Walmart-Confidence Interval and CLT

About Walmart- Walmart is an American multinational retail corporation that operates a chain of supercenters, discount departmental stores, and grocery stores from the United States. Walmart has more than 100 million customers worldwide.

Business Problem- The Management team at Walmart Inc. wants to analyze the customer purchase behavior (specifically, purchase amount) against the customer's gender and the various other factors to help the business make better decisions.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import stats
import warnings
warnings.filterwarnings('ignore')
import copy
```

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

	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	

```
df_w.tail()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_
425730	1005550	P00028542	M	36-45	15.0	B	
425731	1005550	P00121042	M	36-45	15.0	B	
425732	1005550	P00295942	M	36-45	15.0	B	

```
df_w.shape
```

```
(425735, 10)
```

```
df_w.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 425735 entries, 0 to 425734
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     425735 non-null  int64  
 1   Product_ID                  425734 non-null  object 
 2   Gender                      425734 non-null  object 
 3   Age                         425734 non-null  object 
 4   Occupation                  425734 non-null  float64
 5   City_Category               425734 non-null  object 
 6   Stay_In_Current_City_Years  425734 non-null  object 
 7   Marital_Status              425734 non-null  float64
 8   Product_Category            425734 non-null  float64
 9   Purchase                    425734 non-null  float64
dtypes: float64(4), int64(1), object(5)
memory usage: 32.5+ MB
```

Insights-

1.From the above analysis it is clear that the data has total of 10 features and mixed of alphanumeric data. 2.Apart from Purchase column all other have categorical in nature.

Changing the datatype of columns-

```
for i in df_w.columns[:-1]:
  df_w[i]=df_w[i].astype('category')
```

```
df_w.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 425735 entries, 0 to 425734
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype   
---  ------                      --------------   -----   
 0   User_ID                     425735 non-null  category
 1   Product_ID                  425734 non-null  category
 2   Gender                      425734 non-null  category
 3   Age                         425734 non-null  category
 4   Occupation                  425734 non-null  category
 5   City_Category               425734 non-null  category
 6   Stay_In_Current_City_Years  425734 non-null  category
 7   Marital_Status              425734 non-null  category
 8   Product_Category            425734 non-null  category
 9   Purchase                    425734 non-null  float64 
dtypes: category(9), float64(1)
memory usage: 8.0 MB
```

Statistical Summary-

Statistical summary of object type columns-

```
df_w.describe(include='category')
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Curre
count	425735	425734	425734	425734	425734.0	425734	
unique	5892	3586	2	7	21.0	3	
top	1001680	P00265242	M	26-35	4.0	B	
freq	816	1428	320939	169750	56166.0	179698	

Insights- 1.UserId-out of total 425735 there are 5892 unique userid indicating same customers buying multiple products. 2.ProductId-Among 425735 transactions there are 3586 unique products with the product P00265242 being the highest seller. 3.Gender-Out of total transactions,320939 transactions has been done by male gender indicating a significant disparity in the purchase behaviour between males and females during the Black Friday sale. 4.Age-We have 7 unique age groups in the dataset.26-25 has the 169750 transactions. 5.Stay_In_Current_City_Years-customers who has 1 year of stay in current years accounted to maximum of 149942 transactions as compared to those who lived for 2,3,4 years. 6.Marital_Status-59% of transactions were done by unmarried customers and rest is being done by married cutomers.

Statistical summary of numerical data types-

```
df_w.describe()
```

	Purchase
count	425734.000000
mean	9329.499890
std	4978.200428
min	185.000000
25%	5866.000000
50%	8061.000000
75%	12070.000000
max	23961.000000

Insights- The Purchase behaviour is varying in nature with 185$ being the min amount and 23961 being the max amount.The median purchase amount is noticably lower than the mean purchase amount indicating the right-skewed behaviour where some high values are pulling the mean towards the right side.

Duplicate detection-

```
df_w.duplicated().sum()
df_w
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_
0	1000001	P00069042	F	0-17	10.0	A	
1	1000001	P00248942	F	0-17	10.0	A	
2	1000001	P00087842	F	0-17	10.0	A	
3	1000001	P00085442	F	0-17	10.0	A	
4	1000002	P00285442	M	55+	16.0	C	
...	...	...	...	...	...	...	
425730	1005550	P00028542	M	36-45	15.0	B	
425731	1005550	P00121042	M	36-45	15.0	B	
425732	1005550	P00295942	M	36-45	15.0	B	
425733	1005550	P00123842	M	36-	15.0	B	

```
df_w.duplicated().value_counts()
df_w
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_
0	1000001	P00069042	F	0-17	10.0	A	
1	1000001	P00248942	F	0-17	10.0	A	
2	1000001	P00087842	F	0-17	10.0	A	
3	1000001	P00085442	F	0-17	10.0	A	
4	1000002	P00285442	M	55+	16.0	C	
...	...	...	...	...	...	...	
425730	1005550	P00028542	M	36-45	15.0	B	
425731	1005550	P00121042	M	36-45	15.0	B	
425732	1005550	P00295942	M	36-45	15.0	B	
425733	1005550	P00123842	M	36-	15.0	B	

Insights-There are no duplicate values in the dataset.

Checking the unique values for columns-

```
for i in df_w.columns:
  print('Unique values in',i,'column are:-')
  print(df_w[i].unique())
  print("-"*70)
```

```
Unique values in User_ID column are:-
[1000001, 1000002, 1000003, 1000004, 1000005, ..., 1004871, 1004113, 1005391, 100152
Length: 5892
Categories (5892, int64): [10055, 1000001, 1000002, 1000003, ..., 1006037, 1006038, 1
----------------------------------------------------------------------
Unique values in Product_ID column are:-
['P00069042', 'P00248942', 'P00087842', 'P00085442', 'P00285442', ..., 'P00330142', '
Length: 3587
Categories (3586, object): ['P00000142', 'P00000242', 'P00000342', 'P00000442', ...,
                            'P0099742', 'P0099842', 'P0099942']
----------------------------------------------------------------------
Unique values in Gender column are:-
['F', 'M', NaN]
Categories (2, object): ['F', 'M']
----------------------------------------------------------------------
Unique values in Age column are:-
['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25', NaN]
Categories (7, object): ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
----------------------------------------------------------------------
Unique values in Occupation column are:-
```

