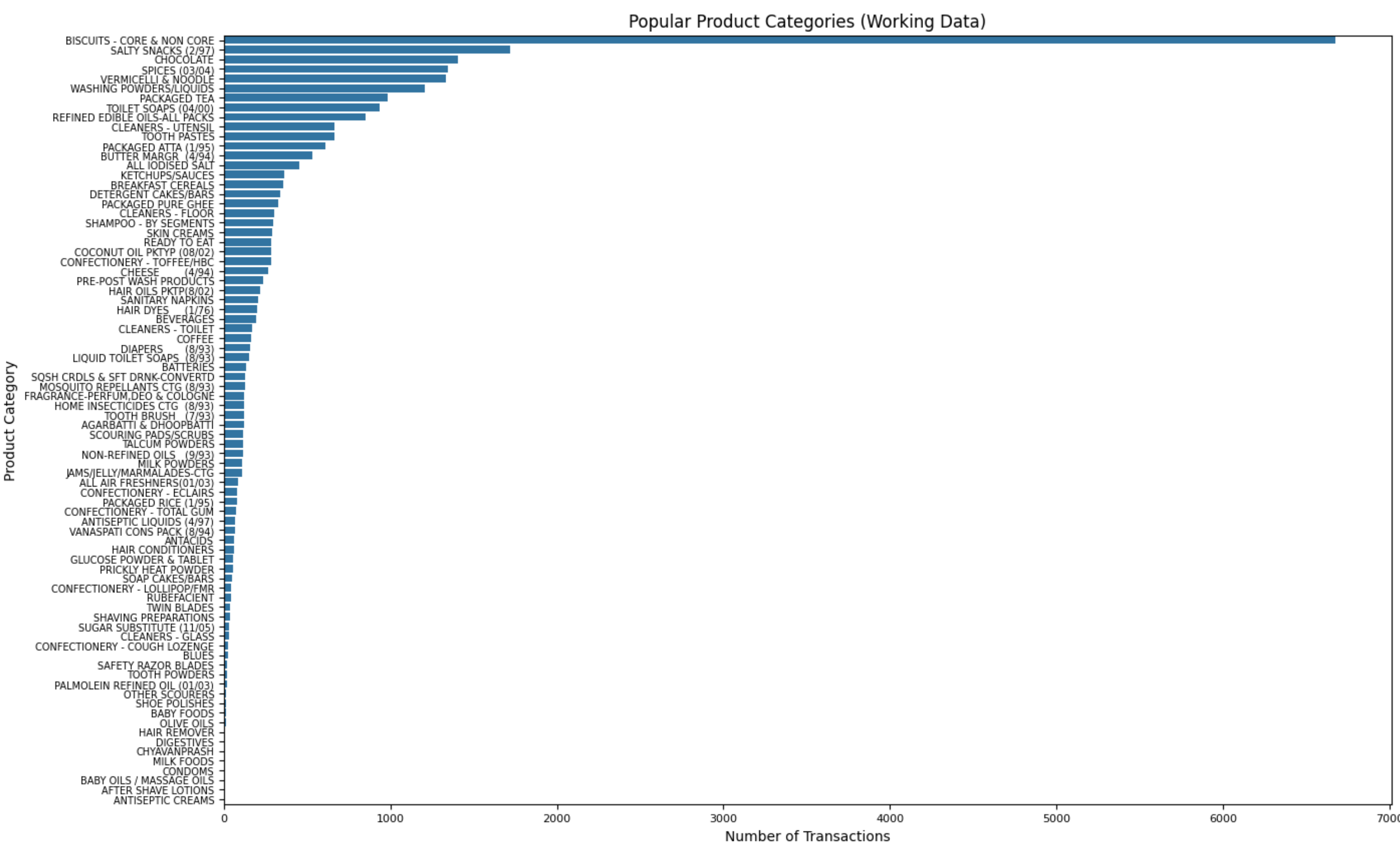
Relationship between Quantity and Value (Ideal Data)
Value
25000
20000
15000
10000
5000
0
0
100
200
300
400
500
600
Quantity

