Business Case: Walmart - Confidence Interval and CLT

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. 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).

In [106…

```
# importing the required packages


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

Importing the dataset and do usual data analysis steps like checking the structure & characteristics of the dataset

In [107…

```
df = pd.read_csv('C:/Users/pshashank3/Desktop/Data Science/Scaler/Datasets/Projects/walmart/walmart_data.csv')
df.head()
```

Out[107…

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

So we have 10 columns and 550068 rows of datapoints of sold information of treadmill to customer shown below

In [108…

```
df.shape
```

Out[108…

```
(550068, 10)
```

So the datatype of all columns is shown below

In [109…

```
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
```

In [110…

```
df.describe()
```

Out[110…

	User_ID	Occupation	Marital_Status	Product_Category	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9263.968713
std	1.727592e+03	6.522660	0.491770	3.936211	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	12.000000
25%	1.001516e+06	2.000000	0.000000	1.000000	5823.000000
50%	1.003077e+06	7.000000	0.000000	5.000000	8047.000000

	User_ID	Occupation	Marital_Status	Product_Category	Purchase
75%	1.004478e+06	14.000000	1.000000	8.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	23961.000000

In [111…

```
# These are the columns which are autodetected as int but they are categorical columns in reference
cols = ['Occupation', 'Marital_Status', 'Product_Category']
df[cols] = df[cols].astype('object')
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
```

In [ ]:

Univariante Analysis

In [112…

```
df.describe()
```

Out[112…

	User_ID	Purchase
count	5.500680e+05	550068.000000

	User_ID	Purchase
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

In [113…

```
df.astype("category").describe(include = 'all').T
```

Out[113…

	count	unique	top	freq
User_ID	550068	5891	1001680	1026
Product_ID	550068	3631	P00265242	1880
Gender	550068	2	M	414259
Age	550068	7	26-35	219587
Occupation	550068	21	4	72308
City_Category	550068	3	B	231173
Stay_In_Current_City_Years	550068	5	1	193821
Marital_Status	550068	2	0	324731
Product_Category	550068	20	5	150933
Purchase	550068	18105	7011	191

Observations

- There are no missing values in the dataset as count of each columns are matching with total row of the dataset.
- Purchase amount might have outliers as mean and 50%(median) differ by large gap.

- Also some vital informations like 5891 users, 3631 products, 20 product categories comprising the data

In [ ]:

```

```

Seeing distinct valus with count of each categorical values

In [114…

```
categorical_cols = ['Gender', 'Age', 'Occupation', 'City_Category', 'Stay_In_Current_City_Years',
'Marital_Status', 'Product_Category']
a = np.round(df[categorical_cols].melt().groupby(['variable', 'value'])[['value']].count()/df.shape[0],3)
a.rename(columns = {'value': 'freq'}, inplace = True)
a.reset_index()
```

Out[114…

	variable	value	freq
0	Age	0-17	0.027
1	Age	18-25	0.181
2	Age	26-35	0.399
3	Age	36-45	0.200
4	Age	46-50	0.083
5	Age	51-55	0.070
6	Age	55+	0.039
7	City_Category	A	0.269
8	City_Category	B	0.420
9	City_Category	C	0.311
10	Gender	F	0.247
11	Gender	M	0.753
12	Marital_Status	0	0.590
13	Marital_Status	1	0.410

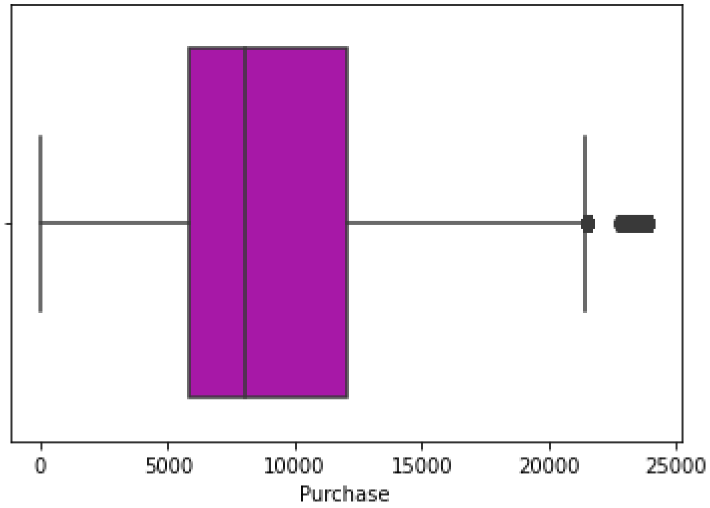
	variable	value	freq
14	Occupation	0	0.127
15	Occupation	1	0.086
16	Occupation	2	0.048
17	Occupation	3	0.032
18	Occupation	4	0.131
19	Occupation	5	0.022
20	Occupation	6	0.037
21	Occupation	7	0.108
22	Occupation	8	0.003
23	Occupation	9	0.011
24	Occupation	10	0.024
25	Occupation	11	0.021
26	Occupation	12	0.057
27	Occupation	13	0.014
28	Occupation	14	0.050
29	Occupation	15	0.022
30	Occupation	16	0.046
31	Occupation	17	0.073
32	Occupation	18	0.012
33	Occupation	19	0.015
34	Occupation	20	0.061
35	Product_Category	1	0.255
36	Product_Category	2	0.043
37	Product_Category	3	0.037

	variable	value	freq
38	Product_Category	4	0.021
39	Product_Category	5	0.274
40	Product_Category	6	0.037
41	Product_Category	7	0.007
42	Product_Category	8	0.207
43	Product_Category	9	0.001
44	Product_Category	10	0.009
45	Product_Category	11	0.044
46	Product_Category	12	0.007
47	Product_Category	13	0.010
48	Product_Category	14	0.003
49	Product_Category	15	0.011
50	Product_Category	16	0.018
51	Product_Category	17	0.001
52	Product_Category	18	0.006
53	Product_Category	19	0.003
54	Product_Category	20	0.005
55	Stay_In_Current_City_Years	0	0.135
56	Stay_In_Current_City_Years	1	0.352
57	Stay_In_Current_City_Years	2	0.185
58	Stay_In_Current_City_Years	3	0.173
59	Stay_In_Current_City_Years	4+	0.154

In [115…

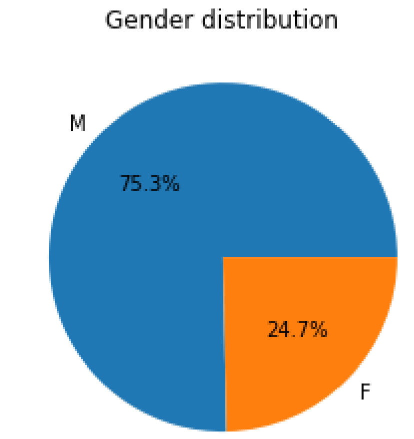
```
sns.boxplot(df['Purchase'],color='m')
```

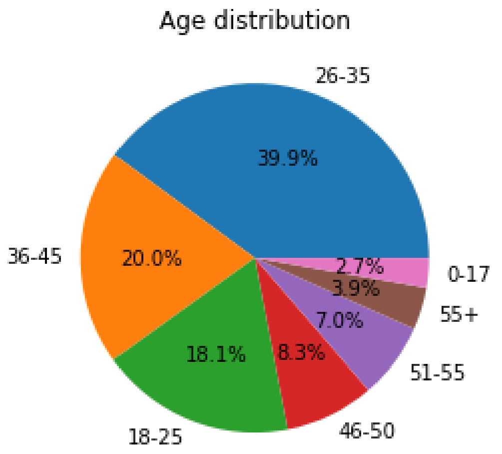
Out[115… <AxesSubplot:xlabel='Purchase'>

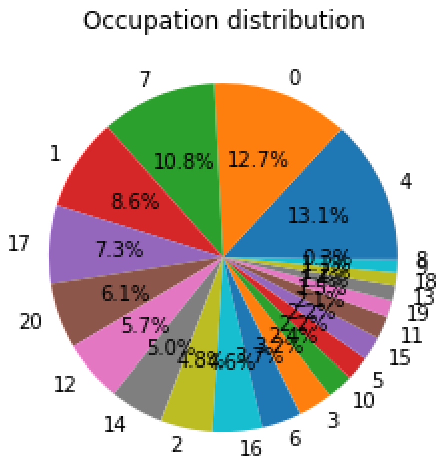


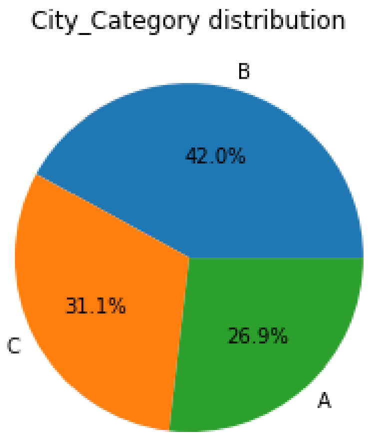
In [116…

