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

from scipy.stats import norm,ttest_1samp,ttest_ind
```

```
! gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094
To: /content/walmart_data.csv?1641285094
100% 23.0M/23.0M [00:00<00:00, 303MB/s]
```

```
#Reading the DATA using pd.read_csv("File_Location")
data=pd.read_csv("walmart_data.csv?1641285094")
Dataframe=pd.DataFrame(data)
df=data
```

```
#Head function is used to get the top 5 rows from the data
df.head()
```

	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

1.Defining Problem Statement and Analyzing basic metrics (10 Points)

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

```
#Shape function is used to get the Dimensions of the DATA
df.shape
```

```
(550068, 10)
```

```
#TYPE function is used to see the data_types of the each column of the data
df.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
```

```
#Describe function will give the statistical summary of the each column like meam,max,min,std e.t.c
df.describe()
```

	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
75%	1.004478e+06	14.000000	1.000000	8.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	23961.000000

```
#Tail function used to get the buttom 5 rows for the data
df.tail()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Produc
550063	1006033	P00372445	M	51-55	13	B	1	1	
550064	1006035	P00375436	F	26-35	1	C	3	0	
550065	1006036	P00375436	F	26-35	15	B	4+	1	

2. **Non-Graphical Analysis: Value counts and unique attributes**

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

```
5891
```

```
df["Product_ID"].nunique()
```

```
3631
```

```
df["Gender"].value_counts()
```

```
M    414259
F    135809
Name: Gender, dtype: int64
```

```
df["Age"].value_counts()
```

```
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
```

```
df["Occupation"].value_counts().sort_values()
```

```
8      1546
9      6291
18     6622
13     7728
19     8461
11    11586
15    12165
5     12177
10    12930
3     17650
6     20355
16    25371
2     26588
14    27309
12    31179
20    33562
17    40043
1     47426
7     59133
0     69638
4     72308
Name: Occupation, dtype: int64
```

```
df["City_Category"].value_counts()
```

```
B    231173
C    171175
A    147720
Name: City_Category, dtype: int64
```

```
df["Stay_In_Current_City_Years"].value_counts()
```

```
1     193821
2     101838
3      95285
4+     84726
0      74398
Name: Stay_In_Current_City_Years, dtype: int64
```

```
df["Marital_Status"].value_counts()
```

```
0    324731
1    225337
Name: Marital_Status, dtype: int64
```

```
df["Product_Category"].value_counts()
```

```
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
```

```
df["Purchase"].sum()
```

```
5095812742
```

3. Visual Analysis - Univariate & Bivariate

3.1 For continuous variable(s): Distplot, countplot, histogram for univariate analysis

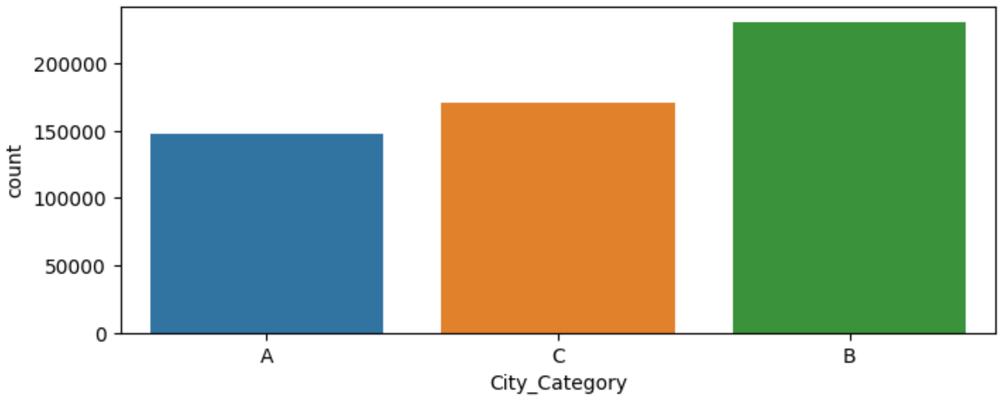
3.2 For categorical variable(s): Boxplot

