black-friday-sales-eda-analysis

September 4, 2024

1 Rahul Manjhi

1.0.1 Black Friday Sales Analysis

1.1 Import Libraries:

[5]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

1.2 Load the Dataset:

[6]:
```python
# Load the dataset
file_path = 'BlackFridaySales.csv'
df = pd.read_csv(file_path)
```

[3]:
```python
df.head()
```

[3]:
```
   User_ID Product_ID Gender   Age  Occupation City_Category  \
0  1000001  P00069042      F  0-17          10             A
1  1000001  P00248942      F  0-17          10             A
2  1000001  P00087842      F  0-17          10             A
3  1000001  P00085442      F  0-17          10             A
4  1000002  P00285442      M   55+          16             C

  Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
0                          2               0                   3
1                          2               0                   1
2                          2               0                  12
3                          2               0                  12
4                         4+               0                   8

   Product_Category_2  Product_Category_3  Purchase
0                 NaN                 NaN      8370
1                 6.0                14.0     15200
2                 NaN                 NaN      1422
3                14.0                 NaN      1057
4                 NaN                 NaN      7969
```

[ ]:

2 Data Cleaning:

2.0.1 Missing Values:

[4]:
```python
# Check for missing values
missing_values = df.isnull().sum()

# Display missing values
missing_values
```

[4]:
```
User_ID                            0
Product_ID                         0
Gender                             0
Age                                0
Occupation                         0
City_Category                      0
Stay_In_Current_City_Years         0
Marital_Status                     0
Product_Category_1                 0
Product_Category_2            173638
Product_Category_3            383247
Purchase                           0
dtype: int64
```

[8]:
```python
df = df.assign(Product_Category_2=df['Product_Category_2'].fillna(-1),
               Product_Category_3=df['Product_Category_3'].fillna(-1))

# Verify that missing values have been handled
missing_values_after_imputation = df.isnull().sum()

# Display the updated missing values
missing_values_after_imputation
```

[8]:
```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category_1            0
Product_Category_2            0
Product_Category_3            0
```

```
Purchase                      0
dtype: int64
```

[10]:
```python
data_types = df.dtypes

# Display data types
data_types
```

[10]:
```
User_ID                        int64
Product_ID                    object
Gender                        object
Age                           object
Occupation                     int64
City_Category                 object
Stay_In_Current_City_Years    object
Marital_Status                 int64
Product_Category_1             int64
Product_Category_2           float64
Product_Category_3           float64
Purchase                       int64
dtype: object
```

[ ]:

3 Data Understanding:

[12]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     550068 non-null  int64
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64
 8   Product_Category_1          550068 non-null  int64
 9   Product_Category_2          550068 non-null  float64
 10  Product_Category_3          550068 non-null  float64
 11  Purchase                    550068 non-null  int64
dtypes: float64(2), int64(5), object(5)
```

```
memory usage: 50.4+ MB
```

[13]:
```
df.describe()
```

[13]:
```
              User_ID     Occupation  Marital_Status  Product_Category_1  \
count  5.500680e+05  550068.000000   550068.000000       550068.000000
mean   1.003029e+06       8.076707        0.409653            5.404270
std    1.727592e+03       6.522660        0.491770            3.936211
min    1.000001e+06       0.000000        0.000000            1.000000
25%    1.001516e+06       2.000000        0.000000            1.000000
50%    1.003077e+06       7.000000        0.000000            5.000000
75%    1.004478e+06      14.000000        1.000000            8.000000
max    1.006040e+06      20.000000        1.000000           20.000000

       Product_Category_2  Product_Category_3       Purchase
count        550068.00000       550068.000000  550068.000000
mean              6.41977            3.145215    9263.968713
std               6.56511            6.681039    5023.065394
min              -1.00000           -1.000000      12.000000
25%              -1.00000           -1.000000    5823.000000
50%               5.00000           -1.000000    8047.000000
75%              14.00000            8.000000   12054.000000
max              18.00000           18.000000   23961.000000
```

[14]:
```
df['Gender'].value_counts()
df['Age'].value_counts()
df['City_Category'].value_counts()
df['Marital_Status'].value_counts()
```

[14]:
```
Marital_Status
0    324731
1    225337
Name: count, dtype: int64
```

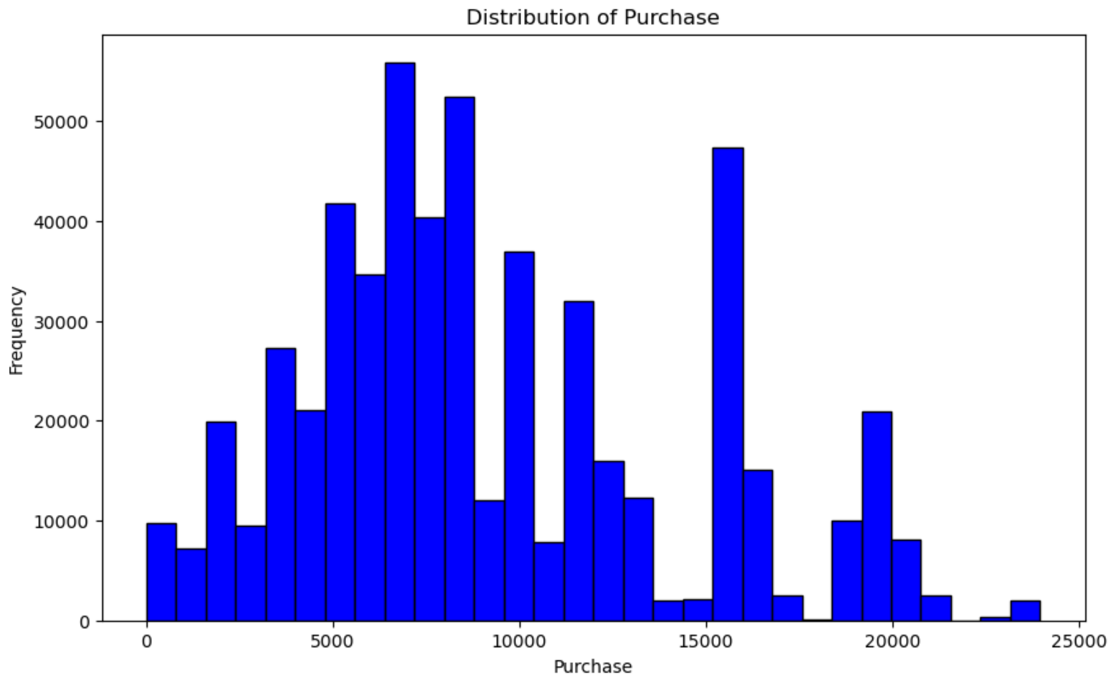
[ ]:
```

```

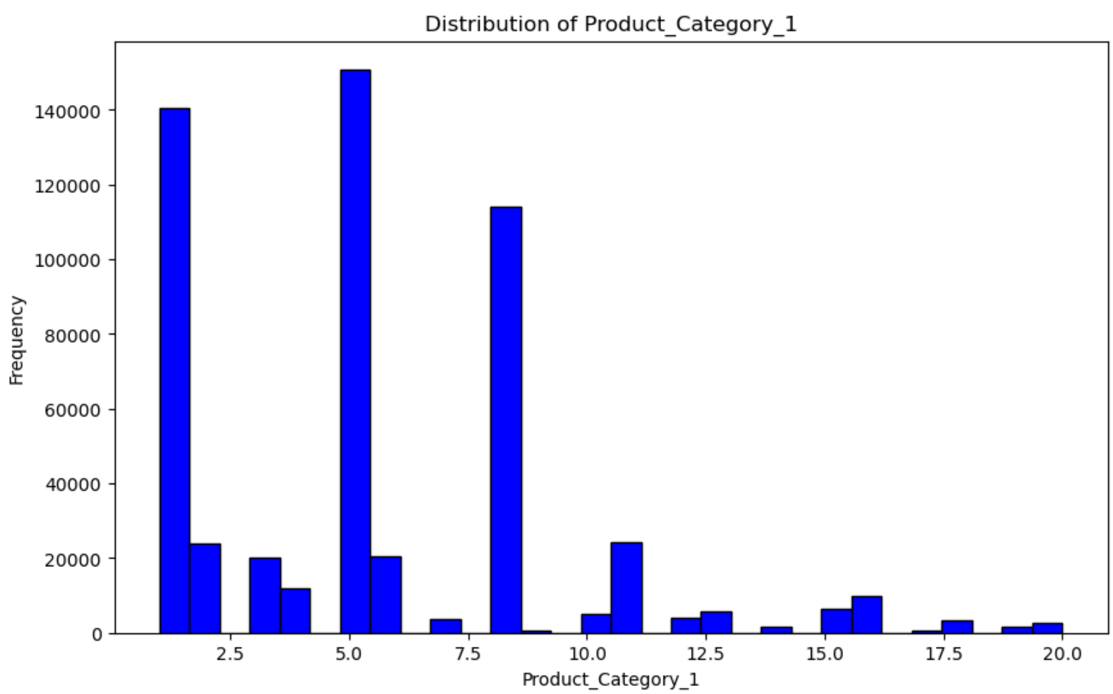
4 Data Exploration and Visualization

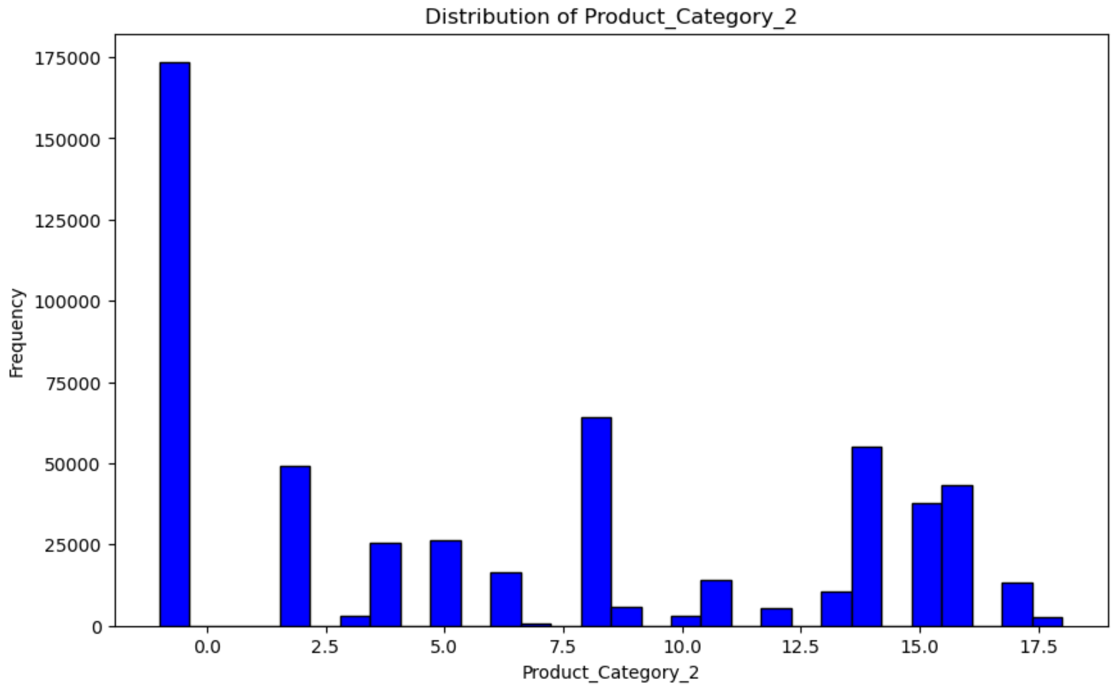
4.1 1. Numerical Feature Distribution: Histograms

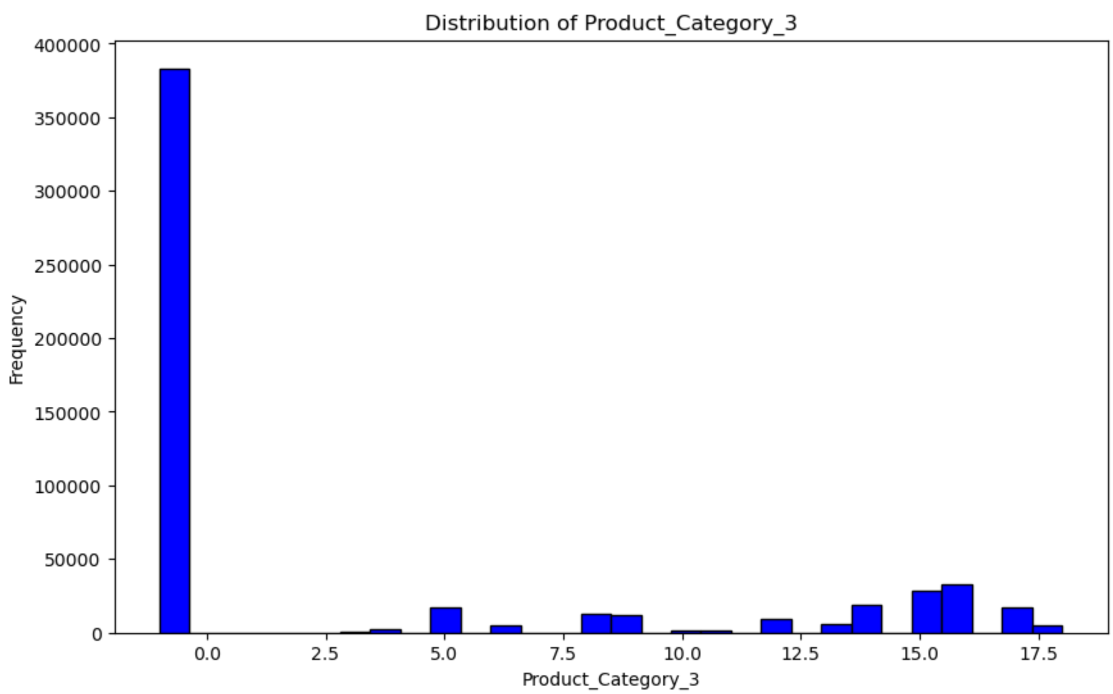
[18]:
```
# Plot histograms for numerical features
numerical_features = ['Purchase', 'Product_Category_1', 'Product_Category_2',␣
 ↪'Product_Category_3']

for feature in numerical_features:
    plt.figure(figsize=(10, 6))
    plt.hist(df[feature], bins=30, color='blue', edgecolor='black')
```

