1. Checking the structure & characteristics of the dataset

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

```
df =pd.read_csv("/content/walmart_data.csv")
```

```
print(f"Number of rows: {df.shape[0]:,} \nNumber of columns:
{df.shape[1]}")
```

```
[ ]  1 import numpy as np
     2 import pandas as pd
     3 import matplotlib.pyplot as plt
     4 import seaborn as sns
```

```
[ ]  1 df =pd.read_csv("/content/walmart_data.csv")
```

```
[ ]  1 print(f"Number of rows: {df.shape[0]:,} \nNumber of columns: {df.shape[1]}")
```

```
Number of rows: 50,080
Number of columns: 10
```

```
df.info()
```

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50080 entries, 0 to 50079
Data columns (total 10 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   User_ID                     50080 non-null  int64
 1   Product_ID                  50080 non-null  object
 2   Gender                      50080 non-null  object
 3   Age                         50080 non-null  object
 4   Occupation                  50080 non-null  int64
 5   City_Category               50080 non-null  object
 6   Stay_In_Current_City_Years  50080 non-null  object
 7   Marital_Status              50080 non-null  int64
 8   Product_Category            50079 non-null  float64
 9   Purchase                    50079 non-null  float64
dtypes: float64(2), int64(3), object(5)
memory usage: 3.8+ MB
```

Change the data types of - `Occupation`, `Marital_Status`, `Product_Category`

```
cols = ['Occupation', 'Marital_Status', 'Product_Category']
df[cols] = df[cols].astype('object')
```

```
[ ]  1 cols = ['Occupation', 'Marital_Status', 'Product_Category']
     2 df[cols] = df[cols].astype('object')
```

```
df.dtypes
```

```
1 df.dtypes
```

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

```
df.describe()
```

```
1 df.describe()
```

	User_ID	Purchase
count	5.008000e+04	50079.000000
mean	1.002552e+06	9279.490525
std	1.786666e+03	4952.962432
min	1.000001e+06	185.000000
25%	1.001016e+06	5852.000000
50%	1.002100e+06	8045.000000
75%	1.004064e+06	12033.000000
max	1.006040e+06	23958.000000

Observations

- There are no missing values in the dataset.
- Purchase amount might have outliers.

2. Determining Null and Outliers

```
# checking null values
df.isnull().sum()
```

```
1 # checking null values
2 df.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                1
Purchase                        1
dtype: int64
```

How many users are there in the dataset?

```
df['User_ID'].nunique()
```

```
1 df['User_ID'].nunique()
```

```
5426
```

How many products are there?

```
1 df['Product_ID'].nunique()
```

```
3098
```

Value_counts for the following:

- Gender
- Age
- Occupation
- City_Category
- Stay_In_Current_City_Years
- Marital_Status
- Product_Category

