```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm
```

```
walmart_data = pd.read_csv('/content/walmart_data.csv')
walmart_data.head(20)
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City
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	
5	1000003	P00193542	M	26-35	15	A	
6	1000004	P00184942	M	46-50	7	B	
7	1000004	P00346142	M	46-50	7	B	
8	1000004	P0097242	M	46-50	7	B	
9	1000005	P00274942	M	26-35	20	A	
10	1000005	P00251242	M	26-35	20	A	
11	1000005	P00014542	M	26-35	20	A	
12	1000005	P00031342	M	26-35	20	A	

```
walmart_data.shape
```

```
(250446, 10)
```

```
walmart_data.dtypes
```

```
User_ID                         int64
Product_ID                     object
Gender                         object
Age                            object
Occupation                    float64
City_Category                  object
Stay_In_Current_City_Years     object
Marital_Status                float64
Product_Category              float64
Purchase                      float64
dtype: object
```

```
walmart_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250446 entries, 0 to 250445
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     250446 non-null  int64
 1   Product_ID                  250446 non-null  object
 2   Gender                      250445 non-null  object
 3   Age                         250445 non-null  object
 4   Occupation                  250445 non-null  float64
 5   City_Category               250445 non-null  object
 6   Stay_In_Current_City_Years  250445 non-null  object
 7   Marital_Status              250445 non-null  float64
 8   Product_Category            250445 non-null  float64
 9   Purchase                    250445 non-null  float64
```

```
dtypes: float64(4), int64(1), object(5)
memory usage: 19.1+ MB
```

```
walmart_data.isna().sum()
```

```
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
dtype: int64
```

```
walmart_data.describe()
```

	User_ID	Occupation	Marital_Status	Product_Category	Purchase
count	2.504460e+05	250445.000000	250445.000000	250445.000000	250445.000000
mean	1.002913e+06	8.073805	0.410489	5.294588	9313.668754
std	1.735069e+03	6.529494	0.491924	3.744684	4967.982450
min	1.000001e+06	0.000000	0.000000	1.000000	185.000000
25%	1.001395e+06	2.000000	0.000000	1.000000	5862.000000
50%	1.002869e+06	7.000000	0.000000	5.000000	8058.000000
75%	1.004384e+06	14.000000	1.000000	8.000000	12055.000000
max	1.006040e+06	20.000000	1.000000	18.000000	23961.000000

```
walmart_data["Age"].value_counts()
```

```
26-35    99577
36-45    49795
18-25    45937
46-50    20622
51-55    17716
55+       9889
0-17      6909
Name: Age, dtype: int64
```

```
walmart_data["Gender"].value_counts()
```

```
M    188692
F     61753
Name: Gender, dtype: int64
```

```
walmart_data["City_Category"].value_counts()
```

```
B    105469
C     77316
A     67660
Name: City_Category, dtype: int64
```

```
walmart_data["Marital_Status"].value_counts()
```

```
0.0    147640
1.0    102805
Name: Marital_Status, dtype: int64
```

```
walmart_data["Stay_In_Current_City_Years"].value_counts()
```

```
1     88077
2     46092
3     43461
4+    38849
0     33966
Name: Stay_In_Current_City_Years, dtype: int64
```

```
for i in walmart_data.columns:
  print(i , ":",walmart_data[i].nunique())
```

```
User_ID : 5891
Product_ID : 3503
Gender : 2
Age : 7
```

```
Occupation : 21
City_Category : 3
Stay_In_Current_City_Years : 5
Marital_Status : 2
Product_Category : 18
Purchase : 16345
```

CONVERSION OF CATEGORICAL ATTRIBUTES TO 'category'