```
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Frequency')
    plt.show()
```






Distribution of Product_Category_2
Frequency
175000
150000
125000
100000
75000
50000
25000
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Product_Category_2


Distribution of Product_Category_3
Frequency
400000
350000
300000
250000
200000
150000
100000
50000
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Product_Category_3

[ ]:

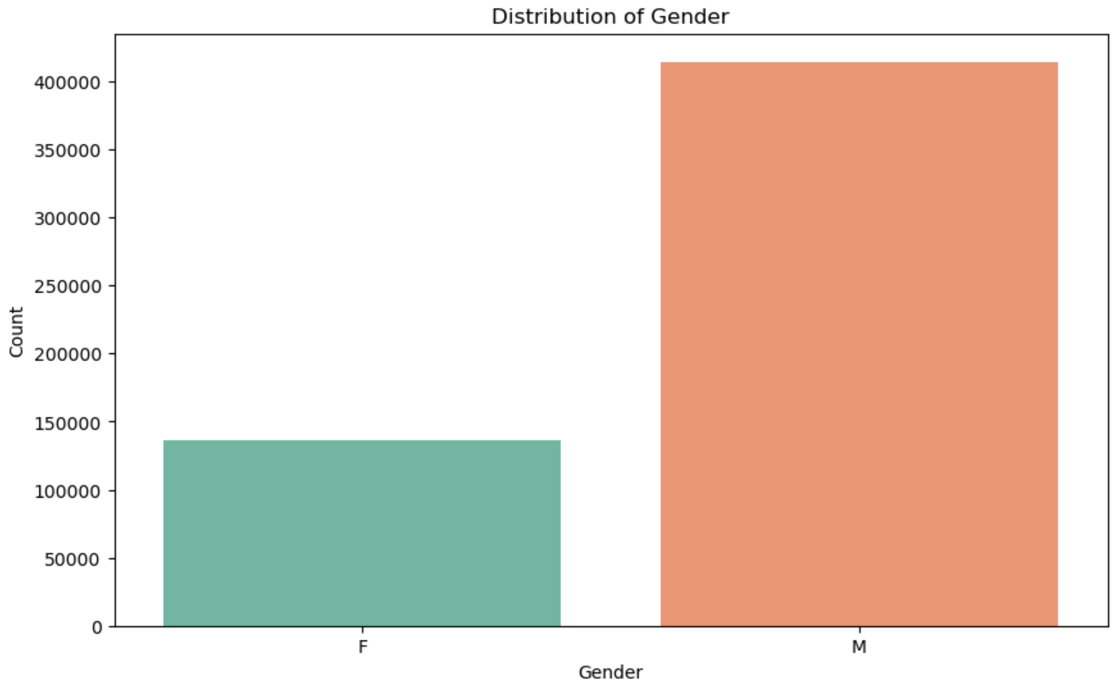
```

```

4.2 2. Categorical Feature Distribution: Bar Charts and Pie Charts

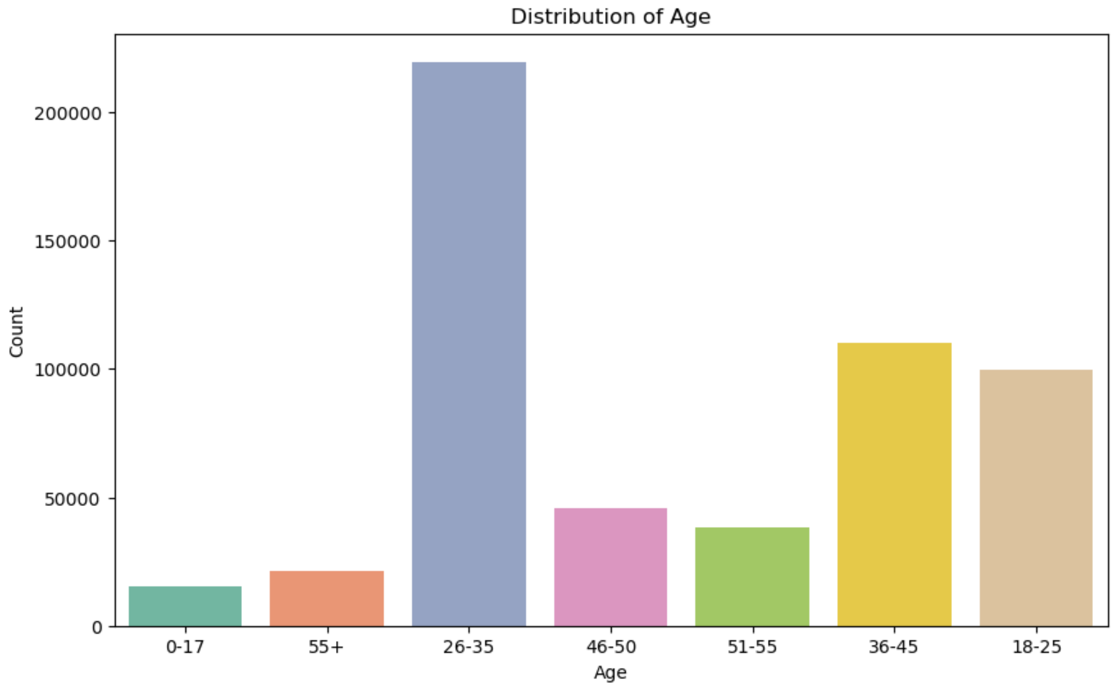
[20]:

```
# Bar chart for categorical features
categorical_features = ['Gender', 'Age', 'City_Category', 'Marital_Status']

for feature in categorical_features:
    plt.figure(figsize=(10, 6))
    sns.countplot(data=df, x=feature, palette='Set2')
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Count')
    plt.show()

# Pie chart for a specific categorical feature
df['Gender'].value_counts().plot.pie(autopct='%1.1f%%', figsize=(8, 8),␣
↪colors=['#ff9999','#66b3ff'])
plt.title('Gender Distribution')
plt.show()
```

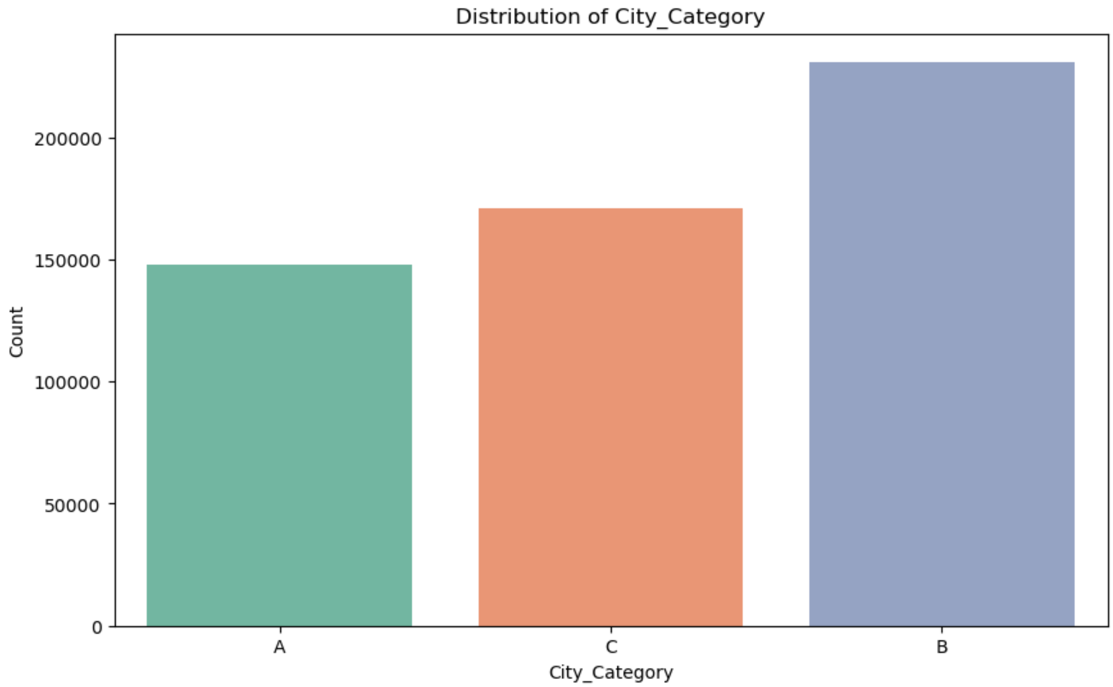
```
C:\Users\manjh\AppData\Local\Temp\ipykernel_4992\1913232349.py:6: FutureWarning:

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(data=df, x=feature, palette='Set2')
```



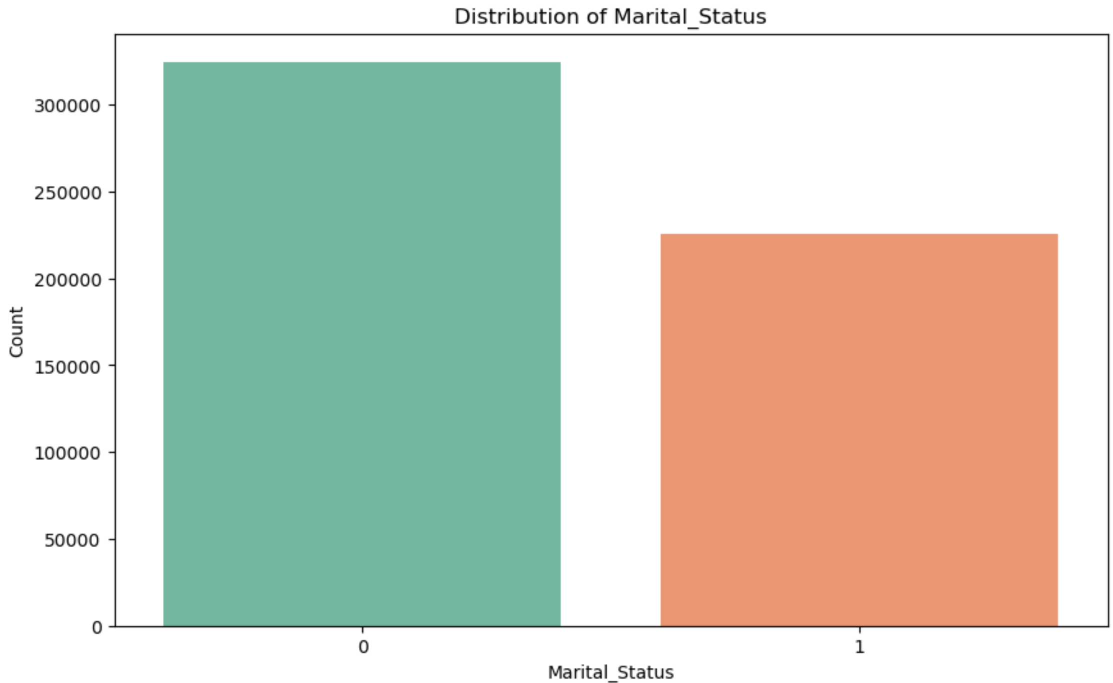

```
C:\Users\manjh\AppData\Local\Temp\ipykernel_4992\1913232349.py:6: FutureWarning:

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(data=df, x=feature, palette='Set2')
```

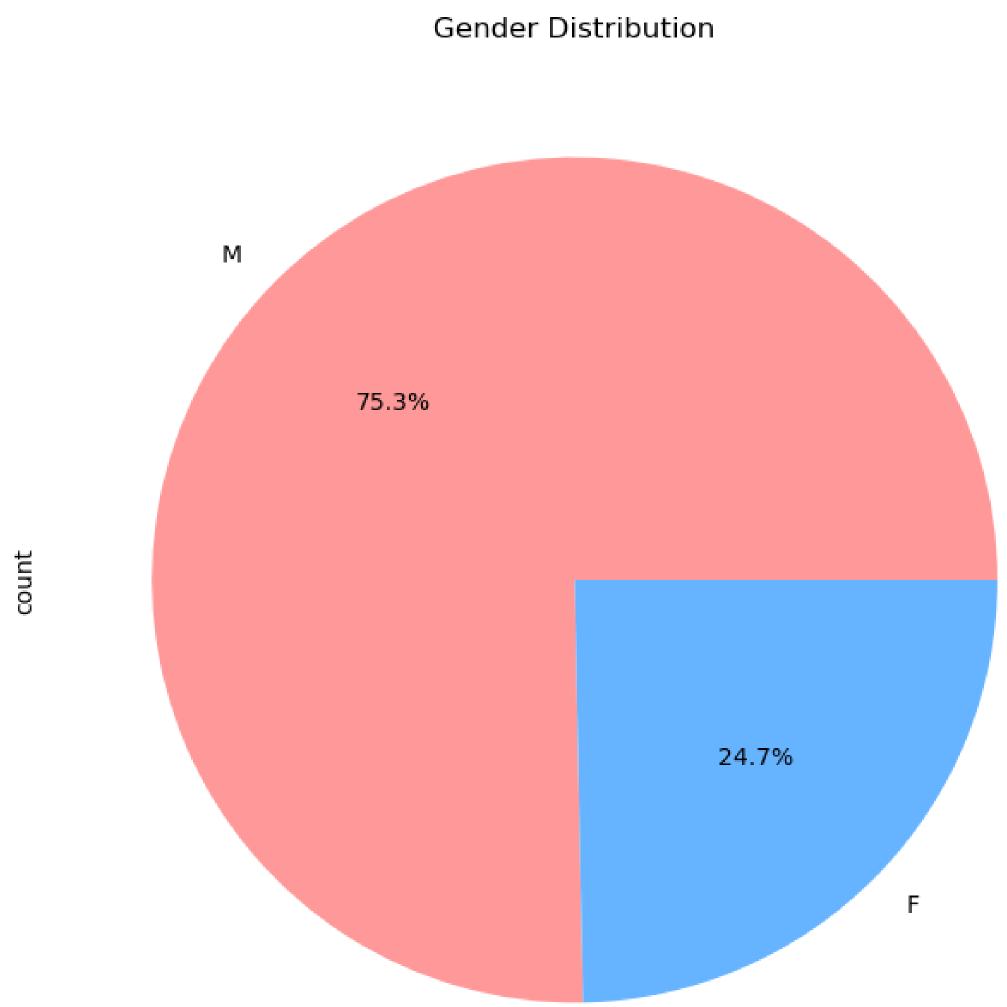