```
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)
```

```
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)
```

variable	value	value
Age	0-17	0.028135
	18-25	0.192332
	26-35	0.394469
	36-45	0.196985
	46-50	0.080112
	51-55	0.069988
	55+	0.037979
City_Category	A	0.280571
	B	0.414876
	C	0.304553
Gender	F	0.243850
	M	0.756150
Marital_Status	0	0.590355
	1	0.409645

Observations

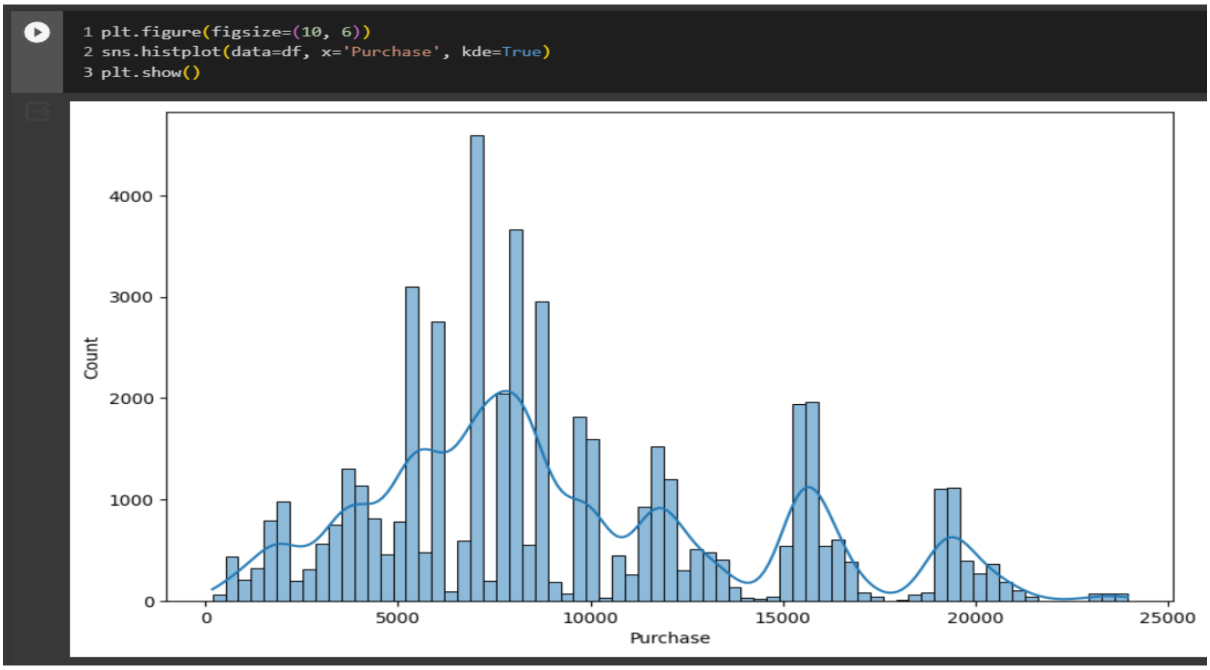
- ~ 80% of the users are between the age 18-50 (40%: 26-35, 18%: 18-25, 20%: 36-45)
- 75% of the users are Male and 25% are Female