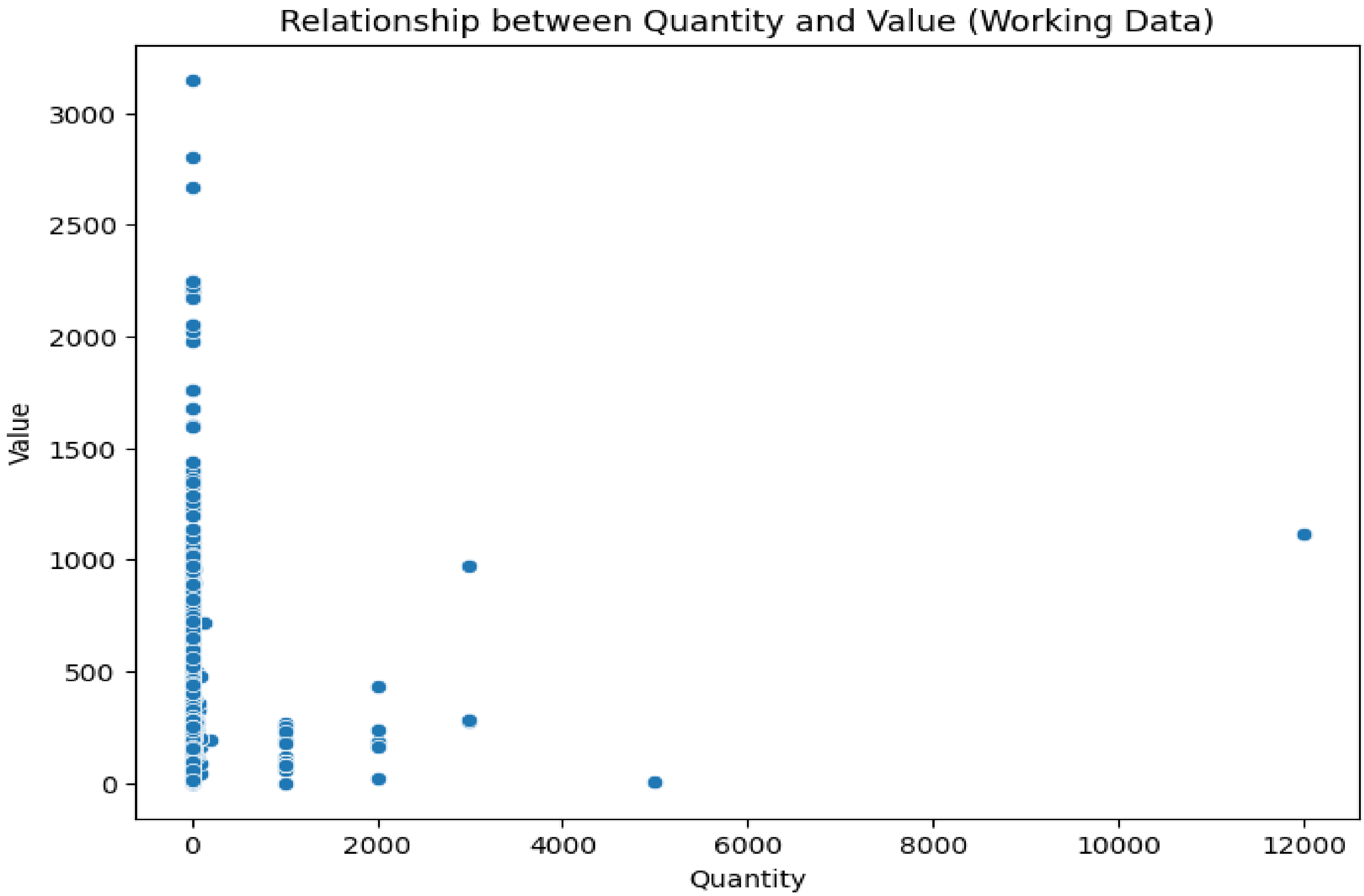
Distribution of Purchases Across Stores (Ideal Data)
Number of Transactions
2500
2000
1500
1000
500
0
P8
P6
P10
P2
P4
P3
P9
P1
P7
P5
Store Code