3.3 For correlation: Heatmaps, Pairplots

```
df.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Cat
0	1000001	P00069042	F	0-17	10	A	2	0	
1	1000001	P00248942	F	0-17	10	A	2	0	

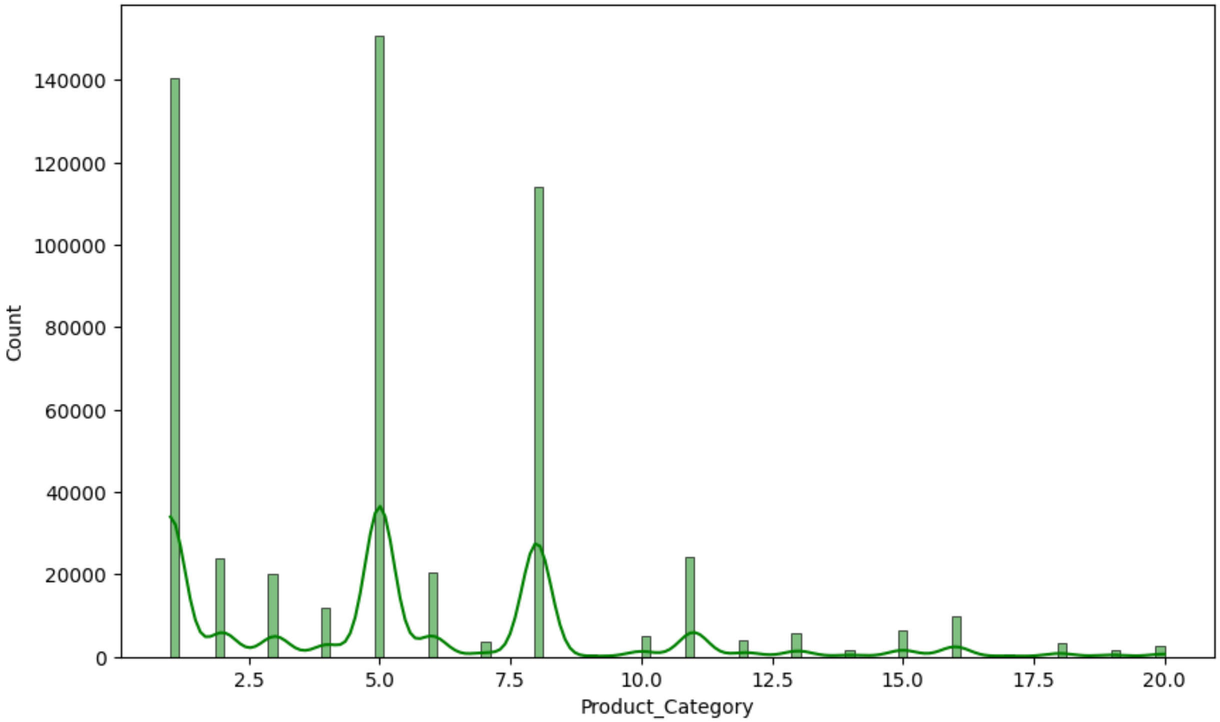
```
plt.figure(figsize=(8,3))
sns.countplot(x="City_Category",data=df)
plt.show()
```




```
df["Product_Category"].value_counts().sort_values()
```

```
9        410
17       578
14      1523
19      1603
20      2550
18      3125
7       3721
12      3947
10      5125
13      5549
15      6290
16      9828
4      11753
3      20213
6      20466
2      23864
11     24287
8     113925
1     140378
5     150933
Name: Product_Category, dtype: int64
```

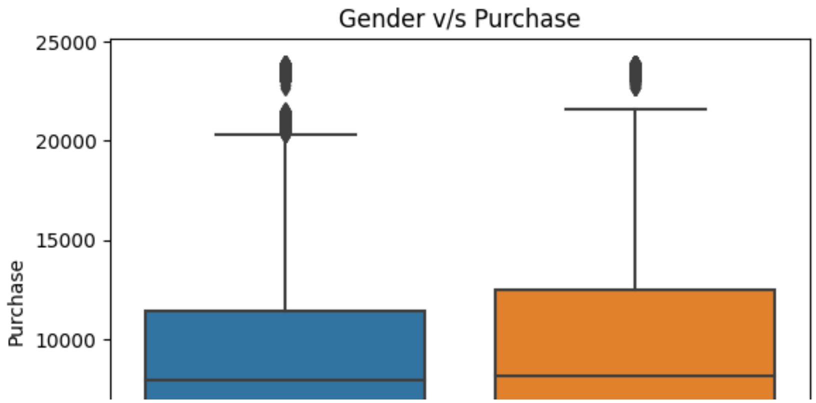
```
plt.figure(figsize=(10,6))
sns.histplot(df["Product_Category"],color="g",kde=True)
plt.show()
```