```
for col in categorical_cols:
    data = df[col].value_counts().values.tolist()
    lbls = df[col].value_counts().index.tolist()
    plt.pie(data, labels = lbls, autopct='%1.1f%%')
    plt.title(col + ' distribution')
    plt.show()
```

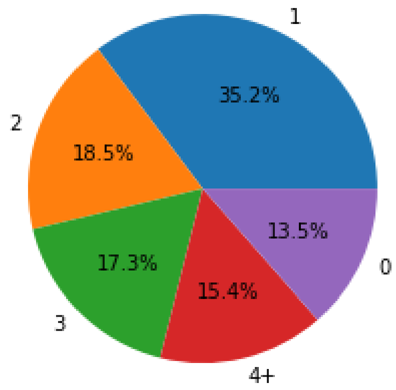

Gender distribution
M
75.3%
24.7%
F


Age distribution
26-35
39.9%
36-45
20.0%
2.7%
0-17
3.9%
55+
7.0%
18.1%
8.3%
51-55
46-50
18-25

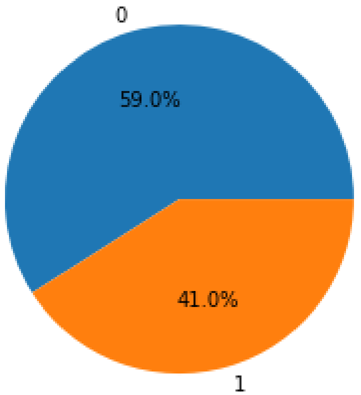

Occupation distribution
4
13.1%
0
12.7%
7
10.8%
1
8.6%
17
7.3%
20
6.1%
12
5.7%
14
5.0%
2
4.8%
16
4.6%
6
3
10
5
15
11
19
13
18
9
8

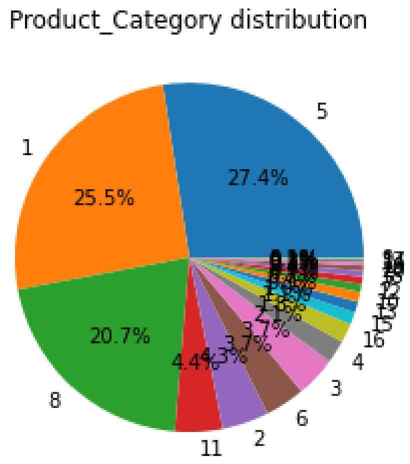

City_Category distribution
B
42.0%
C
31.1%
A
26.9%

Stay_In_Current_City_Years distribution



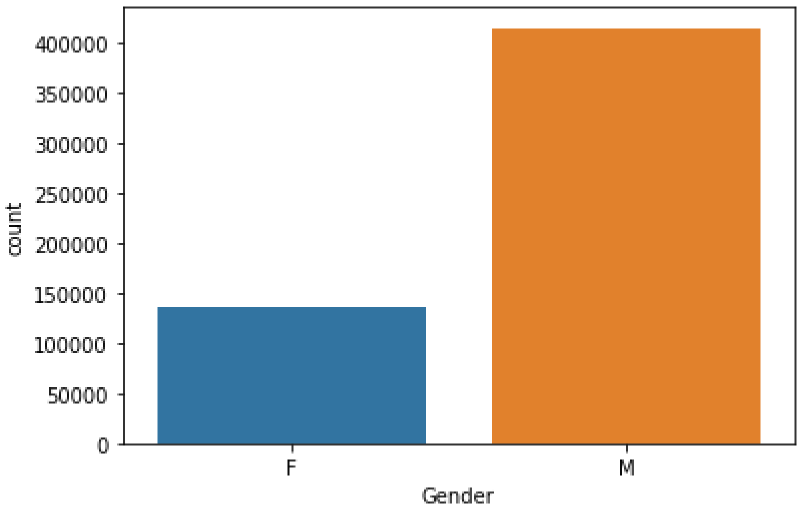
Marital_Status distribution

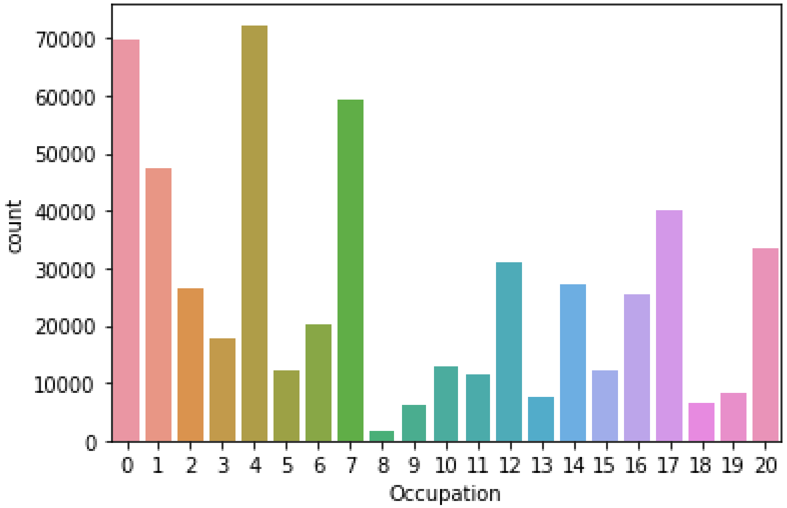


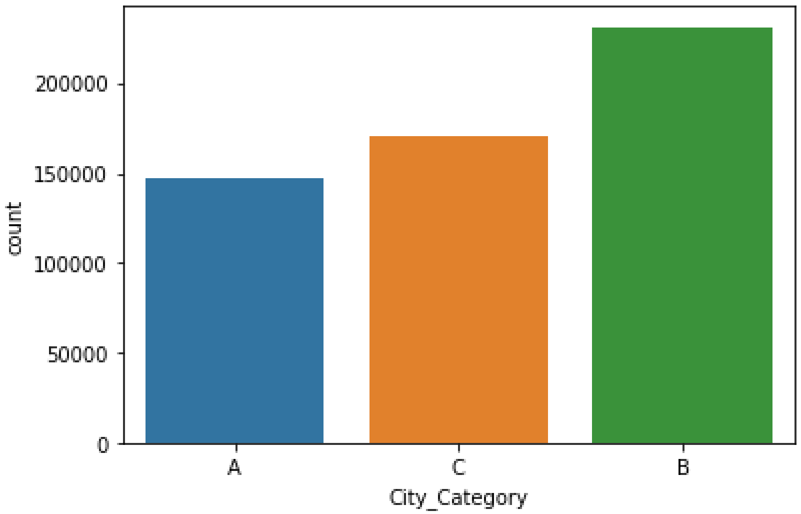


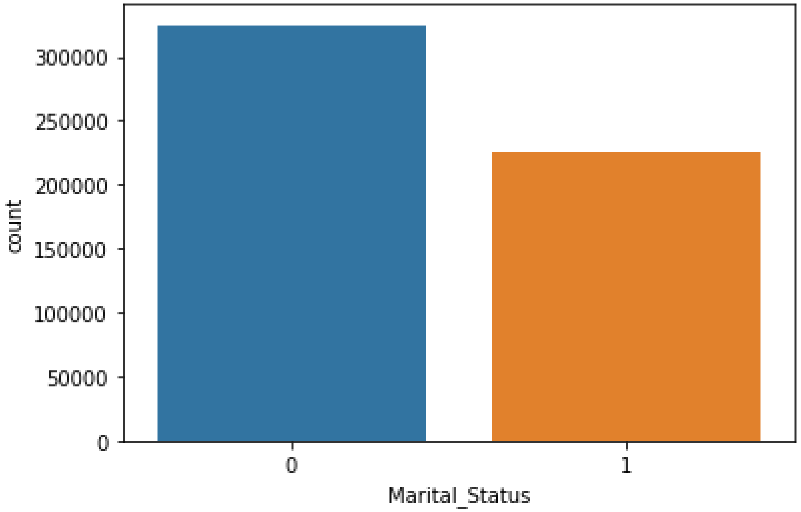
In [117…

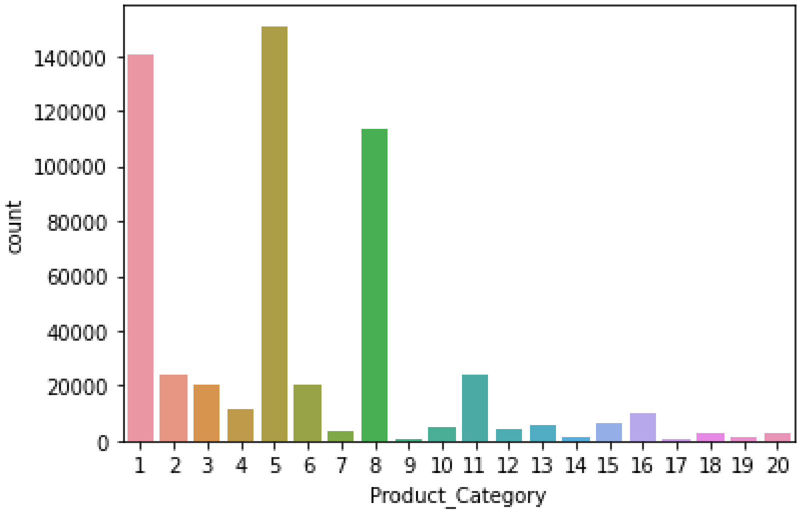
```
for val in ['Gender', 'Occupation', 'City_Category','Marital_Status', 'Product_Category']:
#     plt.figure(figsize=(10, 8))
    sns.countplot(data=df, x=val)
    plt.show()
```




70000
60000
50000
40000
30000
20000
10000
0
count
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Occupation


200000
150000
100000
50000
0
count
A
C
B
City_Category





Observations

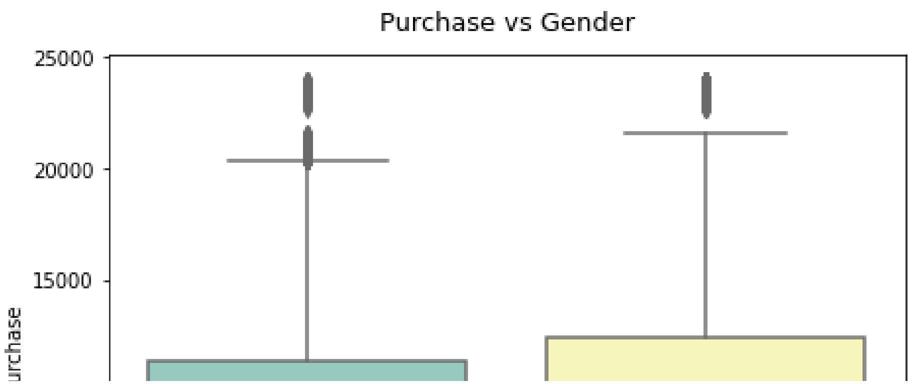
- 75% of the users are `Male` and 25% are `Female`
- 60% of the users are `Male` and 40% are `Female`
- There are 20 different types of `Occupation` and `Product_Category`
- More users belong to `City_Category - B`
- `Product_Category - 1, 5, 8, 11, & 2` are highest purchased category.

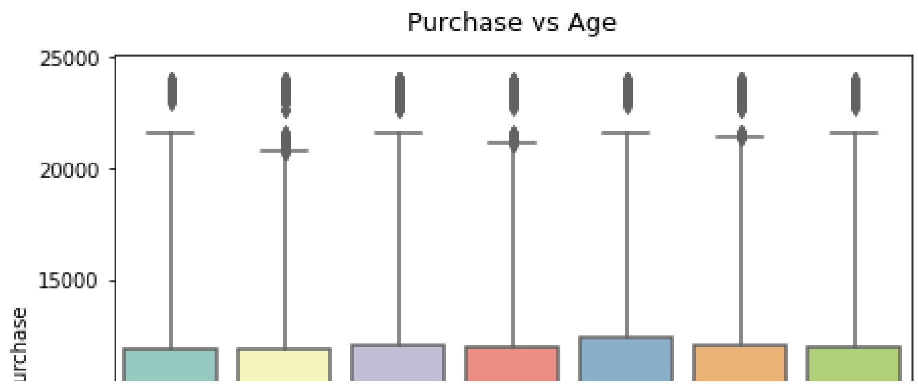
- People aged `26-35` , `36-45` and '18-25' likely to purchase a lot

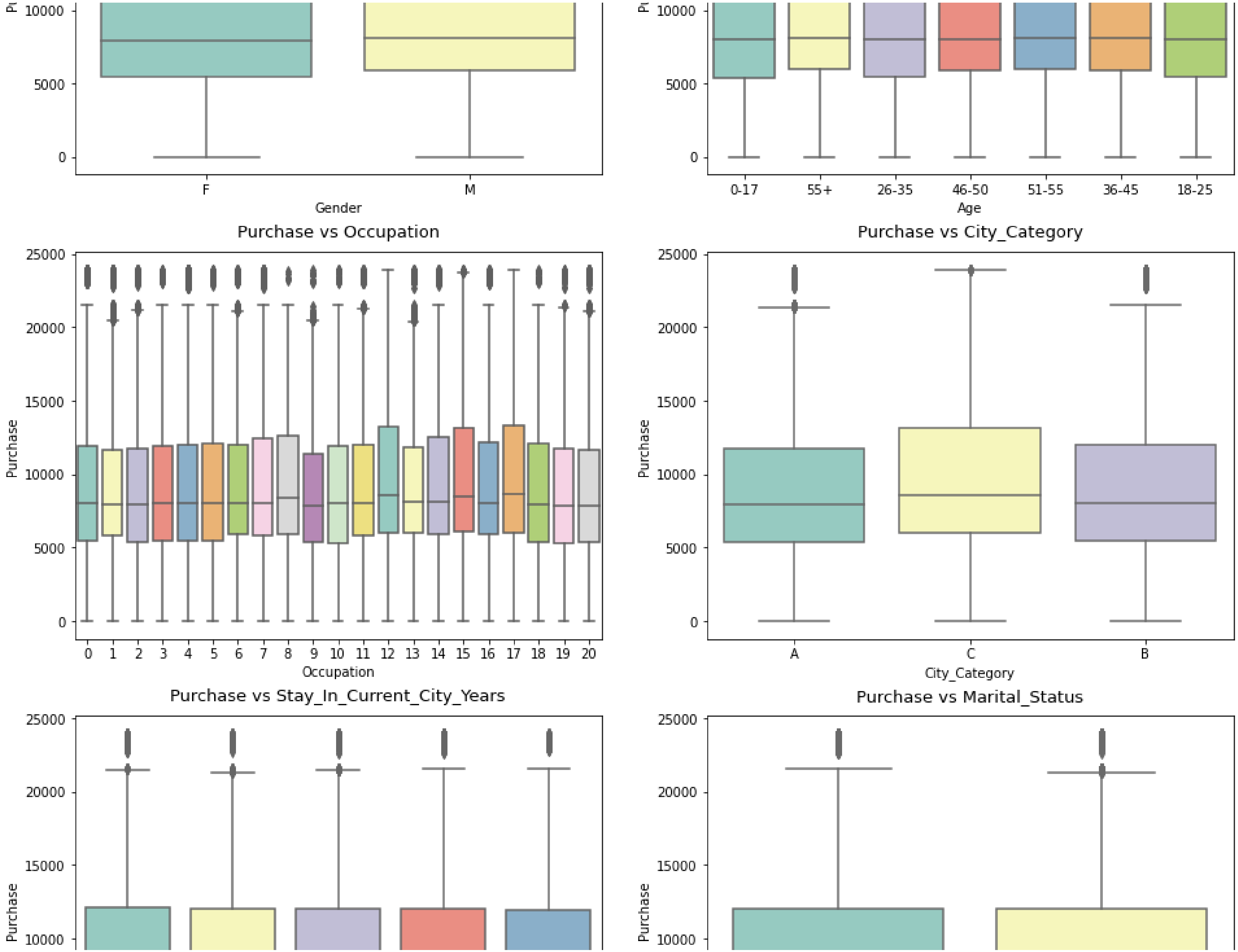
Bi-variate Analysis

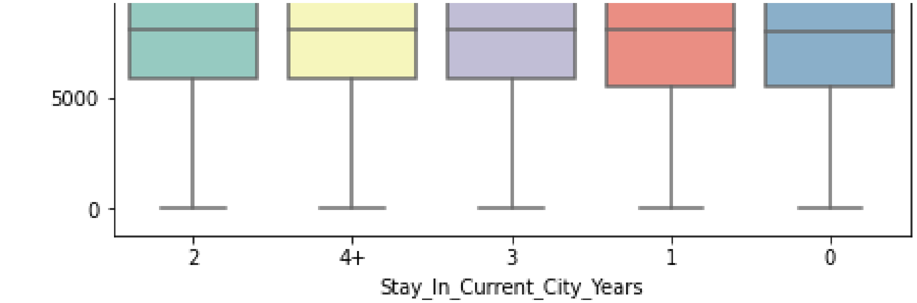
In [118…

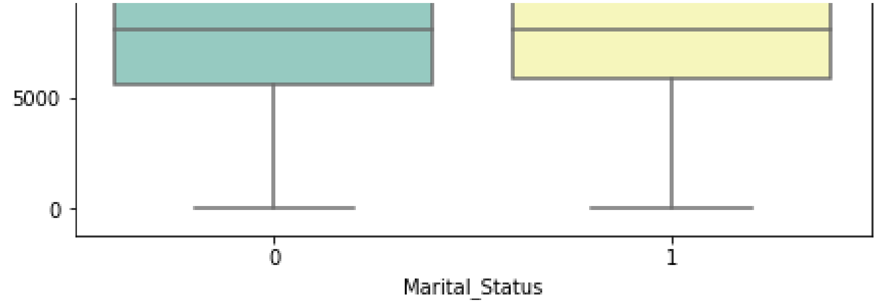
```
attrs = ['Gender', 'Age', 'Occupation', 'City_Category', 'Stay_In_Current_City_Years', 'Marital_Status',
'Product_Category']
# sns.set_style("white")