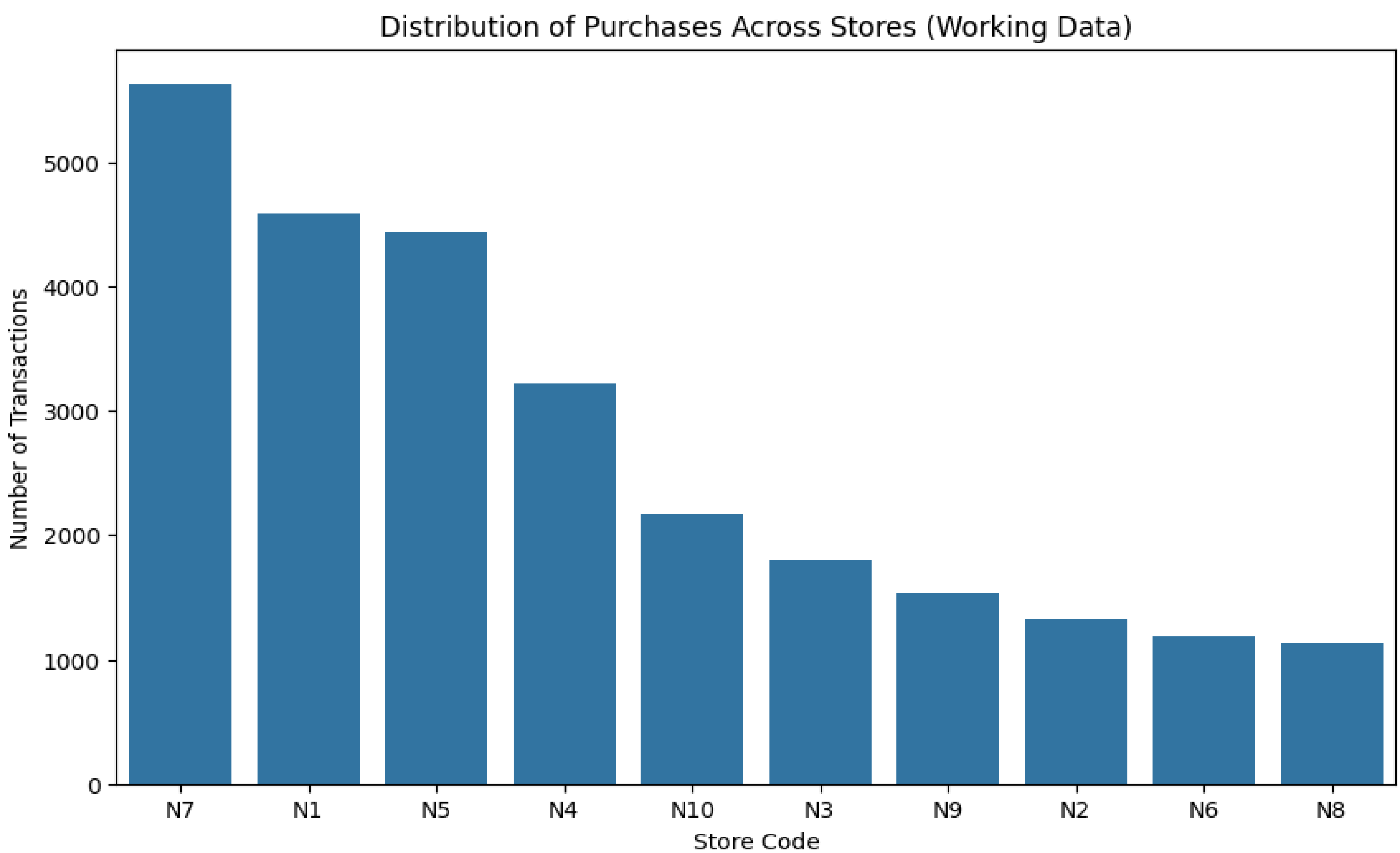
For Working_data

```
# For working_data
# Analyze purchasing patterns over time
plt.figure(figsize=(10, 6))
sns.countplot(data=working_data, x='MONTH', order=working_data['MONTH'].value_counts().index)
plt.title('Purchasing Patterns Over Time (Working Data)')
plt.xlabel('Month')
plt.ylabel('Number of Transactions')
plt.show()

# Identify popular product categories
plt.figure(figsize=(15, 10))
sns.countplot(data=working_data, y='GRP', order=working_data['GRP'].value_counts().index)
plt.title('Popular Product Categories (Working Data)')
plt.xlabel('Number of Transactions')
plt.ylabel('Product Category')
plt.xticks(fontsize=8)
plt.yticks(fontsize=7)
plt.show()

# Explore the relationship between quantity and value
plt.figure(figsize=(8, 6))
sns.scatterplot(data=working_data, x='QTY', y='VALUE')
plt.title('Relationship between Quantity and Value (Working Data)')
plt.xlabel('Quantity')
plt.ylabel('Value')
plt.show()

# Analyze the distribution of purchases across different stores
plt.figure(figsize=(10, 6))
sns.countplot(data=working_data, x='STORECODE', order=working_data['STORECODE'].value_counts().index)
plt.title('Distribution of Purchases Across Stores (Working Data)')
plt.xlabel('Store Code')
plt.ylabel('Number of Transactions')
plt.show()
```


