# eda-of-friday-data-analytics

August 13, 2023

## 1 Black Friday Dataset EDA Dataset Exploratory Data Analysis

## 2 Author: *Iqra Iqbal*

**Email**: iqra.iqbal4143@gmail.com

[109]:
```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[110]:
```
#import Dataset
df_train=pd.read_csv("train.csv")
```

[112]:
```
#checking Top 5
df_train.head(5)
```

[112]:

| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category_1 | Product_Category_2 | Product_Category_3 | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1000001 | P00069042 | F | 0-17 | 10 | A | 2 | 0 | 3 | NaN | NaN | 8370 |
| 1 | 1000001 | P00248942 | F | 0-17 | 10 | A | 2 | 0 | 1 | 6.0 | 14.0 | 15200 |
| 2 | 1000001 | P00087842 | F | 0-17 | 10 | A | 2 | 0 | 12 | NaN | NaN | 1422 |
| 3 | 1000001 | P00085442 | F | 0-17 | 10 | A | 2 | 0 | 12 | 14.0 | NaN | 1057 |
| 4 | 1000002 | P00285442 | M | 55+ | 16 | C | 4+ | 0 | 8 | | | |

```
4                NaN                NaN       7969
```

[113]:
```
df_train.shape
```

[113]:
```
(550068, 12)
```

## 3 What is Data about

**A retail company "ABC Private Limited" wants to understand the customer purchase behaviour (specifically, purchase amount) against various products of different categories. They have shared purchase summary of various customers for selected high volume products from last month. The data set also contains customer demographics (age, gender, marital status, city_type, stay_in_current_city), product details (product_id and product category) and Total purchase_amount from last month.

[208]:
```
df=df_train
```

[115]:
```
df.columns
```

[115]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3', 'Purchase'],
      dtype='object')
```

[116]:
```
#Basic
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          376430 non-null  float64
 10  Product_Category_3          166821 non-null  float64
 11  Purchase                    550068 non-null  int64
dtypes: float64(2), int64(5), object(5)
memory usage: 50.4+ MB
```

[117]:
```
df.describe()
```

[117]:

| | User_ID | Occupation | Marital_Status | Product_Category_1 | Product_Category_2 | Product_Category_3 | Purchase |
|---|---|---|---|---|---|---|---|
| count | 5.500680e+05 | 550068.000000 | 550068.000000 | 550068.000000 | 376430.000000 | 166821.000000 | 550068.000000 |
| mean | 1.003029e+06 | 8.076707 | 0.409653 | 5.404270 | 9.842329 | 12.668243 | 9263.968713 |
| std | 1.727592e+03 | 6.522660 | 0.491770 | 3.936211 | 5.086590 | 4.125338 | 5023.065394 |
| min | 1.000001e+06 | 0.000000 | 0.000000 | 1.000000 | 2.000000 | 3.000000 | 12.000000 |
| 25% | 1.001516e+06 | 2.000000 | 0.000000 | 1.000000 | 5.000000 | 9.000000 | 5823.000000 |
| 50% | 1.003077e+06 | 7.000000 | 0.000000 | 5.000000 | 9.000000 | 14.000000 | 8047.000000 |
| 75% | 1.004478e+06 | 14.000000 | 1.000000 | 8.000000 | 15.000000 | 16.000000 | 12054.000000 |
| max | 1.006040e+06 | 20.000000 | 1.000000 | 20.000000 | 18.000000 | 18.000000 | 23961.000000 |

[118]:
```
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          376430 non-null  float64
 10  Product_Category_3          166821 non-null  float64
 11  Purchase                    550068 non-null  int64
dtypes: float64(2), int64(5), object(5)
memory usage: 50.4+ MB
```

[119]:
```
df.drop(["User_ID"],axis=1,inplace=True)
```

[120]:
```
#Handling catagorical feature Gender
# Using replace function
df["Gender"].replace({"F": 0, "M": 1}, inplace=True)
```

[121]:
```
df.head()
```

[121]:

| | Product_ID | Gender | Age | Occupation | City_Category | \ |
|---|---|---|---|---|---|---|
| 0 | P00069042 | 0 | 0-17 | 10 | A | |
| 1 | P00248942 | 0 | 0-17 | 10 | A | |
| 2 | P00087842 | 0 | 0-17 | 10 | A | |
| 3 | P00085442 | 0 | 0-17 | 10 | A | |
| 4 | P00285442 | 1 | 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 |

[122]:
```
#Handling catagorical feature Age
df["Age"].unique()
```

[122]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

[123]:
```
df["Age"].unique()
```

[123]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

[124]:
```
#we can do labeling for age column by maping and labeling  method
age_mapping = {
    '0-17': 1,
    '18-25': 2,
    '26-35': 3,
    '36-45': 4,
    '46-50': 5,
    '55+': 6
}
```

```
df["Age"] = df["Age"].map(age_mapping)

print(df)
```

```
       Product_ID  Gender  Age  Occupation City_Category  \
0       P00069042       0  1.0          10             A
1       P00248942       0  1.0          10             A
2       P00087842       0  1.0          10             A
3       P00085442       0  1.0          10             A
4       P00285442       1  6.0          16             C
…             …     …  …          …            …
550063  P00372445       1  NaN          13             B
550064  P00375436       0  3.0           1             C
550065  P00375436       0  3.0          15             B
550066  P00375436       0  6.0           1             C
550067  P00371644       0  5.0           0             B

       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
…                             …             …                   …
550063                          1               1                  20
550064                          3               0                  20
550065                         4+               1                  20
550066                          2               0                  20
550067                         4+               1                  20

        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
…                     …                   …        …
550063                 NaN                 NaN       368
550064                 NaN                 NaN       371
550065                 NaN                 NaN       137
550066                 NaN                 NaN       365
550067                 NaN                 NaN       490

[550068 rows x 11 columns]
```

[125]:
```
df.head()
```

[125]:
```
  Product_ID  Gender  Age  Occupation City_Category  \
0  P00069042       0  1.0          10             A
1  P00248942       0  1.0          10             A
2  P00087842       0  1.0          10             A
3  P00085442       0  1.0          10             A
4  P00285442       1  6.0          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
```

[126]:
```
#Second Technique
from sklearn.preprocessing import LabelEncoder
import pandas as pd

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Apply label encoding to the "Gender" column
df["Age"] = label_encoder.fit_transform(df["Age"])

print(df)
```

```
       Product_ID  Gender  Age  Occupation City_Category  \
0       P00069042       0    0          10             A
1       P00248942       0    0          10             A
2       P00087842       0    0          10             A
3       P00085442       0    0          10             A
4       P00285442       1    5          16             C
...           ...     ...  ...         ...           ...
550063  P00372445       1    6          13             B
550064  P00375436       0    2           1             C
550065  P00375436       0    2          15             B
550066  P00375436       0    5           1             C
550067  P00371644       0    4           0             B

       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
…                              …                 …                   …
550063                           1                1                  20
550064                           3                0                  20
550065                          4+                1                  20
550066                           2                0                  20
550067                          4+                1                  20

        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
…                     …                    …        …
550063                 NaN                 NaN       368
550064                 NaN                 NaN       371
550065                 NaN                 NaN       137
550066                 NaN                 NaN       365
550067                 NaN                 NaN       490

[550068 rows x 11 columns]
```

[127]:
```
df.head()
```

[127]:
```
   Product_ID  Gender  Age  Occupation City_Category  \
0  P00069042       0    0          10             A
1  P00248942       0    0          10             A
2  P00087842       0    0          10             A
3  P00085442       0    0          10             A
4  P00285442       1    5          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
```

[128]:
```
df["City_Category"].unique()
```

[128]:
```
array(['A', 'C', 'B'], dtype=object)
```

[129]:
```
#Fixing catagorical city   catgory
df_city=pd.get_dummies(df["City_Category"],drop_first=True)
```

[130]:
```
df_city.head()
```

[130]:
```
   B  C
0  0  0
1  0  0
2  0  0
3  0  0
4  0  1
```

[133]:
```
df = pd.concat([df,df_city],axis=1)
df.head()
```

[133]:
```
  Product_ID  Gender  Age  Occupation City_Category  \
0  P00069042       0    0          10             A
1  P00248942       0    0          10             A
2  P00087842       0    0          10             A
3  P00085442       0    0          10             A
4  P00285442       1    5          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  B  C
0                 NaN                 NaN      8370  0  0
1                 6.0                14.0     15200  0  0
2                 NaN                 NaN      1422  0  0
3                14.0                 NaN      1057  0  0
4                 NaN                 NaN      7969  0  1
```

[159]:
```
df=df.drop("City_Category",axis=1)
```

[160]:
```
df.isnull().sum()
```

[160]:
```
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category_1            0
Product_Category_2            0
Product_Category_3            0
Purchase                      0
B                             0
C                             0
dtype: int64
```

[136]:
```
#Replacing missing values
df["Product_Category_2"].unique()
```

[136]:
```
array([nan,  6., 14.,  2.,  8., 15., 16., 11.,  5.,  3.,  4., 12.,  9.,
       10., 17., 13.,  7., 18.])
```

[137]:
```
#replacing values with mode Product_Category_2
df["Product_Category_2"]=df["Product_Category_2"].
 ↪fillna(df["Product_Category_2"]).mode()[0]
```

