In [1]:

```
import pandas as pd
df = pd.read_csv('C:/Users/maheshmangaonkar/Desktop/Walmart.csv')
```

In [2]:

```
df
```

Out[2]:

	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
...	...	...	...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1	1	20	368
550064	1006035	P00375436	F	26-35	1	C	3	0	20	371
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	137
550066	1006038	P00375436	F	55+	1	C	2	0	20	365
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	490

550068 rows × 10 columns

In [3]:

```
df['User_ID'].unique
```

Out[3]:

```
<bound method Series.unique of 0         1000001
1         1000001
2         1000001
3         1000001
4         1000002
           ...
550063    1006033
550064    1006035
550065    1006036
550066    1006038
550067    1006039
Name: User_ID, Length: 550068, dtype: int64>
```

In [4]:

```
# Checking details
df.describe(include='all')
```

Out[4]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purc
count	5.500680e+05	550068	550068	550068	550068.000000	550068	550068	550068.000000	550068.000000	550068.0
unique	NaN	3631	2	7	NaN	3	5	NaN	NaN	
top	NaN	P00265242	M	26-35	NaN	B	1	NaN	NaN	
freq	NaN	1880	414259	219587	NaN	231173	193821	NaN	NaN	
mean	1.003029e+06	NaN	NaN	NaN	8.076707	NaN	NaN	0.409653	5.404270	9263.9
std	1.727592e+03	NaN	NaN	NaN	6.522660	NaN	NaN	0.491770	3.936211	5023.0
min	1.000001e+06	NaN	NaN	NaN	0.000000	NaN	NaN	0.000000	1.000000	12.0
25%	1.001516e+06	NaN	NaN	NaN	2.000000	NaN	NaN	0.000000	1.000000	5823.0
50%	1.003077e+06	NaN	NaN	NaN	7.000000	NaN	NaN	0.000000	5.000000	8047.0
75%	1.004478e+06	NaN	NaN	NaN	14.000000	NaN	NaN	1.000000	8.000000	12054.0
max	1.006040e+06	NaN	NaN	NaN	20.000000	NaN	NaN	1.000000	20.000000	23961.0

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64 
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

In [6]:

```
columns=['User_ID','Occupation', 'Marital_Status', 'Product_Category']
df[columns]=df[columns].astype('object')
```

In [7]:

```
df.describe()
```

Out[7]:

	Purchase
count	550068.000000
mean	9263.968713
std	5023.065394
min	12.000000
25%	5823.000000
50%	8047.000000
75%	12054.000000
max	23961.000000

In [8]:

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

Out[8]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
count	550068.0	550068	550068	550068	550068.0	550068	550068	550068.0	550068.0	550068.000000
unique	5891.0	3631	2	7	21.0	3	5	2.0	20.0	NaN
top	1001680.0	P00265242	M	26-35	4.0	B	1	0.0	5.0	NaN
freq	1026.0	1880	414259	219587	72308.0	231173	193821	324731.0	150933.0	NaN
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	9263.968713
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5023.065394
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	12.000000
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5823.000000
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	8047.000000
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	12054.000000
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	23961.000000

In [ ]:

Observations:

1. There are 5891 unique user_id in the data. The top user_id is 1001680.
2. 3631 unique products are there with P00265242 being the top product.
3. males are purchasing more than females.
4. 26-35 age groups people are more into purchasing of products.

In [9]:

```
df
```

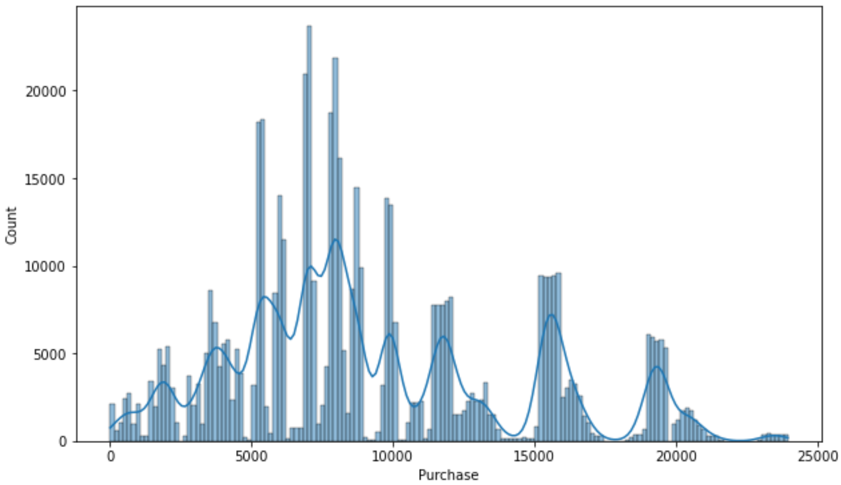
Out[9]:

	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
...	...	...	...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1	1	20	368
550064	1006035	P00375436	F	26-35	1	C	3	0	20	371
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	137
550066	1006038	P00375436	F	55+	1	C	2	0	20	365
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	490

550068 rows × 10 columns

In [10]:

```
# Univariate analysis
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x="Purchase", kde=True)
plt.show()
```



In [11]:

```
#From histplot we can say that the highest purchase is 10000.
```

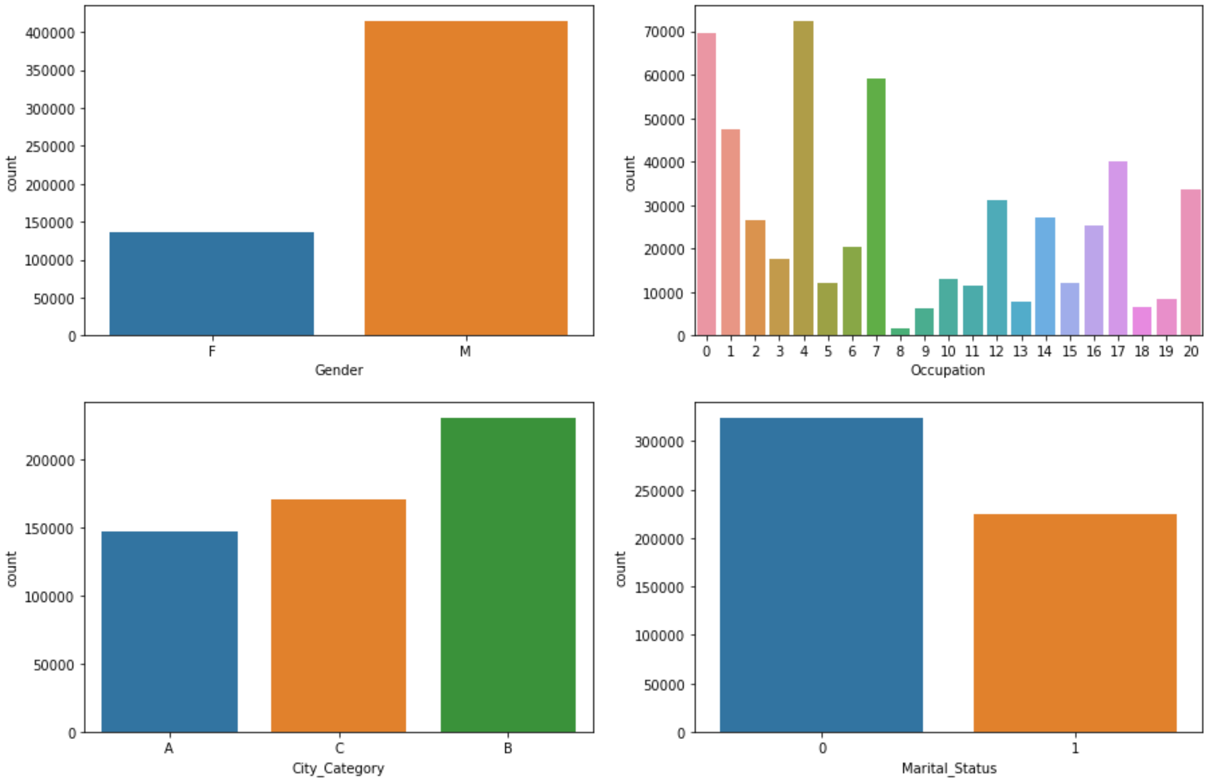
plt.figure(figsize=(5, 4)) sns.boxplot(data=df, y='Purchase') plt.show()