```
C:\Users\manjh\AppData\Local\Temp\ipykernel_4992\1913232349.py:6: FutureWarning:

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(data=df, x=feature, palette='Set2')
```




```
C:\Users\manjh\AppData\Local\Temp\ipykernel_4992\1913232349.py:6: 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(data=df, x=feature, palette='Set2')
```


Distribution of Marital_Status
300000
250000
200000
150000
100000
50000
0
Count
0
1
Marital_Status




```
[ ]:
```

```
[ ]:
```

4.3 3. Boxplots: Purchase Distribution Across Groups

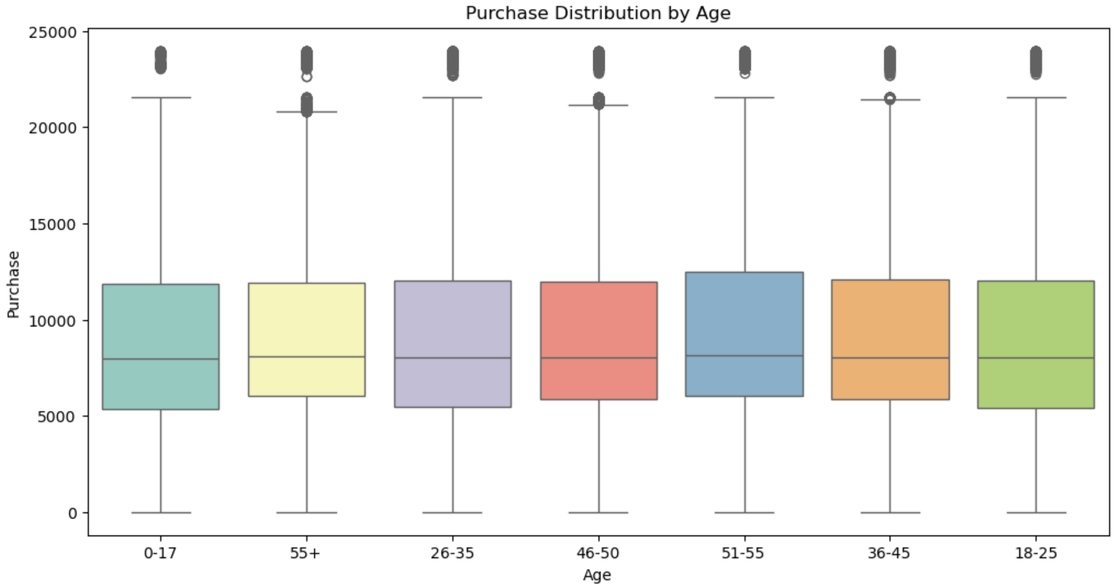
```
[22]: # Boxplots for Purchase across different categories
plt.figure(figsize=(12, 6))
sns.boxplot(df, x='Gender', y='Purchase', palette='Set3')
plt.title('Purchase Distribution by Gender')
plt.show()
```

```
C:\Users\manjh\AppData\Local\Temp\ipykernel_4992\1111442385.py:3: 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.boxplot(df, x='Gender', y='Purchase', palette='Set3')
```

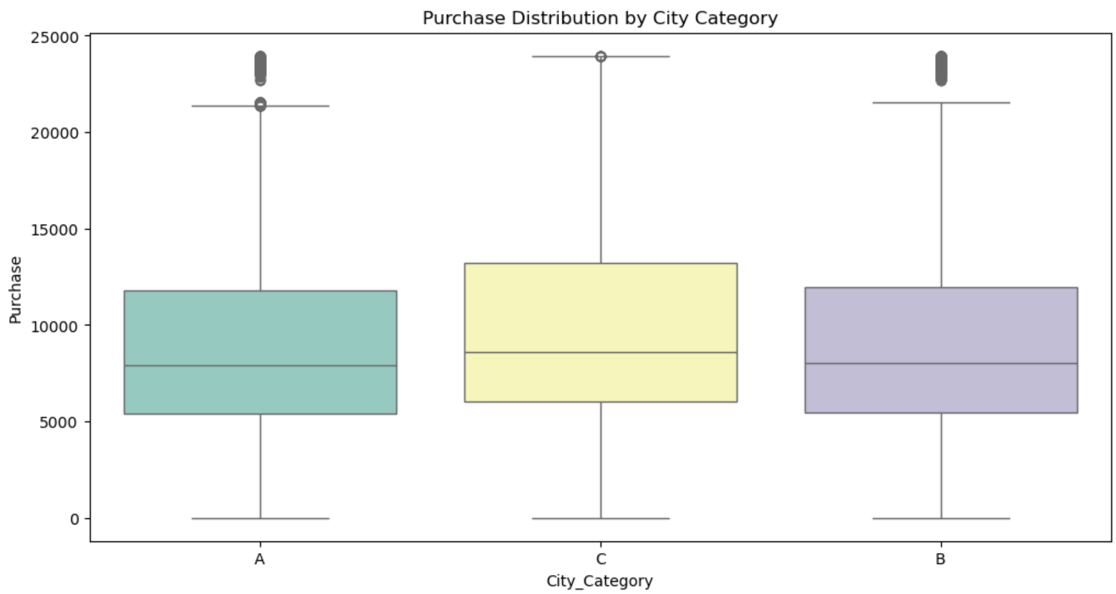
[24]:
```python
plt.figure(figsize=(12, 6))
sns.boxplot(df, x='Age', y='Purchase', palette='Set3')
plt.title('Purchase Distribution by Age')
plt.show()
```

```
C:\Users\manjh\AppData\Local\Temp\ipykernel_4992\2150581432.py: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.boxplot(df, x='Age', y='Purchase', palette='Set3')
```



[25]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(df, x='City_Category', y='Purchase', palette='Set3')
plt.title('Purchase Distribution by City Category')
plt.show()
```

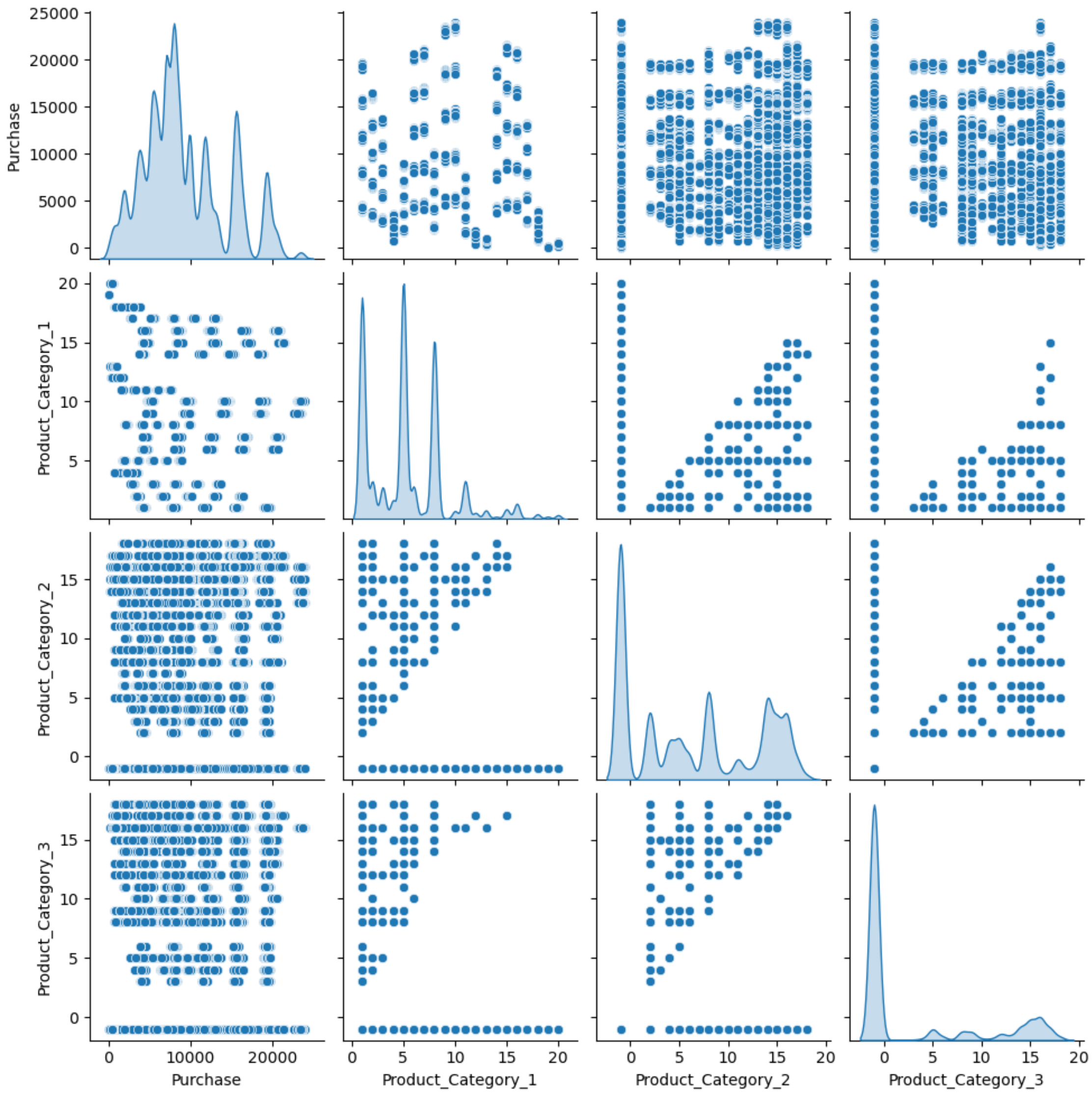
```
C:\Users\manjh\AppData\Local\Temp\ipykernel_4992\3881002303.py:2: FutureWarning:

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.boxplot(df, x='City_Category', y='Purchase', palette='Set3')
```



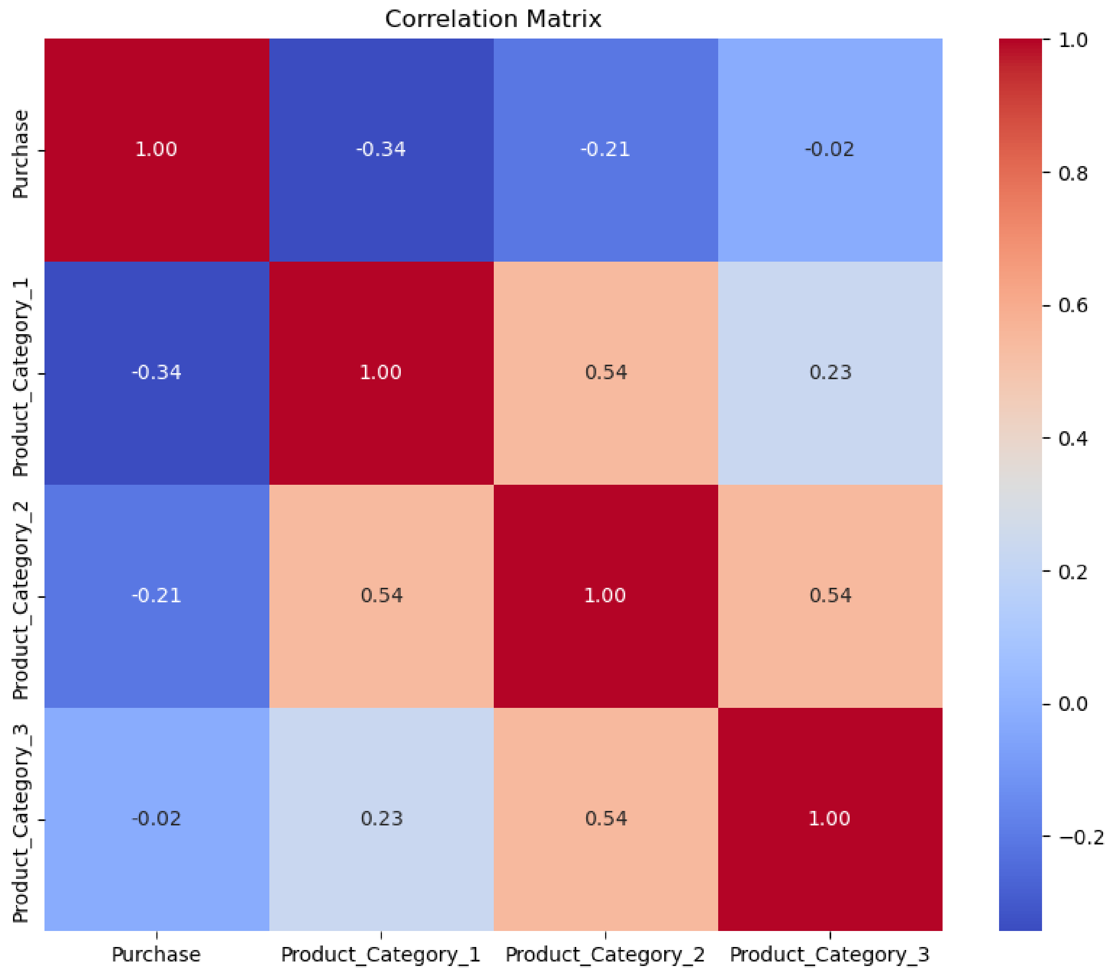
[ ]:

4.4 4. Correlation Analysis: Scatter Plots and Correlation Matrix

[27]:
```
# Scatter plots
sns.pairplot(df[['Purchase', 'Product_Category_1', 'Product_Category_2',␣
↪'Product_Category_3']], diag_kind='kde')
plt.show()
```



[9]:
```
# Correlation matrix
plt.figure(figsize=(10, 8))
corr_matrix = df[['Purchase', 'Product_Category_1', 'Product_Category_2',␣
↪'Product_Category_3']].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```

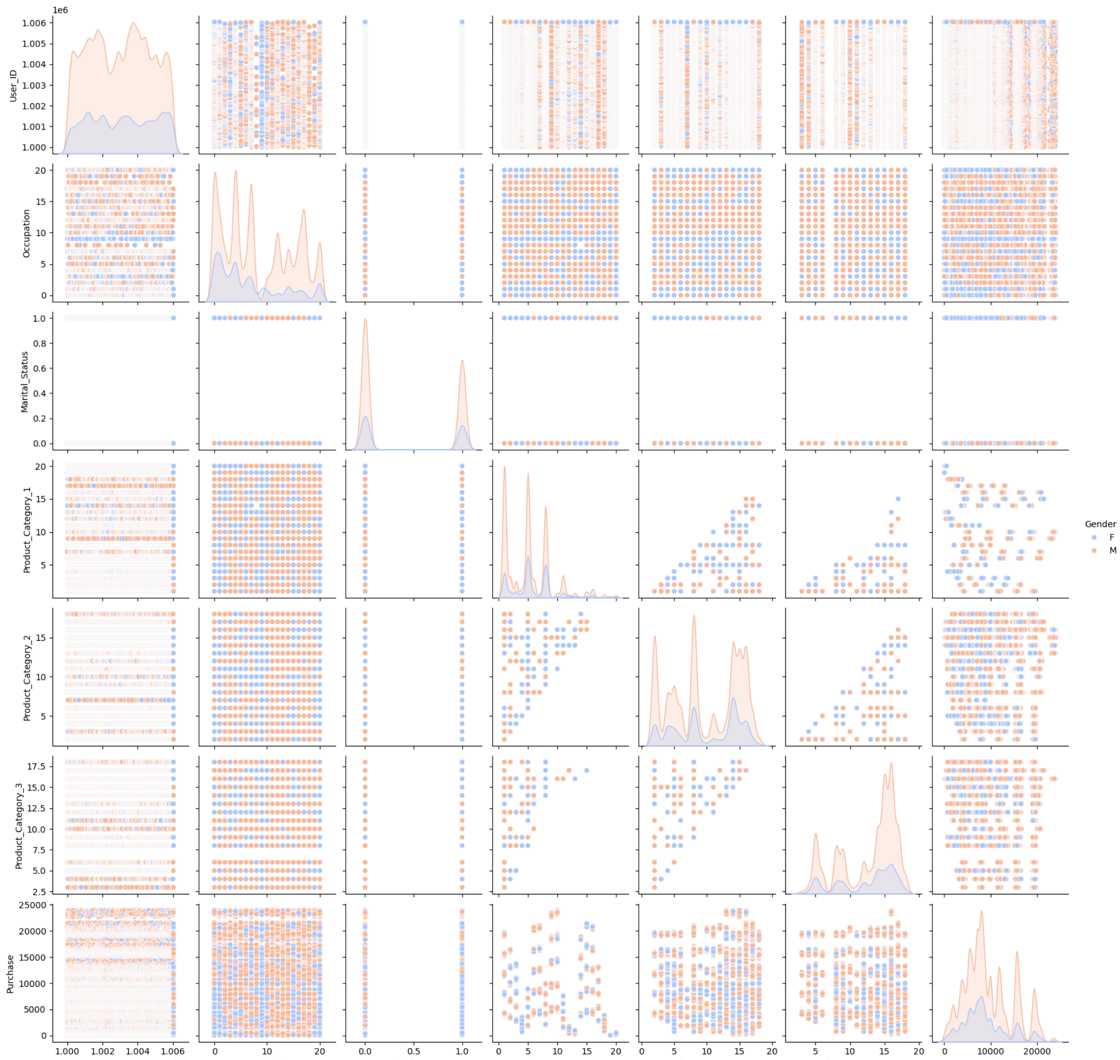


[ ]:

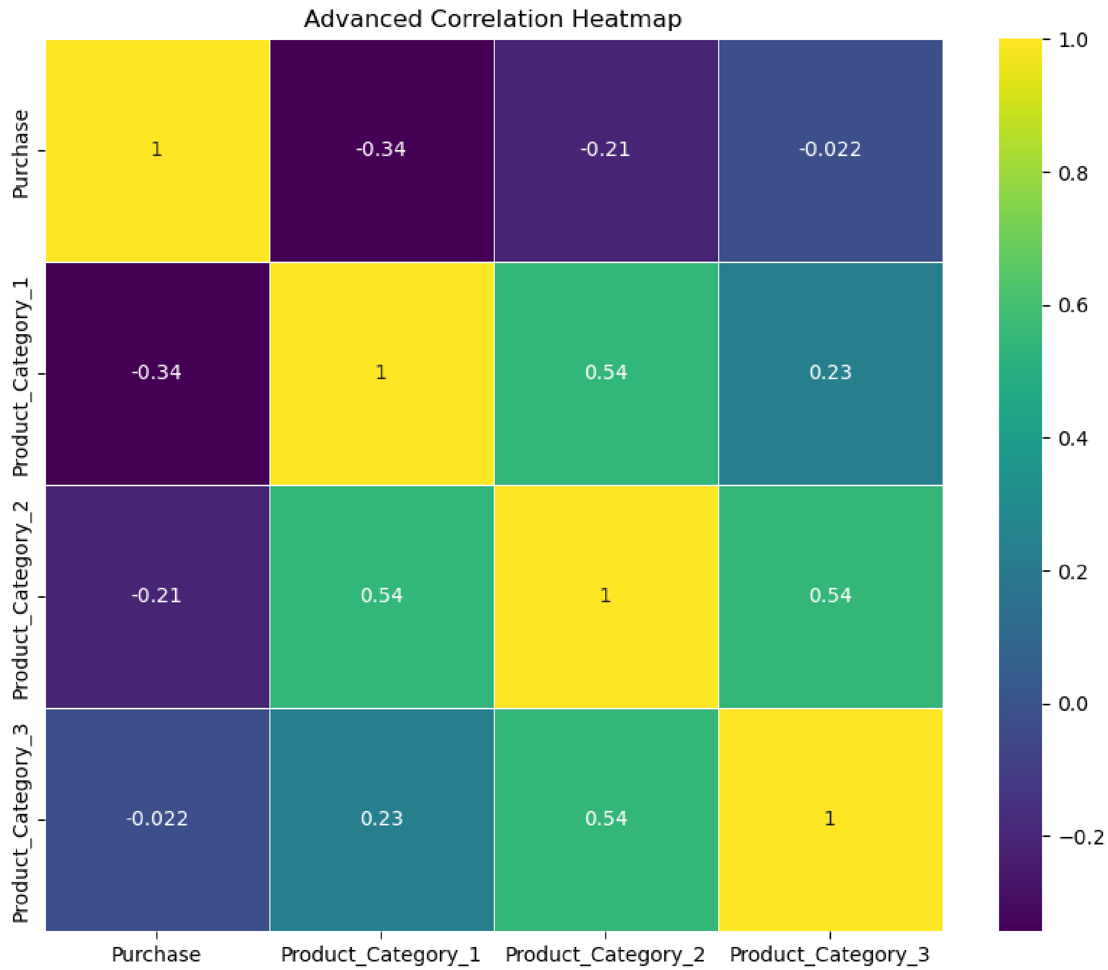
4.5 5. Advanced Visualizations with Seaborn

[7]:

```python
# Pairplot for detailed relationship analysis
sns.pairplot(df, hue='Gender', palette='coolwarm')
plt.show()
```



[10]:
```
# Advanced heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='viridis', linewidths=0.5)
plt.title('Advanced Correlation Heatmap')
plt.show()
```



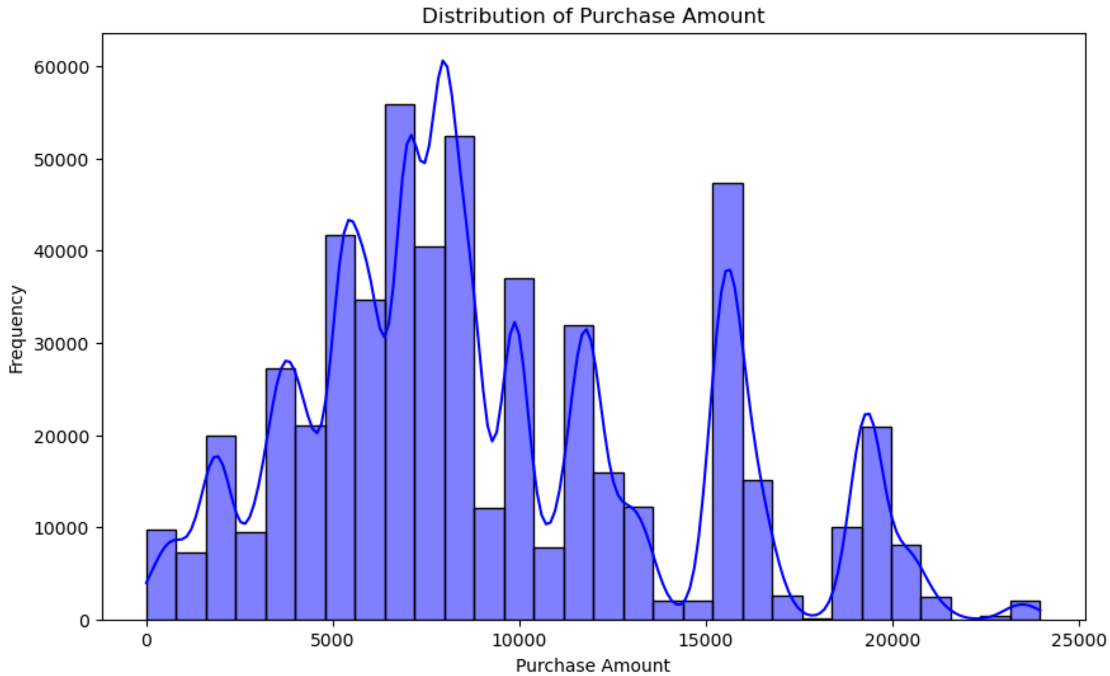
[ ]:

5 Analysis Questions:

6 Q1 Purchase Behavior Analysis:

6.1 1. Distribution of Purchase Amount

[11]:
```
# Distribution of Purchase Amount
plt.figure(figsize=(10, 6))
sns.histplot(df['Purchase'], bins=30, kde=True, color='blue')
plt.title('Distribution of Purchase Amount')
plt.xlabel('Purchase Amount')
plt.ylabel('Frequency')
plt.show()
```



6.2 2. Purchase Behavior by Gender

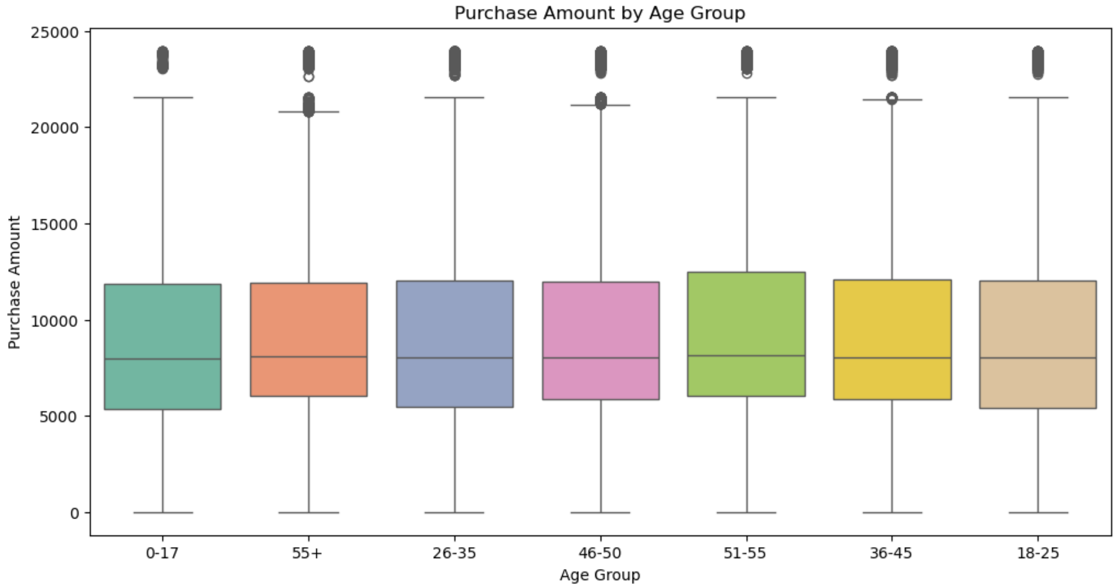
[21]:
```
# Average Purchase Amount by Gender
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Gender', palette='Set3',␣
↪legend=False)
plt.title('Purchase Amount by Gender')
plt.xlabel('Gender')
plt.ylabel('Purchase Amount')
plt.show()
```

[15]:
```
# Mean Purchase Amount by Gender
gender_purchase = df.groupby('Gender')['Purchase'].mean()
print(gender_purchase)
```