[138]:
```
df["Product_Category_2"].isnull().sum()
```

[138]:
```
0
```

[139]:
```
#replacing values with mode Product_Category_23
df["Product_Category_3"]=df["Product_Category_3"].
 ↪fillna(df["Product_Category_3"]).mode()[0]
```

[140]:
```
df["Product_Category_3"].isnull().sum()
```

[140]:
```
0
```

[162]:
```
df.shape
```

[162]:
```
(550068, 12)
```

[163]:
```
df["Stay_In_Current_City_Years"].unique()
```

[163]:
```
array([2, 4, 3, 1, 0])
```

[166]:
```
# # Ensure the column contains string values
df["Stay_In_Current_City_Years"] = df["Stay_In_Current_City_Years"].astype(str)
```

```
# Replace "+" with space
df["Stay_In_Current_City_Years"] = df["Stay_In_Current_City_Years"].str.
 ↪replace("+", " ")

# Convert the column to integer data type
df["Stay_In_Current_City_Years"] = df["Stay_In_Current_City_Years"].astype(int)
```

```
<ipython-input-166-7aec9fbfc767>:6: FutureWarning: The default value of regex
will change from True to False in a future version. In addition, single
character regular expressions will *not* be treated as literal strings when
regex=True.
  df["Stay_In_Current_City_Years"] =
df["Stay_In_Current_City_Years"].str.replace("+", " ")
```

[167]:
```
df.head()
```

[167]:

| | Product_ID | Gender | Age | Occupation | Stay_In_Current_City_Years | \ |
|---|---|---|---|---|---|---|
| 0 | P00069042 | 0 | 0 | 10 | 2 | |
| 1 | P00248942 | 0 | 0 | 10 | 2 | |
| 2 | P00087842 | 0 | 0 | 10 | 2 | |
| 3 | P00085442 | 0 | 0 | 10 | 2 | |
| 4 | P00285442 | 1 | 5 | 16 | 4 | |

| | Marital_Status | Product_Category_1 | Product_Category_2 | Product_Category_3 | \ |
|---|---|---|---|---|---|
| 0 | 0 | 3 | 8.0 | 16.0 | |
| 1 | 0 | 1 | 8.0 | 16.0 | |
| 2 | 0 | 12 | 8.0 | 16.0 | |
| 3 | 0 | 12 | 8.0 | 16.0 | |
| 4 | 0 | 8 | 8.0 | 16.0 | |

| | Purchase | B | C |
|---|---|---|---|
| 0 | 8370 | 0 | 0 |
| 1 | 15200 | 0 | 0 |
| 2 | 1422 | 0 | 0 |
| 3 | 1057 | 0 | 0 |
| 4 | 7969 | 0 | 1 |

[168]:
```
#convert object into integer
df["Stay_In_Current_City_Years"]=df["Stay_In_Current_City_Years"].astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Product_ID                  550068 non-null  object
```

```
 1   Gender                      550068 non-null  int64
 2   Age                         550068 non-null  int64
 3   Occupation                  550068 non-null  int64
 4   Stay_In_Current_City_Years  550068 non-null  int64
 5   Marital_Status              550068 non-null  int64
 6   Product_Category_1          550068 non-null  int64
 7   Product_Category_2          550068 non-null  float64
 8   Product_Category_3          550068 non-null  float64
 9   Purchase                    550068 non-null  int64
 10  B                           550068 non-null  int64
 11  C                           550068 non-null  int64
dtypes: float64(2), int64(9), object(1)
memory usage: 50.4+ MB
```

[148]:
```
df["B"]=df["B"].astype(int)
df["C"]=df["C"].astype(int)
```

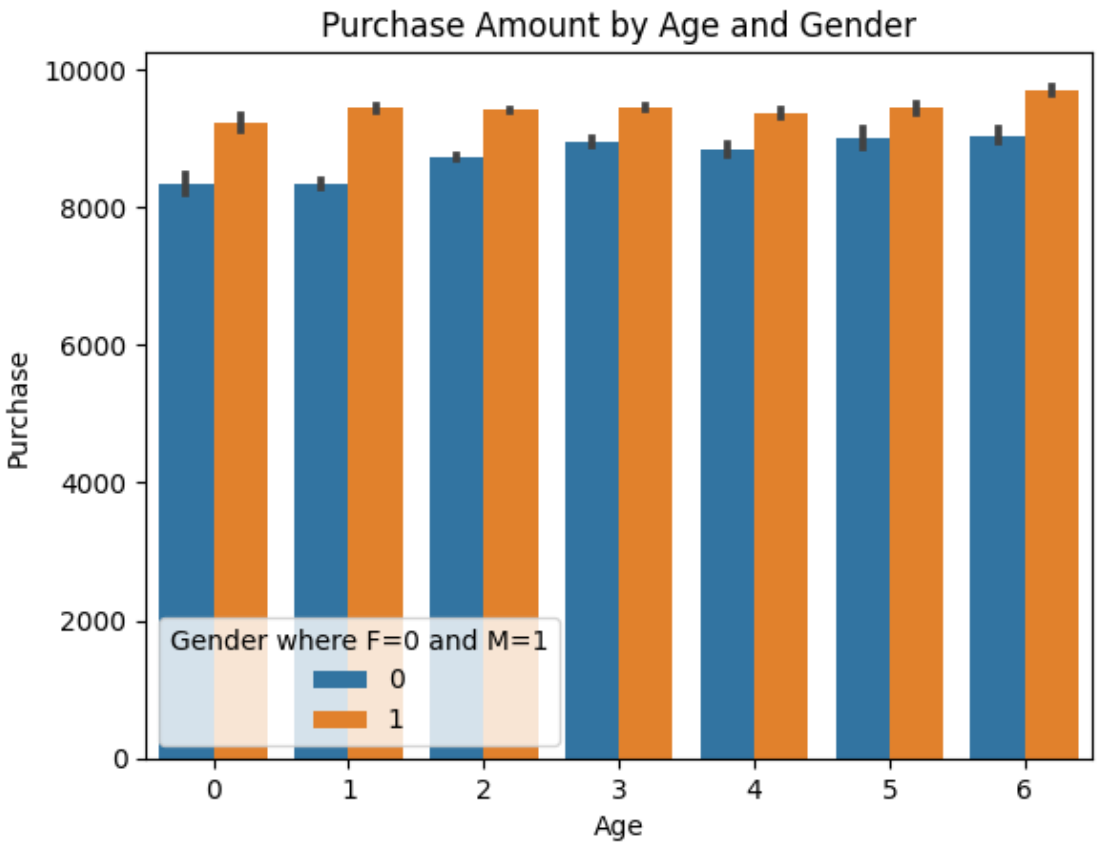
## 4 Visulization

[169]:
```
df.columns
```

[169]:
```
Index(['Product_ID', 'Gender', 'Age', 'Occupation',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3', 'Purchase', 'B', 'C'],
      dtype='object')
```

[172]:
```
#purchases on bases of age F=0 and M=1
sns.barplot(x="Age", y="Purchase", hue="Gender", data=df)

plt.title('Purchase Amount by Age and Gender')
plt.xlabel('Age')
plt.ylabel('Purchase')
plt.legend(title='Gender where F=0 and M=1')
```

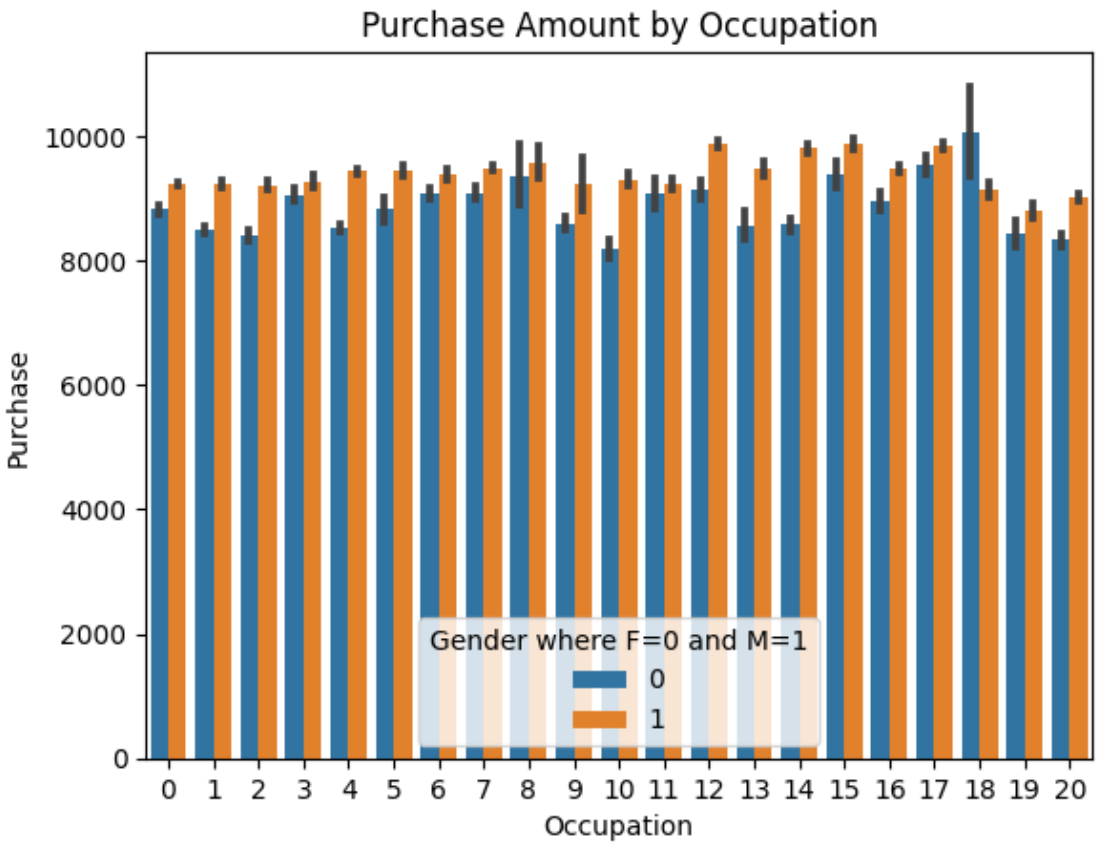
[172]:
```
<matplotlib.legend.Legend at 0x78da2c05c190>
```