```
print(df[df["Gender"]=="F"]["Purchase"].max())
print(df[df["Gender"]=="M"]["Purchase"].max())
```

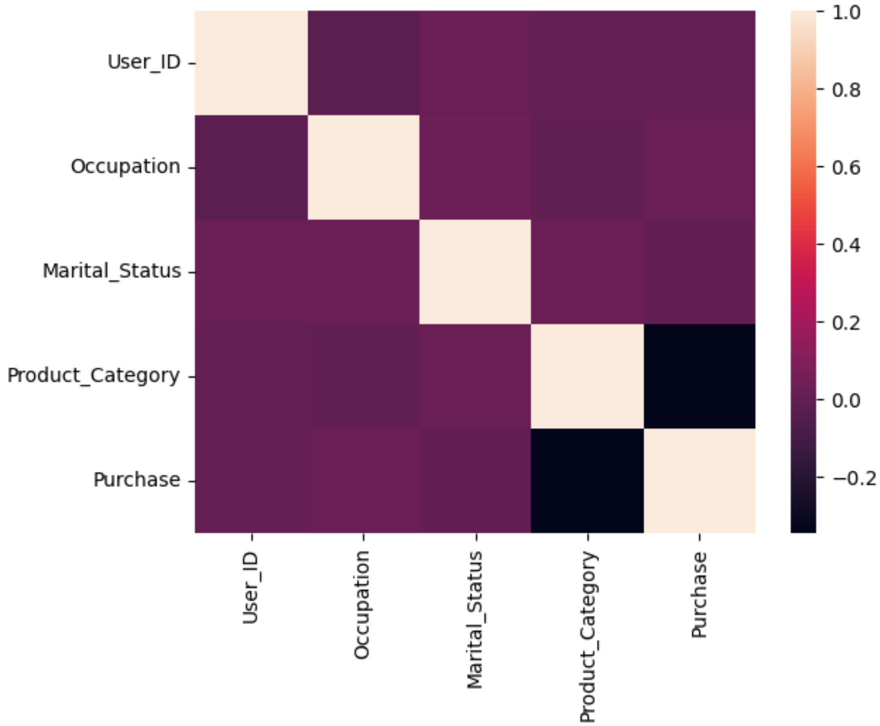
```
23959
23961
```

```
plt.title("Gender v/s Purchase")
sns.boxplot(data=df,x="Gender",y="Purchase")
plt.show()
```




```
sns.heatmap(df.corr())
```

```
<ipython-input-25-aa4f4450a243>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In
  sns.heatmap(df.corr())
<Axes: >
```




```
#sns.pairplot(df)
```

Question 2. Missing Values and Outlier detection

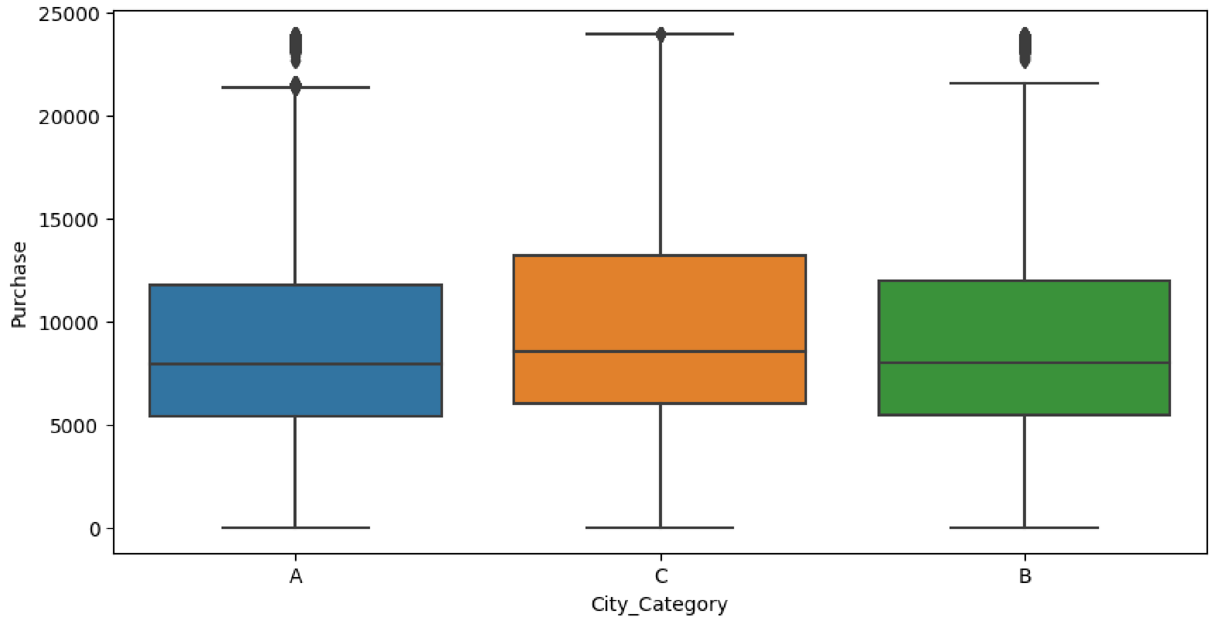
```
df.isna().sum()
```