fig, axs = plt.subplots(nrows=3, ncols=2, figsize=(16, 12))
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()
```

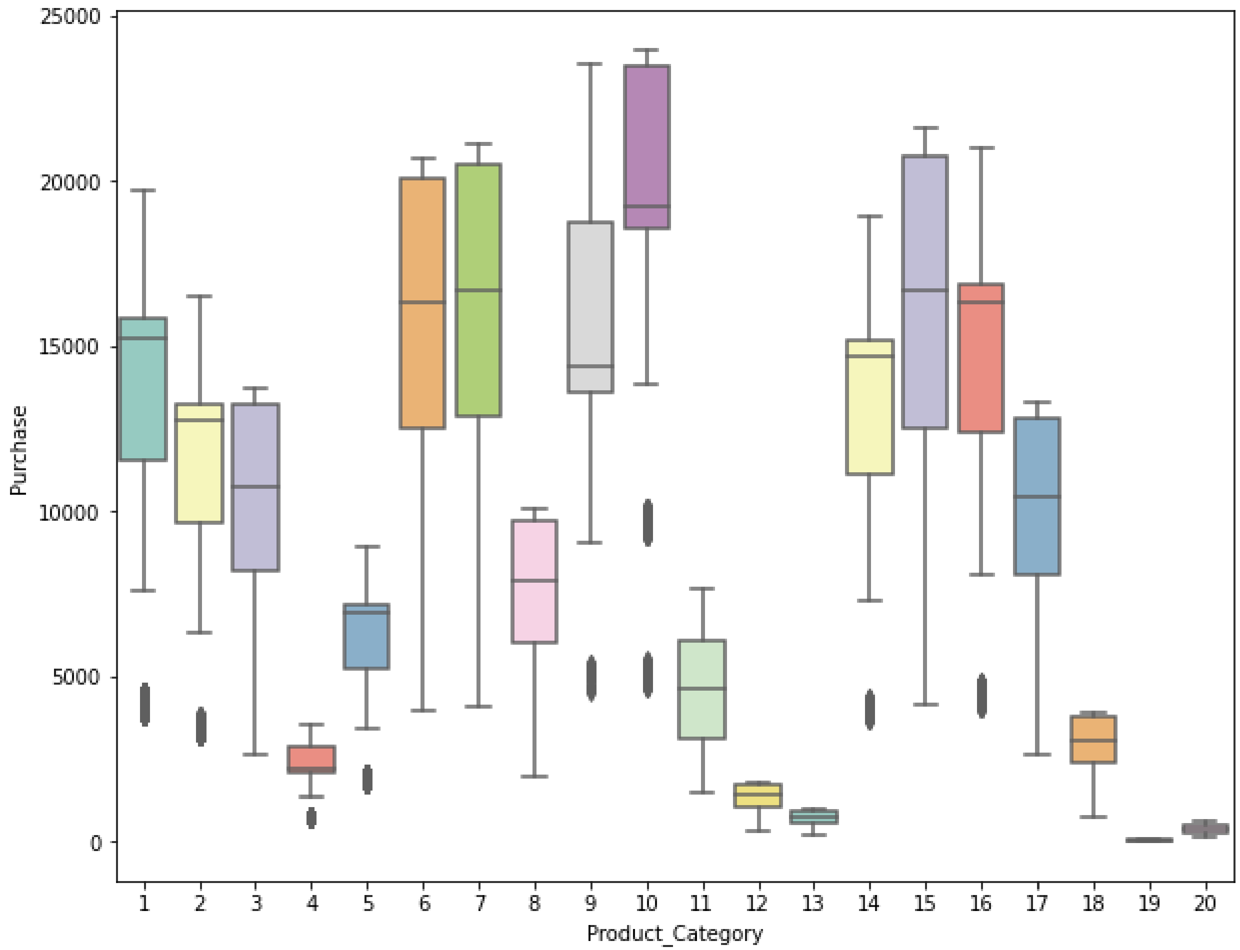





Gender
F
M
Age
0-17
55+
26-35
46-50
51-55
36-45
18-25
Purchase vs Occupation
Purchase
Occupation
Purchase vs City_Category
Purchase
City_Category
A
C
B
Purchase vs Stay_In_Current_City_Years
Purchase
Purchase vs Marital_Status