- 60% Single, 40% Married
- 35% Staying in the city from 1 year, 18% from 2 years, 17% from 3 years
- Total of 20 product categories are there
- There are 20 diffenent types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

3. Visual Analysis - Univariate & Bivariate

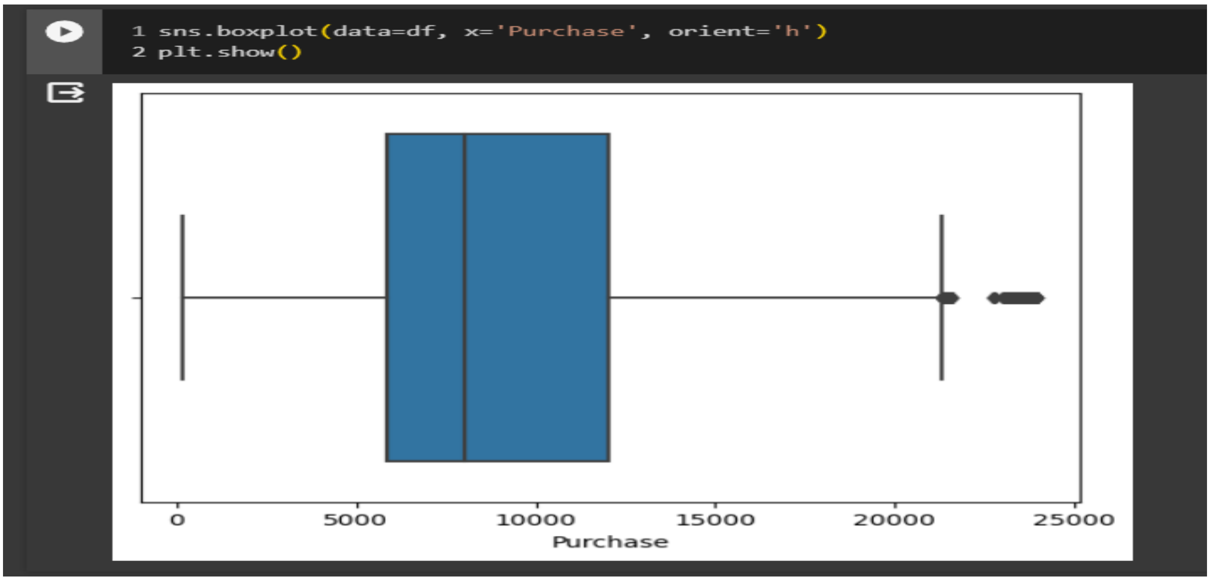
Univariate Analysis

Understanding the distribution of data and detecting outlies for continuous variables

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



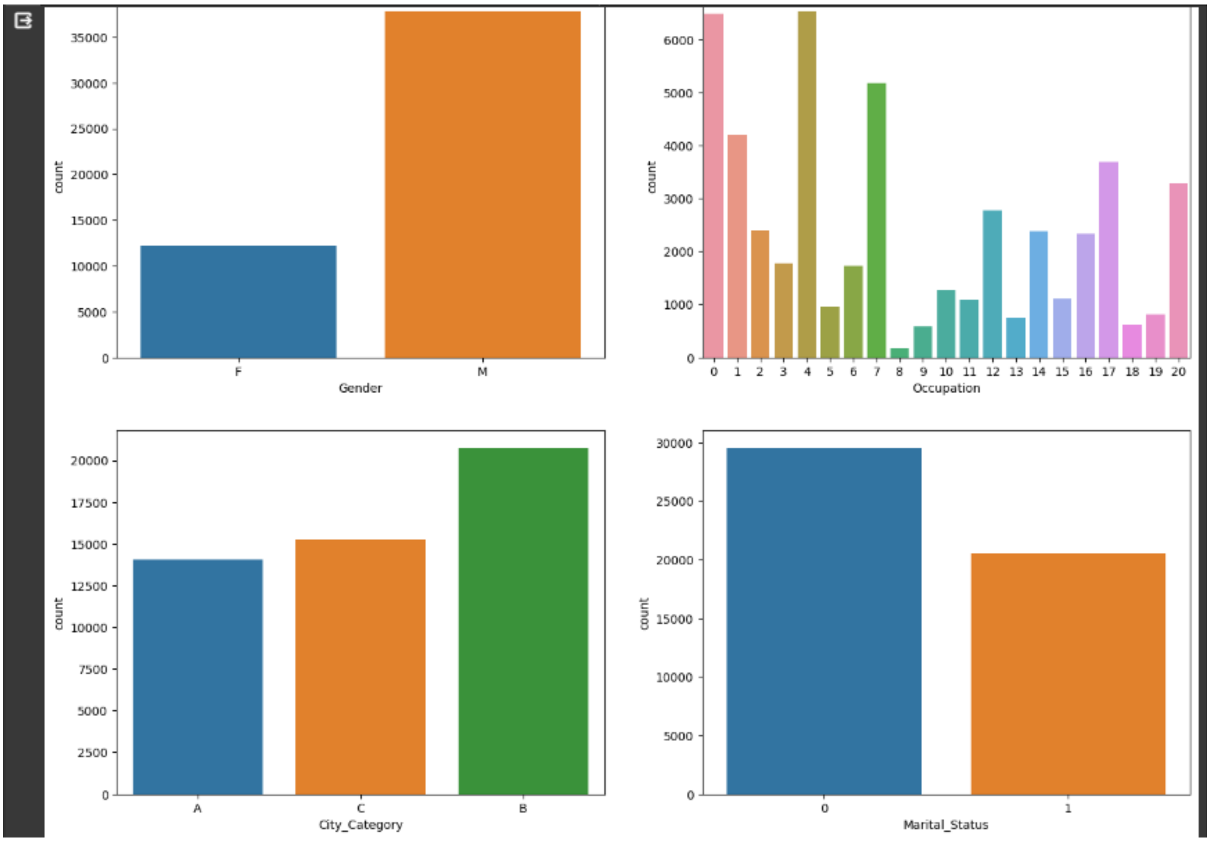

