Supermarket Customers Data Analysis

Capstone Project - Hendry Hansel Kurniawan Lessy

Background

Supermarkets compete to increase customer satisfaction and retain loyalty through personalized services and targeted marketing strategies. By analyzing customer purchasing data and responses to promotions, the company can gain insight into customer behavior patterns. This analysis aims to uncover key trends that will assist in designing more effective marketing strategies and improving customer experience.

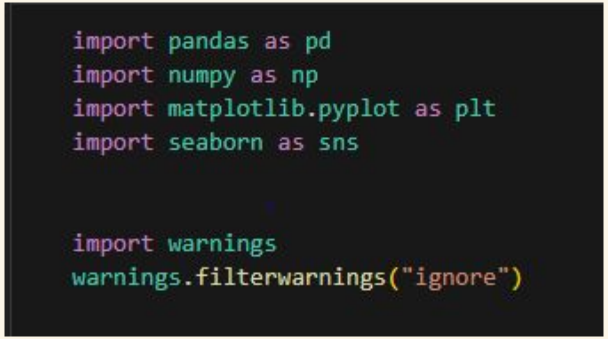
Problem Statement

Main Question:
What is the biggest factor influencing customer spending in supermarkets, and how can the company enhance customer loyalty through more effective marketing strategies?

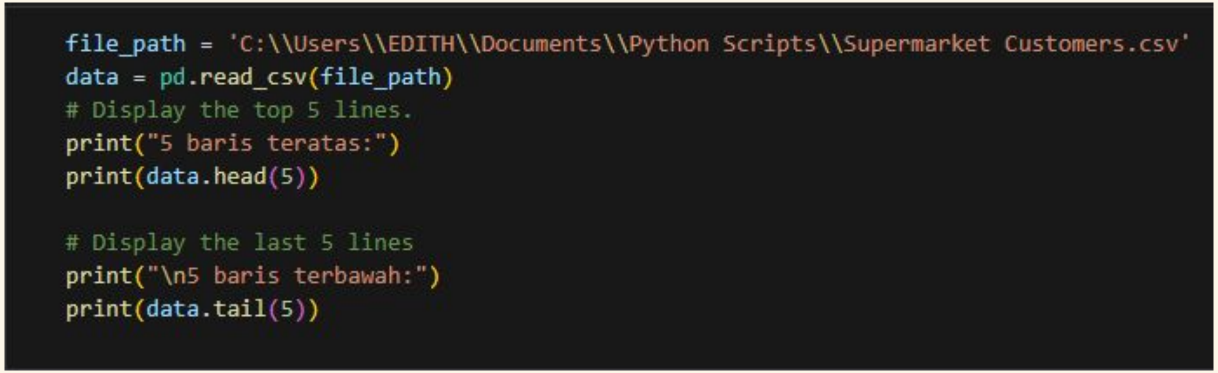
Sub-Question:

Are there spending patterns based on demographics, such as age, marital status, or number of children?
What is the average spending of customers on specific product categories (meat, wine, fish, etc.)?
Are customers who receive regular marketing campaigns more likely to make a purchase?

Data Understanding

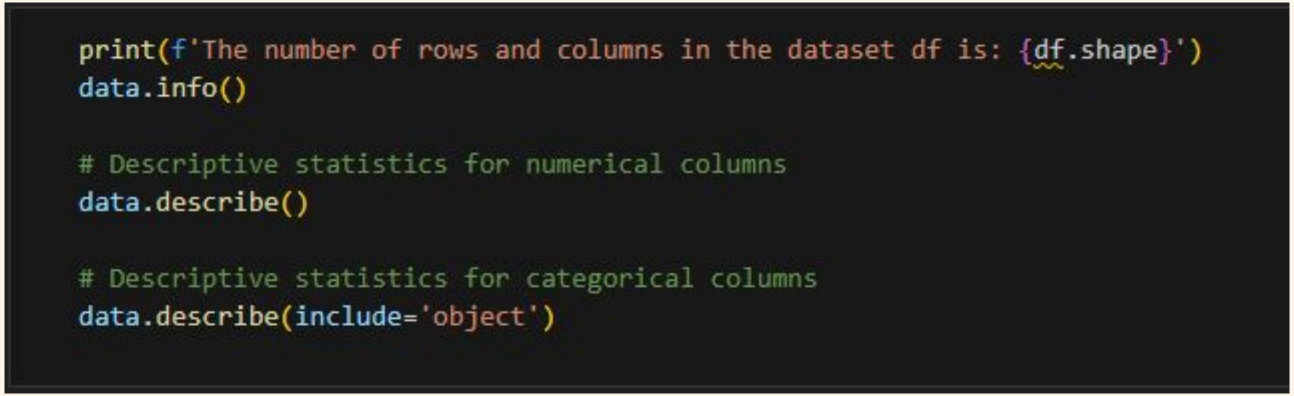


```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


import warnings
warnings.filterwarnings("ignore")
```



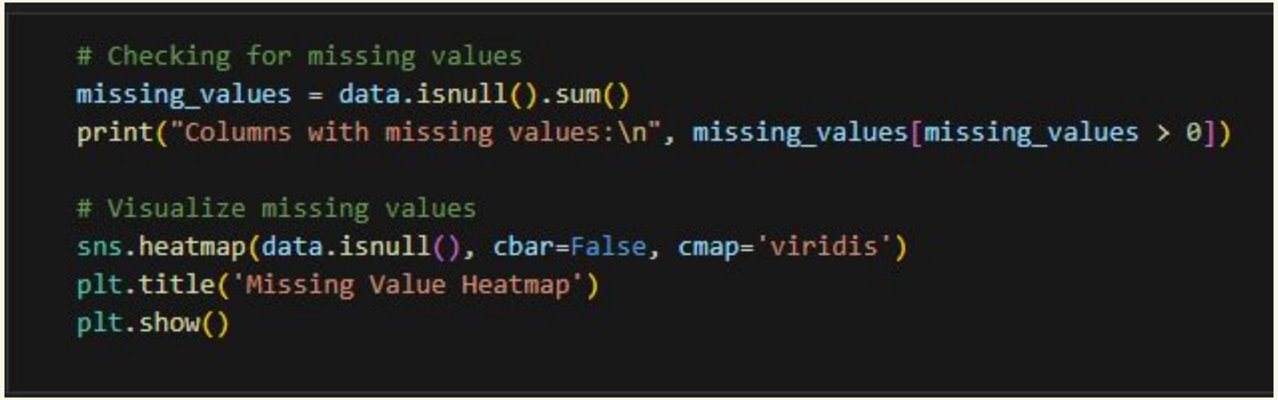
```python
file_path = 'C:\\Users\\EDITH\\Documents\\Python Scripts\\Supermarket Customers.csv'
data = pd.read_csv(file_path)
# Display the top 5 lines.
print("5 baris teratas:")
print(data.head(5))

# Display the last 5 lines
print("\n5 baris terbawah:")
print(data.tail(5))
```

