Business Problem:

The Management team at Walmart Inc. wants to analyse the customer purchase behaviour (specifically, purchase amount) against the customer's gender and the various other factors to help the business make better decisions. They want to understand if the spending habits differ between male and female customers: Do women spend more on Black Friday than men? (Assume 50 million customers are male and 50 million are female).

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

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

After running the cell above, you'll be prompted to authorize Colab to access your Google Drive. Once authorized, your Drive will be mounted at `/content/drive`.

Now, you can read the CSV file. Replace `'Path/to/your/file/walmart_data.csv'` with the actual path to your file within your Google Drive. You can find the path by navigating to the file in the file explorer on the left side of Colab, right-clicking on the file, and selecting "Copy path".

```
import pandas as pd

df = pd.read_csv('/content/drive/MyDrive/walmart_data.csv') # Replace with the
display(df.head())
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Curre
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	

```
df.info()
```

```
<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
```

```
df[['Occupation','Marital_Status','Product_Category']] = df[['Occupation','Mari
```

```
df.info()
```

```
<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  object
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  object
 8   Product_Category            550068 non-null  object
 9   Purchase                    550068 non-null  int64 
dtypes: int64(2), object(8)
memory usage: 42.0+ MB
```

-Categorical Variables:

Product_ID Gender Age City_Category Stay_In_Current_City_Years Product Category Marital_Status

-Numerical Variables:

Purchase

```
df.shape
```

```
(550068, 10)
```

```
#Statistical Summary

df.describe()
```

	User_ID	Purchase
count	5.500680e+05	550068.000000
mean	1.003029e+06	9263.968713
std	1.727592e+03	5023.065394
min	1.000001e+06	12.000000
25%	1.001516e+06	5823.000000
50%	1.003077e+06	8047.000000
75%	1.004478e+06	12054.000000
max	1.006040e+06	23961.000000

```
df.isna().sum()
```

	0
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

No null values in the dataframe.

```
df.duplicated().sum()
```

```
np.int64(0)
```

No duplicate values in the columns.

```
df.head(10)
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Curre
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	
9	1000005	P00274942	M	26-35	20	A	

```
print("Product_ID:",df['Product_ID'].unique())
print("Product_IDs:", df['Product_ID'].nunique())
```

```
Product_ID: ['P00069042' 'P00248942' 'P00087842' ... 'P00370293' 'P00371644
 'P00370853']
Product_IDs: 3631
```

```
print("Marital Statuses:",df['Marital_Status'].unique())
```

```
Marital Statuses: [0 1]
```

```
print("Types of gender:",df['Gender'].unique())
```

```
Types of gender: ['F' 'M']
```

```
print("Types of Occupation:",df['Occupation'].unique())
print("No of occupations:", df['Occupation'].nunique())
```

```
Types of Occupation: [10 16 15 7 20 9 1 12 17 0 3 4 11 8 19 2 18 5 14 13 6]
No of occupations: 21
```

```
print("Product_Categories:",df['Product_Category'].unique())
print("No of product categories:", df['Product_Category'].nunique())
```

```
Product_Categories: [3 1 12 8 5 4 2 6 14 11 13 15 7 16 18 10 17 9 20 19]
No of product categories: 20
```

```
print("Age groups:",df['Age'].unique())
print("Age groups:", df['Age'].nunique())
```

```
Age groups: ['0-17' '55+' '26-35' '46-50' '51-55' '36-45' '18-25']
Age groups: 7
```

Value counts for following columns:

```
categorical_cols = ['Gender', 'Age', 'Occupation', 'City_Category',
'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category']

df[categorical_cols].melt().groupby(['variable',
'value'])[['value']].count()/len(df)
```

		value
variable	**value**	
Age	**0-17**	0.027455
	18-25	0.181178
	26-35	0.399200
	36-45	0.199999
	46-50	0.083082
	51-55	0.069993
	55+	0.039093
City_Category	**A**	0.268549
	B	0.420263

	C	0.311189
Gender	**F**	0.246895
	M	0.753105
Marital_Status	**0**	0.590347
	1	0.409653
Occupation	**0**	0.126599
	1	0.086218
	2	0.048336
	3	0.032087
	4	0.131453
	5	0.022137
	6	0.037005
	7	0.107501
	8	0.002811
	9	0.011437
	10	0.023506
	11	0.021063
	12	0.056682
	13	0.014049
	14	0.049647
	15	0.022115
	16	0.046123
	17	0.072796
	18	0.012039
	19	0.015382
	20	0.061014
Product_Category	**1**	0.255201
	2	0.043384
	3	0.036746

	4	0.021366
	5	0.274390
	6	0.037206
	7	0.006765
	8	0.207111
	9	0.000745
	10	0.009317
	11	0.044153
	12	0.007175
	13	0.010088
	14	0.002769
	15	0.011435
	16	0.017867
	17	0.001051
	18	0.005681
	19	0.002914
	20	0.004636
Stay_In_Current_City_Years	0	0.135252
	1	0.352358
	2	0.185137
	3	0.173224
	4+	0.154028

Observations:

Age:

1. 39.9% of users are from 26-35 age group.
2. 18% of users are grom 18-25 age group.
3. 19 % of users are from 36-45age group.

City Category:

1. 42 % of users are from Coty Cateogry B.

Gender:

1. 75% of users are Male.
2. 25% of users are Female.

Marital Status:

1. 59% users are Married.
2. 40% users are unmarried.

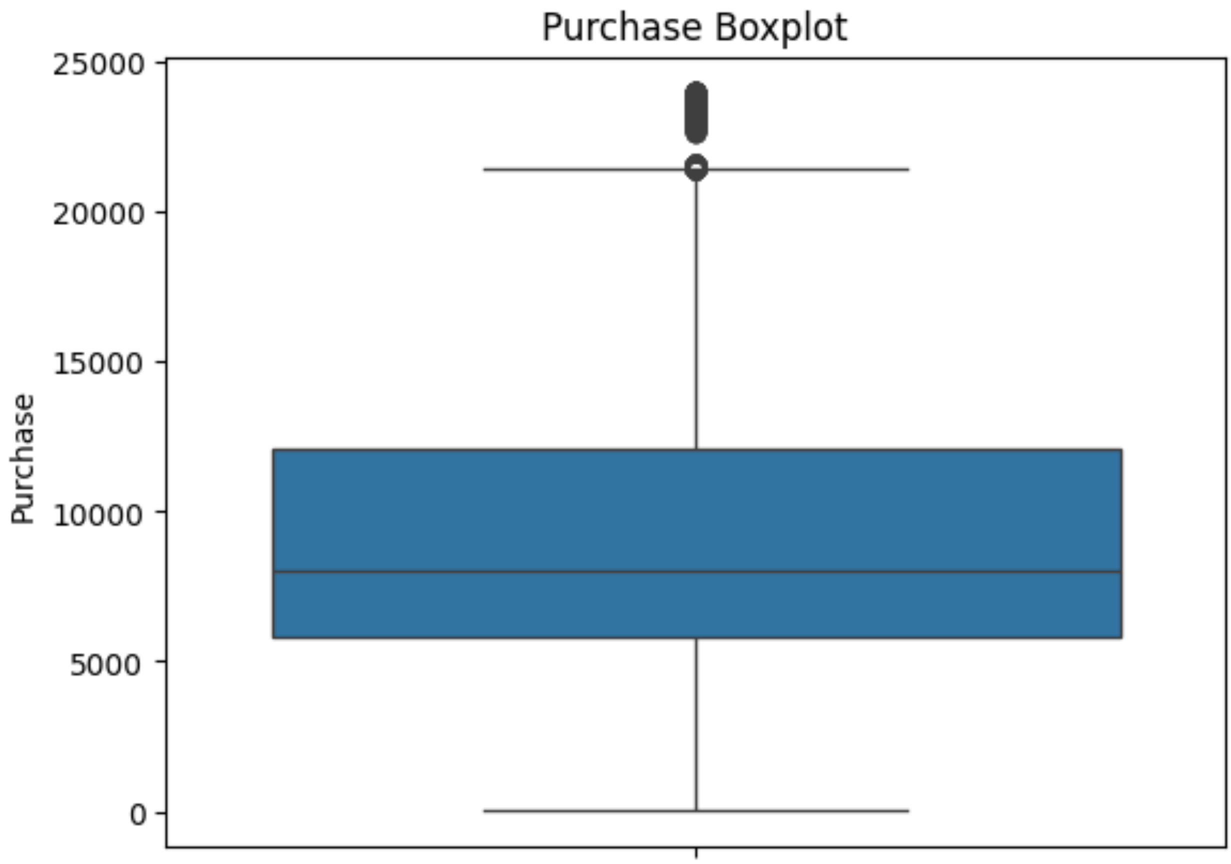
Stay in current city years:

1. Most of the users have lived for 1 year in city i.e; 35%.
2. Next, users living for 2 years i.e; 18 %.

Total Occuaption Categories: 21

FInding outliers

```
sns.boxplot(y=df['Purchase'])
plt.title('Purchase Boxplot')
plt.show()
```




```
#Total Outliers
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)]
print("Number of outliers:", len(outliers))
```

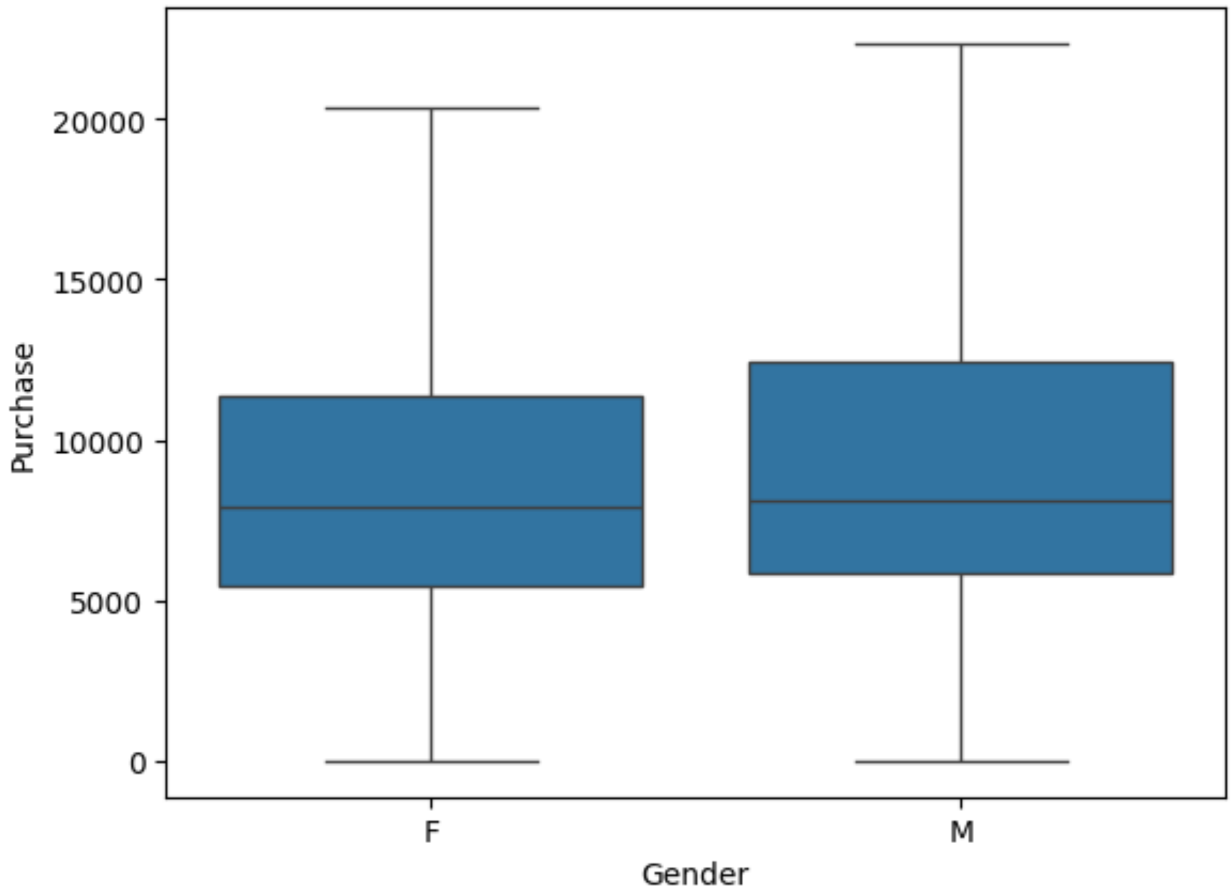
```
Number of outliers: 2677
```

```
import numpy as np

def winsorize_purchase(group):
    Q1 = group['Purchase'].quantile(0.25)
    Q3 = group['Purchase'].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    group['Purchase'] = np.where(group['Purchase'] < lower_bound, lower_bound,
                        np.where(group['Purchase'] > upper_bound, upper_bounc
    return group

df_capped = df.groupby('Gender', group_keys=False).apply(winsorize_purchase)
```

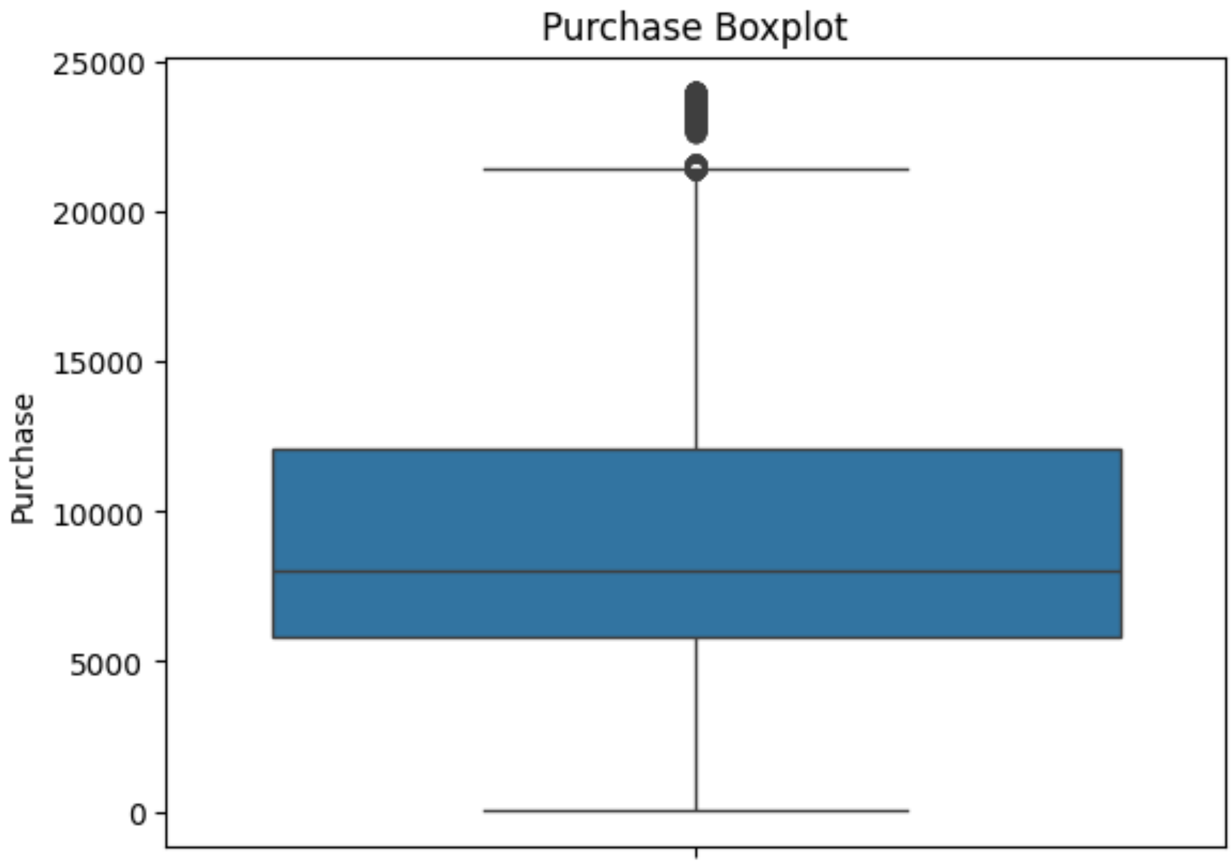
```
/tmp/ipython-input-2668692087.py:13: DeprecationWarning: DataFrameGroupBy.a
  df_capped = df.groupby('Gender', group_keys=False).apply(winsorize_purcha
```

```
import seaborn as sns
sns.boxplot(data=df_capped, x='Gender', y='Purchase')
```

```
<Axes: xlabel='Gender', ylabel='Purchase'>
```