Purchase



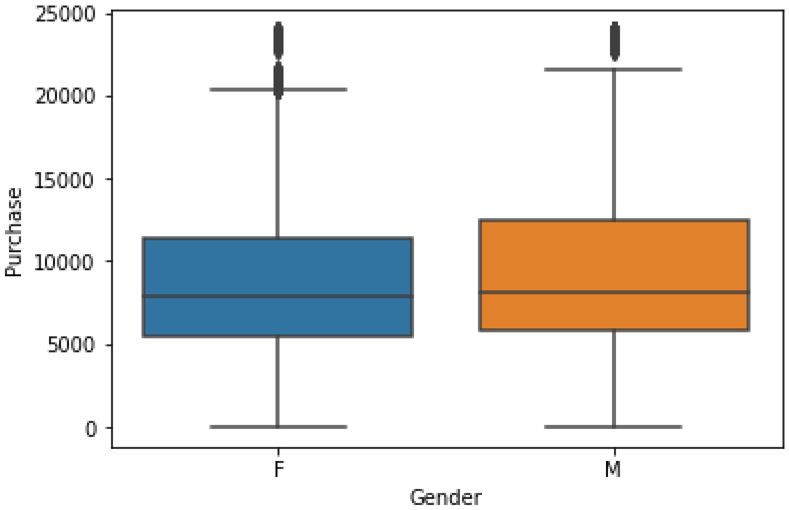


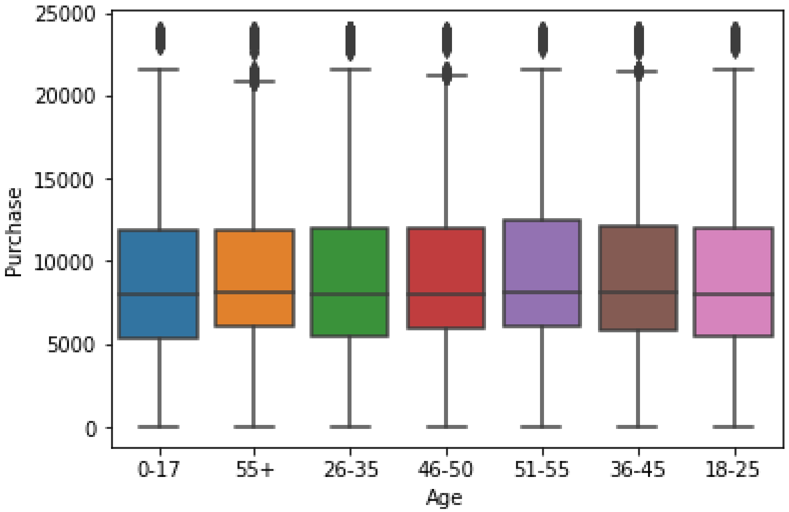


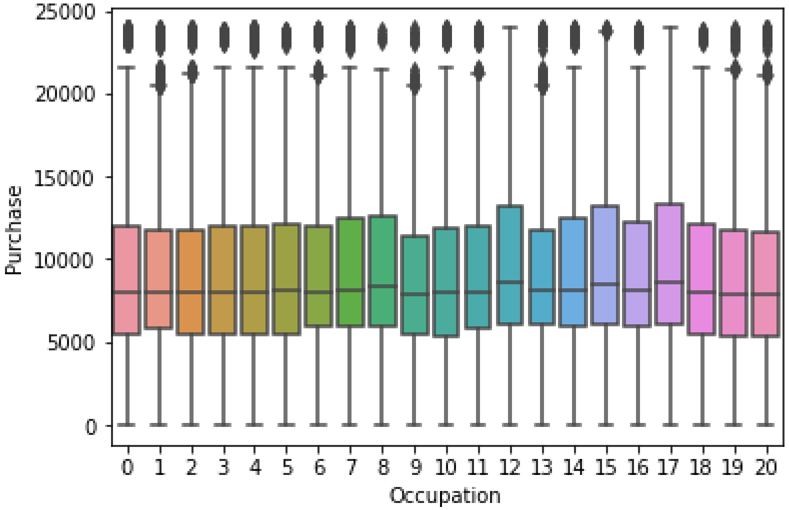
In [119…

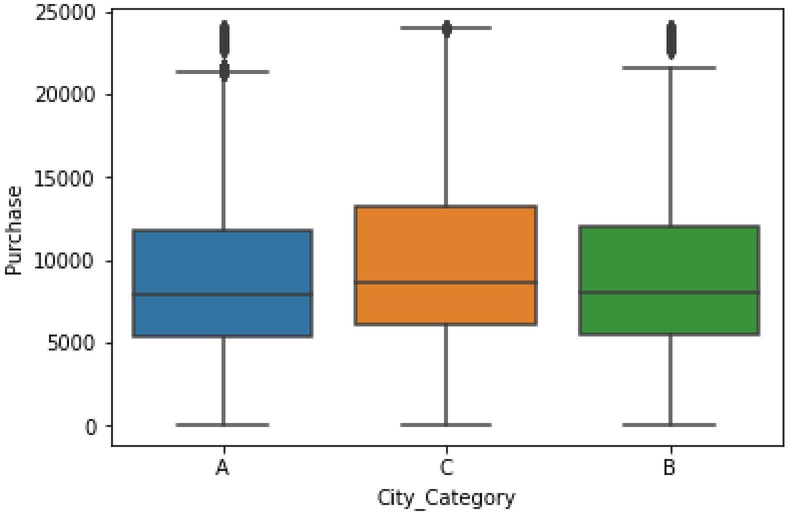
```
cols = ['Gender', 'Age', 'Occupation', 'City_Category', 'Stay_In_Current_City_Years', 'Marital_Status',
'Product_Category']
```

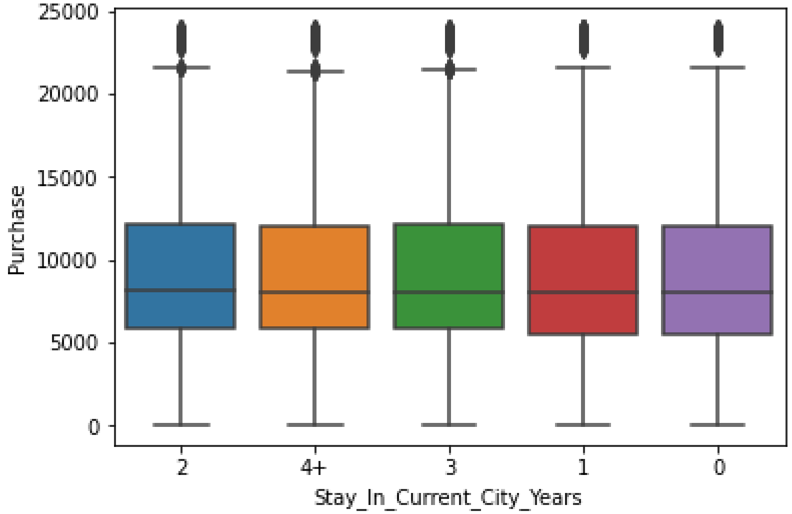
```python
# sns.set_style("white")

