1. Defining Problem Statement and Analyzing basic metrics

The Goal of this project is to provide data driven insights so as to improve the descision making and to increase revenue and identify bottleneck by customer profiling.

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

df=pd.read_csv('C:/Users/rahul.kumar/Downloads/walmart.csv')
```

1.1 Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), statistical summary

```
df.shape
```

```
(550068, 10)
```

```
df.head()
```

```
   User_ID Product_ID Gender   Age  Occupation City_Category  \
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

  Stay_In_Current_City_Years  Marital_Status  Product_Category
Purchase
0                          2               0                 3
8370
1                          2               0                 1
15200
2                          2               0                12
1422
3                          2               0                12
1057
4                         4+               0                 8
7969
```

```
df.describe(include='all')
```

```
             User_ID Product_ID  Gender     Age     Occupation
City_Category  \
count   5.500680e+05     550068  550068  550068  550068.000000
550068
unique           NaN       3631       2       7            NaN
3
top              NaN  P00265242       M   26-35            NaN
B
```

```
freq             NaN       1880  414259  219587            NaN
231173
mean    1.003029e+06        NaN     NaN     NaN       8.076707
NaN
std     1.727592e+03        NaN     NaN     NaN       6.522660
NaN
min     1.000001e+06        NaN     NaN     NaN       0.000000
NaN
25%     1.001516e+06        NaN     NaN     NaN       2.000000
NaN
50%     1.003077e+06        NaN     NaN     NaN       7.000000
NaN
75%     1.004478e+06        NaN     NaN     NaN      14.000000
NaN
max     1.006040e+06        NaN     NaN     NaN      20.000000
NaN

       Stay_In_Current_City_Years  Marital_Status  Product_Category  \
count                      550068   550068.000000     550068.000000
unique                          5             NaN               NaN
top                             1             NaN               NaN
freq                       193821             NaN               NaN
mean                          NaN        0.409653          5.404270
std                           NaN        0.491770          3.936211
min                           NaN        0.000000          1.000000
25%                           NaN        0.000000          1.000000
50%                           NaN        0.000000          5.000000
75%                           NaN        1.000000          8.000000
max                           NaN        1.000000         20.000000

            Purchase
count   550068.000000
unique            NaN
top               NaN
freq              NaN
mean      9263.968713
std       5023.065394
min         12.000000
25%       5823.000000
50%       8047.000000
75%      12054.000000
max      23961.000000
```

1.2 Non-Graphical Analysis: Value counts and unique attributes

```
df.nunique()

User_ID                     5891
Product_ID                  3631
Gender                         2
Age                            7
```

```
Occupation                    21
City_Category                  3
Stay_In_Current_City_Years     5
Marital_Status                 2
Product_Category              20
Purchase                   18105
dtype: int64
```

```
columns=list(df)
for column in columns:
    print(column)
    print(df[column].value_counts())

print('..........................................................
.....................................................')
```

```
User_ID
1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
           ...
1002690       7
1002111       7
1005810       7
1004991       7
1000708       6
Name: User_ID, Length: 5891, dtype: int64
..........................................................................
.........................................
Product_ID
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
             ...
P00314842       1
P00298842       1
P00231642       1
P00204442       1
P00066342       1
Name: Product_ID, Length: 3631, dtype: int64
..........................................................................
.........................................
Gender
M    414259
F    135809
Name: Gender, dtype: int64
..........................................................................
```

```
.........................................
Age
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
.....................................................................
.........................................
Occupation
4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: Occupation, dtype: int64
.....................................................................
.........................................
City_Category
B    231173
C    171175
A    147720
Name: City_Category, dtype: int64
.....................................................................
.........................................
Stay_In_Current_City_Years
1     193821
2     101838
3      95285
4+     84726
0      74398
```

```
Name: Stay_In_Current_City_Years, dtype: int64
..........................................................................
........................................
Marital_Status
0    324731
1    225337
Name: Marital_Status, dtype: int64
..........................................................................
........................................
Product_Category
5     150933
1     140378
8     113925
11     24287
2      23864
6      20466
3      20213
4      11753
16      9828
15      6290
13      5549
10      5125
12      3947
7       3721
18      3125
20      2550
19      1603
14      1523
17       578
9        410
Name: Product_Category, dtype: int64
..........................................................................
........................................
Purchase
7011     191
7193     188
6855     187
6891     184
7012     183
        ...
23491      1
18345      1
3372       1
855        1
21489      1
Name: Purchase, Length: 18105, dtype: int64
..........................................................................
........................................
```

1.3 Visual Analysis - Univariate & Bivariate

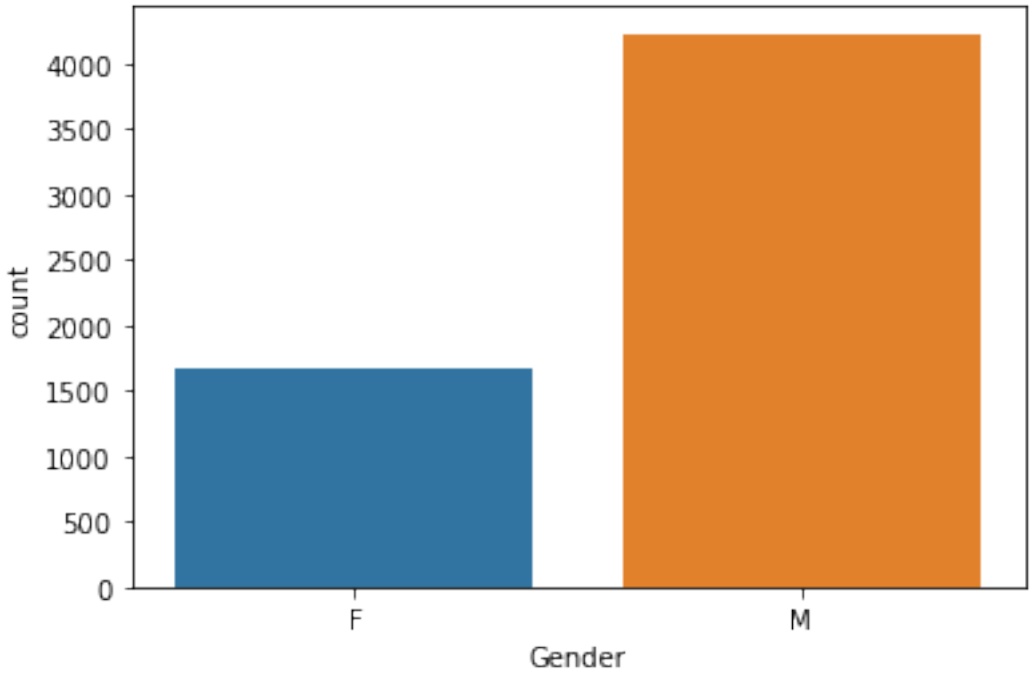
```
def func(x):
    x['Sum_Purchase']=x['Purchase'].sum()
    return x

df_GroupBy_User_ID=df.groupby("User_ID").apply(func)
df_GroupBy_User_ID.sort_values("User_ID")
df_GroupBy_User_ID.drop(['Product_Category','Purchase'],
axis=1,inplace=True)
df_GroupBy_User_ID.drop_duplicates("User_ID",inplace=True)

df_GroupBy_User_ID
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Sum_Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	334093
4	1000002	P00285442	M	55+	16	C	4+	0	810472
5	1000003	P00193542	M	26-35	15	A	3	0	341635
6	1000004	P00184942	M	46-50	7	B	2	1	206468
9	1000005	P00274942	M	26-35	20	A	1	1	821001
...	...	...	...	...	...	...	...	...	...
185450	1004588	P00260042	F	26-35	4	C	0	0	140990
187076	1004871	P00242742	M	18-25	12	C	2	0	108545
221494	1004113	P00351842	M	36-45	17	C	3	0	213550
229480	1005391	P00339342	M	26-35	7	A	0	0	60182
243533	1001529	P00000242	M	18-25	4	C	4+	1	152942

[5891 rows x 9 columns]

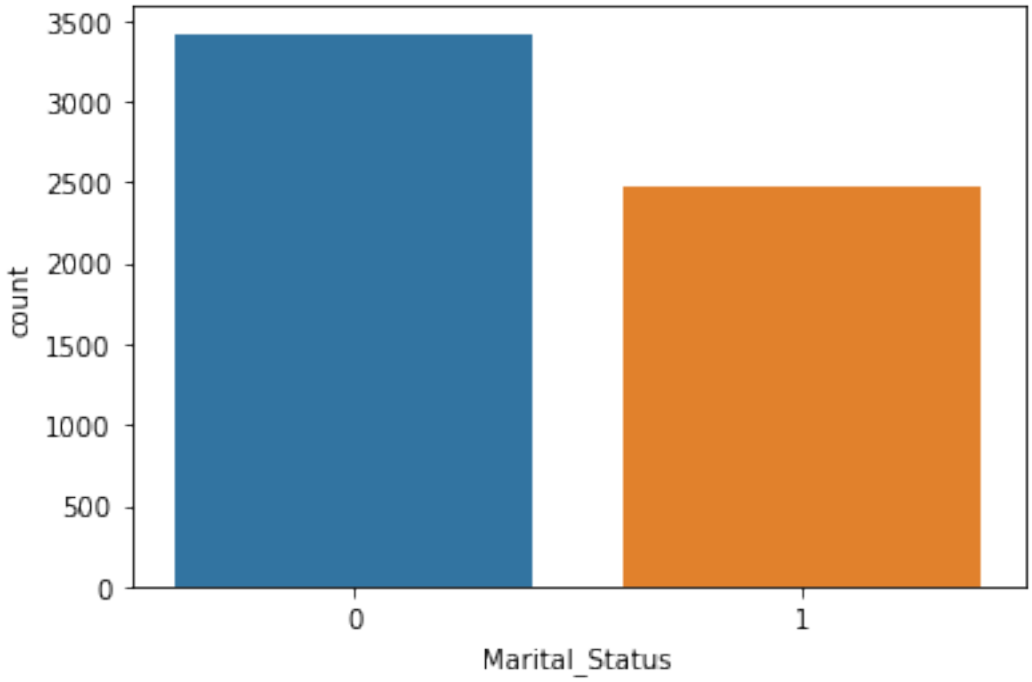
```
sns.countplot(data=df_GroupBy_User_ID, x='Gender')
```