```
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
```

There is no Missing values from the Data as isna.sum()equal to ZERO.

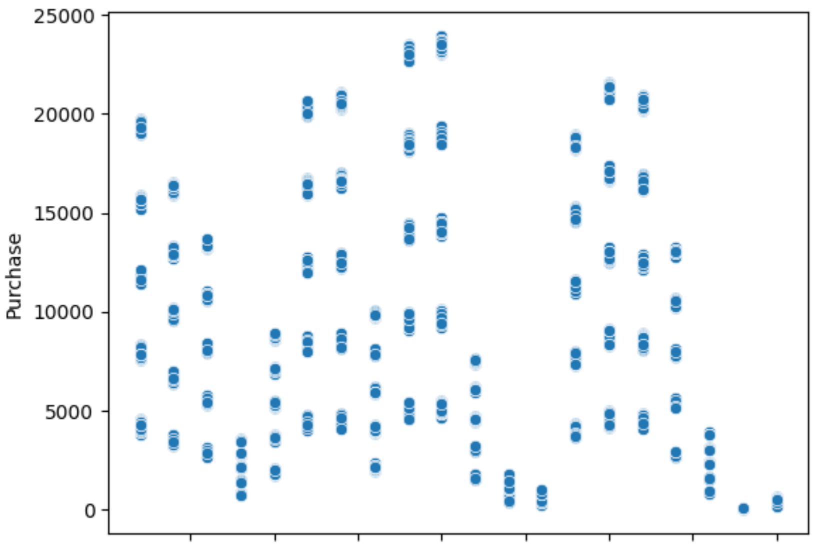
```
plt.figure(figsize=(10,5))
sns.boxplot(y=df["Purchase"],x=df["City_Category"])
```

```
<Axes: xlabel='City_Category', ylabel='Purchase'>
```




```
#SCATTERPLOT is used to detect the outliers.
sns.scatterplot(x=df["Product_Category"],y=df["Purchase"],data=df)
```

```
<Axes: xlabel='Product_Category', ylabel='Purchase'>
```



3 Business insights based on Non-Graphical and Visual Analysis

```
df.head(2)
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Cat
0	1000001	P00069042	F	0-17	10	A	2	0	

```
Gender_count=df[["User_ID","Gender"]].value_counts()
```

We could see that 5891 unique customers have brought the products on Black Friday using Walmart store.

```
df[df["Gender"]=="M"][["User_ID","Gender"]].value_counts()
```

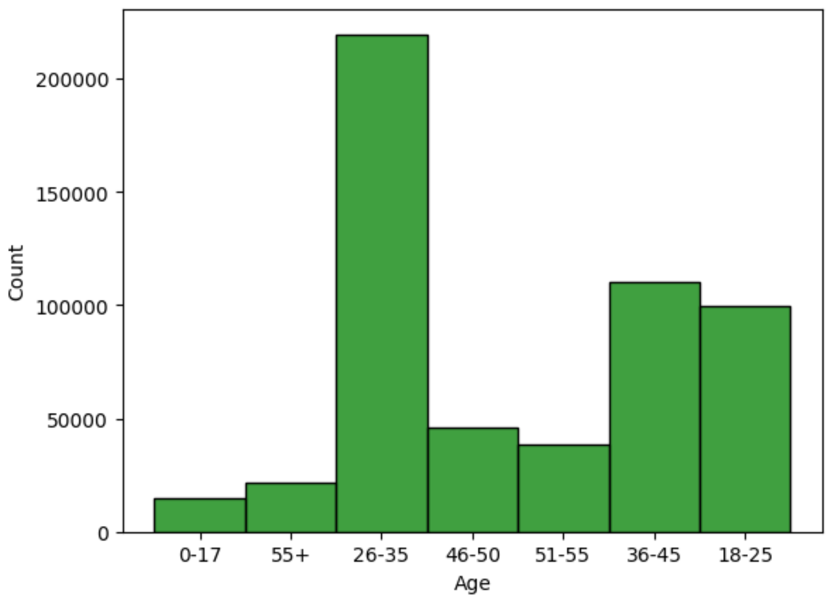
```
User_ID  Gender
1001680  M         1026
1004277  M          979
1001941  M          898
1001181  M          862
1000889  M          823
                   ...
1005391  M            7
1005608  M            7
1005810  M            7
1002690  M            7
1000708  M            6
Length: 4225, dtype: int64
```

```
df[df["Gender"]=="F"][["User_ID","Gender"]].value_counts()
```

```
User_ID  Gender
1001150  F         752
1001088  F         680
1003224  F         622
1003539  F         617
1005643  F         573
                  ...
1002965  F           8
1005904  F           8
1002488  F           8
1003291  F           8
1004991  F           7
Length: 1666, dtype: int64
```

out of 5891 customers Males customers are 4225 and Female customers are 1666

```
sns.histplot(df["Age"],color="g")
plt.show()
```