```
[10.0, 16.0, 15.0, 7.0, 20.0, ..., 5.0, 14.0, 13.0, 6.0, NaN]
Length: 22
Categories (21, float64): [0.0, 1.0, 2.0, 3.0, ..., 17.0, 18.0, 19.0, 20.0]
----------------------------------------------------------------------
Unique values in City_Category column are:-
['A', 'C', 'B', NaN]
Categories (3, object): ['A', 'B', 'C']
----------------------------------------------------------------------
Unique values in Stay_In_Current_City_Years column are:-
['2', '4+', '3', '1', '0', NaN]
Categories (5, object): ['0', '1', '2', '3', '4+']
----------------------------------------------------------------------
Unique values in Marital_Status column are:-
[0.0, 1.0, NaN]
Categories (2, float64): [0.0, 1.0]
----------------------------------------------------------------------
Unique values in Product_Category column are:-
[3.0, 1.0, 12.0, 8.0, 5.0, ..., 18.0, 10.0, 17.0, 9.0, NaN]
Length: 19
Categories (18, float64): [1.0, 2.0, 3.0, 4.0, ..., 15.0, 16.0, 17.0, 18.0]
----------------------------------------------------------------------
Unique values in Purchase column are:-
[ 8370. 15200.  1422. ...  9065.  9519.    nan]
----------------------------------------------------------------------
```

Insights- 1.The dataset doesn"t contain any abnormal values. 2.We will convert the 0,1 in Marital columns as Married and Unmarried.

Replacing the values of Mrital column with married and unmarried-

```
df_w['Marital_Status']=df_w['Marital_Status'].replace({'0':'Unmarried','1':'Married'})
df_w['Marital_Status'].unique()
```

```
[0.0, 1.0, NaN]
Categories (2, float64): [0.0, 1.0]
```

Missing value detection-

```
df_w.isnull().sum()
```

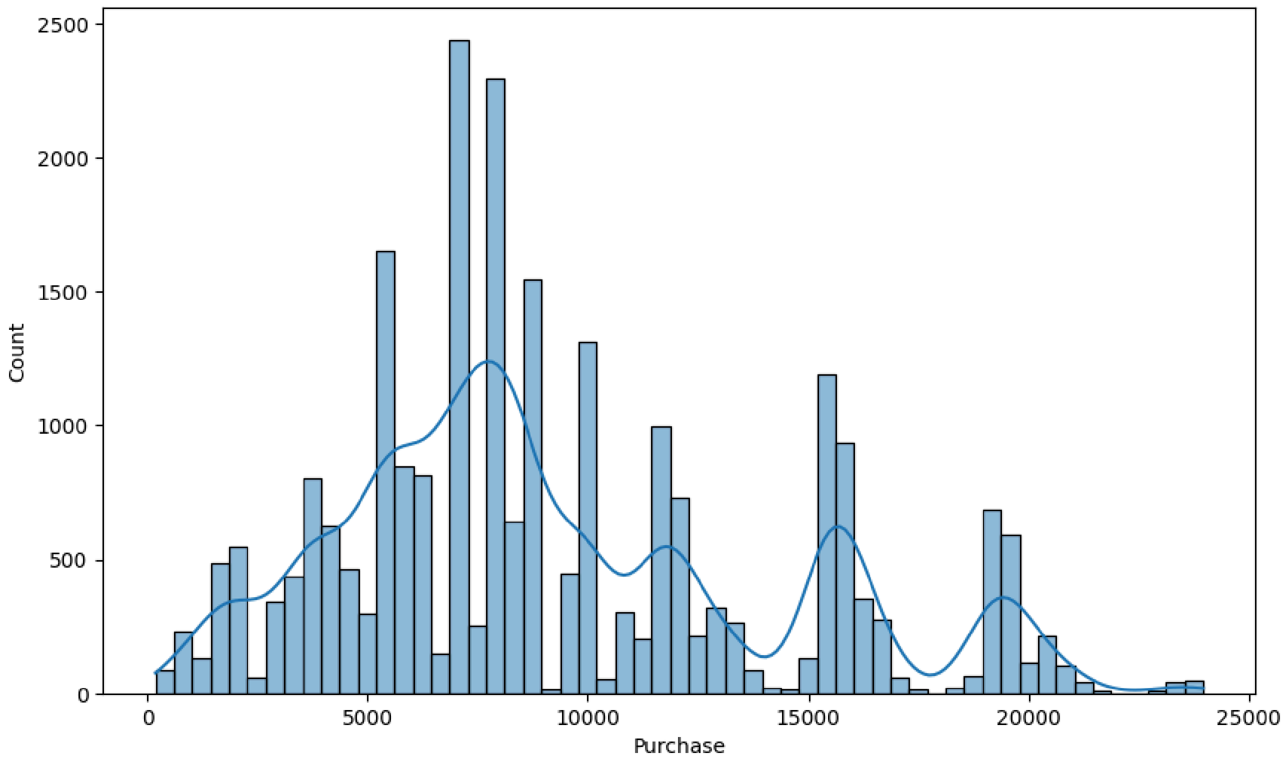
	0
User_ID	0
Product_ID	1
Gender	1
Age	1
Occupation	1
City_Category	1
Stay_In_Current_City_Years	1
Marital_Status	1
Product_Category	1
Purchase	1

dtype: int64

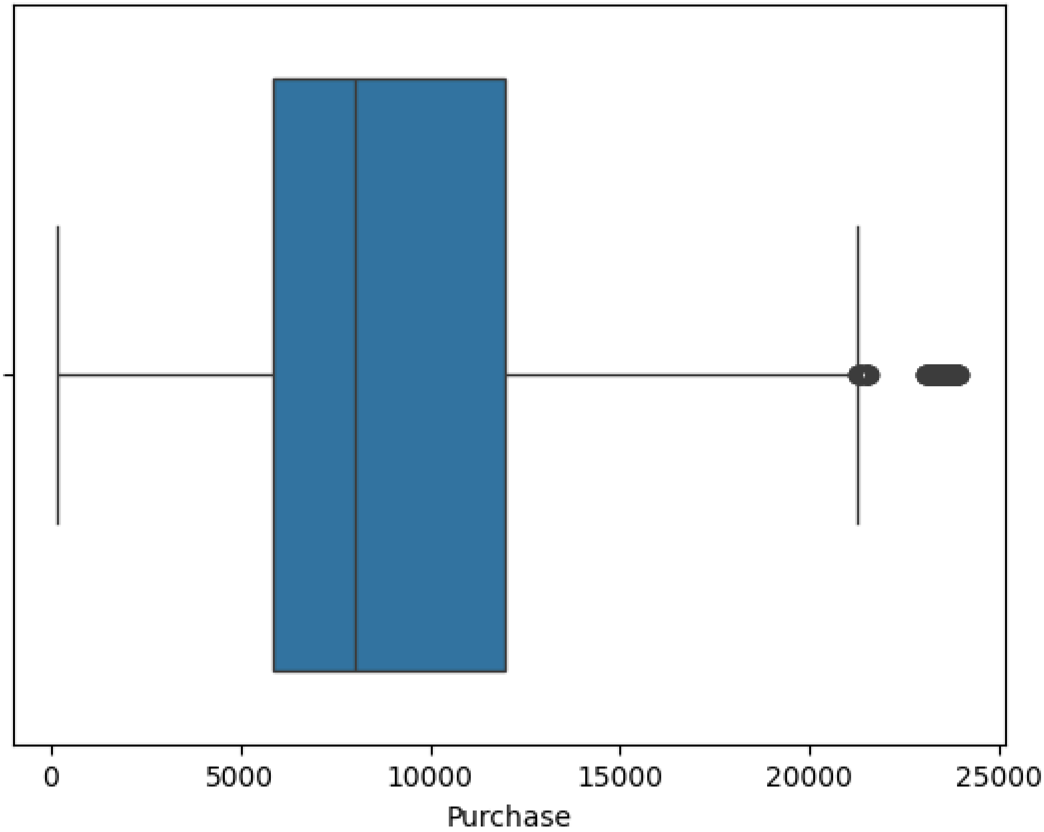
Insights- Dataset contains some missing values in it.

Univariate Analysis-