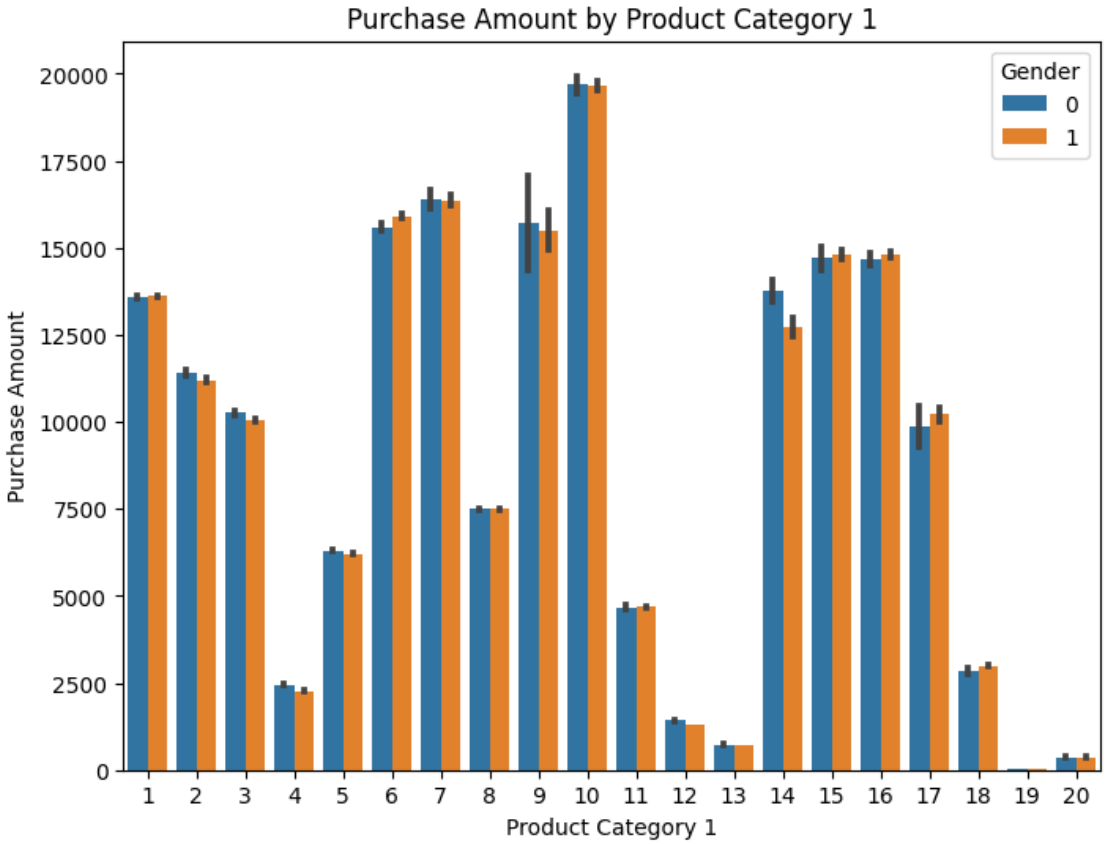

**Purchase of mens are high then women**

[173]:
```
#visulize the purchase with occupation
sns.barplot(x="Occupation", y="Purchase", hue="Gender", data=df)

plt.title('Purchase Amount by Occupation')
plt.xlabel('Occupation')
plt.ylabel('Purchase')
plt.legend(title='Gender where F=0 and M=1')
```

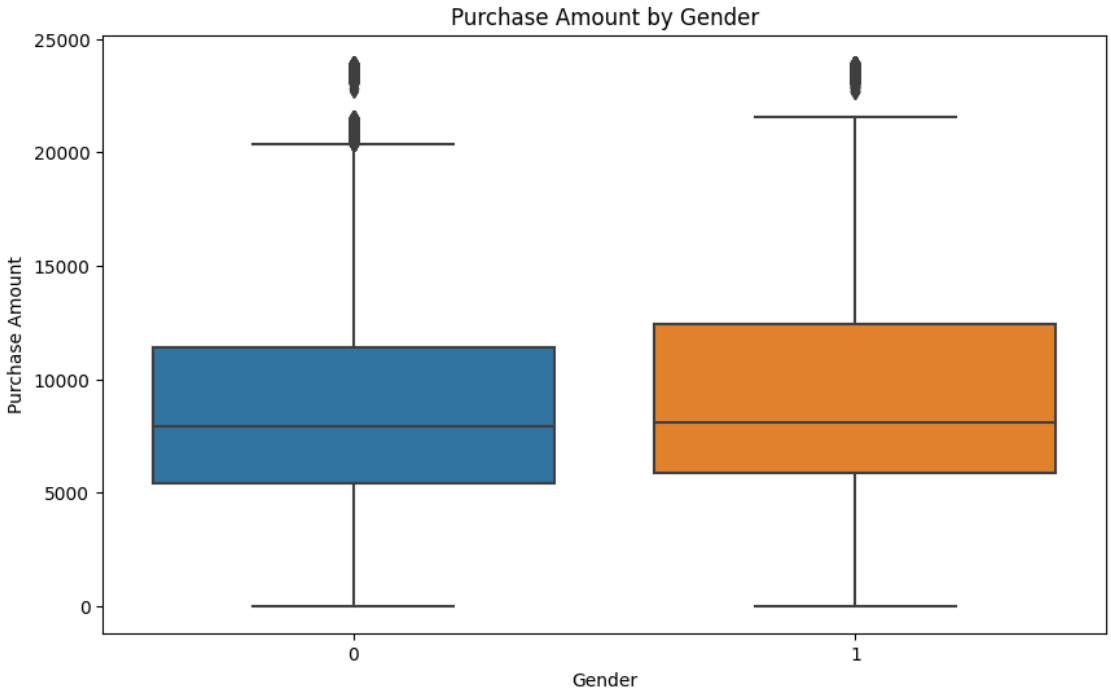
[173]: <matplotlib.legend.Legend at 0x78da2c0a5780>



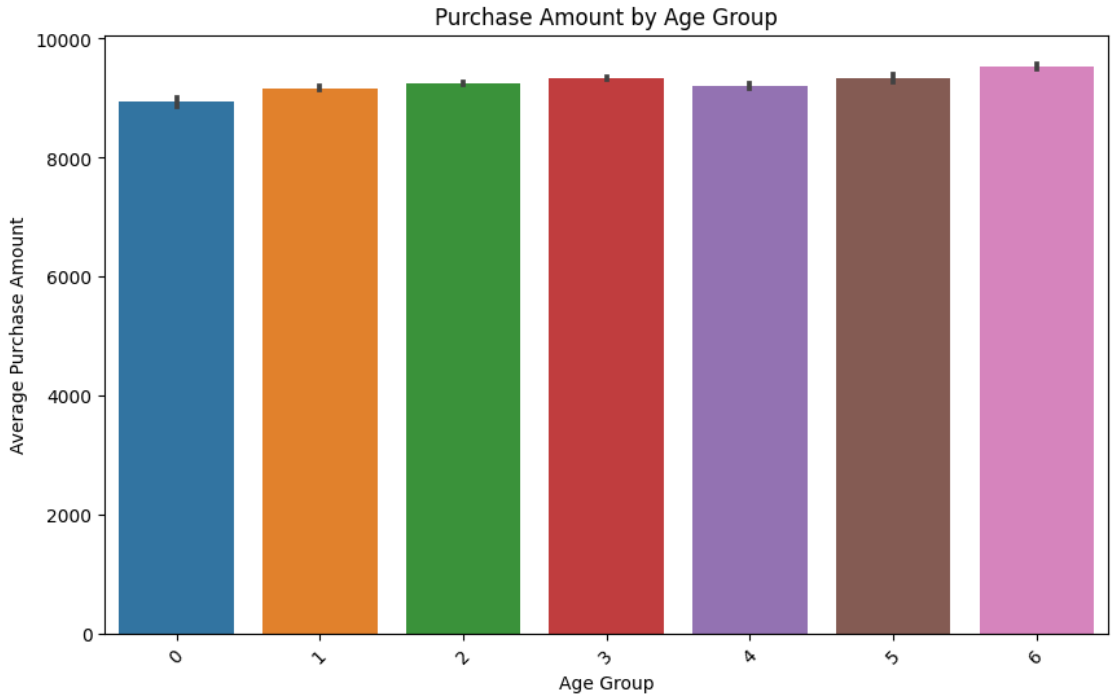

[177]:
```
#Graph with Product_Category_1
plt.figure(figsize=(8, 6))
sns.barplot(x='Product_Category_1', y='Purchase',hue="Gender", data=df)

plt.title('Purchase Amount by Product Category 1')
plt.xlabel('Product Category 1')
plt.ylabel('Purchase Amount')

plt.show()
```