```
Gender
F    8734.565765
M    9437.526040
Name: Purchase, dtype: float64
```

6.3 3. Purchase Behavior by Age Group

[19]:
```
# Purchase Amount by Age Group
plt.figure(figsize=(12, 6))
sns.boxplot(data=df, y='Purchase', x='Age', hue='Age', palette='Set2',␣
↪legend=False)
plt.title('Purchase Amount by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Purchase Amount')
plt.show()
```



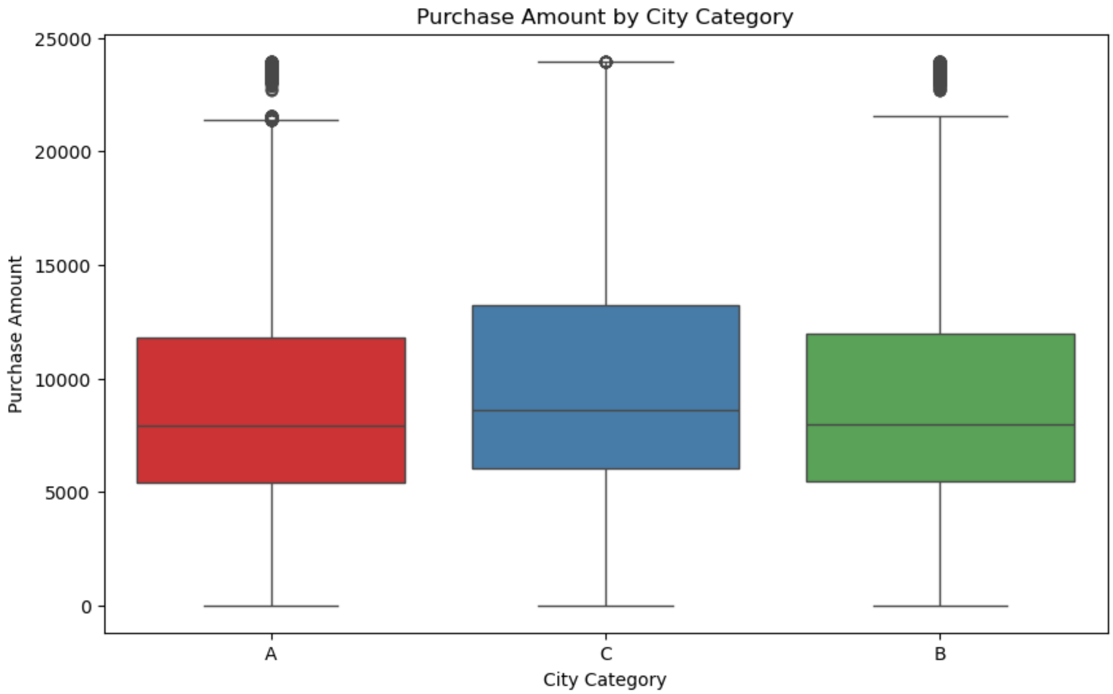
[17]:
```python
# Mean Purchase Amount by Age Group
age_purchase = df.groupby('Age')['Purchase'].mean().sort_values(ascending=False)
print(age_purchase)
```

```
Age
51-55    9534.808031
55+      9336.280459
36-45    9331.350695
26-35    9252.690633
46-50    9208.625697
18-25    9169.663606
0-17     8933.464640
Name: Purchase, dtype: float64
```

[ ]:
```python

```

6.4 4. Purchase Behavior by City Category

[22]:
```python
# Purchase Amount by City Category
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, y='Purchase', x='City_Category', hue='City_Category',␣
 ↪palette='Set1', legend=False)
plt.title('Purchase Amount by City Category')
plt.xlabel('City Category')
plt.ylabel('Purchase Amount')
plt.show()
```



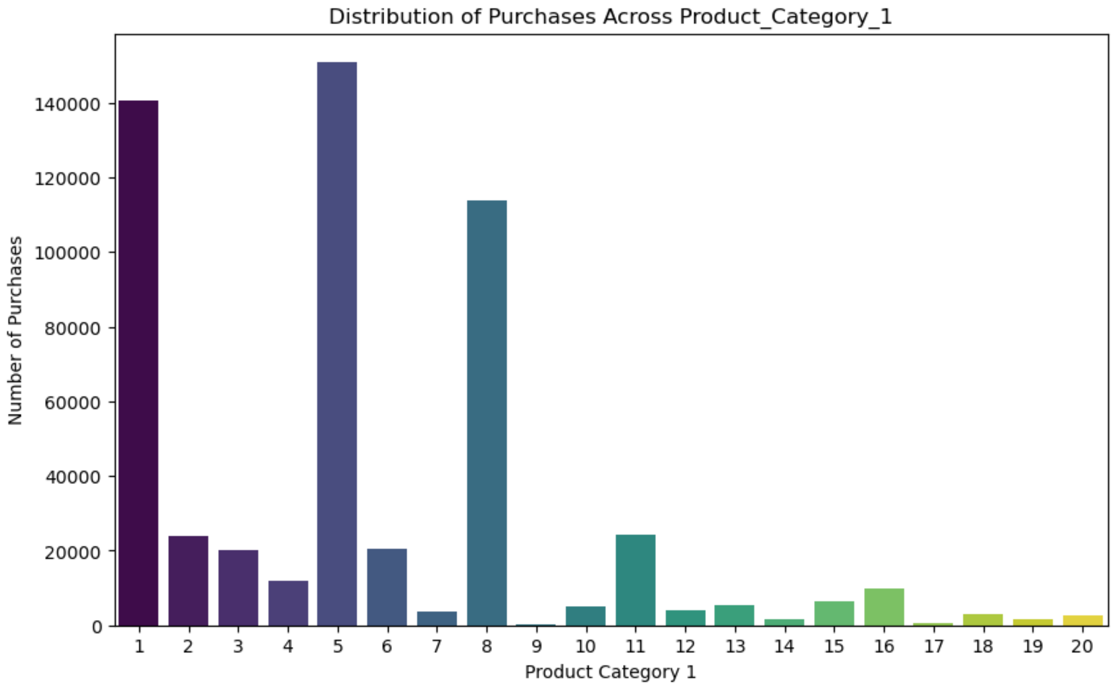
[ ]:

[ ]:

7 Q2 Product Category Insights:

7.1 Product_Category_1

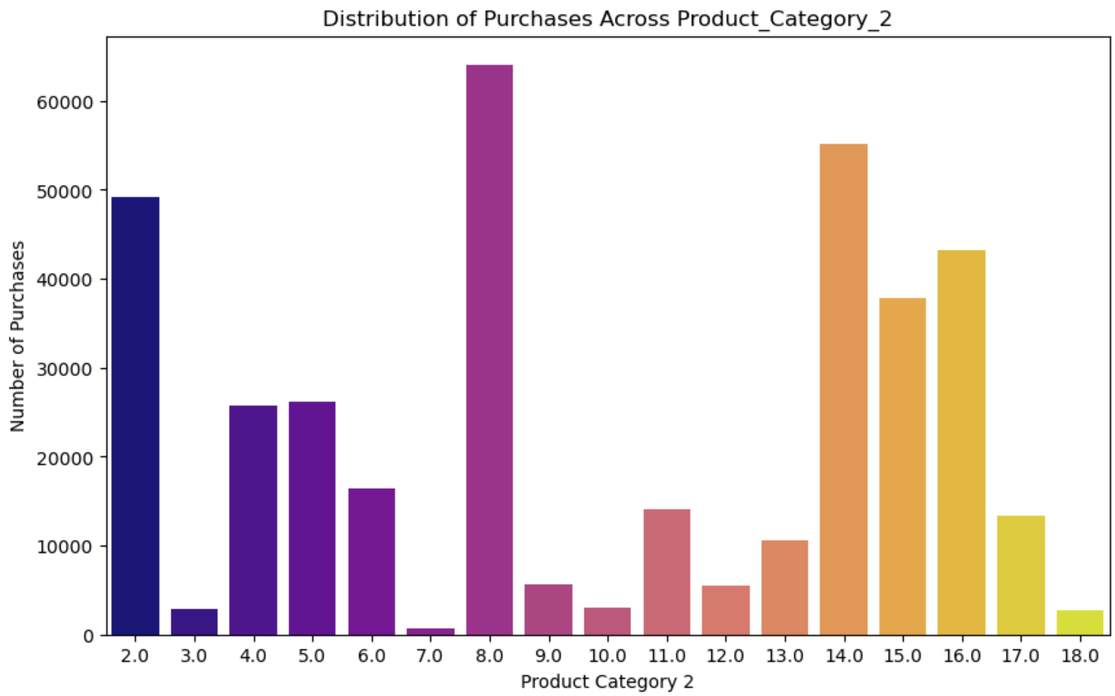
[26]:
```python
# Distribution of Product_Category_1
plt.figure(figsize=(10, 6))
sns.countplot(x='Product_Category_1', hue='Product_Category_1', data=df,␣
 ↪palette='viridis', legend=False)
plt.title('Distribution of Purchases Across Product_Category_1')
plt.xlabel('Product Category 1')
plt.ylabel('Number of Purchases')
plt.show()
```




```
[ ]:
```

7.2 Product_Category_2

```
[27]: # Distribution of Product_Category_2
plt.figure(figsize=(10, 6))
sns.countplot(x='Product_Category_2', hue='Product_Category_2', data=df,
    palette='plasma', legend=False)
plt.title('Distribution of Purchases Across Product_Category_2')
plt.xlabel('Product Category 2')
plt.ylabel('Number of Purchases')
plt.show()
```



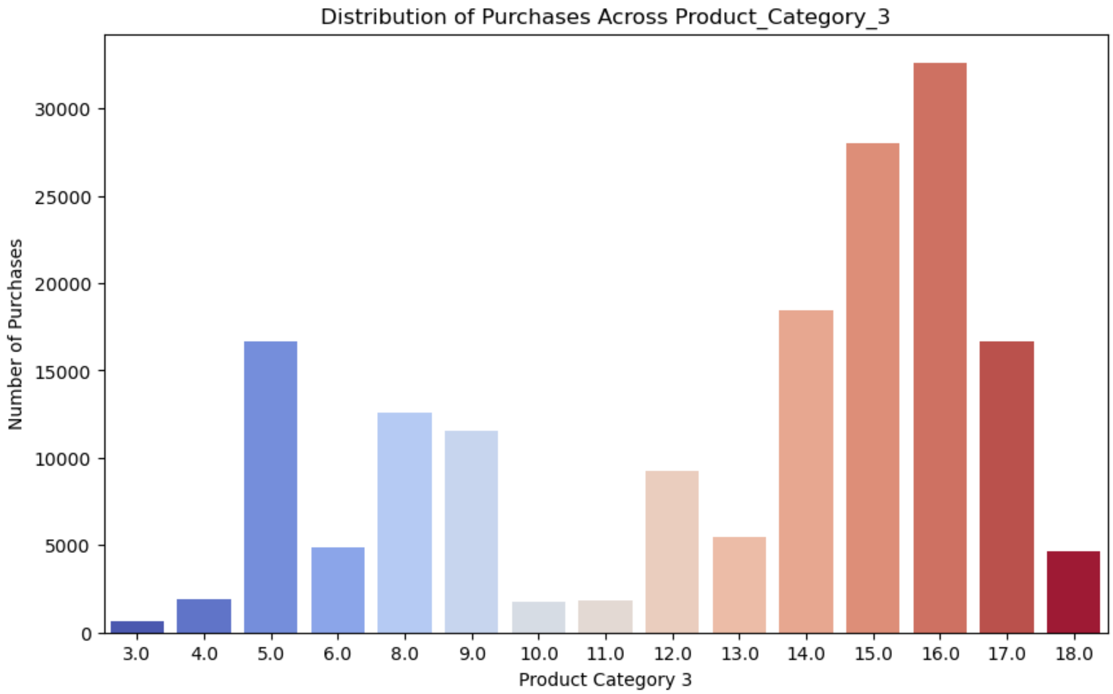
[ ]:

```

```

7.3 Product_Category_3

[28]:

```
# Distribution of Product_Category_3
plt.figure(figsize=(10, 6))
sns.countplot(x='Product_Category_3', hue='Product_Category_3', data=df,␣
    ↪palette='coolwarm', legend=False)
plt.title('Distribution of Purchases Across Product_Category_3')
plt.xlabel('Product Category 3')
plt.ylabel('Number of Purchases')
plt.show()
```



[ ]:

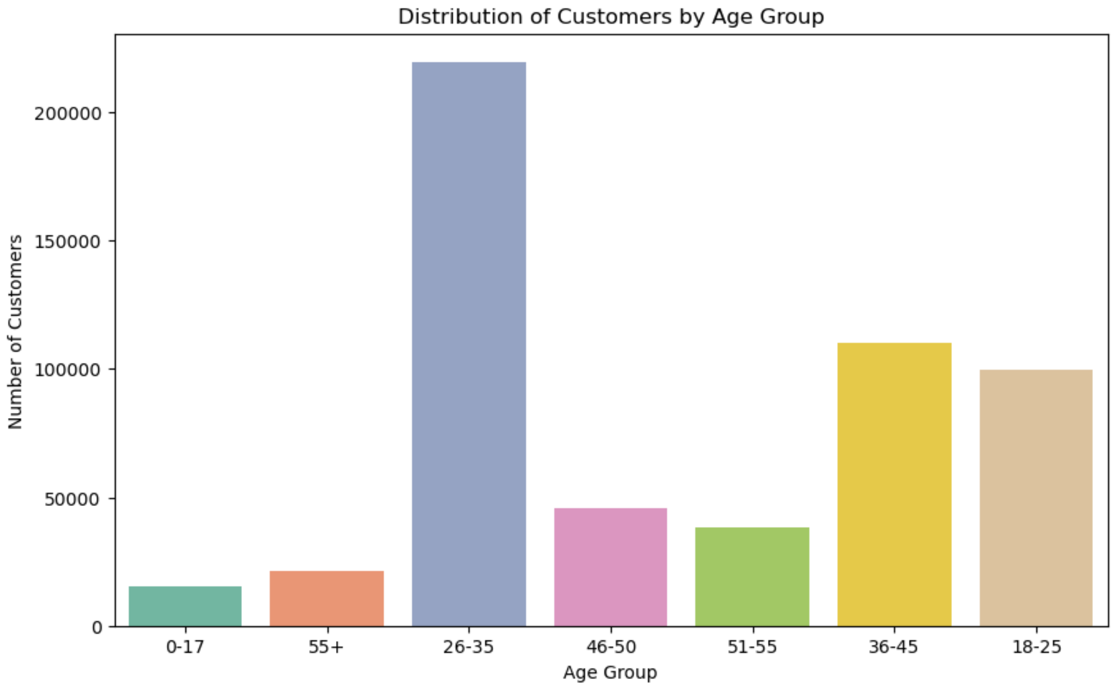
[ ]:

8 Q3 Customer Demographics:

8.1 Distribution by Age

[30]:

```
# Distribution of customers by Age
plt.figure(figsize=(10, 6))
sns.countplot(x='Age', hue='Age', data=df, palette='Set2', legend=False)
plt.title('Distribution of Customers by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Number of Customers')
plt.show()
```



[31]:
```python
# Calculate the number of customers in each age group
age_distribution = df['Age'].value_counts().sort_index()
print(age_distribution)
```

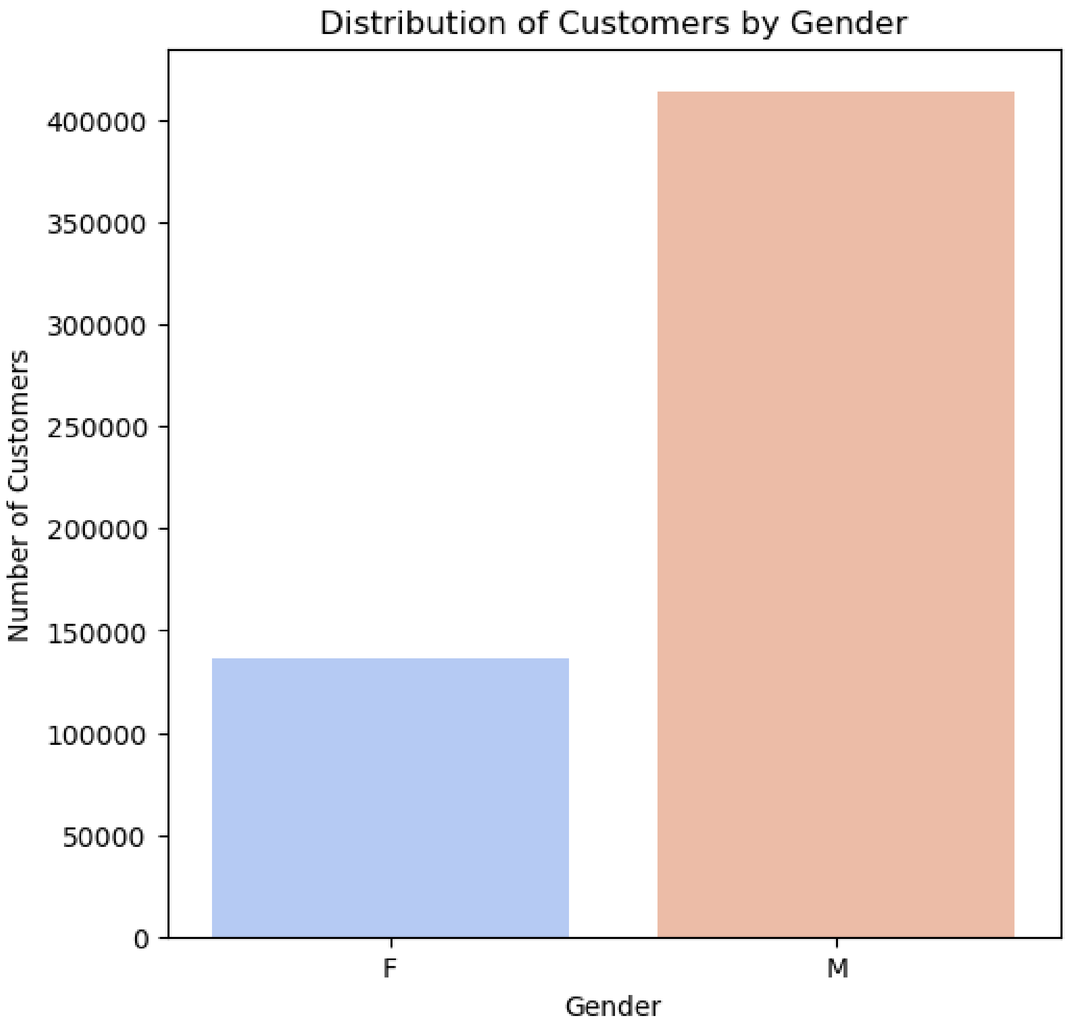
```
Age
0-17      15102
18-25     99660
26-35    219587
36-45    110013
46-50     45701
51-55     38501
55+       21504
Name: count, dtype: int64
```

[ ]:
```python

```

8.2 Distribution by Gender

[32]:
```python
# Distribution of customers by Gender
plt.figure(figsize=(6, 6))
sns.countplot(x='Gender', hue='Gender', data=df, palette='coolwarm',␣
 ↪legend=False)
plt.title('Distribution of Customers by Gender')
plt.xlabel('Gender')
```

```python
plt.ylabel('Number of Customers')
plt.show()
```



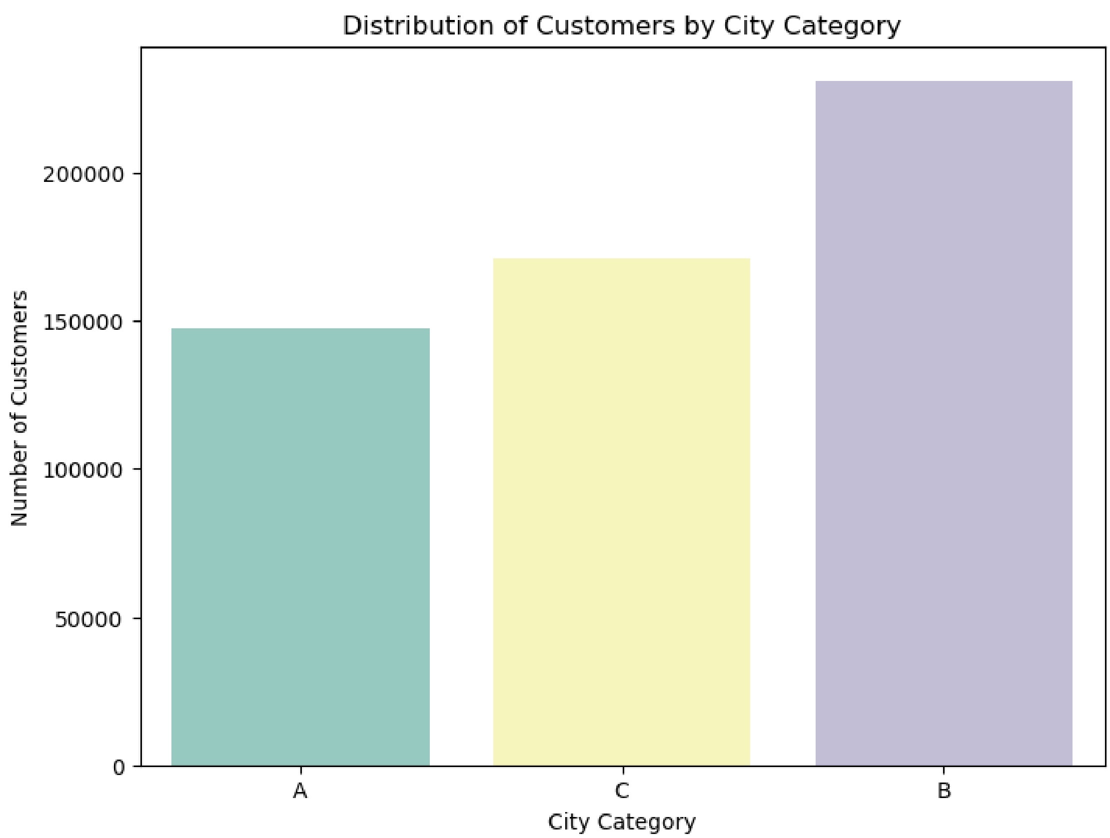
[33]:
```python
# Calculate the number of customers for each gender
gender_distribution = df['Gender'].value_counts()
print(gender_distribution)
```

```
Gender
M    414259
F    135809
Name: count, dtype: int64
```

[ ]:

8.3 Distribution by City Category

[34]:
```
# Distribution of customers by City Category
plt.figure(figsize=(8, 6))
sns.countplot(x='City_Category', hue='City_Category', data=df, palette='Set3',␣
↪legend=False)
plt.title('Distribution of Customers by City Category')
plt.xlabel('City Category')
plt.ylabel('Number of Customers')
plt.show()
```



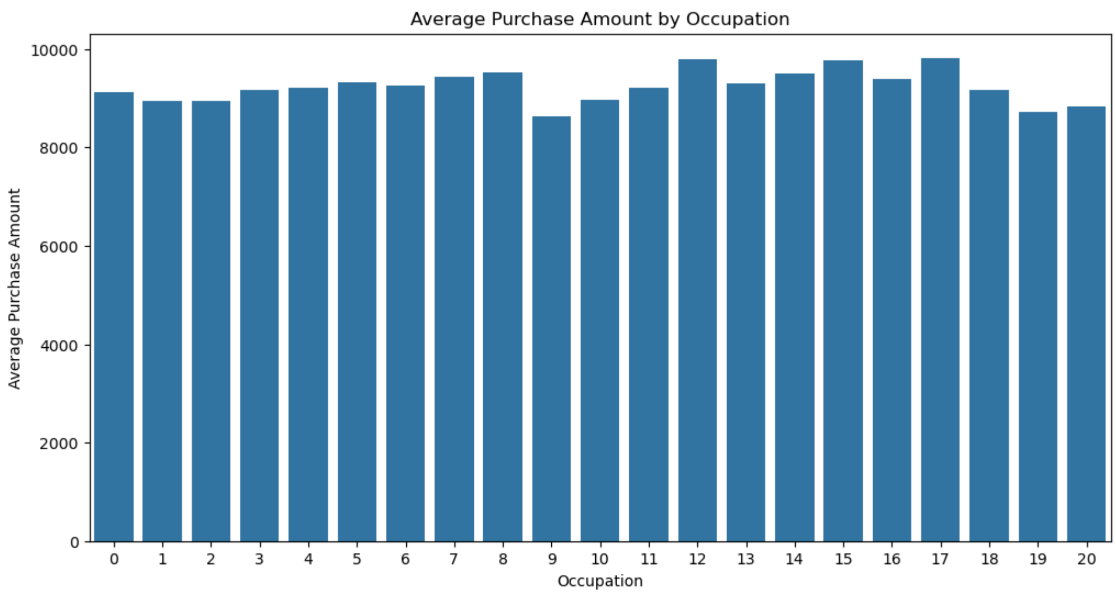
[35]:
```
# Calculate the number of customers in each city category
city_distribution = df['City_Category'].value_counts()
print(city_distribution)
```