<AxesSubplot:xlabel='Gender', ylabel='count'>



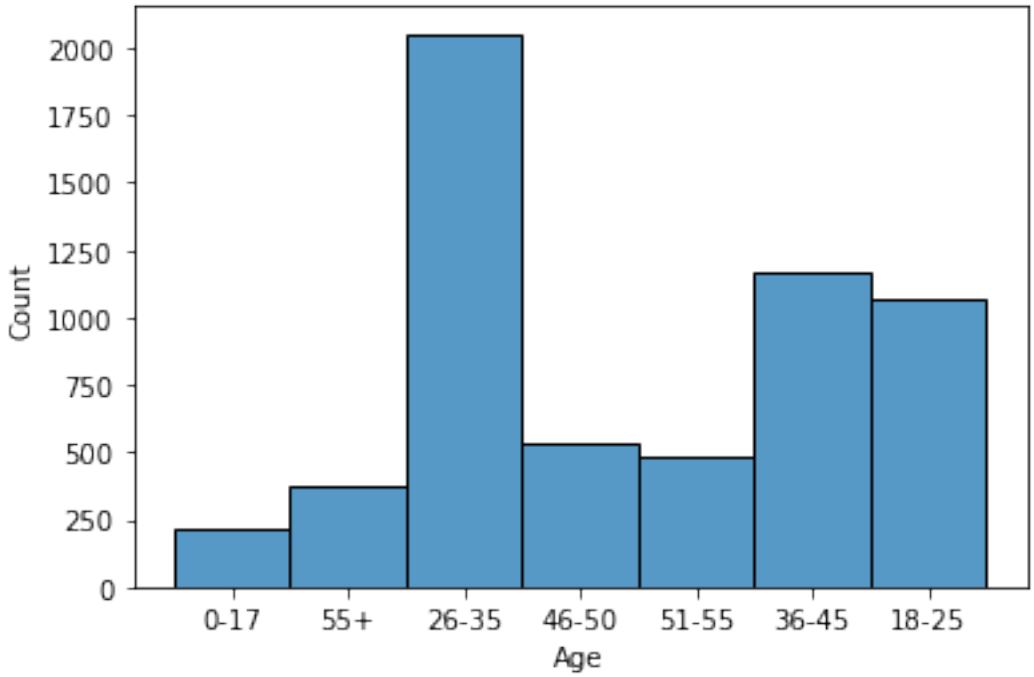

```
sns.countplot(data=df_GroupBy_User_ID, x='Marital_Status')
```

```
<AxesSubplot:xlabel='Marital_Status', ylabel='count'>
```




```
sns.histplot(data=df_GroupBy_User_ID, x='Age')
```

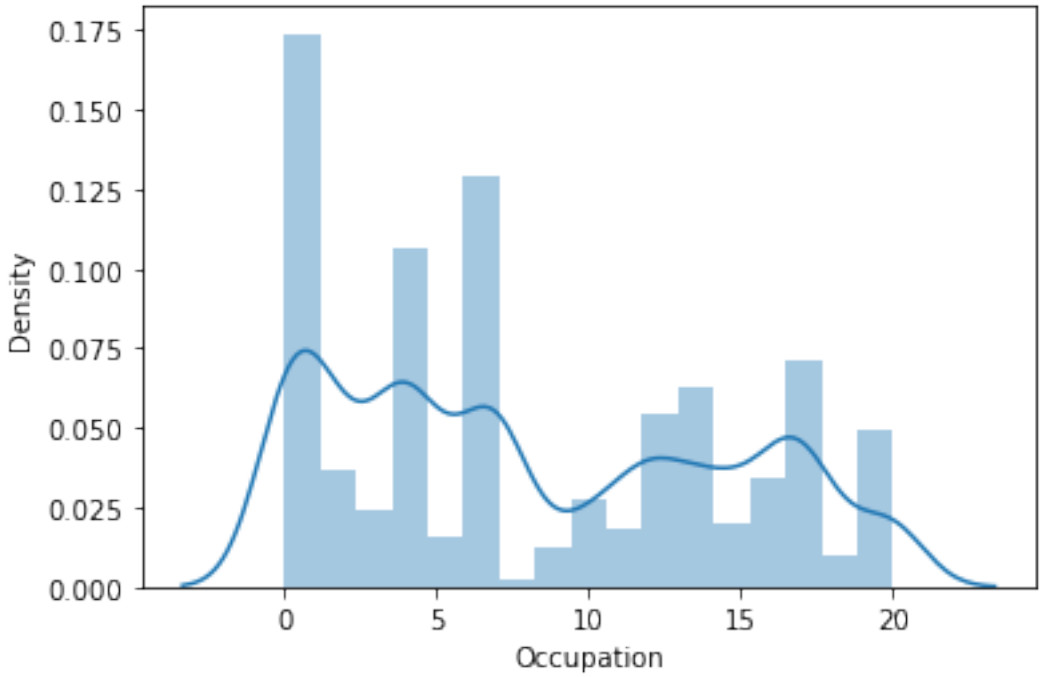
```
<AxesSubplot:xlabel='Age', ylabel='Count'>
```




```
sns.distplot(df_GroupBy_User_ID['Occupation'])
```

```
C:\Users\rahul.kumar\Anaconda3\lib\site-packages\seaborn\
distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

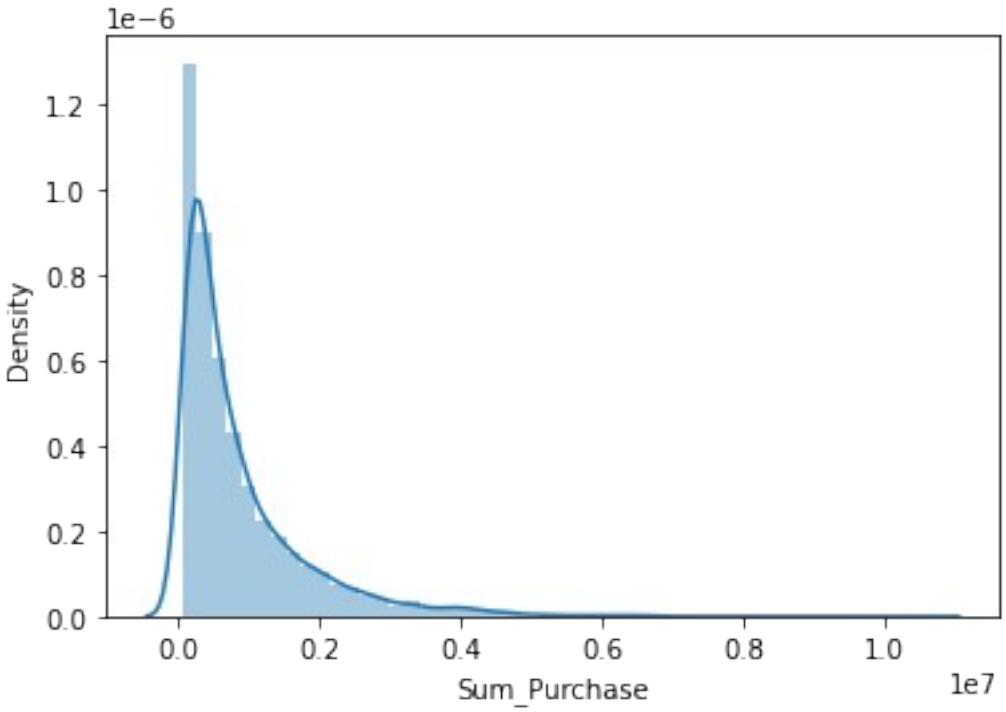
```
<AxesSubplot:xlabel='Occupation', ylabel='Density'>
```




