
```
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094
To: /content/walmart_data.csv?1641285094
100% 23.0M/23.0M [00:00<00:00, 134MB/s]
```

1. Defining Problem Statement and Analyzing basic metrics:

The management team at Walmart Inc. aims to analyze customer purchase behavior, specifically focusing on the purchase amount, with respect to the customer's gender and other factors. They want to understand if there are differences in spending habits between male and female customers, particularly on Black Friday. By gaining insights into these spending patterns, Walmart can make informed decisions to enhance its marketing strategies, product offerings, and customer experience.

 # Basic Matrix

```
import numpy as np
import pandas as pd
```

```
df = pd.read_csv('walmart_data.csv')
```

```
df.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	7969

```
# Check the structure of the dataset
print("\nDataset structure:")
print(df.info())
```

```
Dataset structure:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 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            550068 non-null  int64
 9   Purchase                    550068 non-null  int64
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
None
```

1. Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), statistical summary:

Shape of the Data

```
df.shape
print("No of Rows=", df.shape[0])
print("No of cols=", df.shape[1])
```

```
No of Rows= 550068
No of cols= 10
```

Data types of all attributes

```
data_type = df.dtypes
data_type
```

```
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             int64
Purchase                     int64
dtype: object
```

Convert categorical attributes to 'category' if required

```
# Convert categorical attributes to 'category' if required
categorical_cols = df.select_dtypes(include='object').columns
df[categorical_cols] = df[categorical_cols].astype('category')
print("\nData types after conversion:")
print(df.dtypes)
```

```
Data types after conversion:
User_ID                         int64
Product_ID                   category
Gender                       category
Age                          category
Occupation                      int64
City_Category                category
Stay_In_Current_City_Years   category
Marital_Status                  int64
Product_Category                int64
Purchase                        int64
dtype: object
```

Statistical summary

```
# Statistical summary
print("\nStatistical summary:")
print(df.describe(include='all'))
```

```
Statistical summary:
            User_ID Product_ID  Gender     Age     Occupation City_Category  \
count   5.500680e+05     550068  550068  550068  550068.000000        550068
unique           NaN       3631       2       7            NaN             3
top              NaN  P00265242       M   26-35            NaN             B
freq             NaN       1880  414259  219587            NaN        231173
mean    1.003029e+06        NaN     NaN     NaN       8.076707           NaN
std     1.727592e+03        NaN     NaN     NaN       6.522660           NaN
min     1.000001e+06        NaN     NaN     NaN       0.000000           NaN
25%     1.001516e+06        NaN     NaN     NaN       2.000000           NaN
50%     1.003077e+06        NaN     NaN     NaN       7.000000           NaN
75%     1.004478e+06        NaN     NaN     NaN      14.000000           NaN
max     1.006040e+06        NaN     NaN     NaN      20.000000           NaN

       Stay_In_Current_City_Years  Marital_Status  Product_Category  \
count                      550068   550068.000000     550068.000000
unique                          5             NaN               NaN
top                             1             NaN               NaN
freq                       193821             NaN               NaN
mean                          NaN        0.409653          5.404270
std                           NaN        0.491770          3.936211
min                           NaN        0.000000          1.000000
25%                           NaN        0.000000          1.000000
50%                           NaN        0.000000          5.000000
75%                           NaN        1.000000          8.000000
max                           NaN        1.000000         20.000000

            Purchase
count   550068.000000
unique            NaN
top               NaN
freq              NaN
mean      9263.968713
std       5023.065394
min         12.000000
25%       5823.000000
50%       8047.000000
75%      12054.000000
max      23961.000000
```

2. Non-Graphical Analysis: Value counts and unique attributes

Value counts for categorical variables

```python
# Value counts for categorical variables
print("Value counts for categorical variables:")
for col in df.select_dtypes(include='category'):
    print(f"\n{col}:\n{df[col].value_counts()}")
```

```
Value counts for categorical variables:

Product_ID:
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
             ...
P00068742       1
P00012342       1
P00162742       1
P00091742       1
P00231642       1
Name: Product_ID, Length: 3631, dtype: int64

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


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

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

Stay_In_Current_City_Years:
1     193821
2     101838
3      95285
4+     84726
0      74398
Name: Stay_In_Current_City_Years, dtype: int64
```

Value counts for Integer variables

```python
# Value counts for Integer variables
print("Value counts for Integer variables:")
for col in df.select_dtypes(include='int64'):
    print(f"\n{col}:\n{df[col].value_counts()}")
```

```
Value counts for Integer variables:

User_ID:
1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
           ...
1002690       7
1002111       7
1005810       7
1004991       7
1000708       6
Name: User_ID, Length: 5891, dtype: int64

Occupation:
4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: Occupation, dtype: int64
```

```
Marital_Status:
0    324731
1    225337
Name: Marital_Status, dtype: int64

Product_Category:
5     150933
1     140378
8     113925
11     24287
2      23864
6      20466
3      20213
4      11753
16      9828
15      6290
13      5549
10      5125
12      3947
7       3721
18      3125
20      2550
19      1603
14      1523
17       578
9        410
Name: Product_Category, dtype: int64

Purchase:
7011     191
7193     188
6855     187
6891     184
7012     183
        ...
23491      1
18345      1
3372       1
855        1
21489      1
Name: Purchase, Length: 18105, dtype: int64
```

Unique attributes for each column

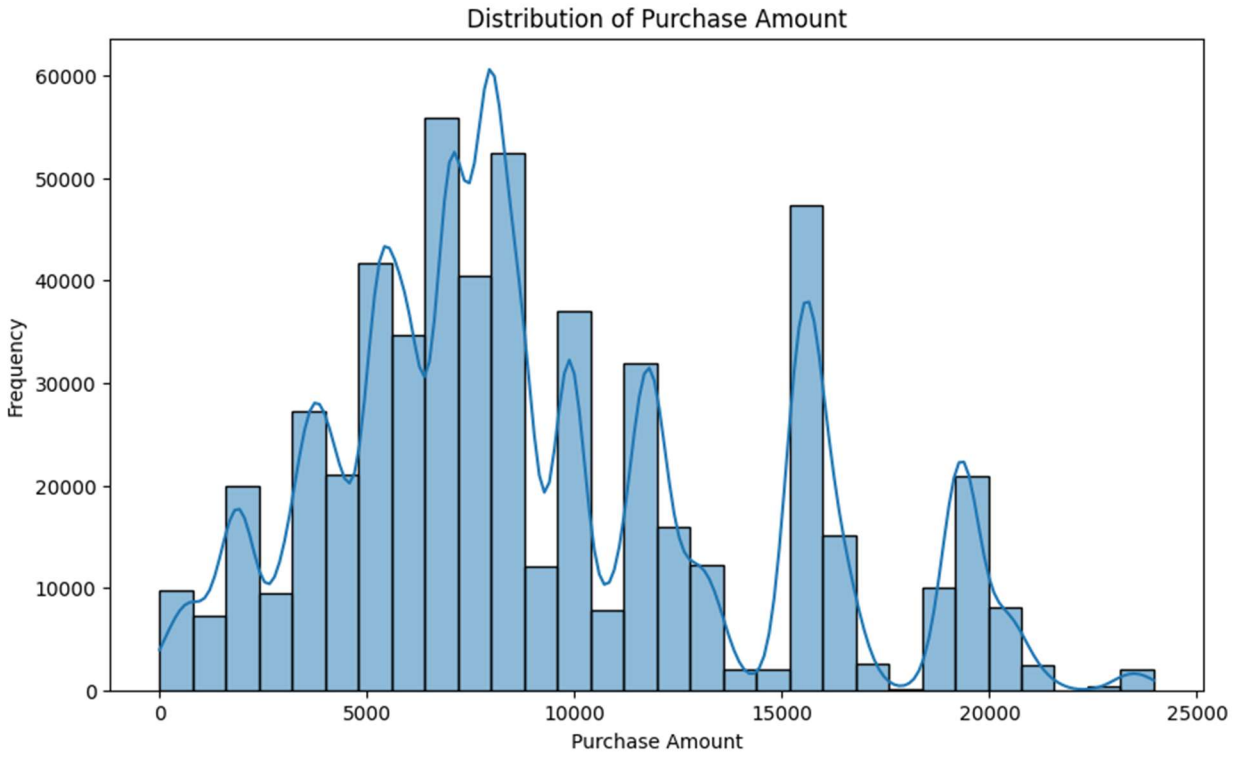
```
# Unique attributes for each column
print("\nUnique attributes for each column:")
for col in df.columns:
    print(f"\n{col}:\n{df[col].unique()}")
```

```
Unique attributes for each column:

User_ID:
[1000001 1000002 1000003 ... 1004113 1005391 1001529]

Product_ID:
['P00069042' 'P00248942' 'P00087842' ... 'P00370293' 'P00371644'
 'P00370853']