```
sns.boxplot(data=df, x='Purchase', orient='h')
plt.show()
```

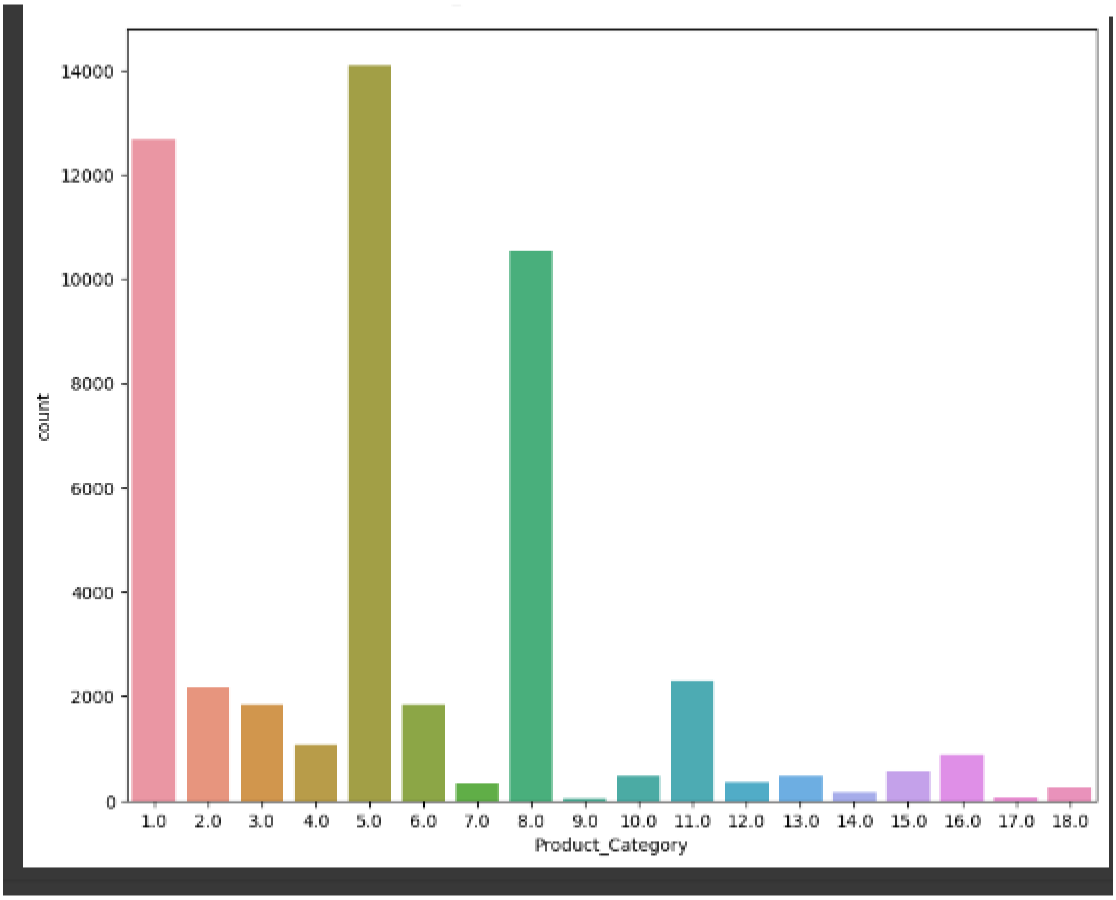


Observation

Purchase is having outliers

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

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

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



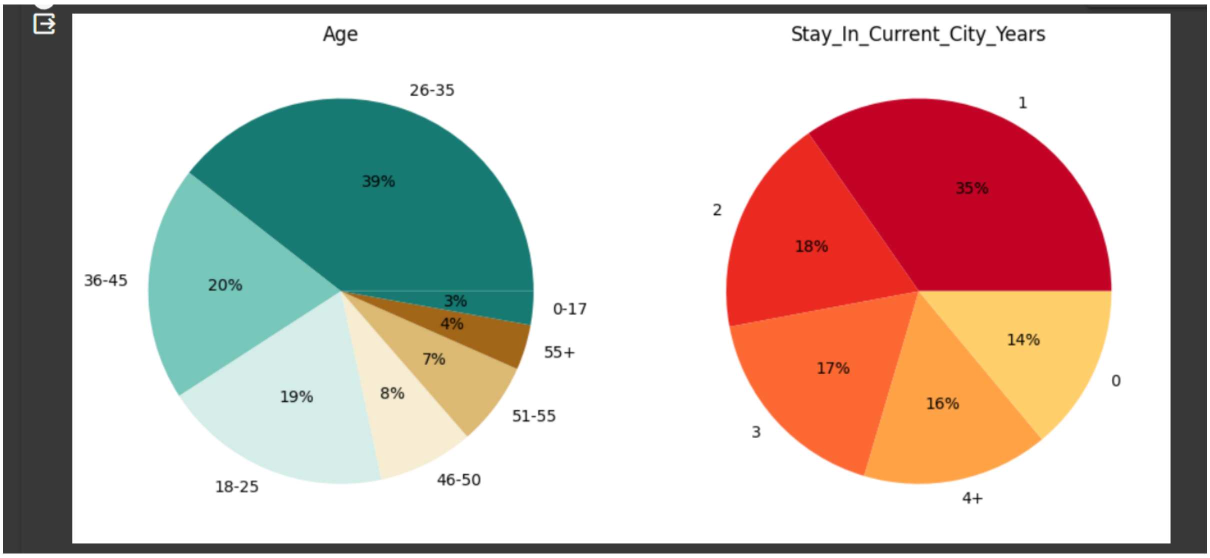


Observations

- 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.

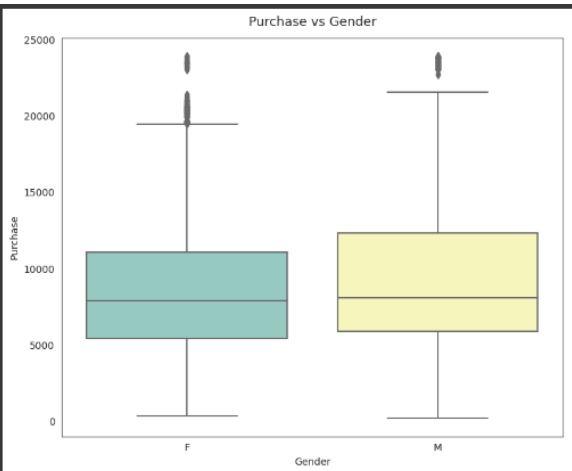
```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(12, 8))