for col in cols:
    sns.boxplot(data=df, y='Purchase', x=col)
    plt.show()
```

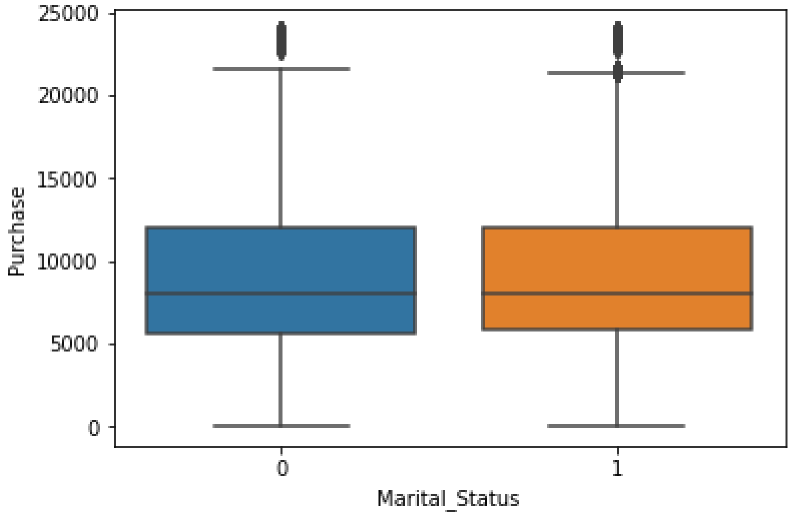


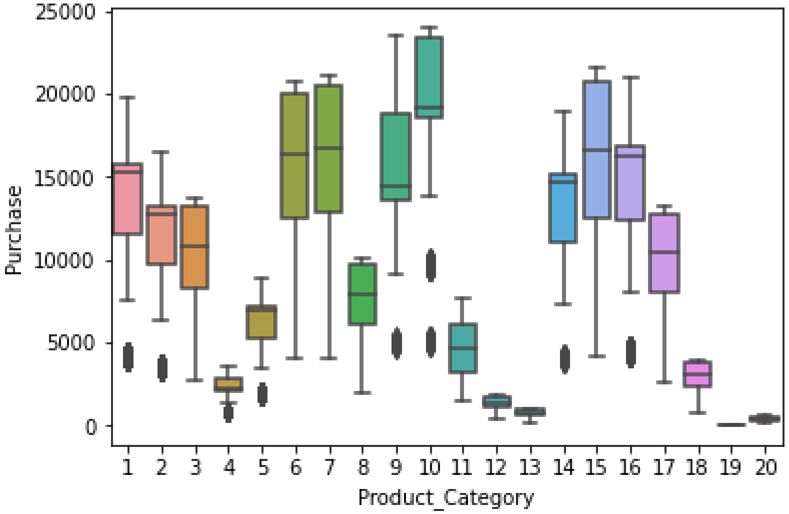



25000
20000
15000
10000
5000
0
Purchase
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Occupation


25000
20000
15000
10000
5000
0
Purchase
A
C
B
City_Category


25000
20000
15000
10000
5000
0
Purchase
2
4+
3
1
0
Stay_In_Current_City_Years

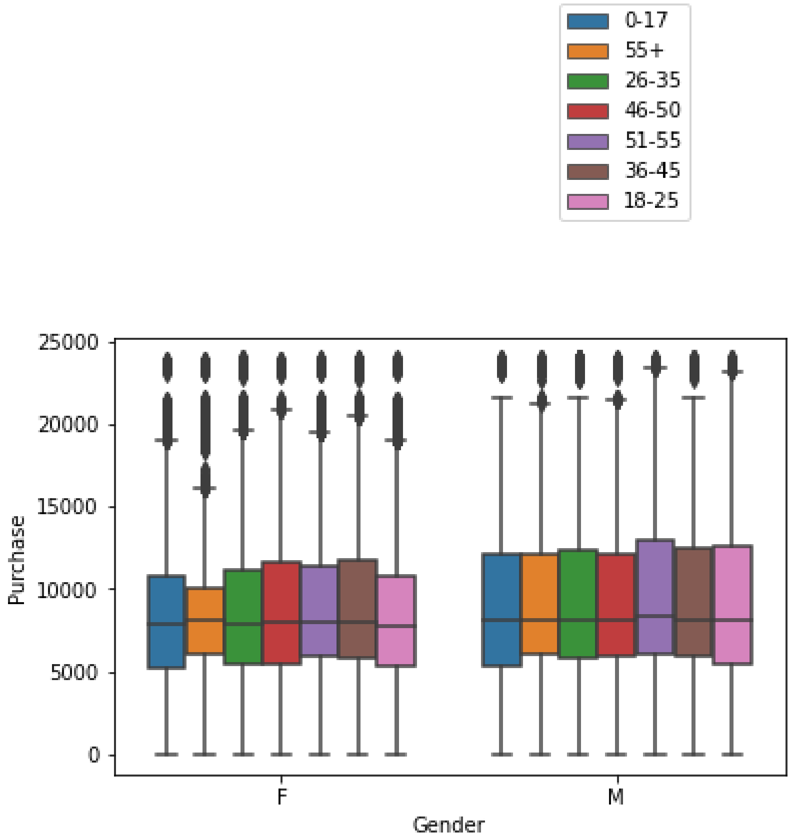

25000
20000
15000
10000
5000
0
Purchase
0
1
Marital_Status

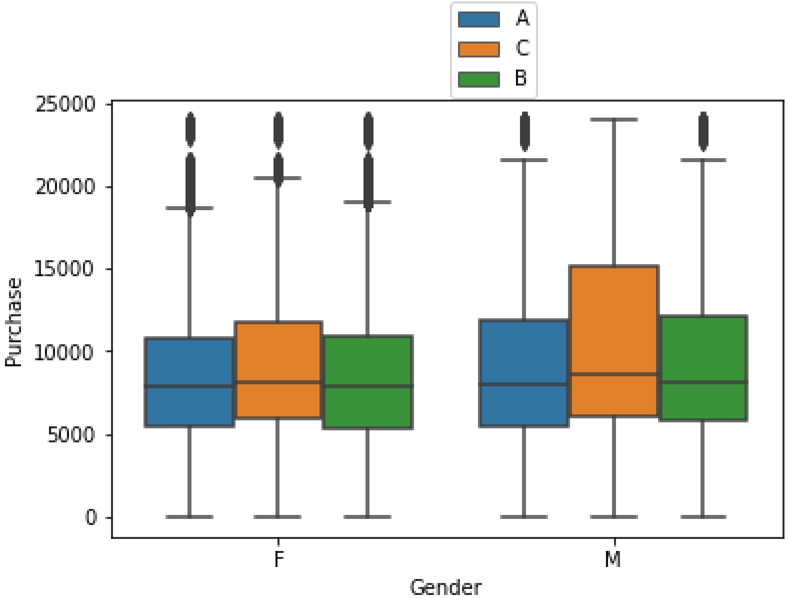


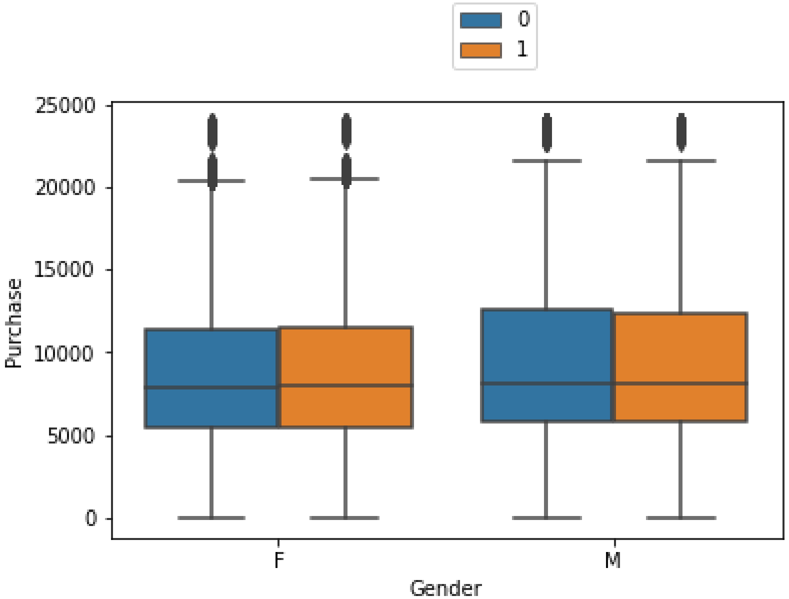
Multivariate Analysis

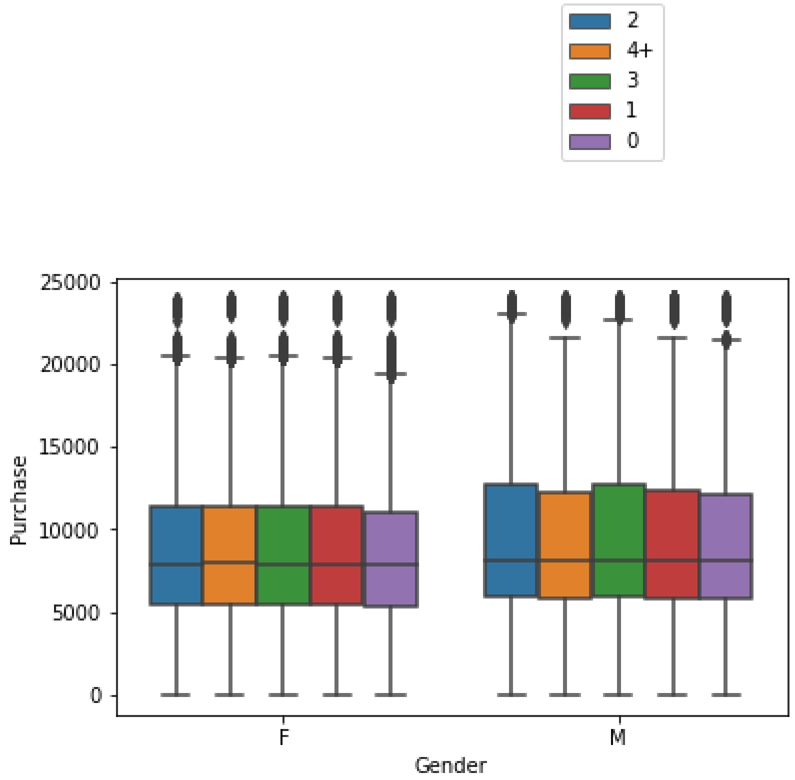
In [133…