Purchasing Patterns Over Time (Working Data)
Number of Transactions
0
2000
4000
6000
8000
M3
M2
M1
Month


Popular Product Categories (Working Data)
Product Category
BISCUITS - CORE & NON CORE
SALTY SNACKS (2/97)
CHOCOLATE
SPICES (03/04)
VERMICELLI & NOODLE
WASHING POWDERS/LIQUIDS
PACKAGED TEA
TOILET SOAPS (04/00)
REFINED EDIBLE OILS-ALL PACKS
CLEANERS - UTENSIL
TOOTH PASTES
PACKAGED ATTA (1/95)
BUTTER MARGR (4/94)
ALL IODISED SALT
KETCHUPS/SAUCES
BREAKFAST CEREALS
DETERGENT CAKES/BARS
PACKAGED PURE GHEE
CLEANERS - FLOOR
SHAMPOO - BY SEGMENTS
SKIN CREAMS
READY TO EAT
COCONUT OIL PKTYP (08/02)
CONFECTIONERY - TOFFEE/HBC
CHEESE (4/94)
PRE-POST WASH PRODUCTS
HAIR OILS PKTP(8/02)
SANITARY NAPKINS
HAIR DYES (1/76)
BEVERAGES
CLEANERS - TOILET
COFFEE
DIAPERS (8/93)
LIQUID TOILET SOAPS (8/93)
BATTERIES
SQSH CRDLS & SFT DRNK-CONVERTD
MOSQUITO REPELLANTS CTG (8/93)
FRAGRANCE-PERFUM,DEO & COLOGNE
HOME INSECTICIDES CTG (8/93)
TOOTH BRUSH (7/93)
AGARBATTI & DHOOPBATTI
SCOURING PADS/SCRUBS
TALCUM POWDERS
NON-REFINED OILS (9/93)
MILK POWDERS
JAMS/JELLY/MARMALADES-CTG
ALL AIR FRESHNERS(01/03)
CONFECTIONERY - ECLAIRS
PACKAGED RICE (1/95)
CONFECTIONERY - TOTAL GUM
ANTISEPTIC LIQUIDS (4/97)
VANASPATI CONS PACK (8/94)
ANTACIDS
HAIR CONDITIONERS
GLUCOSE POWDER & TABLET
PRICKLY HEAT POWDER
SOAP CAKES/BARS
CONFECTIONERY - LOLLIPOP/FMR
RUBEFACIENT
TWIN BLADES
SHAVING PREPARATIONS
SUGAR SUBSTITUTE (11/05)
CLEANERS - GLASS
CONFECTIONERY - COUGH LOZENGE
BLUES
SAFETY RAZOR BLADES
TOOTH POWDERS
PALMOLEIN REFINED OIL (01/03)
OTHER SCOURERS
SHOE POLISHES
BABY FOODS
OLIVE OILS
HAIR REMOVER
DIGESTIVES
CHYAVANPRASH
MILK FOODS
CONDOMS
BABY OILS / MASSAGE OILS
AFTER SHAVE LOTIONS
ANTISEPTIC CREAMS
0
1000
2000
3000
4000
5000
6000
7000
Number of Transactions