```
categorical_columns = ['Gender', 'Age', 'City_Category', 'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category']
walmart_data[categorical_columns] = walmart_data[categorical_columns].astype('category')
print(walmart_data.dtypes)
walmart_data
```

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

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_
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	
...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	
550064	1006035	P00375436	F	26-35	1	C	

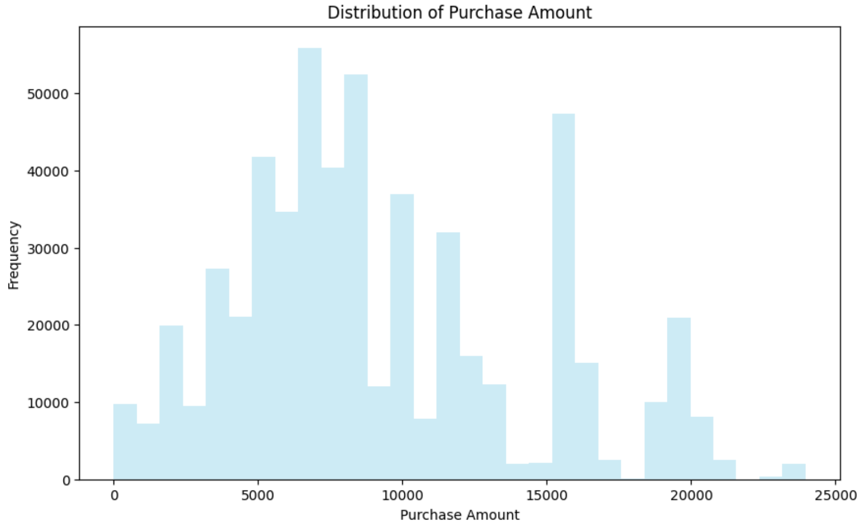
INSIGHTS :

1. People belonging to age group 26-35 does the highest shopping from Walamart.
2. Compared to women men tend to shop more from Walamart.
3. Mostly people have stayed for 1 year in a city.
4. The minimum purchase amount is 185 and the maximum is 23,961 and average purchase being 9313.

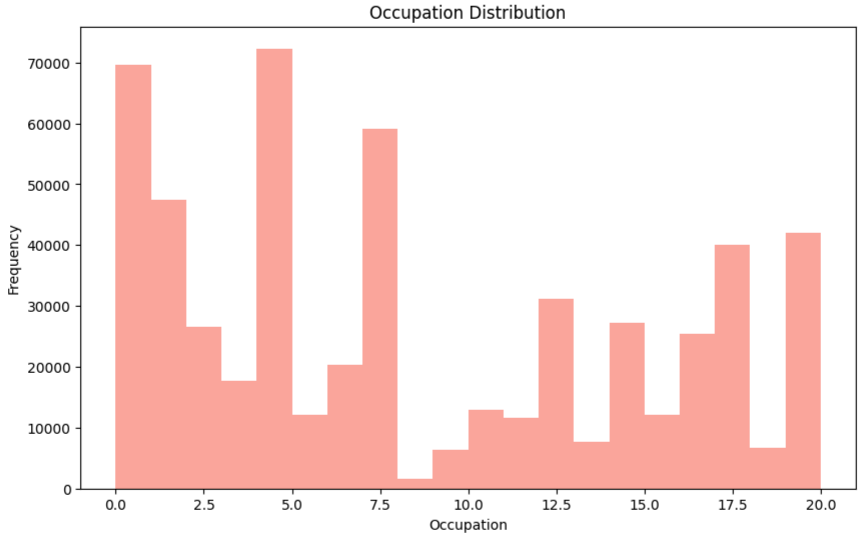
UNIVARIATE AND BIVARIATE ANALYSIS

```
plt.figure(figsize=(10, 6))
sns.distplot(walmart_data['Purchase'], kde=False, bins=30, color='skyblue')
plt.title('Distribution of Purchase Amount')
plt.xlabel('Purchase Amount')
plt.ylabel('Frequency')
plt.show()
```

```
<ipython-input-32-efb10f4f2653>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(walmart_data['Purchase'], kde=False, bins=30, color='skyblue')
```

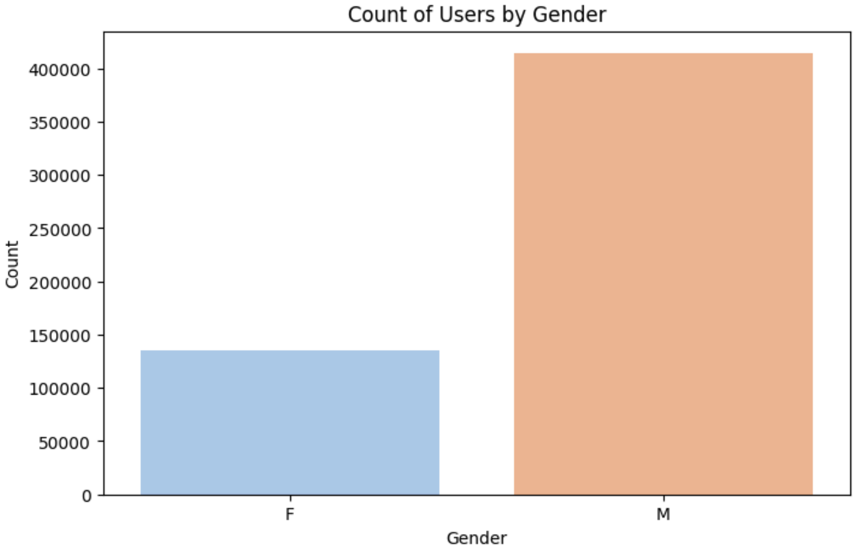