```
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Age')
plt.legend(bbox_to_anchor =(0.65, 1.25))
plt.show()

sns.boxplot(data=df, y='Purchase', x='Gender', hue='City_Category')
plt.legend(bbox_to_anchor =(0.65, 1.25))
plt.show()

sns.boxplot(data=df, y='Purchase', x='Gender', hue='Marital_Status')
plt.legend(bbox_to_anchor =(0.65, 1.25))
plt.show()

sns.boxplot(data=df, y='Purchase', x='Gender', hue='Stay_In_Current_City_Years')
plt.legend(bbox_to_anchor =(0.65, 1.25))
plt.show()
```


0-17
55+
26-35
46-50
51-55
36-45
18-25
Purchase
25000
20000
15000
10000
5000
0
F
M
Gender


A
C
B
25000
20000
15000
10000
5000
0
Purchase
F
M
Gender


0
1
25000
20000
15000
10000
5000
0
Purchase
F
M
Gender



Observations

- Male customers living in `City_Category C` spend more money

Using the Central Limit Theorem

let 95% interwal width

average amount spend for each customer - Male & Female

In [134…

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

Out[134…

	User_ID	Gender	Purchase
0	1000001	F	334093
1	1000002	M	810472
2	1000003	M	341635
3	1000004	M	206468
4	1000005	M	821001
...	...	...	...
5886	1006036	F	4116058
5887	1006037	F	1119538
5888	1006038	F	90034
5889	1006039	F	590319
5890	1006040	M	1653299

5891 rows × 3 columns

In [135…

```
new_df['Gender'].value_counts()
```

Out[135…

```
M    4225
F    1666
Name: Gender, dtype: int64
```

In [123…

```
male_avg = new_df[new_df['Gender']=='M']['Purchase'].mean()
female_avg = new_df[new_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
```

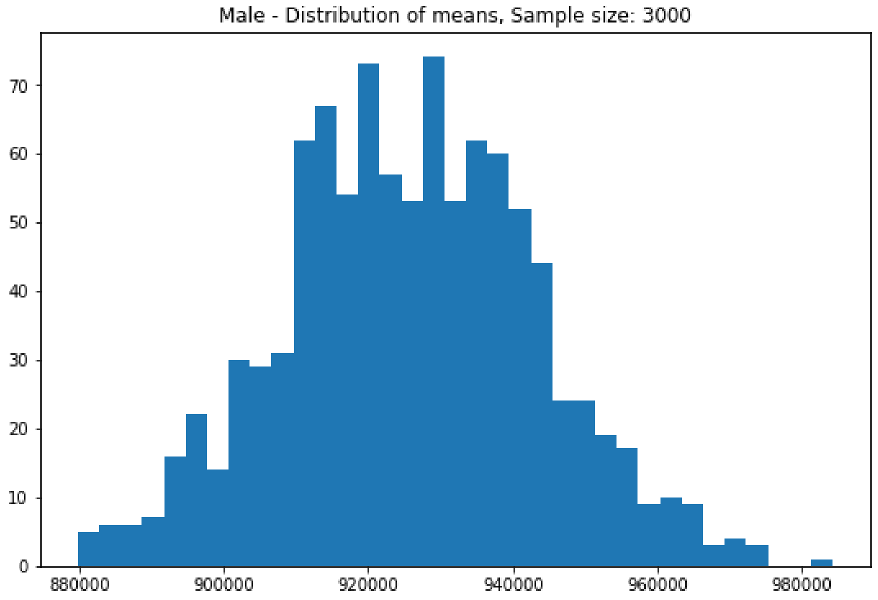
Observation

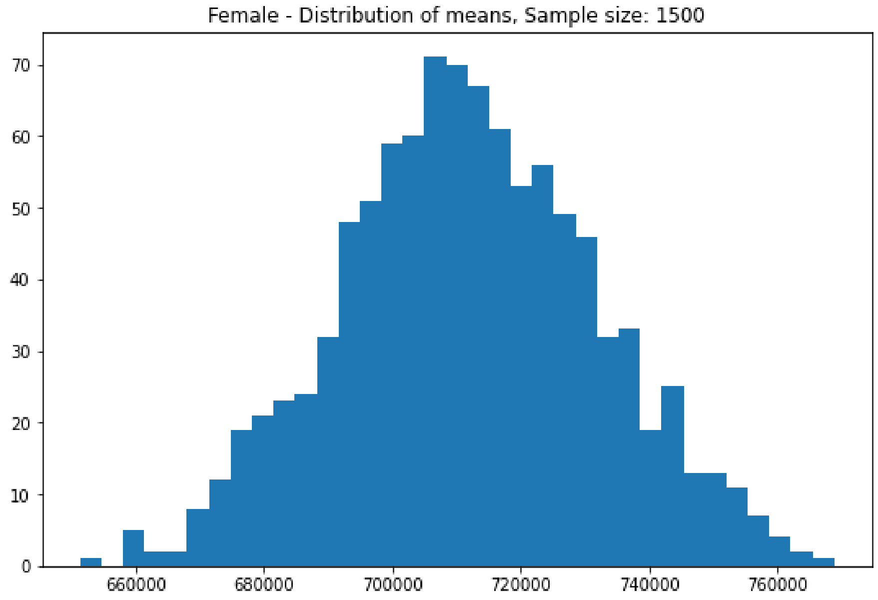
1. Male customers spend more money than female customers

In [139…

```
male_df = new_df[new_df['Gender']=='M']
female_df = new_df[new_df['Gender']=='F']
genders = ["M", "F"]

male_sample_size = 3000
female_sample_size = 1500
n = 1000
male_means = []
female_means = []

for _ in range(n):
    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('Observation')
print(" Population mean - Mean of sample means")
print("Male: {:.2f}".format(np.mean(male_means)))
print("Female: {:.2f}".format(np.mean(female_means)))

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

print("\n Confidence Interval of means")
#z-score value for 97.5% is 1.96
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 - clt
male_upper_lim = male_sample_mean + clt
print("Male : ({:.2f}, {:.2f})".format(male_lower_lim, male_upper_lim))

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 - clt
female_upper_lim = female_sample_mean + clt
print("Female : ({:.2f}, {:.2f})".format(female_lower_lim, female_upper_lim))
```






```
Observation
 Population mean - Mean of sample means
Male: 925777.80
Female: 711928.21