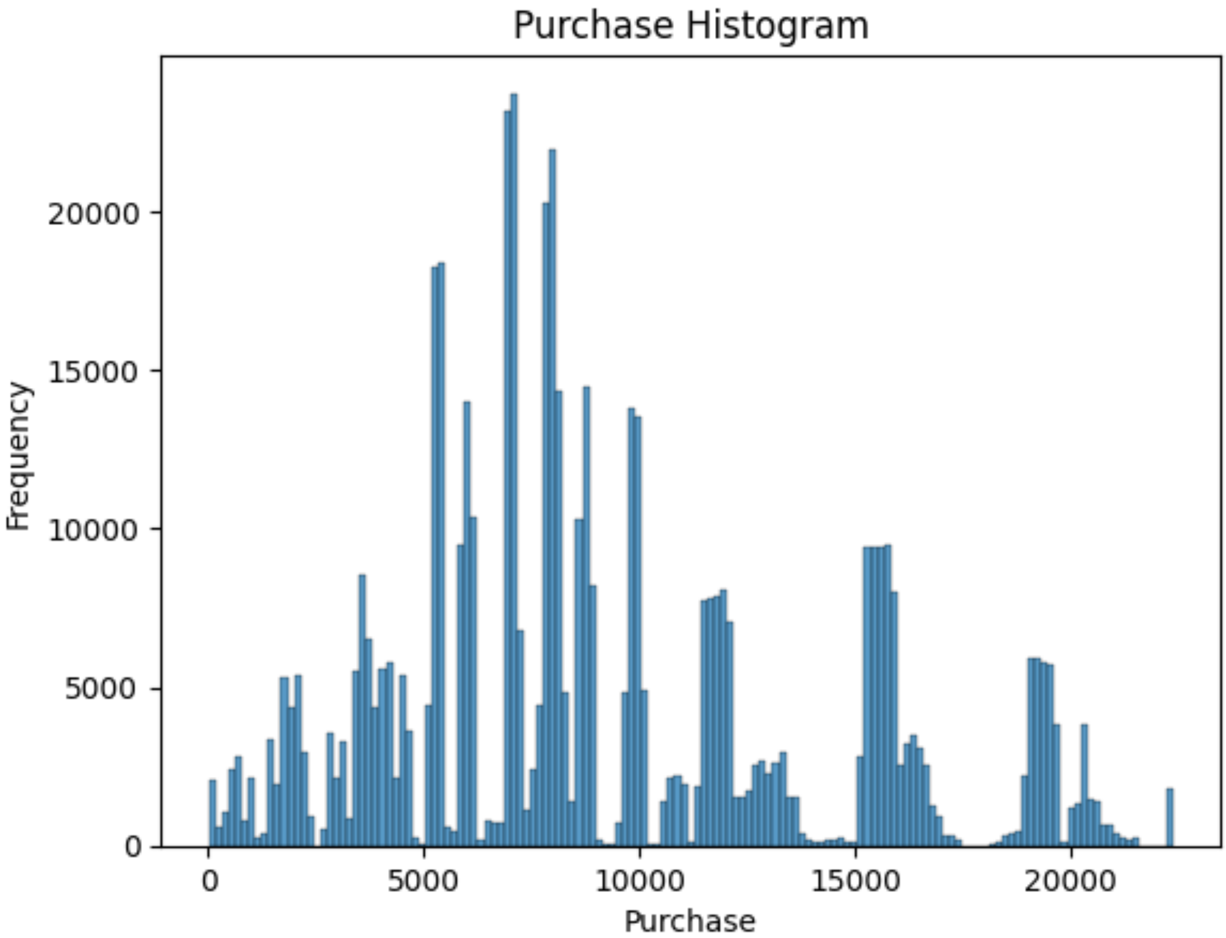


```
sns.boxplot(y=df['Purchase'])
plt.title('Purchase Boxplot')
plt.show()
```

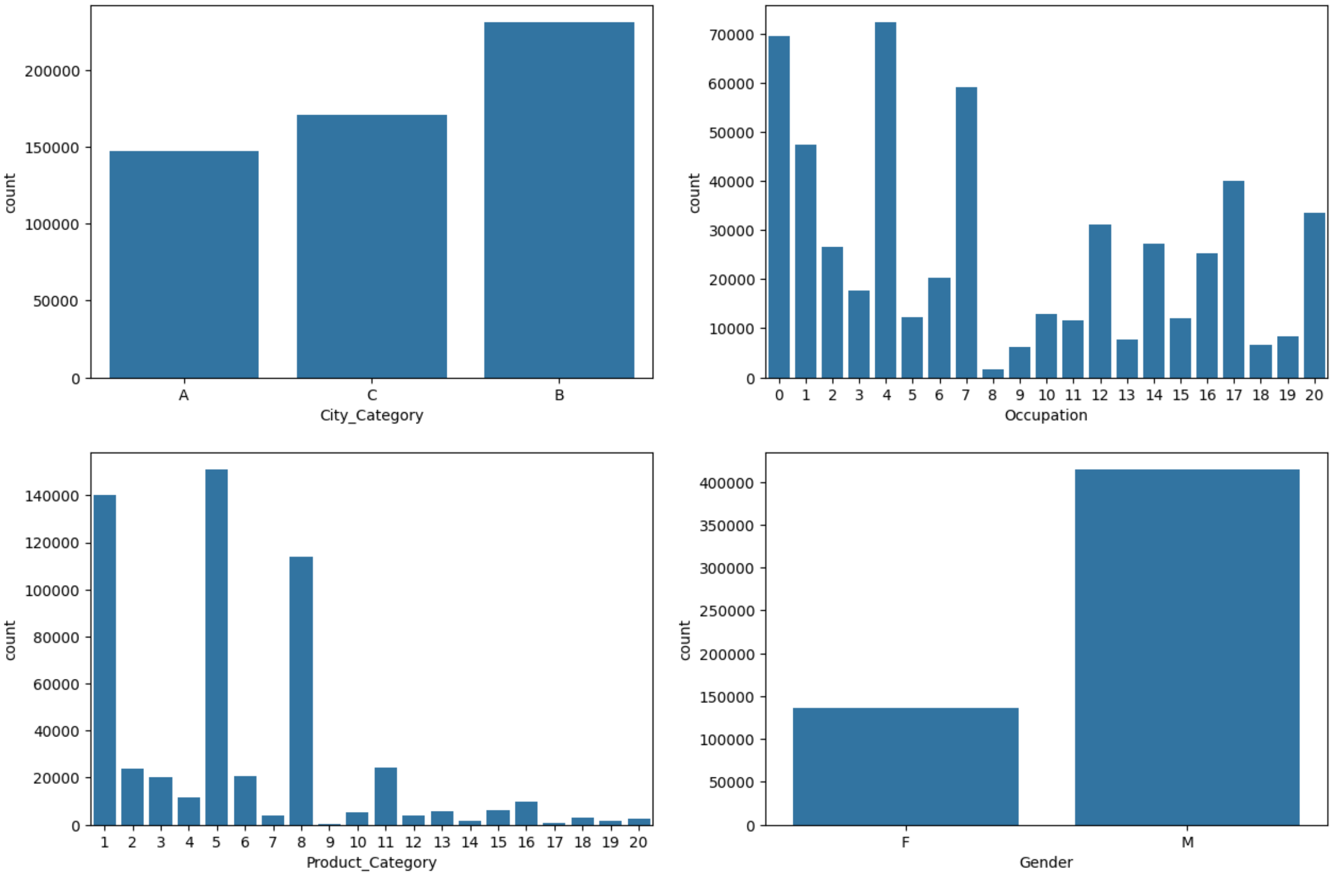


Univariate Analysis

```
sns.histplot(df_capped['Purchase'])
plt.title('Purchase Histogram')
plt.xlabel('Purchase')
plt.ylabel('Frequency')
plt.show()
```



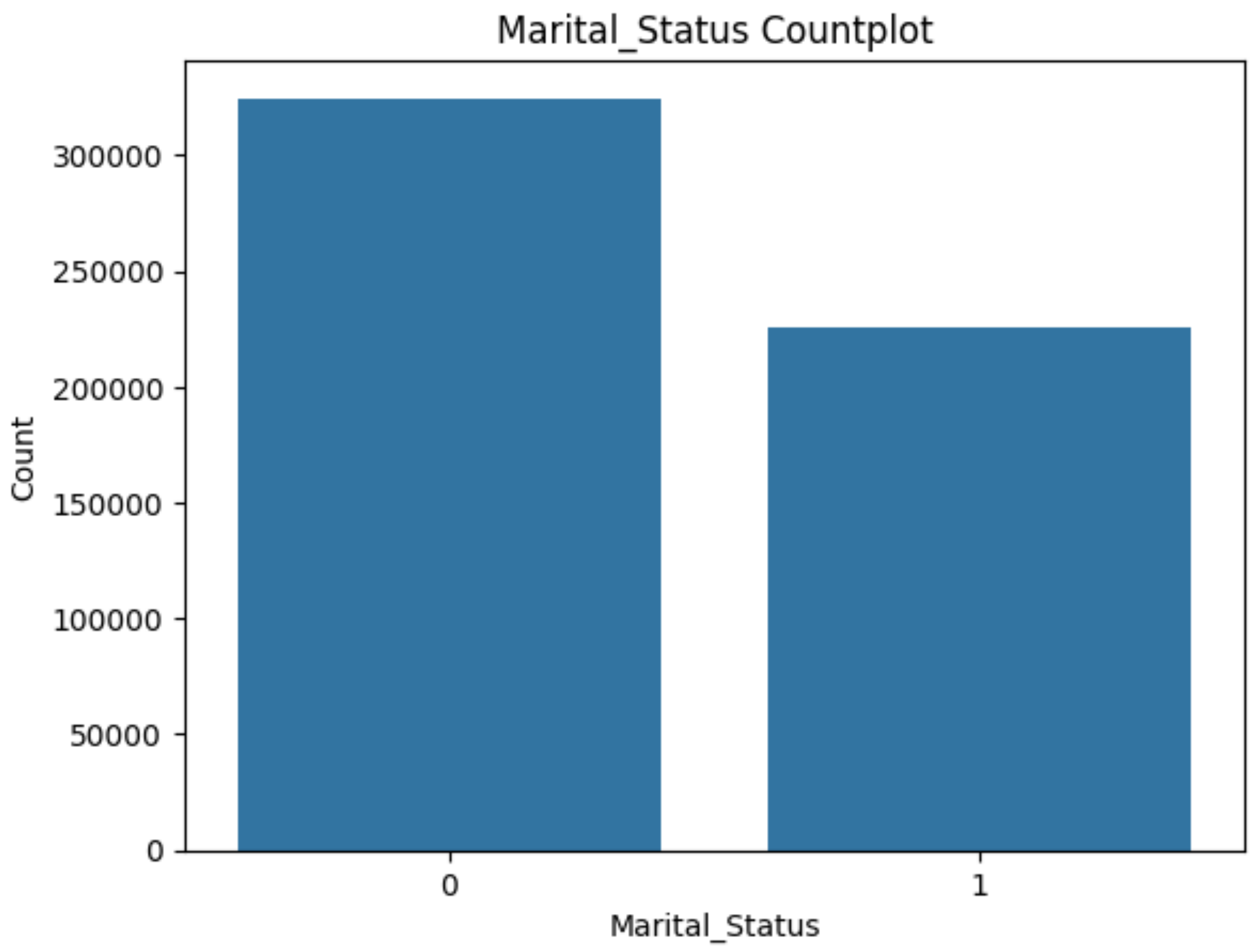

```
#Plotting categorical variables
fig, axes = plt.subplots(nrows =2, ncols =2, figsize=(15,10))
sns.countplot(x = df_capped['Occupation'], ax=axes[0,1])
sns.countplot(x = df_capped['City_Category'], ax = axes[0,0])
sns.countplot(x = df_capped['Product_Category'], ax= axes[1,0])
sns.countplot(x= df_capped['Gender'], ax=axes[1,1])
plt.show()
```


count
City_Category
A
C
B
Occupation
Product_Category
Gender
F
M

Observations:

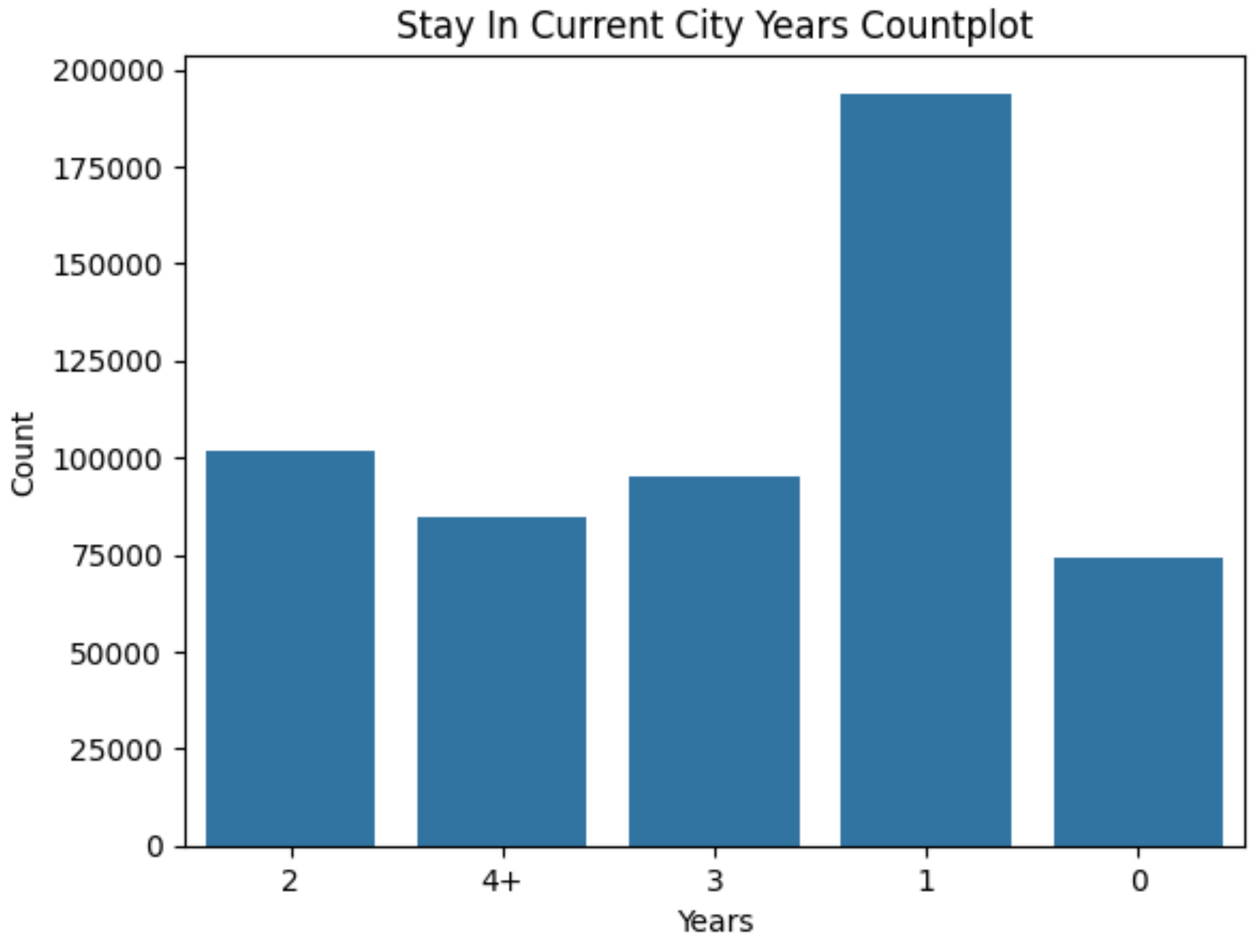
- City Category B customers are more.
- Occupation id 4 has the highest no of customers.
- Male users are more in number than Female.
- Product Category 1,5,8,11 has highest frequency of purchases.

```
sns.countplot(x=df_capped['Marital_Status'])
plt.title('Marital_Status Countplot')
plt.xlabel('Marital_Status')
plt.ylabel('Count')
plt.show()
```



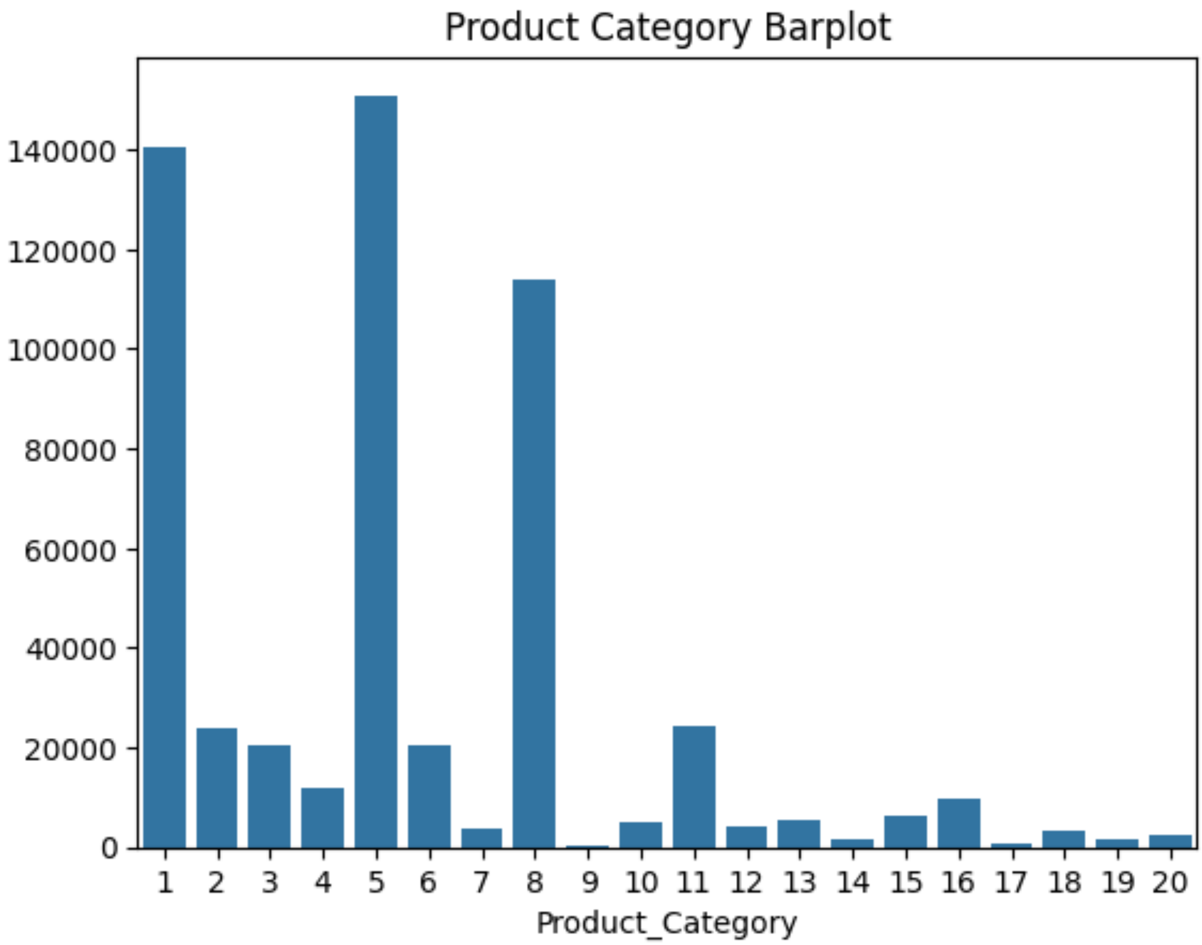
Unmarried customers are more than married customers.

```
sns.countplot(x=df_capped['Stay_In_Current_City_Years'])
plt.title('Stay In Current City Years Countplot')
plt.xlabel('Years')
plt.ylabel('Count')
plt.show()
```

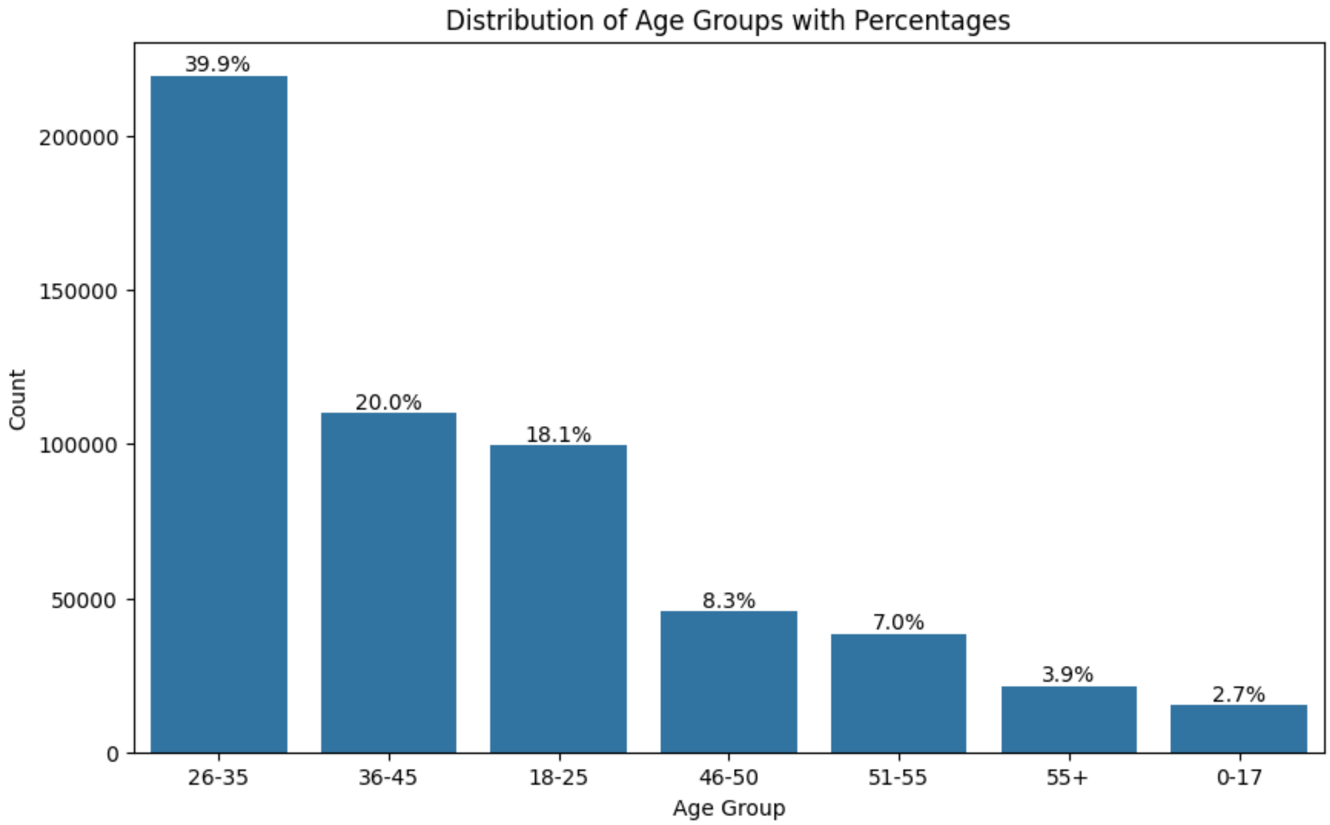


Most of the Users in system have stayed for 1 year in the current city.

```
sns.barplot(x= df_capped['Product_Category'].value_counts().index, y=df_capped|
plt.title('Product Category Barplot')
plt.show()
```




```
# Calculate value counts and their percentages
age_counts = df_capped['Age'].value_counts()
age_percentages = df_capped['Age'].value_counts(normalize=True).mul(100)

# Create a countplot
plt.figure(figsize=(10, 6))
sns.countplot(x='Age', data=df_capped, order=age_counts.index)
plt.title('Distribution of Age Groups with Percentages')
plt.xlabel('Age Group')
plt.ylabel('Count')

# Add percentage labels to the bars
for i, count in enumerate(age_counts.values):
    percentage = age_percentages.values[i]
    plt.text(i, count, f'{percentage:.1f}%', ha='center', va='bottom')

plt.show()
```

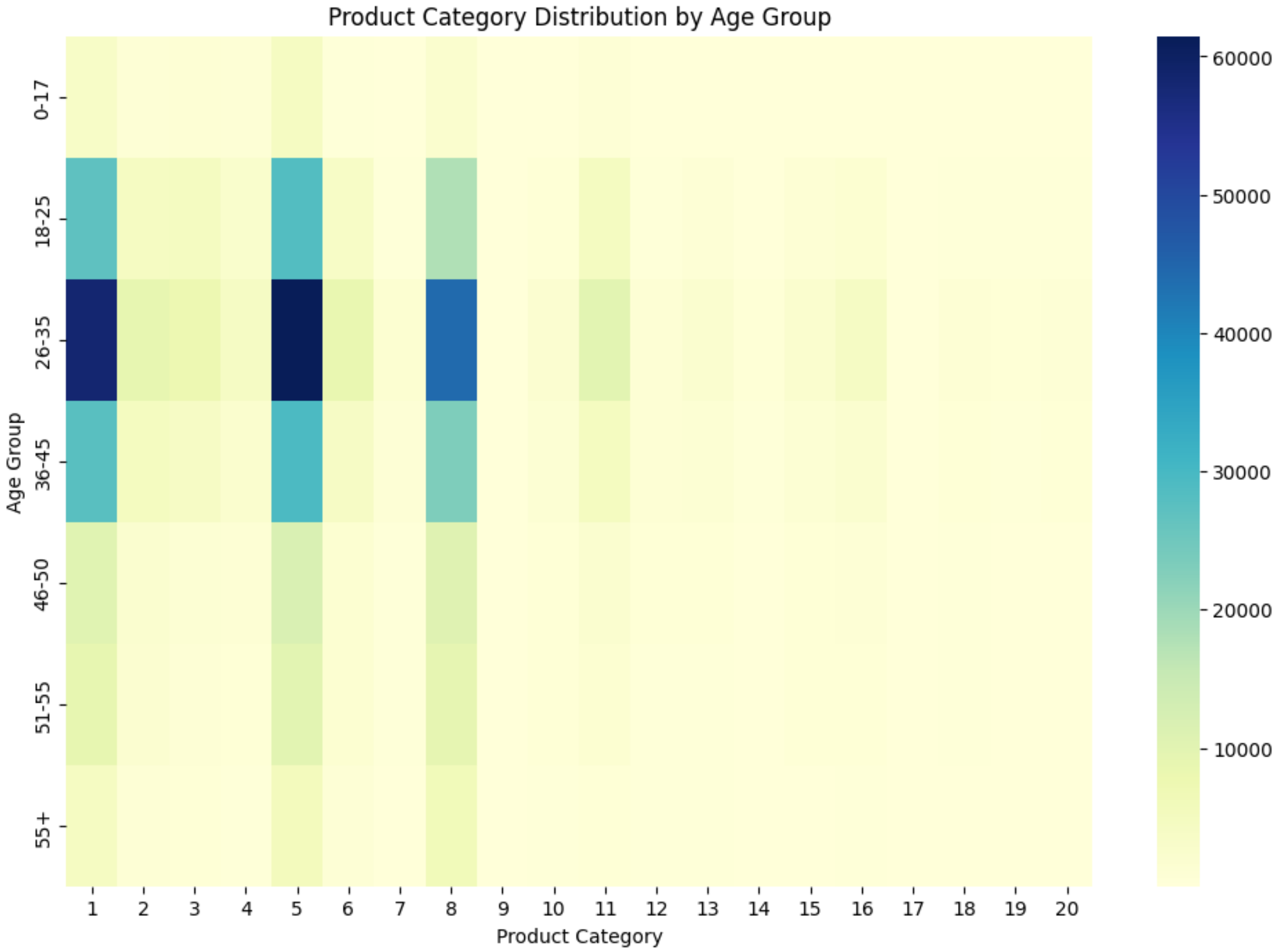


26 - 35 is the age group with most no of the users.