Gender:
['F' 'M']

Age:
['0-17' '55+' '26-35' '46-50' '51-55' '36-45' '18-25']

Occupation:
[10 16 15  7 20  9  1 12 17  0  3  4 11  8 19  2 18  5 14 13  6]

City_Category:
['A' 'C' 'B']

Stay_In_Current_City_Years:
['2' '4+' '3' '1' '0']

Marital_Status:
[0 1]

Product_Category:
[ 3  1 12  8  5  4  2  6 14 11 13 15  7 16 18 10 17  9 20 19]

Purchase:
[ 8370 15200  1422 ...   135   123   613]
```

3. Visual Analysis - Univariate & Bivariate

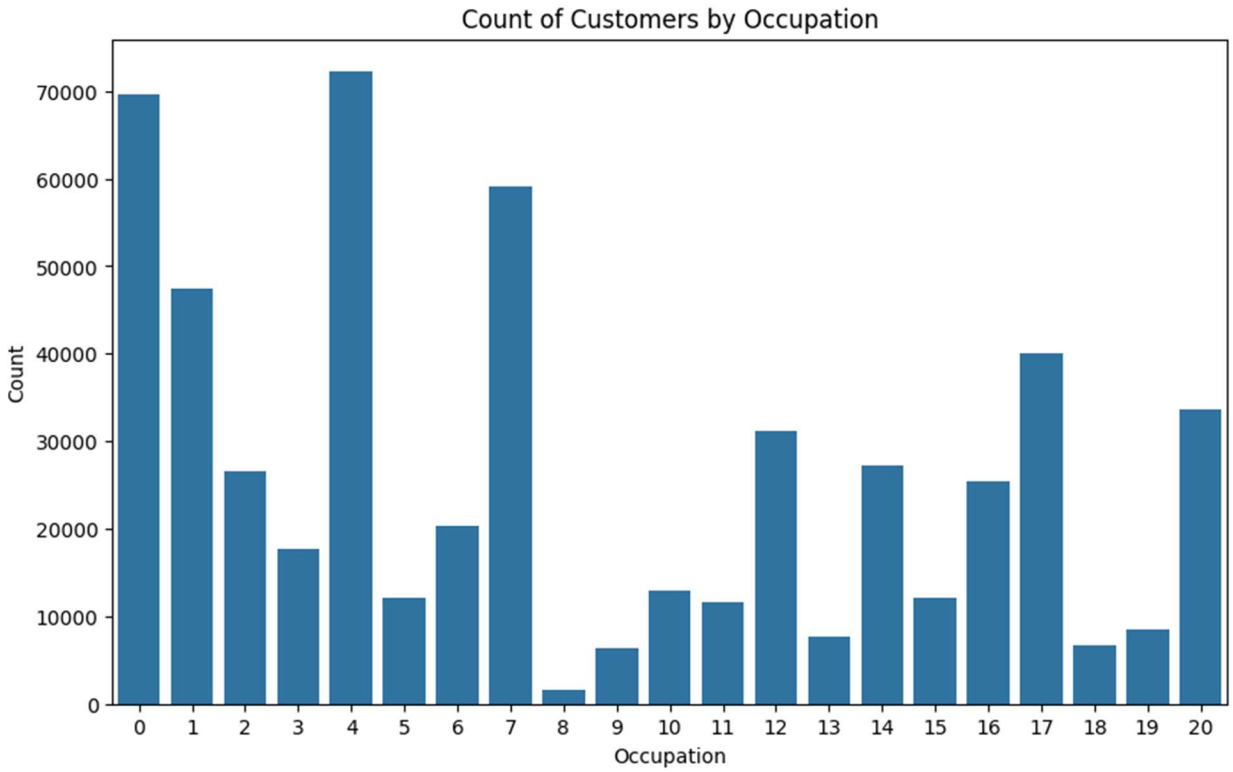
Distribution of Purchase Amount

```
# Distplot for continuous variable
plt.figure(figsize=(10, 6))
sns.histplot(data['Purchase'], bins=30, kde=True)
plt.title("Distribution of Purchase Amount")
plt.xlabel("Purchase Amount")
plt.ylabel("Frequency")
plt.show()
```



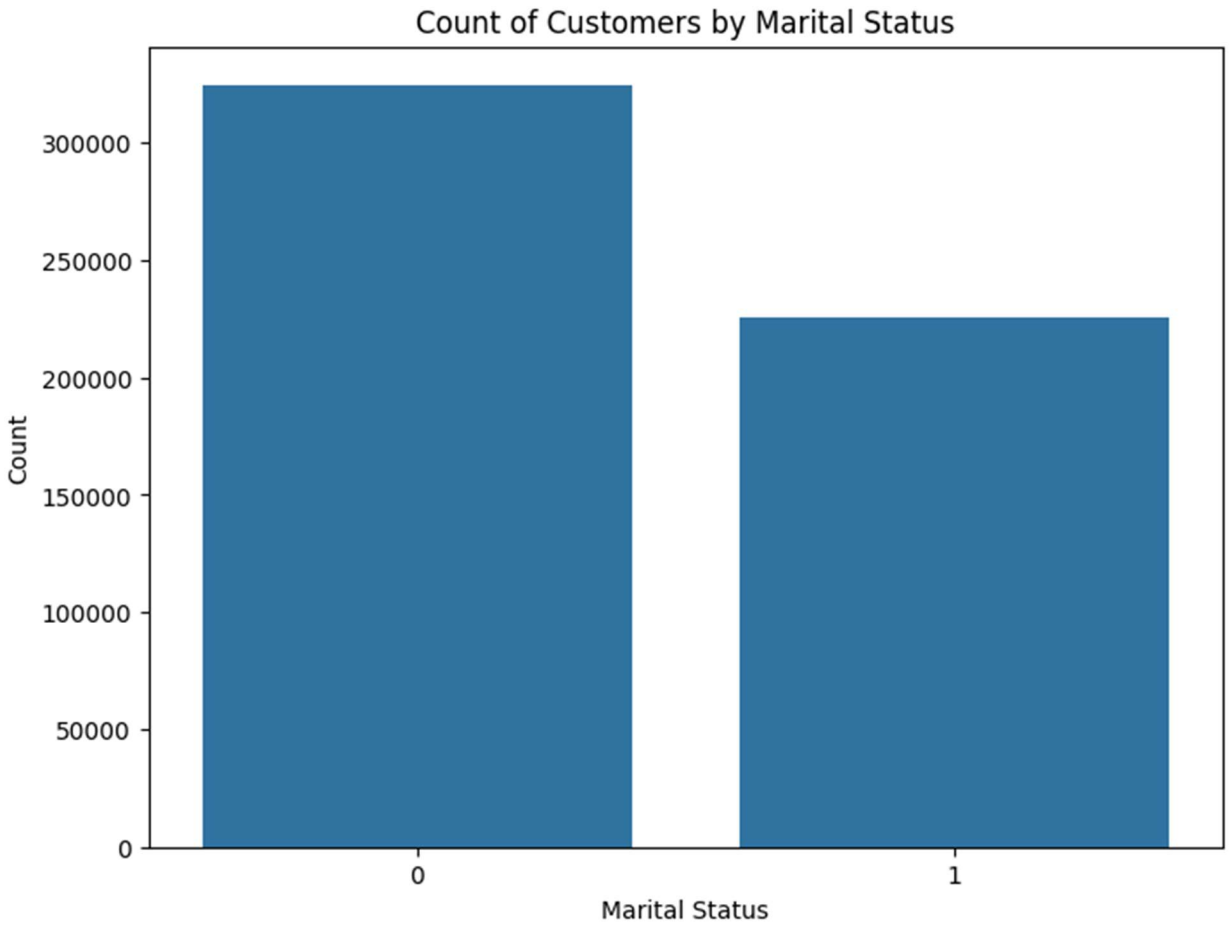
Count of Customers by Occupation