```
sns.distplot(df_GroupBy_User_ID['Sum_Purchase'])
```

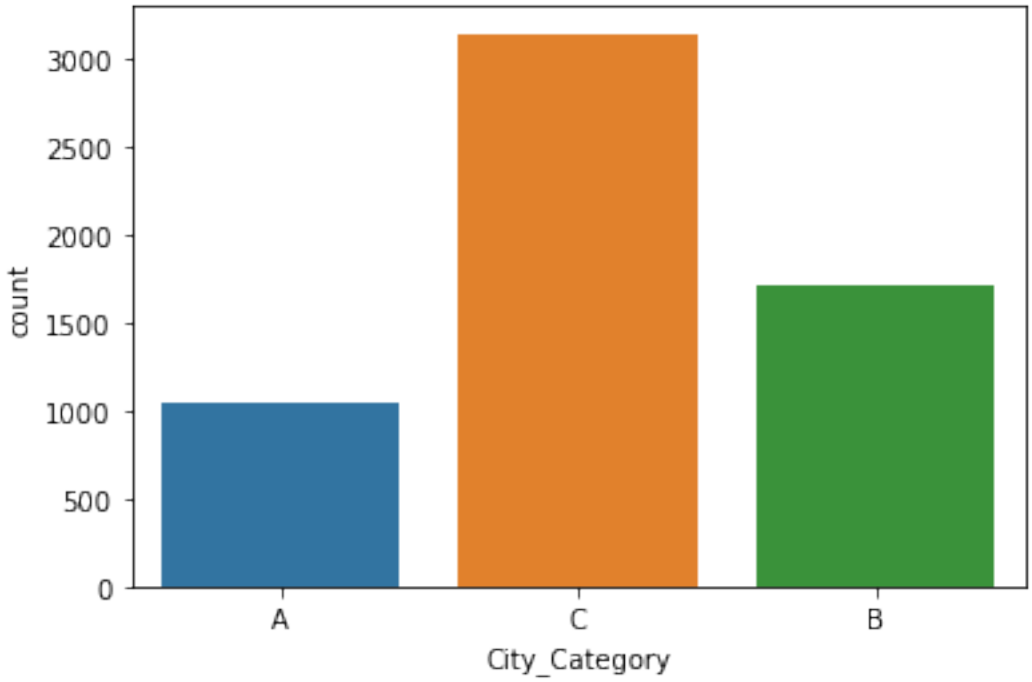
```
C:\Users\rahul.kumar\Anaconda3\lib\site-packages\seaborn\
distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:xlabel='Sum_Purchase', ylabel='Density'>
```



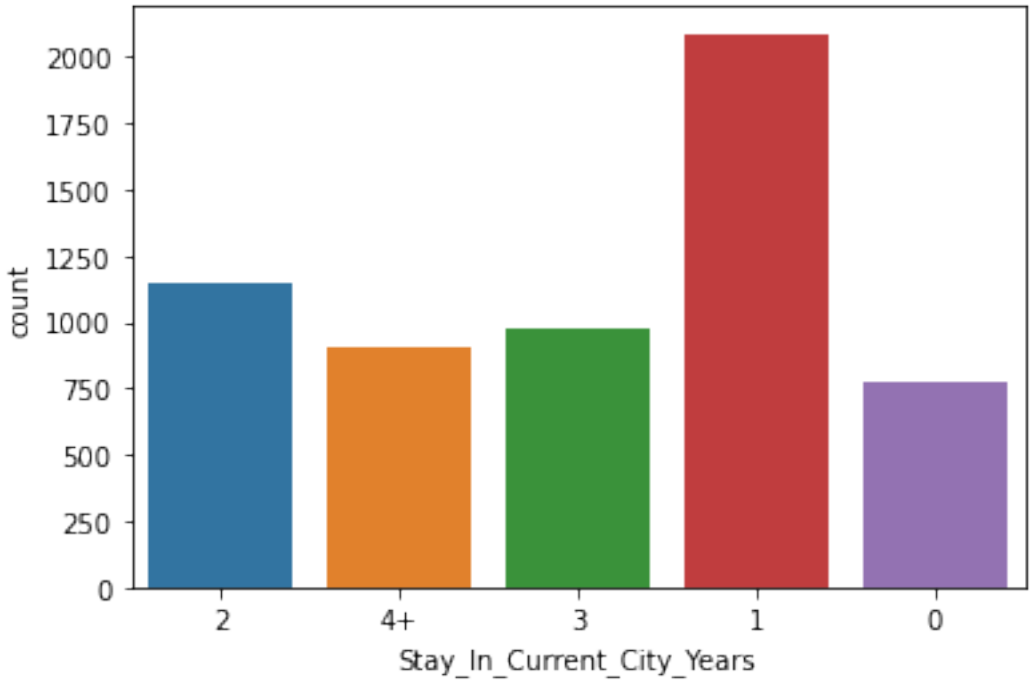

```
sns.countplot(data=df_GroupBy_User_ID, x='City_Category')
```

```
<AxesSubplot:xlabel='City_Category', ylabel='count'>
```



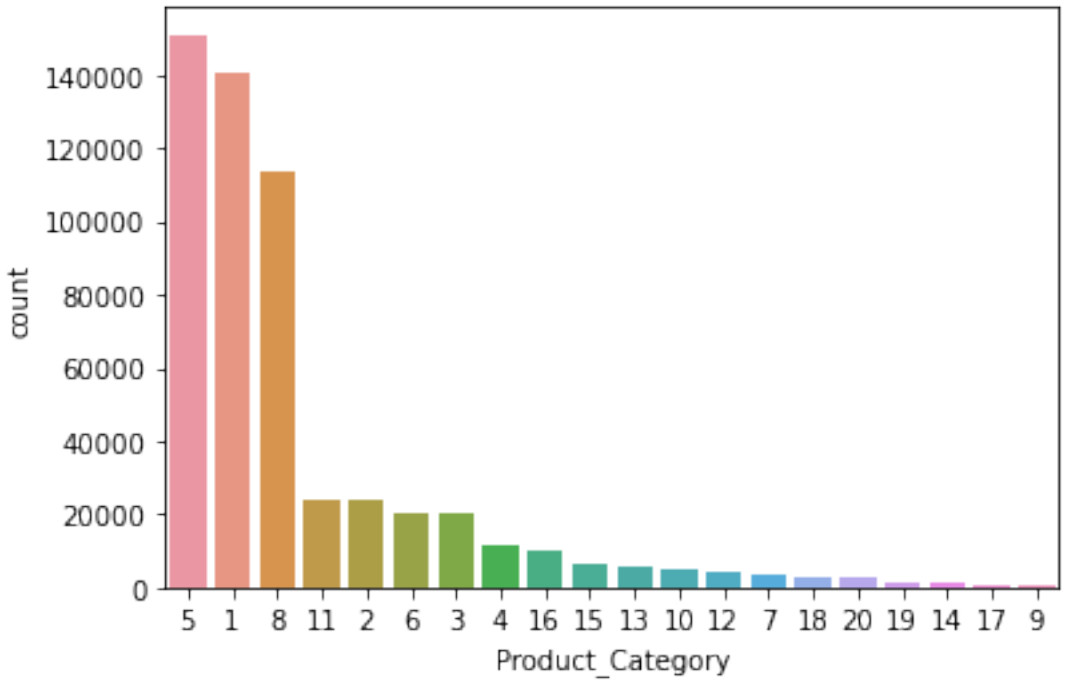

```
sns.countplot(data=df_GroupBy_User_ID, x='Stay_In_Current_City_Years')
```

```
<AxesSubplot:xlabel='Stay_In_Current_City_Years', ylabel='count'>
```



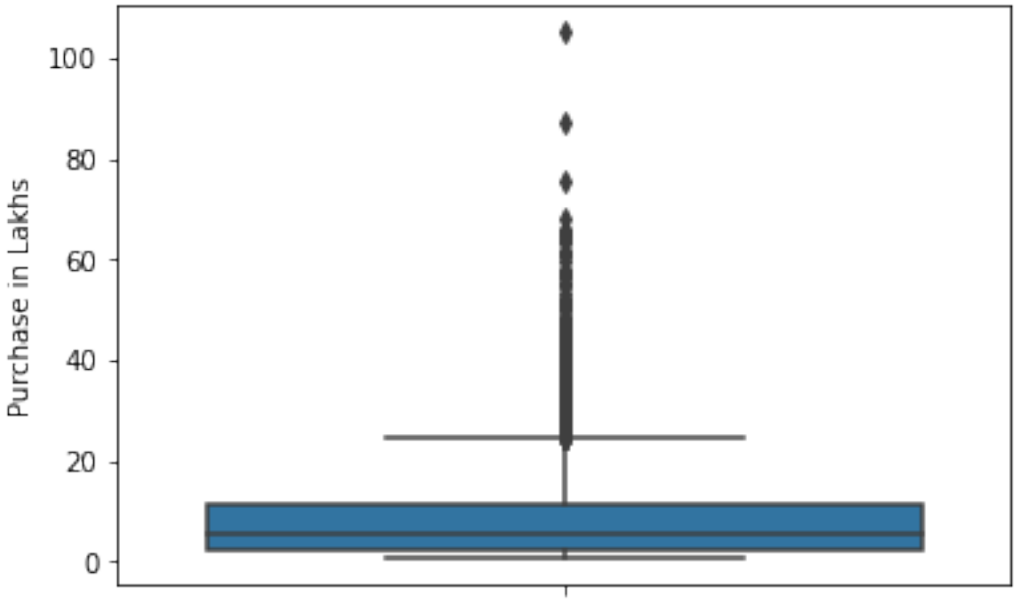

```
sns.countplot(data=df,
x='Product_Category',order=df['Product_Category'].value_counts().index
)
```

```
<AxesSubplot:xlabel='Product_Category', ylabel='count'>
```



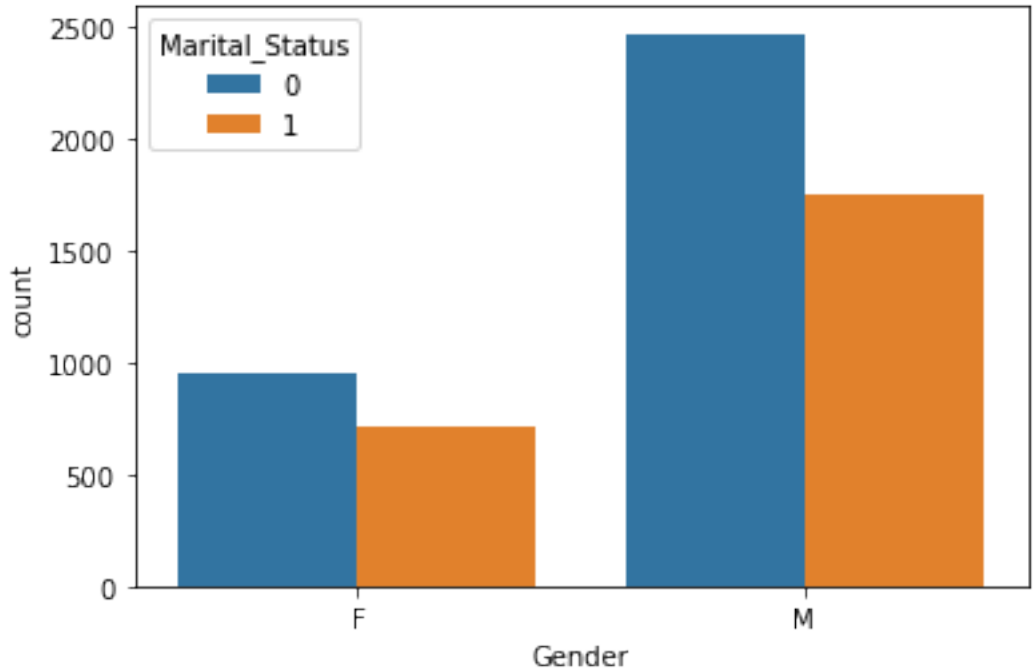