Data Understanding



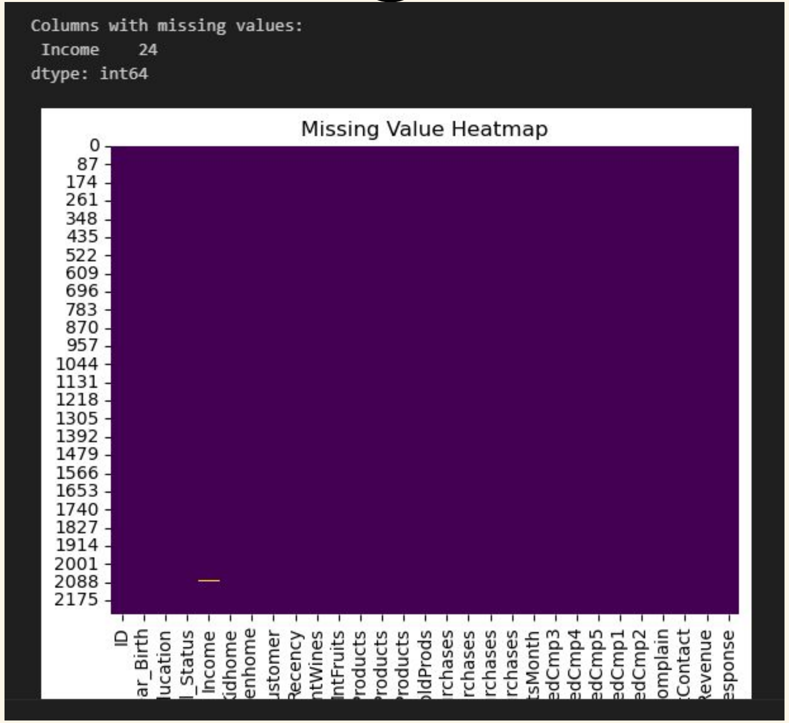
```
print(f'The number of rows and columns in the dataset df is: {df.shape}')
data.info()

# Descriptive statistics for numerical columns
data.describe()

# Descriptive statistics for categorical columns
data.describe(include='object')
```

Missing value



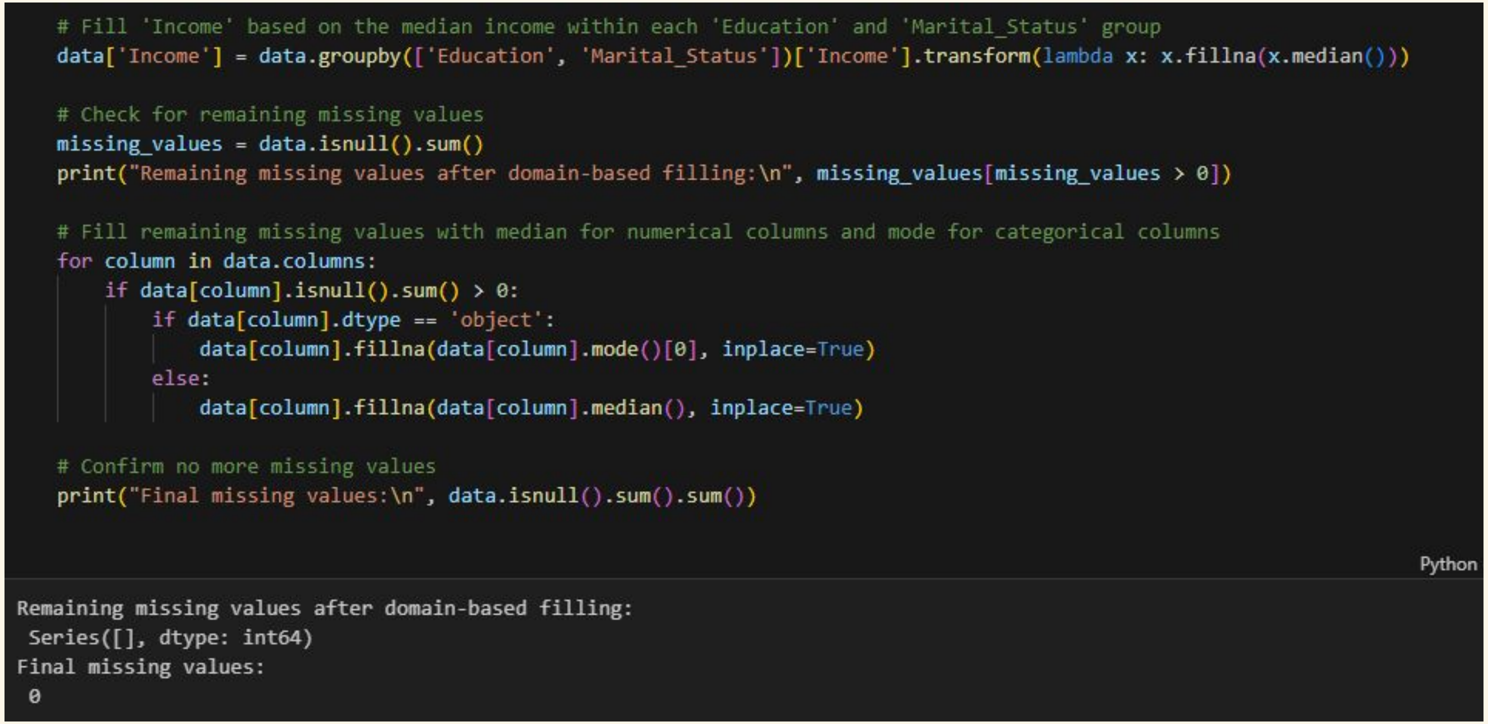
```
# Checking for missing values
missing_values = data.isnull().sum()
print("Columns with missing values:\n", missing_values[missing_values > 0])

# Visualize missing values
sns.heatmap(data.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Value Heatmap')
plt.show()
```

Missing Value



```
Columns with missing values:
 Income    24
dtype: int64
```

Handling Missing Value

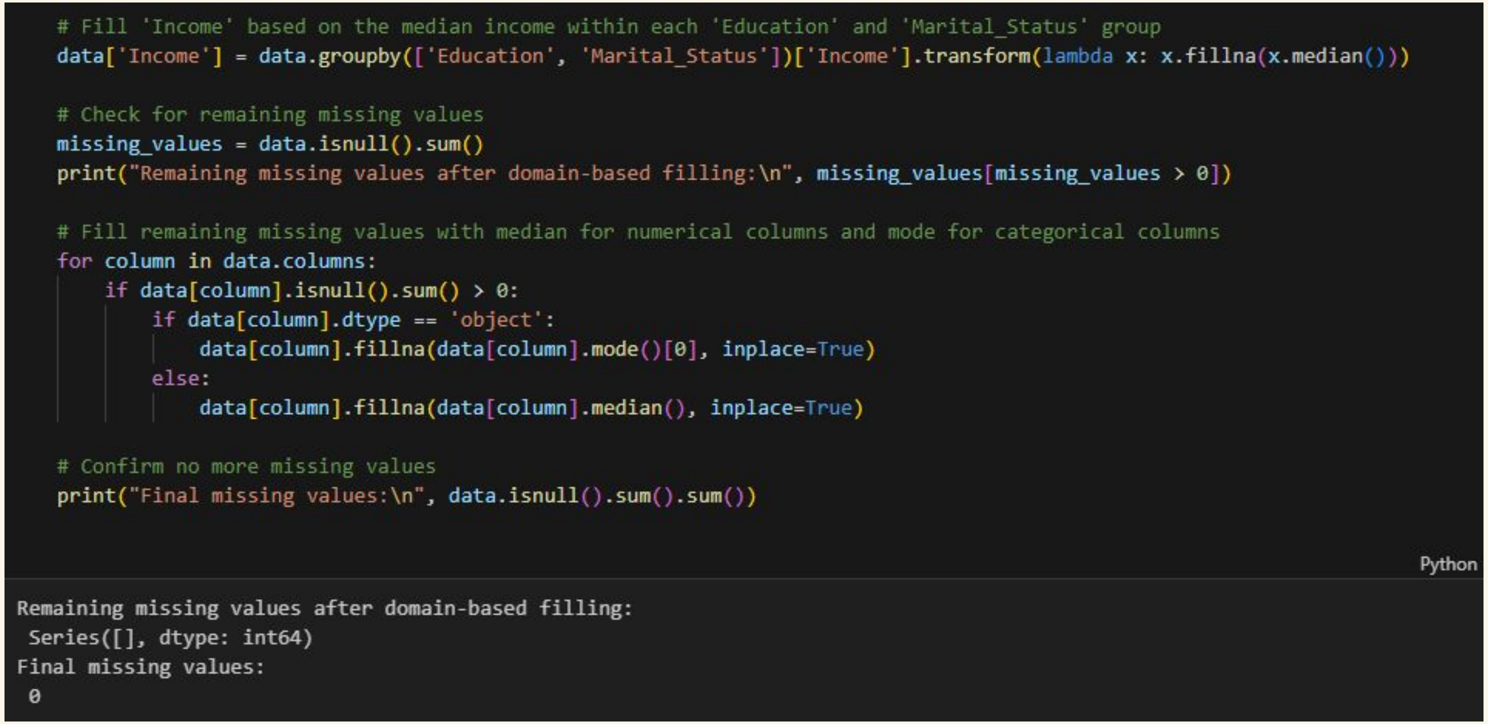


```python
# Fill 'Income' based on the median income within each 'Education' and 'Marital_Status' group
data['Income'] = data.groupby(['Education', 'Marital_Status'])['Income'].transform(lambda x: x.fillna(x.median()))

# Check for remaining missing values
missing_values = data.isnull().sum()
print("Remaining missing values after domain-based filling:\n", missing_values[missing_values > 0])

# Fill remaining missing values with median for numerical columns and mode for categorical columns
for column in data.columns:
    if data[column].isnull().sum() > 0:
        if data[column].dtype == 'object':
            data[column].fillna(data[column].mode()[0], inplace=True)
        else:
            data[column].fillna(data[column].median(), inplace=True)

# Confirm no more missing values
print("Final missing values:\n", data.isnull().sum().sum())
```