Age between 26-35 Customers are involing more percentage in sales compare to remanining age group customers.

```
df.groupby("Age")["Purchase"].sum().sort_values()
```

```
Age
0-17      134913183
55+       200767375
51-55     367099644
```

```
46-50     420843403
18-25     913848675
36-45    1026569884
26-35    2031770578
Name: Purchase, dtype: int64
```

From the above Data we could see that 26 -35 Customers are buying more sales comapre to remaining Age groups.

```
print(df.groupby("Marital_Status")["Purchase"].sum())
```

```
Marital_Status
0    3008927447
1    2086885295
Name: Purchase, dtype: int64
```

From the above Data we could see that unmarried Customers are buying more sales comapre to Married couples.

4A Are women spending more money per transaction than men? why or why not?

```
df.groupby("Gender")["Purchase"].mean()
```

```
Gender
F    8734.565765
M    9437.526040
Name: Purchase, dtype: float64
```

```
df[df["Gender"]=="M"].groupby("Marital_Status")["Purchase"].sum()
```

```
Marital_Status
0    2324773320
1    1584806780
Name: Purchase, dtype: int64
```

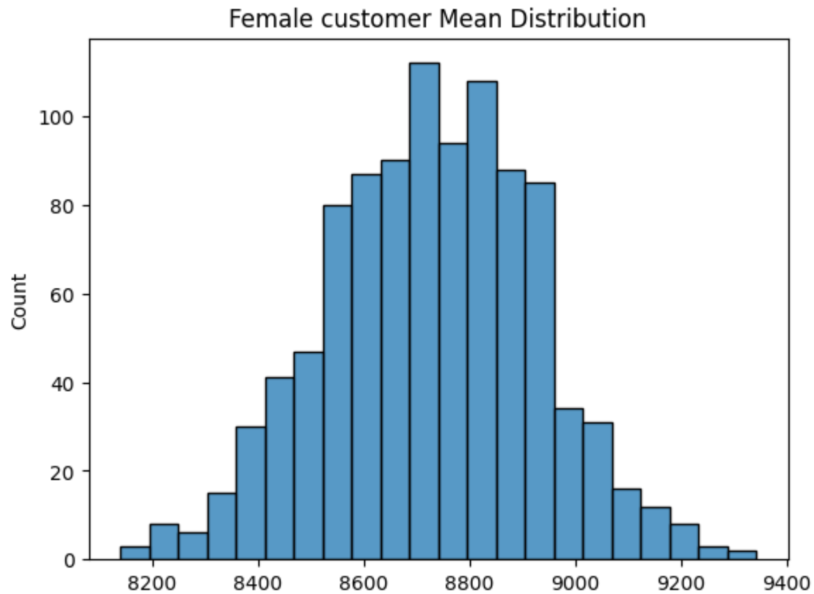
Men are Spending more money per transaction than Women because most of the mens are unmarried so they are spending more money.

4b Confidence interval and distribution of the mean of the expenses by female and male customers

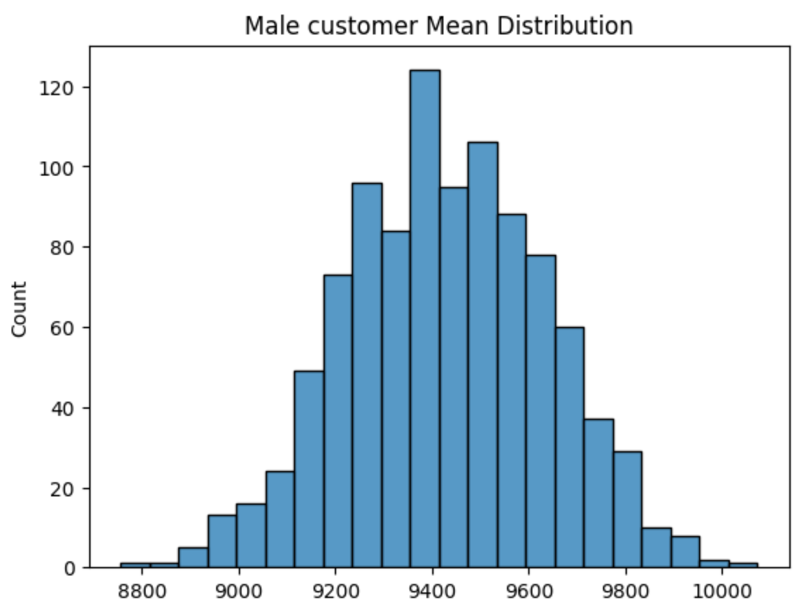
```
Female_purchase=df[df["Gender"]=="F"]["Purchase"]
Male_purchase=df[df["Gender"]=="M"]["Purchase"]
```

```
Female_sample_mean=[Female_purchase.sample(600).mean() for i  in range(1000)]
Male_sample_mean=[Male_purchase.sample(600).mean() for i  in range(1000)]
```

```
plt.title("Female customer Mean Distribution")
sns.histplot(Female_sample_mean)
plt.show()
```




```
plt.title("Male customer Mean Distribution")
sns.histplot(Male_sample_mean)
plt.show()
```