data = df['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['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)
```

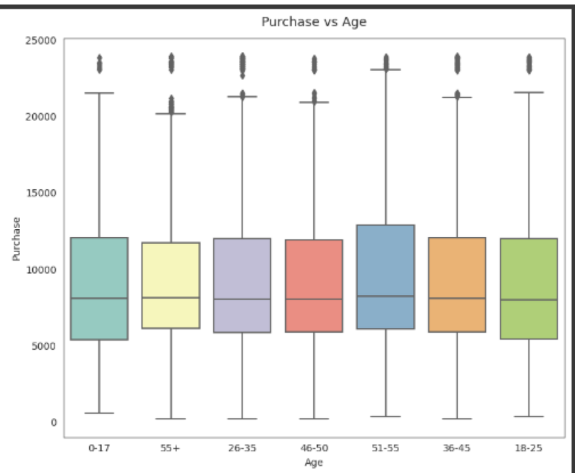
```
axs[1].set_title("Stay_In_Current_City_Years")


plt.show()
```

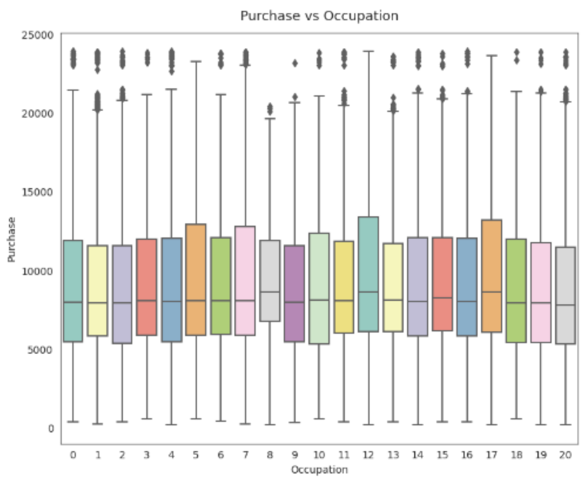


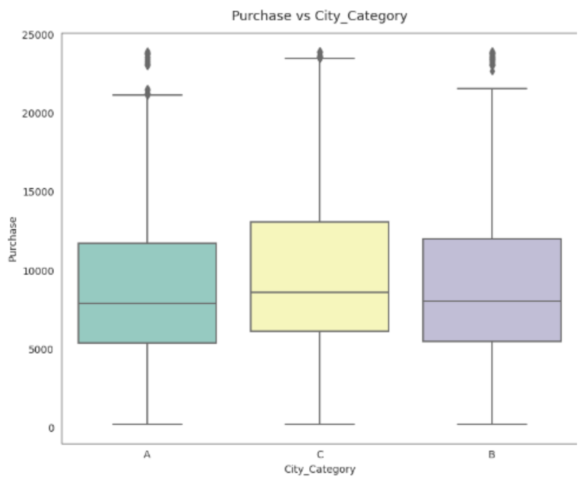
Bi-variate Analysis

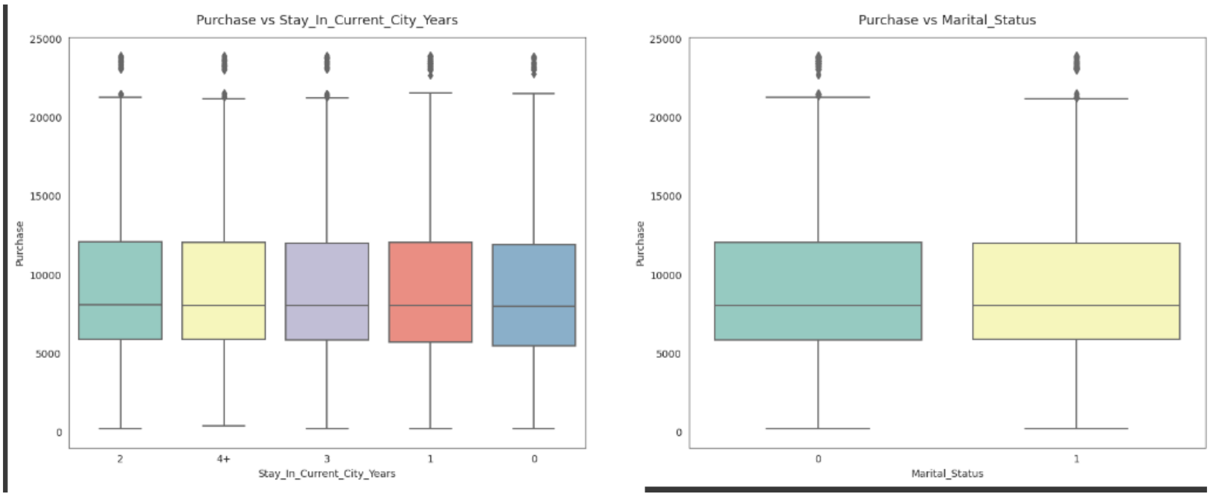
```
attrs = ['Gender', 'Age', 'Occupation', 'City_Category',
'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category']
sns.set_style("white")