```
plt.figure(figsize=(10, 6))
plt.hist(walmart_data['Occupation'], bins=20, color='salmon', alpha=0.7)
plt.title('Occupation Distribution')
plt.xlabel('Occupation')
plt.ylabel('Frequency')
plt.show()
```



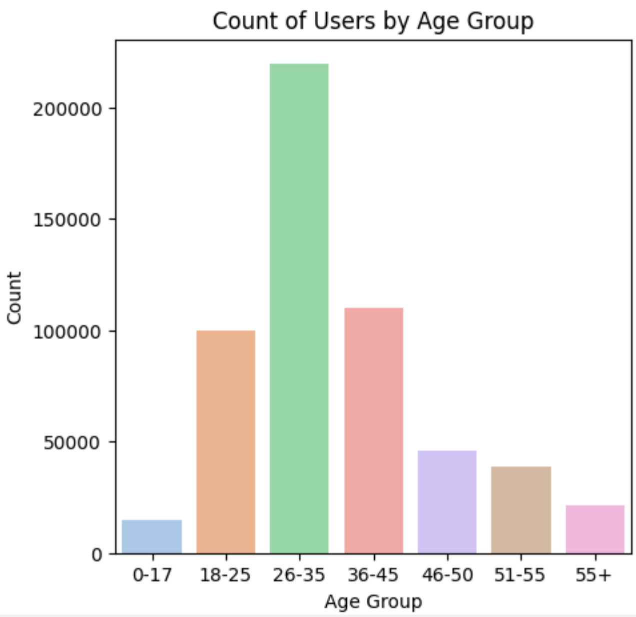
```
plt.figure(figsize=(8, 5))
sns.countplot(data=walmart_data, x='Gender', palette='pastel')
plt.title('Count of Users by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```

```
<ipython-input-34-2d1665625b6f>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.countplot(data=walmart_data, x='Gender', palette='pastel')
```



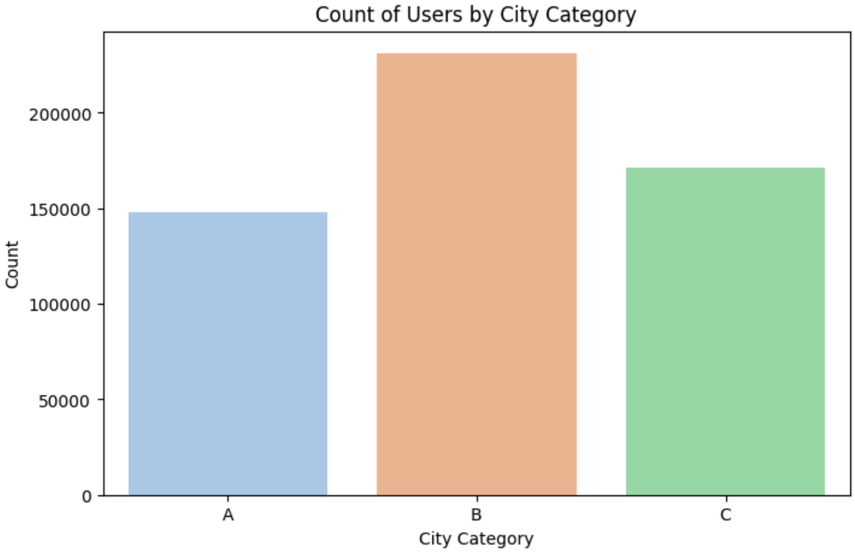

```
plt.figure(figsize=(5, 5))
sns.countplot(data=walmart_data, x='Age', palette='pastel')
plt.title('Count of Users by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Count')
plt.show()
```

```
<ipython-input-39-49c44ffbfb92>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.countplot(data=walmart_data, x='Age', palette='pastel')
```



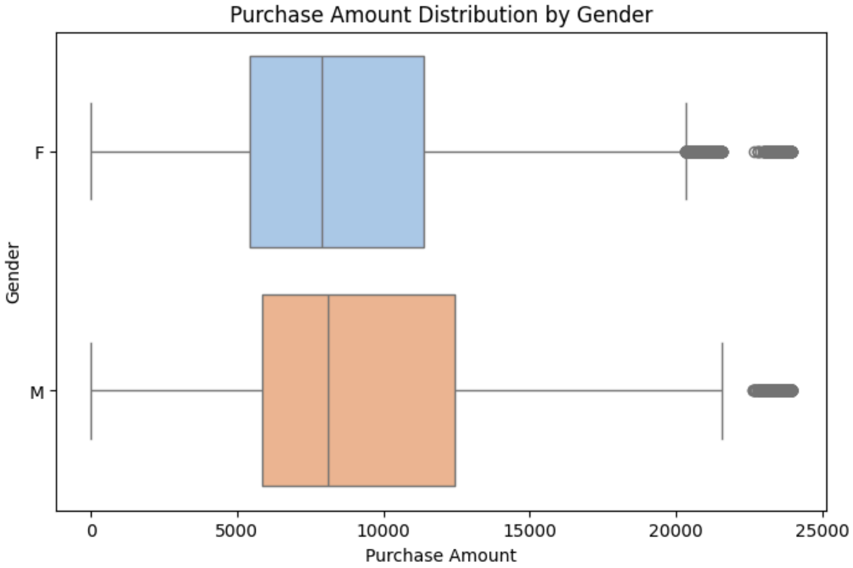

```
plt.figure(figsize=(8, 5))
sns.countplot(data=walmart_data, x='City_Category', palette='pastel')
plt.title('Count of Users by City Category')
plt.xlabel('City Category')
plt.ylabel('Count')
plt.show()
```

```
<ipython-input-36-8a240ffe0304>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.countplot(data=walmart_data, x='City_Category', palette='pastel')
```