```
plt.figure(figsize=(10, 6))
sns.histplot(data=df_w, x='Purchase', kde=True)
plt.show()
```



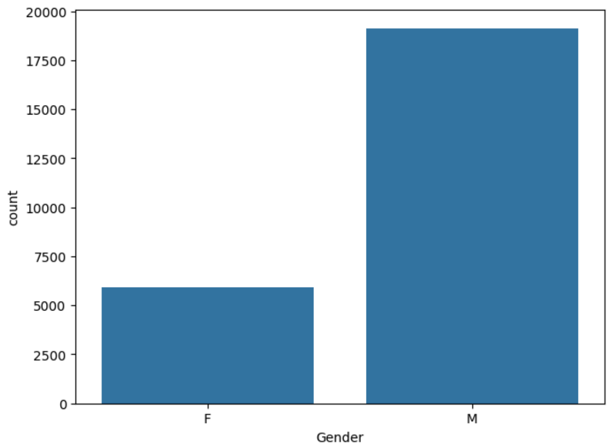

```
sns.boxplot(data=df_w, x='Purchase', orient='h')
plt.show()
```

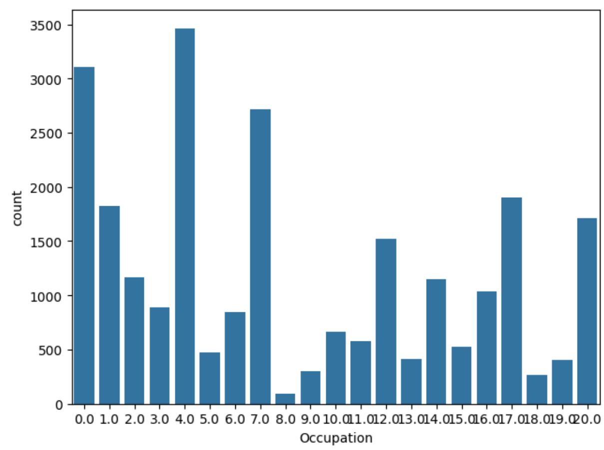


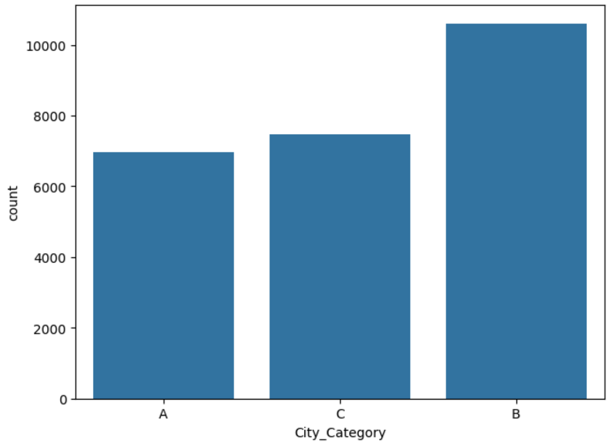
Observation Purchase is having outliers

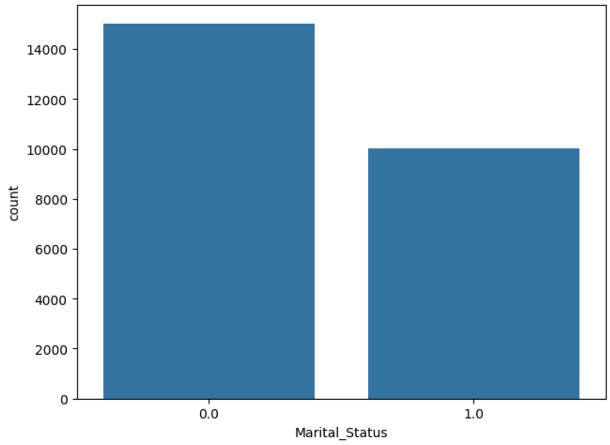
Understanding the distribution of data for the categorical variables¶ Gender Age Occupation City_Category Stay_In_Current_City_Years Marital_Status Product_Category

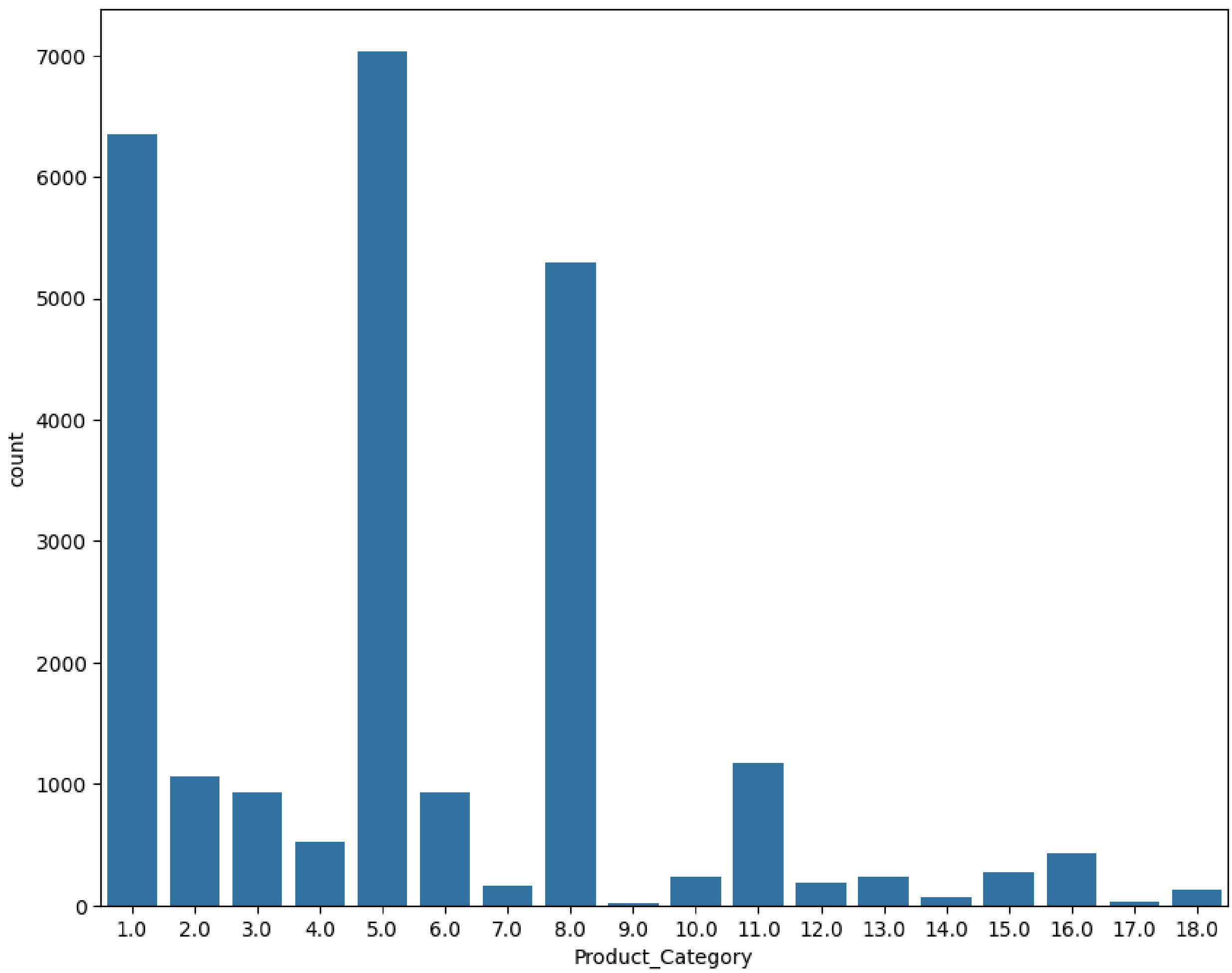
```
categorical_cols = ['Gender', 'Occupation','City_Category','Marital_Status','Product_Cate

fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(16, 12))
sns.countplot(data=df_w, x='Gender', ax=axs[0,0])
sns.countplot(data=df_w, x='Occupation', ax=axs[0,1])
sns.countplot(data=df_w, x='City_Category', ax=axs[1,0])
sns.countplot(data=df_w, x='Marital_Status', ax=axs[1,1])
plt.show()

plt.figure(figsize=(10, 8))
sns.countplot(data=df_w, x='Product_Category')
plt.show()
```


count
20000
17500
15000
12500
10000
7500
5000
2500
0
F
M
Gender


count
3500
3000
2500
2000
1500
1000
500
0
Occupation

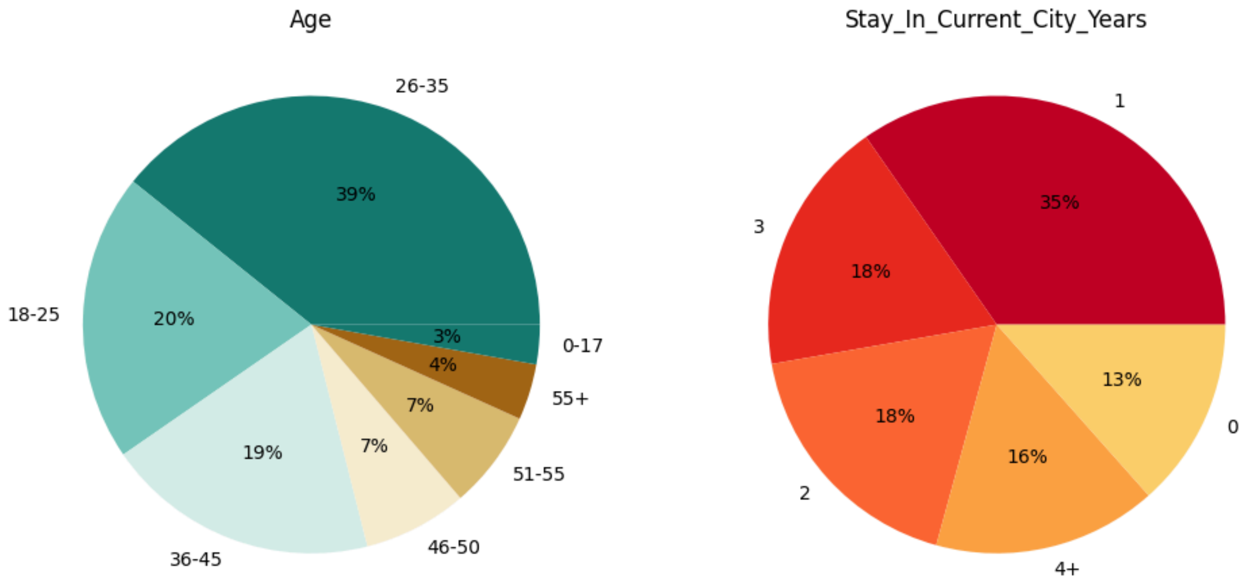

count
10000
8000
6000
4000
2000
0
A
C
B
City_Category


count
14000
12000
10000
8000
6000
4000
2000
0
0.0
1.0
Marital_Status


count
7000
6000
5000
4000
3000
2000
1000
0
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0 17.0 18.0
Product_Category

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

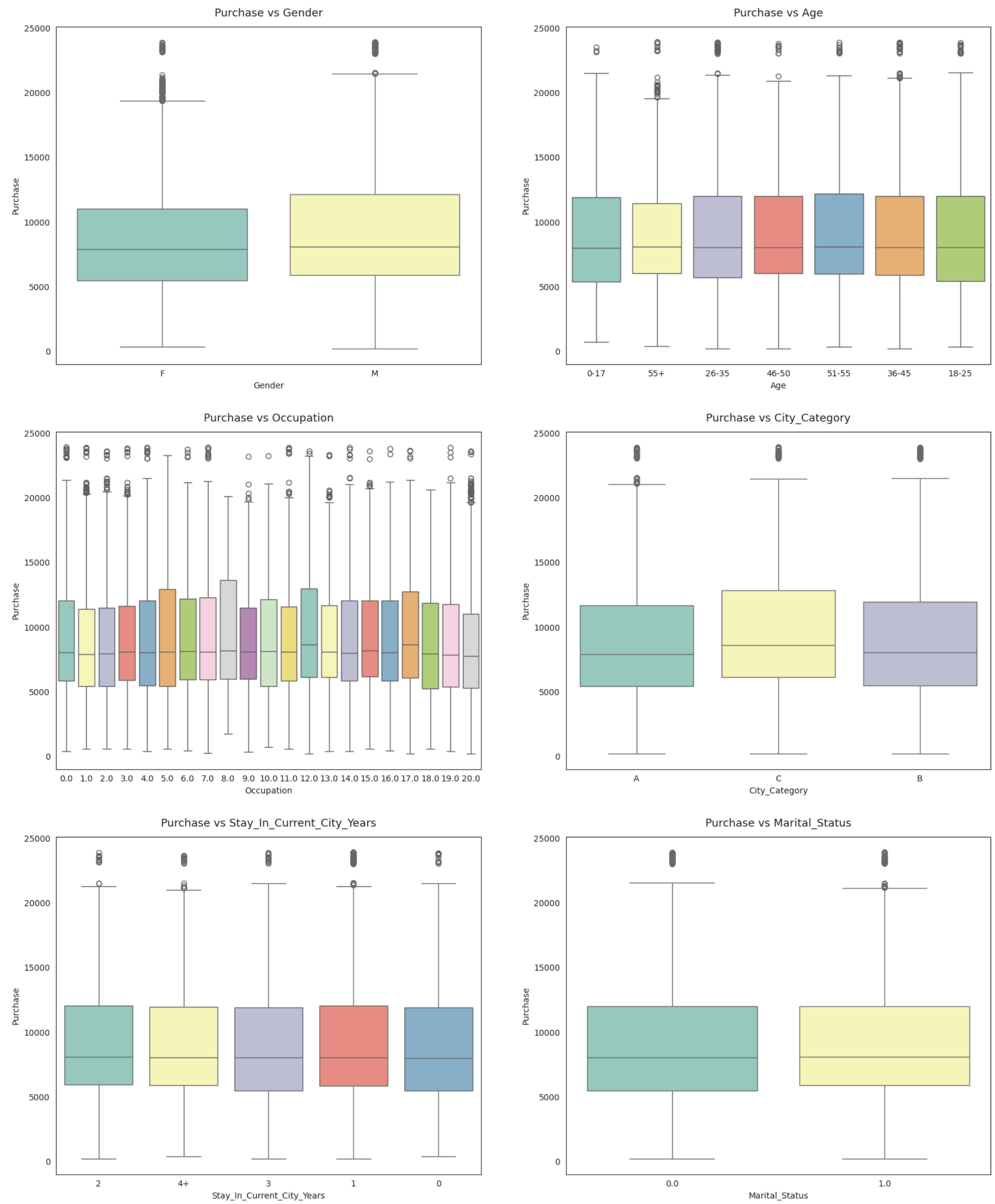
```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(12, 8))

data = df_w['Age'].value_counts(normalize=True)*100
palette_color = sns.color_palette('BrBG_r')
axs[0].pie(x=data.values, labels=data.index, autopct='%.0f%%', colors=palette_color)
axs[0].set_title("Age")

data = df_w['Stay_In_Current_City_Years'].value_counts(normalize=True)*100
palette_color = sns.color_palette('YlOrRd_r')
axs[1].pie(x=data.values, labels=data.index, autopct='%.0f%%', colors=palette_color)

axs[1].set_title("Stay_In_Current_City_Years")