```
City_Category
B    231173
C    171175
A    147720
Name: count, dtype: int64
```

[ ]:

9 Q4 Impact of Occupation:

[37]:
```
# Updated plot to avoid the warning
plt.figure(figsize=(12, 6))
sns.barplot(x=occupation_purchase.index, y=occupation_purchase.values)
plt.title('Average Purchase Amount by Occupation')
plt.xlabel('Occupation')
plt.ylabel('Average Purchase Amount')
plt.show()
```

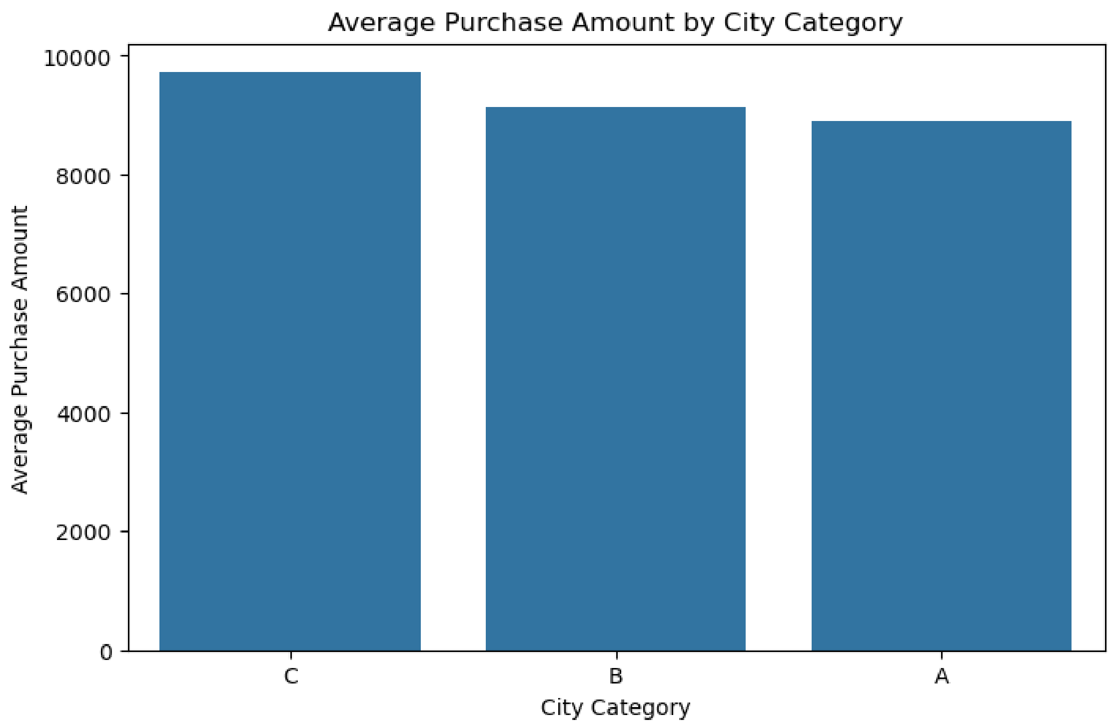


[ ]:

10 Q5 City-wise Purchase Behavior:

[40]:
```
# Calculate the average purchase amount for each city category
city_purchase = df.groupby('City_Category')['Purchase'].mean().
 ↪sort_values(ascending=False)

# Plot the results
plt.figure(figsize=(8, 5))
sns.barplot(x=city_purchase.index, y=city_purchase.values)
plt.title('Average Purchase Amount by City Category')
plt.xlabel('City Category')
```

```
plt.ylabel('Average Purchase Amount')
plt.show()
```



[45]:
```
city_purchase
```

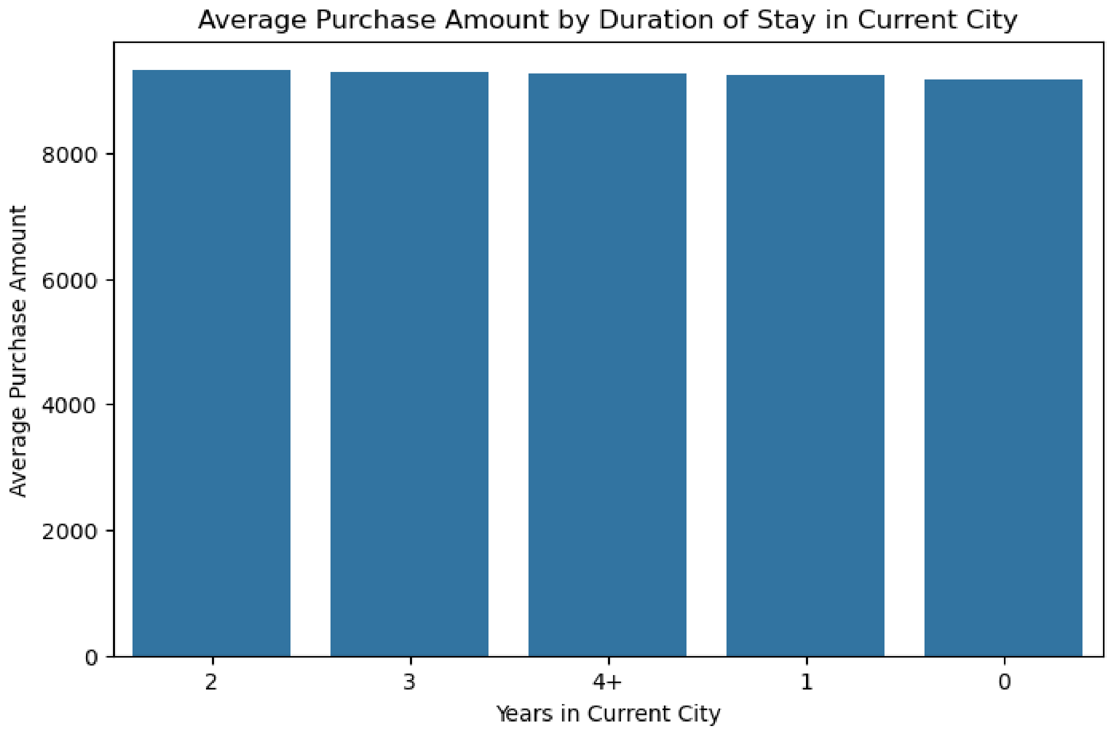
[45]:
```
City_Category
C    9719.920993
B    9151.300563
A    8911.939216
Name: Purchase, dtype: float64
```

[ ]:
```

```

11 Q6 Stay in Current City:

[43]:
```
# Calculate the average purchase amount by duration of stay in the current city
stay_purchase = df.groupby('Stay_In_Current_City_Years')['Purchase'].mean().
↪sort_values(ascending=False)

# Plot the results
plt.figure(figsize=(8, 5))
sns.barplot(x=stay_purchase.index, y=stay_purchase.values)
```

```python
plt.title('Average Purchase Amount by Duration of Stay in Current City')
plt.xlabel('Years in Current City')
plt.ylabel('Average Purchase Amount')
plt.show()
```



[44]:
```python
print(stay_purchase)
```

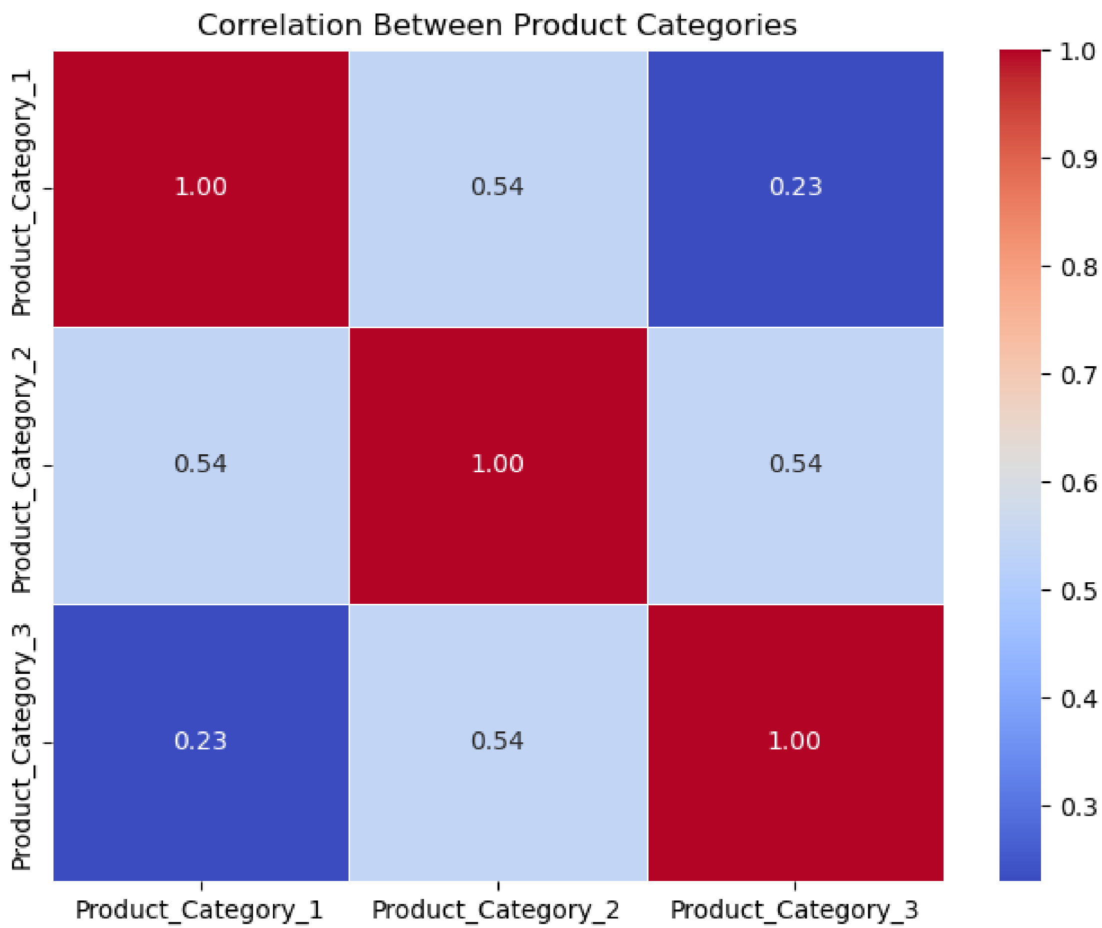
```
Stay_In_Current_City_Years
2     9320.429810
3     9286.904119
4+    9275.598872
1     9250.145923
0     9180.075123
Name: Purchase, dtype: float64
```

[ ]:
```python

```

12 Q7 Correlation Between Product Categories:

[46]:
```python
# Select the relevant product category columns
product_categories = df[['Product_Category_1', 'Product_Category_2',␣
↪'Product_Category_3']]
```

```python
# Calculate the correlation matrix
correlation_matrix = product_categories.corr()

# Plot the correlation matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",␣
↪linewidths=0.5)
plt.title('Correlation Between Product Categories')
plt.show()
```



[47]:
```python
print(correlation_matrix)
```

```
                    Product_Category_1  Product_Category_2  Product_Category_3
Product_Category_1            1.000000            0.540583            0.229678
Product_Category_2            0.540583            1.000000            0.543649
Product_Category_3            0.229678            0.543649            1.000000
```

[ ]:

```python

```

13 Q8 Outlier Analysis:

13.0.1 Identifying Outliers Using IQR

[56]:

```python
# Calculate Q1 (25th percentile) and Q3 (75th percentile)
Q1 = df['Purchase'].quantile(0.25)
Q3 = df['Purchase'].quantile(0.75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = df[(df['Purchase'] < lower_bound) | (df['Purchase'] > upper_bound)]

num_outliers = outliers.shape[0]
print(f"Number of outliers in the Purchase data: {num_outliers}")
```

```
Number of outliers in the Purchase data: 2677
```

[ ]:

```python

```

13.0.2 Assessing the Impact of Outliers

[59]:

```python
# Calculate mean and median before removing outliers
mean_before = df['Purchase'].mean()
median_before = df['Purchase'].median()
df_no_outliers = df[(df['Purchase'] >= lower_bound) & (df['Purchase'] <=
    upper_bound)]

# Calculate mean and median after removing outliers
mean_after = df_no_outliers['Purchase'].mean()
median_after = df_no_outliers['Purchase'].median()
```

[60]:

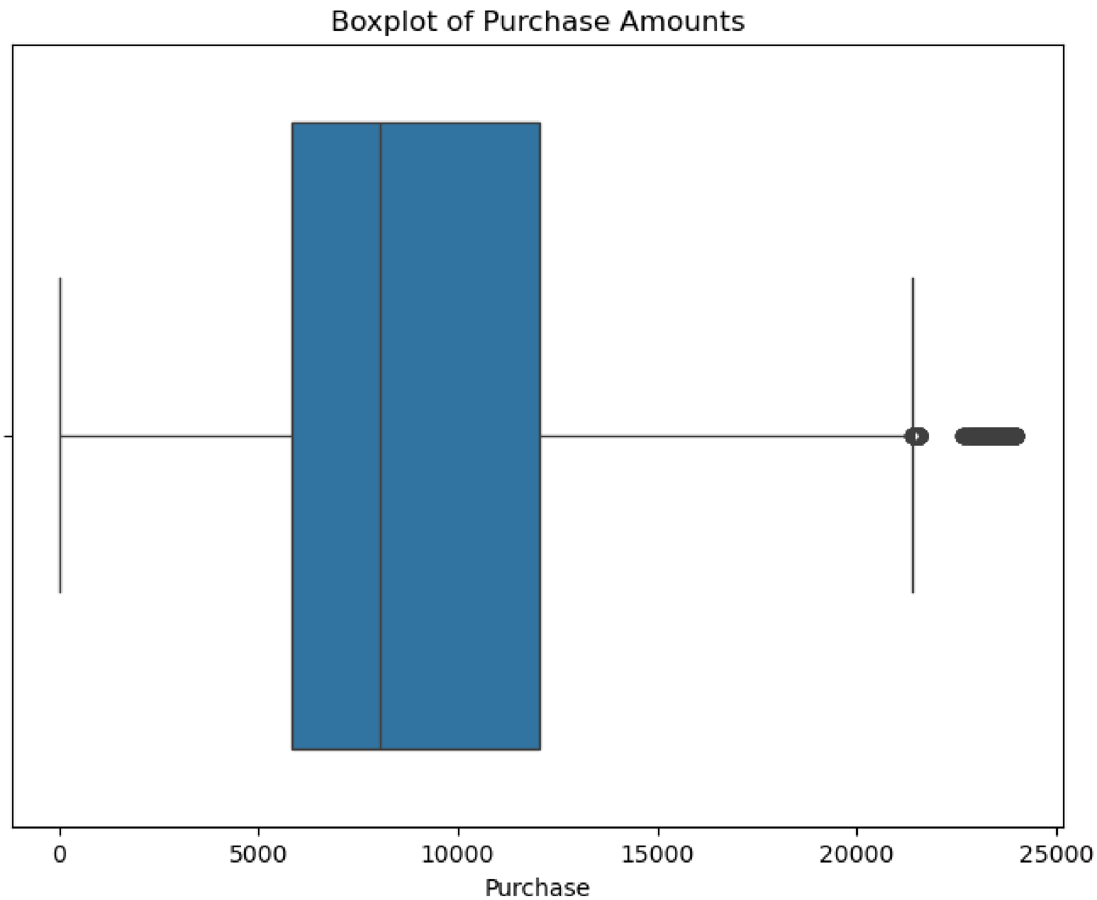
```python
# Display the results
print(f"Mean Purchase before removing outliers: {mean_before:.2f}")
print(f"Median Purchase before removing outliers: {median_before:.2f}")
print(f"Mean Purchase after removing outliers: {mean_after:.2f}")
print(f"Median Purchase after removing outliers: {median_after:.2f}")
```