```
plot=sns.boxplot(y=df_GroupBy_User_ID['Sum_Purchase']/100000)
plot.set(ylabel='Purchase in Lakhs')
```

```
[Text(0, 0.5, 'Purchase in Lakhs')]
```



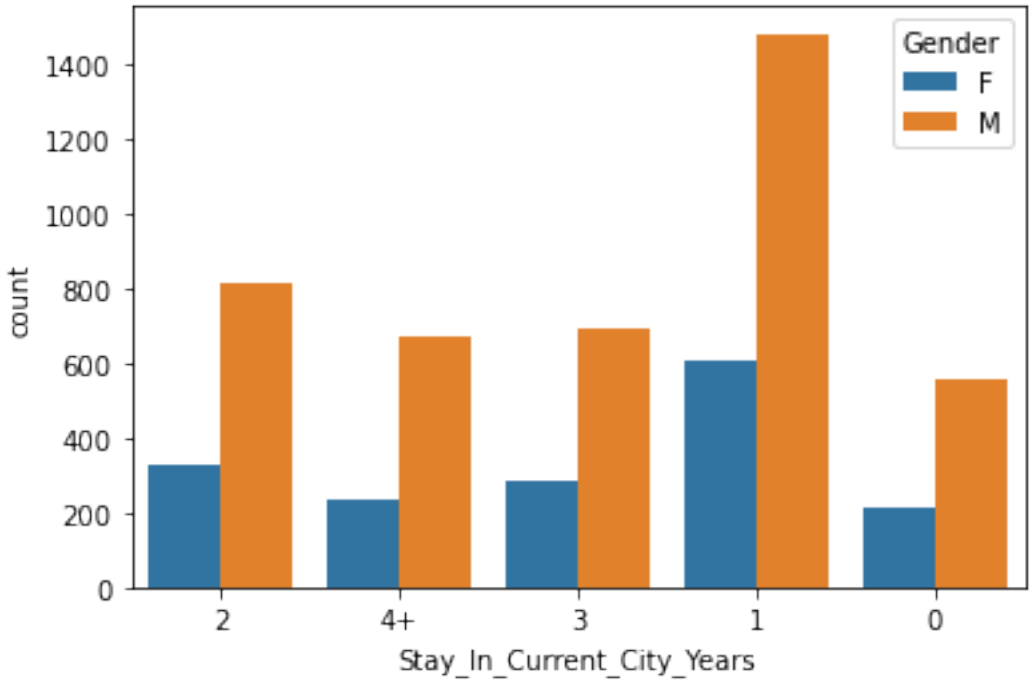

```
sns.countplot(data=df_GroupBy_User_ID,
x='Gender',hue='Marital_Status')
```

```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



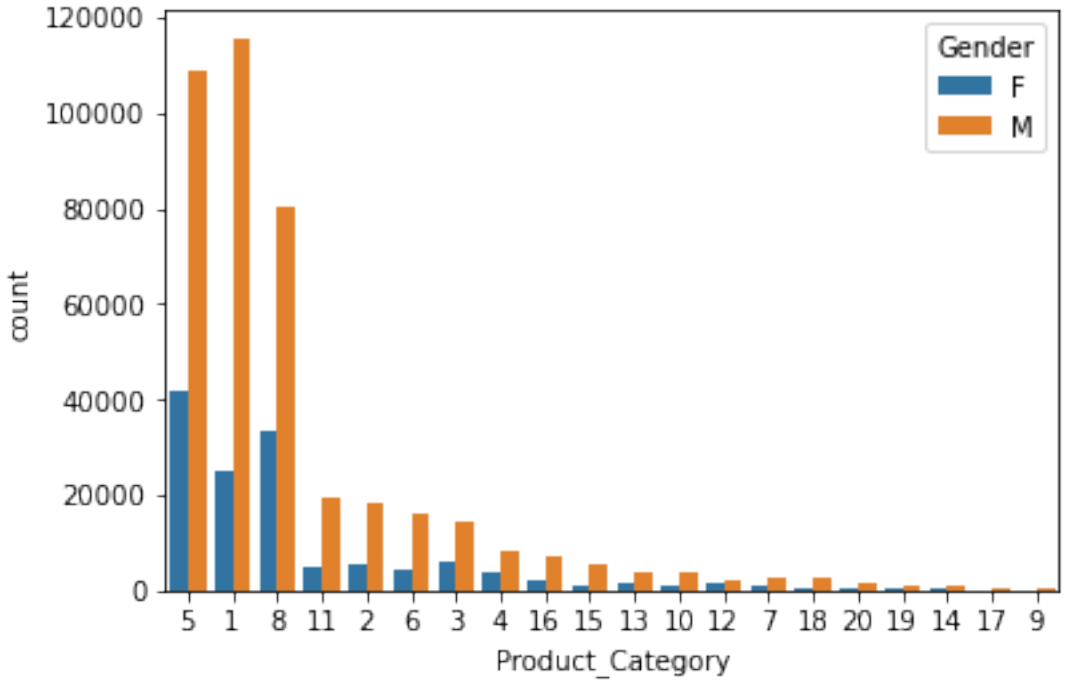

```
sns.countplot(data=df_GroupBy_User_ID,
x='Stay_In_Current_City_Years',hue='Gender')
```

```
<AxesSubplot:xlabel='Stay_In_Current_City_Years', ylabel='count'>
```



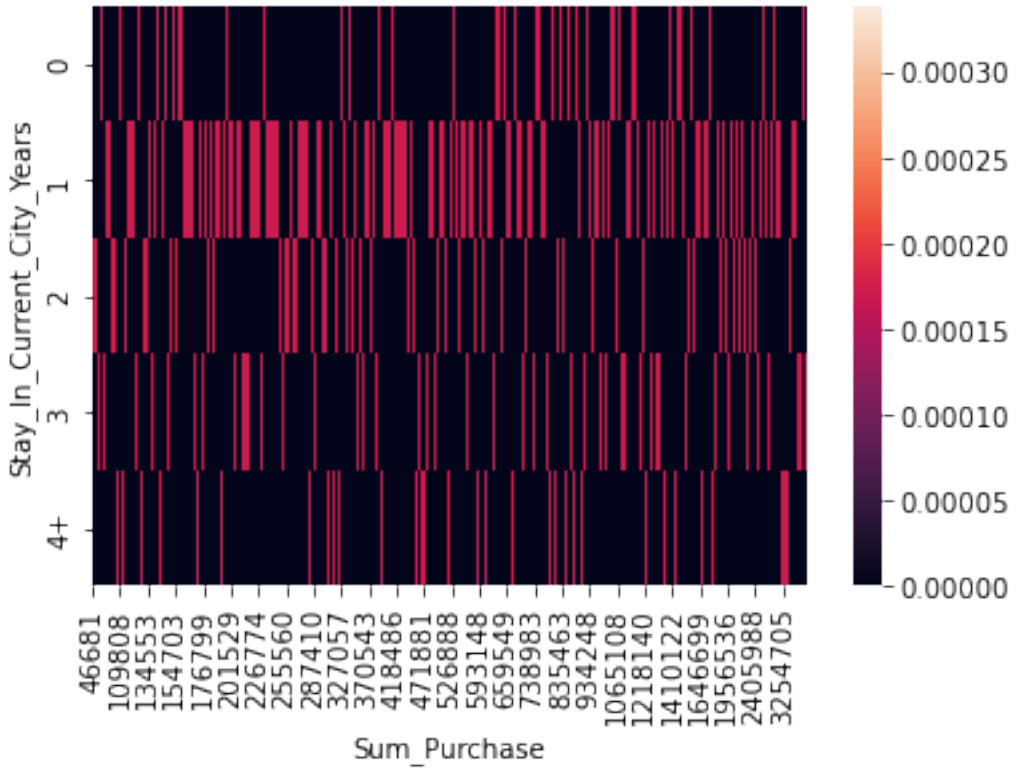

```
sns.countplot(data=df,
x='Product_Category',hue=df['Gender'],order=df['Product_Category'].val
ue_counts().index)
```

```
<AxesSubplot:xlabel='Product_Category', ylabel='count'>
```