```
# Countplot for Occupation
plt.figure(figsize=(10, 6))
sns.countplot(x='Occupation', data=data)
plt.title("Count of Customers by Occupation")
plt.xlabel("Occupation")
plt.ylabel("Count")
plt.show()
```



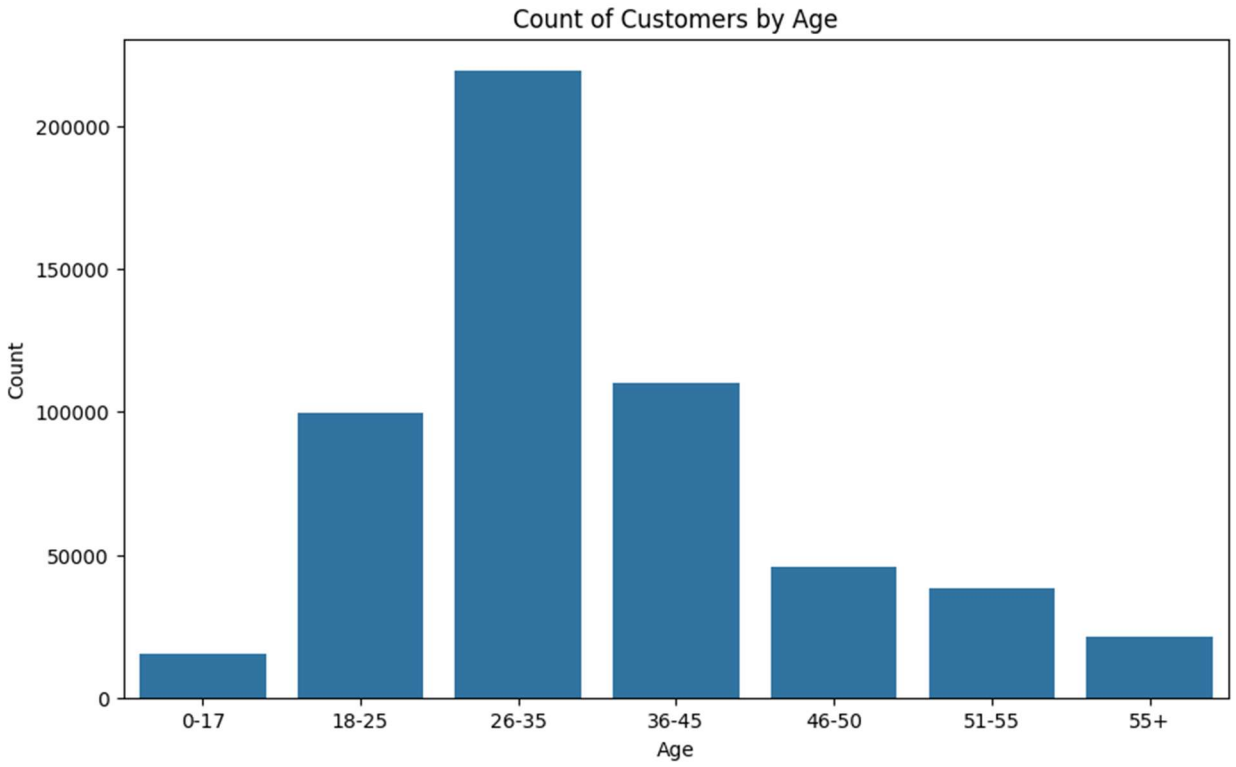
Count of Customers by marital status

```
# Countplot for Marital Status
plt.figure(figsize=(8, 6))
sns.countplot(x='Marital_Status', data=data)
plt.title("Count of Customers by Marital Status")
plt.xlabel("Marital Status")
plt.ylabel("Count")
plt.show()
```

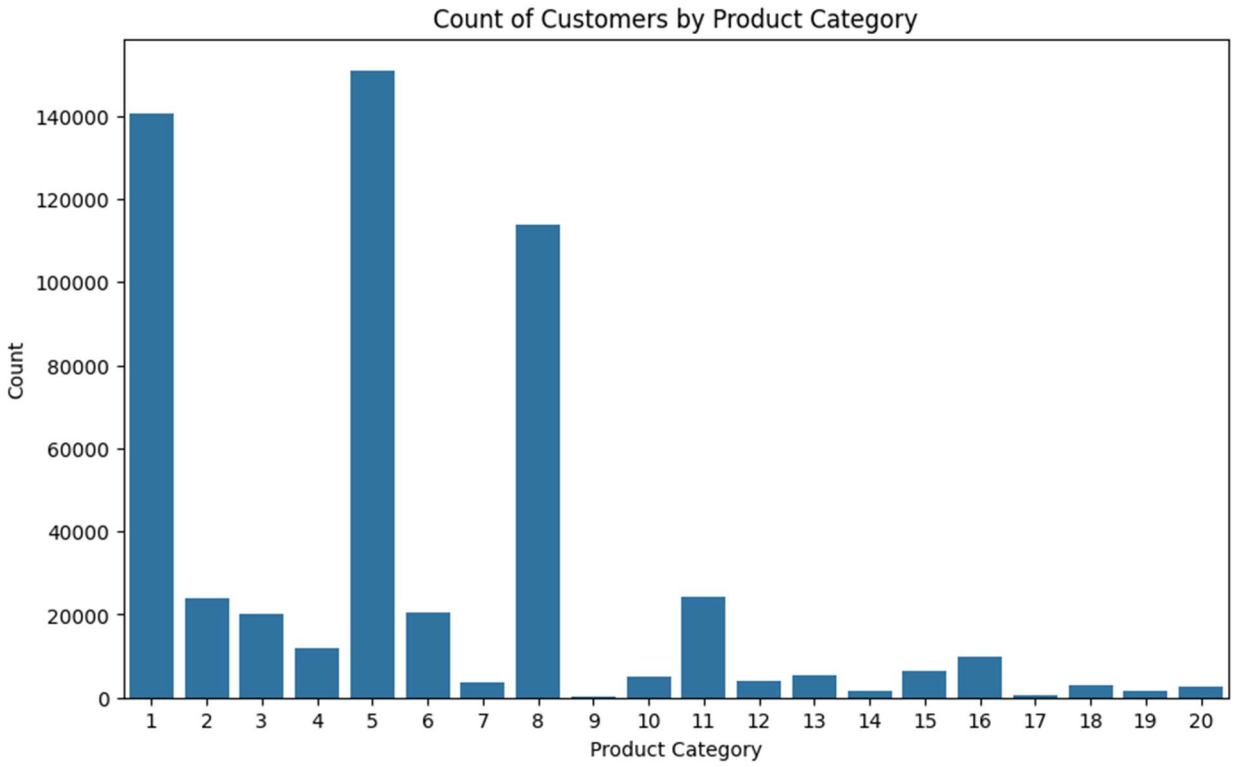


count of customers by Age

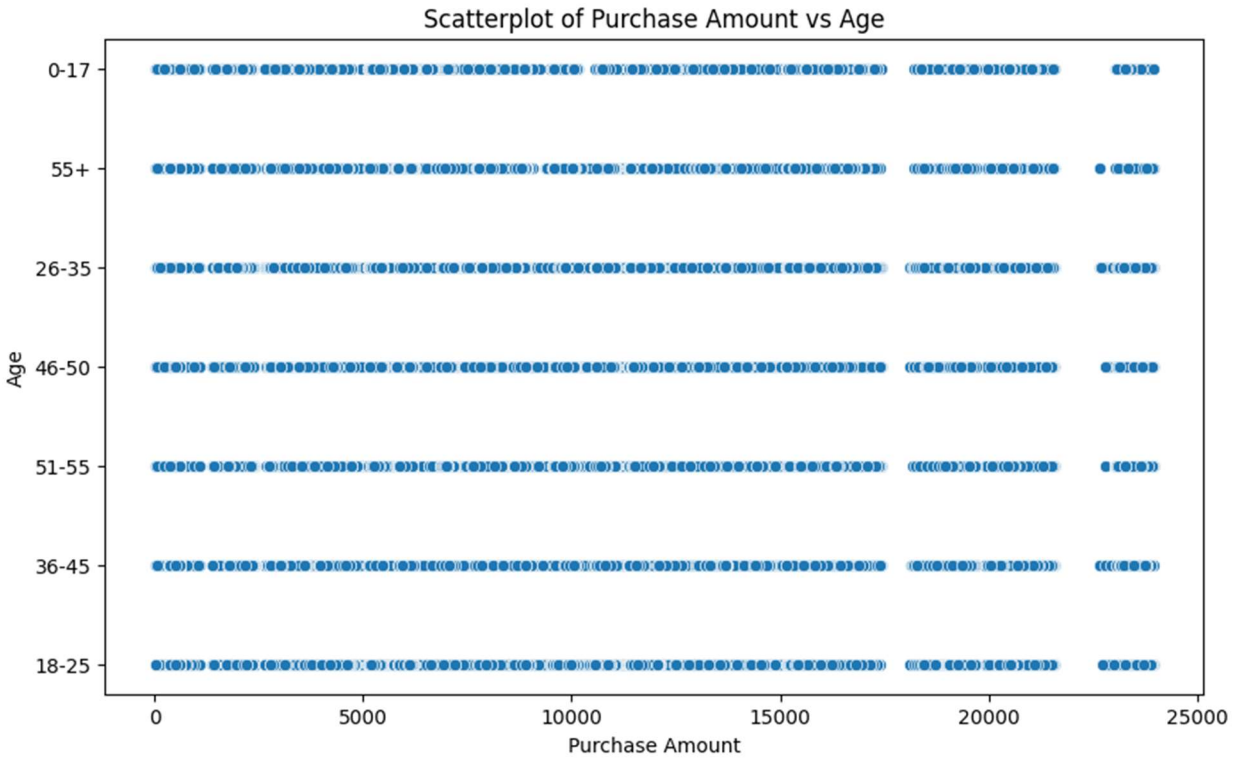
```
# Sort age groups
def sort_age_group(age_group):
    if '+' in age_group:
        return float('inf')  # Assign a high value for age group with '+'
    else:
        return int(age_group.split('-')[0])  # Extract the lower bound of the age group

sorted_age_groups = sorted(data['Age'].unique(), key=sort_age_group)