Bi-Variate Analysis

```
#What products are different age groups buying?
#df_capped.groupby('Age')['Product_Category'].value_counts().plot.hist()

# Create a pivot table with Age as index and Product_Category as columns
pivot = df_capped.pivot_table(index='Age', columns='Product_Category', aggfunc=

# Plot a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(pivot,  fmt="d", cmap="YlGnBu")
plt.title('Product Category Distribution by Age Group')
```

```
plt.xlabel('Product Category')
plt.ylabel('Age Group')
plt.show()
```

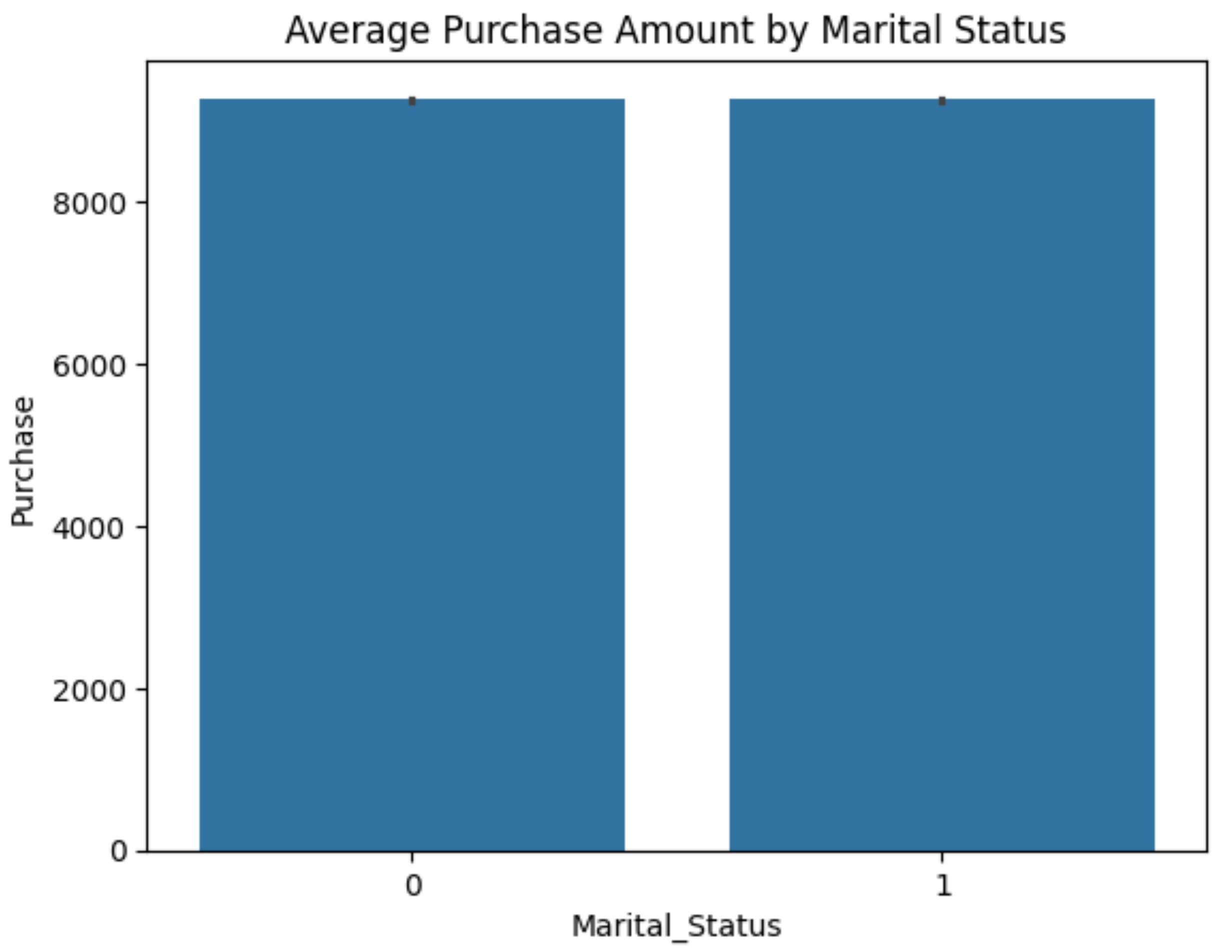


Highest purchaser is from age group 26 - 35 years, and they bought products from categories [1, 5, 8].

```
df_capped.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Curre
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	

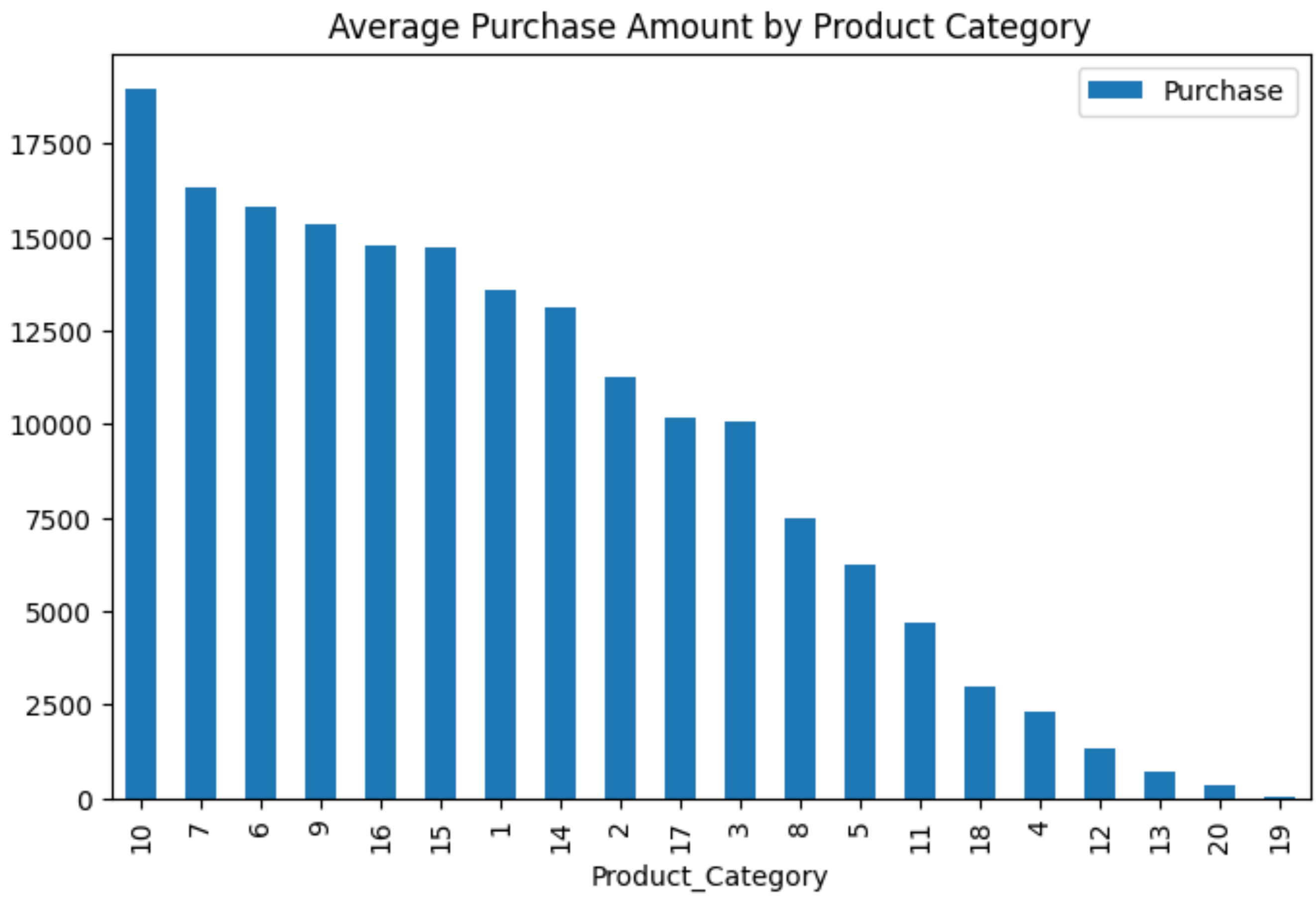
```
#Check if marital status has any impact on purchase behavior

sns.barplot(x='Marital_Status', y='Purchase', data=df_capped, estimator='mean')
plt.title('Average Purchase Amount by Marital Status')
plt.show()
```



```
#Objective: Find if purchase behavior by category varies by gender.
sns.barplot(x='Gender', y='Purchase', data=df_capped, estimator='mean')
plt.title('Average Purchase Amount by Gender')
plt.show()
```

Male have spent more than Female in purchasing.

```
pivot = df_capped.pivot_table(values='Purchase', index='Product_Category', agg1
pivot.sort_values('Purchase', ascending=False).plot(kind='bar', figsize=(8,5))
plt.title('Average Purchase Amount by Product Category')
plt.show()
```

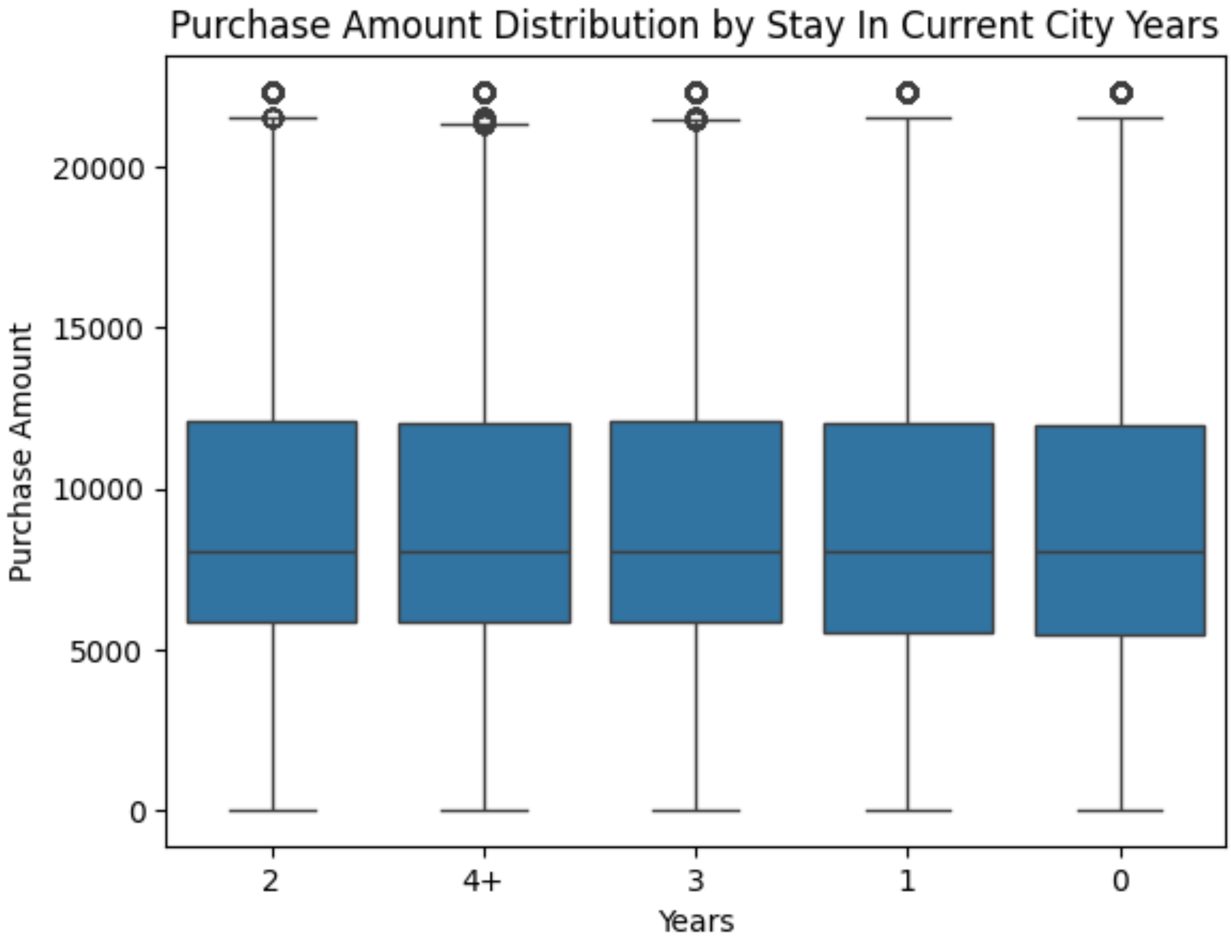


Product Category '10' has highest Avg Purchase Amount, most people bought category 10 items.

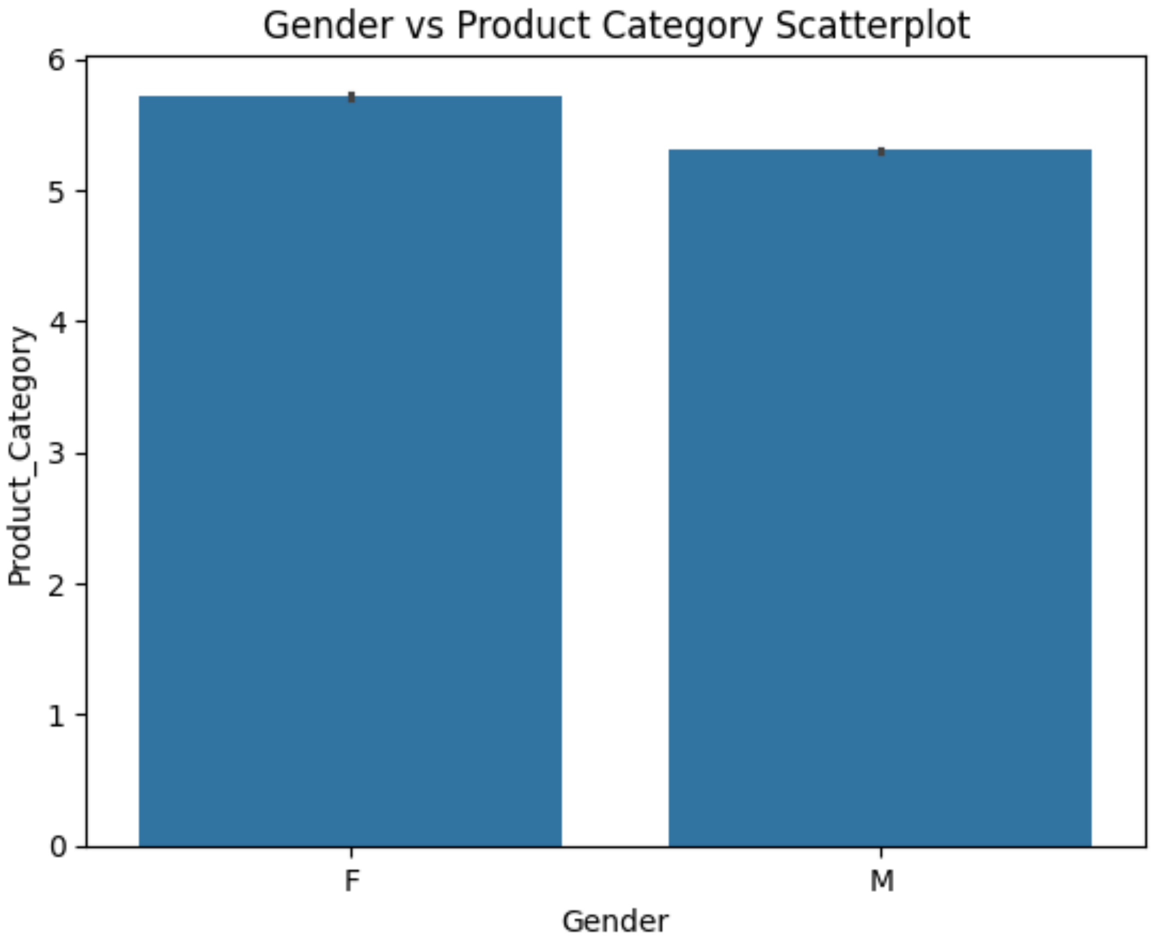
Recommendations:

- May be increase discounts to lure customers to buy the products of categories[19,20,13,12,4,18]
- Product Categories- [7,3] engage the customers of these categories, so as not to lose them.