Python

```
Remaining missing values after domain-based filling:
 Series([], dtype: int64)
Final missing values:
 0
```

Handling Missing Value

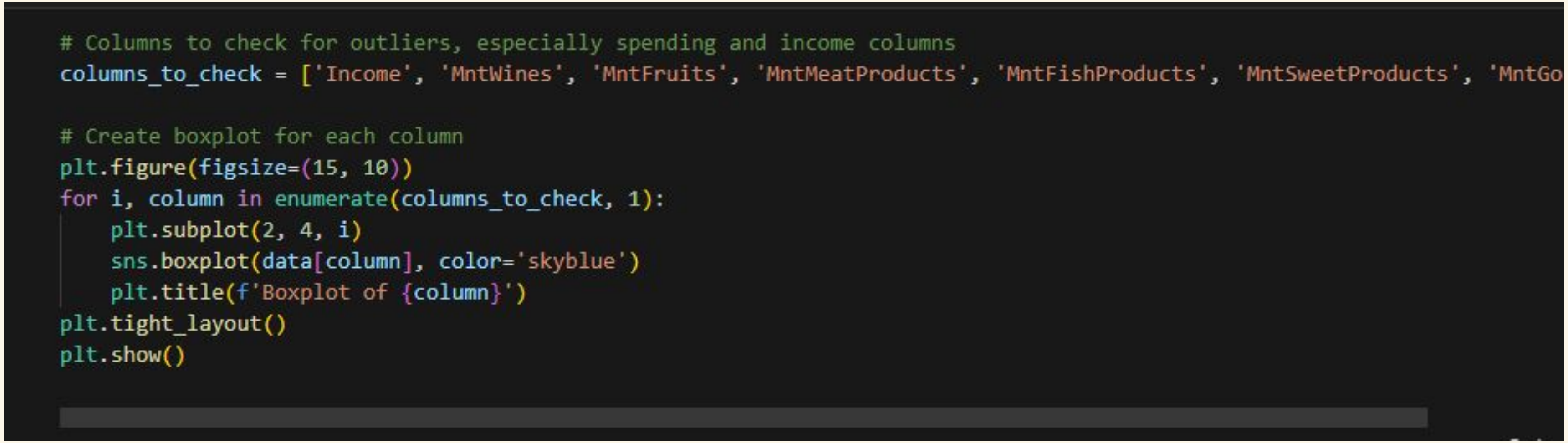


```python
# Fill 'Income' based on the median income within each 'Education' and 'Marital_Status' group
data['Income'] = data.groupby(['Education', 'Marital_Status'])['Income'].transform(lambda x: x.fillna(x.median()))

# Check for remaining missing values
missing_values = data.isnull().sum()
print("Remaining missing values after domain-based filling:\n", missing_values[missing_values > 0])

# Fill remaining missing values with median for numerical columns and mode for categorical columns
for column in data.columns:
    if data[column].isnull().sum() > 0:
        if data[column].dtype == 'object':
            data[column].fillna(data[column].mode()[0], inplace=True)
        else:
            data[column].fillna(data[column].median(), inplace=True)

# Confirm no more missing values
print("Final missing values:\n", data.isnull().sum().sum())
```

Python

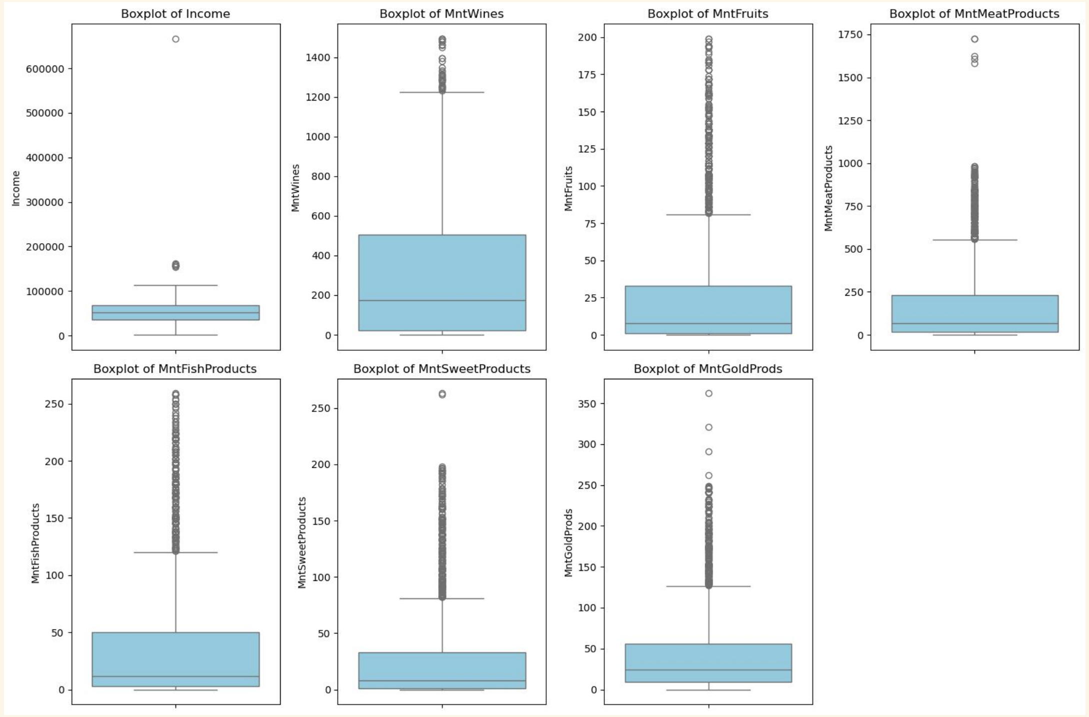
```
Remaining missing values after domain-based filling:
 Series([], dtype: int64)
Final missing values:
 0
```