# Countplot for Age with sorted labels
plt.figure(figsize=(10, 6))
sns.countplot(x='Age', data=data, order=sorted_age_groups)
plt.title("Count of Customers by Age")
plt.xlabel("Age")
plt.ylabel("Count")
plt.show()
```



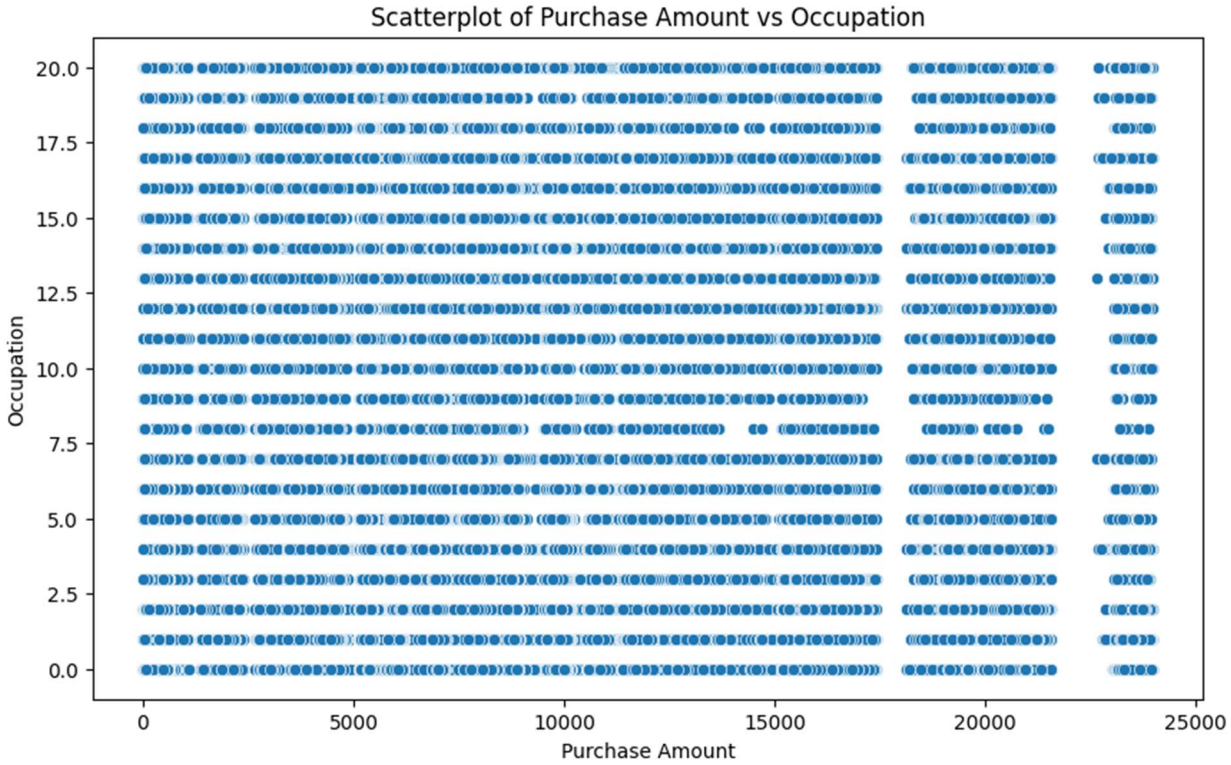

```
# Countplot for Product Category
plt.figure(figsize=(10, 6))
sns.countplot(x='Product_Category', data=data)
plt.title("Count of Customers by Product Category")
plt.xlabel("Product Category")
plt.ylabel("Count")
plt.show()
```



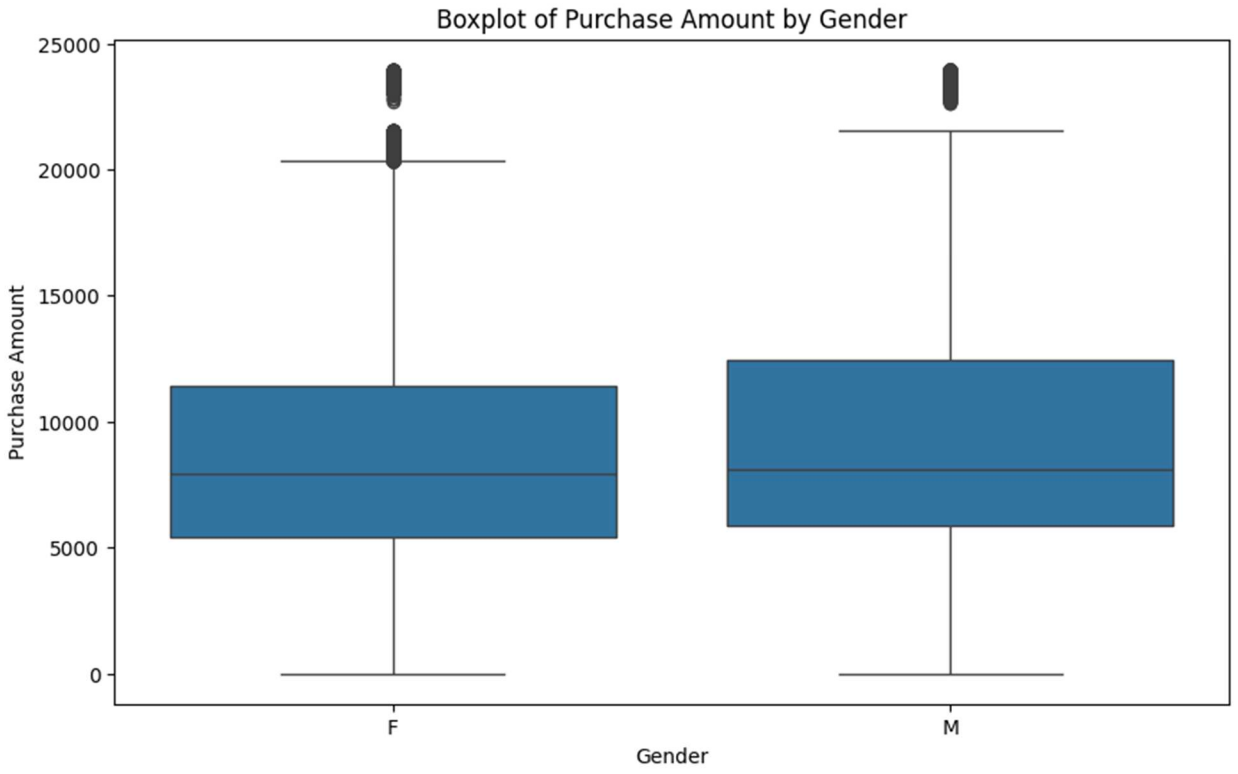