Are Confidence interval of average male and female spending overlapping?How can Walmart leverage this conclusion to make changes or improvements?

```
Normal_female_mean=df[df["Gender"]=="F"]["Purchase"].mean()
sample_female_mean=np.mean(Female_sample_mean)
print(Normal_female_mean)
print(sample_female_mean)
```

```
8734.565765155476
8728.742145
```

```
#Female purchases Mean and STD
F_mu=np.mean(Female_sample_mean)
F_std=np.std(Female_sample_mean)
```

We can see that normal Female mean and sample Means of purchase are approximatly equal to each other

```
Normal_male_mean=df[df["Gender"]=="M"]["Purchase"].mean()
sample_male_mean=np.mean(Male_sample_mean)
print(Normal_male_mean)
print(sample_male_mean)
```

```
9437.526040472265
9428.685660000001
```

```
#Male purchases Mean and STD
M_mu=np.mean(Male_sample_mean)
M_std=np.std(Male_sample_mean)
```

We can see that normal Male mean and sample Means of purchase are approximatly equal to each other

Checking for 97% CI for male and female

```
(M_mu-1.96*M_std , M_mu+1.96*M_std)
```

```
(9020.66947902075, 9836.701840979253)
```

```
(F_mu-1.96*F_std , F_mu+1.96*F_std)
```

```
(8338.506602966692, 9118.977687033308)
```

Here Males and Females Purchases intervals are overlapping at 97%

Checking for 95% CI for male and female

```
(norm.cdf(1.96) , norm.cdf(1.64))
```

```
(0.9750021048517795, 0.9494974165258963)
```

```
(M_mu-1.64*M_std , M_mu+1.64*M_std)
```

```
(9087.284365711239, 9770.086954288763)
```

```
(F_mu-1.64*F_std , F_mu+1.64*F_std)
```

```
(8402.21852819662, 9055.26576180338)
```

At 97% confidence,it's difficult but at 95% we may say that the purchase amount for males are different that of females in the population

Ques : 4D Result when the same activity is performed for married vs Unmarried?

```
df.groupby("Marital_Status")["Purchase"].sum()
```

```
Marital_Status
0    3008927447
1    2086885295
Name: Purchase, dtype: int64
```

Unmarried customers are purchasing more than Married customers.

```
#Retriving the Respective datasets for married and unmarried
Unmarried=df[df["Marital_Status"]==0]["Purchase"]
Married=df[df["Marital_Status"]==1]["Purchase"]
```

```
# sample data of married and unmarried
Unmarried_sample=[Unmarried.sample(600).mean() for i  in range(1000)]
Married_sample=[Married.sample(600).mean() for i  in range(1000)]
```

```
#Means of original data and sample data for UNMARRIED customers
Unmarried_mean=df[df["Marital_Status"]==0]["Purchase"].mean()
Unmarried_sample_mean=np.mean(Unmarried_sample)
print(Unmarried_mean)
print(Unmarried_sample_mean)
```

```
9265.907618921507
9266.824899999998
```

```
#Means of original data and sample data for MARRIED customers
Married_mean=df[df["Marital_Status"]==1]["Purchase"].mean()
Married_sample_mean=np.mean(Married_sample)
print(Married_mean)
print(Married_sample_mean)
```

```
9261.174574082374
9259.929423333333
```

```
#Unmarried purchases Mean and STD
UM_mu=np.mean(Unmarried_sample)
UM_std=np.std(Unmarried_sample)
print(UM_mu,UM_std)
```

```
9266.824899999998 201.43286207944763
```

```
#Married purchases Mean and STD
MM_mu=np.mean(Married_sample)
MM_std=np.std(Married_sample)
print(MM_mu,MM_std)
```

```
9259.929423333333 203.65053330730754
```

```
print((UM_mu-1.96*UM_std , UM_mu+1.96*UM_std))
print((MM_mu-1.96*MM_std , MM_mu+1.96*MM_std))
```

```
(8872.01649032428, 9661.633309675715)
(8860.77437805101, 9659.084468615656)
```