Outlier Control

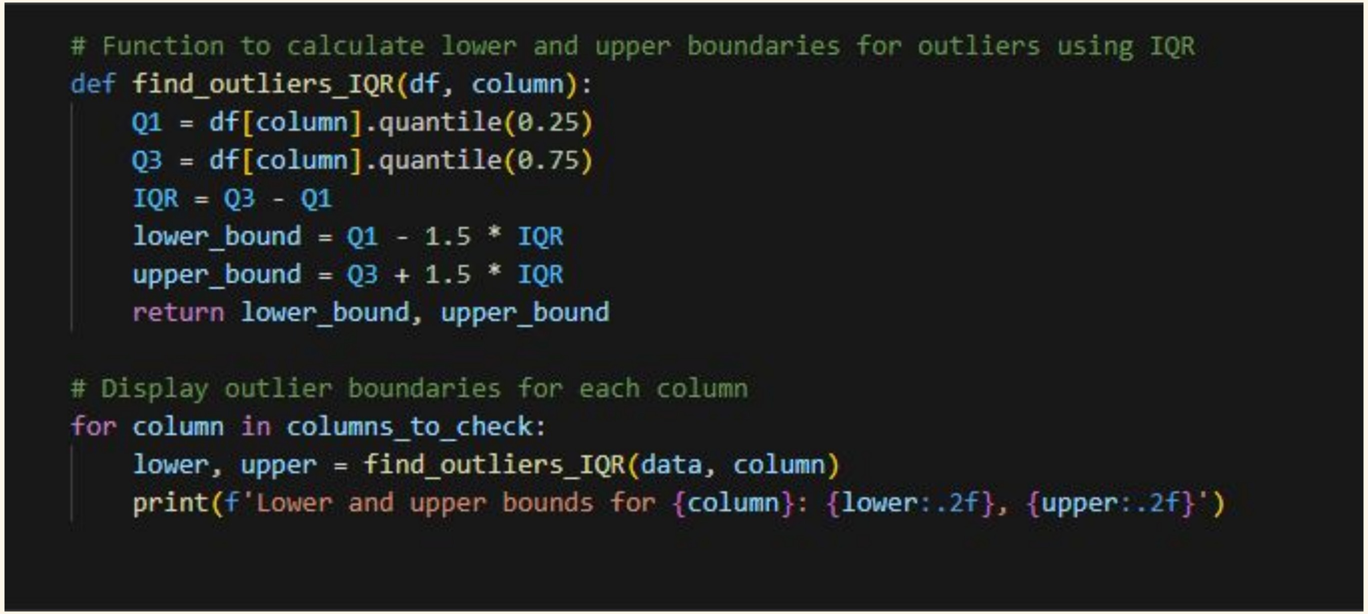


```python
# Columns to check for outliers, especially spending and income columns
columns_to_check = ['Income', 'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGo

# Create boxplot for each column
plt.figure(figsize=(15, 10))
for i, column in enumerate(columns_to_check, 1):
    plt.subplot(2, 4, i)
    sns.boxplot(data[column], color='skyblue')
    plt.title(f'Boxplot of {column}')
plt.tight_layout()
plt.show()
```

Outlier Control

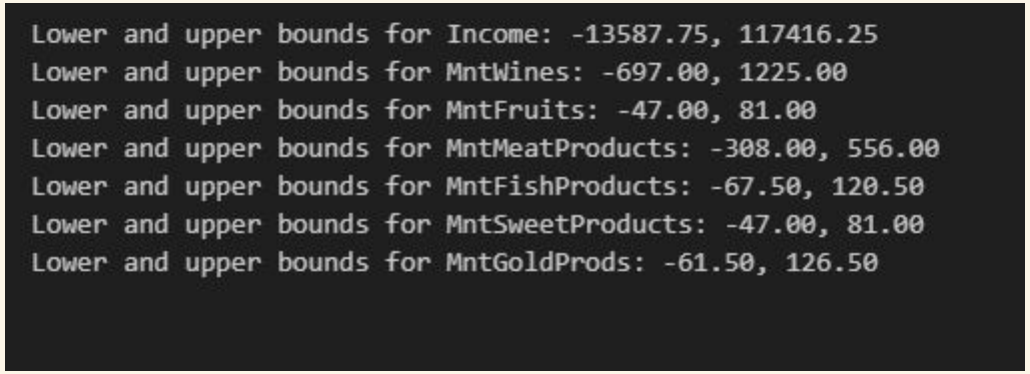


Outlier Control



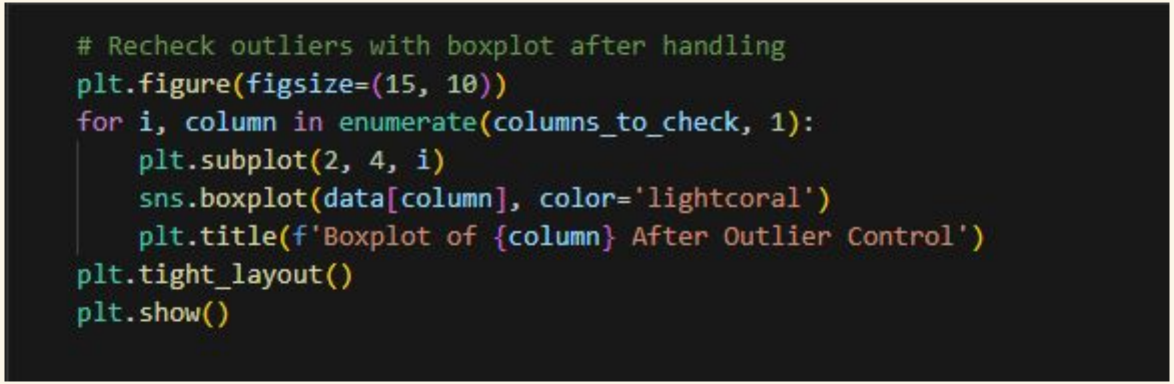
```python
# Function to calculate lower and upper boundaries for outliers using IQR
def find_outliers_IQR(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return lower_bound, upper_bound

# Display outlier boundaries for each column
for column in columns_to_check:
    lower, upper = find_outliers_IQR(data, column)
    print(f'Lower and upper bounds for {column}: {lower:.2f}, {upper:.2f}')
```

Outlier Control



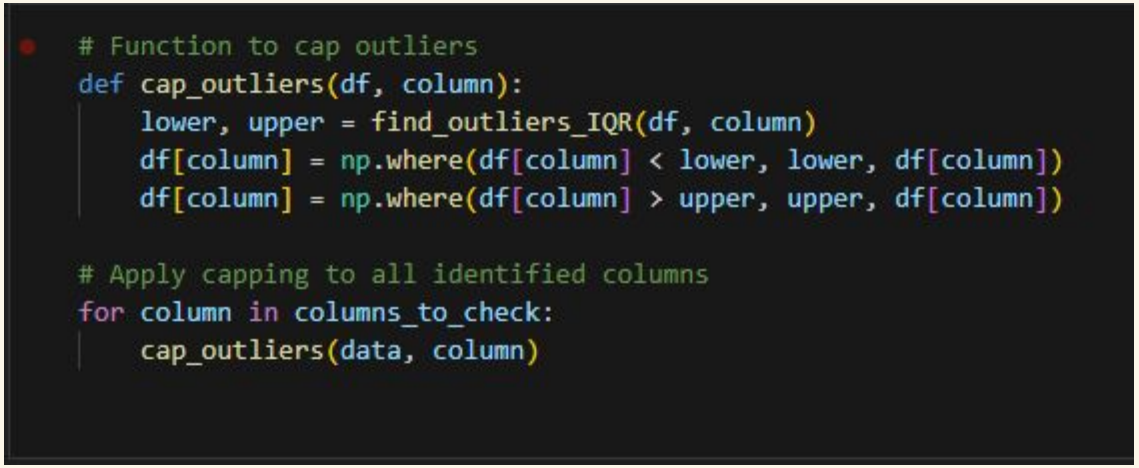
```
Lower and upper bounds for Income: -13587.75, 117416.25
Lower and upper bounds for MntWines: -697.00, 1225.00
Lower and upper bounds for MntFruits: -47.00, 81.00
Lower and upper bounds for MntMeatProducts: -308.00, 556.00
Lower and upper bounds for MntFishProducts: -67.50, 120.50
Lower and upper bounds for MntSweetProducts: -47.00, 81.00
Lower and upper bounds for MntGoldProds: -61.50, 126.50
```