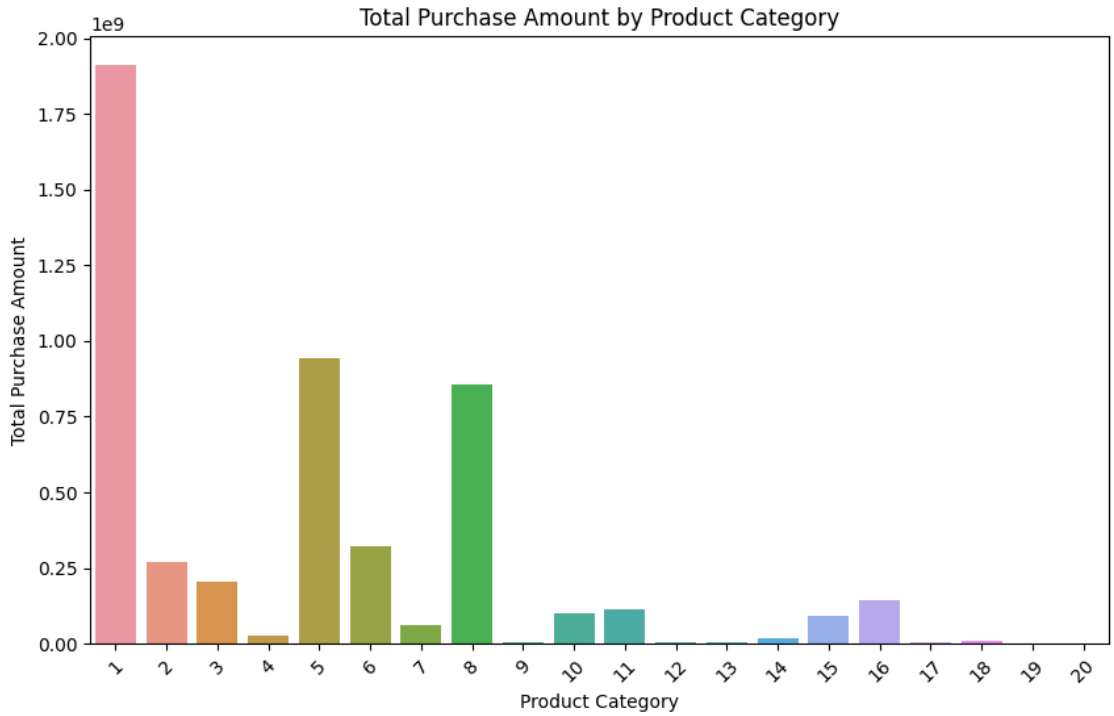

[182]:
```
# Gender and Purchase Analysis box plot
plt.figure(figsize=(10, 6))
sns.boxplot(x="Gender", y="Purchase", data=df)
plt.title("Purchase Amount by Gender")
plt.xlabel("Gender")
plt.ylabel("Purchase Amount")
plt.show()
```



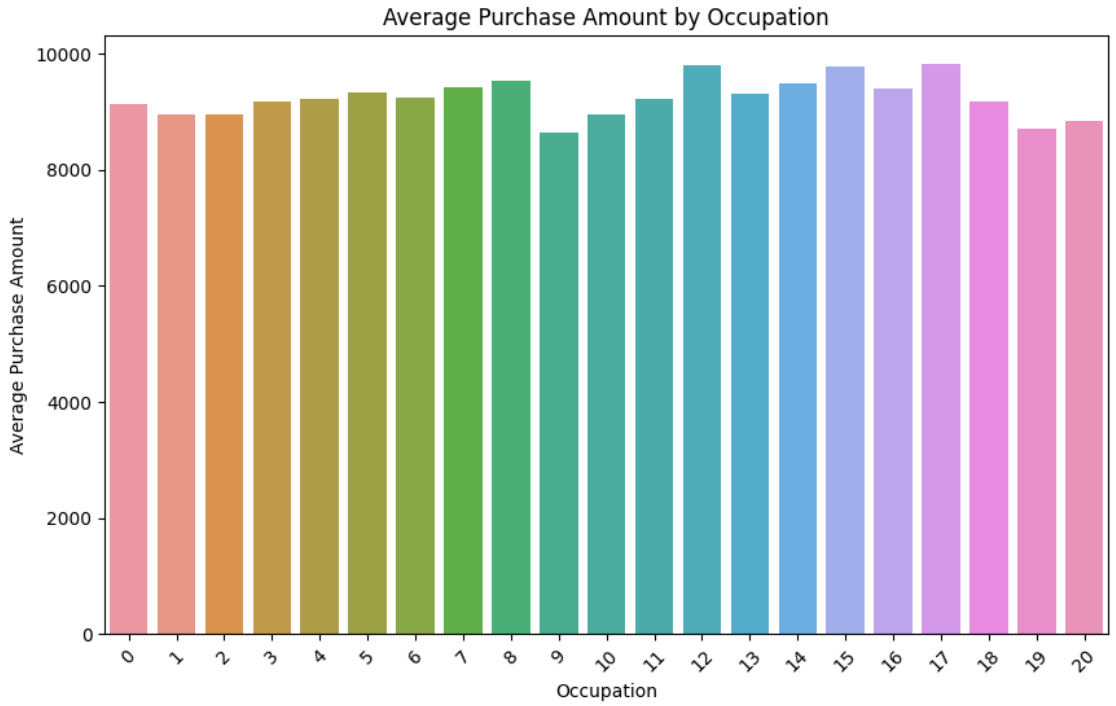

[183]:
```
# Age Group Analysis
plt.figure(figsize=(10, 6))
sns.barplot(x="Age", y="Purchase", data=df)
plt.title("Purchase Amount by Age Group")
plt.xlabel("Age Group")
plt.ylabel("Average Purchase Amount")
plt.xticks(rotation=45)
plt.show()
```



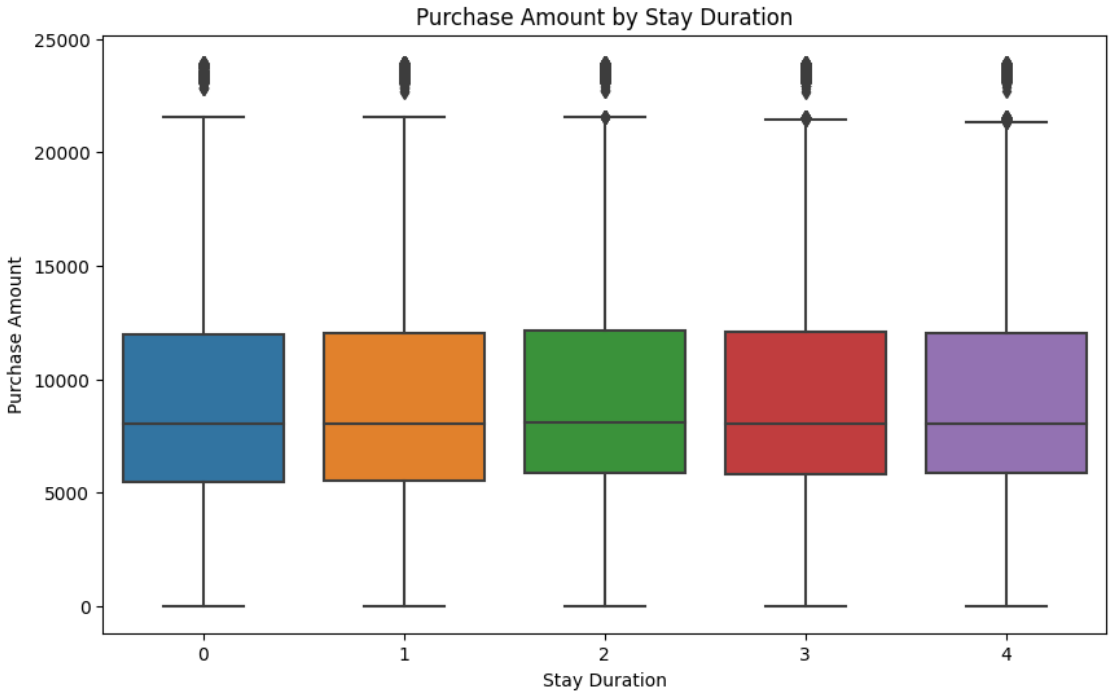

[185]:
```
# Product Category Analysis
product_category_totals = df.groupby("Product_Category_1")["Purchase"].sum().
  ↪reset_index()
plt.figure(figsize=(10, 6))
sns.barplot(x="Product_Category_1", y="Purchase", data=product_category_totals)
plt.title("Total Purchase Amount by Product Category")
plt.xlabel("Product Category")
plt.ylabel("Total Purchase Amount")
plt.xticks(rotation=45)
plt.show()
```