```
# Scatterplot for continuous vs continuous variables
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Purchase', y='Age', data=data)
plt.title("Scatterplot of Purchase Amount vs Age")
plt.xlabel("Purchase Amount")
plt.ylabel("Age")
plt.show()
```



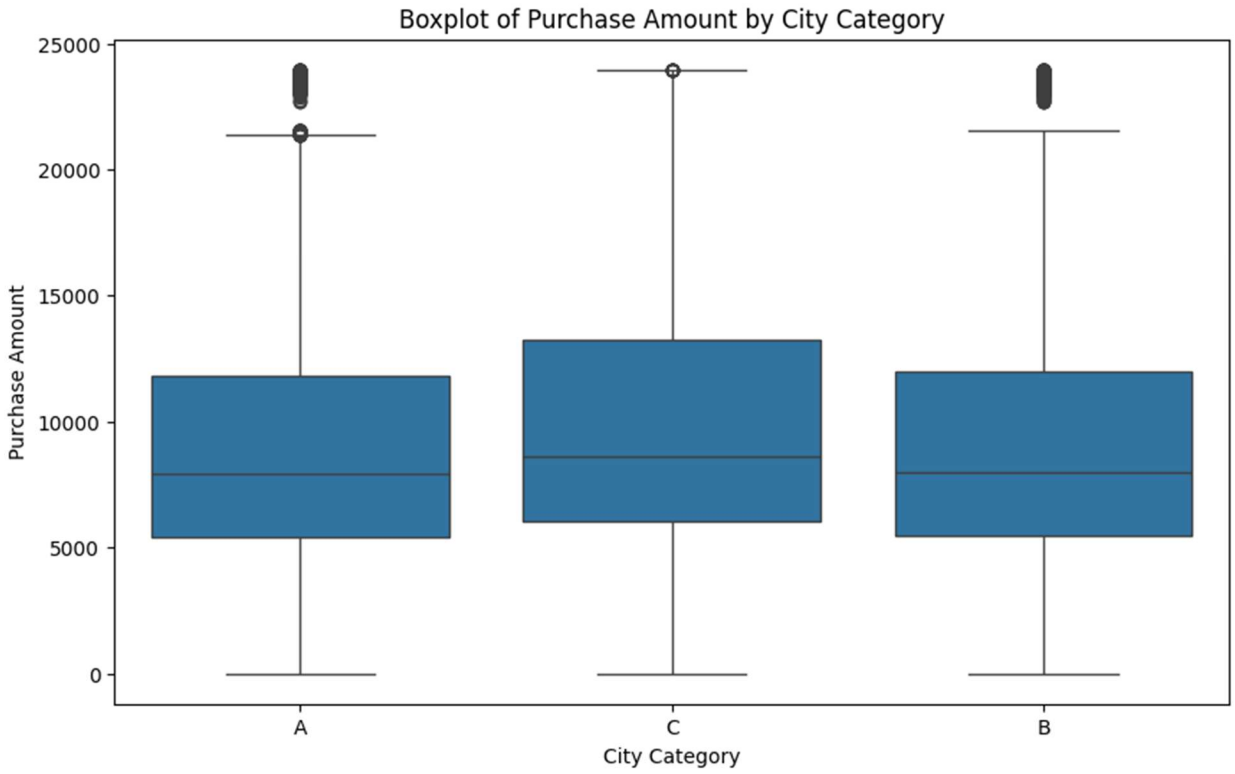

```
# Scatterplot for continuous vs continuous variables
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Purchase', y='Occupation', data=data)
plt.title("Scatterplot of Purchase Amount vs Occupation")
plt.xlabel("Purchase Amount")
plt.ylabel("Occupation")
plt.show()
```



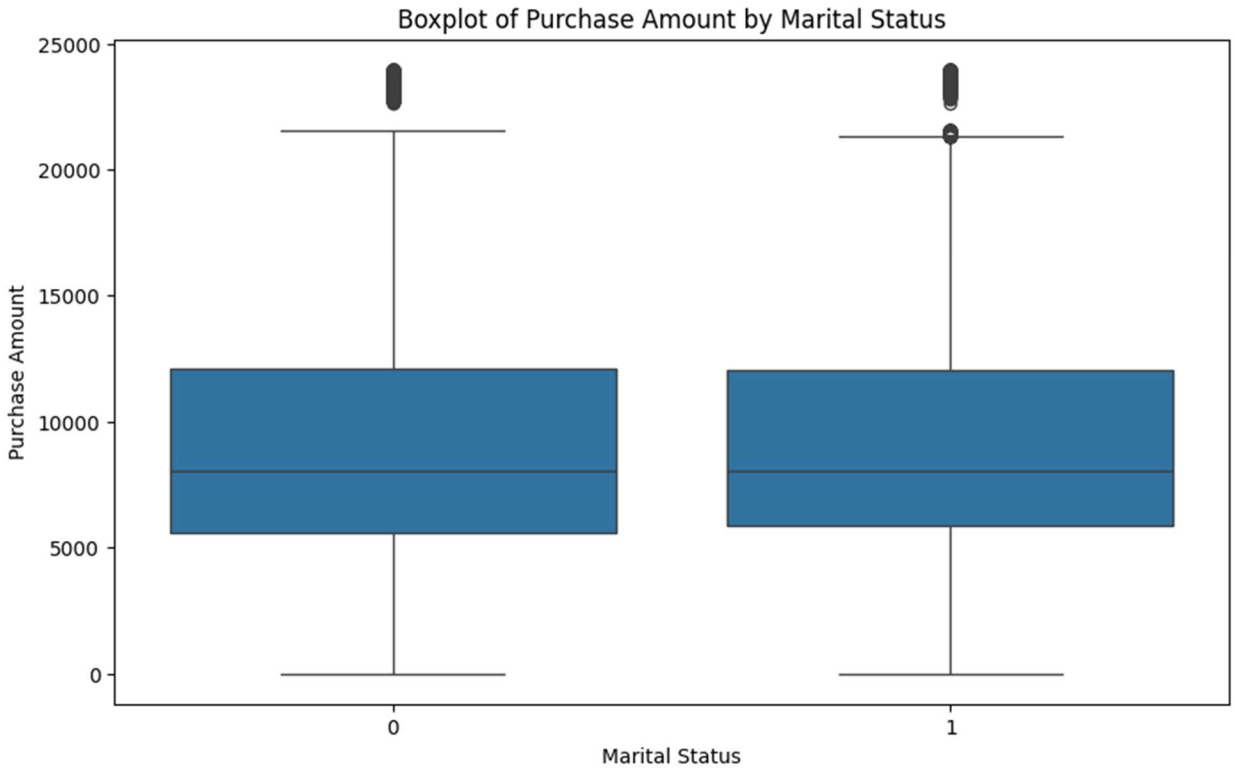

```
# Boxplot for categorical vs continuous variables
plt.figure(figsize=(10, 6))
sns.boxplot(x='Gender', y='Purchase', data=data)
plt.title("Boxplot of Purchase Amount by Gender")
plt.xlabel("Gender")
plt.ylabel("Purchase Amount")
plt.show()
```



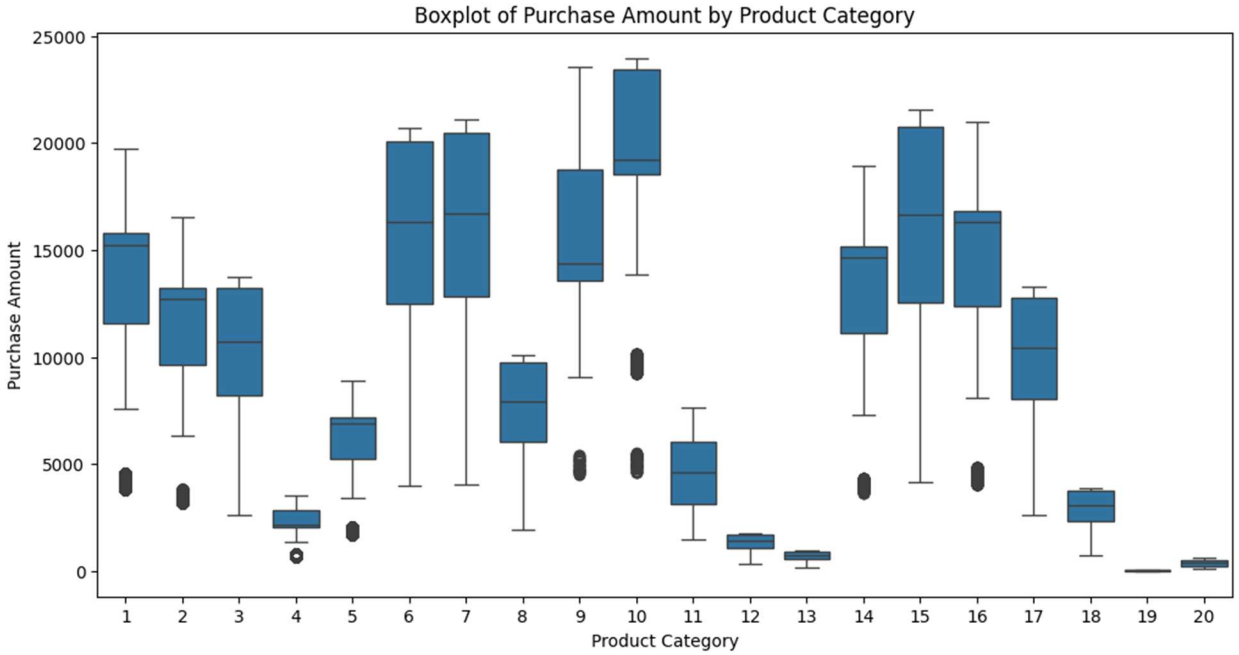