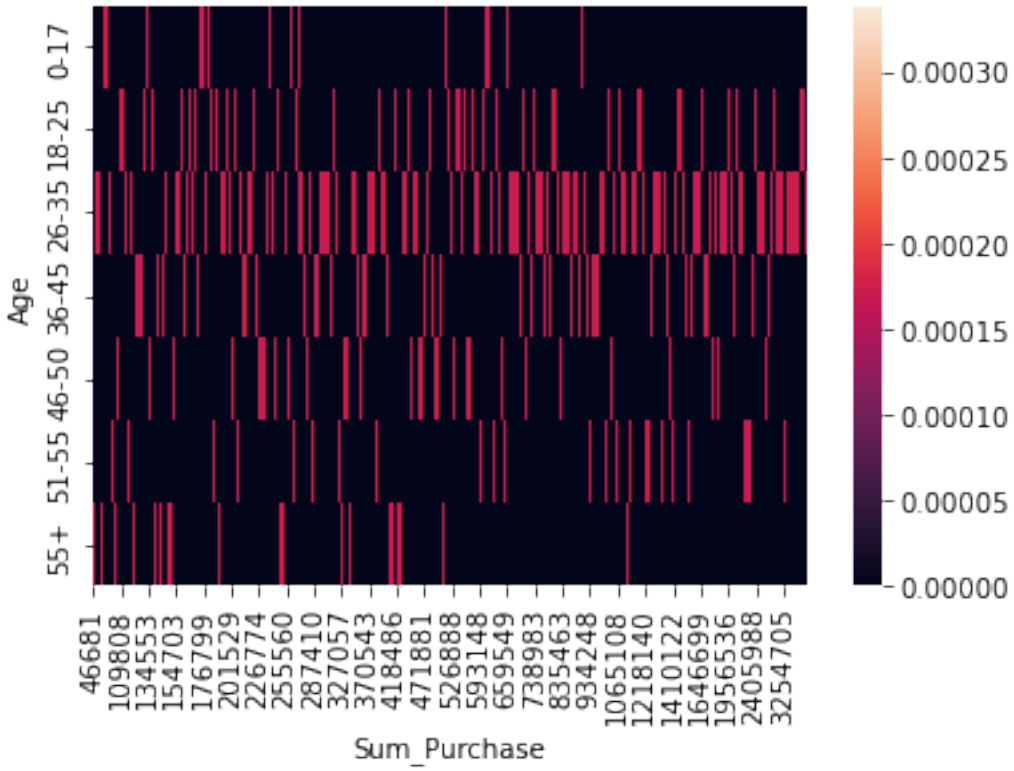

```
sns.heatmap(data=pd.crosstab(df_GroupBy_User_ID['Stay_In_Current_City_
Years'],df_GroupBy_User_ID['Sum_Purchase'],normalize=True))
```

```
<AxesSubplot:xlabel='Sum_Purchase',
ylabel='Stay_In_Current_City_Years'>
```



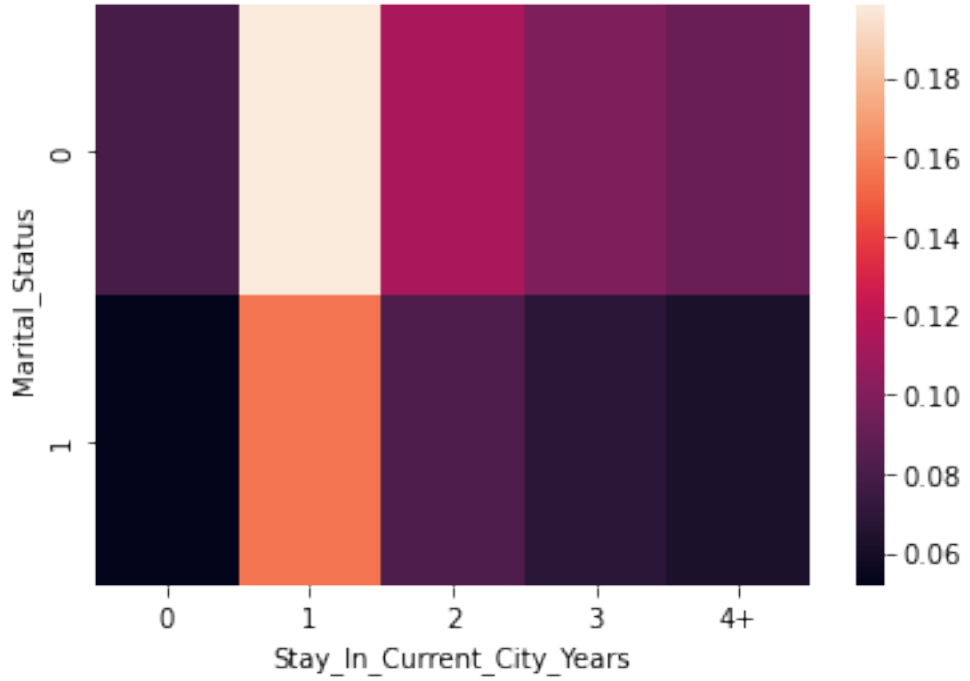

```
sns.heatmap(data=pd.crosstab(df_GroupBy_User_ID['Age'],df_GroupBy_User
_ID['Sum_Purchase'],normalize=True))
```

```
<AxesSubplot:xlabel='Sum_Purchase', ylabel='Age'>
```



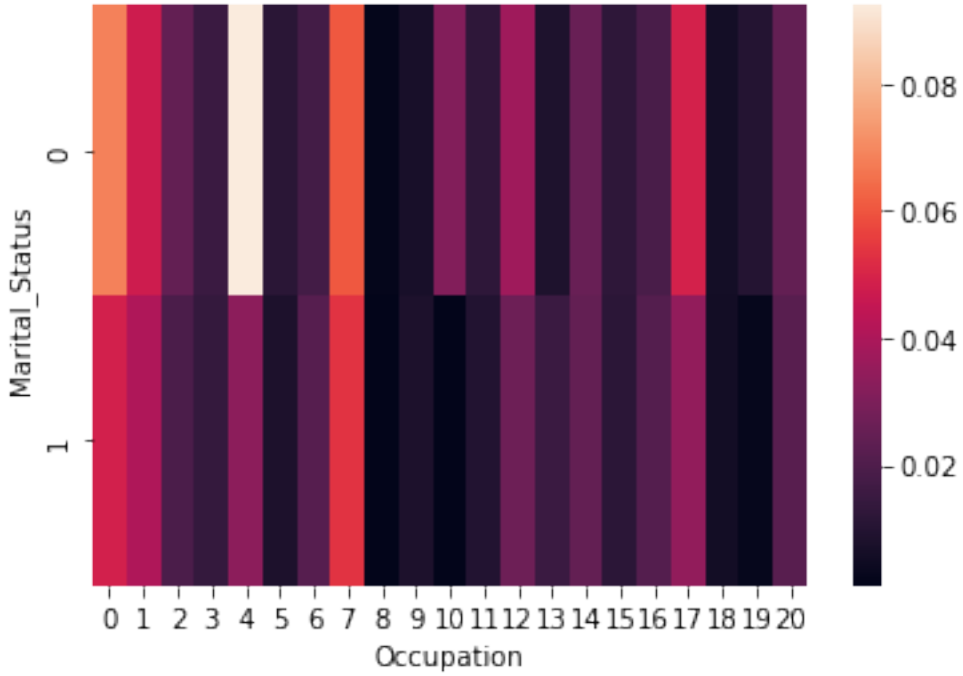

```
sns.heatmap(data=pd.crosstab(df_GroupBy_User_ID['Marital_Status'],df_G
roupBy_User_ID['Stay_In_Current_City_Years'],normalize=True))
```

```
<AxesSubplot:xlabel='Stay_In_Current_City_Years',
ylabel='Marital_Status'>
```



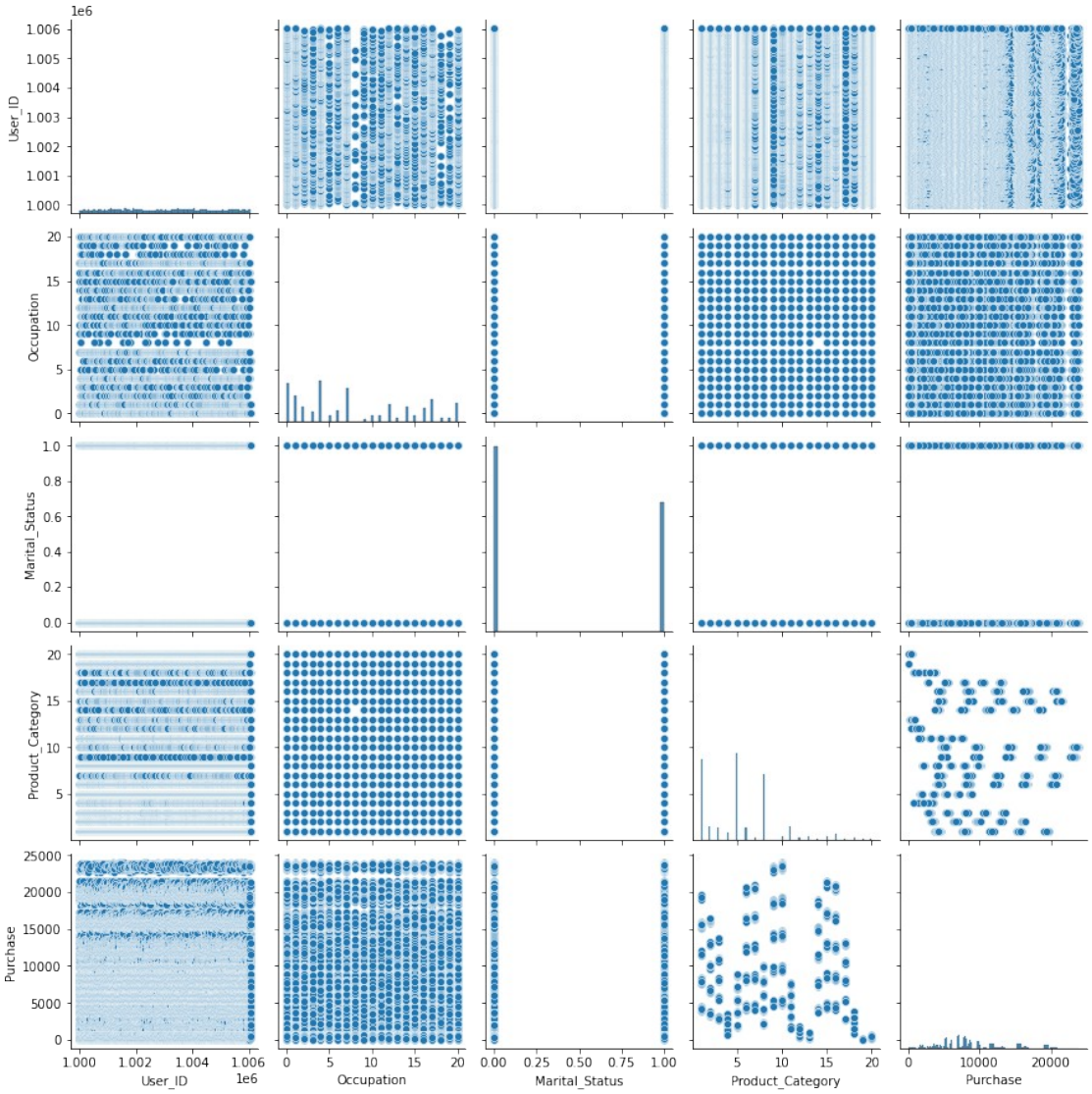
```
sns.heatmap(data=pd.crosstab(df_GroupBy_User_ID['Marital_Status'],df_G
roupBy_User_ID['Occupation'],normalize=True))
```

```
<AxesSubplot:xlabel='Occupation', ylabel='Marital_Status'>
```