Checking Outlier Handling



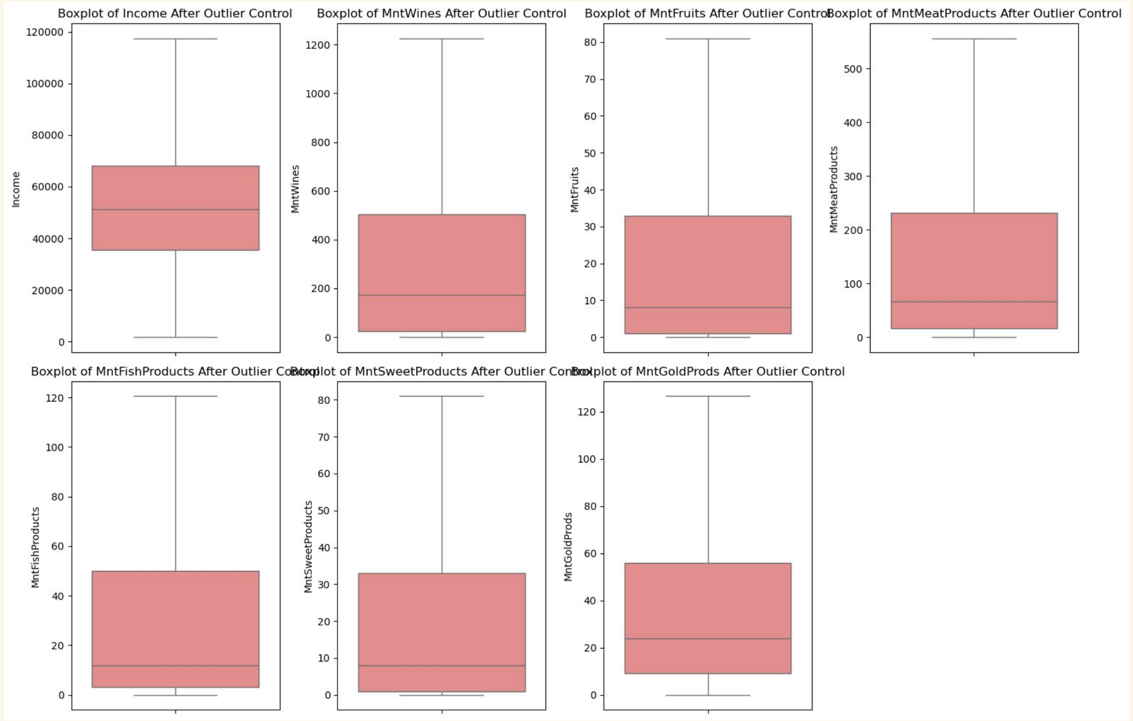
```
# Recheck outliers with boxplot after handling
plt.figure(figsize=(15, 10))
for i, column in enumerate(columns_to_check, 1):
    plt.subplot(2, 4, i)
    sns.boxplot(data[column], color='lightcoral')
    plt.title(f'Boxplot of {column} After Outlier Control')
plt.tight_layout()
plt.show()
```

Outlier Control With Capping



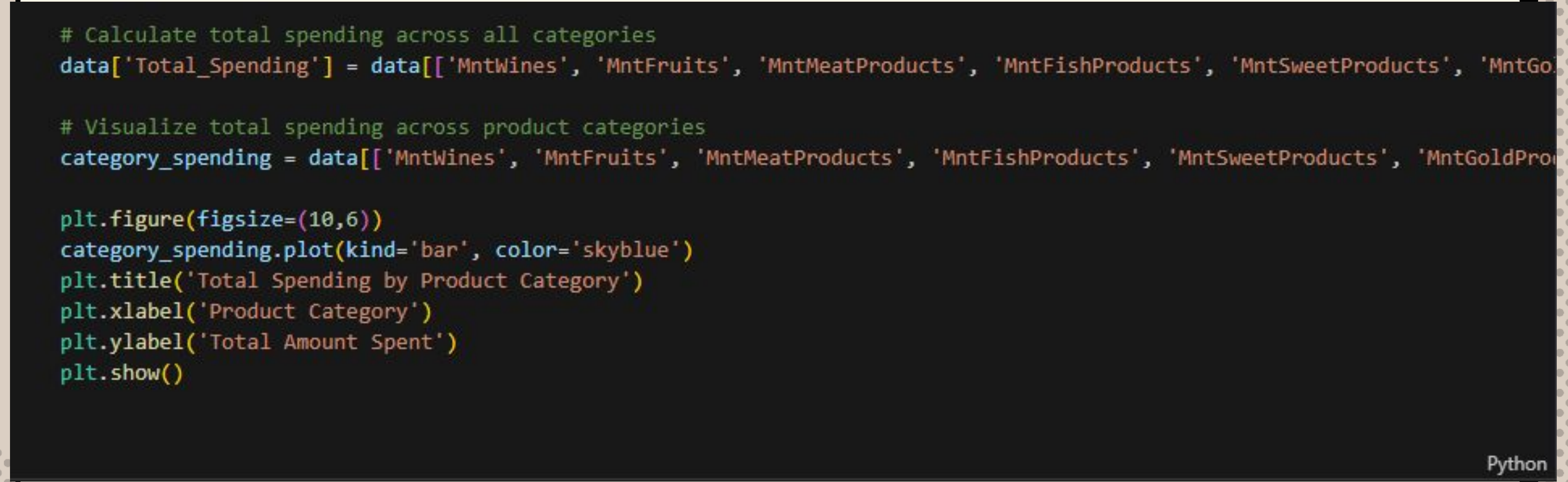
```
# Function to cap outliers
def cap_outliers(df, column):
    lower, upper = find_outliers_IQR(df, column)
    df[column] = np.where(df[column] < lower, lower, df[column])
    df[column] = np.where(df[column] > upper, upper, df[column])

# Apply capping to all identified columns
for column in columns_to_check:
    cap_outliers(data, column)
```

Outlier Control With Capping



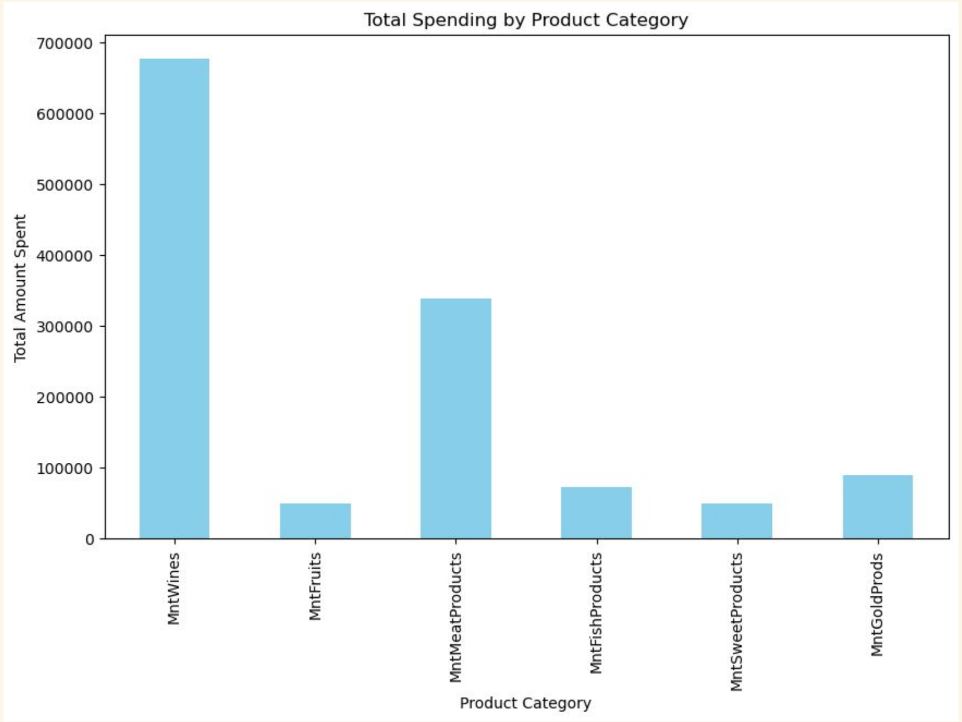
Data Analysis

»Total Spending by Product Category



```python
# Calculate total spending across all categories
data['Total_Spending'] = data[['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGo

# Visualize total spending across product categories
category_spending = data[['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldPro

plt.figure(figsize=(10,6))
category_spending.plot(kind='bar', color='skyblue')
plt.title('Total Spending by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Total Amount Spent')
plt.show()
```

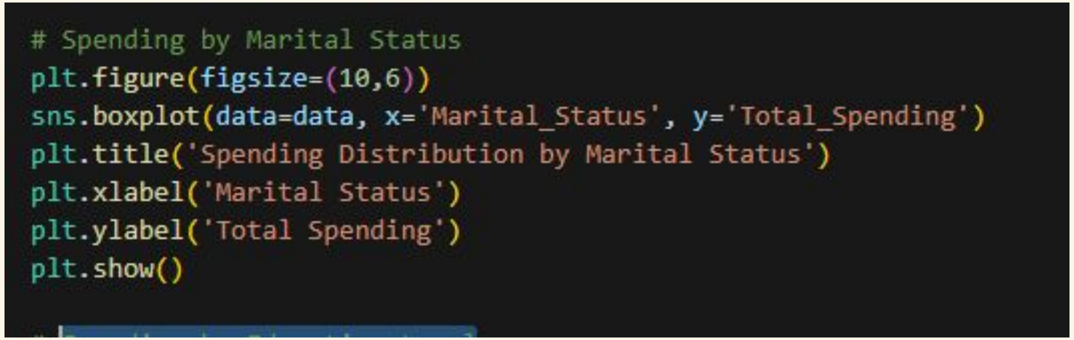
»Total Spending by Product Category



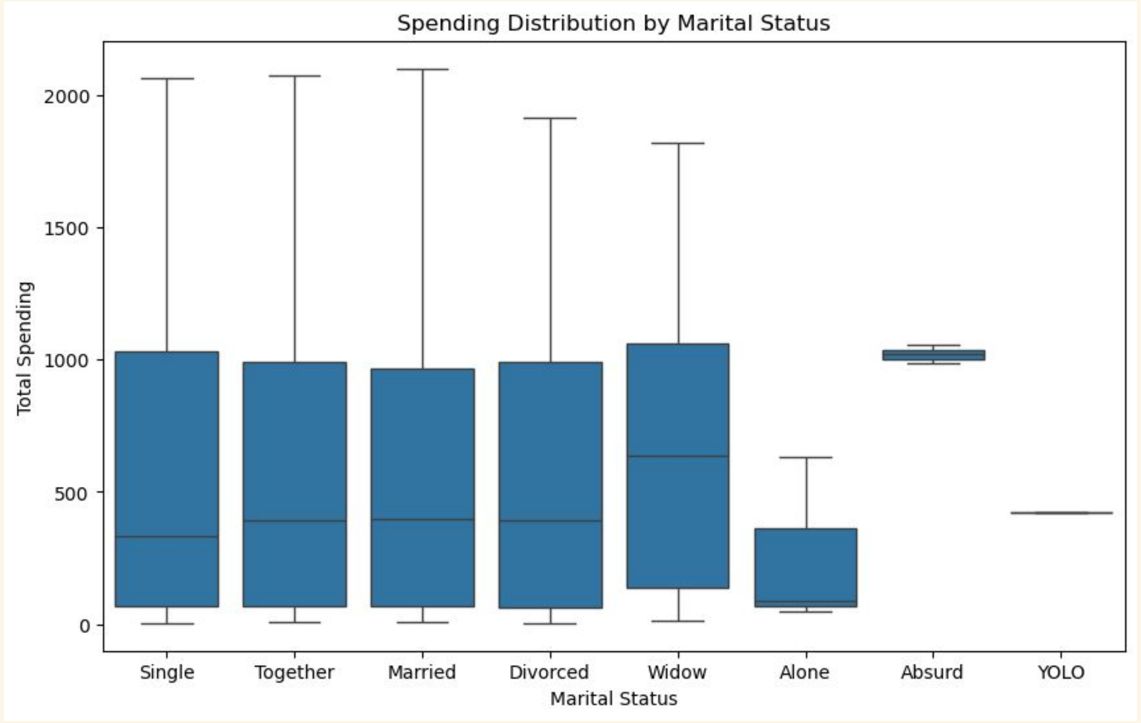
Spending Analysis by Demographic Groups

- Spending by Marital Status
- Spending by Education Level

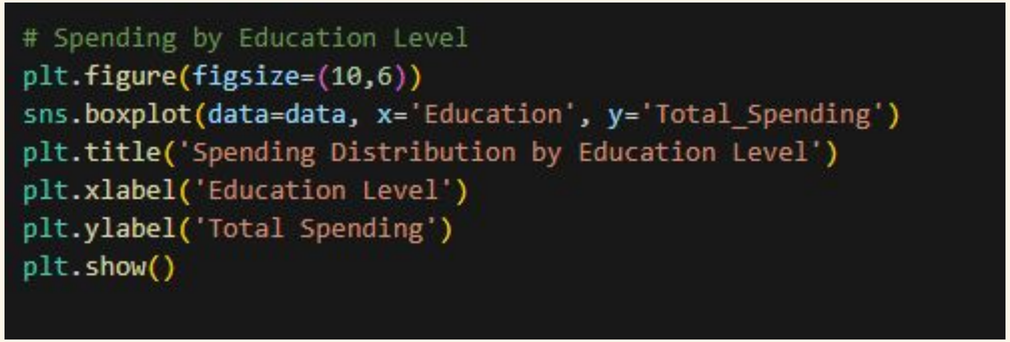
Spending by Marital Status


```
# Spending by Marital Status
plt.figure(figsize=(10,6))
sns.boxplot(data=data, x='Marital_Status', y='Total_Spending')
plt.title('Spending Distribution by Marital Status')
plt.xlabel('Marital Status')
plt.ylabel('Total Spending')
plt.show()
```