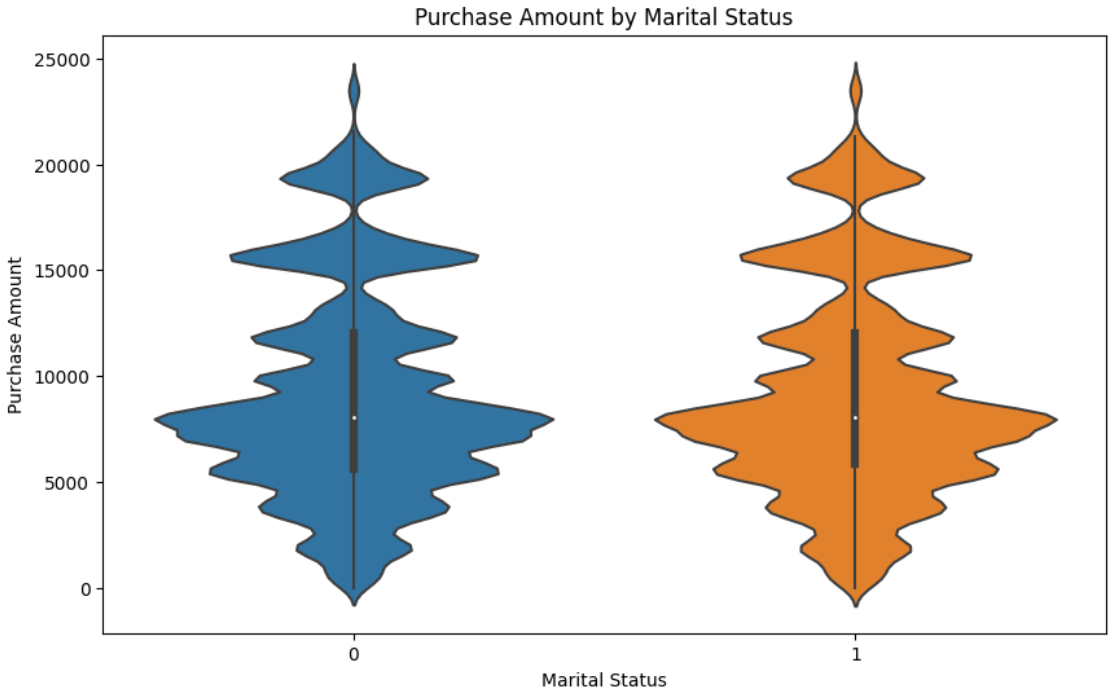

[228]:
```
# Occupation Analysis
occupation_purchase = df.groupby("Occupation")["Purchase"].mean().reset_index()
plt.figure(figsize=(10, 6))
sns.barplot(x="Occupation", y="Purchase", data=occupation_purchase )
plt.title("Average Purchase Amount by Occupation")
plt.xlabel("Occupation")
plt.ylabel("Average Purchase Amount")
plt.xticks(rotation=45)
plt.show()
```




[187]:
```
# City Stay Duration Analysis
plt.figure(figsize=(10, 6))
sns.boxplot(x="Stay_In_Current_City_Years", y="Purchase", data=df)
plt.title("Purchase Amount by Stay Duration")
plt.xlabel("Stay Duration")
plt.ylabel("Purchase Amount")
plt.show()
```

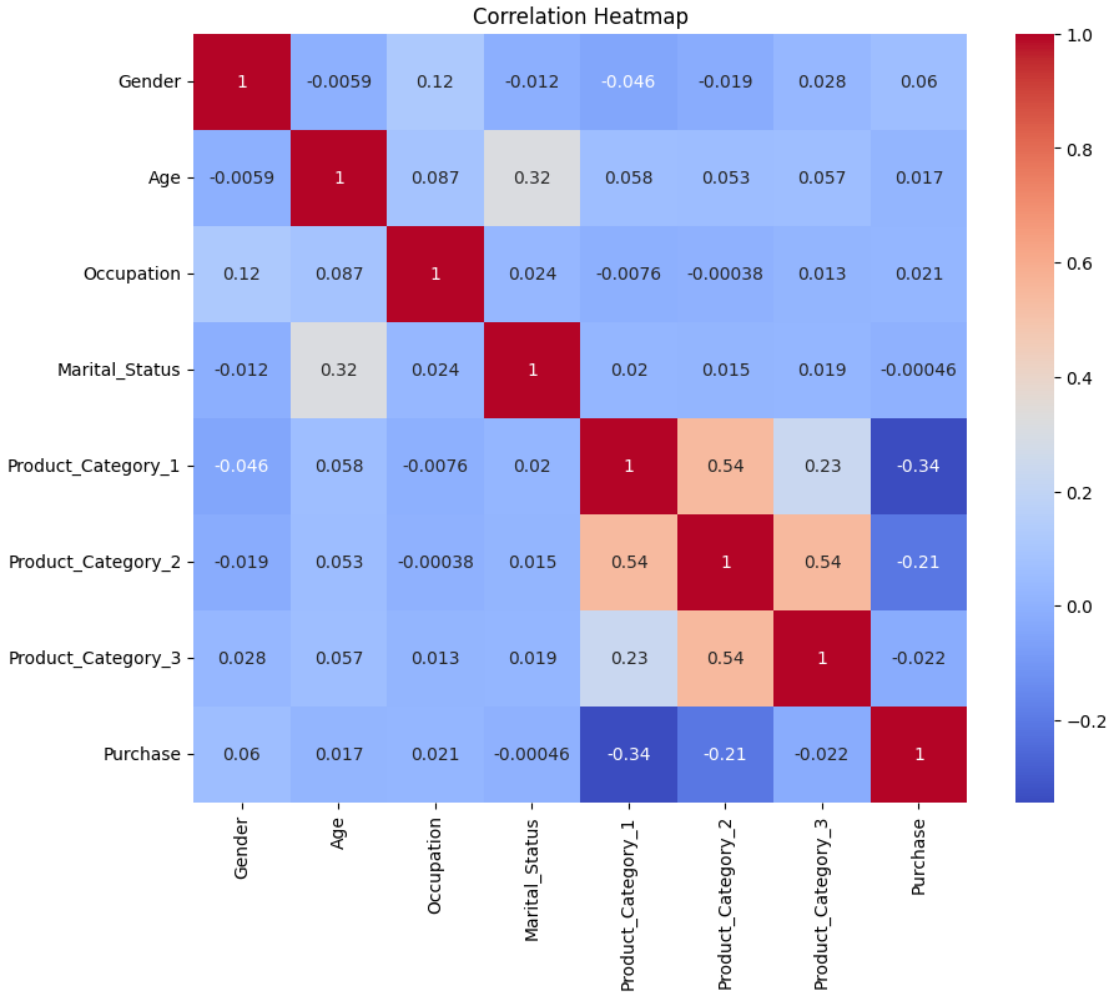



[188]:
```
# Marital Status Analysis
plt.figure(figsize=(10, 6))
sns.violinplot(x="Marital_Status", y="Purchase", data=df)
plt.title("Purchase Amount by Marital Status")
plt.xlabel("Marital Status")
plt.ylabel("Purchase Amount")
plt.show()
```




[209]:
```
# Correlation Heatmap
correlation_matrix = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```

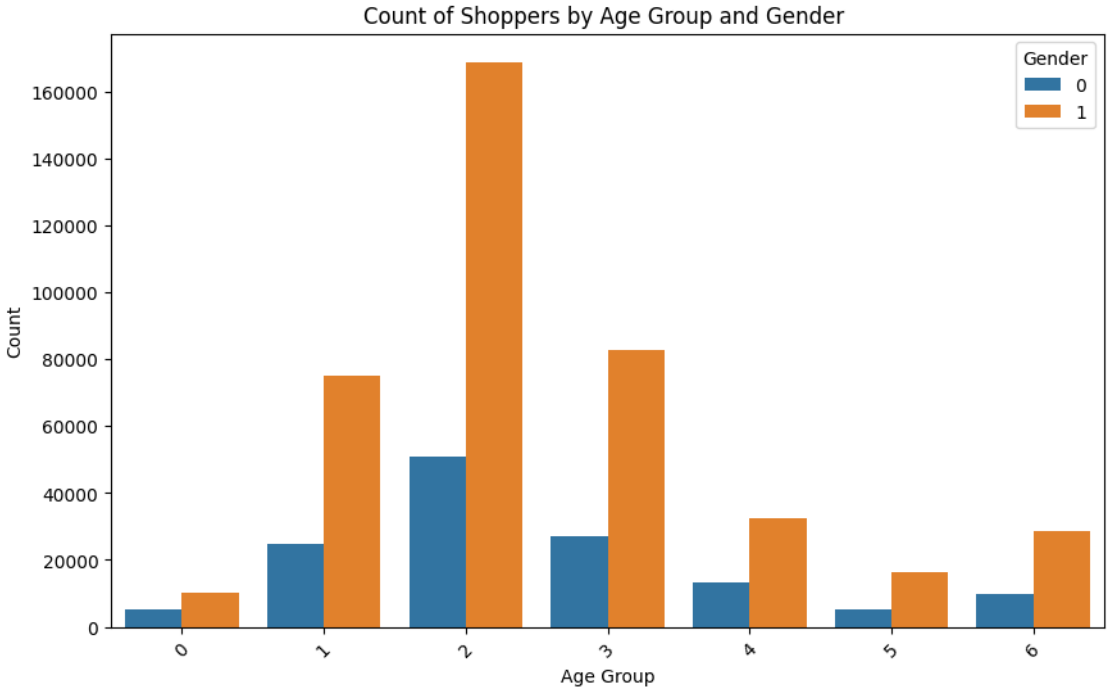
```
<ipython-input-209-986d89a622d1>:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  correlation_matrix = df.corr()
```




[191]:
```
# Descriptive Statistics
purchase_stats = df["Purchase"].describe()