Maried and Unmarried Purchase amount are always overlapping

```
#H0--> mu1=mu2
#Ha--> mu1 !=mu2

t_stats,p_value=ttest_ind(Female_purchase,Male_purchase)
print(p_value)
if p_value < 0.05:
  print("Reject Null Hypothesis")
  print("Female purchases are not Equal to Male purchases")
else :
  print("Fail to reject Null Hypothesis")
  print("Female purchases are Equal to Male Purchases")
```

```
0.0
Reject Null Hypothesis
Female purchases are not Equal to Male purchases
```

```
couples=df[df["Marital_Status"]==1]["Purchase"].mean()
singles=df[df["Marital_Status"]==0]["Purchase"].mean()
```

```
print(couples,singles)
```

```
9261.174574082374 9265.907618921507
```

```
df.groupby("Marital_Status")["Purchase"].mean()
```

```
Marital_Status
0    9265.907619
1    9261.174574
Name: Purchase, dtype: float64
```

```
#H0--> mu1=mu2
#Ha--> mu1 !=mu2

t_stats,p_value=ttest_ind(couples,singles)#,alternative="less")
print(p_value)
if p_value < 0.05:
  print("Reject Null Hypothesis")
  print("couples purchases are not Equal to singles purchases")
else :
  print("Fail to reject Null Hypothesis")
  print("couples purchases mean are approximatly Equal to singles purchases")
```

```
nan
Fail to reject Null Hypothesis
couples purchases mean are approximatly Equal to singles purchases
/usr/local/lib/python3.10/dist-packages/scipy/stats/_stats_py.py:7030: RuntimeWarning: invalid value encountered in double_scalars
  svar = ((n1 - 1) * v1 + (n2 - 1) * v2) / df
```

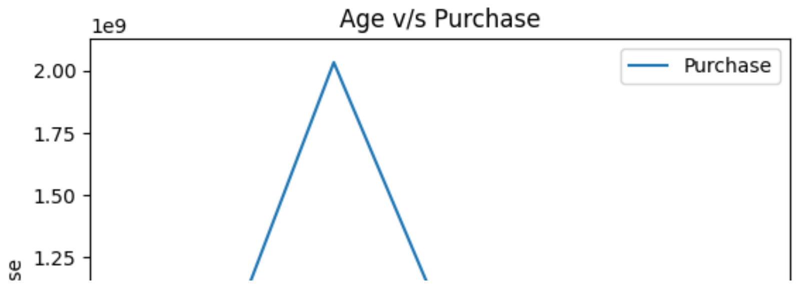
5 INSIGHTS on exploration

AGE Category Purchases

```
Age_Cat_purchase=df[["Age","Purchase"]].groupby("Age").sum()
Age_Cat_purchase.index
```

```
Index(['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+'], dtype='object', name='Age')
```

```
sns.lineplot(Age_Cat_purchase,color="r")
plt.title("Age v/s Purchase")
plt.ylabel("Purchase")
plt.show()
```