```
# Boxplot for categorical vs continuous variables
plt.figure(figsize=(10, 6))
sns.boxplot(x='City_Category', y='Purchase', data=data)
plt.title("Boxplot of Purchase Amount by City Category")
plt.xlabel("City Category")
plt.ylabel("Purchase Amount")
plt.show()
```



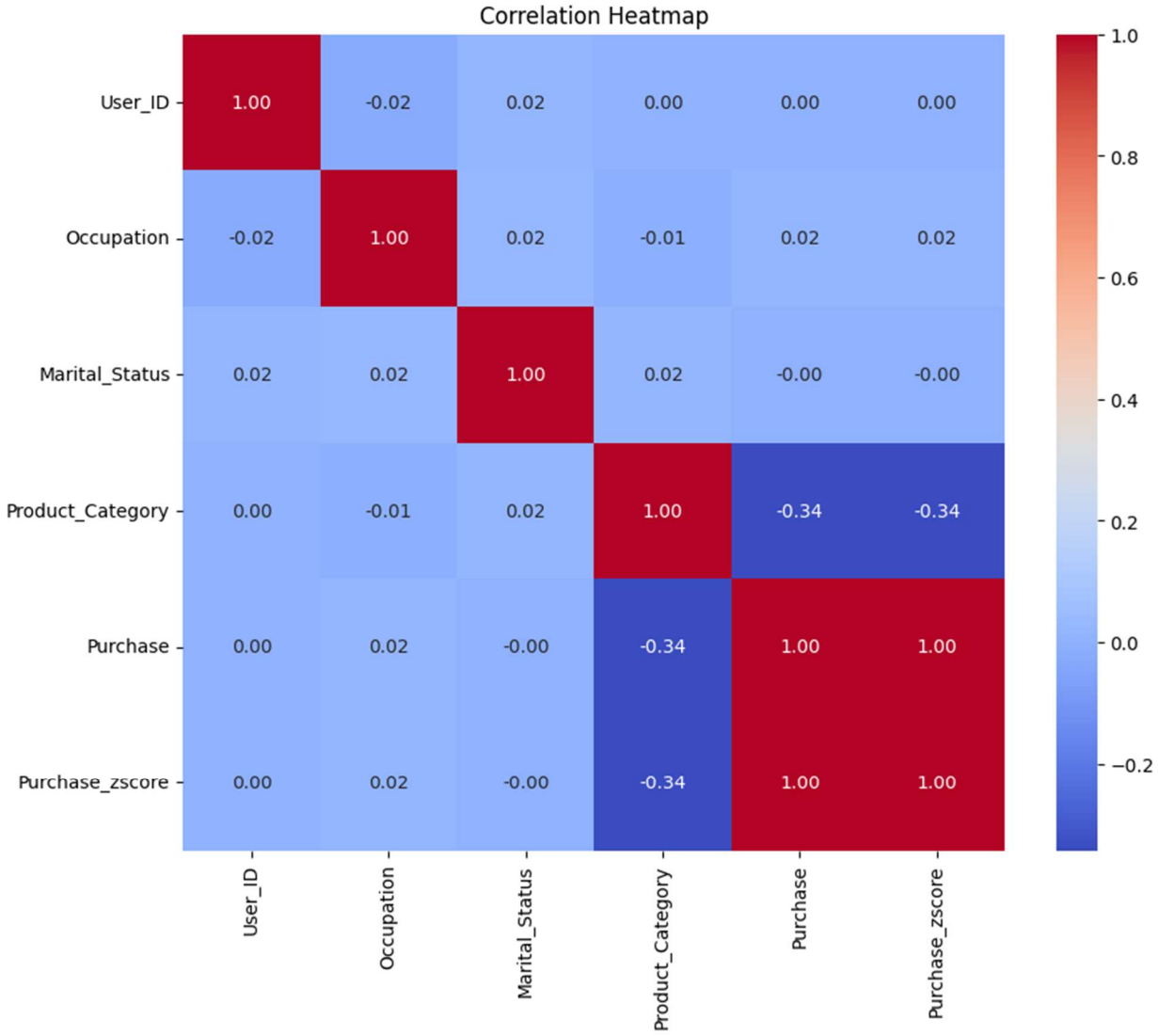

```
# Boxplot for categorical vs continuous variables
plt.figure(figsize=(10, 6))
sns.boxplot(x='Marital_Status', y='Purchase', data=data)
plt.title("Boxplot of Purchase Amount by Marital Status")
plt.xlabel("Marital Status")
plt.ylabel("Purchase Amount")
plt.show()
```



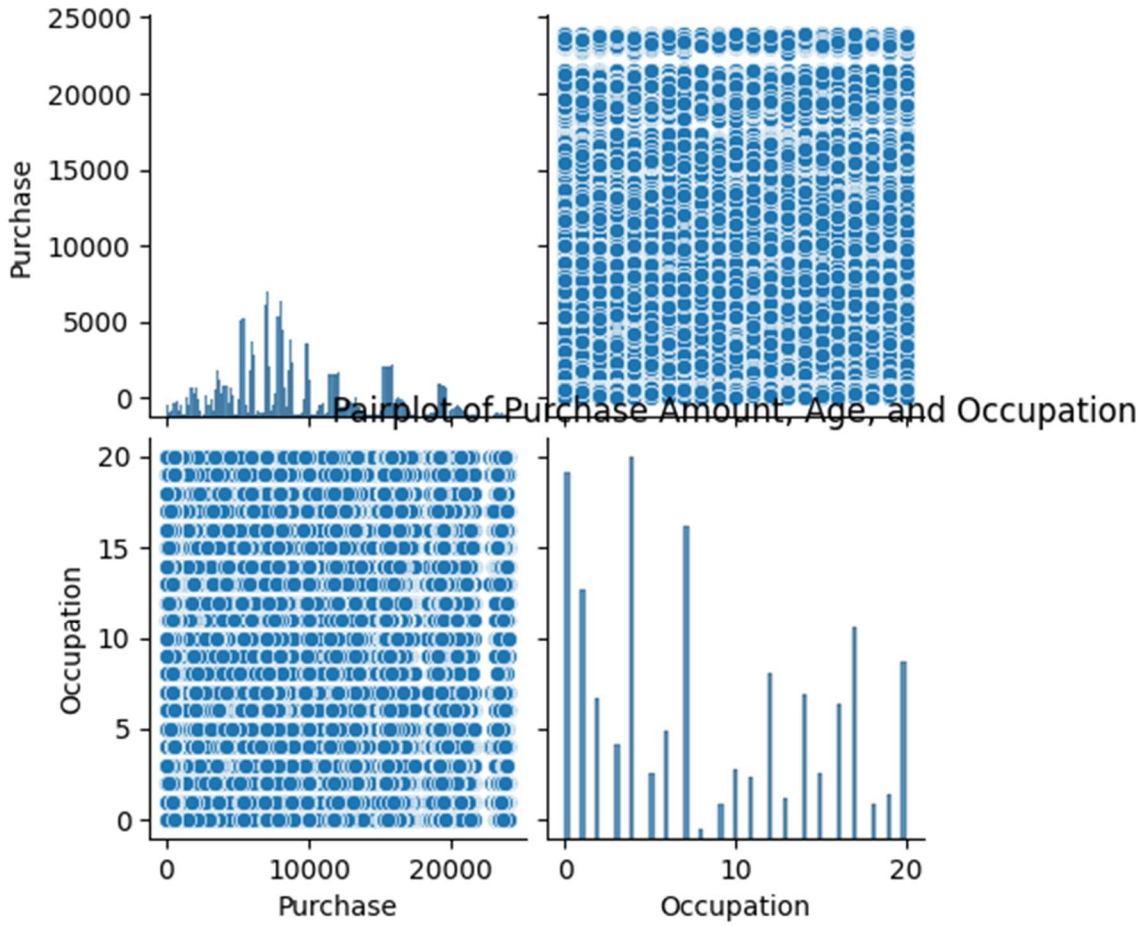

```
# Boxplot for Product_Category and Purchase
plt.figure(figsize=(12, 6))
sns.boxplot(x='Product_Category', y='Purchase', data=data)
plt.title('Boxplot of Purchase Amount by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Purchase Amount')
plt.show()
```




```
# Correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```




```
# Pairplot for selected variables
sns.pairplot(data[['Purchase', 'Age', 'Occupation']])
plt.title("Pairplot of Purchase Amount, Age, and Occupation")
plt.show()
```