```
sns.boxplot(x='Stay_In_Current_City_Years', y='Purchase', data=df_capped)
plt.title('Purchase Amount Distribution by Stay In Current City Years')
plt.xlabel('Years')
plt.ylabel('Purchase Amount')
plt.show()
```



```
#Gender vs Product Category

sns.barplot(x=df_capped['Gender'], y=df_capped['Product_Category'], data=df_cap
plt.title('Gender vs Product Category Scatterplot')
plt.show()
```



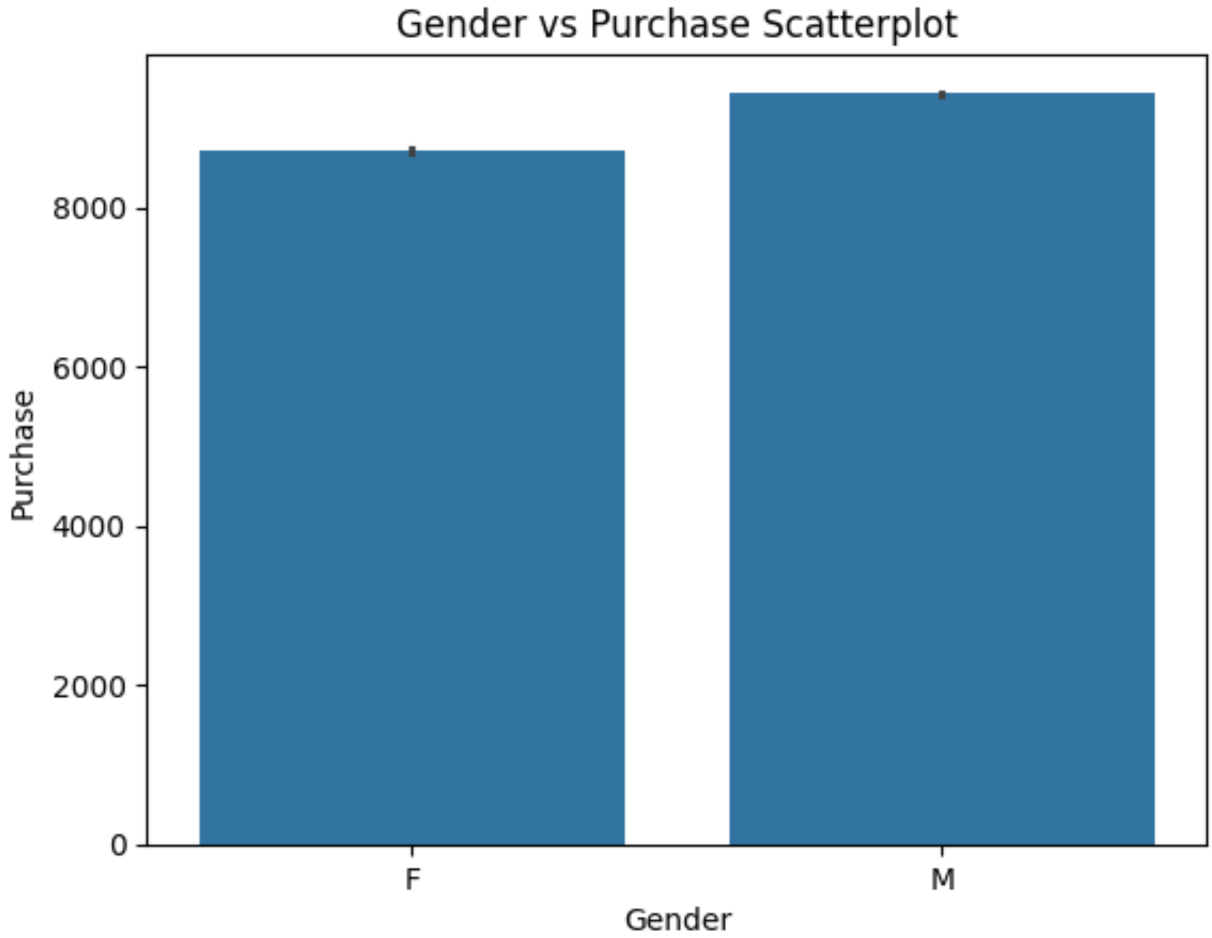

```
mean_purchase_gender = df_capped.groupby('Gender')['Purchase'].mean()
mean_purchase_gender
```

Gender	Purchase
F	8718.127823
M	9432.546011

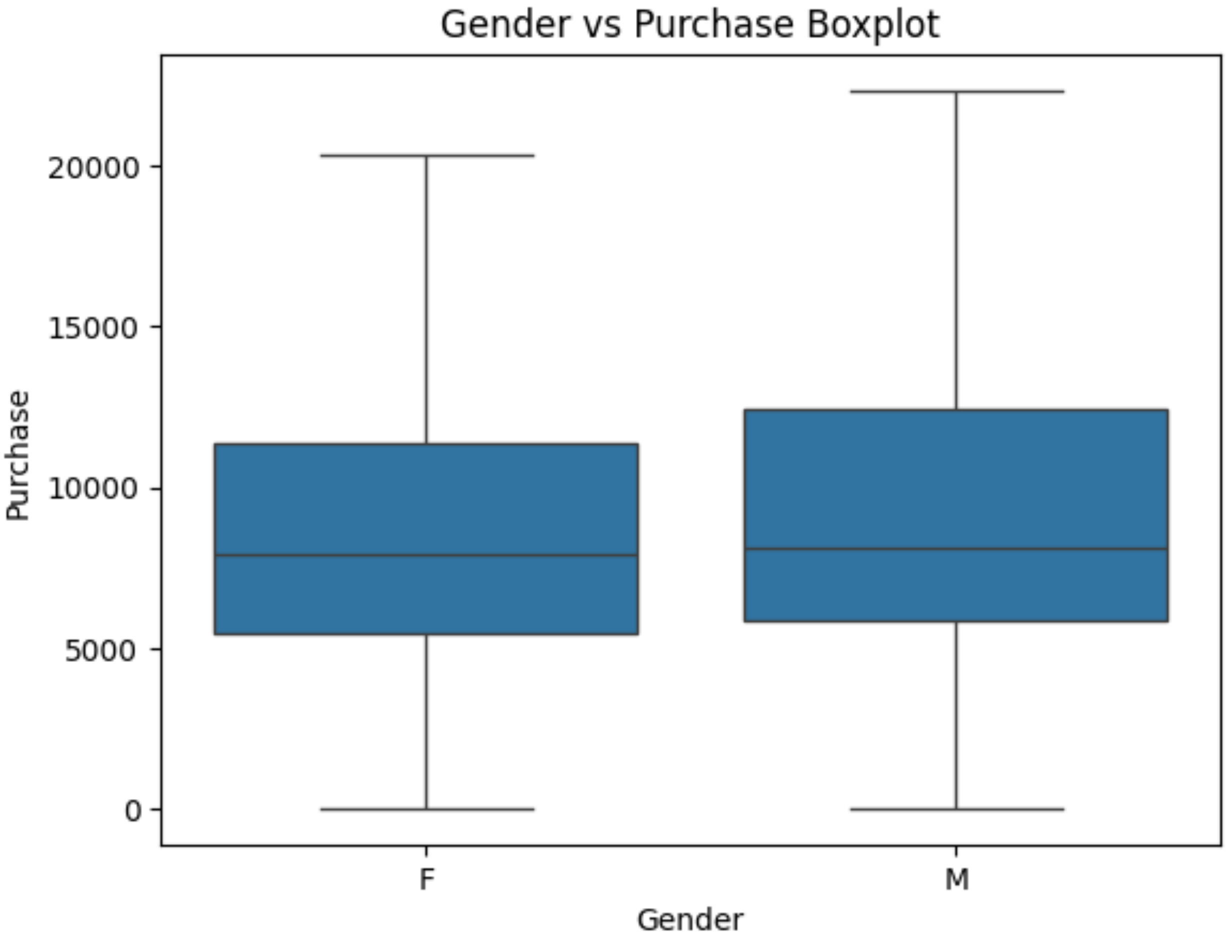
dtype: float64

On an average, Male have purchased more than Female.

```
sns.barplot(x=df_capped['Gender'], y=df_capped['Purchase'], data=df_capped)
plt.title('Gender vs Purchase Scatterplot')
plt.show()
```



```
sns.boxplot(x= df_capped['Gender'], y = df_capped['Purchase'])
plt.title('Gender vs Purchase Boxplot')
plt.show()
```



```
# Group by 'Age' and find the index of the maximum 'Purchase' in each group
idx_max_purchase = df_capped.groupby('Age')['Purchase'].idxmax()

# Select the rows with the maximum purchase value for each age group
rows_max_purchase = df_capped.loc[idx_max_purchase]

print("Rows with the maximum purchase value for each age group:")
print('' '')

# Display the result
display(rows_max_purchase)
```

```
Rows with the maximum purchase value for each age group:
```

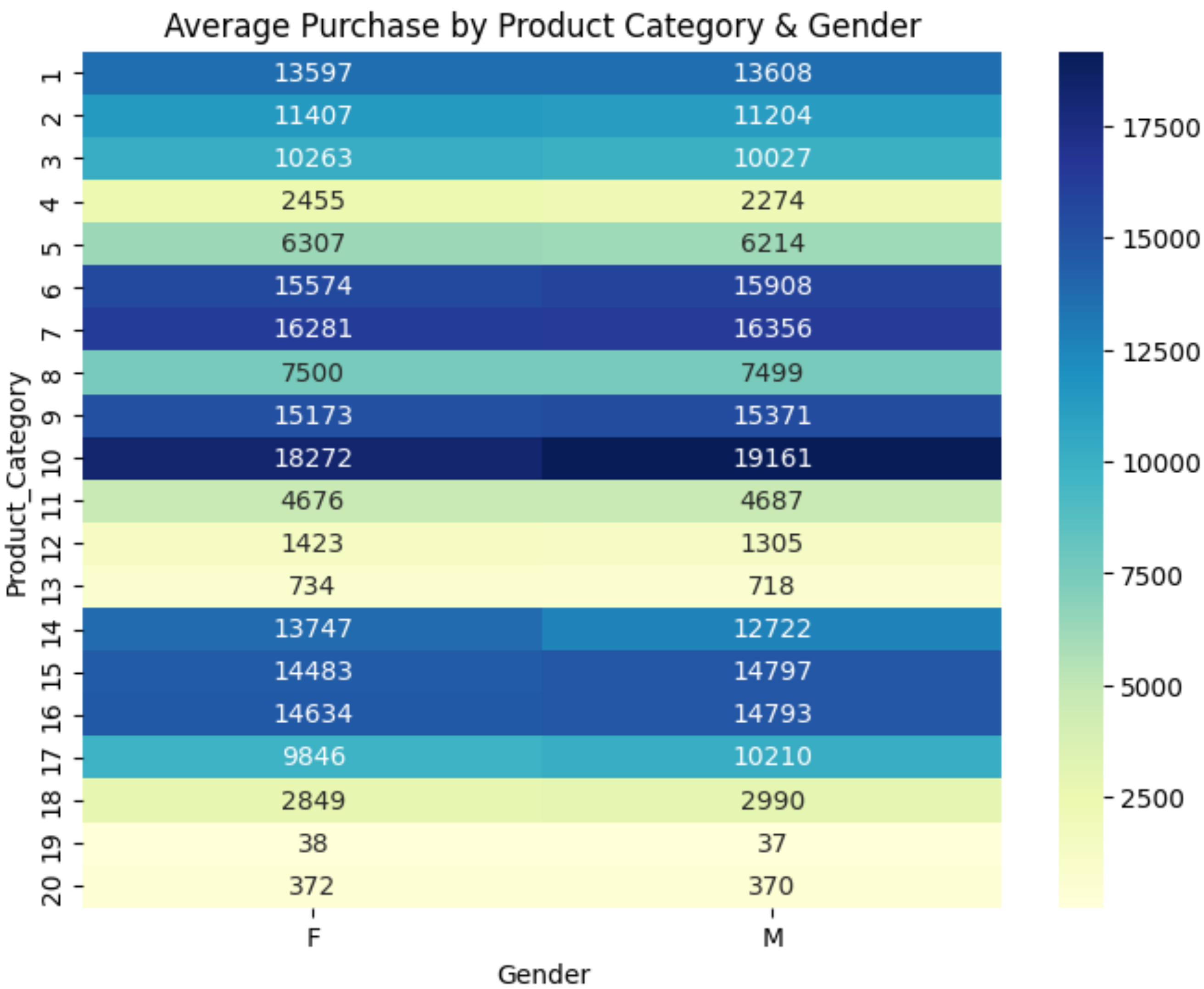
	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Cu
5059	1000829	P00085342	M	0-17	19	A	
652	1000126	P00087042	M	18-25	9	B	
343	1000058	P00117642	M	26-35	2	B	
3908	1000645	P00116142	M	36-45	20	A	
5493	1000889	P00117642	M	46-50	20	A	
3166	[illegible]	[illegible]	M	51-55	7	A	
7542	1001178	P00116142	M	55+	0	C	

3 Maximum purchases are made by age group's 18-25, 26-35, 51-55 and all are from City Category: A