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, y='Purchase', x=attrs[count], ax=axs[row,
col], palette='Set3')
        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, y='Purchase', x=attrs[-1], palette='Set3')
plt.show()
```

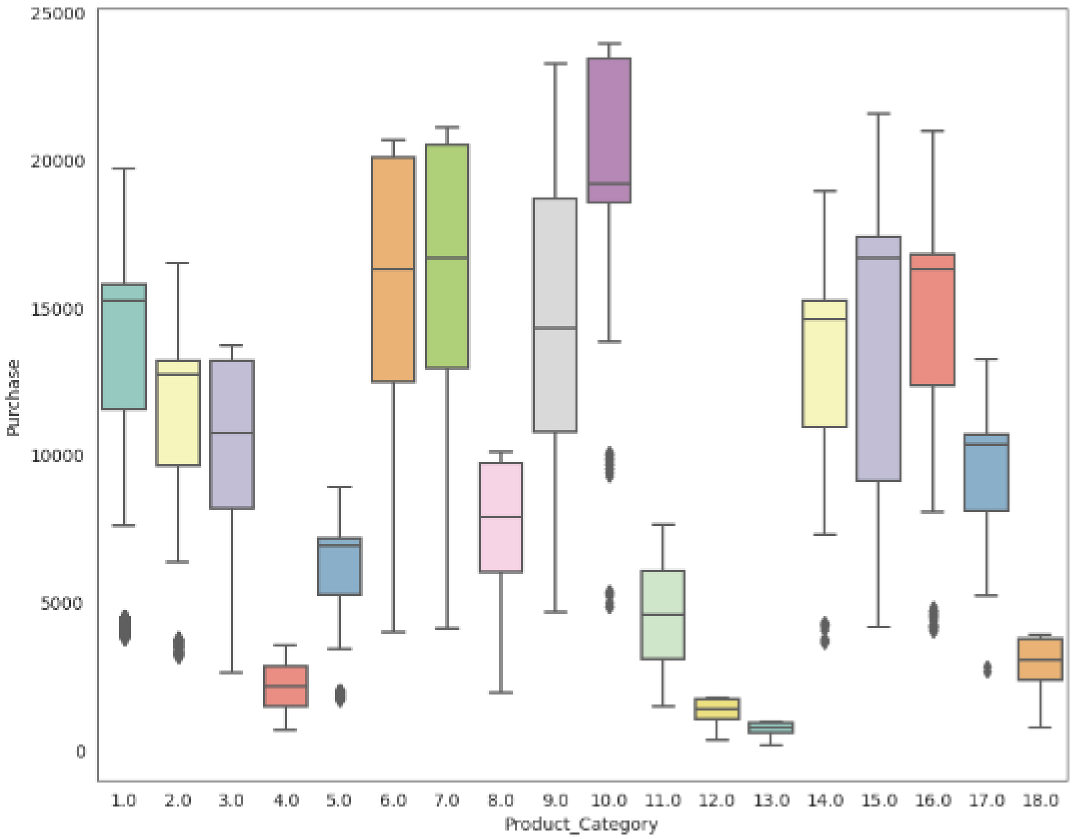

Purchase vs Gender
Purchase
Gender
F
M


Purchase vs Age
Purchase
Age
0-17
55+
26-35
46-50
51-55
36-45
18-25


Purchase vs Occupation
Purchase
Occupation


Purchase vs City_Category
Purchase
City_Category
A
C
B

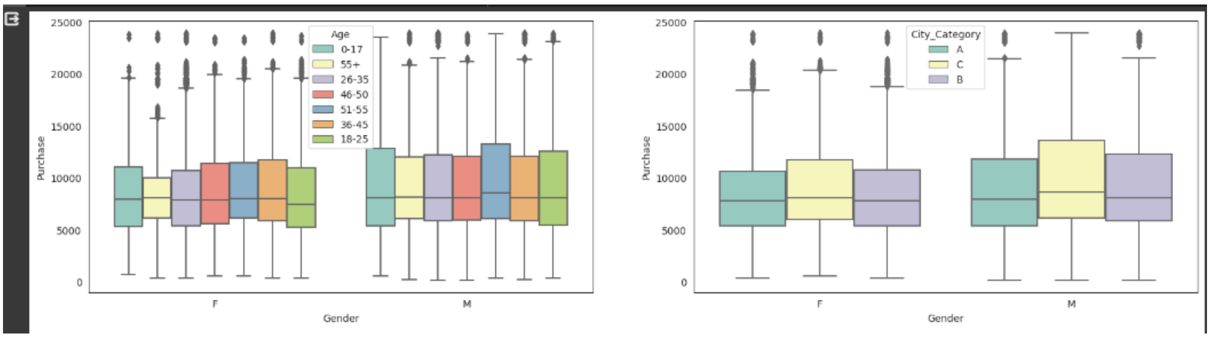


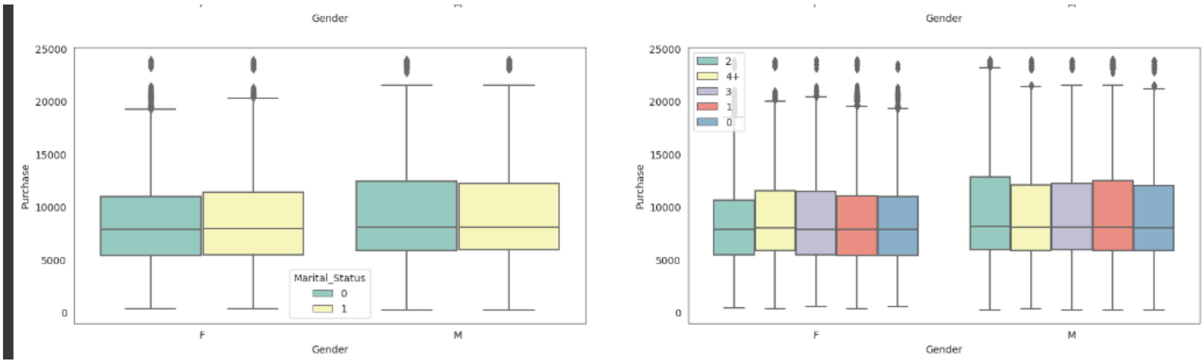


Multivariate Analysis