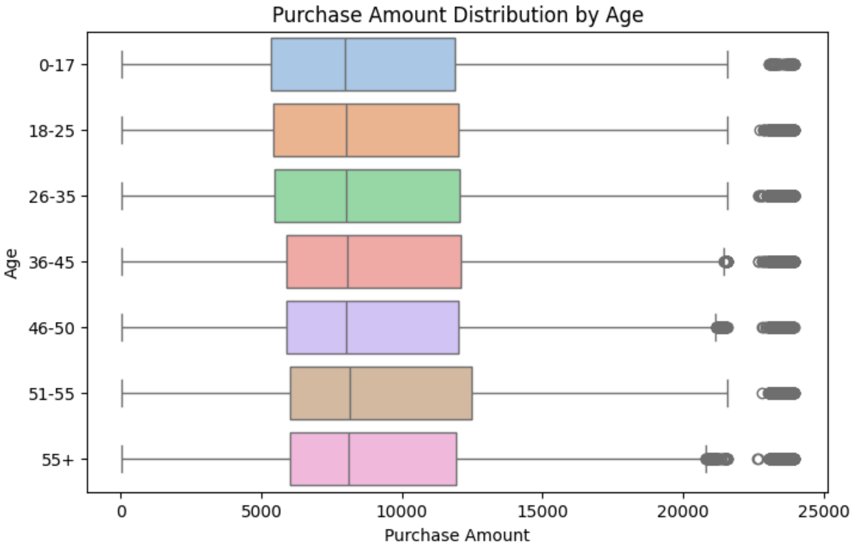



```
categorical_columns = ['Gender', 'Age', 'City_Category', 'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category']
for column in categorical_columns:
    plt.figure(figsize=(8, 5))
    sns.boxplot(data=walmart_data, x='Purchase', y=column, palette='pastel', orient='h')
    plt.title(f'Purchase Amount Distribution by {column}')
    plt.xlabel('Purchase Amount')
    plt.ylabel(column)
    plt.show()
```

```
<ipython-input-45-9ea8110dfa47>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.boxplot(data=walmart_data, x='Purchase', y=column, palette='pastel', orient='h'
```

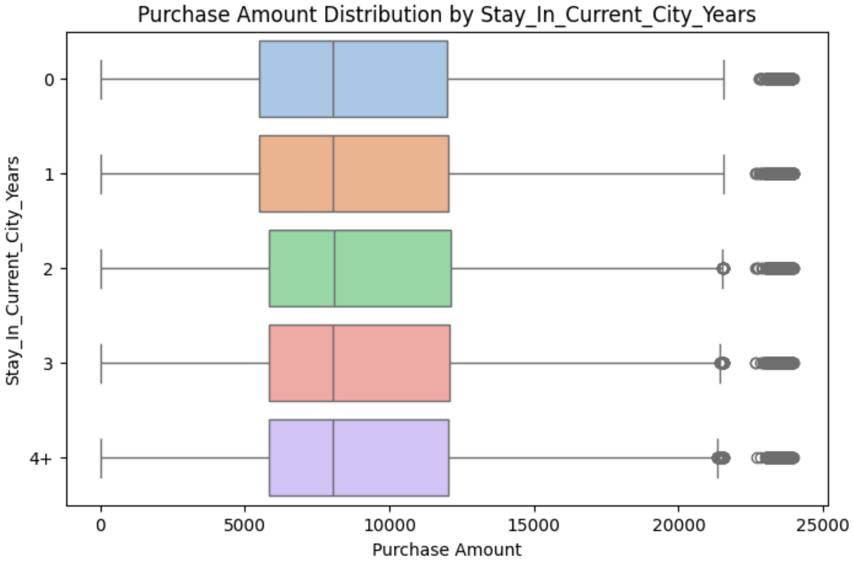



```
<ipython-input-45-9ea8110dfa47>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.boxplot(data=walmart_data, x='Purchase', y=column, palette='pastel', orient='h'
```