In [12]:

```
# There are outliers in purchase
```

In [13]:

```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))
sns.countplot(data=df, x='Gender', ax=axs[0,0])
sns.countplot(data=df, x='Occupation', ax=axs[0,1])
sns.countplot(data=df, x='City_Category', ax=axs[1,0])
sns.countplot(data=df, x='Marital_Status', ax=axs[1,1])
plt.show()
```



1. The count of males is more than females
2. Occupation category 4, 0, and 7 are with higher number of purchases and category 8 with the lowest number of purchases.
3. B city_category are highest.
4. Unmarried people are more than married.

Type *Markdown* and LaTeX: α^2

sns.countplot(data=df, x='Product_Category')

In [14]:

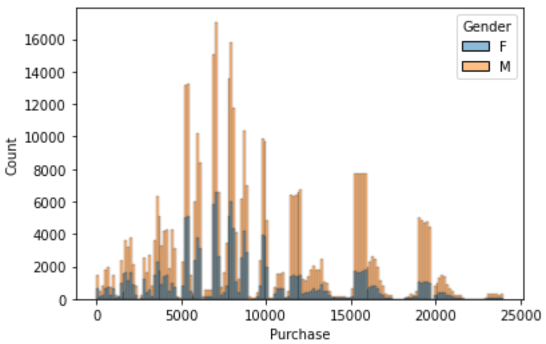
```
#Product_category no 5 is highest in number.
```

In [15]:

```
#Bivariate Analysis
sns.histplot(data=df, x="Purchase", hue = 'Gender')
```

Out[15]:

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



In [16]:

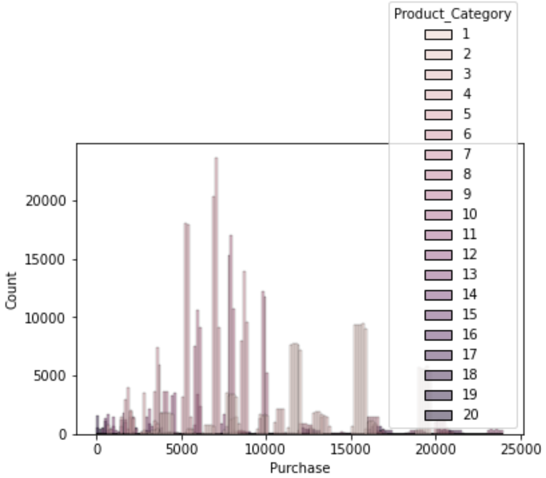
```
#Male purchases are higher than females
```

In [17]:

```
sns.histplot(data=df, x="Purchase", hue = 'Product_Category')
```

Out[17]:

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

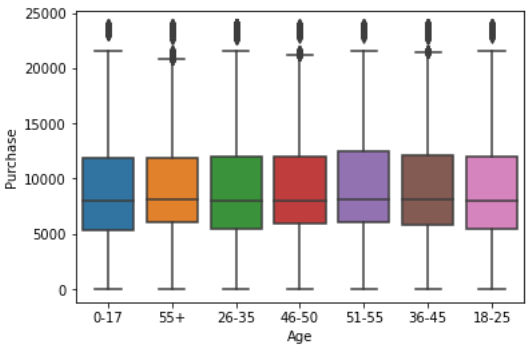


In [18]:

```
sns.boxplot(data=df, y="Purchase", x= 'Age')
```

Out[18]:

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



In [19]:

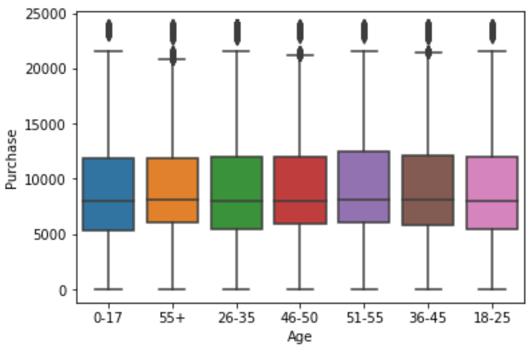
```
# For all age groups the count lies between 5000 to 10000 with some outliers
```

In [20]:

```
sns.boxplot(data=df, x="Age", y= 'Purchase')
```

Out[20]:

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



In [21]:

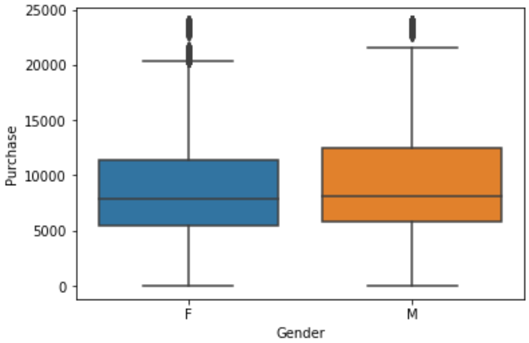
```
#Most of purchases are between 5000 to 10000
```

In [22]:

```
sns.boxplot(data=df, x="Gender", y= 'Purchase')
```

Out[22]:

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



In [23]:

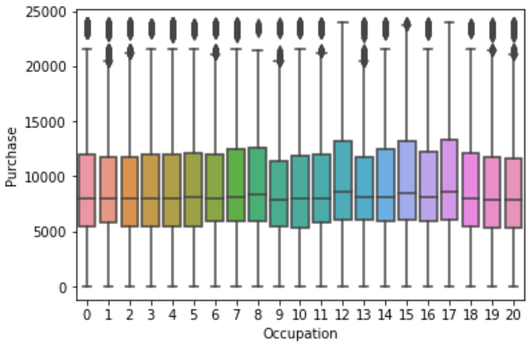
```
#Purchase for male are more than females
```

In [24]:

```
sns.boxplot(data=df, x="Occupation", y= 'Purchase')
```

Out[24]:

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



In [25]:

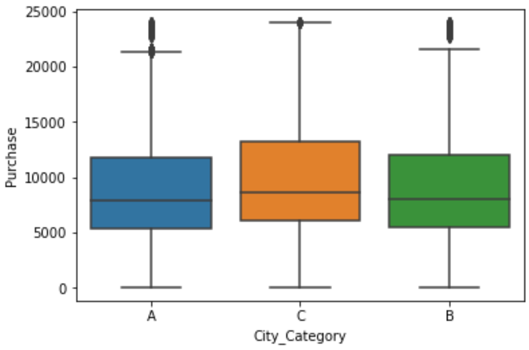
```
#The purchase count is almost same for all occupation categories
```

In [26]:

```
sns.boxplot(data=df, x="City_Category", y= 'Purchase')
```

Out[26]:

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



In [27]:

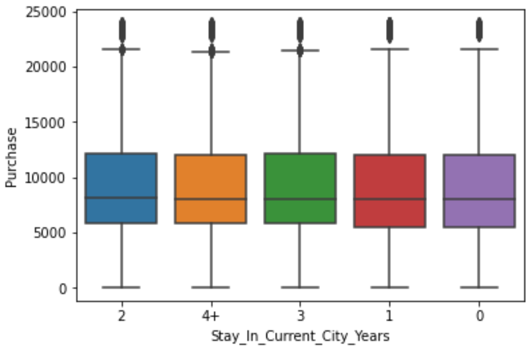
```
# C city category has highest count of purchases
```

In [28]:

```
sns.boxplot(data=df, x="Stay_In_Current_City_Years", y= 'Purchase')
```

Out[28]:

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



In [29]:

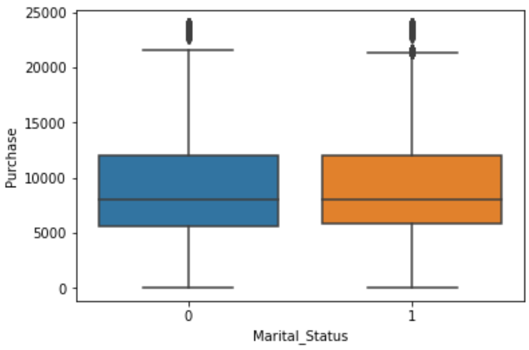
```
# same for all the people
```

In [30]:

```
sns.boxplot(data=df, x="Marital_Status", y= 'Purchase')
```

Out[30]:

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



In [31]:

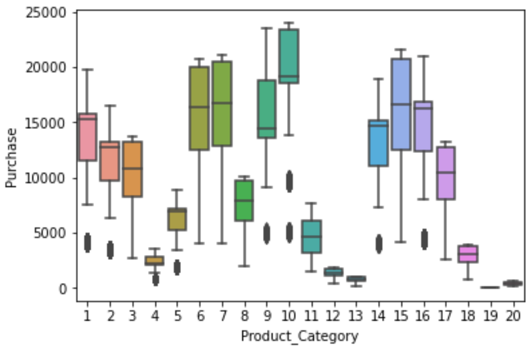
```
#
```

In [32]:

```
sns.boxplot(data=df, x="Product_Category", y= 'Purchase')
```

Out[32]:

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



In [33]:

```
# Product n0 10 is the costliest
```

In [34]:

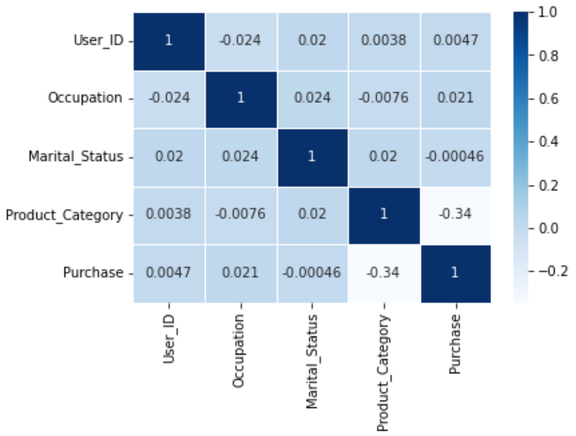
```
df1 = pd.read_csv('C:/Users/maheshmangaonkar/Desktop/Walmart.csv')
```

In [35]:

```
# Corelation
sns.heatmap(df1.corr(), annot=True, cmap="Blues", linewidth=.5)
```

Out[35]:

```
<AxesSubplot:>
```



In [36]:

```
#There is no rative signifance between the values
```

In [37]:

```
#Average money spend
avgamt = df.groupby(['User_ID', 'Gender'])[['Purchase']].sum()
avgamt = avgamt.reset_index()
avgamt
```

Out[37]:

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

5891 rows × 3 columns

In [38]:

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

Out[38]:

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

In [39]:

```
avgamt.groupby(['Gender'])['Purchase'].mean()
```

Out[39]:

```
Gender
F    712024.394958
M    925344.402367
Name: Purchase, dtype: float64
```

In [40]:

```
avgamt_male = avgamt[avgamt['Gender']=='M']
avgamt_female = avgamt[avgamt['Gender']=='F']
avgamt_male
```

Out[40]:

	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 × 3 columns

In [41]:

```
#Mean of males is higher than females. Average spend is more in males
```

In [42]:

```
# Population means for male and female
```

In [43]:

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

from scipy.stats import norm
avgamt_male_mean = np.mean(avgamt_male['Purchase'])
avgamt_female_mean = np.mean(avgamt_female['Purchase'])
```

In [44]:

```
avgamt_male_mean
```

Out[44]:

925344.4023668639

In [45]:

```
avgamt_female_mean
```

Out[45]:

712024.3949579832

In [46]:

```
#Population std deviation for male and female
```

In [47]:

```
avgamt_male_std = np.std(avgamt_male['Purchase'])
avgamt_female_std = np.std(avgamt_female['Purchase'])
```

In [48]:

```
avgamt_male_std
```

Out[48]:

985713.4276071227

In [49]:

```
avgamt_female_std
```

Out[49]:

807128.3816336752

In [50]:

```
#Finding samples with n= 1000
sample_male= avgamt_male.sample(1000)
sample_male
```

Out[50]:

	User_ID	Gender	Purchase
2319	1002388	M	231198
550	1000566	M	2114831
317	1000323	M	556238
2385	1002457	M	2480975
2064	1002124	M	3543720
...	...	...	...
2182	1002242	M	1668515
4264	1004377	M	1462363
1158	1001197	M	167477
2109	1002169	M	130747
3222	1003314	M	1981086

1000 rows × 3 columns

In [51]:

```
sample_female = avgamt_female.sample(1000)
sample_female
```

Out[51]:

	User_ID	Gender	Purchase
263	1000268	F	1248334
354	1000361	F	77684
2572	1002647	F	160686
2557	1002631	F	388449
3126	1003215	F	149101
...	...	...	...
1195	1001235	F	256097
2593	1002668	F	825610
1051	1001088	F	5628655
3666	1003766	F	429008
964	1000995	F	237680

1000 rows × 3 columns

In [52]:

```
#Sample mean
sample_male_mean = np.mean(sample_male)
sample_male_mean
```

```
C:\Users\maheshmangaonkar\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3438: FutureWarning: In a future version, D
ataFrame.mean(axis=None) will return a scalar mean over the entire DataFrame. To retain the old behavior, use 'frame.mean(a
xis=0)' or just 'frame.mean()'
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
C:\Users\maheshmangaonkar\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3438: FutureWarning: Dropping of nuisance c
olumns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Se
lect only valid columns before calling the reduction.
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
```

Out[52]:

```
User_ID     1002977.886
Purchase     896774.389
dtype: float64
```

In [53]:

```
sample_female_mean = np.mean(sample_female)
sample_female_mean
```

Out[53]:

```
User_ID     1003047.244
Purchase     715957.438
dtype: float64
```

In [54]:

```
#Sample std deviation
```

In [55]:

```
sample_male_std = np.std(sample_male)
sample_male_std
```

```
C:\Users\maheshmangaonkar\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3579: FutureWarning: Dropping of nuisance c
olumns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Se
lect only valid columns before calling the reduction.
  return std(axis=axis, dtype=dtype, out=out, ddof=ddof, **kwargs)
```

Out[55]:

```
User_ID       1710.298685
Purchase    924593.977986
dtype: float64
```

In [56]:

```
sample_female_std = np.std(sample_female)
sample_male_std
```

Out[56]:

```
User_ID       1710.298685
Purchase    924593.977986
dtype: float64
```

In [57]:

```
#Standard error for sample
```

In [58]:

```
import math
sample_male_error = sample_male_std/np.sqrt(1000)
sample_male_error
```

Out[58]:

```
User_ID        54.084393
Purchase    29238.228813
dtype: float64
```

In [59]:

```
sample_female_error = sample_female_std/np.sqrt(1000)
sample_female_error
```

Out[59]:

```
User_ID        56.723394
Purchase    25642.468076
dtype: float64
```

In [60]:

```
#CI for 90% CI
```

In [61]:

```
norm.ppf(0.05)
```

Out[61]:

```
-1.6448536269514729
```

In [62]:

```
z90 = norm.ppf(0.95)
```

In [63]:

```
Upper_Limit_male=z90*sample_male_error + sample_male_mean
Lower_Limit_male=sample_male_mean - z90*sample_male_error
```

In [64]:

```
Upper_Limit_male
Lower_Limit_male
```

Out[64]:

```
User_ID     1.002889e+06
Purchase    8.486818e+05
dtype: float64
```

In [65]:

```
Upper_Limit_male
```

Out[65]:

```
User_ID     1.003067e+06
Purchase    9.448670e+05
dtype: float64
```

In [66]:

```
Male_CI = [Upper_Limit_male, Lower_Limit_male]
Male_CI
```

Out[66]:

```
[User_ID     1.003067e+06
 Purchase    9.448670e+05
 dtype: float64,
 User_ID     1.002889e+06
 Purchase    8.486818e+05
 dtype: float64]
```

In [67]:

```
Upper_Limit_female=z90*sample_female_error + sample_female_mean
Lower_Limit_female=sample_female_mean - z90*sample_female_error
```

In [68]:

```
FeMale_CI = [Upper_Limit_female, Lower_Limit_female]
FeMale_CI
```

Out[68]:

```
[User_ID     1.003141e+06
 Purchase    7.581355e+05
 dtype: float64,
 User_ID     1.002954e+06
 Purchase    6.737793e+05
 dtype: float64]
```

In [ ]:

```
#Average amount spend by male customers lie in the range 9.448670e+05 - 8.486818e+05

#Average amount spend by female customers lie in range 7.581355e+05 - 6.737793e+05
```

In [ ]:

```
#Calculating 95% confidence interval for sample size 1000:
```

In [74]:

```
z95 = norm.ppf(0.975)
```

In [75]:

```
z95
```

Out[75]:

```
1.959963984540054
```

In [76]:

```
Upper_Limit_male=z95*sample_male_error + sample_male_mean
Lower_Limit_male=sample_male_mean - z95*sample_male_error
```

In [77]:

```
Upper_Limit_male
```

Out[77]:

```
User_ID     1.003084e+06
Purchase    9.540803e+05
dtype: float64
```

In [78]:

```
Lower_Limit_male
```

Out[78]:

```
User_ID     1.002872e+06
Purchase    8.394685e+05
dtype: float64
```

In [79]:

```
Upper_Limit_female=z95*sample_female_error + sample_female_mean
Lower_Limit_female=sample_female_mean - z95*sample_female_error
```

In [80]:

```
Upper_Limit_female
```

Out[80]:

```
User_ID     1.003158e+06
Purchase    7.662158e+05
dtype: float64
```

In [81]:

```
Lower_Limit_female
```

Out[81]:

```
User_ID     1.002936e+06
Purchase    6.656991e+05
dtype: float64
```

In [82]:

```
Male_CI95 = [Upper_Limit_male, Lower_Limit_male]
Male_CI95
```

Out[82]:

```
[User_ID     1.003084e+06
 Purchase    9.540803e+05
 dtype: float64,
 User_ID     1.002872e+06
 Purchase    8.394685e+05
 dtype: float64]
```

In [83]:

```
FeMale_CI95 = [Upper_Limit_female, Lower_Limit_female]
FeMale_CI95
```

Out[83]:

```
[User_ID     1.003158e+06
 Purchase    7.662158e+05
 dtype: float64,
 User_ID     1.002936e+06
 Purchase    6.656991e+05
 dtype: float64]
```

In [ ]:

```
#Using 95% confidence
#Average amount spend by male customers lie in the range 9.540803e+05- 8.394685e+05

#Average amount spend by female customers lie in range 7.662158e+05 - 6.656991e+05
```

In [ ]:

```
#Calculating 99% confidence interval for sample size 1000:
```

In [94]:

```
z99 = norm.ppf(0.995)
```

In [95]:

```
z99
```

Out[95]:

```
2.5758293035489004
```

In [96]:

```
Upper_Limit_male=z99*sample_male_error + sample_male_mean
Lower_Limit_male=sample_male_mean - z99*sample_male_error
```

In [97]:

```
Male_CI99 = [Upper_Limit_male, Lower_Limit_male]
Male_CI99
```

Out[97]:

```
[User_ID     1.003117e+06
 Purchase    9.720871e+05
 dtype: float64,
 User_ID     1.002839e+06
 Purchase    8.214617e+05
 dtype: float64]
```

In [98]:

```
Upper_Limit_female=z99*sample_female_error + sample_female_mean
Lower_Limit_female=sample_female_mean - z99*sample_female_error
```

In [99]:

```
FeMale_CI99 = [Upper_Limit_male, Lower_Limit_male]
FeMale_CI99
```

Out[99]:

```
[User_ID     1.003117e+06
 Purchase    9.720871e+05
 dtype: float64,
 User_ID     1.002839e+06
 Purchase    8.214617e+05
 dtype: float64]
```

In [ ]:

```
#Using 99% confidence
#Average amount spend by male customers lie in the range 9.720871e+05- 8.214617e+05

#Average amount spend by female customers lie in range 9.720871e+05- 8.214617e+05
```

In [ ]:

```
lence level increases we get a closer insight of actual purchases in the CI. This will happen even if the number of samples are increased.
```