```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20, 6))
fig.subplots_adjust(top=1.5)
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Age',
palette='Set3', ax=axs[0,0])
sns.boxplot(data=df, y='Purchase', x='Gender', hue='City_Category',
palette='Set3', ax=axs[0,1])

sns.boxplot(data=df, y='Purchase', x='Gender', hue='Marital_Status',
palette='Set3', ax=axs[1,0])
sns.boxplot(data=df, y='Purchase', x='Gender',
hue='Stay_In_Current_City_Years', palette='Set3', ax=axs[1,1])
```

```
axs[1,1].legend(loc='upper left')

plt.show()
```





4. CLT

Confidence Interval by Gender

Now using the Central Limit Theorem for the population:

- **Average amount spend by** male **customers is** 9,26,341.86
- **Average amount spend by** female **customers is** 7,11,704.09

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)**

Confidence Interval by Marital_Status

- **Married** confidence interval of means: **(806668.83, 880384.76)**
- **Unmarried** confidence interval of means: **(848741.18, 912410.38)**

Confidence Interval by Age

- 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)**

5. Confidence Interval of avg male and female spends

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

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

	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	3.0	8370.0
1	1000001	P00248942	F	0-17	10	A	2	0	1.0	15200.0
2	1000001	P00087842	F	0-17	10	A	2	0	12.0	1422.0
3	1000001	P00085442	F	0-17	10	A	2	0	12.0	1057.0
4	1000002	P00285442	M	55+	16	C	4+	0	8.0	7969.0
5	1000003	P00193542	M	26-35	15	A	3	0	1.0	15227.0
6	1000004	P00184942	M	46-50	7	B	2	1	1.0	19215.0
7	1000004	P00346142	M	46-50	7	B	2	1	1.0	15854.0
8	1000004	P0097242	M	46-50	7	B	2	1	1.0	15686.0
9	1000005	P00274942	M	26-35	20	A	1	1	8.0	7871.0

Average amount spend per customer for Male and Female

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

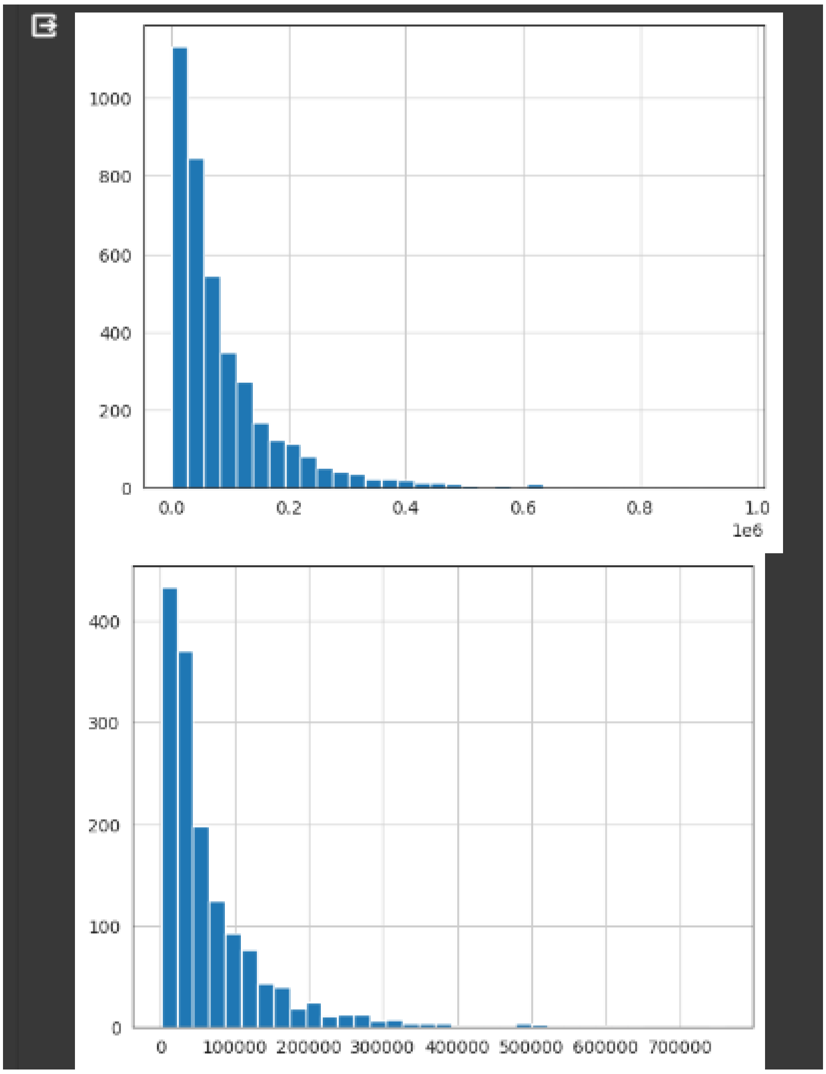
	User_ID	Gender	Purchase
0	1000001	F	38891.0
1	1000002	M	37417.0
2	1000003	M	49947.0
3	1000004	M	66607.0
4	1000005	M	50684.0
...	...	...	...
5421	1006035	F	42357.0
5422	1006036	F	196339.0
5423	1006037	F	86597.0
5424	1006039	F	50364.0
5425	1006040	M	95780.0

5426 rows × 3 columns

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

```
M    3916
F    1510
Name: Gender, 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: 91458.66
Average amount spend by Female customers: 70566.55

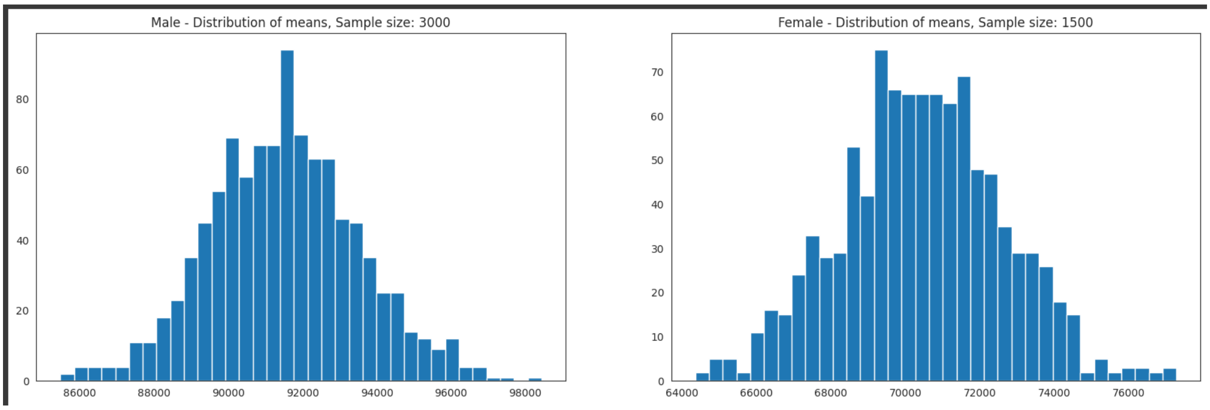
Observation

1. Male customers spend more money than female customers

```
male_df = amt_df[amt_df['Gender']=='M']
female_df = amt_df[amt_df['Gender']=='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.mean(male_means)))
print("Population mean - Mean of sample means of amount spend for
Female: {:.2f}".format(np.mean(female_means)))
```

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

Population mean - Mean of sample means of amount spend for Male: 91506.88
Population mean - Mean of sample means of amount spend for Female: 70439.17

Male - Sample mean: 91458.66 Sample std: 108605.26
Female - Sample mean: 70566.55 Sample std: 85768.24

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_upper_lim))
print("Female confidence interval of means: ({:.2f},
{:.2f})".format(female_lower_lim, female_upper_lim))
```

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)

Confidence Interval of married & unmarried and age

Doing the same activity for married vs unmarried