plt.show()
```

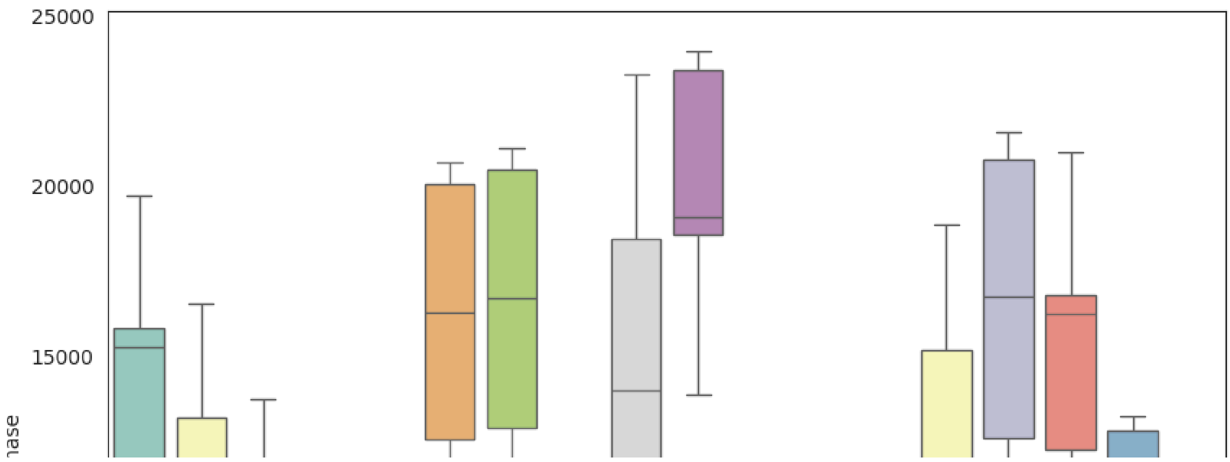


Bi-variate Analysis-

```
attrs = ['Gender', 'Age', 'Occupation', 'City_Category', 'Stay_In_Current_City_Years', 'M
sns.set_style("white")
```

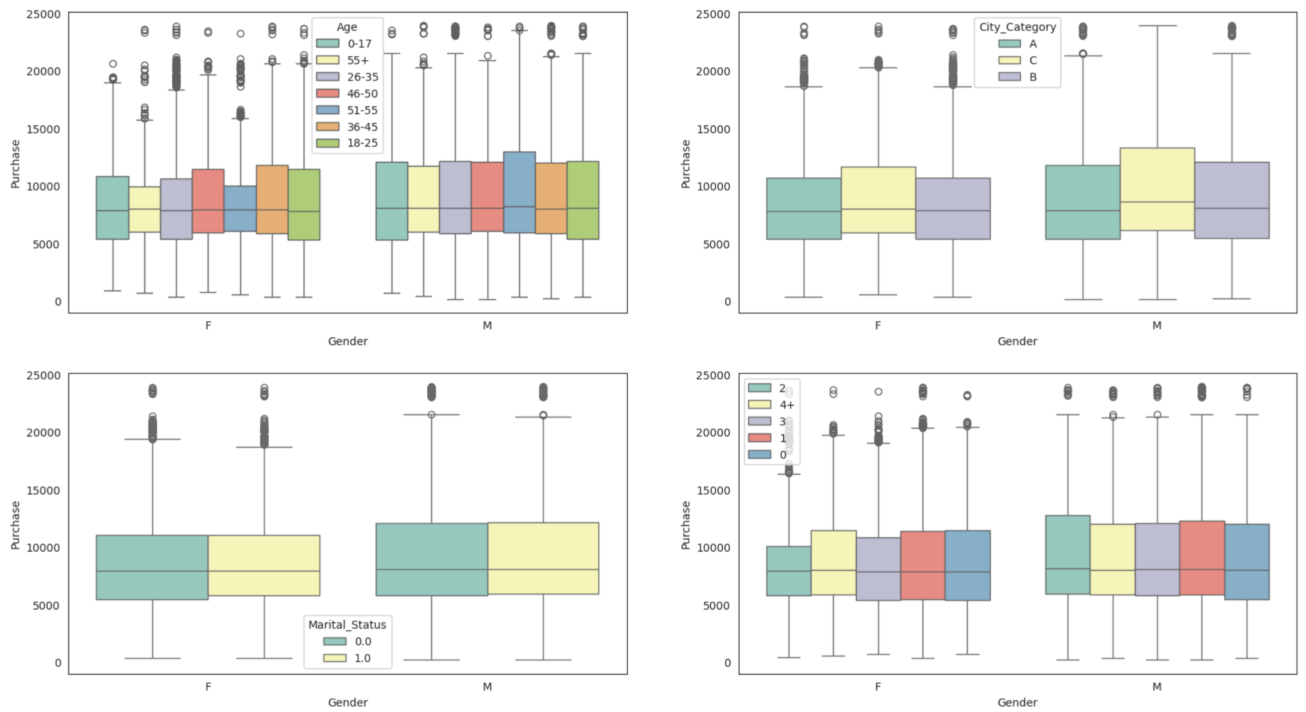
```
fig, axs = plt.subplots(nrows=3, ncols=2, figsize=(20, 16))
fig.subplots_adjust(top=1.3)
count = 0
for row in range(3):
    for col in range(2):
        sns.boxplot(data=df_w, y='Purchase', x=attrs[count], ax=axs[row, col], palette='S
        axs[row,col].set_title(f"Purchase vs {attrs[count]}", pad=12, fontsize=13)
        count += 1
plt.show()

plt.figure(figsize=(10, 8))
sns.boxplot(data=df_w, y='Purchase', x=attrs[-1], palette='Set3')
plt.show()
```





Multivariate Analysis-

```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20, 6))
fig.subplots_adjust(top=1.5)
sns.boxplot(data=df_w, y='Purchase', x='Gender', hue='Age', palette='Set3', ax=axs[0,0])
sns.boxplot(data=df_w, y='Purchase', x='Gender', hue='City_Category', palette='Set3', ax=

sns.boxplot(data=df_w, y='Purchase', x='Gender', hue='Marital_Status', palette='Set3', ax
sns.boxplot(data=df_w, y='Purchase', x='Gender', hue='Stay_In_Current_City_Years', palett
axs[1,1].legend(loc='upper left')