# Gender and Purchase Analysis (Violin Plot)
plt.figure(figsize=(10, 6))
sns.violinplot(x="Gender", y="Purchase", data=df)
plt.title("Purchase Amount by Gender")
plt.xlabel("Gender")
plt.ylabel("Purchase Amount")
plt.show()
```

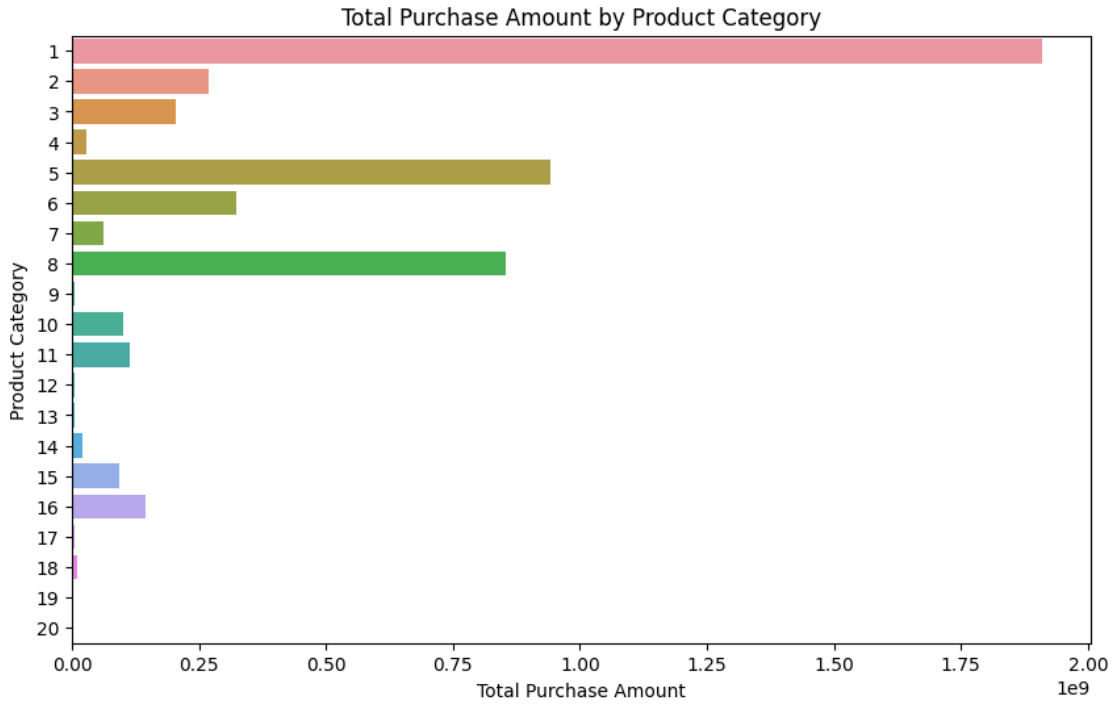

**The violin plot visually represents the distribution of purchase amounts for different genders and allows you to compare the data distribution and central tendencies between categories. The plot can help identify potential differences in purchasing behavior between genders.**

[217]:
```
# # Age Group Analysis (Count Plot)
# "'0-17': 1,
#     '18-25': 2,
#     '26-35': 3,
#     '36-45': 4,
#     '46-50': 5,
#     '55+': 6"
plt.figure(figsize=(10, 6))
sns.countplot(x="Age", data=df, hue="Gender")
plt.title("Count of Shoppers by Age Group and Gender ")
plt.xlabel("Age Group")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```

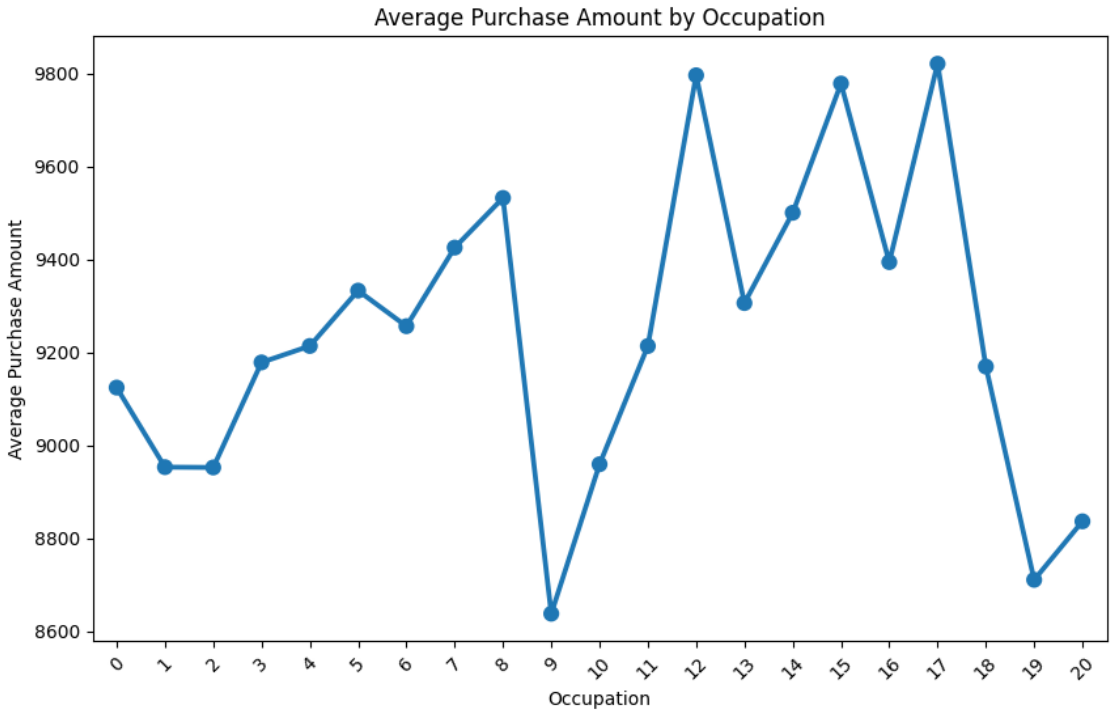



**18 -25 age groups males do more shoping**

[193]:
```
# Product Category Analysis (Horizontal Bar Plot)
product_category_totals = df.groupby("Product_Category_1")["Purchase"].sum().
↪reset_index()
plt.figure(figsize=(10, 6))
sns.barplot(y="Product_Category_1", x="Purchase", data=product_category_totals,␣
↪orient="h")
plt.title("Total Purchase Amount by Product Category")
plt.xlabel("Total Purchase Amount")
plt.ylabel("Product Category")
plt.show()
```




[194]:
```
#Average Purchase Amount by Occupation
occupation_purchase = df.groupby("Occupation")["Purchase"].mean().reset_index()
plt.figure(figsize=(10, 6))
sns.pointplot(x="Occupation", y="Purchase", data=occupation_purchase)
plt.title("Average Purchase Amount by Occupation")
plt.xlabel("Occupation")
plt.ylabel("Average Purchase Amount")
plt.xticks(rotation=45)
plt.show()
```



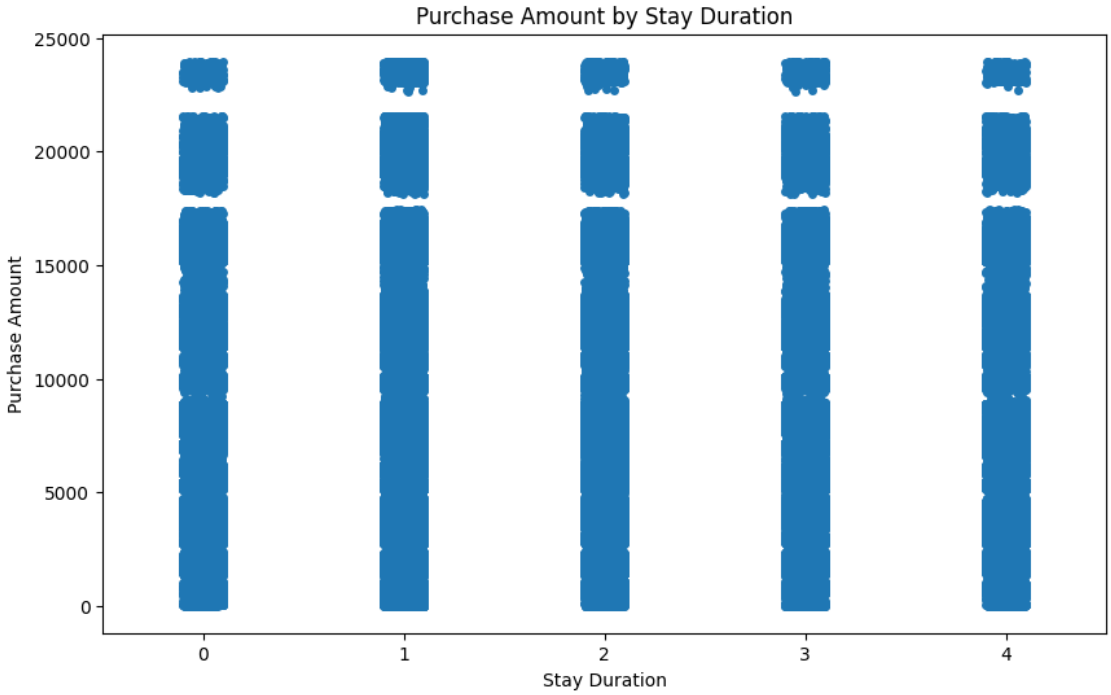

[218]:
```
df["Occupation"]
```

[218]:
```
0          10
1          10
2          10
3          10
4          16
           ..
550063     13
550064      1
550065     15
550066      1
550067      0
Name: Occupation, Length: 550068, dtype: int64
```