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

 Confidence Interval of means
Male : (895617.83, 955070.97)
Female : (673254.77, 750794.02)
```

average amount spend for each customer - married and unmarried

In [140…

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

Out[140…

	User_ID	Marital_Status	Purchase
0	1000001	0	334093
1	1000002	0	810472

	User_ID	Marital_Status	Purchase
2	1000003	0	341635
3	1000004	1	206468
4	1000005	1	821001
...	...	...	...
5886	1006036	1	4116058
5887	1006037	0	1119538
5888	1006038	0	90034
5889	1006039	1	590319
5890	1006040	0	1653299

5891 rows × 3 columns

In [141…

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

Out[141…

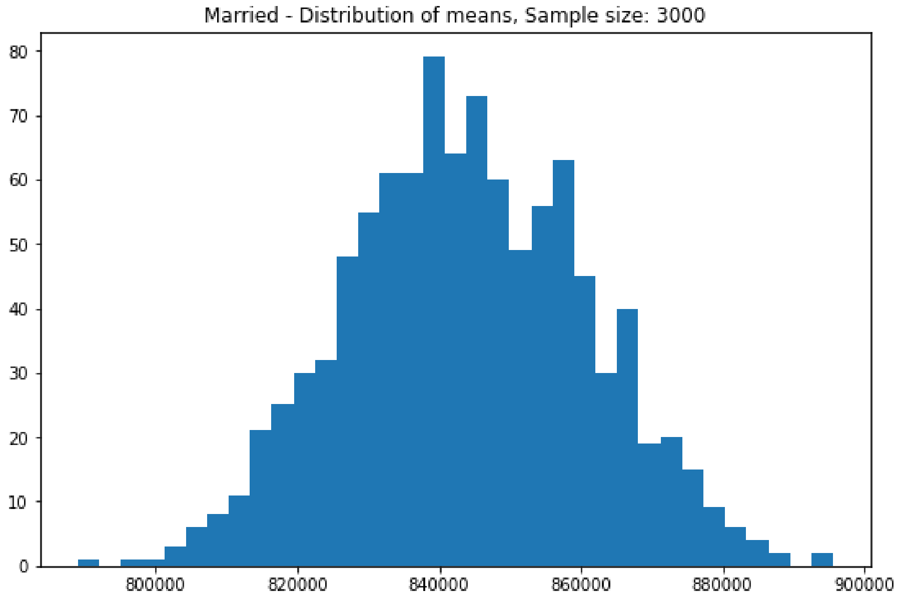
```
0    3417
1    2474
Name: Marital_Status, dtype: int64
```

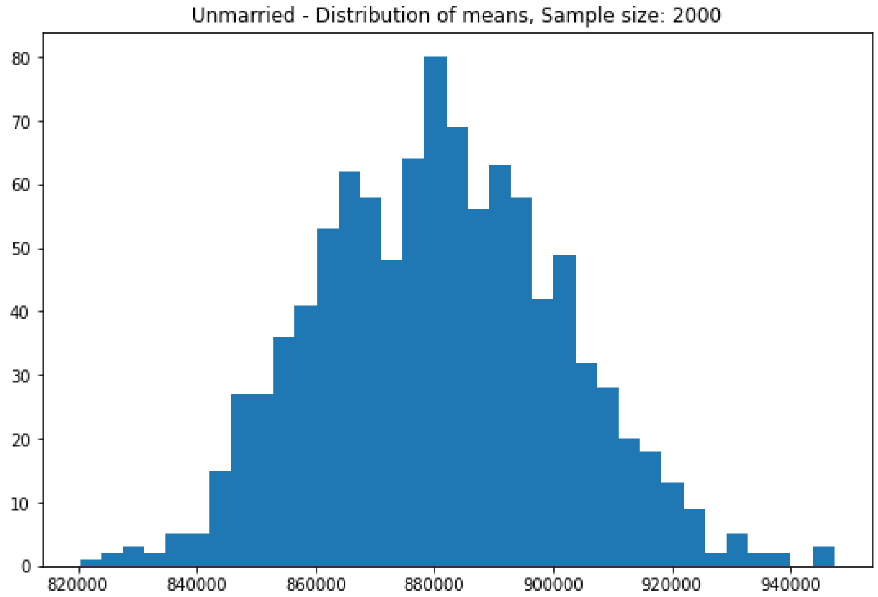
In [143…

```
M_sample_size = 3000
U_sample_size = 2000
n = 1000
M_means = []
U_means = []

for _ in range(n):
    M_mean = new_df[new_df['Marital_Status']==1].sample(M_sample_size, replace=True)['Purchase'].mean()
    U_mean = new_df[new_df['Marital_Status']==0].sample(U_sample_size, replace=True)['Purchase'].mean()
```

```
    M_means.append(M_mean)
    U_means.append(U_mean)


fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))

axis[0].hist(M_means, bins=35)
axis[1].hist(U_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('Observation')
print(" Population mean - Mean of sample means")
print("Married: {:.2f}".format(np.mean(M_means)))
print("Unmarried: {:.2f}".format(np.mean(U_means)))

print("\n Sample mean")
print("Married : {:.2f} Sample std: {:.2f}".format(new_df[new_df['Marital_Status']==1]['Purchase'].mean(),
new_df[new_df['Marital_Status']==1]['Purchase'].std()))
print("Unmarried : {:.2f} Sample std: {:.2f}".format(new_df[new_df['Marital_Status']==0]['Purchase'].mean(),
new_df[new_df['Marital_Status']==0]['Purchase'].std()))

print("\n Confidence Interval of means")
for val in ["Married", "Unmarried"]:
    new_val = 1 if val == "Married" else 0
```

```
    new_df1 = new_df[new_df['Marital_Status']==new_val]

    clt = 1.96*new_df1['Purchase'].std()/np.sqrt(len(new_df))

    sample_mean = new_df1['Purchase'].mean()
    lower_lim = sample_mean - clt
    upper_lim = sample_mean + clt

    print("{} : ({:.2f}, {:.2f})".format(val, lower_lim, upper_lim))
```






```
Observation
 Population mean - Mean of sample means
Married: 843910.53
Unmarried: 880845.94
 Sample mean
Married : 843526.80 Sample std: 935352.12
Unmarried : 880575.78 Sample std: 949436.25

 Confidence Interval of means
Married : (819641.17, 867412.43)
Unmarried : (856330.49, 904821.07)
```

average amount spend for each customer - Age

In [146…

```
new_df = df.groupby(['User_ID', 'Age'])[['Purchase']].sum()
new_df = new_df.reset_index()
new_df
```

Out[146…

	User_ID	Age	Purchase
0	1000001	0-17	334093
1	1000002	55+	810472
2	1000003	26-35	341635
3	1000004	46-50	206468
4	1000005	26-35	821001
...	...	...	...
5886	1006036	26-35	4116058
5887	1006037	46-50	1119538
5888	1006038	55+	90034
5889	1006039	46-50	590319
5890	1006040	26-35	1653299

5891 rows × 3 columns

In [147…

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

Out[147…

```
26-35    2053
36-45    1167
18-25    1069
46-50     531
51-55     481
55+       372
0-17      218
Name: Age, dtype: int64
```

In [148…

```
sample_size = 200
n = 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(n):
        mean = new_df[new_df['Age']==age_interval].sample(sample_size, replace=True)['Purchase'].mean()
        all_means[age_interval].append(mean)

print('Observation')
print("\n Confidence Interval of means")
for val in ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']:
    new_df1 = new_df[new_df['Age']==val]
    clt = 1.96*new_df1['Purchase'].std()/np.sqrt(len(new_df))
    sample_mean = new_df1['Purchase'].mean()
    lower_lim = sample_mean - clt
    upper_lim = sample_mean + clt

    print("Age {} : ({:.2f}, {:.2f})".format(val, lower_lim, upper_lim))
```

```
Observation

 Confidence Interval of means
Age 26-35 : (963315.59, 1016003.04)
Age 36-45 : (854599.57, 904731.85)
Age 18-25 : (832187.79, 877538.45)
```

```
Age 46-50 : (768817.73, 816279.83)
Age 51-55 : (742967.78, 783434.07)
Age 55+ : (523928.99, 555465.50)
Age 0-17 : (601322.78, 636412.84)
```

Insights

- 75% of the users are `Male` and 25% are `Female`
- 60% of the users are `Un-Married` and 40% are `Married`
- There are 20 different types of `Occupation` and `Product_Category`
- More users belong to `City_Category - B`
- `Product_Category - 1, 5, 8, 11, & 2` are highest purchased category.
- People aged `26-35`, `36-45` and '18-25' likely to purchase a lot.
- Male customers living in `City_Category C` spend more money
- Male customers spend more money than female customers
- **Average amount** spend by **Male** customers: **925777.80**
- **Average amount** spend by **Female** customers: **711928.21**

taking 95% confidence interval width

Confidence Interval by Gender

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 by Marital_Status

Married confidence interval of means: **(819641.17, 867412.43)** **Unmarried** confidence interval of means: **(856330.49, 904821.07)**

Confidence Interval by Age

- Age 26-35 : (963315.59, 1016003.04)
- Age 36-45 : (854599.57, 904731.85)
- Age 18-25 : (832187.79, 877538.45)

- Age 46-50 : (768817.73, 816279.83)
- Age 51-55 : (742967.78, 783434.07)
- Age 55+ : (523928.99, 555465.50)
- Age 0-17 : (601322.78, 636412.84)

Recomendations

1. Since Men are purchasing more, company should retain them as they should tend buy more and increase the tactics to aboard women on other hand
2. Since Un-Married people buy more stuff, company should market and able to brind married people as well.
3. As male from `City_Category C` spend more, they are likely to be elite, high rated purchases can be expected by them. Also from other cities the company should try to aqcuire with extra efforts like advertising, descounts etc..
4. Since `Product_Category - 1, 5, 8, 11, & 2` are highest purchased category. They should be made easy for availability for the customers.

In [ ]:

In [ ]: