```
import pandas as pd
```

```
df=pd.read_csv('C:/Users/e112783/Desktop/walmart.csv')
```

```
df.head(15)
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Currer
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	
5	1000003	P00193542	M	26-35	15	A	
6	1000004	P00184942	M	46-50	7	B	
7	1000004	P00346142	M	46-50	7	B	
8	1000004	P0097242	M	46-50	7	B	
9	1000005	P00274942	M	26-35	20	A	
10	1000005	P00251242	M	26-35	20	A	
11	1000005	P00014542	M	26-35	20	A	
12	1000005	P00031342	M	26-35	20	A	
13	1000005	P00145042	M	26-35	20	A	
14	1000006	P00231342	F	51-55	9	A	

```
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
```

```
#No Missing Value
```

```
df.shape[0]
```

```
550068
```

```
sample_size= df.shape[0]
```

```
#unique users in different genders
df.groupby('Gender')['User_ID'].nunique()
```

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

```
#distribution of gender
df['Gender'].value_counts()
```

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

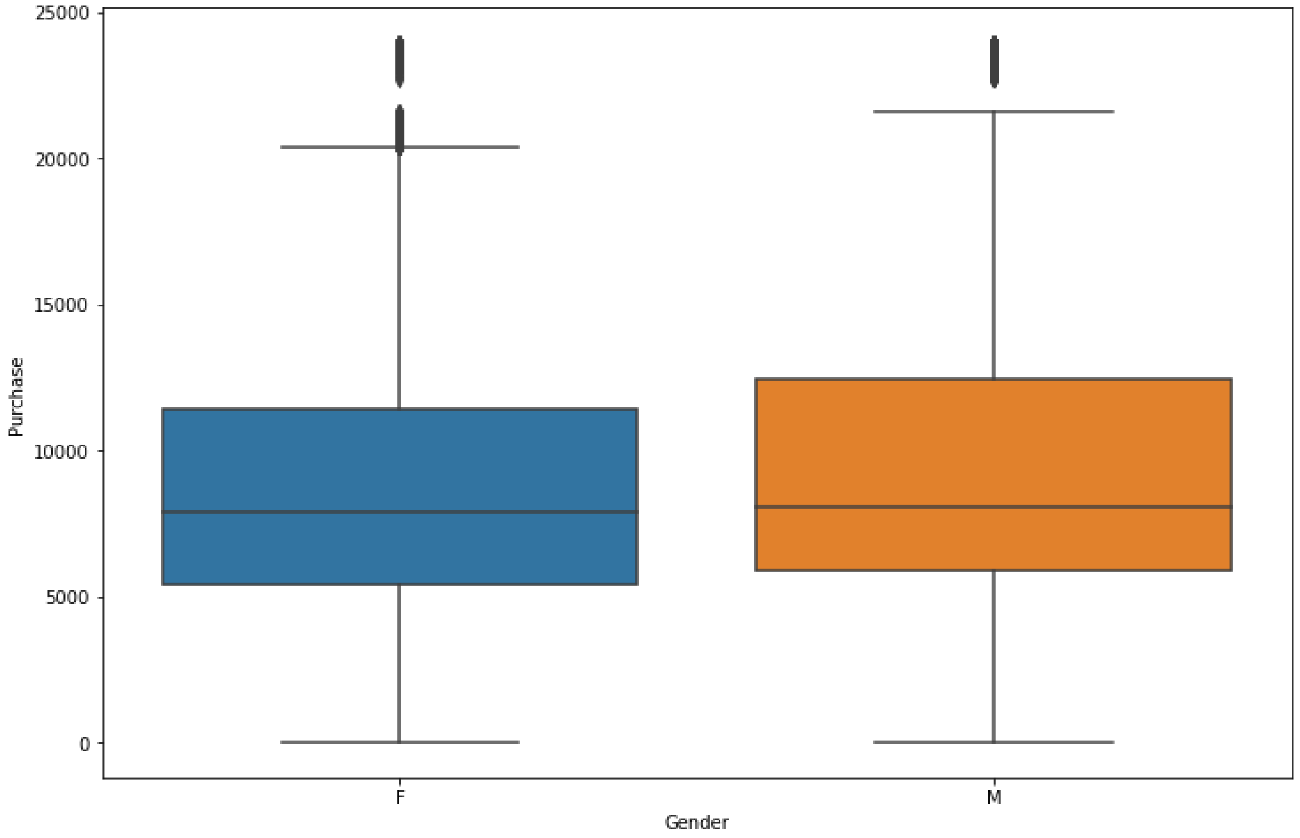
```
#Check different metrics on purchase by different genders
df.groupby('Gender')['Purchase'].describe()
```

	count	mean	std	min	25%	50%	75%	max
Gender								
F	135809.0	8734.565765	4767.233289	12.0	5433.0	7914.0	11400.0	23959.0
M	414259.0	9437.526040	5092.186210	12.0	5863.0	8098.0	12454.0	23961.0

```
import seaborn as sbn
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(12,8))
sbn.boxplot(x='Gender', y='Purchase', data=df)
```

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



In one month, 1666 Females are spending mean -8734.565765 & median - 7914.0

4225 Males are spending mean- 9437.526040 Median - 8098.0

But the results are not conclusive.

```
#Check ratio of males to female
```

```
4225/1666
```

```
2.536014405762305
```

7.59:3

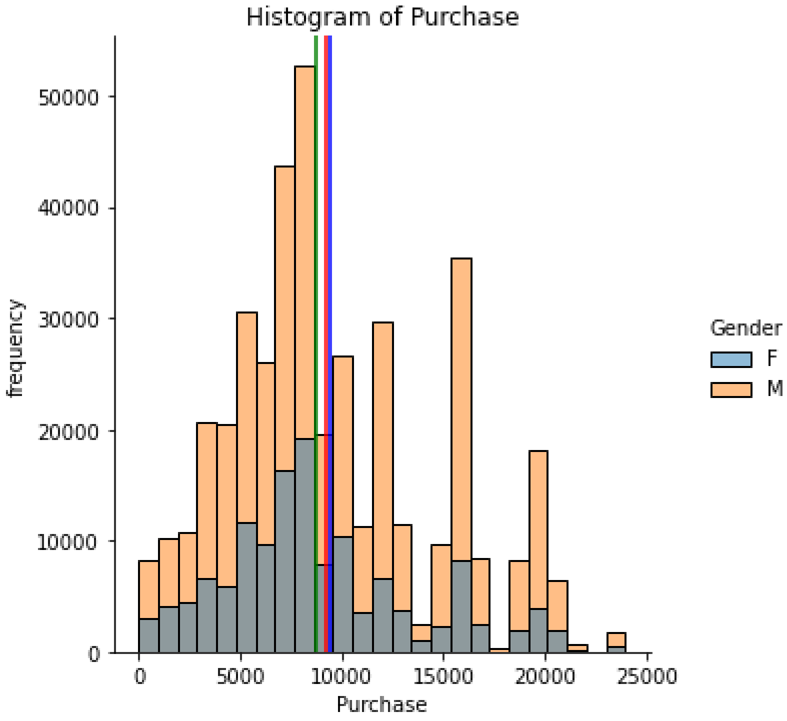
Checking through visualization:

```
# plot all the observation
```

```
plt.figure(figsize=(20,8))
sbn.displot(x='Purchase', data=df, bins=25,hue='Gender')
plt.xlabel('Purchase')
plt.ylabel('frequency')
plt.title('Histogram of Purchase')

plt.axvline(x=df['Purchase'].mean(),color='r')
plt.axvline(x=df[df['Gender']=='M']['Purchase'].mean(),color='b')
plt.axvline(x=df[df['Gender']=='F']['Purchase'].mean(),color='g')
```

```
<matplotlib.lines.Line2D at 0x20f3245ba00>
<Figure size 1440x576 with 0 Axes>
```



We can see that the distribution is close to normal.

```
#checking mean and standard deviation

df.groupby('Gender')['Purchase'].describe()
```

Gender	count	mean	std	min	25%	50%	75%	max
F	135809.0	8734.565765	4767.233289	12.0	5433.0	7914.0	11400.0	23959.0
M	414259.0	9437.526040	5092.186210	12.0	5863.0	8098.0	12454.0	23961.0

```
df[df['Gender']=='F']['Purchase'].mean()
```

```
8734.565765155476
```

```
df[df['Gender']=='F']['Purchase'].std()
```

```
4767.233289291444
```

Thinking of this dataset as population. Let's first check if applying CLT on it's sample gives us actual characteris of the population or not.

Let us take a random sample (size = 300) from this data to analyse the sample mean.

```
df.sample(300).groupby('Gender')['Purchase'].describe()
```

Gender	count	mean	std	min	25%	50%	75%	max
F	69.0	8073.463768	4427.936379	128.0	5410.0	7378.0	9766.0	21451.0
M	231.0	8973.320346	5068.525379	49.0	5563.0	7930.0	11741.0	20836.0

Every time we take a sample, o ur mean value is different. There is variability in the sample mean itself. Does the sample mean itself follow a distribution? Let's assess this. Let us pick around 1,000 random samples of size 300 from the entire data set and calculated the mean of each sample.

```
male_sample_means=[df[df['Gender']=='M']['Purchase'].sample(300).mean() for i in ra
```

```
female_sample_means=[df[df['Gender']=='F']['Purchase'].sample(300).mean() for i in
```

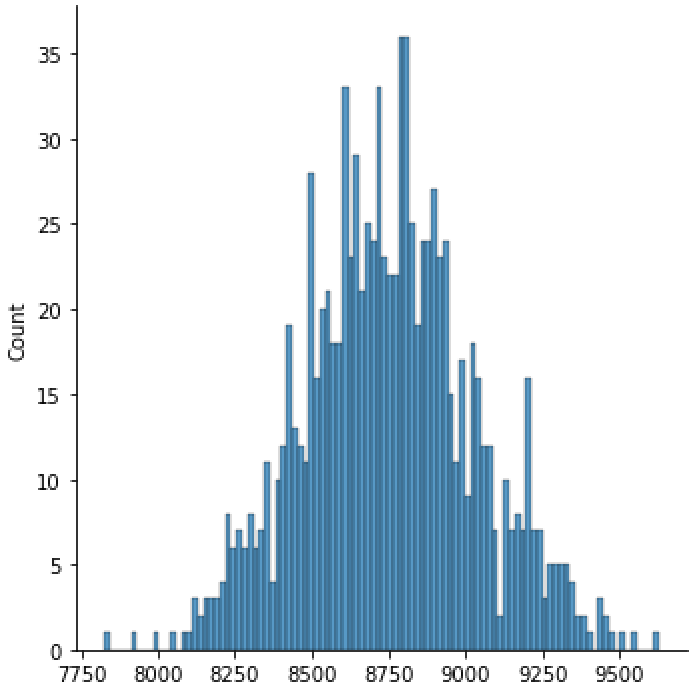
Plot the distribution of all these sample means (This is our sampling distribution).

```
sbn.displot(male_sample_means,bins=100)
```

```
<seaborn.axisgrid.FacetGrid at 0x20f51f4dd00>
```

```
sbn.displot(female_sample_means, bins=100)
```

```
<seaborn.axisgrid.FacetGrid at 0x20f560e09d0>
```



We can observe that the sampling distribution is nearly normal. Now we will compute the mean and standard deviation of this sampling distribution.

```
pd.Series(male_sample_means).mean()
```

```
9431.564926666659
```

```
pd.Series(female_sample_means).mean()
```

```
8750.188953333327
```

The mean of this sampling distribution (or in other words, the mean of all the sample means that we had taken), came out pretty close to the original population mean. This demonstrates the first property of the Central Limit theorem
Sampling Distribution mean= Population Mean

image.png

However, it would not be fair to infer that the population mean is exactly equal to the sample mean. It is because the defects in the sampling process always tend to cause some errors. Therefore, the sample mean’s value must be reported with some margin of error.

```
pd.Series(male_sample_means).std()
```

```
298.6372890743671
```

```
import numpy as np
pd.Series(male_sample_means).std()/np.sqrt(1000)
```

```
9441.00866694399
```

![image.png]

Now that we have verified these two properties, let us observe the effect of sample size on the resulting sampling distribution. In this demonstration, we will observe that as the sample size increases, the underlying sampling distribution will approximate a normal distribution.

![image.png]

95% Confidence Interval

```
lower_limit= pd.Series(female_sample_means).mean() - (pd.Series(female_sample_means
upper_limit= pd.Series(female_sample_means).mean() + (pd.Series(female_sample_means
```

```
lower_limit
```

```
8733.046622396687
```

```
upper_limit
```

```
8767.331284269967
```

```
lower_limit= pd.Series(male_sample_means).mean() - (pd.Series(male_sample_means).st
upper_limit= pd.Series(male_sample_means).mean() + (pd.Series(male_sample_means).st
```

```
lower_limit
```

```
9413.05519572309
```

```
upper_limit
```

```
9450.074657610228
```