```
amt_df
```

```
1 amt_df
```

	User_ID	Gender	Purchase
0	1000001	F	38891.0
1	1000002	M	37417.0
2	1000003	M	49947.0
3	1000004	M	66607.0
4	1000005	M	50684.0
...	...	...	...
5421	1006035	F	42357.0
5422	1006036	F	196339.0
5423	1006037	F	86597.0
5424	1006039	F	50364.0
5425	1006040	M	95780.0

5426 rows × 3 columns

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

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

	User_ID	Marital_Status	Purchase
0	1000001	0	38891.0
1	1000002	0	37417.0
2	1000003	0	49947.0
3	1000004	1	66607.0
4	1000005	1	50684.0
...	...	...	...
5421	1006035	0	42357.0
5422	1006036	1	196339.0
5423	1006037	0	86597.0
5424	1006039	1	50364.0
5425	1006040	0	95780.0

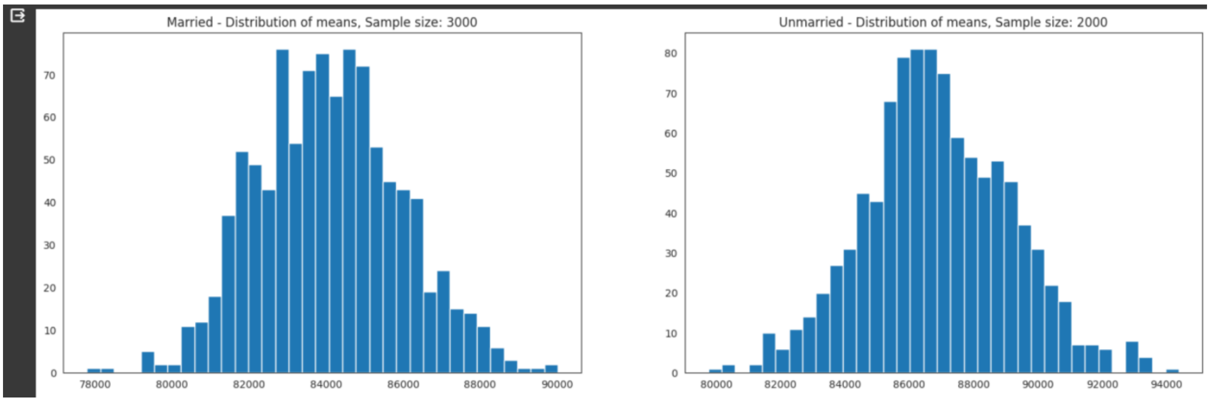
5426 rows × 3 columns

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

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

```
0    3158
1    2268
Name: Marital_Status, 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)['Purchase'].mean()
    unmarid_mean =
amt_df[amt_df['Marital_Status']==0].sample(unmarid_sample_size,
replace=True)['Purchase'].mean()

    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(np.mean(marid_means)))
print("Population mean - Mean of sample means of amount spend for
Unmarried: {:.2f}".format(np.mean(unmarid_means)))

print("\nMarried - Sample mean: {:.2f} Sample std:
{:.2f}".format(amt_df[amt_df['Marital_Status']==1]['Purchase'].mean(),
amt_df[amt_df['Marital_Status']==1]['Purchase'].std()))
print("Unmarried - Sample mean: {:.2f} Sample std:
{:.2f}".format(amt_df[amt_df['Marital_Status']==0]['Purchase'].mean(),
amt_df[amt_df['Marital_Status']==0]['Purchase'].std()))
```



Population mean - Mean of sample means of amount spend for Married: 84078.49
Population mean - Mean of sample means of amount spend for Unmarried: 86881.85

Married - Sample mean: 84074.26 Sample std: 102788.72
Unmarried - Sample mean: 86772.38 Sample std: 103459.85

```
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, upper_lim))
```

Married confidence interval of means: (79843.88, 88304.65)
Unmarried confidence interval of means: (83163.92, 90380.84)

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 = df.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

5426 rows × 3 columns

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


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


```
26-35    1907
36-45    1083
18-25     988
46-50     473
51-55     441
55+       330
0-17      204
Name: Age, dtype: int64
```

```python
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)['Purchase'].mean()
        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(val, lower_lim, upper_lim))
```

```
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(val, lower_lim, upper_lim))
```

```
For age 26-35 --> confidence interval of means: (90905.53, 100952.80)
For age 36-45 --> confidence interval of means: (78848.39, 91057.85)
For age 18-25 --> confidence interval of means: (82824.93, 96037.87)
For age 46-50 --> confidence interval of means: (68722.63, 86800.89)
For age 51-55 --> confidence interval of means: (68729.40, 85291.32)
For age 55+ --> confidence interval of means: (47446.18, 60698.51)
For age 0-17 --> confidence interval of means: (52027.53, 73673.20)
```

Insights

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

- 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.

- **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:

- **Average amount spend by** male **customers is** 9,26,341.86
- **Average amount spend by** female **customers is** 7,11,704.09

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)**

Confidence Interval by Marital_Status

- **Married** confidence interval of means: **(806668.83, 880384.76)**
- **Unmarried** confidence interval of means: **(848741.18, 912410.38)**

Confidence Interval by Age

- 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)**

6. Recommendations

1. Men spent more money than women, So company should focus on retaining the male customers and getting more male customers.
2. 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.
3. Unmarried customers spend more money than married customers, So company should focus on acquisition of Unmarried customers.
4. 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
5. 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.