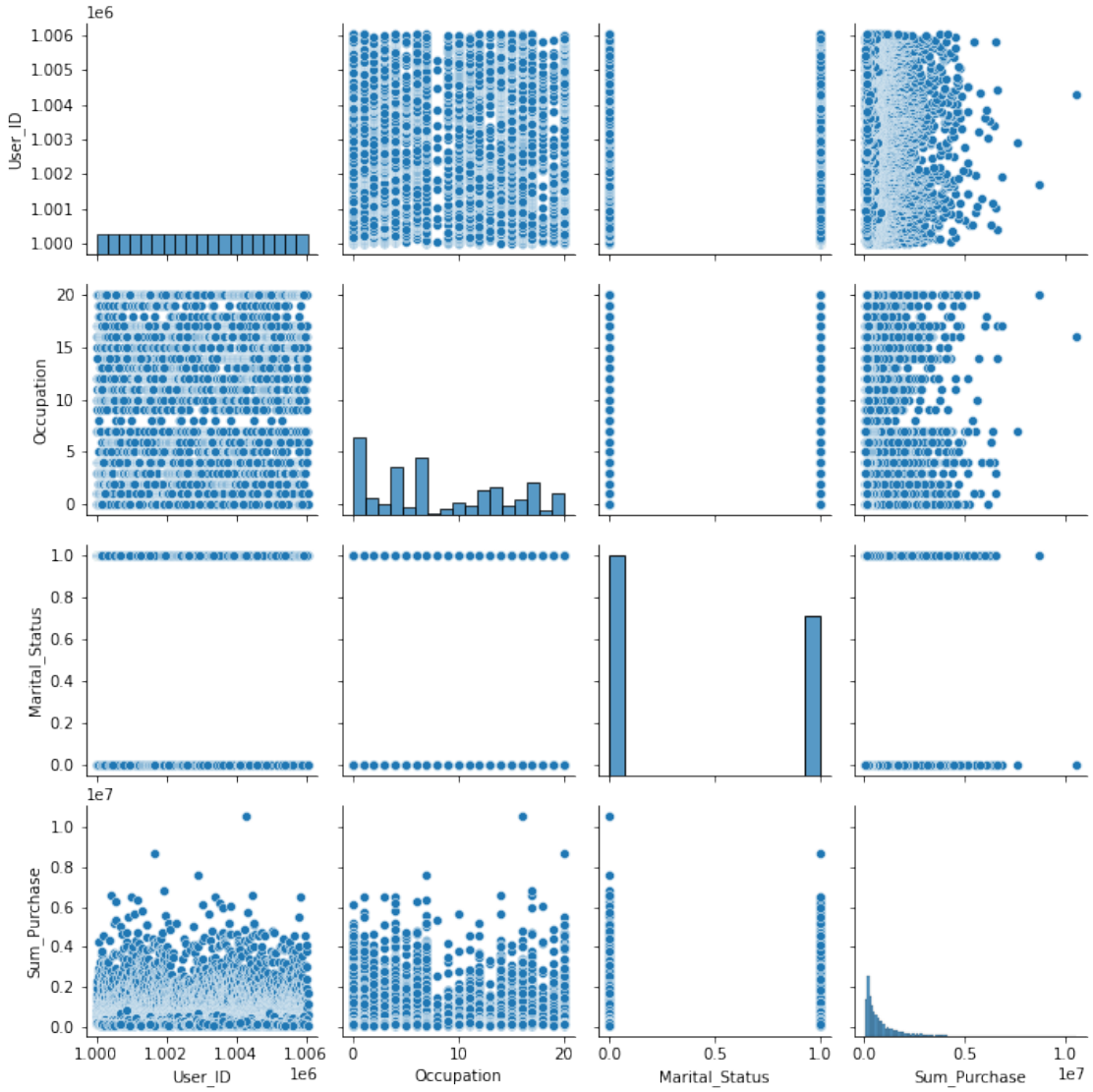

```
sns.pairplot(data=df)
```

```
<seaborn.axisgrid.PairGrid at 0x161ea972e50>
```




```
sns.pairplot(data=df_GroupBy_User_ID)
```

```
<seaborn.axisgrid.PairGrid at 0x16184364a60>
```



2. Missing Value & Outlier Detection

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

```
User_ID                     0
Product_ID                  0
Gender                      0
Age                         0
Occupation                  0
City_Category               0
Stay_In_Current_City_Years  0
Marital_Status              0
Product_Category            0
Purchase                    0
dtype: int64
```

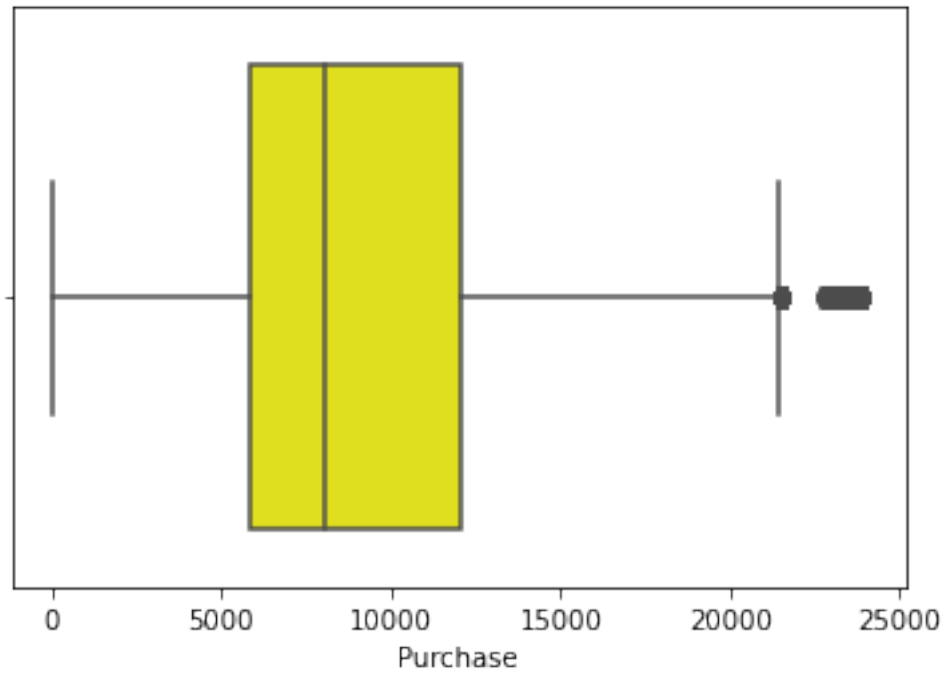
```
Purchase_25=df['Purchase'].quantile(0.25)
Purchase_75=df['Purchase'].quantile(0.75)
```

```
Purchase_IQR=Purchase_75-Purchase_25
LowerLimit_Purchase=max((Purchase_25-
(Purchase_IQR)*1.5),df['Purchase'].min())
UpperLimit_Purchase=min((Purchase_75+
(Purchase_IQR)*1.5),df['Purchase'].max())
Outliers_Purchase=df[(df['Purchase']<LowerLimit_Purchase) |
(df['Purchase']>UpperLimit_Purchase)]['Purchase']
Outliers_Purchase.to_frame().drop_duplicates(keep=False).sort_values(b
y=['Purchase']).reset_index(drop=True).rename(columns={'Purchase':'Out
liers_Purchase'})
```

```
     Outliers_Purchase
0                21402
1                21406
2                21408
3                21409
4                21415
..                 ...
268              23945
269              23950
270              23951
271              23952
272              23956

[273 rows x 1 columns]
```

```
sns.boxplot(data=df,x=df['Purchase'],color='yellow')
```

```
<AxesSubplot:xlabel='Purchase'>
```

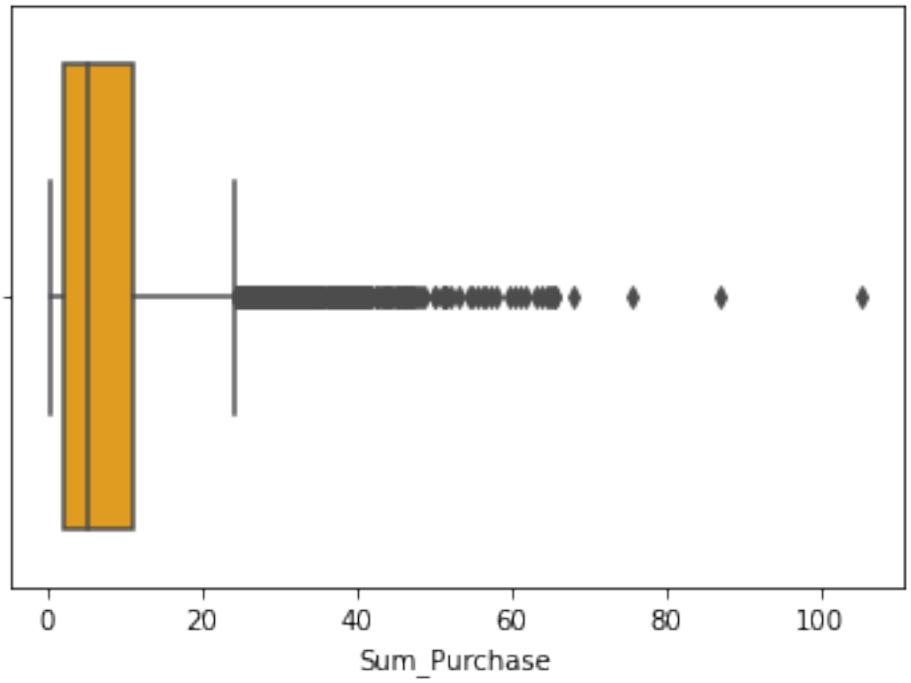


```
Sum_Purchase_25=df_GroupBy_User_ID['Sum_Purchase'].quantile(0.25)
Sum_Purchase_75=df_GroupBy_User_ID['Sum_Purchase'].quantile(0.75)
Sum_Purchase_IQR=Sum_Purchase_75-Sum_Purchase_25
LowerLimit_Sum_Purchase=max((Sum_Purchase_25-
(Sum_Purchase_IQR)*1.5),df_GroupBy_User_ID['Sum_Purchase'].min())
UpperLimit_Sum_Purchase=min((Sum_Purchase_75+
(Sum_Purchase_IQR)*1.5),df_GroupBy_User_ID['Sum_Purchase'].max())
Outliers_Sum_Purchase=df_GroupBy_User_ID[(df_GroupBy_User_ID['Sum_Purc
hase']<LowerLimit_Sum_Purchase) |
(df_GroupBy_User_ID['Sum_Purchase']>UpperLimit_Sum_Purchase)]
['Sum_Purchase']
Outliers_Sum_Purchase.to_frame().drop_duplicates(keep=False).sort_valu
es(by=['Sum_Purchase']).reset_index(drop=True).rename(columns={'Sum_Pu
rchase':'Outliers_Sum_Purchase'})
```

```
     Outliers_Sum_Purchase
0                  2445649
1                  2447282
2                  2448401
3                  2450068
4                  2451245
..                     ...
404                6573609
405                6817493
406                7577756
407                8699596
408               10536909

[409 rows x 1 columns]
```

```
sns.boxplot(data=df_GroupBy_User_ID,x=df_GroupBy_User_ID['Sum_Purchase
']/100000,color='orange')
```

```
<AxesSubplot:xlabel='Sum_Purchase'>
```



3. Business Insights based on Non- Graphical and Visual Analysis

1. For Walmart, almost 70% of Customers are Male and 30% are Female.
2. Unmarried people tends to buy more products as compared to married couple.
3. Most customers falls in 26-35 Year Age group.
4. Walmart is having more popularity in Category C City.
5. Product with ID 1,5,8 are baught frequently by consumers.

4. Central Limit Theorem

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

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

```
df_M=df_GroupBy_User_ID[df_GroupBy_User_ID['Gender']=='M']
df_M
```

```
        User_ID Product_ID Gender    Age  Occupation City_Category  \
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
18      1000007  P00036842      M  36-45           1             B
...         ...        ...    ...    ...         ...           ...
166229  1001674  P00000142      M  36-45           2             C
187076  1004871  P00242742      M  18-25          12             C
221494  1004113  P00351842      M  36-45          17             C
229480  1005391  P00339342      M  26-35           7             A
```

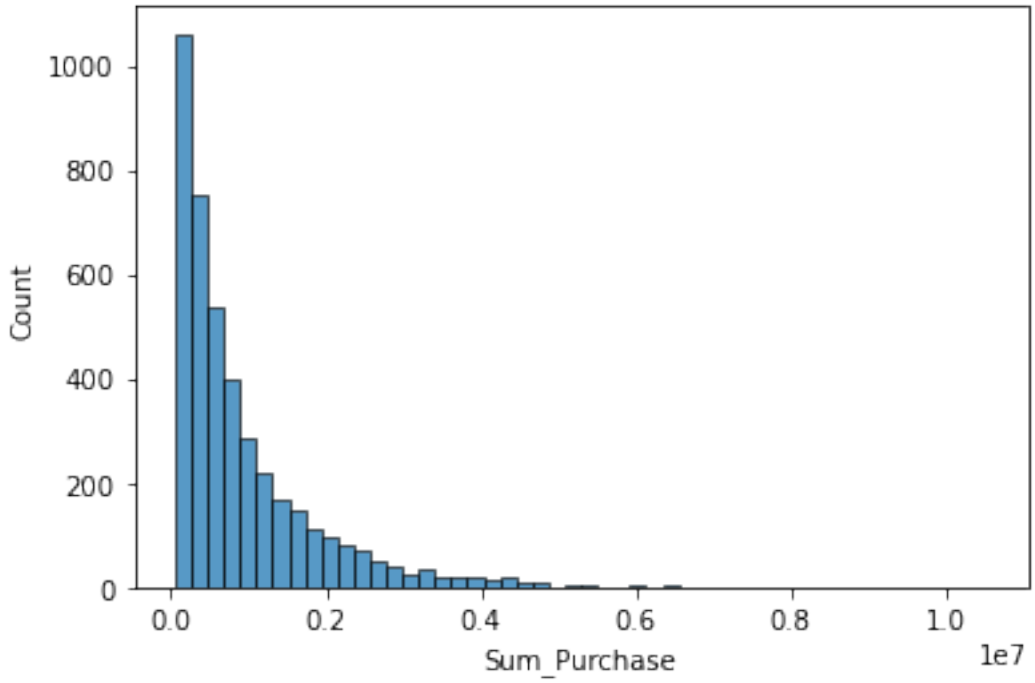
```
243533  1001529  P00000242      M  18-25           4             C

       Stay_In_Current_City_Years  Marital_Status  Sum_Purchase
4                              4+               0        810472
5                               3               0        341635
6                               2               1        206468
9                               1               1        821001
18                              1               1        234668
...                           ...             ...           ...
166229                          3               0         94838
187076                          2               0        108545
221494                          3               0        213550
229480                          0               0         60182
243533                         4+               1        152942

[4225 rows x 9 columns]
```

```
df_F=df_GroupBy_User_ID[df_GroupBy_User_ID['Gender']=='F']
df_F
```

```
        User_ID Product_ID Gender    Age  Occupation City_Category  \
0       1000001  P00069042      F   0-17          10             A
14      1000006  P00231342      F  51-55           9             A
29      1000010  P00085942      F  36-45           1             B
47      1000011  P00192642      F  26-35           1             C
65      1000016  P00244242      F  36-45           0             C
...         ...        ...    ...    ...         ...           ...
158271  1000455  P00117942      F  36-45           2             A
158641  1000527  P00058042      F  26-35           2             B
159635  1000703  P00117842      F    55+           1             C
183417  1004293  P00100642      F  46-50           9             B
185450  1004588  P00260042      F  26-35           4             C

       Stay_In_Current_City_Years  Marital_Status  Sum_Purchase
0                               2               0        334093
14                              1               0        379930
29                             4+               1       2169510
47                              1               0        557023
65                              0               1        150490
...                           ...             ...           ...
158271                          0               0        139887
158641                          2               1         86847
159635                          2               1        102328
183417                          0               1        276411
185450                          0               0        140990

[1666 rows x 9 columns]
```

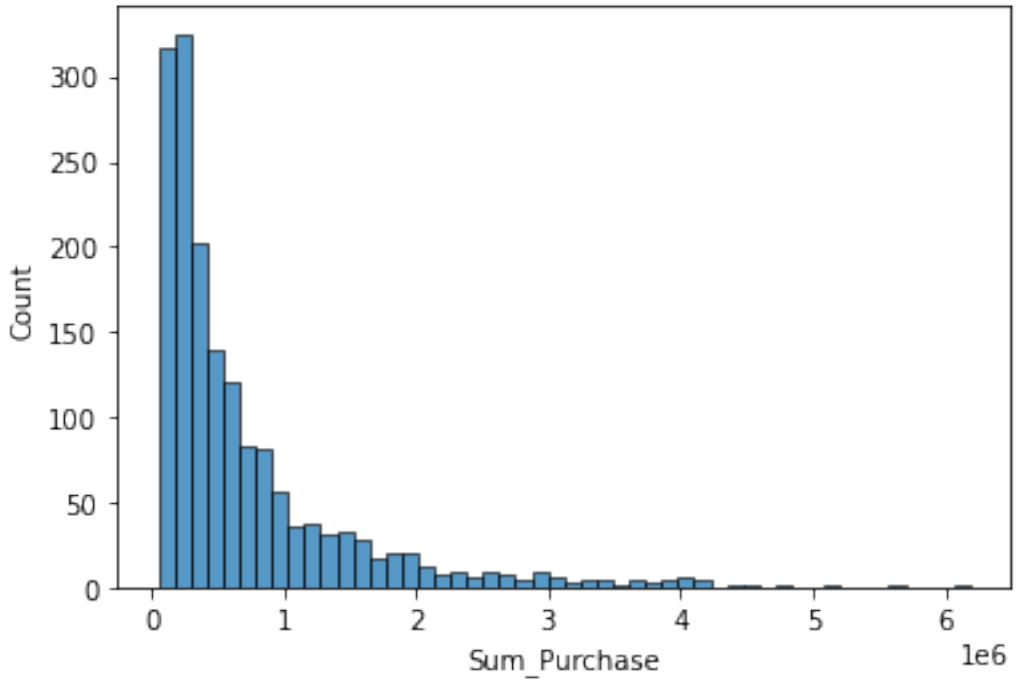
```
sns.histplot(df_M['Sum_Purchase'],bins=50)
```

```
<AxesSubplot:xlabel='Sum_Purchase', ylabel='Count'>
```




```
sns.histplot(df_F['Sum_Purchase'],bins=50)
```

```
<AxesSubplot:xlabel='Sum_Purchase', ylabel='Count'>
```




```
Avg_Purchse_M=round(df_M['Sum_Purchase'].mean(),2)
Avg_Purchse_M
```

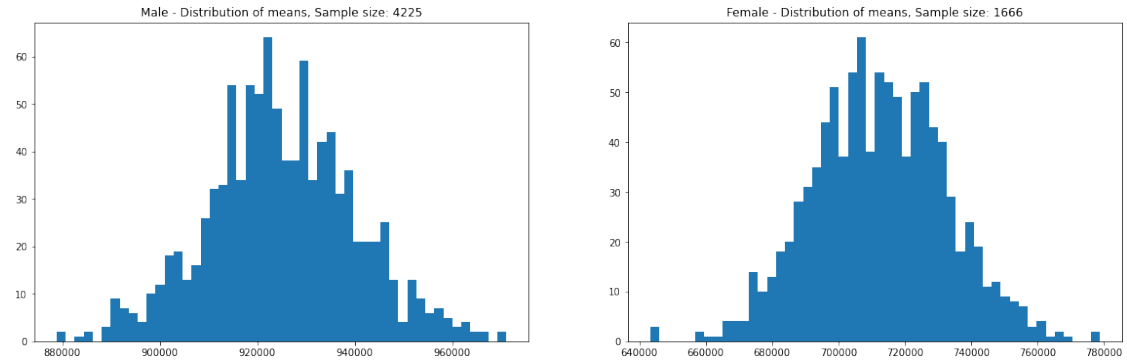
```
925344.4
```

```
Avg_Purchse_F=round(df_F['Sum_Purchase'].mean(),2)
Avg_Purchse_F
```

```
712024.39
```

Men spend more money per transaction than women

```
genders = ["M", "F"]

M_sample_size = 4225
F_sample_size = 1666
size = 1000
M_means = []
F_means = []

for i in range(size):
    M_mean = df_M.sample(M_sample_size, replace=True)
['Sum_Purchase'].mean()
    F_mean = df_F.sample(F_sample_size, replace=True)
['Sum_Purchase'].mean()

    M_means.append(M_mean)
    F_means.append(F_mean)

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

axis[0].hist(M_means, bins=50)
axis[1].hist(F_means, bins=50)
axis[0].set_title("Male - Distribution of means, Sample size: 4225")
axis[1].set_title("Female - Distribution of means, Sample size: 1666")

plt.show()
```




```
np.mean(M_means)
```

```
925071.0127976332
```

```
np.mean(F_means)
```

```
712455.6970366146
```

```
Sample_Mean_M=df_M['Sum_Purchase'].mean()
Sample_Mean_M
```

```
925344.4023668639
```

```
Sample_Mean_F=df_F['Sum_Purchase'].mean()
Sample_Mean_F
```

```
712024.3949579832
```

```
Sample_Std_M=df_M['Sum_Purchase'].std()
Sample_Std_M
```

```
985830.1007953875
```

```
Sample_Std_F=df_F['Sum_Purchase'].std()
Sample_Std_F
```

```
807370.7261464578
```

```
Standard_Error_M=Sample_Std_M/(np.sqrt(len(df_M)))
Standard_Error_M
```

```
15166.616935313654
```

```
Standard_Error_F=Sample_Std_F/(np.sqrt(len(df_F)))
Standard_Error_F
```

```
19780.419602799644
```

```
Lower_Limit_M=Sample_Mean_M-(1.96*Standard_Error_M)
Lower_Limit_M
```

```
895617.8331736492
```

```
Upper_Limit_M=Sample_Mean_M+(1.96*Standard_Error_M)
Upper_Limit_M
```

```
955070.9715600787
```

With 95% Confidence Interval, Average amount spend by male customer will lie in between: (895617.83, 955070.97)

```
Lower_Limit_F=Sample_Mean_F-(1.96*Standard_Error_F)
Lower_Limit_F
```

```
673254.7725364958
```

```
Upper_Limit_F=Sample_Mean_F+(1.96*Standard_Error_F)
Upper_Limit_F
```

```
750794.0173794705
```

With 95% Confidence Interval, Average amount spend by female customer will lie in between: (673254.77, 750794.02)

Same activity for Married vs Unmarried

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

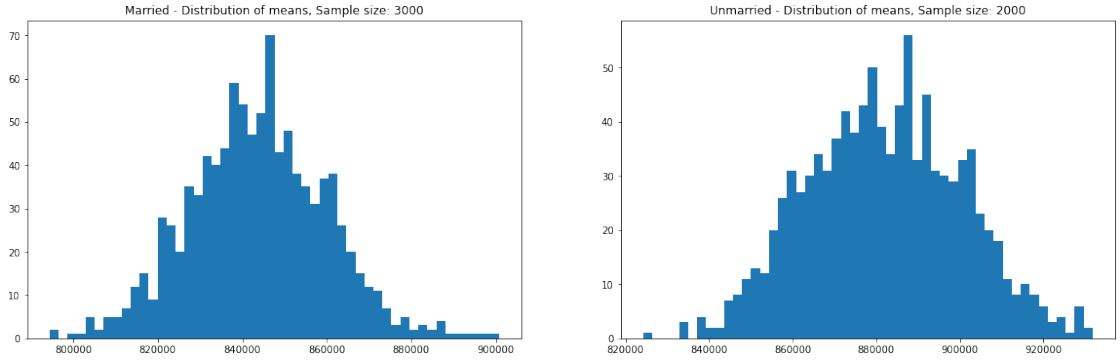
```
      User_ID  Marital_Status  Purchase
0     1000001               0    334093
1     1000002               0    810472
2     1000003               0    341635
3     1000004               1    206468
4     1000005               1    821001
...       ...             ...       ...
5886  1006036   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

[5891 rows x 3 columns]
```

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

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

```
marid_samp_size = 3417
unmarid_sample_size = 2474
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=50)
axis[1].hist(unmarid_means, bins=50)
```

```
axis[0].set_title("Married - Distribution of means, Sample size:
3000")
axis[1].set_title("Unmarried - Distribution of means, Sample size:
2000")

plt.show()
```




```
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("With 95% confidence interval, Average amount spend by {} is
({:.2f}, {:.2f})".format(val, lower_lim, upper_lim))
```

```
With 95% confidence interval, Average amount spend by Married is
(806668.83, 880384.76)
With 95% confidence interval, Average amount spend by Unmarried is
(848741.18, 912410.38)
```

Same activity for Age Group

```
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    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 x 3 columns]
```

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

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

```
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("With 95% confidence interval, Average amount spend by {}
Age Group is : ({:.2f}, {:.2f})".format(val, lower_lim, upper_lim))
```

```
With 95% confidence interval, Average amount spend by 26-35 Age Group
is : (945034.42, 1034284.21)
With 95% confidence interval, Average amount spend by 36-45 Age Group
```

```
is : (823347.80, 935983.62)
With 95% confidence interval, Average amount spend by 18-25 Age Group
is : (801632.78, 908093.46)
With 95% confidence interval, Average amount spend by 46-50 Age Group
is : (713505.63, 871591.93)
With 95% confidence interval, Average amount spend by 51-55 Age Group
is : (692392.43, 834009.42)
With 95% confidence interval, Average amount spend by 55+ Age Group is
: (476948.26, 602446.23)
With 95% confidence interval, Average amount spend by 0-17 Age Group
is : (527662.46, 710073.17)
```

5. Final Insights

- For Walmart, almost 70% of Customers are Male and 30% are Female.
- Unmarried people tends to buy more products as compared to married couple.
- Most customers falls in 26-35 Year Age group.
- Walmart is having more popularity in Category C City.
- Product with ID 1,5,8 are baught frequently by consumers.
- Around 35% customers are Staying in the city from 1 year.
- With 95% Confidence Interval, Average amount spend by male customer will lie in between: (895617.83, 955070.97).
- With 95% Confidence Interval, Average amount spend by female customer will lie in between: (673254.77, 750794.02).
- With 95% confidence interval, Average amount spend by Married is (806668.83, 880384.76).
- With 95% confidence interval, Average amount spend by Unmarried is (848741.18, 912410.38).
- With 95% confidence interval, Average amount spend by 36-45 Age Group is : (823347.80, 935983.62)
- With 95% confidence interval, Average amount spend by 18-25 Age Group is : (801632.78, 908093.46)
- With 95% confidence interval, Average amount spend by 46-50 Age Group is : (713505.63, 871591.93)
- With 95% confidence interval, Average amount spend by 51-55 Age Group is : (692392.43, 834009.42)
- With 95% confidence interval, Average amount spend by 55+ Age Group is : (476948.26, 602446.23)
- With 95% confidence interval, Average amount spend by 0-17 Age Group is : (527662.46, 710073.17)
- With 95% confidence interval, Average amount spend by 26-35 Age Group is : (945034.42, 1034284.21)

6. Recommendations

1. As female consumers are less. So, either we need to include products specific to female or we can give discounts to female.
2. To increase revenue from people with age group [55+] & [0-17], we can give door step service for old age people and set-up gaming stations for young childrens.
3. We need to spend on marketing/advertisement in Category A & B City.