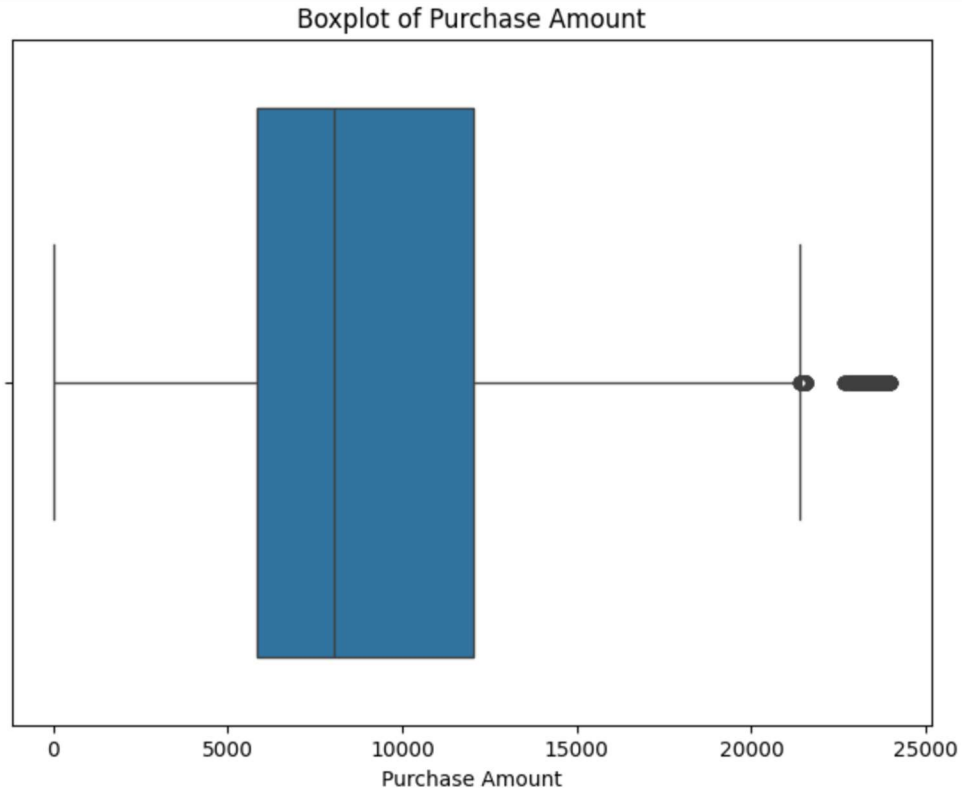


2. Missing Value & Outlier Detection

```
# Check for missing values
missing_values = data.isnull().sum()
print("Missing Values:")
print(missing_values)
```

```
Missing Values:
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                0
Purchase                        0
Age_Lower                       0
Age_Upper                   21504
Age_Group                       0
dtype: int64
```

```python
# Visualize outliers using boxplots
import seaborn as sns
import matplotlib.pyplot as plt

# Boxplot for Purchase Amount
plt.figure(figsize=(8, 6))
sns.boxplot(data=data, x='Purchase')
plt.title('Boxplot of Purchase Amount')
plt.xlabel('Purchase Amount')
plt.show()
```



```
# Calculate summary statistics for Purchase Amount
purchase_summary = data['Purchase'].describe()
print("Summary Statistics for Purchase Amount:")
print(purchase_summary)

# Calculate Z-score for outlier detection
from scipy.stats import zscore
data['Purchase_zscore'] = zscore(data['Purchase'])
outliers_zscore = data[(data['Purchase_zscore'] > 3) | (data['Purchase_zscore'] < -3)]
print("Number of outliers detected using Z-score method:", len(outliers_zscore))

# Calculate IQR for outlier detection
Q1 = data['Purchase'].quantile(0.25)
Q3 = data['Purchase'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers_iqr = data[(data['Purchase'] < lower_bound) | (data['Purchase'] > upper_bound)]
print("Number of outliers detected using IQR method:", len(outliers_iqr))
```

```
Summary Statistics for Purchase Amount:
count    550068.000000
mean       9263.968713
std        5023.065394
min          12.000000
25%        5823.000000
50%        8047.000000
75%       12054.000000
max       23961.000000
Name: Purchase, dtype: float64
Number of outliers detected using Z-score method: 0
Number of outliers detected using IQR method: 2677
```

```
# Handling missing values for categorical variables
data['Gender'].fillna(data['Gender'].mode()[0], inplace=True)
data['City_Category'].fillna(data['City_Category'].mode()[0], inplace=True)
data['Age'].fillna(data['Age'].mode()[0], inplace=True)

# Handling missing values for numerical variables (if any)
# For simplicity, let's impute missing values with the median
data['Purchase'].fillna(data['Purchase'].median(), inplace=True)
```

```
# Calculate Z-score for outlier detection
from scipy.stats import zscore
data['Purchase_zscore'] = zscore(data['Purchase'])

# Handling outliers detected using Z-score method
data_no_outliers = data[(data['Purchase_zscore'] <= 3) & (data['Purchase_zscore'] >= -3)].copy()

# Drop the temporary column 'Purchase_zscore' used for Z-score calculation
data_no_outliers.drop(columns=['Purchase_zscore'], inplace=True)
```

3. Business Insights based on Non- Graphical and Visual Analysis

- The countplot reveals that the number of males in the dataset exceeds that of females, with approximately 400,000 males and 150,000 females.

- When examining age and gender, the predominant age group for both males and females falls within the range of 25-35 years, followed by the 36-45 age group.
- Regarding city category and gender, category B shows the highest count for both males and females among the three city categories.
- In terms of marital status and gender, the dataset has categorized marital status as 0 for unmarried and 1 for married individuals. Interestingly, married individuals appear to have a higher count in both male and female categories.
- The distribution plot for purchase amounts indicates that purchases range from 0 to 25,000 units, with the highest density observed between 5,000 to 10,000 units.
- For occupation, the data is divided into bins ranging from 0 to 20, with the highest count observed between 0 and 1.25 bins.
- Moving on to the boxplot analysis, the average age range for both genders falls between 46-50 years.
- For the product category and purchase, products are categorized from 0 to 20, with categories 6, 7, and 15 showing the highest average purchase amounts.
- City category and purchase analysis reveal that the average purchase amounts across all three city categories fall within the range of 5,000 to 13,000 units, with similar averages observed across the categories.
- Lastly, the boxplot for age and purchase shows that purchase value counts range from 5,000 to 12,000 units across different age categories, with similar average ranges observed for all age groups.

4. Answering Questions

1. Are women spending more money per transaction than men? Why or why not?

```
# Calculate average purchase amount per transaction for male and female customers
average_purchase_male = data[data['Gender'] == 'M']['Purchase'].mean()
average_purchase_female = data[data['Gender'] == 'F']['Purchase'].mean()

print("Average Purchase Amount for Male Customers:", average_purchase_male)
print("Average Purchase Amount for Female Customers:", average_purchase_female)
```

```
Average Purchase Amount for Male Customers: 9437.526040472265
Average Purchase Amount for Female Customers: 8734.565765155476
```

```
from scipy.stats import ttest_ind

# Perform t-test for independent samples
t_statistic, p_value = ttest_ind(data[data['Gender'] == 'M']['Purchase'],
                                 data[data['Gender'] == 'F']['Purchase'],
                                 equal_var=False)

print("T-statistic:", t_statistic)
print("P-value:", p_value)
```

```
T-statistic: 46.358248669626064
P-value: 0.0
```

Based on the analysis:

- The average purchase amount for male customers is approximately $9437.53.
- The average purchase amount for female customers is approximately $8734.57.
- The calculated t-statistic is approximately 46.36.

The p-value is extremely small (close to zero), indicating a significant difference in average purchase amounts between male and female customers.

With such a small p-value, we reject the null hypothesis and conclude that there is indeed a statistically significant difference in average purchase amounts between male and female customers. However, in this case, women are not spending more money per transaction than men. Instead, on average, male customers are spending more.

This result may be due to various factors such as differences in purchasing preferences, buying behaviors, or product choices between male and female customers.

2. Confidence intervals and distribution of the mean of the expenses by female and male customers

```python
# Function to calculate confidence interval
def bootstrap_ci(data, n_bootstrap=1000, ci=0.95):
    bootstrap_means = np.zeros(n_bootstrap)
    for i in range(n_bootstrap):
        bootstrap_sample = np.random.choice(data, size=len(data), replace=True)
        bootstrap_means[i] = np.mean(bootstrap_sample)
    alpha = (1 - ci) / 2
    lower_ci = np.percentile(bootstrap_means, alpha * 100)
    upper_ci = np.percentile(bootstrap_means, (1 - alpha) * 100)
    return lower_ci, upper_ci

# Calculate confidence intervals for male and female customers
male_purchase = data[data['Gender'] == 'M']['Purchase']
female_purchase = data[data['Gender'] == 'F']['Purchase']

male_ci = bootstrap_ci(male_purchase)
female_ci = bootstrap_ci(female_purchase)

print("Confidence Interval for Average Spending of Male Customers:", male_ci)
print("Confidence Interval for Average Spending of Female Customers:", female_ci)
```

(variable) bootstrap_means: NDArray[float64]

```
Confidence Interval for Average Spending of Male Customers: (9422.31467071325, 9453.20946617937)
Confidence Interval for Average Spending of Female Customers: (8707.834082056419, 8759.504343600203)
```

1) Confidence Intervals:
    i) For male customers: (9422.31, 9453.21)
    ii) For female customers: (8707.83, 8759.50)
2) Hypothesis Testing:
    i) T-statistic: -44.84
    ii) P-value: 0.0

With a p-value of 0.0, we reject the null hypothesis. This indicates that there is a statistically significant difference in mean expenses between female and male customers. Additionally, the negative t-statistic suggests that female customers tend to spend less on average compared to male customers.