```
Mean Purchase before removing outliers: 9263.97
Median Purchase before removing outliers: 8047.00
Mean Purchase after removing outliers: 9195.63
Median Purchase after removing outliers: 8038.00
```

[ ]:

13.0.3 Visualizing Outliers with a Boxplot

[61]:
```
# Plot a boxplot to visualize outliers
plt.figure(figsize=(8, 6))
sns.boxplot(x=df['Purchase'])
plt.title('Boxplot of Purchase Amounts')
plt.show()
```



[ ]:

14 Q9 Gender-wise Product Preferences:

14.0.1 Grouping and Calculating Preferences

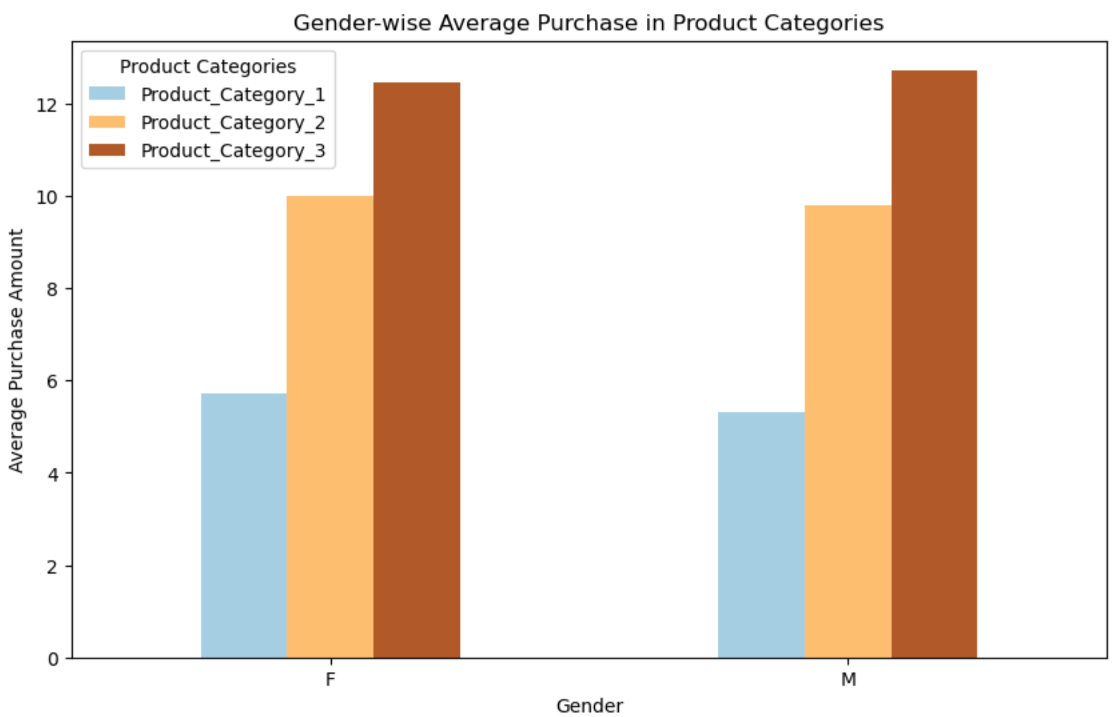
[63]:
```
gender_product_preferences = df.groupby('Gender')[['Product_Category_1',␣
↪'Product_Category_2', 'Product_Category_3']].mean()
print(gender_product_preferences)
```

```
        Product_Category_1  Product_Category_2  Product_Category_3
Gender
F                 5.717714           10.009166           12.453556
M                 5.301512            9.788729           12.730699
```

[ ]:

14.0.2 Visualizing Gender-wise Preferences

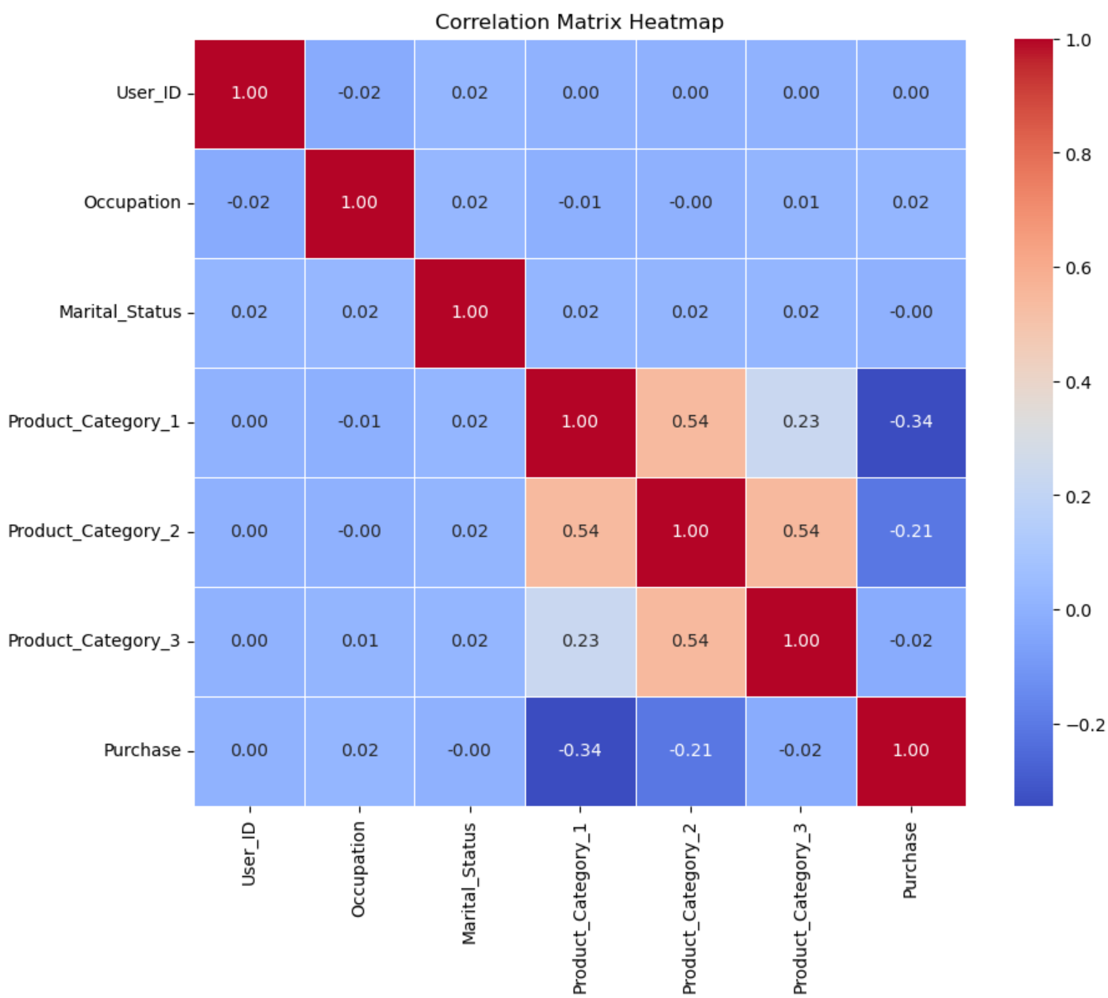
[64]:
```
# Plotting the preferences
gender_product_preferences.plot(kind='bar', figsize=(10, 6), colormap='Paired')
plt.title('Gender-wise Average Purchase in Product Categories')
plt.xlabel('Gender')
plt.ylabel('Average Purchase Amount')
plt.xticks(rotation=0)
plt.legend(title='Product Categories')
plt.show()
```



```
[ ]:
```

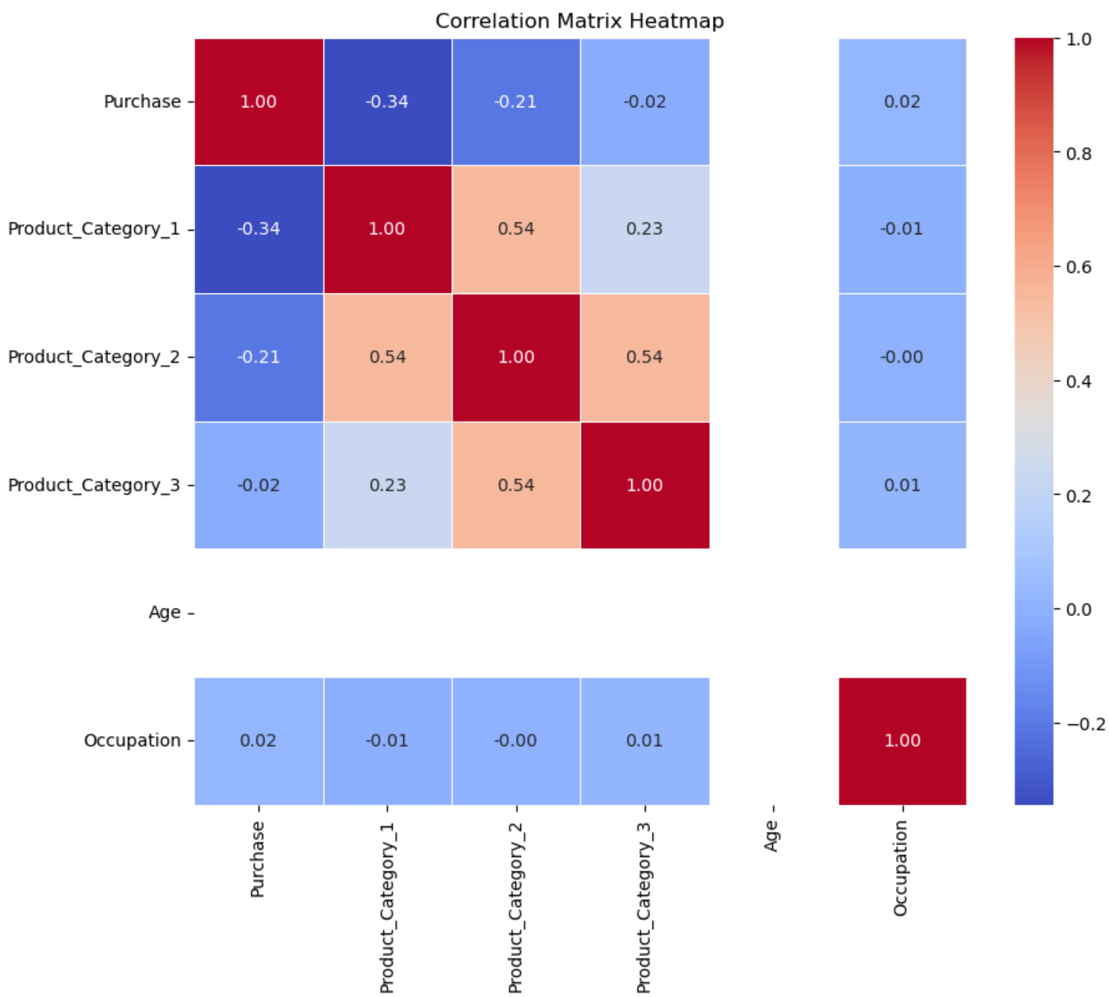
15 Q10 Advanced Insights:

```
[66]: numeric_df = df.select_dtypes(include=['float64', 'int64'])
correlation_matrix = numeric_df.corr()

# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",␣
 ↪linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



[69]:
```
age_mapping = {
    '0-17': 0,
    '18-25': 1,
    '26-35': 2,
    '36-45': 3,
    '46-50': 4,
    '51-55': 5,
    '55+': 6
}
df['Age'] = df['Age'].map(age_mapping)

numeric_columns = ['Purchase', 'Product_Category_1', 'Product_Category_2',␣
 ↪'Product_Category_3', 'Age', 'Occupation']
numeric_df = df[numeric_columns]

correlation_matrix = numeric_df.corr()
```

[70]:
```
# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",␣
 ↪linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```


Correlation Matrix Heatmap
Purchase
Product_Category_1
Product_Category_2
Product_Category_3
Age
Occupation
1.00
-0.34
-0.21
-0.02
0.02
0.54
0.23
-0.01
-0.00
0.01
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

```
[ ]:
```