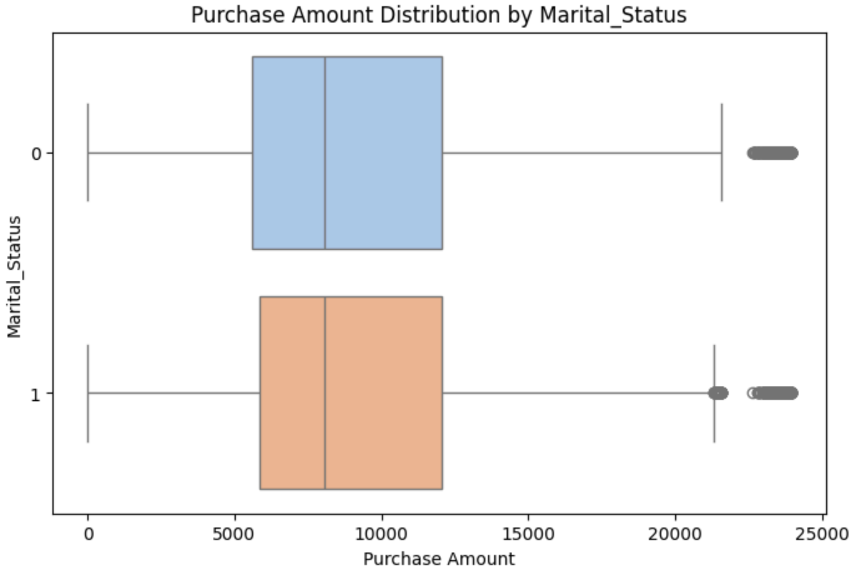



```
<ipython-input-45-9ea8110dfa47>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.boxplot(data=walmart_data, x='Purchase', y=column, palette='pastel', orient='h'
```

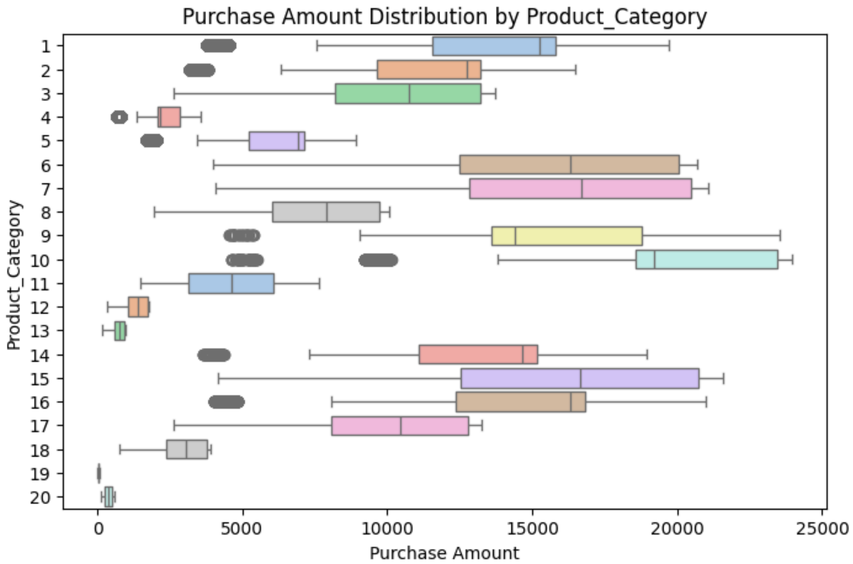
```
<ipython-input-45-9ea8110dfa47>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.boxplot(data=walmart_data, x='Purchase', y=column, palette='pastel', orient='h'
```




```
<ipython-input-45-9ea8110dfa47>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.boxplot(data=walmart_data, x='Purchase', y=column, palette='pastel', orient='h'
```




```
<ipython-input-45-9ea8110dfa47>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.boxplot(data=walmart_data, x='Purchase', y=column, palette='pastel', orient='h'
```

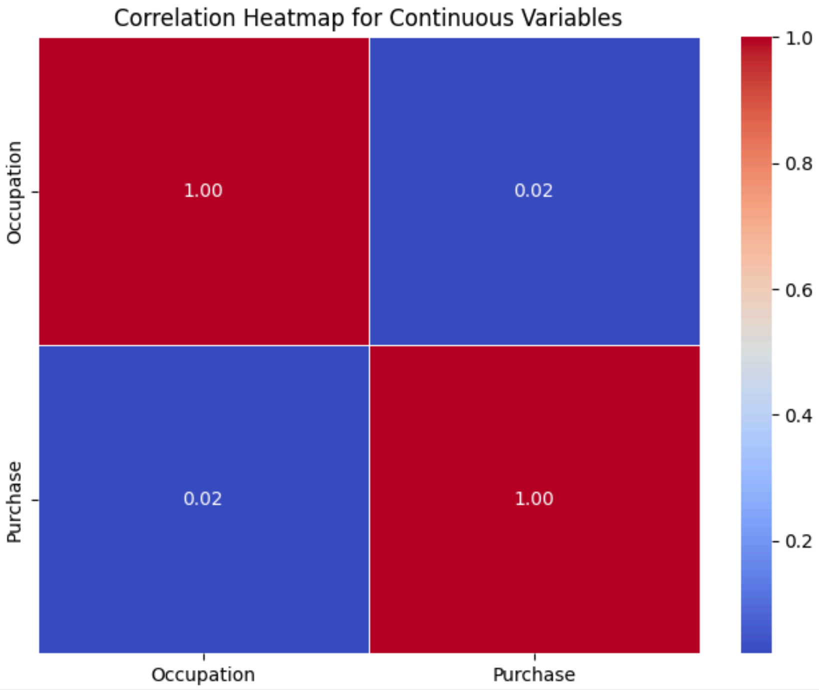


INSIGHTS:

1. Females have more outliers compared to males.
2. Outliers are present in all the age groups.
3. City C has very less number of outliers as compared to City A.
4. The Marital status is almost equal for married as well as single people.

```
continuous_columns = ['Occupation', 'Marital_Status', 'Product_Category', 'Purchase']
plt.figure(figsize=(8, 6))
sns.heatmap(walmart_data[continuous_columns].corr(), annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap for Continuous Variables')
plt.show()
```

```
<ipython-input-50-e19061d64697>:3: FutureWarning: The default value of numeric_only i
  sns.heatmap(walmart_data[continuous_columns].corr(), annot=True, cmap='coolwarm', f
```



Missing Value & Outlier Detection

```
walmart_data.isnull().sum()
```

```
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
dtype: int64
```

```
# Calculate IQR for Purchase column
Q1 = walmart_data['Purchase'].quantile(0.25)
Q3 = walmart_data['Purchase'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers_count = len(walmart_data[(walmart_data['Purchase'] < lower_bound) | (walmart_data['Purchase'] > upper_bound)])

print("Number of outliers in Purchase column :", outliers_count)
```

```
Number of outliers in Purchase column (using IQR method): 2677
```

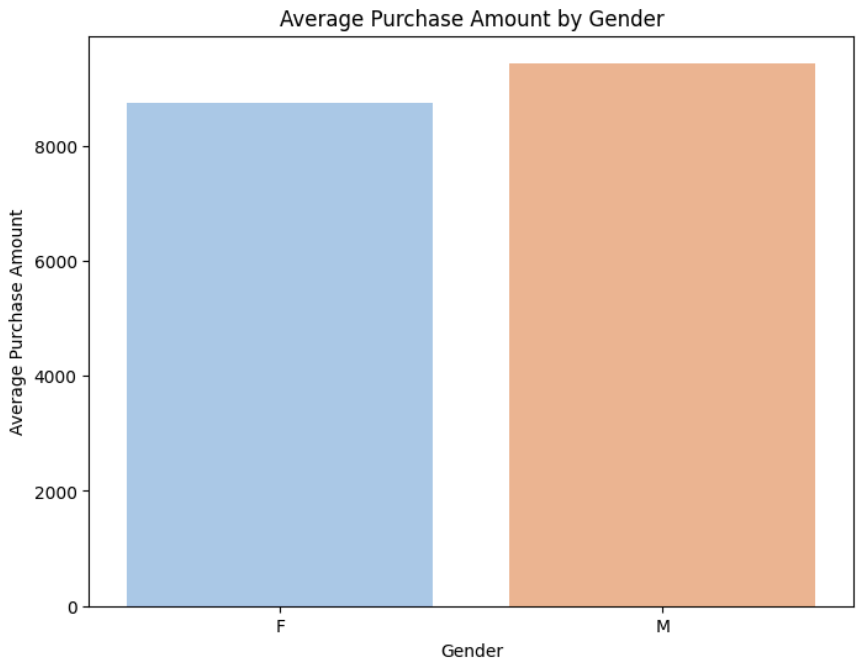
Are women spending more money per transaction than men? Why or Why not?

```
gender_data = walmart_data.dropna(subset=['Gender'])
average_purchase_by_gender = gender_data.groupby('Gender')['Purchase'].mean()
print("Average Purchase Amount by Gender:")
print(average_purchase_by_gender)
```

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

```
plt.figure(figsize=(8, 6))
sns.barplot(x=average_purchase_by_gender.index, y=average_purchase_by_gender.values, palette='pastel')
plt.title('Average Purchase Amount by Gender')
plt.xlabel('Gender')
plt.ylabel('Average Purchase Amount')
plt.show()
```

```
<ipython-input-72-8a5c8dae3a22>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.barplot(x=average_purchase_by_gender.index, y=average_purchase_by_gender.values
```