Spending by Marital Status

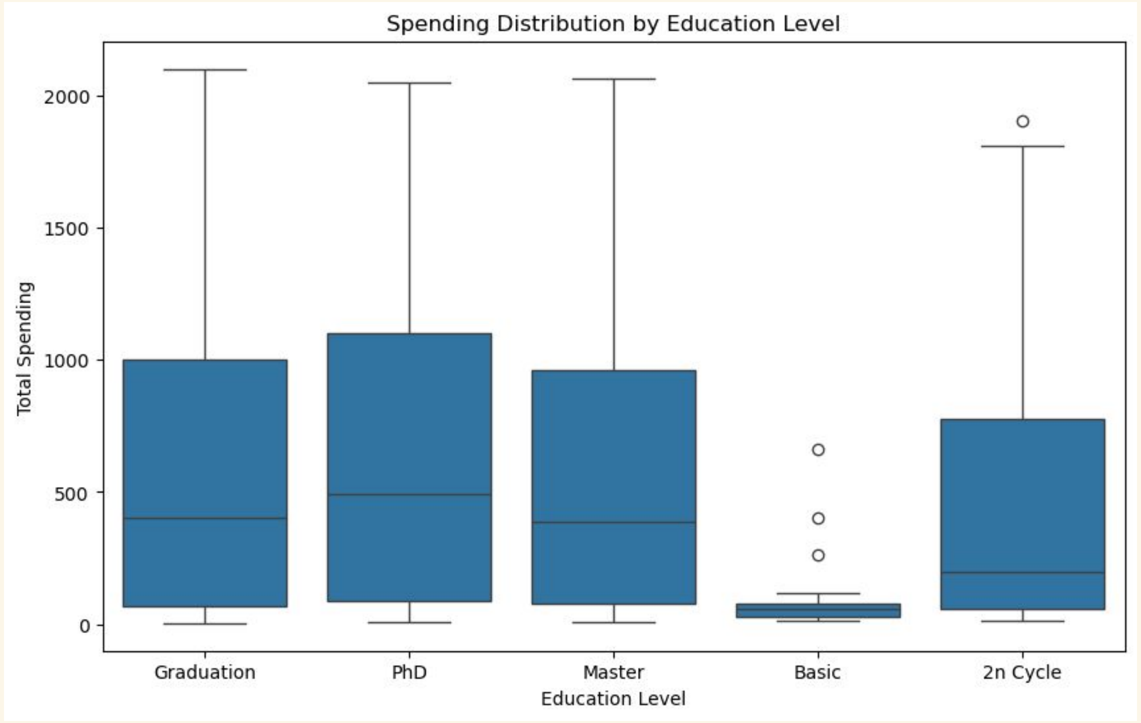


Spending by Education Level

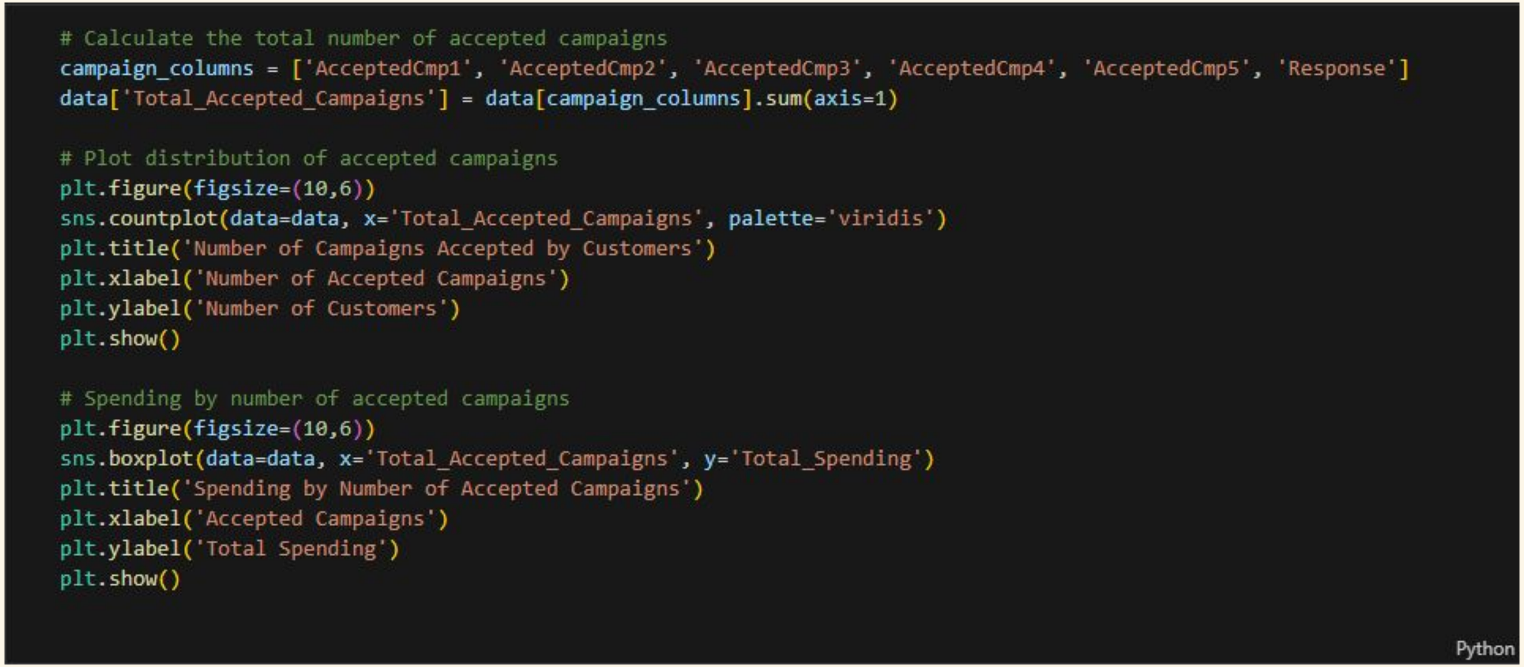


```
# Spending by Education Level
plt.figure(figsize=(10,6))
sns.boxplot(data=data, x='Education', y='Total_Spending')
plt.title('Spending Distribution by Education Level')
plt.xlabel('Education Level')
plt.ylabel('Total Spending')
plt.show()
```