plt.show()
```



Average amount spend per customer for Male and Female-

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

	User_ID	Gender	Purchase
0	1000001	F	26049.0
1	1000002	M	7969.0
2	1000003	M	15227.0
3	1000004	M	50755.0
4	1000005	M	38820.0
...	...	...	...
3383	1003841	M	312575.0
3384	1003842	F	161398.0
3385	1003843	F	25256.0
3386	1003844	M	13615.0
3387	1003845	M	19456.0

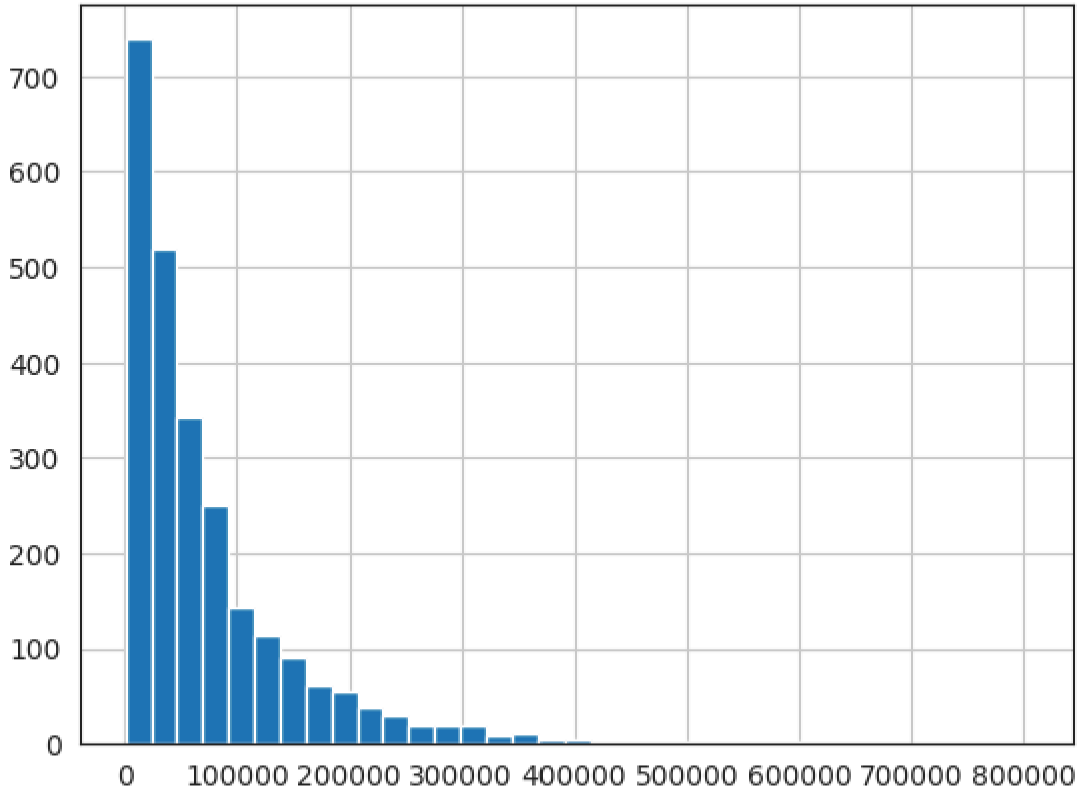
3388 rows × 3 columns

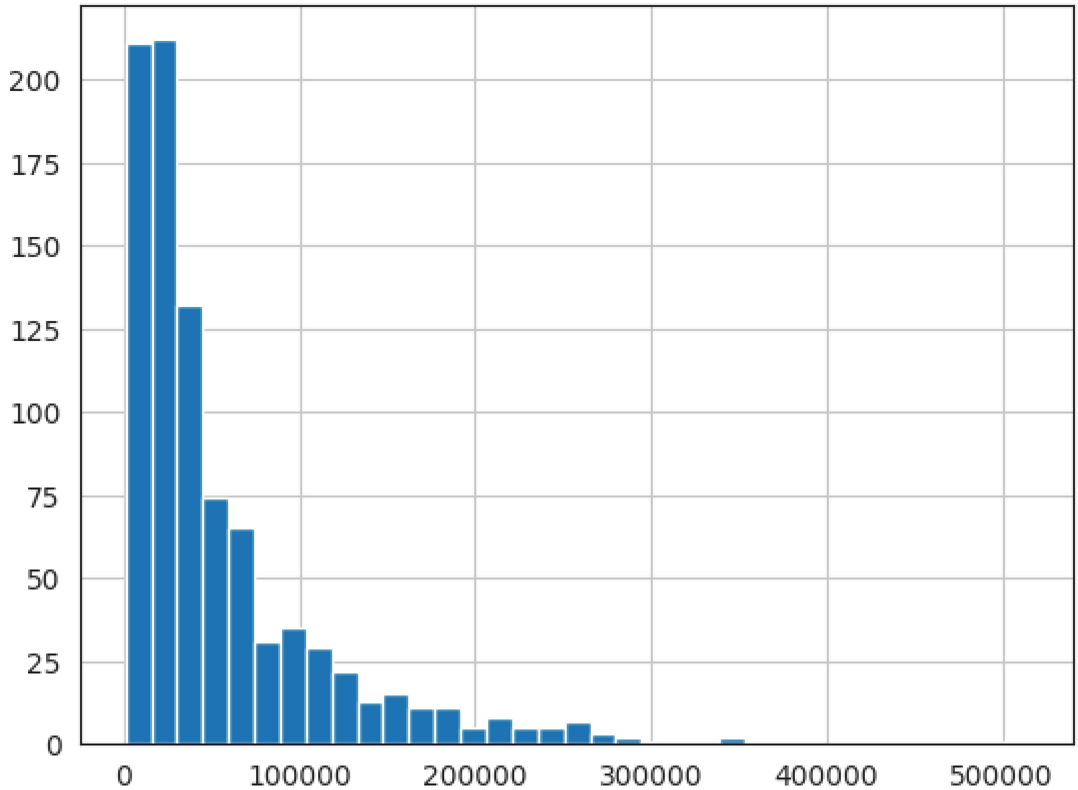
```
# Gender wise value counts in avg_amt_df
amt_df['Gender'].value_counts()
```

Gender	count
M	2482
F	906

dtype: int64

```
# 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()
```





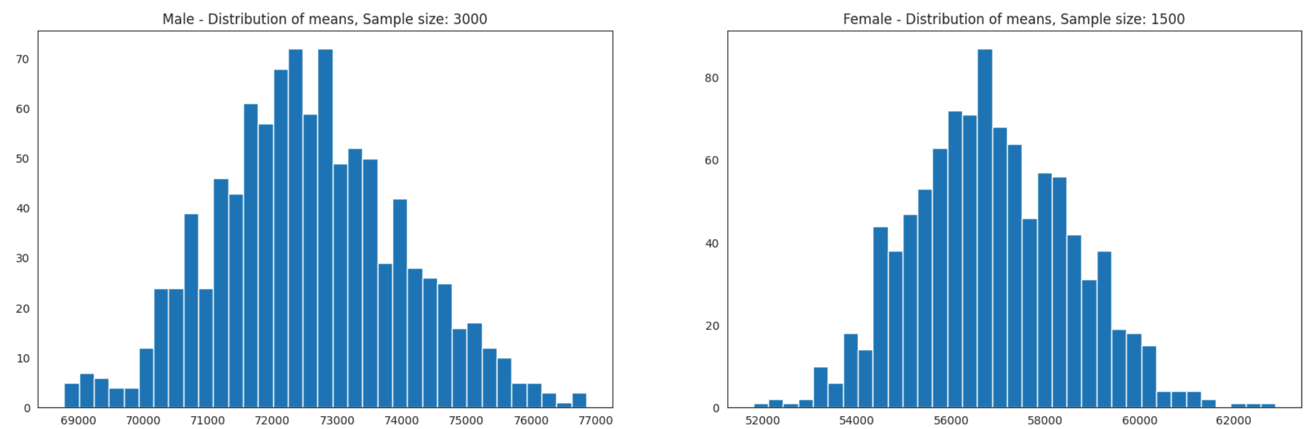