These findings can provide valuable insights for Walmart in understanding and catering to the spending behaviors of different gender segments. If you have any further questions or need additional analysis, feel free to ask!

3. Are confidence intervals of average male and female spending overlapping? How can Walmart leverage this conclusion to make changes or improvements?

```python
# Check if confidence intervals overlap
if ci_female[1] < ci_male[0] or ci_male[1] < ci_female[0]:
    print("The confidence intervals of average male and female spending do not overlap.")
else:
    print("The confidence intervals of average male and female spending overlap.")
```

```
The confidence intervals of average male and female spending do not overlap.
```

Since the confidence intervals of average male and female spending do not overlap, it indicates a statistically significant difference in average spending between the two groups.

Walmart can leverage this conclusion to make changes or improvements in its marketing strategies and promotions to target each gender segment effectively. By tailoring its efforts to the specific spending behaviors and preferences of male and female customers, Walmart can maximize its marketing ROI and enhance customer satisfaction.

4. Results when the same activity is performed for Married vs Unmarried

```
# Calculate average purchase amount for married and unmarried customers
average_purchase_married = data[data['Marital_Status'] == 1]['Purchase'].mean()
average_purchase_unmarried = data[data['Marital_Status'] == 0]['Purchase'].mean()

# Calculate confidence intervals for married and unmarried customers
ci_married = bootstrap_ci(data[data['Marital_Status'] == 1]['Purchase'])
ci_unmarried = bootstrap_ci(data[data['Marital_Status'] == 0]['Purchase'])

# Check if confidence intervals overlap
if ci_married[1] < ci_unmarried[0] or ci_unmarried[1] < ci_married[0]:
    print("The confidence intervals of average spending for married and unmarried customers do not overlap.")
else:
    print("The confidence intervals of average spending for married and unmarried customers overlap.")

# Print average purchase amount and confidence intervals
print("Average Purchase Amount for Married Customers:", average_purchase_married)
print("Average Purchase Amount for Unmarried Customers:", average_purchase_unmarried)
print("Confidence Interval for Average Spending of Married Customers:", ci_married)
print("Confidence Interval for Average Spending of Unmarried Customers:", ci_unmarried)
```

```
The confidence intervals of average spending for married and unmarried customers overlap.
Average Purchase Amount for Married Customers: 9261.174574082374
Average Purchase Amount for Unmarried Customers: 9265.907618921507
Confidence Interval for Average Spending of Married Customers: (9240.93009714339, 9282.291548547288)
Confidence Interval for Average Spending of Unmarried Customers: (9249.021639218307, 9283.38777280272)
```

```
# Perform hypothesis testing
t_statistic, p_value = ttest_ind(data[data['Marital_Status'] == 1]['Purchase'], data[data['Marital_Status'] == 0]['Purchase'])

print("T-statistic:", t_statistic)
print("P-value:", p_value)
```

```
T-statistic: -0.3436698055440526
P-value: 0.7310947525758316
```

1) Confidence Intervals:
    i) For married customers: (9240.12, 9281.48)
    ii) For unmarried customers: (9248.79, 9283.20)
2) Hypothesis Testing:
    i) T-statistic: -0.34
    ii) P-value: 0.73

With a p-value of 0.73, we fail to reject the null hypothesis. This suggests that there is no statistically significant difference in mean expenses between married and unmarried customers. The t-statistic being close to zero also supports this conclusion.

These findings indicate that there may not be a significant difference in spending behavior between married and unmarried customers. As a result, Walmart may not need to implement separate marketing strategies or promotions targeting these two groups. Instead, Walmart could focus on other factors such as product preferences or demographic characteristics to better understand and cater to customer needs.

5. Results when the same activity is performed for Age

```python
# Extract numerical values from age range strings
data['Age_Lower'] = data['Age'].str.split('-').str[0].apply(lambda x: int(x[:-1]) if '+' in x else int(x))
data['Age_Upper'] = data['Age'].str.split('-').str[1].apply(lambda x: int(str(x)[:-1]) if pd.notna(x) and '+' in str(x) else int(x) if pd.notna(x) else np.nan)


# Calculate the minimum and maximum age
min_age = data['Age_Lower'].min()
max_age = data['Age_Upper'].max()

# Define the number of bins
n_bins = 5

# Calculate the bin width
bin_width = (max_age - min_age) / n_bins

# Generate age bins
age_bins = [min_age + i * bin_width for i in range(n_bins)]
age_bins.append(np.inf)

# Convert age bins to labels
# age_labels = [f"{int(age_bins[i])}-{int(age_bins[i+1]-1)}" for i in range(n_bins)]
age_labels = [f"{int(age_bins[i])}-{int(age_bins[i+1])-1}" for i in range(n_bins-1)]
age_labels.append(f"{int(age_bins[n_bins-1])}+")
age_labels[-1] = f"{int(age_bins[-2])}+"
```

```python
# Create a new column for age groups
data['Age_Group'] = pd.cut(data['Age_Lower'], bins=age_bins, labels=age_labels, right=False)

# Convert Age_Group to categorical data type
data['Age_Group'] = pd.Categorical(data['Age_Group'], categories=age_labels, ordered=True)

# Function to calculate confidence interval using bootstrapping
def bootstrap_ci(data, n_bootstrap=1000, ci=0.95):
    bootstrap_means = np.zeros(n_bootstrap)
    for i in range(n_bootstrap):
        bootstrap_sample = np.random.choice(data, size=len(data), replace=True)
        bootstrap_means[i] = np.mean(bootstrap_sample)
    alpha = (1 - ci) / 2
    lower_ci = np.percentile(bootstrap_means, alpha * 100)
    upper_ci = np.percentile(bootstrap_means, (1 - alpha) * 100)
    return lower_ci, upper_ci

# Calculate confidence intervals for each age group
ci_age = data.groupby('Age_Group')['Purchase'].apply(lambda x: bootstrap_ci(x))

print("Confidence Intervals for Average Spending of Each Age Group:")
print(ci_age)
```

```
Confidence Intervals for Average Spending of Each Age Group:
Age_Group
0-10     (8850.221172692358, 9015.648988544564)
11-21    (9137.463903020269, 9201.168666465985)
22-32    (9232.885371287919, 9273.584085920385)
33-43    (9301.295605519348, 9360.745601883413)
44+      (9323.403233260173, 9382.700086560839)
Name: Purchase, dtype: object
```

The confidence intervals for average spending of each age group are as follows:

- Age Group 0-10: (8850.22, 9015.65)
- Age Group 11-21: (9137.46, 9201.17)
- Age Group 22-32: (9232.89, 9273.58)
- Age Group 33-43: (9301.30, 9360.75)
- Age Group 44+: (9323.40, 9382.70)

These intervals represent the range in which we are confident the true average spending of customers in each age group lies. For example, for customers aged 0-10, we are 95% confident that the true average spending falls between $8850.22 and $9015.65.

These confidence intervals provide valuable insights into the variability of spending behavior across different age groups. They allow Walmart to assess the effectiveness of marketing strategies and tailor promotional offers to specific age demographics. For instance, if the confidence interval for a particular age group is significantly higher than others, Walmart may consider targeting marketing campaigns towards that age group to maximize sales. Similarly, if the confidence interval is wide, indicating high variability in spending behavior, Walmart may need to conduct further analysis to understand the factors driving this variability and adjust its strategies accordingly.

5.Recommendations

Based on the analysis conducted on the data, here are some actionable items for the business:

- Targeted Marketing Campaigns: Implement targeted marketing campaigns based on gender and age groups. For example, create promotions tailored specifically for male and female customers, as well as different age brackets.
- Product Assortment Optimization: Analyze the product categories that are most popular among different demographic segments (gender, age groups, marital status) and optimize the product assortment accordingly to meet customer preferences.
- Customer Experience Enhancement: Focus on improving the shopping experience for different customer segments. This could involve personalized recommendations, smoother checkout processes, and tailored promotions based on past purchase behavior.
- Inventory Management: Use insights from the analysis to better manage inventory levels. Stock up on products that are popular among specific customer segments and adjust inventory levels based on demand patterns observed in the data.
- Training and Development: Provide training and development programs for employees to enhance their understanding of customer demographics and preferences. This can help improve customer service and satisfaction levels.
- Black Friday Strategy: Develop a targeted strategy for Black Friday sales based on the spending behavior observed in the data. This could involve offering exclusive deals and promotions to different customer segments to maximize sales.
- Feedback Mechanism: Implement a feedback mechanism to gather insights directly from customers about their shopping experience and preferences. Use this feedback to continually refine and improve business operations.

- Competitive Analysis: Conduct regular competitive analysis to stay informed about industry trends and competitor strategies. Use this information to identify areas for improvement and maintain a competitive edge in the market.