Spending by Education Level

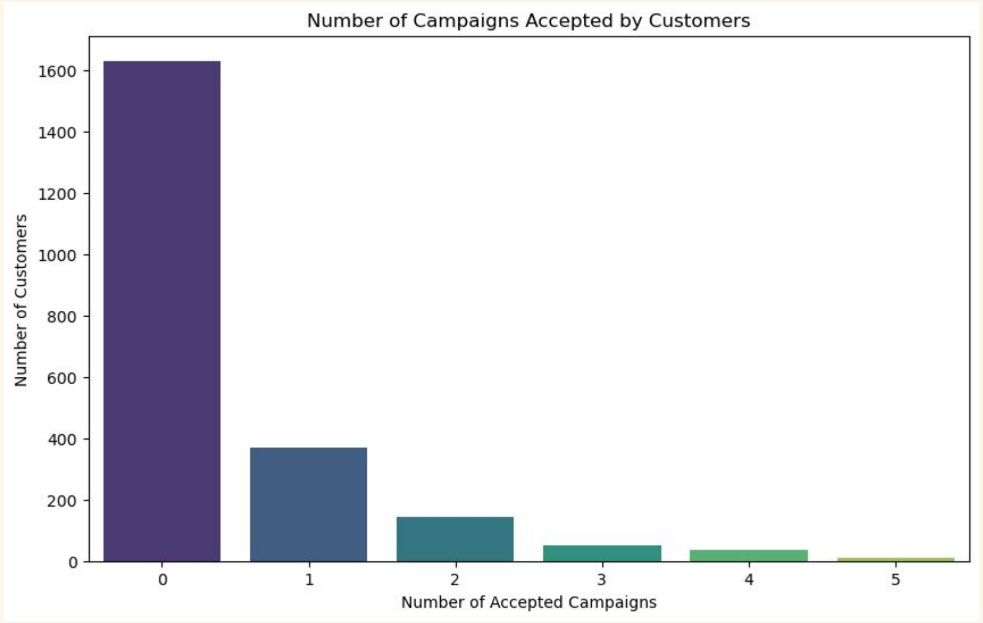


Customer Response to Marketing Campaigns

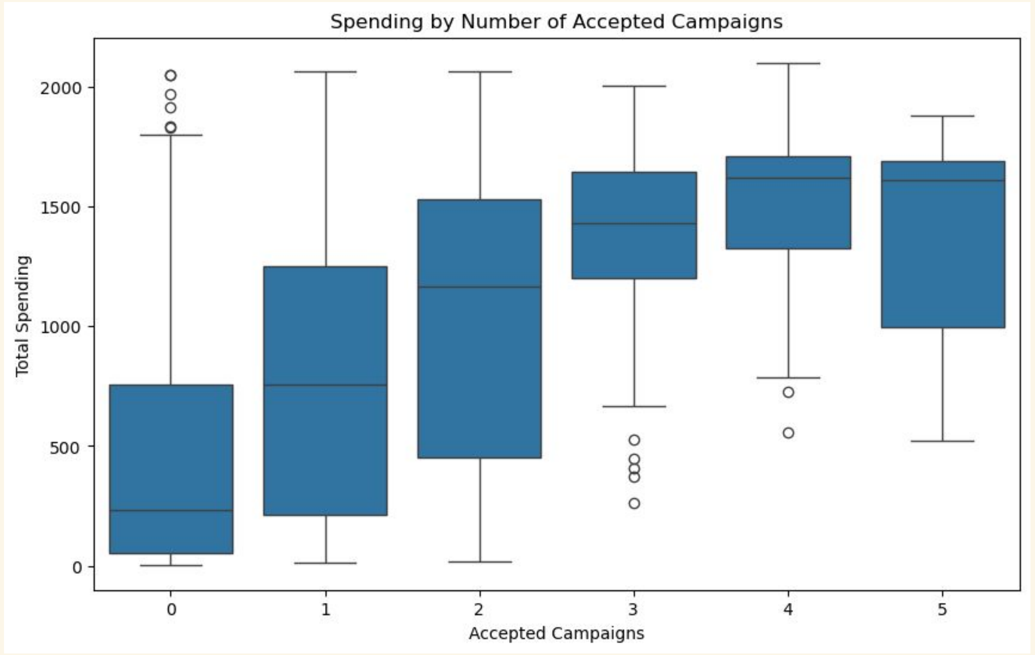


```python
# Calculate the total number of accepted campaigns
campaign_columns = ['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'Response']
data['Total_Accepted_Campaigns'] = data[campaign_columns].sum(axis=1)

# Plot distribution of accepted campaigns
plt.figure(figsize=(10,6))
sns.countplot(data=data, x='Total_Accepted_Campaigns', palette='viridis')
plt.title('Number of Campaigns Accepted by Customers')
plt.xlabel('Number of Accepted Campaigns')
plt.ylabel('Number of Customers')
plt.show()

# Spending by number of accepted campaigns
plt.figure(figsize=(10,6))
sns.boxplot(data=data, x='Total_Accepted_Campaigns', y='Total_Spending')
plt.title('Spending by Number of Accepted Campaigns')
plt.xlabel('Accepted Campaigns')
plt.ylabel('Total Spending')
plt.show()
```

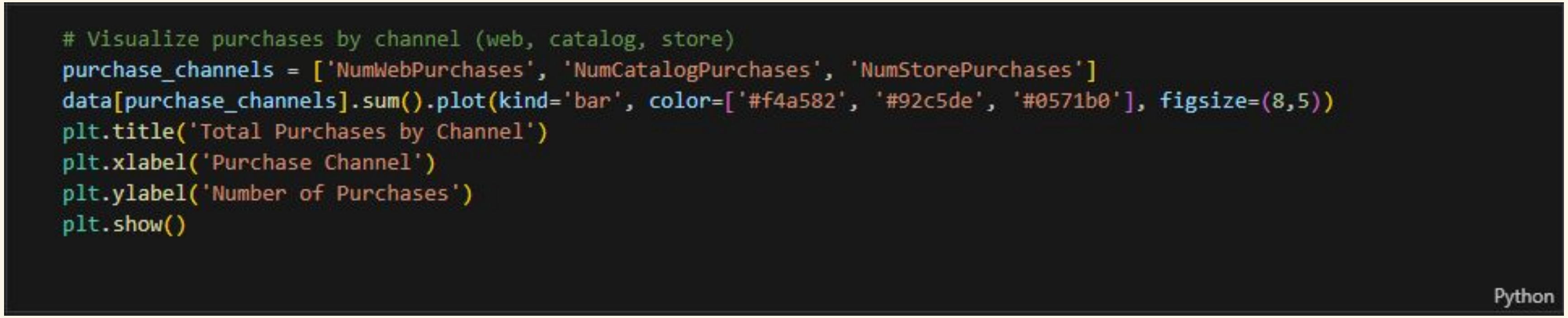
Customer Response to Marketing Campaigns



Customer Response to Marketing Campaigns

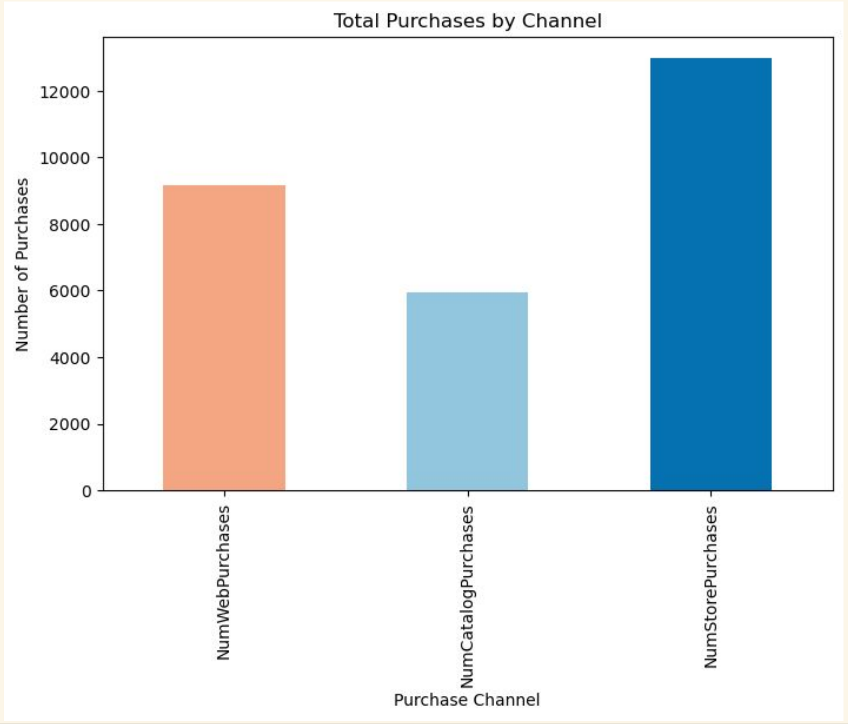


Purchase Channel Analysis



```
# Visualize purchases by channel (web, catalog, store)
purchase_channels = ['NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases']
data[purchase_channels].sum().plot(kind='bar', color=['#f4a582', '#92c5de', '#0571b0'], figsize=(8,5))
plt.title('Total Purchases by Channel')
plt.xlabel('Purchase Channel')
plt.ylabel('Number of Purchases')
plt.show()
```

Purchase Channel Analysis



Conclusions

Wine have the highest spending, followed by Meat products. Targeted promotions on these categories may have the biggest impact.

Married customers tend to have higher spending, so family-oriented promotions may be effective.

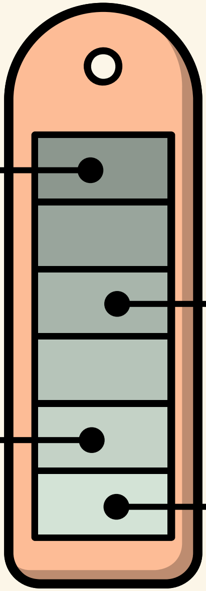
Customers who accept marketing campaigns generally spend more, indicating the effectiveness of such campaigns.

Web purchases are lower than store purchases, suggesting room for growth in online sales.

Recommendations

Increase promotions on high-spending categories like wine and meat products.

Focus targeting campaigns towards married and family-oriented customers.



Maintain and optimize marketing campaigns to drive purchases.

Improve the online shopping experience to boost sales through web channels.

Thanks!