```
male_avg = amt_df[amt_df['Gender']=='M']['Purchase'].mean()
female_avg = amt_df[amt_df['Gender']=='F']['Purchase'].mean()

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: 72471.24
Average amount spend by Female customers: 56823.99
```

Observation-

Male customers spend more money than female 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}".format(np
print("Population mean - Mean of sample means of amount spend for Female: {:.2f}".format(

print("\nMale - Sample mean: {:.2f} Sample std: {:.2f}".format(male_df['Purchase'].mean()
print("Female - Sample mean: {:.2f} Sample std: {:.2f}".format(female_df['Purchase'].mean
```

```
Population mean - Mean of sample means of amount spend for Male: 72554.92
Population mean - Mean of sample means of amount spend for Female: 56857.73

Male - Sample mean: 72471.24 Sample std: 77679.49
Female - Sample mean: 56823.99 Sample std: 65526.13
```

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

```
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_df))
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_lim, male_u
print("Female confidence interval of means: ({:.2f}, {:.2f})".format(female_lower_lim, fe
```

```
Male confidence interval of means: (69415.19, 75527.30)
Female confidence interval of means: (52557.15, 61090.83)
```

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

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

```
amt_df = df_w.groupby(['User_ID', 'Marital_Status'])[['Purchase']].sum()
amt_df = amt_df.reset_index()
amt_df
```

	User_ID	Marital_Status	Purchase
0	1000001	0.0	26049.0
1	1000002	0.0	7969.0
2	1000003	0.0	15227.0
3	1000004	1.0	50755.0
4	1000005	1.0	38820.0
...	...	...	...
3383	1003841	0.0	312575.0
3384	1003842	0.0	161398.0
3385	1003843	0.0	25256.0
3386	1003844	1.0	13615.0
3387	1003845	1.0	19456.0

3388 rows × 3 columns

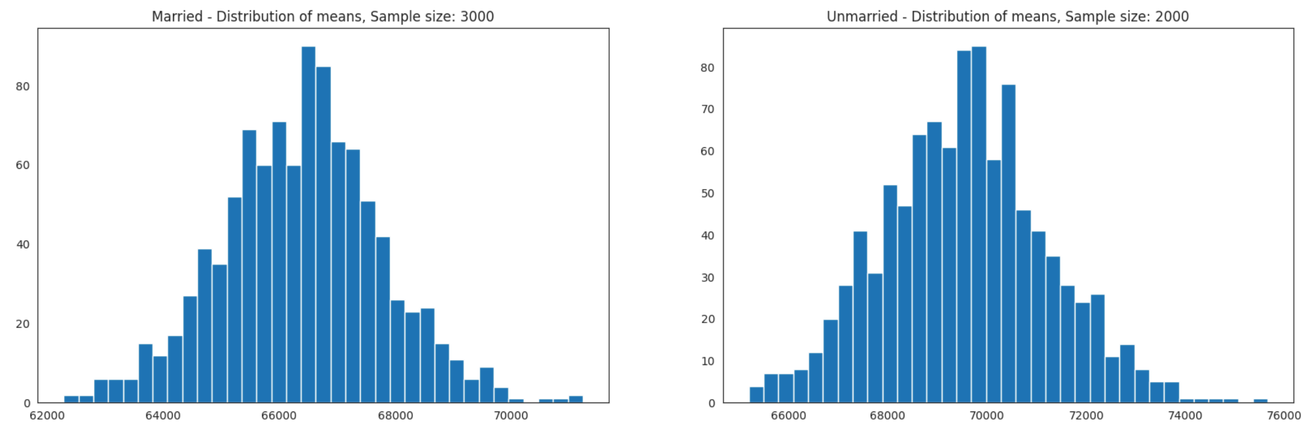
```
amt_df['Marital_Status'].value_counts()
```

	count
Marital_Status	
0.0	1988
1.0	1400

dtype: int64

```
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, replace=True
    unmarid_mean = amt_df[amt_df['Marital_Status']==0].sample(unmarid_sample_size, replac

    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 for Married: {:.2f}".format
print("Population mean - Mean of sample means of amount spend for Unmarried: {:.2f}".form

print("\nMarried - Sample mean: {:.2f} Sample std: {:.2f}".format(amt_df[amt_df['Marital_
print("Unmarried - Sample mean: {:.2f} Sample std: {:.2f}".format(amt_df[amt_df['Marital_
```




```
Population mean - Mean of sample means of amount spend for Married: 66383.43
Population mean - Mean of sample means of amount spend for Unmarried: 69555.30

Married - Sample mean: 66360.11 Sample std: 75285.08
Unmarried - Sample mean: 69643.87 Sample std: 74676.68
```

```
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_lim, uppe
```

Calculating the average amount spent by Age-

```
amt_df = df_w.groupby(['User_ID', 'Age'])[['Purchase']].sum()
amt_df = amt_df.reset_index()
amt_df
```

	User_ID	Age	Purchase
0	1000001	0-17	38891.0
1	1000002	55+	37417.0
2	1000003	26-35	49947.0
3	1000004	46-50	66607.0
4	1000005	26-35	50684.0
...	...	...	...
5421	1006035	26-35	42357.0
5422	1006036	26-35	196339.0
5423	1006037	46-50	86597.0
5424	1006039	46-50	50364.0
5425	1006040	26-35	95780.0

```
amt_df['Age'].value_counts()
```

	count
Age	
26-35	1907