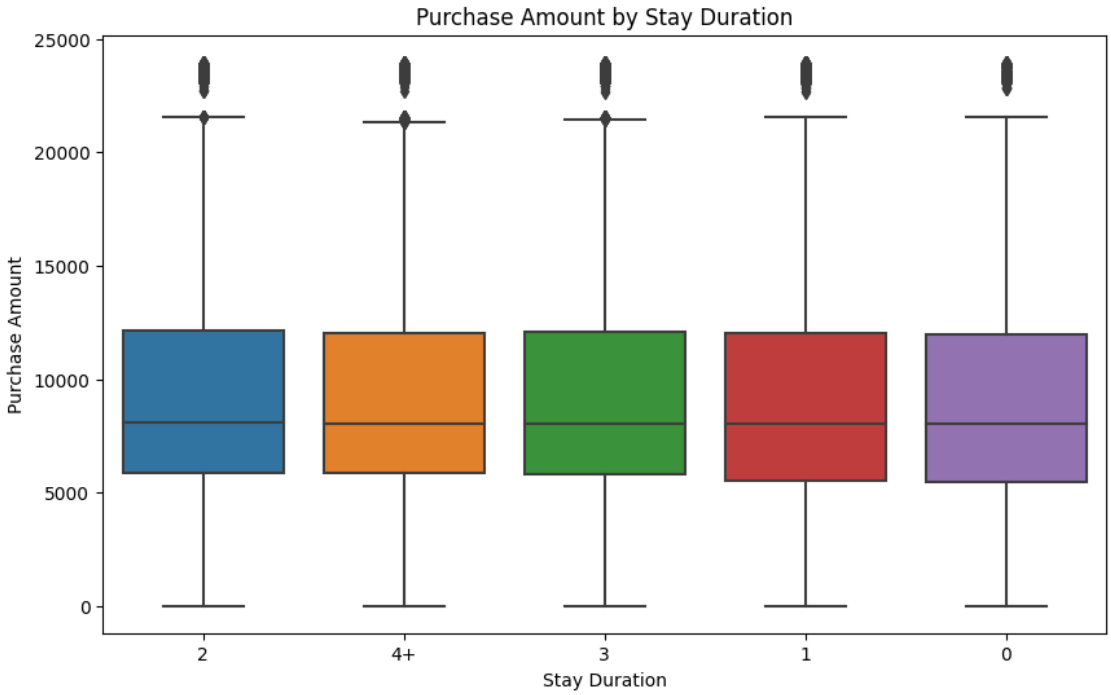
** the purpose of this code is to visually represent and compare the average purchase amounts for different occupation groups using a point plot. The point plot provides insight into how the purchase behavior varies across different occupations in the dataset.**

[195]:
```
# City Stay Duration Analysis (Strip Plot)
plt.figure(figsize=(10, 6))
sns.stripplot(x="Stay_In_Current_City_Years", y="Purchase", data=df,␣
 ↪jitter=True)
plt.title("Purchase Amount by Stay Duration")
```

```
plt.xlabel("Stay Duration")
plt.ylabel("Purchase Amount")
plt.show()
```

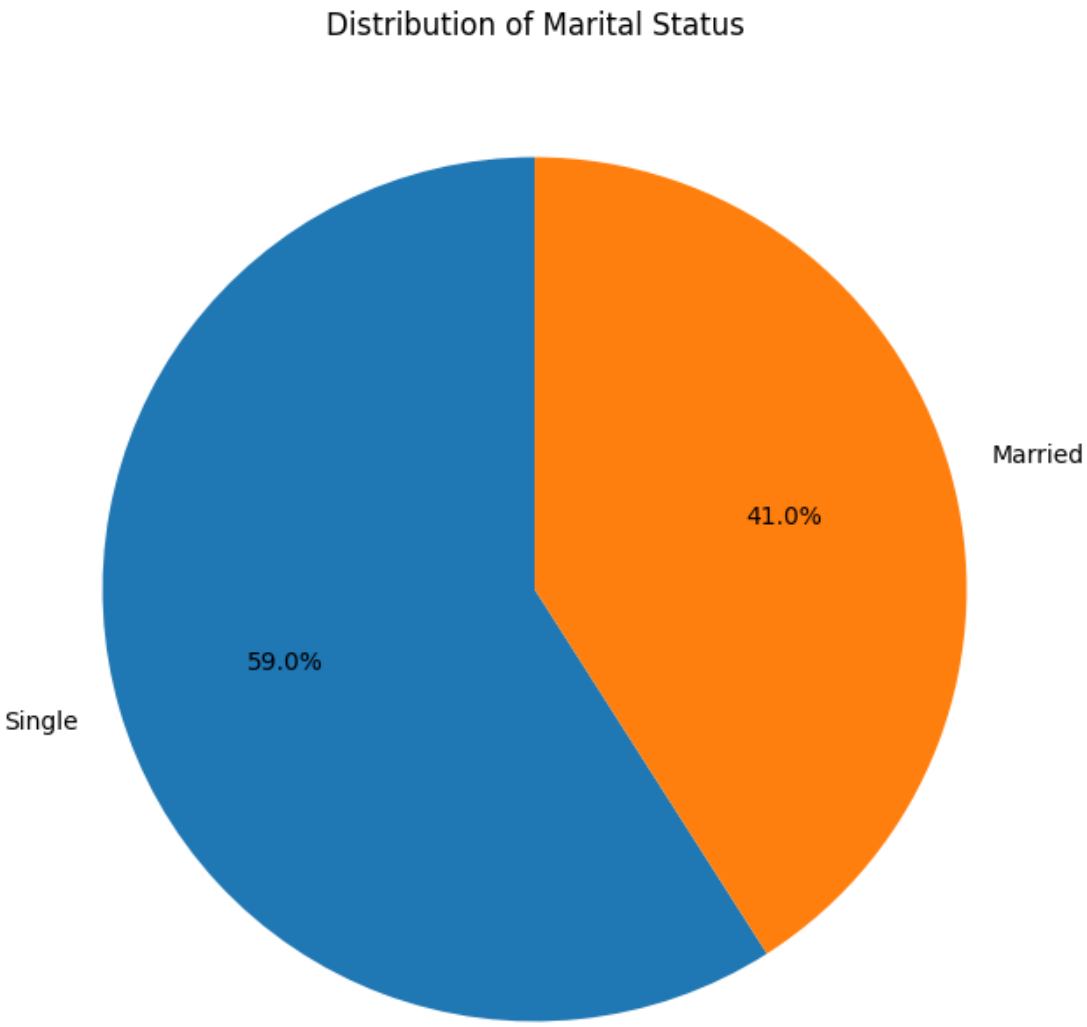



[219]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x="Stay_In_Current_City_Years", y="Purchase", data=df)
plt.title("Purchase Amount by Stay Duration")
plt.xlabel("Stay Duration")
plt.ylabel("Purchase Amount")
plt.show()
```




**From This graph its evident while purcasing moslty people stay in city**

[196]:
```
# Marital Status Analysis (Pie Chart)
marital_status_counts = df["Marital_Status"].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(marital_status_counts, labels=["Single", "Married"], autopct="%1.1f%%",␣
↪startangle=90)
plt.title("Distribution of Marital Status")
plt.show()
```



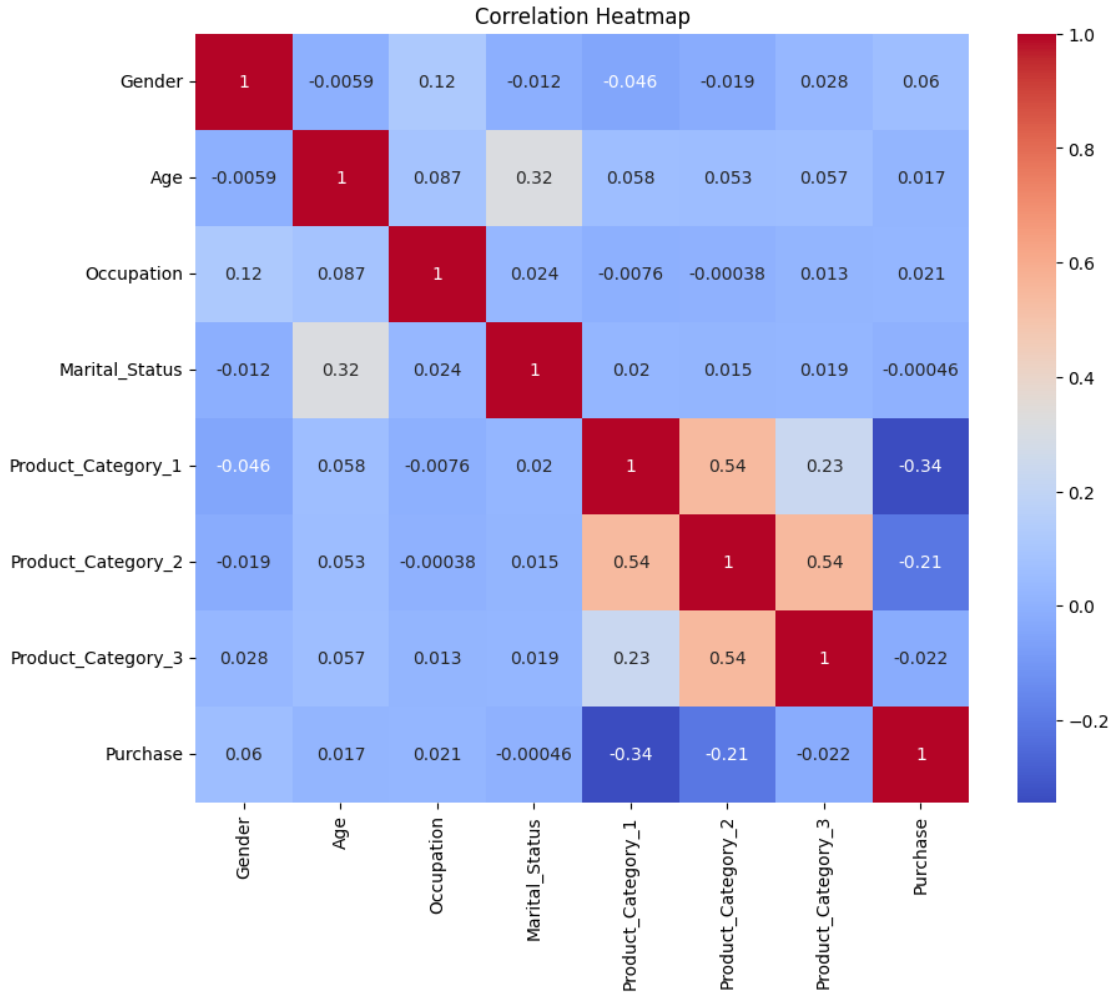

*THis graph show mostly single females do shoping **

## 5 Heatmap checking correlation

[225]:
```
# Correlation Heatmap
correlation_matrix = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```

```
<ipython-input-225-986d89a622d1>:2: FutureWarning: The default value of
```

```
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  correlation_matrix = df.corr()
```