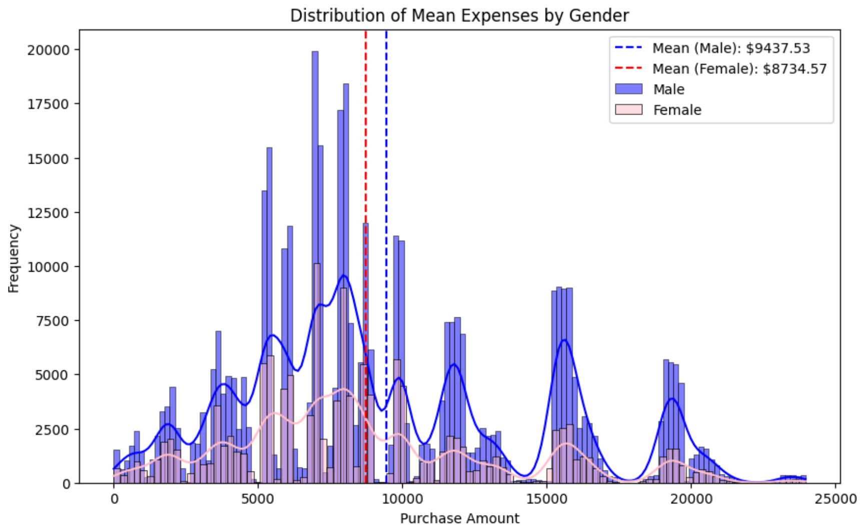
Men are spending more money per transaction.

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

```
gender_data = walmart_data.dropna(subset=['Gender'])

# Separate data for male and female customers
male_data = gender_data[gender_data['Gender'] == 'M']['Purchase']
female_data = gender_data[gender_data['Gender'] == 'F']['Purchase']
```

```
# Calculate mean and standard deviation of expenses for male and female customers
mean_male = male_data.mean()
std_male = male_data.std()
mean_female = female_data.mean()
std_female = female_data.std()

# Set the confidence level
confidence_level = 0.95

# Calculate the margin of error for male customers
margin_of_error_male = norm.ppf((1 + confidence_level) / 2) * (std_male / np.sqrt(len(male_data)))

# Calculate the margin of error for female customers
margin_of_error_female = norm.ppf((1 + confidence_level) / 2) * (std_female / np.sqrt(len(female_data)))

# Calculate the confidence intervals for male and female customers
ci_male = (mean_male - margin_of_error_male, mean_male + margin_of_error_male)
ci_female = (mean_female - margin_of_error_female, mean_female + margin_of_error_female)

print("Confidence Intervals for Male Customers:", ci_male)
print("Confidence Intervals for Female Customers:", ci_female)

# Visualize the distribution of mean expenses for male and female customers
plt.figure(figsize=(10, 6))
sns.histplot(male_data, kde=True, color='blue', label='Male')
sns.histplot(female_data, kde=True, color='pink', label='Female')
plt.axvline(mean_male, color='blue', linestyle='--', label=f'Mean (Male): ${mean_male:.2f}')
plt.axvline(mean_female, color='red', linestyle='--', label=f'Mean (Female): ${mean_female:.2f}')
plt.xlabel('Purchase Amount')
plt.ylabel('Frequency')
plt.title('Distribution of Mean Expenses by Gender')
plt.legend()
plt.show()
```

```
Confidence Intervals for Male Customers: (9422.01944736257, 9453.032633581959)
Confidence Intervals for Female Customers: (8709.21154714068, 8759.919983170272)
```



INSIGHTS:

1. Male customers tend to have slightly higher mean expenses compared to female customers

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

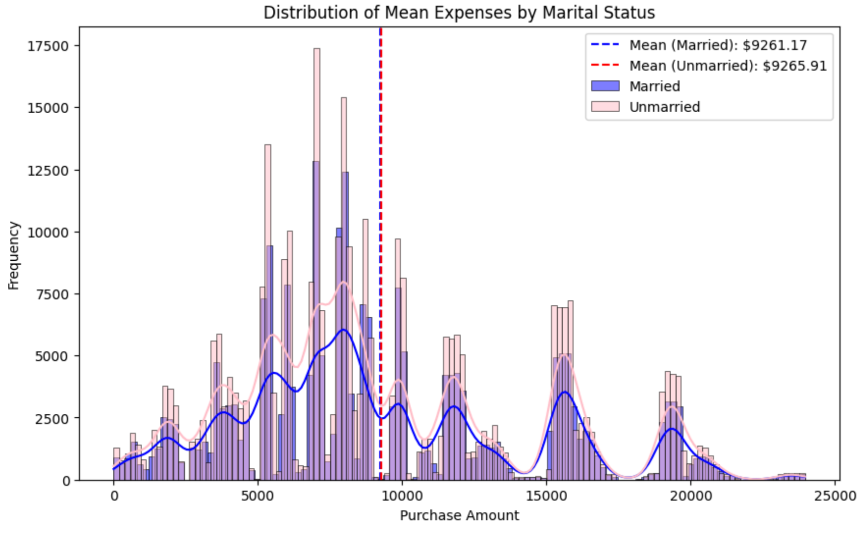
95% of the times:

1. Average amount spend by male customer will lie in between: (9422.01944736257, 9453.032633581959)
2. Average amount spend by female customer will lie in between: (8709.21154714068, 8759.919983170272)