```
pivot = df_capped.pivot_table(values= 'Purchase', index= 'Product_Category', cc
pivot
```

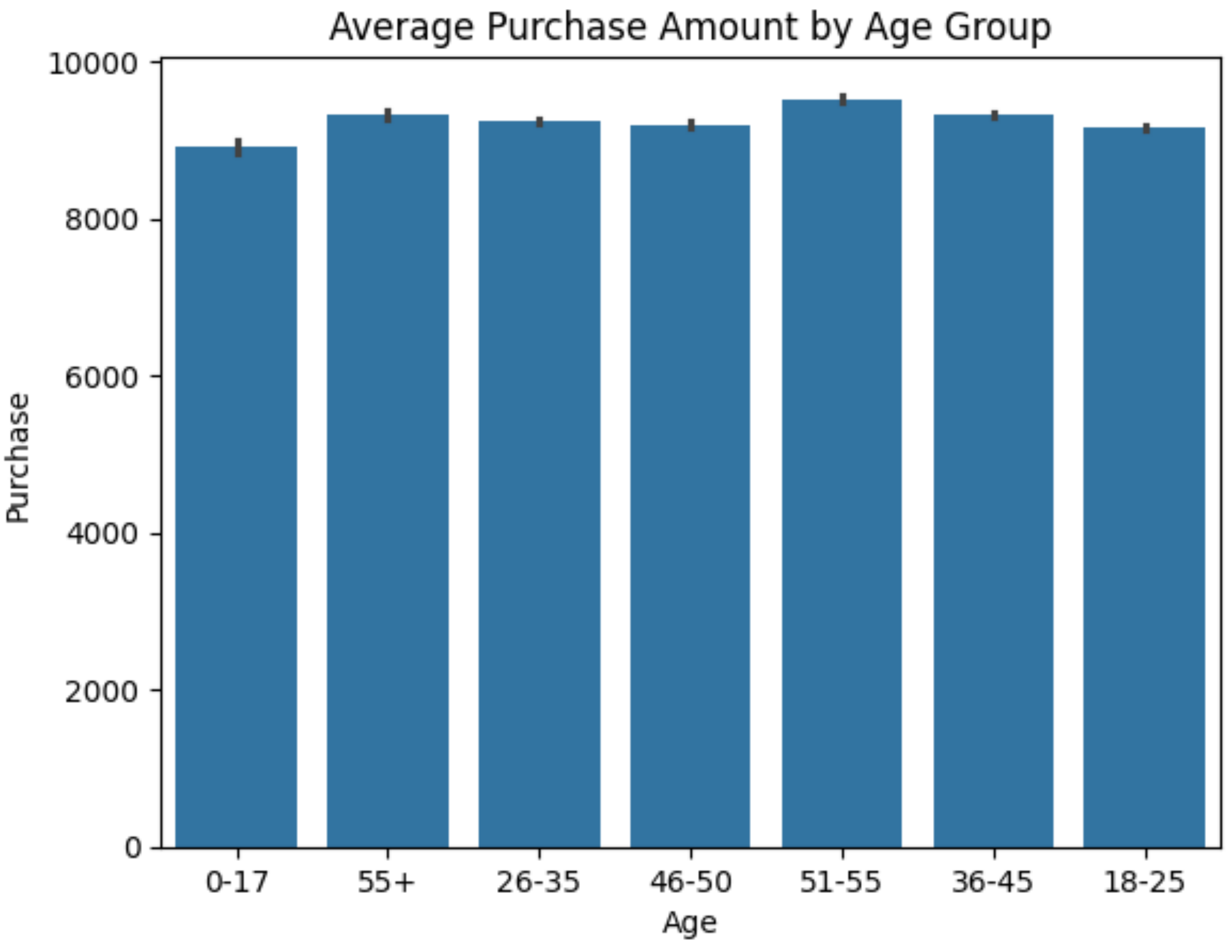
Gender	F	M
Product_Category		
1	13597.162619	13608.164721
2	11407.496819	11203.590520
3	10262.656677	10026.550081
4	2454.851882	2273.512694
5	6307.239532	6214.230729
6	15574.286576	15907.851009
7	16281.435313	16355.789777
8	7499.924787	7498.554419
9	15172.614286	15370.951471
10	18271.508606	19161.136639
11	4676.371808	4687.425261
12	1422.909269	1305.154037
13	733.846785	718.306092
14	13747.362761	12722.321111
15	14483.134799	14797.431350
16	14634.342423	14793.384056
17	9846.403226	10209.732558
18	2848.607330	2990.168793
19	37.676275	36.793403
20	371.564315	370.052545

```
# Heatmap
plt.figure(figsize=(8,6))
sns.heatmap(pivot, annot=True, fmt=".0f", cmap="YlGnBu")
plt.title('Average Purchase by Product Category & Gender')
plt.show()
```



Male has purchased worth 19161 items in product category.

```
sns.barplot(x='Age', y='Purchase', data=df_capped, estimator='mean')
plt.title('Average Purchase Amount by Age Group')
plt.show()

df.groupby('Age')['Purchase'].mean()
```



Age	Purchase
0-17	8933.464640
18-25	9169.663606
26-35	9252.690633
36-45	9331.350695
46-50	9208.625697
51-55	9534.808031
55+	9336.280459

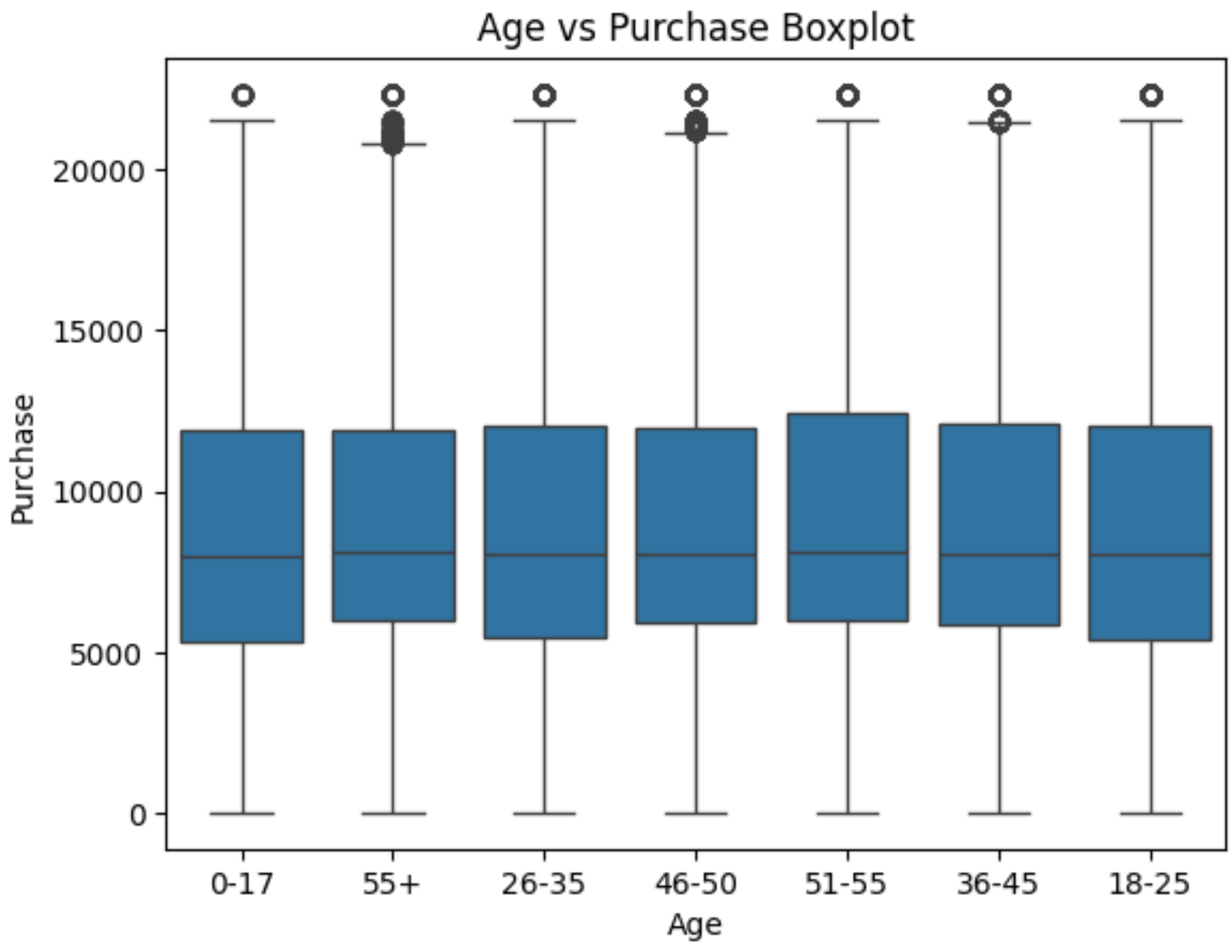
dtype: float64

Customers in the 26–35 age group have the highest average purchase.

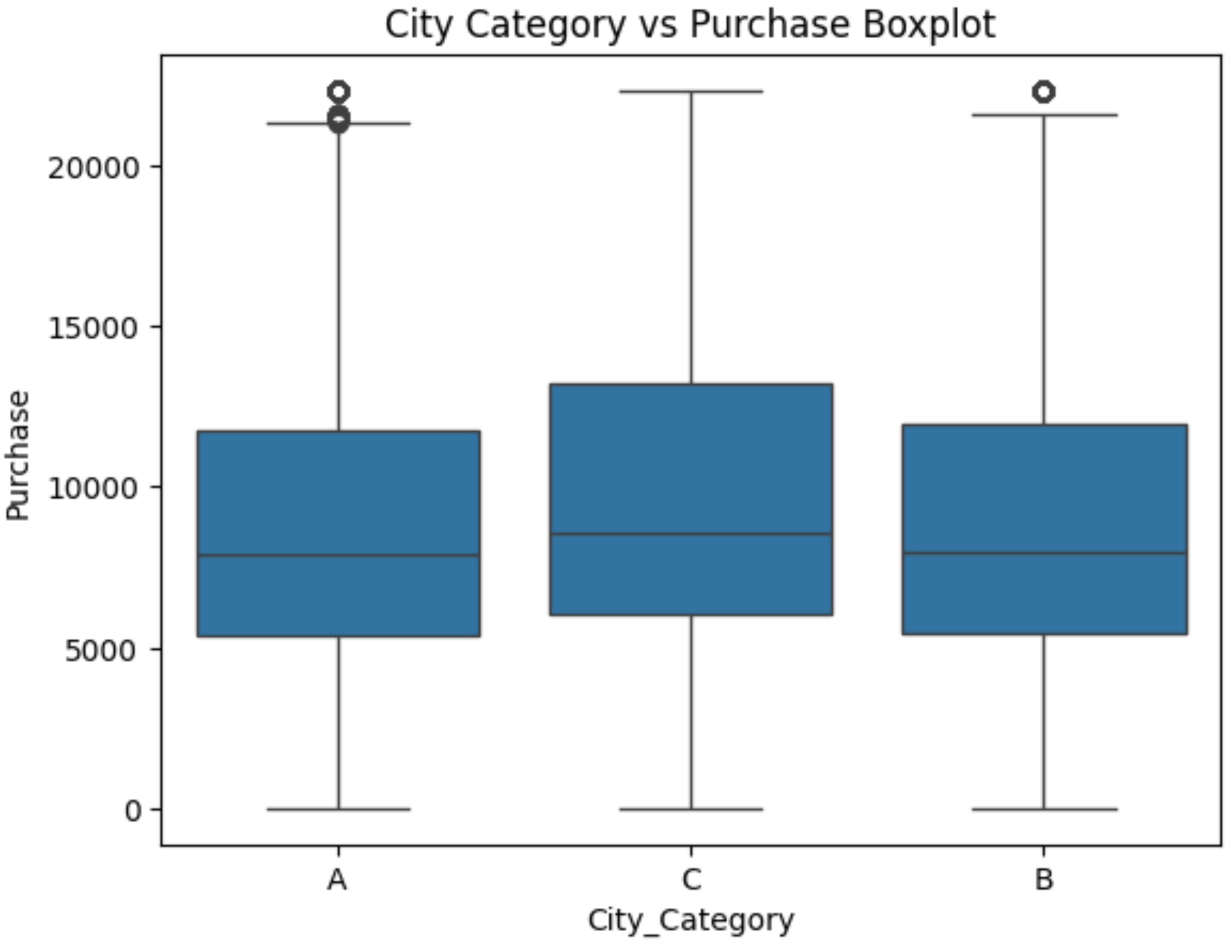
The 51+ segment spends the least, possibly due to different shopping preferences.

Targeted promotions for the 26–35 group may yield higher sales.

```
sns.boxplot(x=df_capped['Age'], y=df_capped['Purchase'])
plt.title('Age vs Purchase Boxplot')
plt.show()
```



```
sns.boxplot(x=df_capped['City_Category'], y=df_capped['Purchase'])
plt.title('City Category vs Purchase Boxplot')
plt.show()
```




```
#Lets create separate dataframes for females and males

df_male = df_capped[df_capped['Gender']=='M']
df_female = df_capped[df_capped['Gender']=='F']
iterations = 1000
sample_size = 100
```

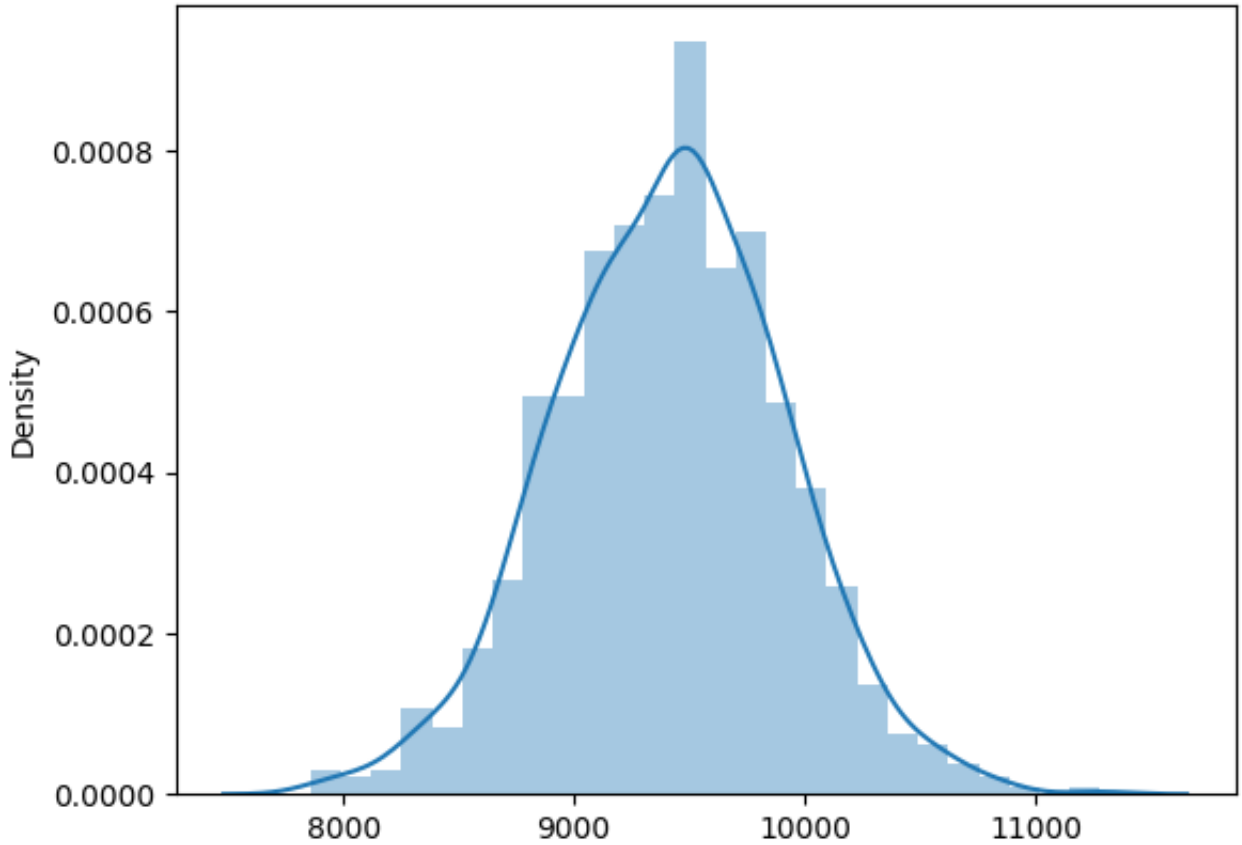
```
#Create samples of a certain sample size

male_sample_means = [df_male['Purchase'].sample(sample_size, replace= True).mea
female_sample_means = [df_female['Purchase'].sample(sample_size, replace= True)