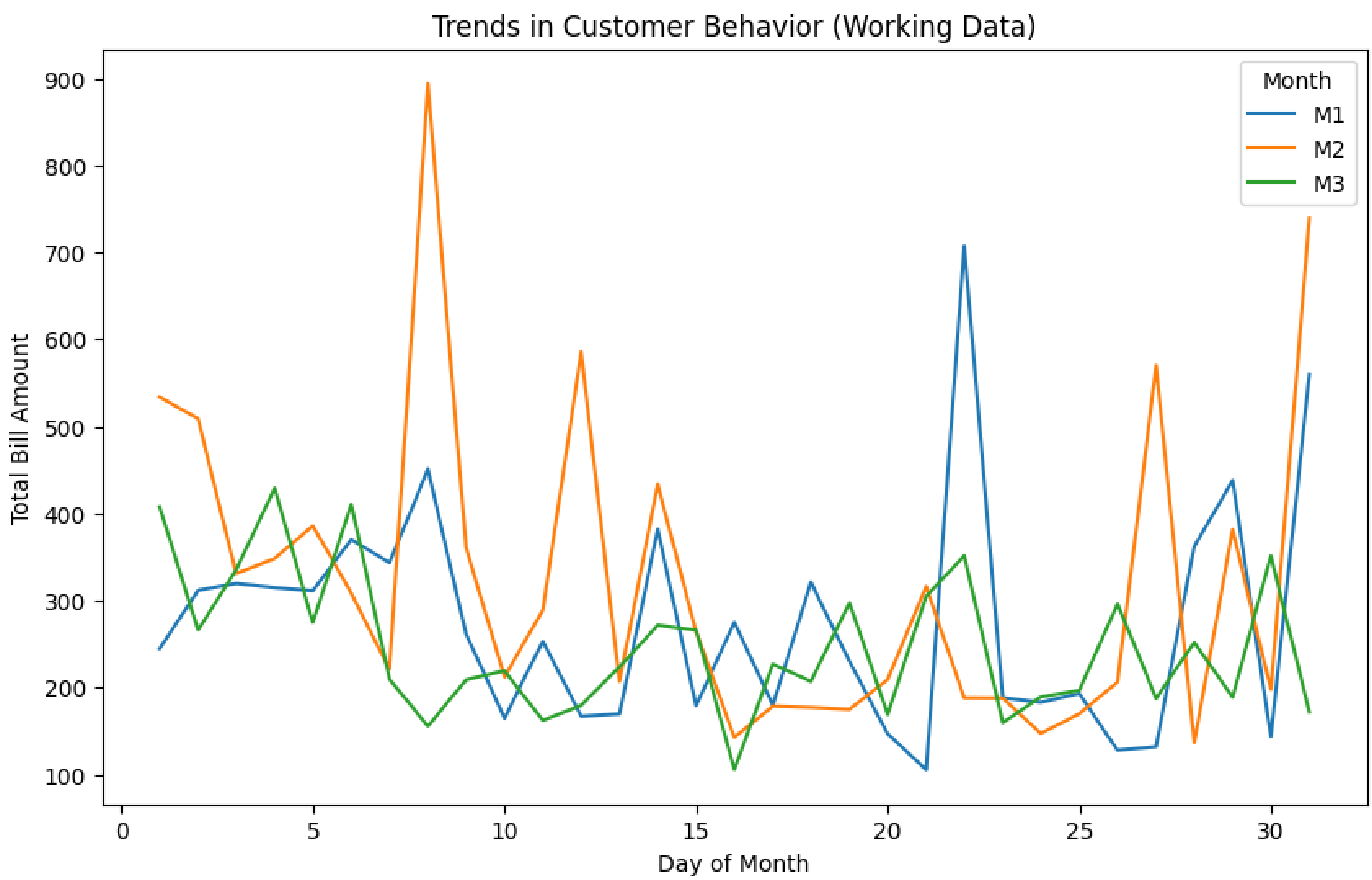

Relationship between Quantity and Value (Working Data)
Value
Quantity
0
500
1000
1500
2000
2500
3000
0
2000
4000
6000
8000
10000
12000


Distribution of Purchases Across Stores (Working Data)
Number of Transactions
Store Code
0
1000
2000
3000
4000
5000
N7
N1
N5
N4
N10
N3
N9
N2
N6
N8

VISUALIZATION AND REPORTING:

Customer Behavior:

```
plt.figure(figsize=(10, 6))
sns.lineplot(data=working_data, x='DAY', y='BILL_AMT', hue='MONTH', ci=None)
plt.title('Trends in Customer Behavior (Working Data)')
plt.xlabel('Day of Month')
plt.ylabel('Total Bill Amount')
plt.legend(title='Month')
plt.show()
```



Conclusion :

In conclusion, the analysis of customer behavior based on transactional data has provided valuable insights into various aspects of customer preferences, purchasing patterns, and overall behavior. Through exploratory data analysis (EDA), we have uncovered key trends, such as the distribution of sales across different stores, brands, and categories, as well as fluctuations in sales over time. By segmenting customers based on their purchasing behavior, we have identified distinct customer groups and their unique preferences.

Moreover, predictive modeling has allowed us to forecast total sales accurately, enabling better decision-making for inventory management and marketing strategies. These insights can be leveraged to tailor marketing campaigns, optimize product offerings, and enhance customer experience, ultimately driving business growth and profitability.

Moving forward, continuous monitoring of customer behavior and refinement of analytical techniques will be crucial for staying responsive to evolving market dynamics and customer needs. By adopting a data-driven approach, businesses can adapt proactively to changes in consumer behavior and maintain a competitive edge in the market.