Results when the same activity is performed for Married vs Unmarried

```
marital_data = walmart_data.dropna(subset=['Marital_Status'])

# Separate data for married and unmarried customers
married_data = marital_data[marital_data['Marital_Status'] == 1]['Purchase']
unmarried_data = marital_data[marital_data['Marital_Status'] == 0]['Purchase']

# Calculate mean and standard deviation of expenses for married and unmarried customers
mean_married = married_data.mean()
std_married = married_data.std()
mean_unmarried = unmarried_data.mean()
std_unmarried = unmarried_data.std()

# Set the confidence level
confidence_level = 0.95

# Calculate the margin of error for married customers
margin_of_error_married = norm.ppf((1 + confidence_level) / 2) * (std_married / np.sqrt(len(married_data)))

# Calculate the margin of error for unmarried customers
margin_of_error_unmarried = norm.ppf((1 + confidence_level) / 2) * (std_unmarried / np.sqrt(len(unmarried_data)))

# Calculate the confidence intervals for married and unmarried customers
ci_married = (mean_married - margin_of_error_married, mean_married + margin_of_error_married)
ci_unmarried = (mean_unmarried - margin_of_error_unmarried, mean_unmarried + margin_of_error_unmarried)

print("Confidence Intervals for Married Customers:", ci_married)
print("Confidence Intervals for Unmarried Customers:", ci_unmarried)

# Visualize the distribution of mean expenses for married and unmarried customers
plt.figure(figsize=(10, 6))
sns.histplot(married_data, kde=True, color='blue', label='Married')
sns.histplot(unmarried_data, kde=True, color='pink', label='Unmarried')
plt.axvline(mean_married, color='blue', linestyle='--', label=f'Mean (Married): ${mean_married:.2f}')
plt.axvline(mean_unmarried, color='red', linestyle='--', label=f'Mean (Unmarried): ${mean_unmarried:.2f}')
plt.xlabel('Purchase Amount')
plt.ylabel('Frequency')
plt.title('Distribution of Mean Expenses by Marital Status')
plt.legend()
plt.show()
```

```
Confidence Intervals for Married Customers: (9240.460427057078, 9281.888721107669)
Confidence Intervals for Unmarried Customers: (9248.61641818668, 9283.198819656332)
```



INSIGHTS: 95% of the times:

1. Average amount spend by unmarried customer will lie in between: (9248.61641818668, 9283.198819656332).
2. Average amount spend by married customer will lie in between: (9240.460427057078, 9281.888721107669)

Results when the same activity is performed for Age

```
age_data = walmart_data.dropna(subset=['Age'])

# Separate data for different age groups (assuming 'Age' is categorical)
age_groups = sorted(age_data['Age'].unique())

# Calculate mean and standard deviation of expenses for each age group
means = []
stds = []
for age_group in age_groups:
    age_group_data = age_data[age_data['Age'] == age_group]['Purchase']
    mean = age_group_data.mean()
    std = age_group_data.std()
    means.append(mean)
    stds.append(std)

# Set the confidence level
confidence_level = 0.95

# Calculate the confidence intervals for each age group
cis = []
for mean, std, size in zip(means, stds, [len(age_data[age_data['Age'] == age_group]) for age_group in age_groups]):
    margin_of_error = norm.ppf((1 + confidence_level) / 2) * (std / np.sqrt(size))
    ci = (mean - margin_of_error, mean + margin_of_error)
    cis.append(ci)

# Print confidence intervals for each age group
for age_group, ci in zip(age_groups, cis):
    print(f"Confidence Intervals for {age_group} age group:", ci)

# Visualize the distribution of mean expenses for each age group
plt.figure(figsize=(10, 6))
for age_group, mean, ci in zip(age_groups, means, cis):
    age_group_data = age_data[age_data['Age'] == age_group]['Purchase']
    sns.histplot(age_group_data, kde=True, label=f'Age {age_group}')
    plt.axvline(mean, color='blue', linestyle='--', label=f'Mean (Age {age_group}): ${mean:.2f}')
plt.xlabel('Purchase Amount')
plt.ylabel('Frequency')
plt.title('Distribution of Mean Expenses by Age Group')
plt.legend()
plt.show()
```

```
Confidence Intervals for 0-17 age group: (8851.947970542686, 9014.981310347262)
Confidence Intervals for 18-25 age group: (9138.407948753442, 9200.919263769136)
Confidence Intervals for 26-35 age group: (9231.733676400028, 9273.647589339747)
Confidence Intervals for 36-45 age group: (9301.669410965314, 9361.031978870433)
Confidence Intervals for 46-50 age group: (9163.085142648752, 9254.166252287903)
Confidence Intervals for 51-55 age group: (9483.991472776577, 9585.624589143894)
Confidence Intervals for 55+ age group: (9269.29883441773, 9403.262084481079)
```