#Male Population Mean
Male_Population_Mean = df_male['Purchase'].mean()
print('Male_Population_Mean:', Male_Population_Mean)

#Female Population Mean
Female_Population_Mean = df_female['Purchase'].mean()
print('Female_Population_Mean:', Female_Population_Mean)
```

```
Male_Population_Mean: 9432.54601107037
Female_Population_Mean: 8718.127822898336
```

```
male_sample_means = np.array(male_sample_means)
female_sample_means = np.array(female_sample_means)
```

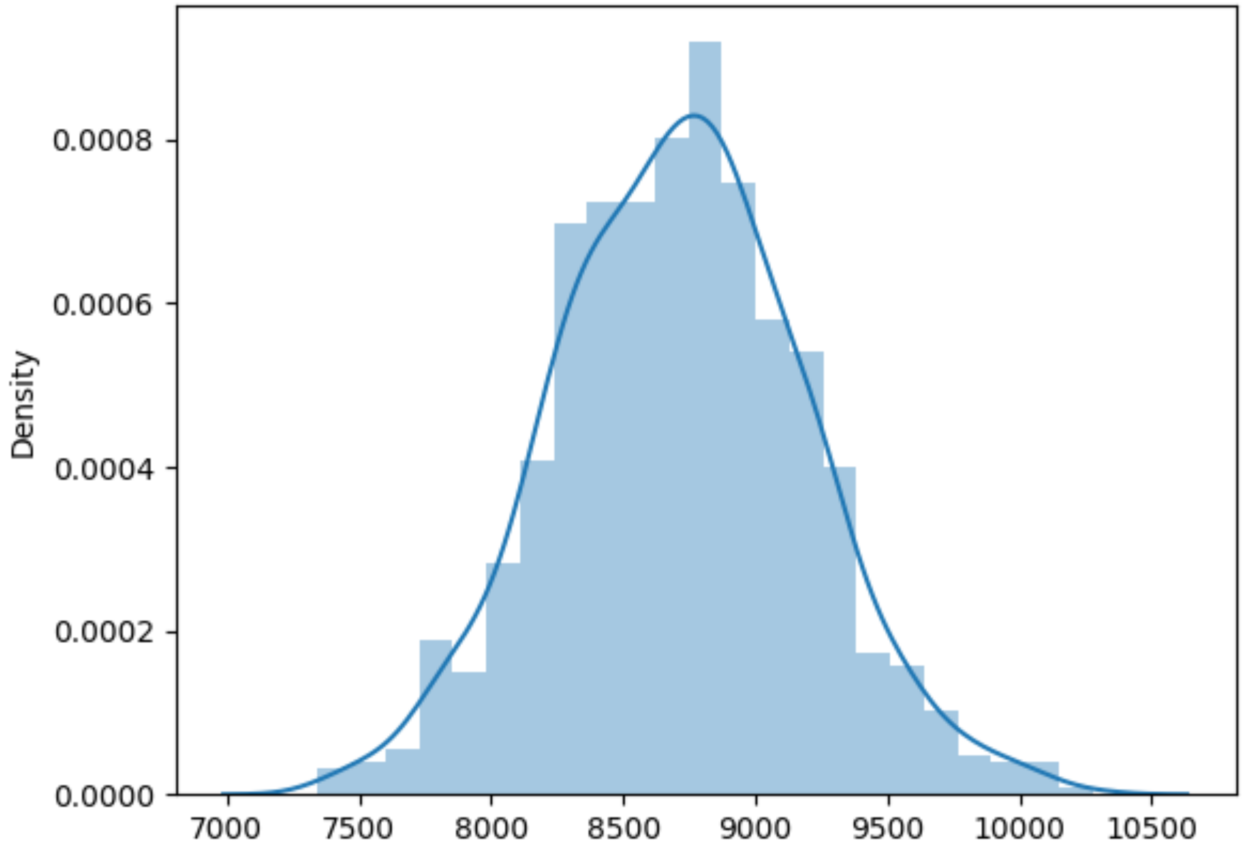
```
sns.distplot(male_sample_means, kde= True)
```

```
/tmp/ipython-input-461831333.py:1: 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 wit
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(male_sample_means, kde= True)
<Axes: ylabel='Density'>
```



```
sns.distplot(female_sample_means, kde= True)
```

```
/tmp/ipython-input-3495695361.py:1: 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 wit
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(female_sample_means, kde= True)
<Axes: ylabel='Density'>
```



```
#Lets build confidence interval of suppose 95% for Male sample and Female sampl
from scipy.stats import norm
z = norm.ppf(0.975)

male_upper_limit = male_sample_means.mean() + z * male_sample_means.std()
male_lower_limit = male_sample_means.mean() - z * male_sample_means.std()
print('male_lower_limit:', male_lower_limit)
print('male_upper_limit:', male_upper_limit)
print(' ')
print(f'We are 95% confident that the male sample mean will be between {male_lc
```

```
male_lower_limit: 8437.558661644105
male_upper_limit: 10390.945708355897

We are 95% confident that the male sample mean will be between 8437.5586616
```

```
female_upper_limit = female_sample_means.mean() + z * female_sample_means.std()
female_lower_limit = female_sample_means.mean() - z * female_sample_means.std()
print('female_lower_limit:', female_lower_limit)
print('female_upper_limit:', female_upper_limit)
print(' ')
print(f'We are 95% confident that the female sample mean will be between {femal
```

```
female_lower_limit: 7783.634629568146
female_upper_limit: 9646.535580431855

We are 95% confident that the female sample mean will be between 7783.63462
```

Can we conclude Males spend more than females?

No, because the range of mean values for male and female are overlapping.

Female range: 7797 - 9672

Male range: 8453 - 10390

To resolve this implication:

- Increase sample_size and check for mean.
- Decrease confidence percentage

```
#Try - decrease confidence percentage to 90 %
iterations1 = 1000
sample_size1 = 100
z= norm.ppf(0.90)

male_sample_means1 = [df_male['Purchase'].sample(sample_size, replace= True).me
female_sample_means1 = [df_female['Purchase'].sample(sample_size, replace= True

male_sample_means1 = np.array(male_sample_means1)
female_sample_means1 = np.array(female_sample_means1)


male_upper_limit1 = male_sample_means1.mean() + z * male_sample_means1.std()
male_lower_limit1 = male_sample_means1.mean() - z * male_sample_means1.std()
print('male_lower_limit1:', male_lower_limit1)
print('male_upper_limit1:', male_upper_limit1)
print(' ')
```

```
male_lower_limit1: 8814.565847478529
male_upper_limit1: 10071.883032521471
```

```
female_upper_limit1 = male_sample_means1.mean() + z * female_sample_means1.std(
female_lower_limit1 = male_sample_means1.mean() - z * female_sample_means1.std(
print('female_lower_limit1:', female_lower_limit1)
print('female_upper_limit1:', female_upper_limit1)
print(' ')
```

```
female_lower_limit1: 8837.713810747766
female_upper_limit1: 10048.735069252234
```

There is still some overlapping, we are cannot conclude about confidence.

Lets try increasing sample size with 90% confidence.

```
#Try - decrease confidence percentage to 90 %
iterations1 = 1000
sample_size1 = 600
z= norm.ppf(0.975)

male_sample_means2 = [df_male['Purchase'].sample(sample_size, replace= True).me
female_sample_means2 = [df_female['Purchase'].sample(sample_size, replace= True

male_sample_means2 = np.array(male_sample_means2)
female_sample_means2 = np.array(female_sample_means2)


male_upper_limit2 = male_sample_means2.mean() + z * male_sample_means2.std()
male_lower_limit2 = male_sample_means2.mean() - z * male_sample_means2.std()
print('male_lower_limit2:', male_lower_limit2)
print('male_upper_limit2:', male_upper_limit2)
print(' ')
```

```
male_lower_limit2: 8438.430632514555
male_upper_limit2: 10455.131337485444
```

```
female_upper_limit2 = male_sample_means2.mean() + z * female_sample_means2.std(
female_lower_limit2 = male_sample_means2.mean() - z * female_sample_means2.std(
print('female_lower_limit2:', female_lower_limit2)
print('female_upper_limit2:', female_upper_limit2)
print(' ')
```

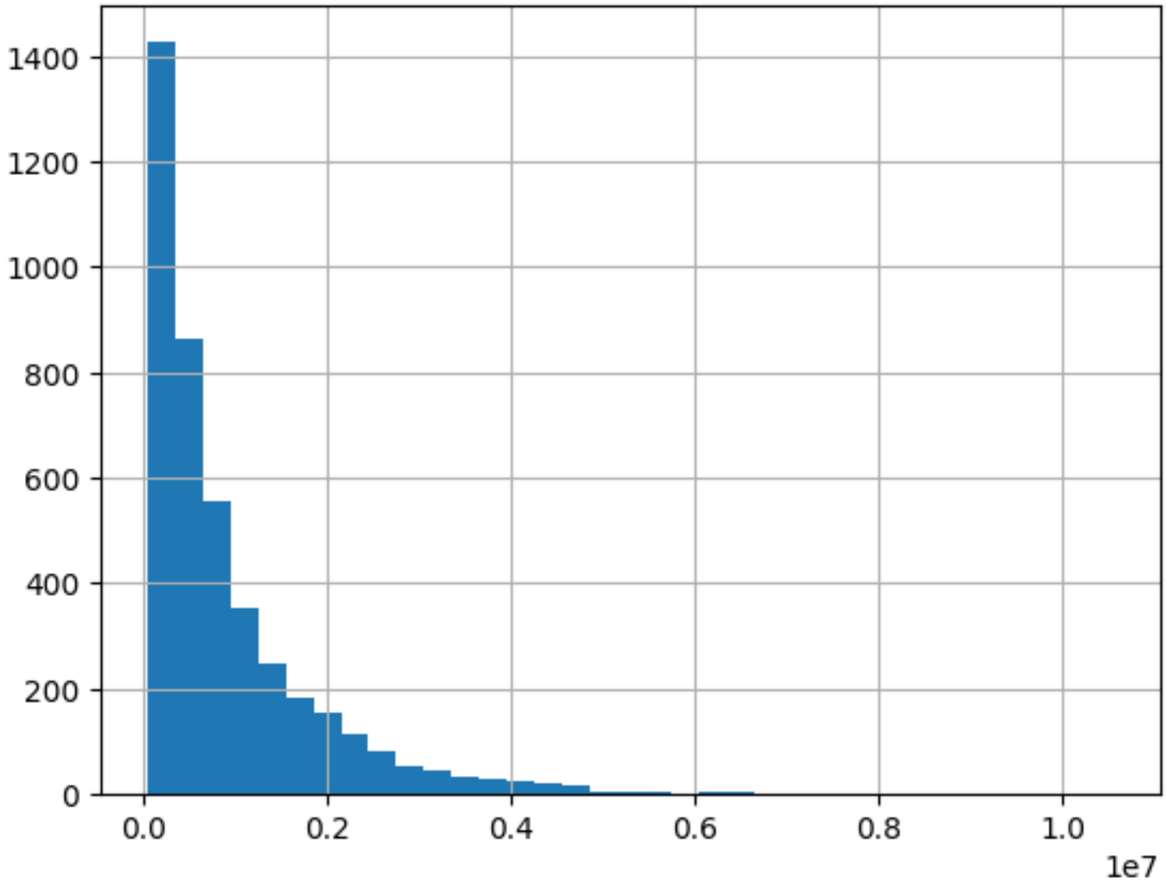
```
female_lower_limit2: 8481.12199332922
female_upper_limit2: 10412.43997667078
```

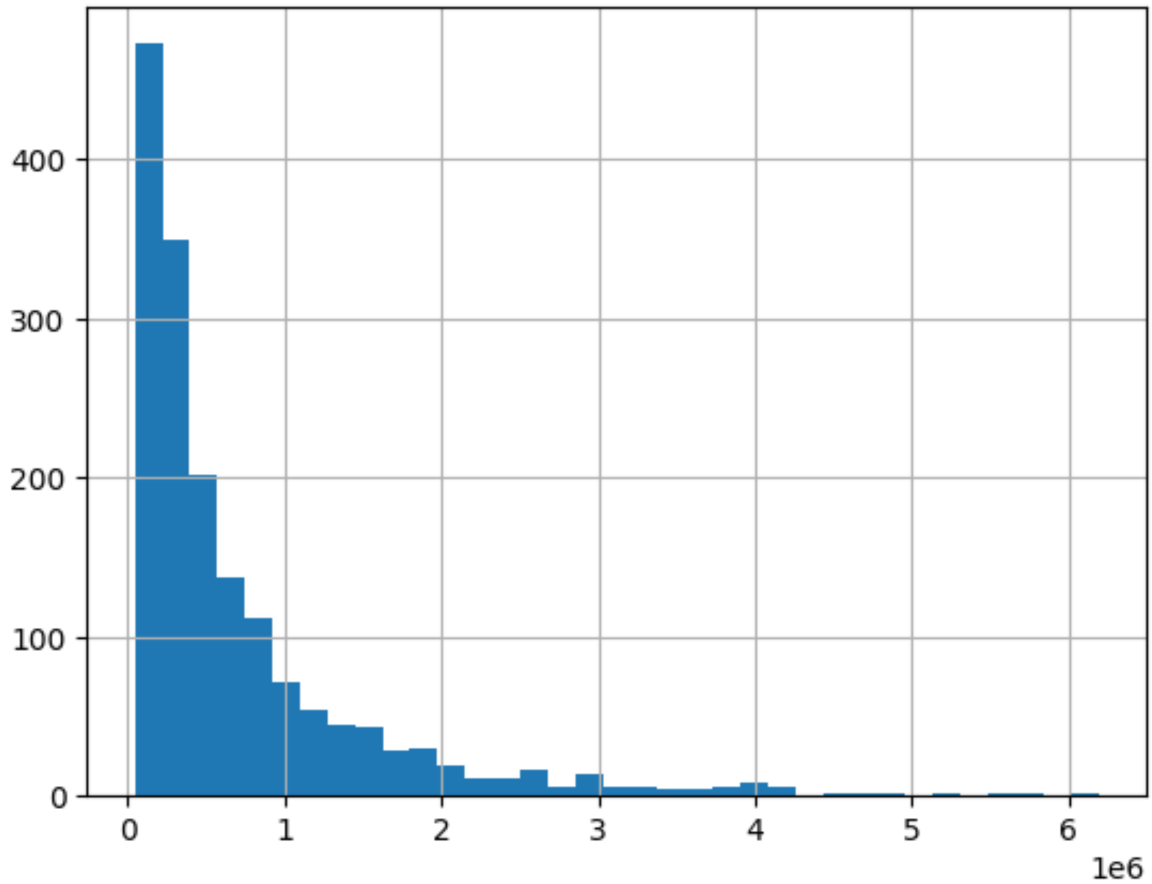
1. Are women spending more money per transaction than men? Why or Why not?