[220]:
```
# Suggestions

print("\nTop 5 Product Categories by Total Purchase Amount:\n",␣
 ↪product_category_totals.nlargest(5, "Purchase"))
print("\nTop 5 Occupations by Average Purchase Amount:\n", occupation_purchase.
 ↪nlargest(5, "Purchase"))
```

```
Top 5 Product Categories by Total Purchase Amount:
    Product_Category_1    Purchase
0                   1  1910013754
4                   5   941835229
```

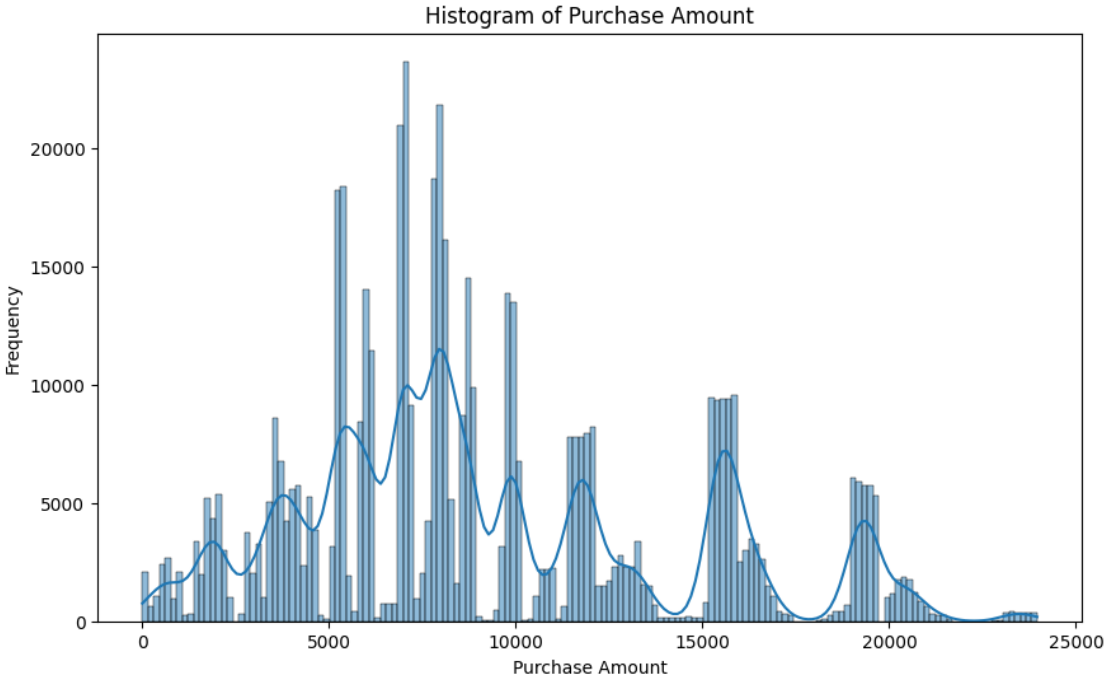
```
7                   8   854318799
5                   6   324150302
1                   2   268516186

Top 5 Occupations by Average Purchase Amount:
     Occupation     Purchase
17           17  9821.478236
12           12  9796.640239
15           15  9778.891163
8             8  9532.592497
14           14  9500.702772
```

## 6 Normality Test

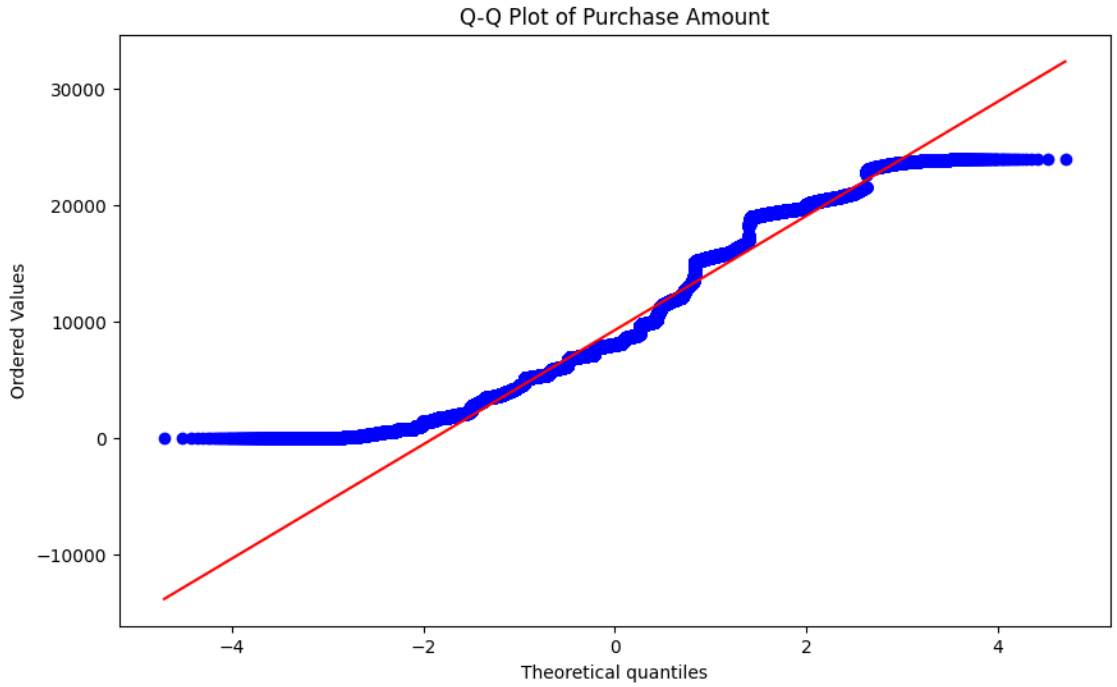
[202]:
```
# Normality Check using Shapiro-Wilk test
from scipy.stats import shapiro

purchase_data = df["Purchase"]
statistic, p_value = shapiro(purchase_data)

alpha = 0.05
print("Shapiro-Wilk Test:")
print(f"Statistic: {statistic:.4f}")
print(f"P-value: {p_value:.4f}")

if p_value > alpha:
    print("The data follows a normal distribution (Fail to reject H0)")
else:
    print("The data does not follow a normal distribution (Reject H0)")
```

```
Shapiro-Wilk Test:
Statistic: 0.9526
P-value: 0.0000
The data does not follow a normal distribution (Reject H0)

/usr/local/lib/python3.10/dist-packages/scipy/stats/_morestats.py:1816:
UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

[203]:
```
# Histogram
plt.figure(figsize=(10, 6))
sns.histplot(purchase_data, kde=True)
plt.title("Histogram of Purchase Amount")
plt.xlabel("Purchase Amount")
plt.ylabel("Frequency")
plt.show()
```




[207]:
```
# Q-Q Plot
plt.figure(figsize=(10, 6))
probplot(purchase_data, plot=plt)
plt.title("Q-Q Plot of Purchase Amount")
plt.show()
```



**By visualization and aplying shapiro test data is not normal**

**The Shapiro-Wilk test revealed that the "Purchase" data does not follow a normal distribution, as evidenced by a very low p-value and a test statistic of 0.9526. The strong evidence against normality suggests caution when applying statistical methods that assume normal distribution to this dataset. Consideration of alternative statistical methods, such as non-parametric tests, bootstrapping, or transformation, may be necessary to conduct valid inferential analyses.**