** Business Case: Walmart - Confidence Interval and CLT **

1 CHAPTER 1: DEFINITION OF THE PROBLEM STATEMENT AND ANALYZING BASIC METRICS

Introduction to Walmart

History

The history of Walmart, an American discount department store chain, began in 1950 when businessman Sam Walton purchased a store from Luther E. Harrison in Bentonville, Arkansas, and opened Walton's 5 & 10.[1] The Walmart chain proper was founded in 1962 with a single store in Rogers, expanding outside Arkansas by 1968 and throughout the rest of the Southern United States by the 1980s, ultimately operating a store in every state of the United States, plus its first stores in Canada, by 1995. The expansion was largely fueled by new store construction, although the chains Mohr-Value and Kuhn's Big K were also acquired. The company introduced its warehouse club chain Sam's Club in 1983 and its first Supercenter stores in 1988. By the second decade of the 21st century, the chain had grown to over 11,000 stores in 27 countries.

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.

##Problem Definition

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).

##Dataset

The company collected the transactional data of customers who purchased products from the Walmart Stores during Black Friday

importing the required libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import binom
from scipy.stats import norm
from scipy.stats import poisson
from scipy.stats import expon
from scipy.stats import lognorm
import io

##Importing the dataset Aerofit_treadmill.
# from google.colab import files

!wget
https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293
/original/walmart_data.csv
```

```
--2023-03-21 16:54:27--
https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293
/original/walmart_data.csv
Resolving d2beiqkhq929f0.cloudfront.net
(d2beiqkhq929f0.cloudfront.net)... 13.225.129.50, 13.225.129.87,
13.225.129.55, ...
Connecting to d2beiqkhq929f0.cloudfront.net
(d2beiqkhq929f0.cloudfront.net)|13.225.129.50|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 23027994 (22M) [text/plain]
Saving to: 'walmart_data.csv.1'

walmart_data.csv.1  100%[===================>]  21.96M  13.4MB/s    in
1.6s

2023-03-21 16:54:29 (13.4 MB/s) - 'walmart_data.csv.1' saved
[23027994/23027994]
```

```
walmart = pd.read_csv("walmart_data.csv")

walmart.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
```

User_ID: User ID

Product_ID: Product ID Gender: Sex of User

Age: Age in bins

Occupation: Occupation(Masked)

City_Category: Category of the City (A,B,C)

StayInCurrentCityYears: Number of years stay in current city

Marital_Status: Marital Status ProductCategory: ProductCategory (Masked)

Purchase: Purchase Amount

```
walmart.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
```

##Analysing Basic metrics

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

```
#Length of data
len(walmart)
```

```
550068
```

```
# Checking datatypes
walmart.dtypes
```

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

String or Text related columns are with Object datatype. Whereas All remaining number related columns are with int64 datatype.

```
#number of unique values in given dataset
for i in walmart.columns:
  print(i,":",walmart[i].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
```

```
# Minimum and Maximum values of Numerical columns such as -
#  User_ID ,Age,Occupation,Marital_Status,Product_Category ,Purchase 
columns
L = 
["User_ID","Age","Occupation","Marital_Status","Product_Category" ,"Pu
rchase"]
for i in L:
```

```python
    print("Maximum value of ",i,"is",walmart[i].max())
    print("Minimum value of ",i,"is",walmart[i].min())
```

```
Maximum value of  User_ID is 1006040
Minimum value of  User_ID is 1000001
Maximum value of  Age is 55+
Minimum value of  Age is 0-17
Maximum value of  Occupation is 20
Minimum value of  Occupation is 0
Maximum value of  Marital_Status is 1
Minimum value of  Marital_Status is 0
Maximum value of  Product_Category is 20
Minimum value of  Product_Category is 1
Maximum value of  Purchase is 23961
Minimum value of  Purchase is 12
```

```python
#Statistical Summary
walmart.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
```

```
#mode value of each column
print("Mode values of all columns (both categorical and numerical")
walmart.mode()
```

```
Mode values of all columns (both categorical and numerical

   User_ID Product_ID Gender    Age  Occupation City_Category  \
0  1001680  P00265242      M  26-35           4             B

  Stay_In_Current_City_Years  Marital_Status  Product_Category
Purchase
0                          1               0                 5
7011
```

2 ```CHAPTER 2: MISSING VALUE AND OUTLIER DETECTION

##Missing value Detection

```
## #checking null values in every column of our data
walmart.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              0
```

```
Purchase                      0
dtype: int64
```

Observation

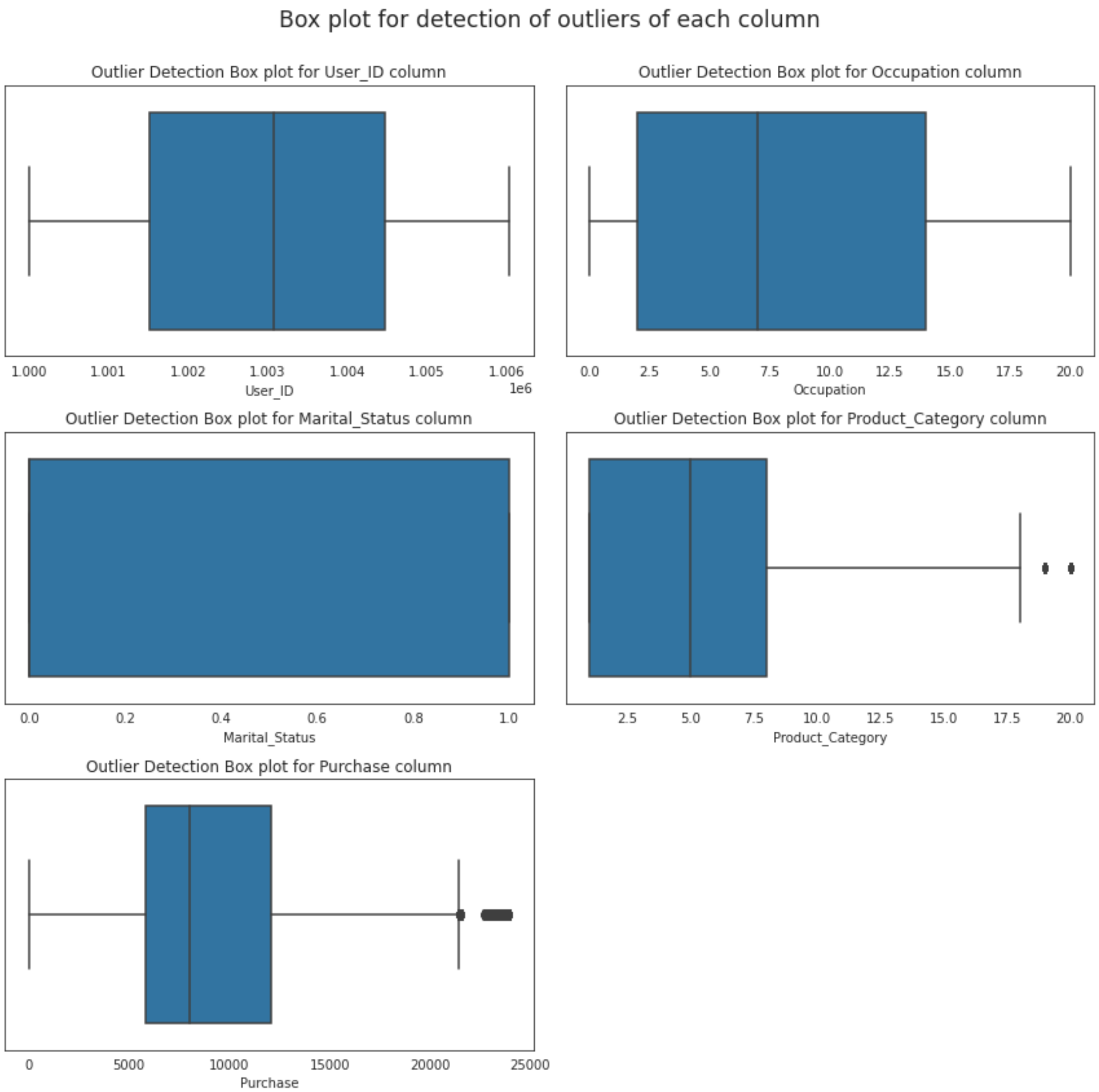
There are no missing values in the dataset.

Purchase amount might have outliers.

##**Outlier Detection**

###Boxplot for detection of outliers of each column

```
fig = plt.figure(figsize = (12,12))
fig.suptitle("Box plot for detection of outliers of each column\
n",fontsize ="xx-large" )
iloc_positions_of_numerical_columns = [0,4,7,8,9]
k = 1
for i in iloc_positions_of_numerical_columns:
  plt.subplot(3,2,k)
  plt.title("Outlier Detection Box plot for {}
column".format(walmart.columns[i]))
  plt.xlabel(walmart.columns[i])
  sns.boxplot(data=walmart, x = walmart.iloc [ :,i],orient="h")
  k = k+1
plt.tight_layout()
plt.show()
```



Tracking the amount spent per transaction of all the 50 million female customers, and all the 50 million male customers, calculate the average, and conclude the results.

3 CHAPTER 3: NON-GRAPHICAL ANALYSIS

3.1 Value_counts

```
# Checking the occurences of each of the column.
for i in walmart.columns:
  print(i,walmart[i].value_counts(),sep="\n")
  print("\n")
```

```
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
```

3.2 **Unique Attributes**

```python
# Checking the unique attributes for all columns
for i in walmart.columns:
  print(i,walmart[i].unique(),sep="\n")
  print("\n")
```

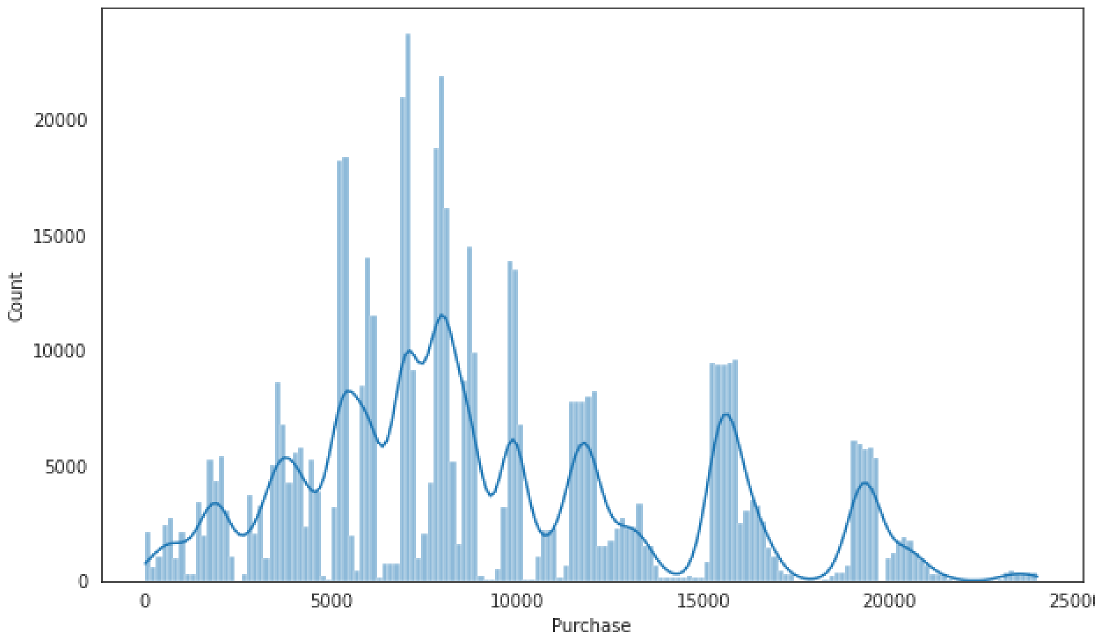
```
User_ID
[1000001 1000002 1000003 ... 1004113 1005391 1001529]


Product_ID
['P00069042' 'P00248942' 'P00087842' ... 'P00370293' 'P00371644'
 'P00370853']


Gender
['F' 'M']
```

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


Occupation
[10 16 15  7 20  9  1 12 17  0  3  4 11  8 19  2 18  5 14 13  6]


City_Category
['A' 'C' 'B']


Stay_In_Current_City_Years
['2' '4+' '3' '1' '0']


Marital_Status
[0 1]


Product_Category
[ 3  1 12  8  5  4  2  6 14 11 13 15  7 16 18 10 17  9 20 19]


Purchase
[ 8370 15200  1422 ...   135   123   613]
```

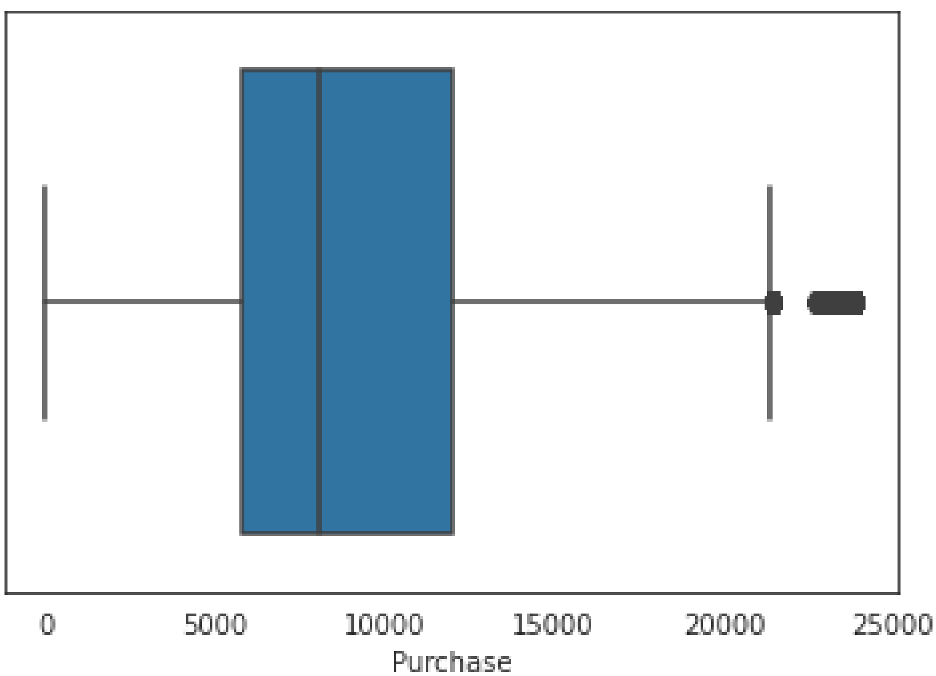
**4 CHAPTER 4: VISUAL ANALYSIS

###univariant subplot

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




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



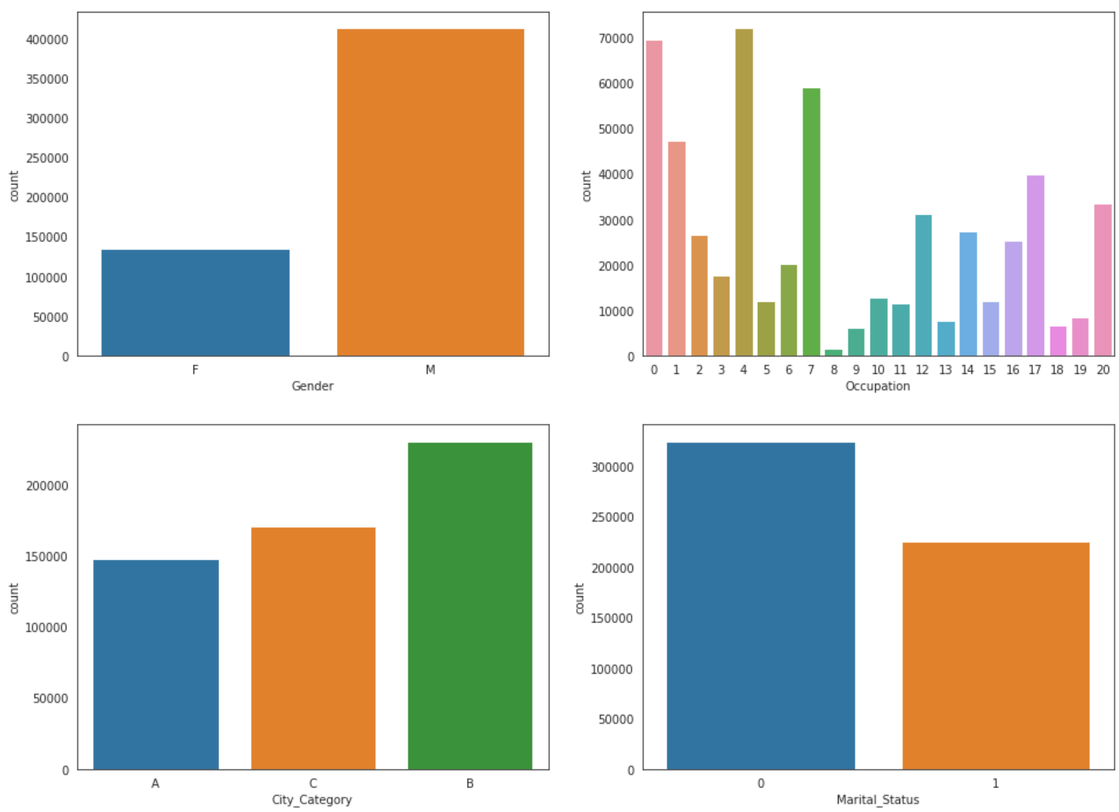
Observation

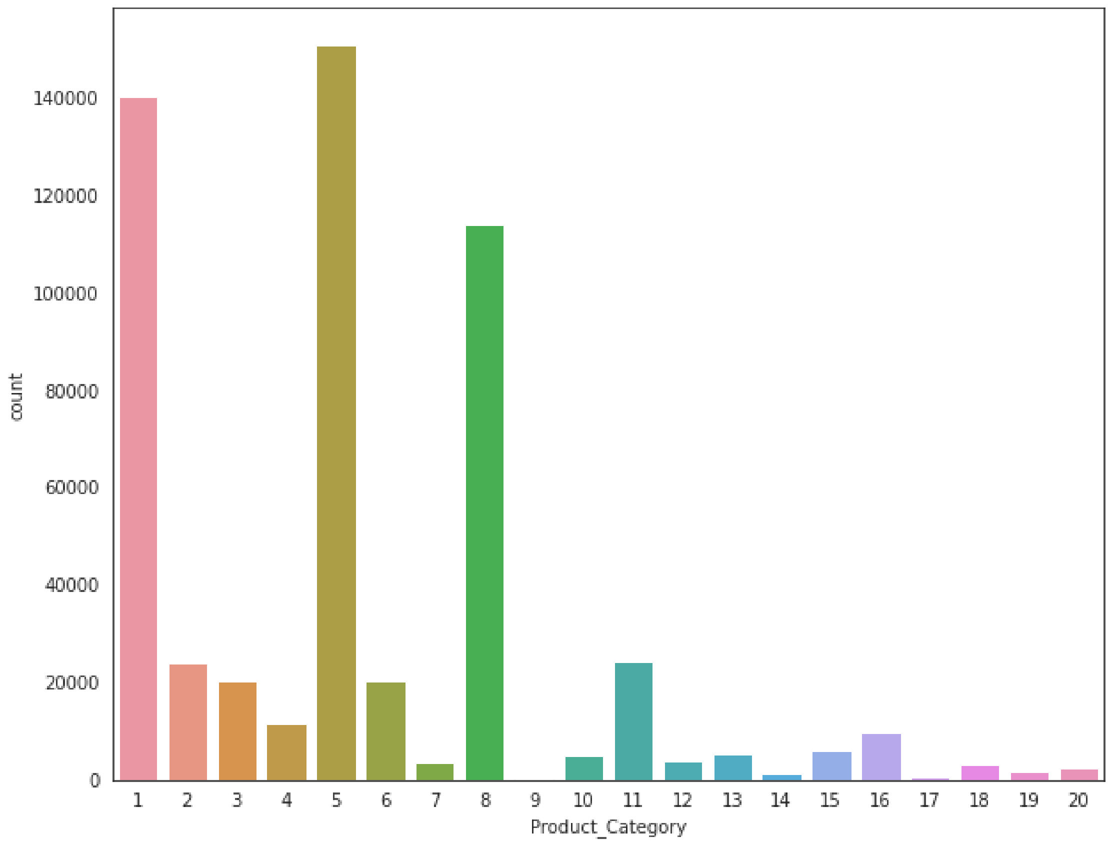
- Purchase is having outliers.

Understanding the distribution of data for the categorical variables

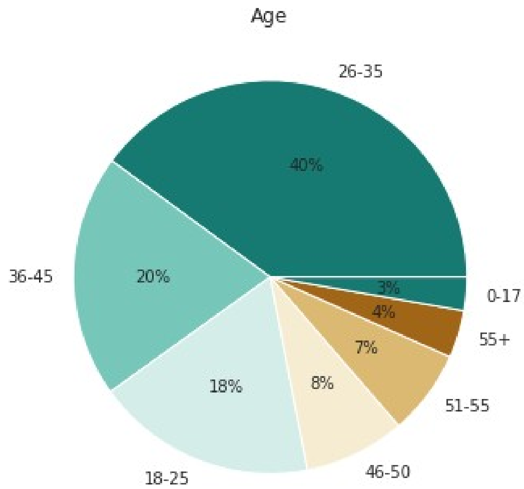
- Gender
- Age
- Occupation
- City_Category
- Stay_In_Current_City_Years
- Marital_Status
- Product_Category

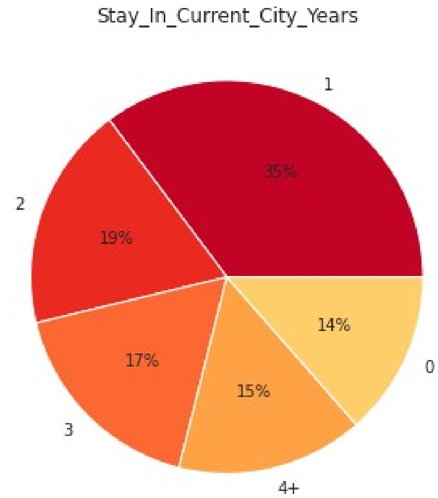
```
categorical_cols = ['Gender',
'Occupation','City_Category','Marital_Status','Product_Category']

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

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





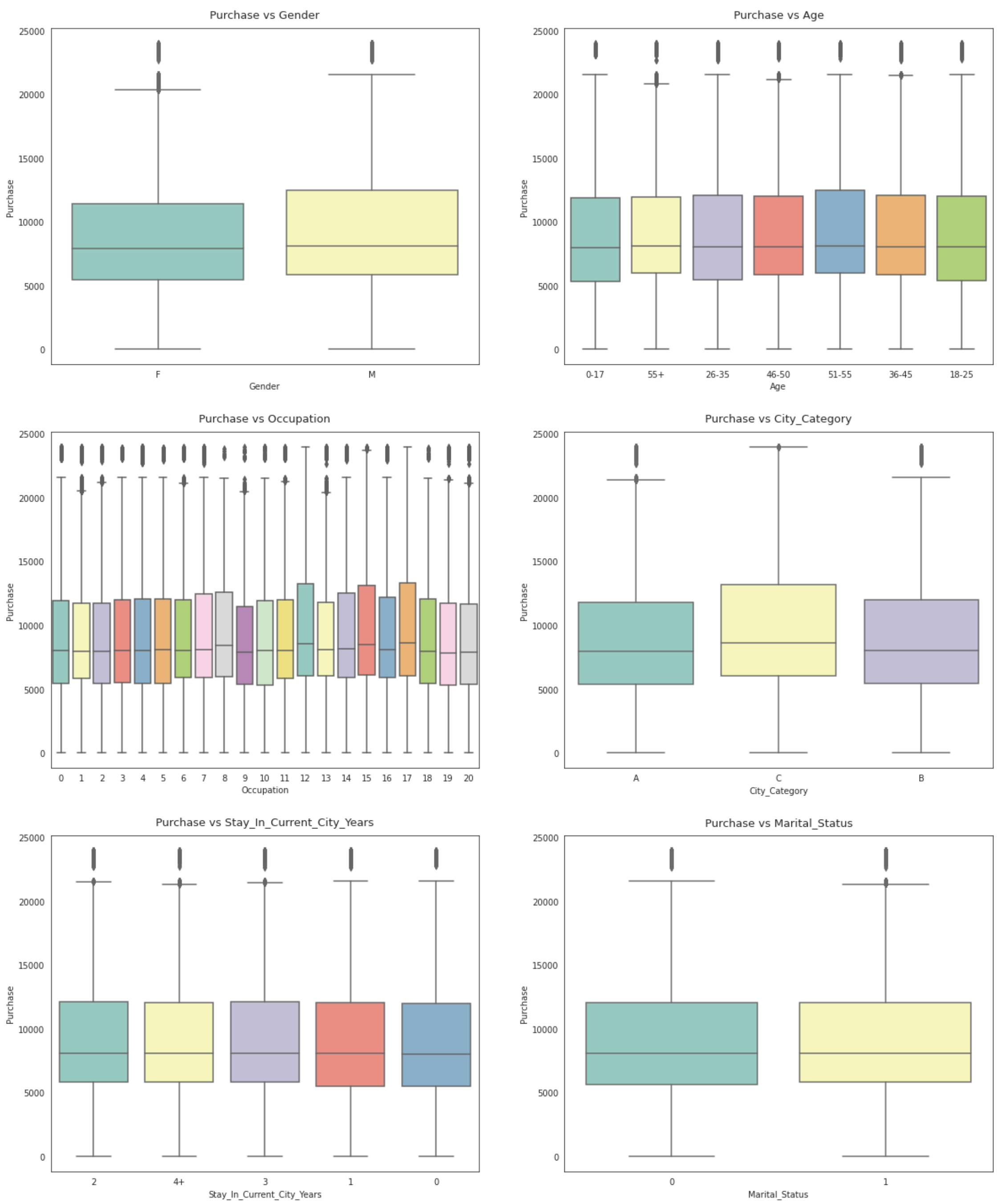

```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(12, 8))

data = walmart['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 =
walmart['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()
```



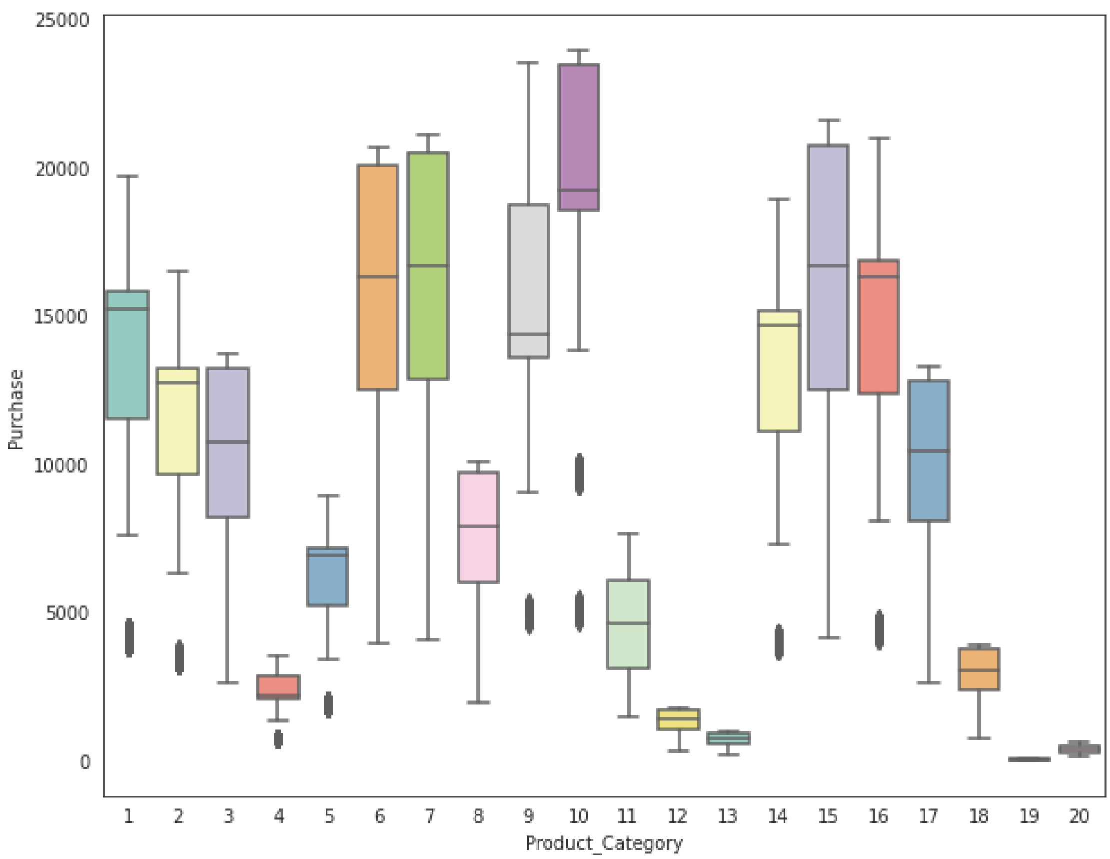


Bivariant Analysis

```
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=walmart, 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=walmart, y='Purchase', x=attrs[-1], palette='Set3')
plt.show()
```

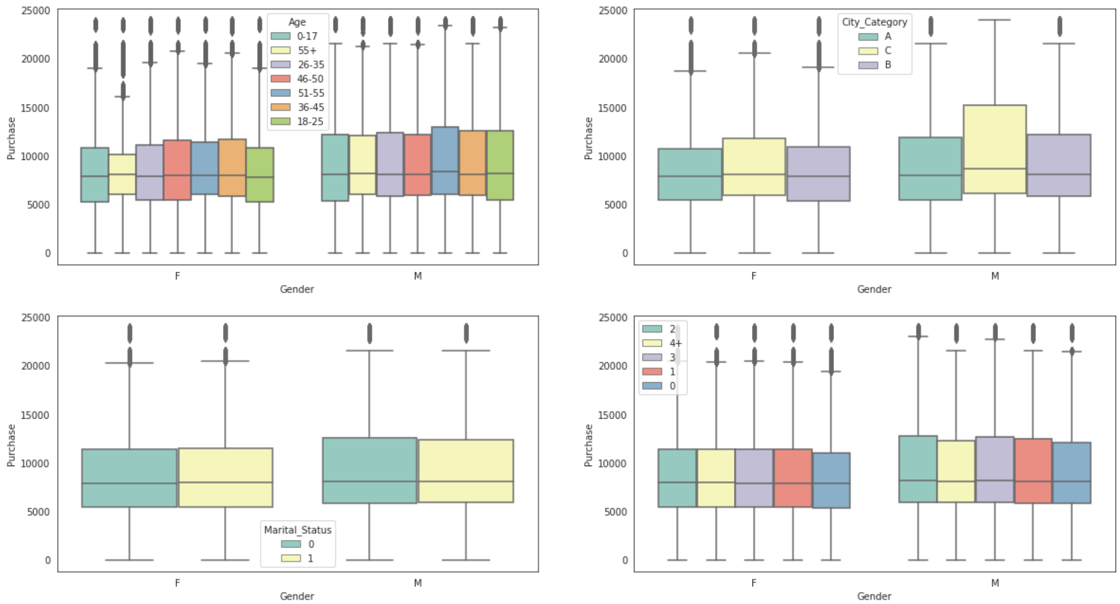


Purchase vs Gender
Purchase
Gender
F
M
Purchase vs Age
Age
0-17
55+
26-35
46-50
51-55
36-45
18-25
Purchase vs Occupation
Occupation
Purchase vs City_Category
City_Category
A
C
B
Purchase vs Stay_In_Current_City_Years
Stay_In_Current_City_Years
2
4+
3
1
0
Purchase vs Marital_Status
Marital_Status
0
1
25000
20000
15000
10000
5000
0



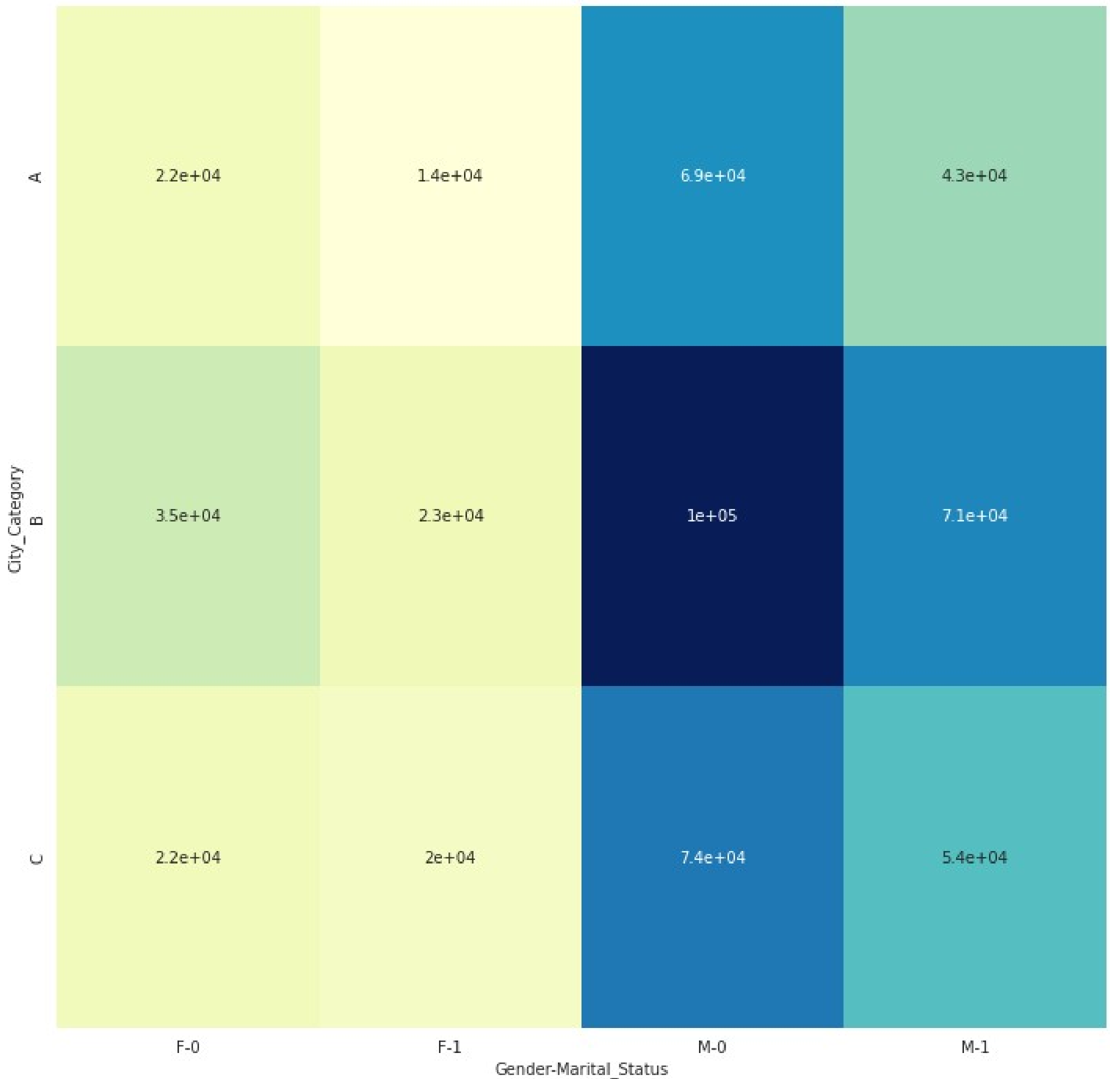
MultiVariate Analysis

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

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

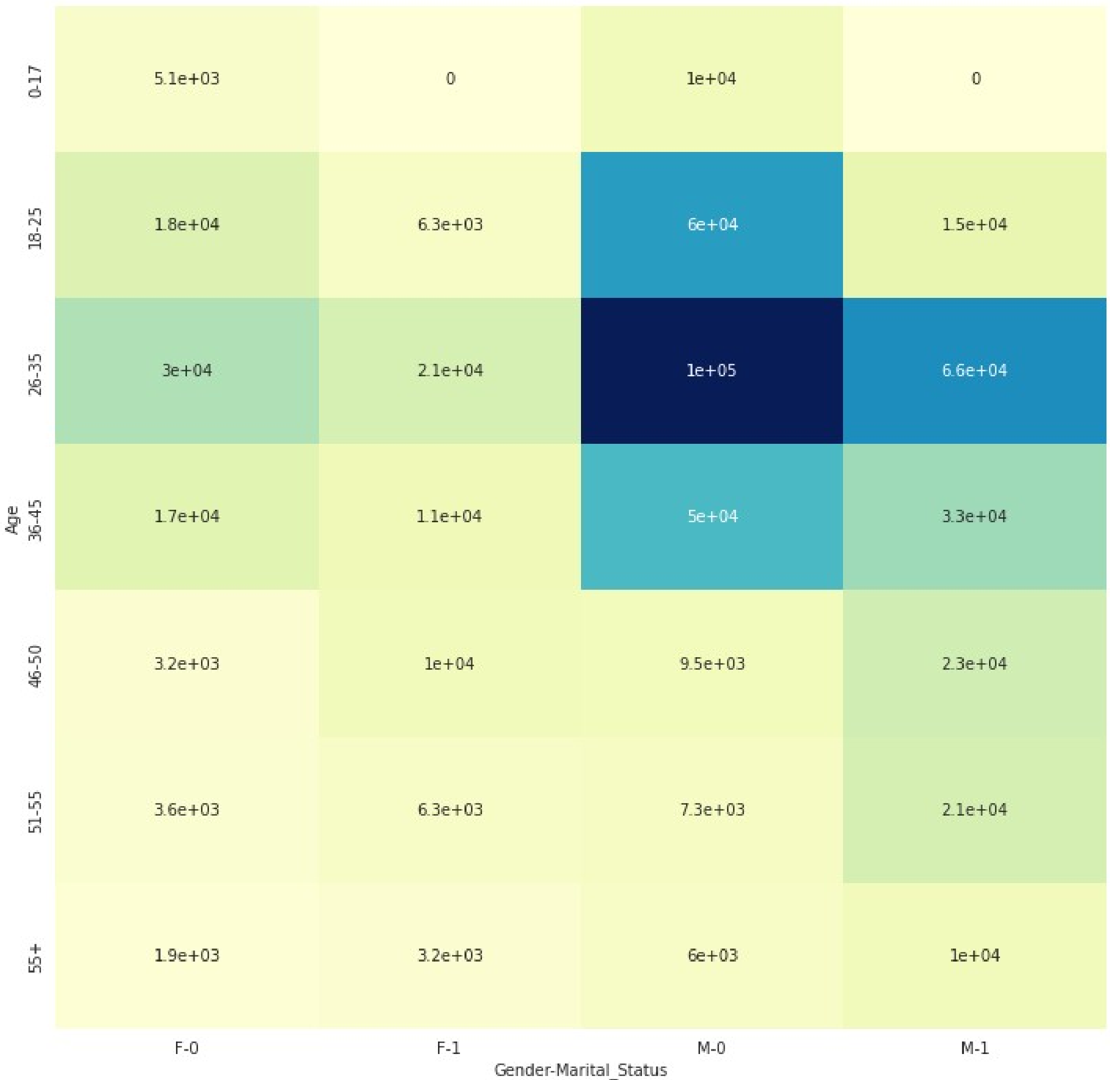


Heatmaps

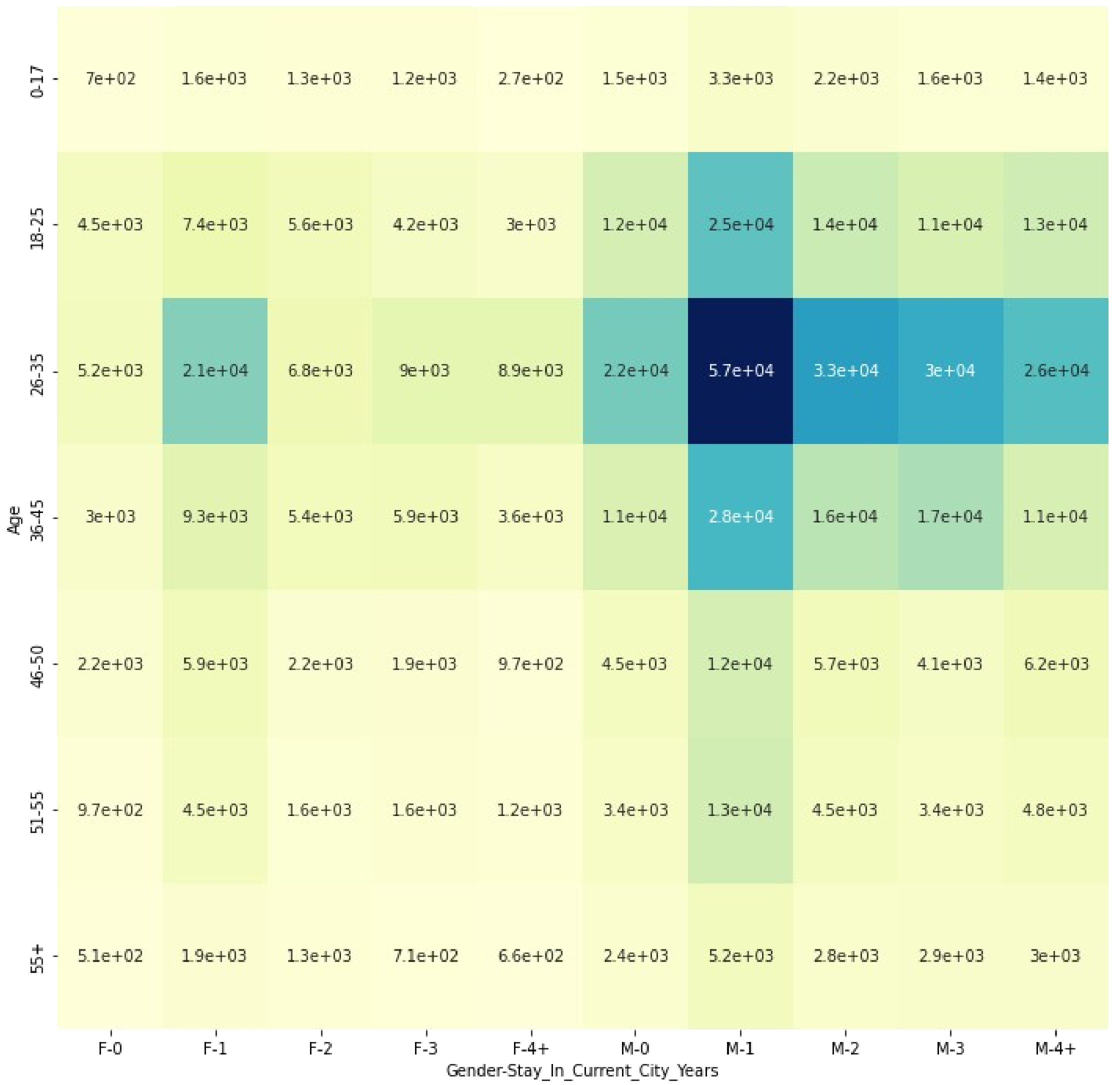
```
fig = plt.figure(figsize = (12,12))
sns.heatmap(pd.crosstab([walmart["City_Category"]],
[walmart["Gender"],walmart["Marital_Status"]]),cmap="YlGnBu",annot
=True,cbar = False)
plt.show()
```



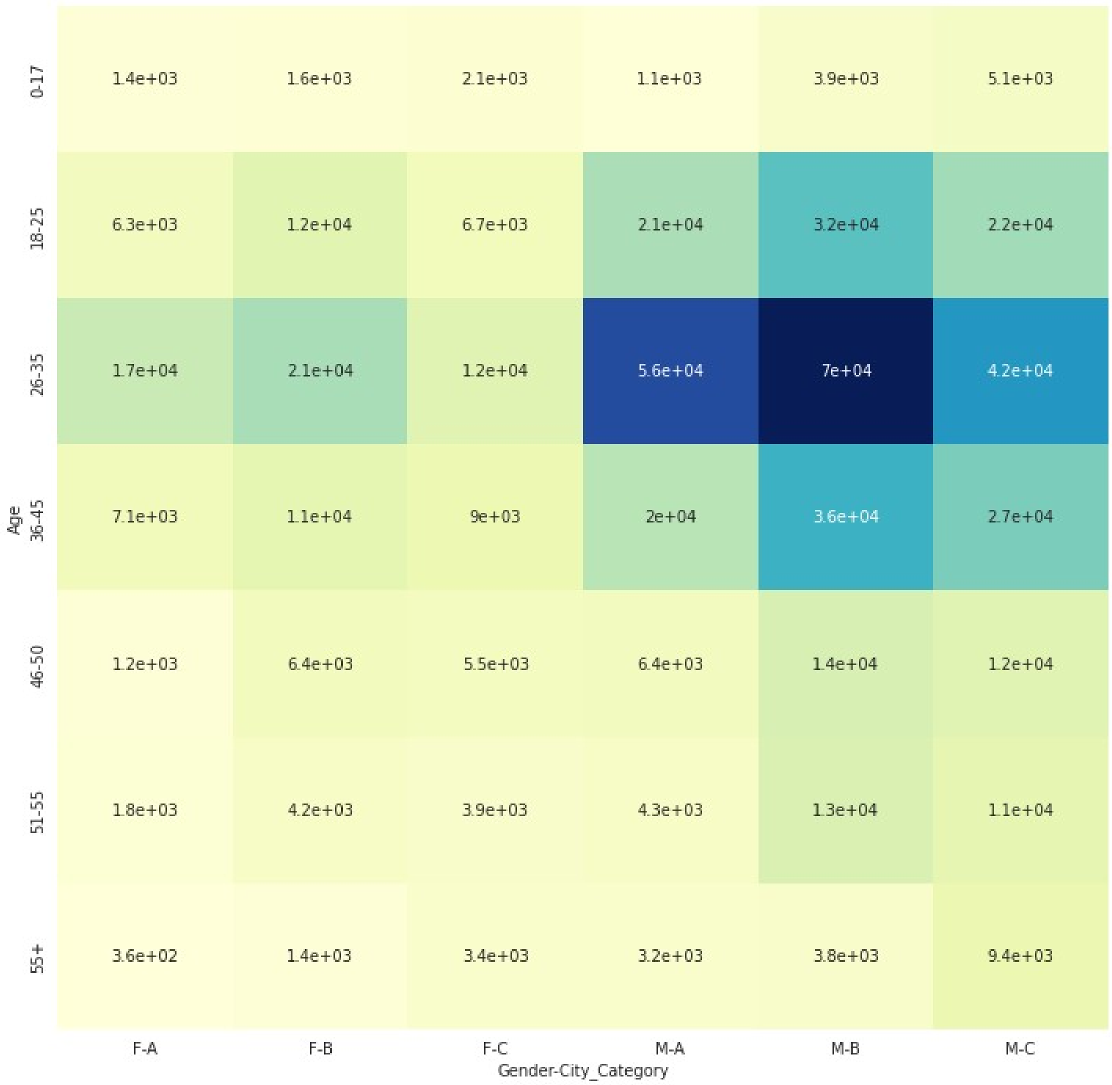

```
fig = plt.figure(figsize = (12,12))
sns.heatmap(pd.crosstab([walmart["Age"]],
[walmart["Gender"],walmart["Marital_Status"]]),cmap="YlGnBu",annot
=True,cbar = False)
plt.show()
```



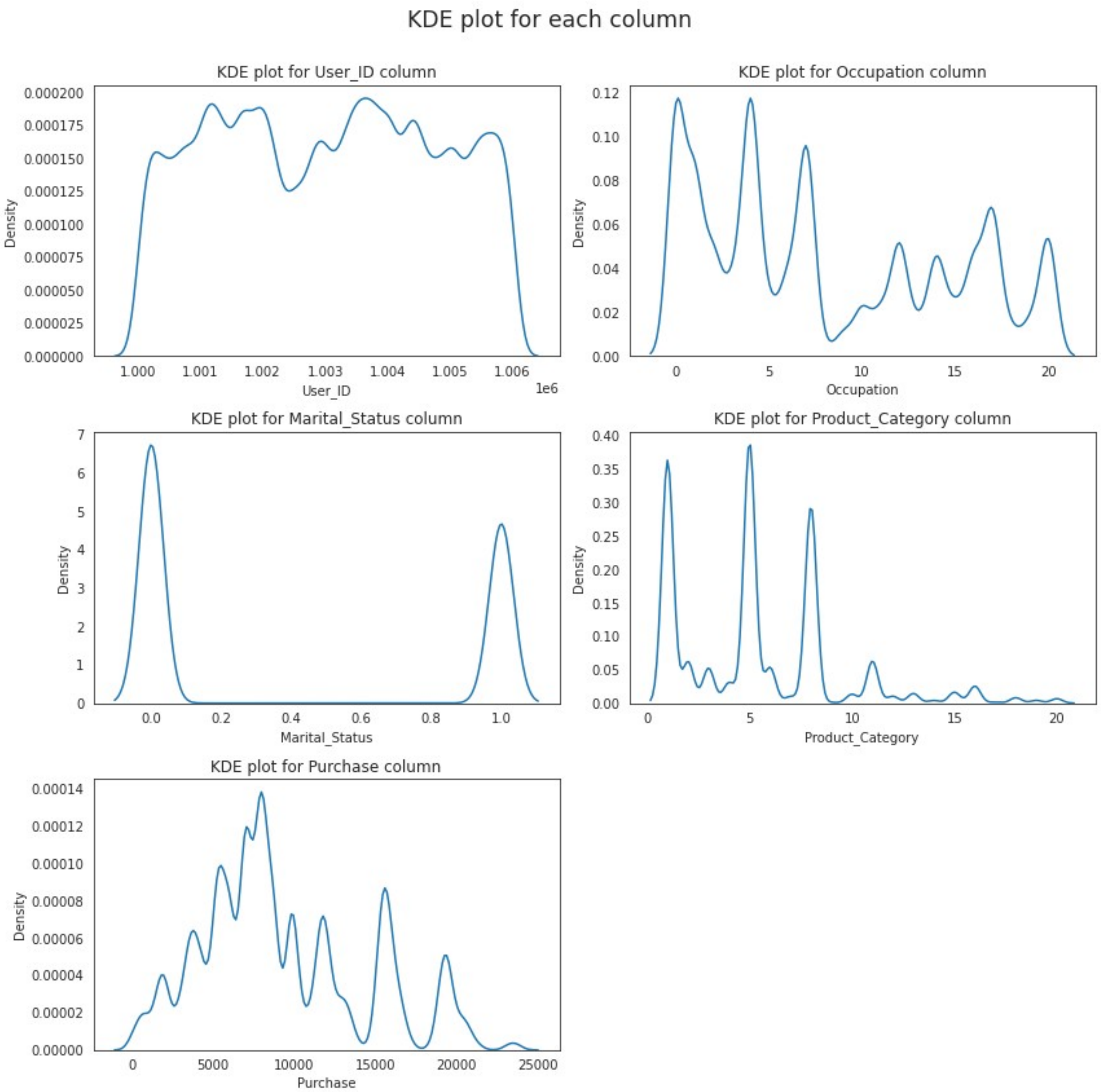

```
fig = plt.figure(figsize = (12,12))
sns.heatmap(pd.crosstab([walmart["Age"]],
[walmart["Gender"],walmart["Stay_In_Current_City_Years"]]),cmap="YlGnB
u",annot =True,cbar = False)
plt.show()
```



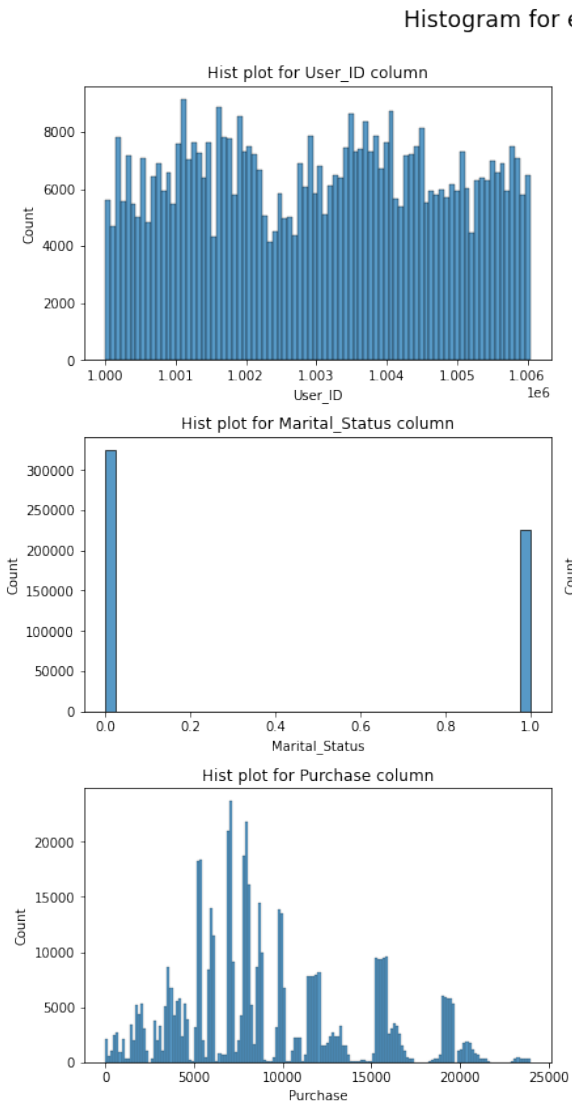

```
fig = plt.figure(figsize = (12,12))
sns.heatmap(pd.crosstab([walmart["Age"]],
[walmart["Gender"],walmart["City_Category"]]),cmap="YlGnBu",annot
=True,cbar = False)
plt.show()
```




```
#create a kde plot
fig = plt.figure(figsize = (12,12))
fig.suptitle("KDE plot for each column\n",fontsize = "xx-large" )
iloc_positions_of_numerical_columns = [0,4,7,8,9]
k = 1
for i in iloc_positions_of_numerical_columns:
  plt.subplot(3,2,k)
  plt.title("KDE plot for {} column".format(walmart.columns[i]))
  plt.xlabel(walmart.columns[i])
  sns.kdeplot(data=walmart, x = walmart.iloc[:,i])
  k = k+1
plt.tight_layout()
plt.show()
```

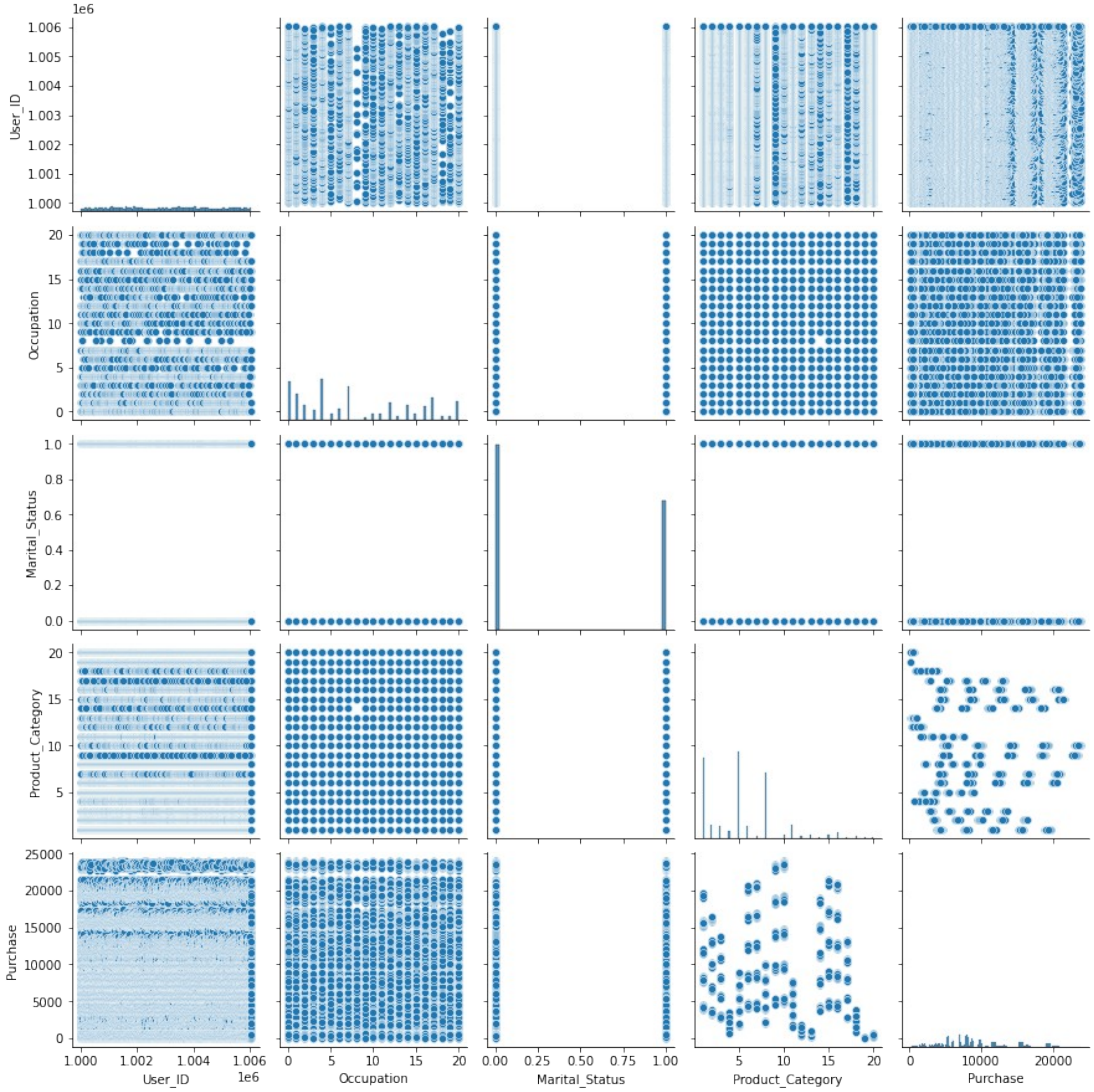



```
#create a histogram
fig = plt.figure(figsize = (12,12))
fig.suptitle("Histogram for each column\n",fontsize = "xx-large" )
iloc_positions_of_numerical_columns = [0,4,7,8,9]
k = 1
for i in iloc_positions_of_numerical_columns:
  plt.subplot(3,2,k)
  plt.title("Hist plot for {} column".format(walmart.columns[i]))
  plt.xlabel(walmart.columns[i])
  sns.histplot(data=walmart, x = walmart.iloc[:,i])
  k = k+1
plt.tight_layout()
plt.show()
```




```
plt.figure(figsize = (50,50))
sns.pairplot(data = walmart)
plt.show()
```

```
<Figure size 3600x3600 with 0 Axes>
```

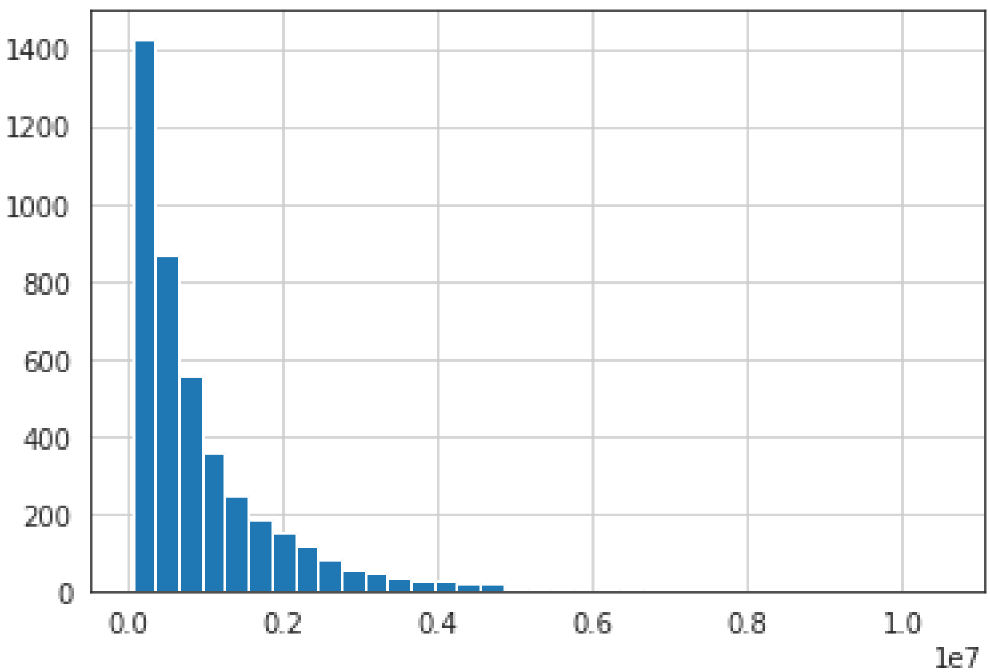


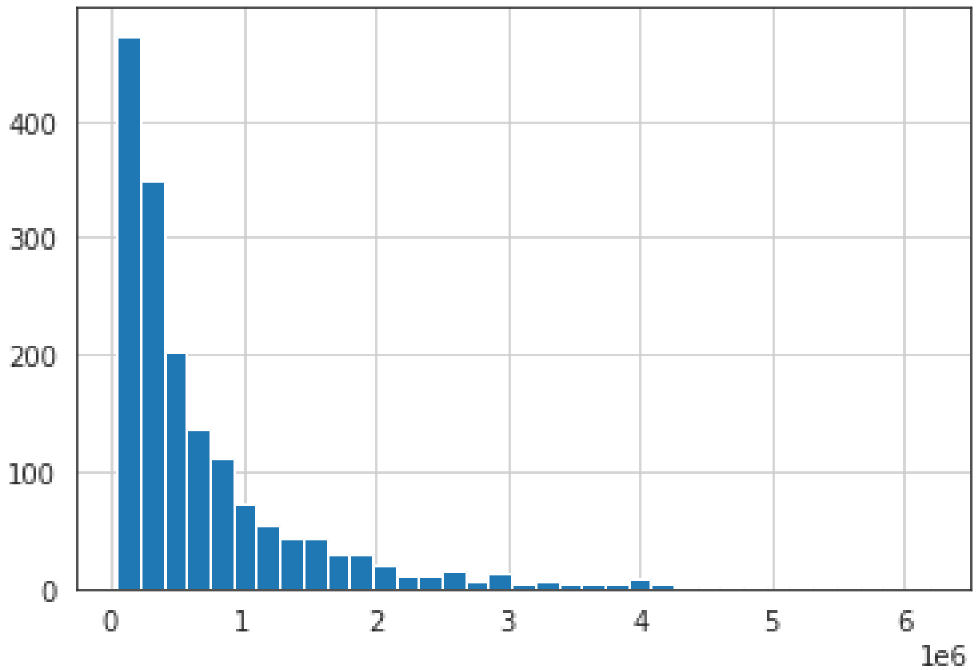
Average amount spend per customer for Male and Female

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

```
      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 x 3 columns]

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

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']

male_df
```

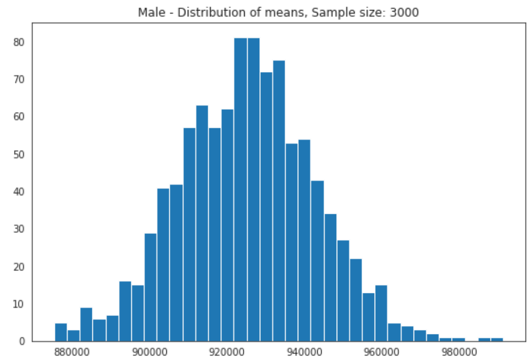
```
      User_ID Gender  Purchase
1     1000002      M    810472
2     1000003      M    341635
3     1000004      M    206468
4     1000005      M    821001
6     1000007      M    234668
...       ...    ...       ...
5880  1006030      M    737361
5882  1006032      M    517261
5883  1006033      M    501843
```

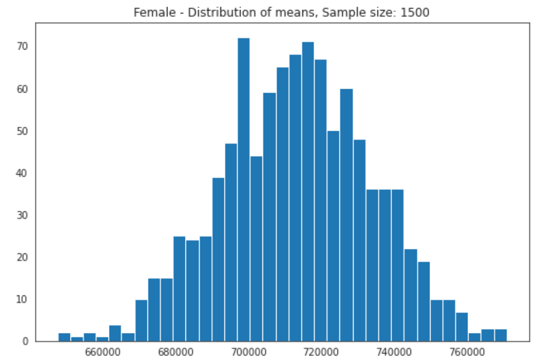
```
5884  1006034      M    197086
5890  1006040      M   1653299

[4225 rows x 3 columns]
```

```
female_df
```

```
      User_ID Gender  Purchase
0     1000001      F    334093
5     1000006      F    379930
9     1000010      F   2169510
10    1000011      F    557023
15    1000016      F    150490
...       ...    ...       ...
5885  1006035      F    956645
5886  1006036      F   4116058
5887  1006037      F   1119538
5888  1006038      F     90034
5889  1006039      F    590319

[1666 rows x 3 columns]
```

```
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.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:
925221.52
Population mean - Mean of sample means of amount spend for Female:
713351.97

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

Observation

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

- Average amount spend by male customers is 9,26,341.86
- 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))
```

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

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

- 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)

Doing the same activity for married vs unmarried

```
amt_df
```

```
      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 x 3 columns]
```

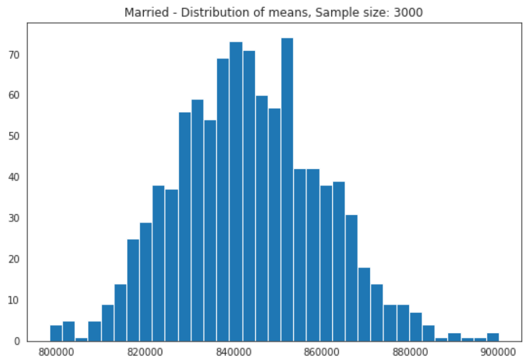
```
amt_df = walmart.groupby(['User_ID', 'Marital_Status'])
[['Purchase']].sum()
amt_df = amt_df.reset_index()
amt_df
```

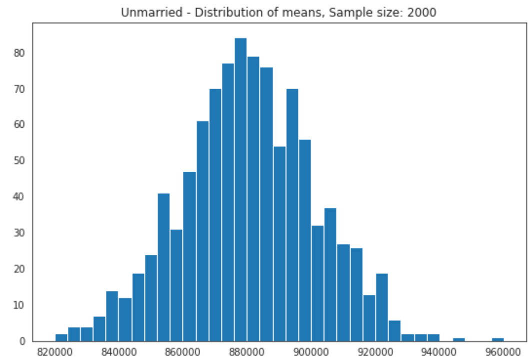
```
      User_ID  Marital_Status  Purchase
0     1000001               0    334093
1     1000002               0    810472
2     1000003               0    341635
3     1000004               1    206468
4     1000005               1    821001
...       ...             ...       ...
5886  1006036               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 = 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:
843442.28
Population mean - Mean of sample means of amount spend for Unmarried:
880501.14

Married - Sample mean: 843526.80 Sample std: 935352.12
Unmarried - Sample mean: 880575.78 Sample std: 949436.25
```

```
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: (806668.83, 880384.76)
Unmarried confidence interval of means: (848741.18, 912410.38)
```

Calculating the average amount spent by Age

```
amt_df = walmart.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("For age {} --> confidence interval of means: ({:.2f},
{:.2f})".format(val, lower_lim, upper_lim))
```

```
For age 26-35 --> confidence interval of means: (945034.42,
1034284.21)
For age 36-45 --> confidence interval of means: (823347.80, 935983.62)
```

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

Insights

1. Total of 20 product categories are there
2. There are 20 different types of Occupation and Product_Category
3. More users belong to B City_Category
4. Product_Category - 1, 5, 8, & 11 have highest purchasing frequency.
5. More users are Single as compare to Married
6. 80% of the users are between the age 18-50 (40%: 26-35, 18%: 18-25, 20%: 36-45)
7. 75% of the users are Male and 25% are Female
8. 60% Single, 40% Married
9. 35% Staying in the city from 1 year, 18% from 2 years, 17% from 3 years
10. There are 20 diffrenent types of occupations in the city.
11. Average amount spend by Male customers: 925344.40
12. Average amount spend by Female customers: 712024.39

Confidence Interval by Gender

Now using the Central Limit Theorem for the population:

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

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

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: (806668.83, 880384.76)

Unmarried confidence interval of means: (848741.18, 912410.38)

Confidence Interval by Age

For age 0-17 --> confidence interval of means: (527662.46, 710073.17)

For age 18-25 --> confidence interval of means: (801632.78, 908093.46)

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 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)

Recommendations

Men spent more money than women, So company should focus on retaining the male customers and getting more male customers.

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.

Unmarried customers spend more money than married customers, So company should focus on acquisition of Unmarried customers.

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.

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.