Gender category Purchases

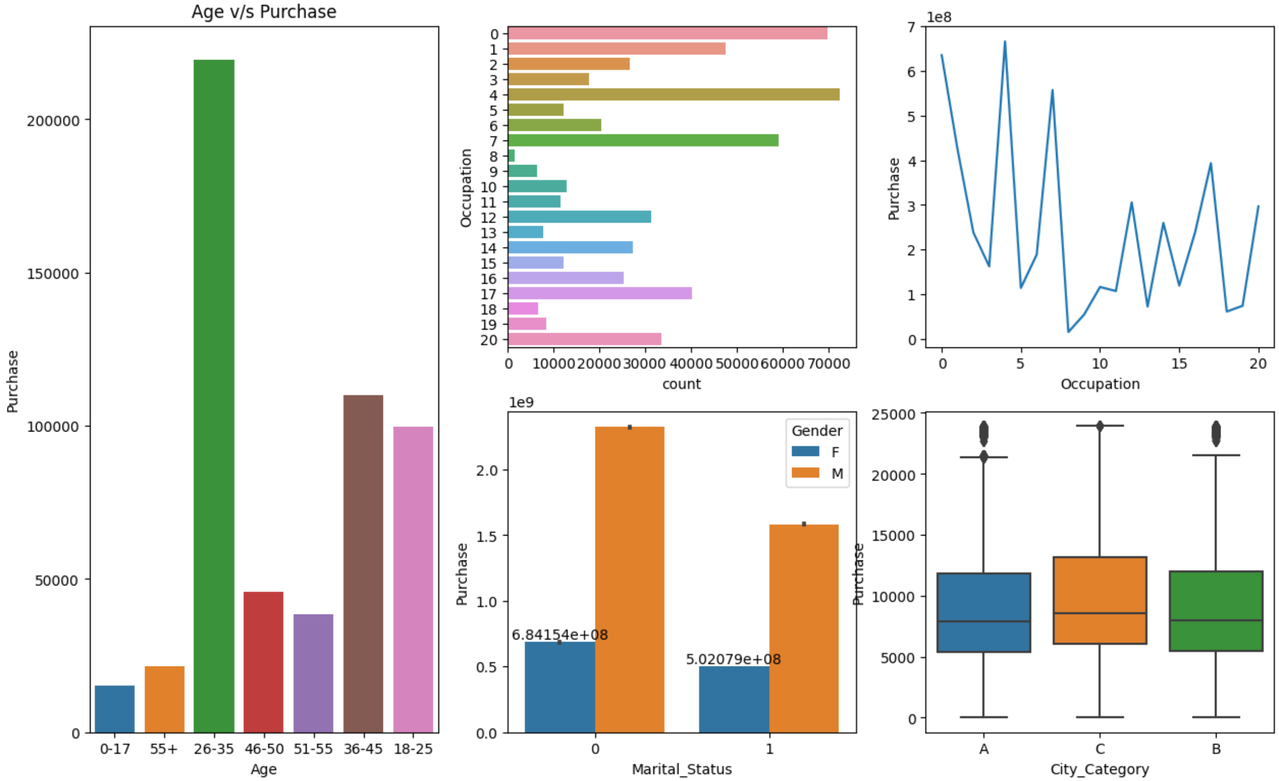
```
Genderpurchase=df[["Gender","Purchase"]].groupby("Gender").sum().reset_index().rename(columns={"Purchase":"Total_amount"})
```

```
Genderpurchase
```

	Gender	Total_amount
0	F	1186232642
1	M	3909580100

```
plt.figure(figsize=(15,9))
plt.subplot(2,3,2)
sns.countplot(data=df,y=df["Occupation"])

plt.subplot(2,3,5)
ax=sns.barplot(data=df,x="Marital_Status",y="Purchase",estimator="sum",hue="Gender")
ax.bar_label(ax.containers[0], fontsize=10);

plt.subplot(2,3,3)
sns.lineplot(df,x=df["Occupation"],y=df["Purchase"],estimator="sum")
plt.xlabel("Occupation")

plt.subplot(1,3,1)
sns.countplot(data=df,x=df["Age"])
plt.title("Age v/s Purchase")
plt.ylabel("Purchase")

plt.subplot(2,3,6)
sns.boxplot(data=df,x="City_Category",y="Purchase")
```

<Axes: xlabel='City_Category', ylabel='Purchase'>



From the above survey provided a picture of the average Walmart shopper are unmarried Males,late-middle aged[26-35] customers with occupation 4th level, from City_Category B.but the mean of males and females purchase are almost equal to each others.

```
df.groupby("City_Category")["Purchase"].aggregate(["sum","mean","count"]).sort_values(by="sum")
```

City_Category	sum	mean	count
A	1316471661	8911.939216	147720
C	1663807476	9719.920993	171175
B	2115533605	9151.300563	231173

most of the sales from City_category B,but C category customers purchase mean is more than the B Category.

```
df.groupby("Occupation")["Purchase"].aggregate(["sum","count","mean"]).sort_values(by="sum")
```

Occupation	sum	count	mean
8	14737388	1546	9532.592497
9	54340046	6291	8637.743761
18	60721461	6622	9169.655844
13	71919481	7728	9306.351061
19	73700617	8461	8710.627231
11	106751618	11586	9213.845848
5	113649759	12177	9333.149298
10	115844465	12930	8959.355375
15	118960211	12165	9778.891163
3	162002168	17650	9178.593088
6	188416784	20355	9256.535691
2	238028583	26588	8952.481683
16	238346955	25371	9394.464349
14	259454692	27309	9500.702772
20	296570442	33562	8836.494905
12	305449446	31179	9796.640239
17	393281453	40043	9821.478236
1	424614144	47426	8953.193270
7	557371587	59133	9425.728223
0	635406958	69638	9124.428588
4	666244484	72308	9213.980251

Occupation level 4 customers are involing more in the Walmart sales on the Friday followed by level 3 customers. but Average wise occupation 17 th customers having more sales..

RECOMMENDATIONS :

Walmart has to do survey on products before the Black Friday sales arrives.So, that they can alert with products and based on that they can plan to give discounts over the products. they need to check the champions,loyal and promising coustomers as they have to provide more discounts/coupons/free shippings,so that they can make more sales. parllel they need to check the city wise category also.