```
#Average amount spends per customer for Male and Female

amt_df = df.groupby(['User_ID', 'Gender'])[['Purchase']].sum()
amt_df = amt_df.reset_index()
amt_df.head()
```

	User_ID	Gender	Purchase
0	1000001	F	334093
1	1000002	M	810472
2	1000003	M	341635
3	1000004	M	206468
4	1000005	M	821001

```
# histogram of average amount spend for each customer - Male & Female
amt_df[amt_df['Gender']=='M']['Purchase'].hist(bins=35)
plt.show()
amt_df[amt_df['Gender']=='F']['Purchase'].hist(bins=35)
plt.show()
```


1400
1200
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0
1e7


400
300
200
100
0
0
1
2
3
4
5
6
1e6

```
male_avg = amt_df[amt_df['Gender']=='M']['Purchase'].mean()
female_avg = amt_df[amt_df['Gender']=='F']['Purchase'].mean()
print("Average amount spend by Male customers:{:.2f}".format(male_avg))
print("Average amount spend by Female customers:{:.2f}".format(female_avg))
```

```
Average amount spend by Male customers:925344.40
Average amount spend by Female customers:712024.39
```

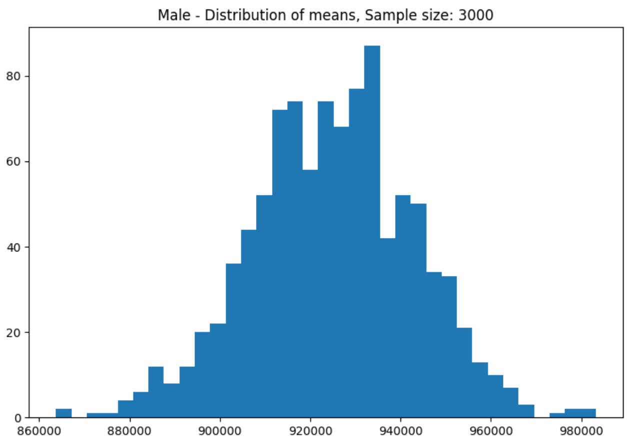
Male customers spend more money than female customers.

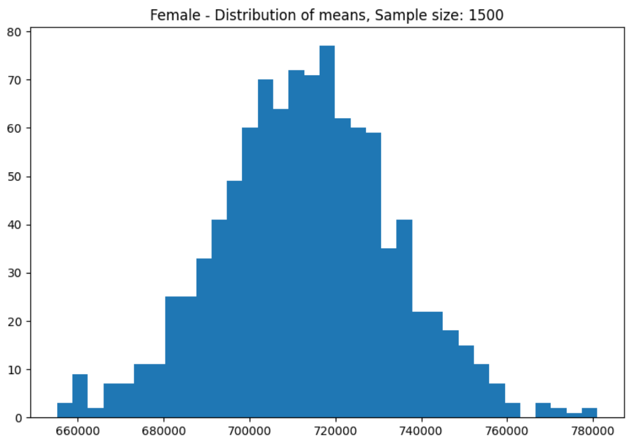
2.Confidence intervals and distribution of the mean of the expenses by female and

male customers

```
male_df = amt_df[amt_df['Gender']=='M']
female_df = amt_df[amt_df['Gender']=='F']
genders = ["M", "F"]
male_sample_size = 3000
female_sample_size = 1500
num_repitions = 1000
```

```
male_means = []
female_means = []
for _ in range(num_repitions):
  male_mean = male_df.sample(male_sample_size,
  replace=True)['Purchase'].mean()
  female_mean = female_df.sample(female_sample_size,
  replace=True)['Purchase'].mean()
  male_means.append(male_mean)
  female_means.append(female_mean)
########################################################################
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))
axis[0].hist(male_means, bins=35)
axis[1].hist(female_means, bins=35)
axis[0].set_title("Male - Distribution of means, Sample size: 3000")
axis[1].set_title("Female - Distribution of means, Sample size: 1500")
plt.show()
```





```
print("Population mean - Mean of sample means of amount spend for Male:{:.2f}".
print("Population mean - Mean of sample means of amount spend forFemale: {:.2f}
print("\nMale - Sample mean: {:.2f} Sample std:{:.2f}".format(male_df['Purchase
print("Female - Sample mean: {:.2f} Sample std:{:.2f}".format(female_df['Purcha
female_df['Purchase'].std()))
```

```
Population mean - Mean of sample means of amount spend for Male:925046.39
Population mean - Mean of sample means of amount spend forFemale: 713177.45

Male - Sample mean: 925344.40 Sample std:985830.10
Female - Sample mean: 712024.39 Sample std:807370.73
```

Observation Now using the Central Limit Theorem for the population we can say that:

1. Average amount spend by male customers is 9,26,341.86
2. Average amount spend by female customers is 7,11,704.09

3. Are confidence intervals of average male and female spending overlapping? How can Walmart

leverage this conclusion to make changes or improvements?

```
male_margin_of_error_clt = 1.96*male_df['Purchase'].std()/np.sqrt(len(male_df))
male_sample_mean = male_df['Purchase'].mean()
male_lower_lim = male_sample_mean - male_margin_of_error_clt
male_upper_lim = male_sample_mean + male_margin_of_error_clt
female_margin_of_error_clt =1.96*female_df['Purchase'].std()/np.sqrt(len(female
female_sample_mean = female_df['Purchase'].mean()
female_lower_lim = female_sample_mean - female_margin_of_error_clt
female_upper_lim = female_sample_mean + female_margin_of_error_clt
print("Male confidence interval of means: ({:.2f},{:.2f})".format(male_lower_li
print("Female confidence interval of means: ({:.2f},{:.2f})".format(female_lowe
```

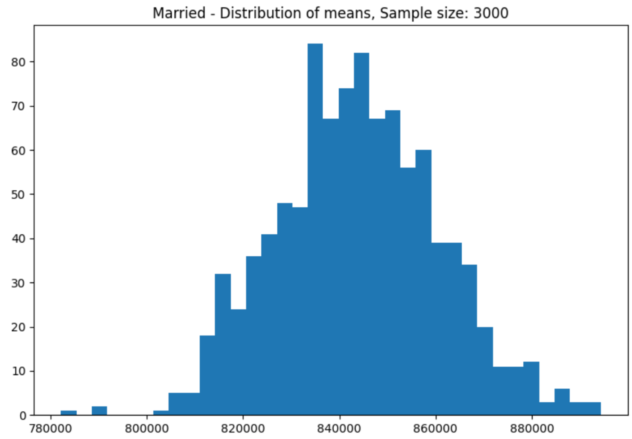
```
Male confidence interval of means: (895617.83,955070.97)
Female confidence interval of means: (673254.77,750794.02)
```

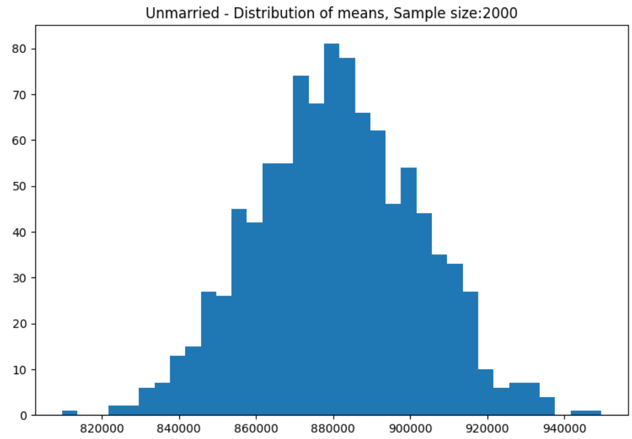
Now we can infer about the population that, 95% of the times:

1. Average amount spend by male customer will lie in between: (895617.83, 955070.97)
2. Average amount spend by female customer will lie in between: (673254.77, 750794.02)

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

```
amt_df = df.groupby(['User_ID', 'Marital_Status'])[['Purchase']].sum()
amt_df = amt_df.reset_index()
amt_df
amt_df['Marital_Status'].value_counts()
marid_samp_size = 3000
unmarid_sample_size = 2000
num_repitions = 1000
marid_means = []


unmarid_means = []
for _ in range(num_repitions):
  marid_mean =amt_df[amt_df['Marital_Status']==1].sample(marid_samp_size,replac
  unmarid_mean =amt_df[amt_df['Marital_Status']==0].sample(unmarid_sample_size,
  marid_means.append(marid_mean)
  unmarid_means.append(unmarid_mean)

fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))
axis[0].hist(marid_means, bins=35)
axis[1].hist(unmarid_means, bins=35)
axis[0].set_title("Married - Distribution of means, Sample size: 3000")
axis[1].set_title("Unmarried - Distribution of means, Sample size:2000")
plt.show()
print("Population mean - Mean of sample means of amount spend forMarried: {:.2f
print("Population mean - Mean of sample means of amount spend forUnmarried: {:.
print("\nMarried - Sample mean: {:.2f} Sample std:{:.2f}".format(amt_df[amt_df[
print("Unmarried - Sample mean: {:.2f} Sample std:{:.2f}".format(amt_df[amt_df[
for val in ["Married", "Unmarried"]:
  new_val = 1 if val == "Married" else 0
  new_df = amt_df[amt_df['Marital_Status']==new_val]
  margin_of_error_clt =1.96*new_df['Purchase'].std()/np.sqrt(len(new_df))
  sample_mean = new_df['Purchase'].mean()
  lower_lim = sample_mean - margin_of_error_clt
  upper_lim = sample_mean + margin_of_error_clt
  print("{} confidence interval of means: ({:.2f},{:.2f})".format(val, lower_li
```






```
Population mean - Mean of sample means of amount spend forMarried: 843861.7
Population mean - Mean of sample means of amount spend forUnmarried: 881008

Married - Sample mean: 843526.80 Sample std:935352.12
Unmarried - Sample mean: 880575.78 Sample std:949436.25
Married confidence interval of means: (806668.83,880384.76)
Unmarried confidence interval of means: (848741.18,912410.38)
```

5: Results when the same activity is performed for Age:

Calculating the average amount spent by Age

```
amt_df = df.groupby(['User_ID', 'Age'])[['Purchase']].sum()
amt_df = amt_df.reset_index()
amt_df
amt_df['Age'].value_counts()
sample_size = 200
num_repitions = 1000
all_means = {}
age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+',
'0-17']
for age_interval in age_intervals:
  all_means[age_interval] = []
for age_interval in age_intervals:
  for _ in range(num_repitions):
    mean = amt_df[amt_df['Age']==age_interval].sample(sample_size, replace=True)
    all_means[age_interval].append(mean)

for val in ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']:
  new_df = amt_df[amt_df['Age']==val]
  margin_of_error_clt = 1.96*new_df['Purchase'].std()/np.sqrt(len(new_df))
  sample_mean = new_df['Purchase'].mean()

  lower_lim = sample_mean - margin_of_error_clt
  upper_lim = sample_mean + margin_of_error_clt
  print("For age {} --> confidence interval of means: ({:.2f},{:.2f})".format(va
```

```
For age 26-35 --> confidence interval of means: (945034.42,1034284.21)
For age 36-45 --> confidence interval of means: (823347.80,935983.62)
For age 18-25 --> confidence interval of means: (801632.78,908093.46)
For age 46-50 --> confidence interval of means: (713505.63,871591.93)
For age 51-55 --> confidence interval of means: (692392.43,834009.42)
For age 55+ --> confidence interval of means: (476948.26,602446.23)
For age 0-17 --> confidence interval of means: (527662.46,710073.17)
```

Insights

1. ~ 80% of the users are between the age 18-50 (40%: 26-35, 18%: 18-25, 20%: 36-45)
2. 75% of the users are Male and 25% are Female
3. 60% Single, 40% Married
4. 35% Staying in the city from 1 year, 18% from 2 years, 17% from 3 years Total of 20 product categories are there
5. There are 20 diffrenent types of occupations in the city
6. Most of the users are Male

7. There are 20 different types of Occupation and Product_Category
8. More users belong to B City_Category
9. More users are Single as compare to Married
10. Product_Category - 1, 5, 8, & 11 have highest purchasing frequency.

Average amount spend by Male customers: 925344.40

Average amount spend by Female customers: 712024.39

Confidence Interval by Gender:

Now using the Central Limit Theorem for the population:

1. Average amount spend by male customers is 9,26,341.86
2. Average amount spend by female customers is 7,11,704.09 Now we can infer about the population that, 95% of the times:
3. Average amount spend by male customer will lie in between: (895617.83, 955070.97)
4. Average amount spend by female customer will lie in between: (673254.77, 750794.02) Confidence Interval by Marital_Status
5. Married confidence interval of means: (806668.83, 880384.76)
6. Unmarried confidence interval of means: (848741.18, 912410.38) Confidence Interval by Age
7. For age 26-35 --> confidence interval of means: (945034.42, 1034284.21)
8. For age 36-45 --> confidence interval of means: (823347.80, 935983.62)
9. For age 18-25 --> confidence interval of means: (801632.78, 908093.46)
10. For age 46-50 --> confidence interval of means: (713505.63, 871591.93)
11. For age 51-55 --> confidence interval of means: (692392.43, 834009.42)
12. For age 55+ --> confidence interval of means: (476948.26, 602446.23)
13. For age 0-17 --> confidence interval of means: (527662.46, 710073.17) Recommendations
14. Men spent more money than women, So company should focus on retaining the male customers and getting more male customers.
15. Product_Category - 1, 5, 8, & 11 have highest purchasing frequency. it means these are the products in these categories are liked more by customers. Company can focus on

selling more of these products or selling more of the products which are purchased less.

16. Unmarried customers spend more money than married customers, So company should focus on acquisition of Unmarried customers.
17. Customers in the age 18-45 spend more money than the others, So company should focus on acquisition of customers who are in the age 18-45
18. Male customers living in City_Category C spend more money than other male customers living in B or C, Selling more products in the City_Category C will help the company increase the revenue.

```
Start coding or generate with AI.
```