```
!pip install wget
import wget
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import binom,geom
from scipy.stats import f_oneway
from scipy.stats import t,ttest_ind,ttest_rel,ttest_1samp,kstest,chi2,chi2_contingency
url="https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094"
filename="walmart_data.csv"
wget.download(url,filename)
```

```
Collecting wget
  Downloading wget-3.2.zip (10 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: wget
  Building wheel for wget (setup.py) ... done
  Created wheel for wget: filename=wget-3.2-py3-none-any.whl size=9655 sha256=8aeecff5cda6605017f88cc0
  Stored in directory: /root/.cache/pip/wheels/8b/f1/7f/5c94f0a7a505ca1c81cd1d9208ae2064675d97582078e6
Successfully built wget
Installing collected packages: wget
Successfully installed wget-3.2
'walmart_data.csv'
```

```
df=pd.read_csv("walmart_data.csv")
df.head(10)
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Sta
0	1000001	P00069042	F	0-17	10	A	2	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	
5	1000003	P00193542	M	26-35	15	A	3	
6	1000004	P00184942	M	46-50	7	B	2	

```
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
```

```
df.shape
```

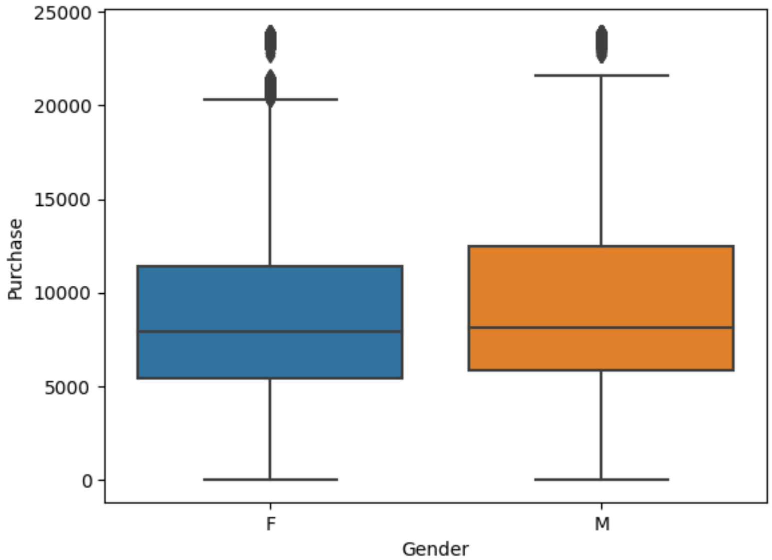
```
(550068, 10)
```

```
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
```

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

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




```
#calculating mean purchases of the three cities
#The mean purchase rate of men is slightly greater than of female
df1=df.groupby('Gender')['Purchase'].mean()
df1
```

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

```
 #Detecting Outliers
df['Purchase'].describe()
```

```
count    550068.000000
mean       9263.968713
std        5023.065394
min          12.000000
25%        5823.000000
50%        8047.000000
75%       12054.000000
max       23961.000000
Name: Purchase, dtype: float64
```

```
Q3=12054
Q1=5823
IQR=12054-5823
upper=Q3+1.5*IQR
lower=Q1-1.5*IQR
print(IQR)
print(upper)
print(lower)
```

```
6231
21400.5
-3523.5
```

```
#The values falling above 21400.5 and below -3523.5 are outliers for purchase rate
```

```
df.describe()
```

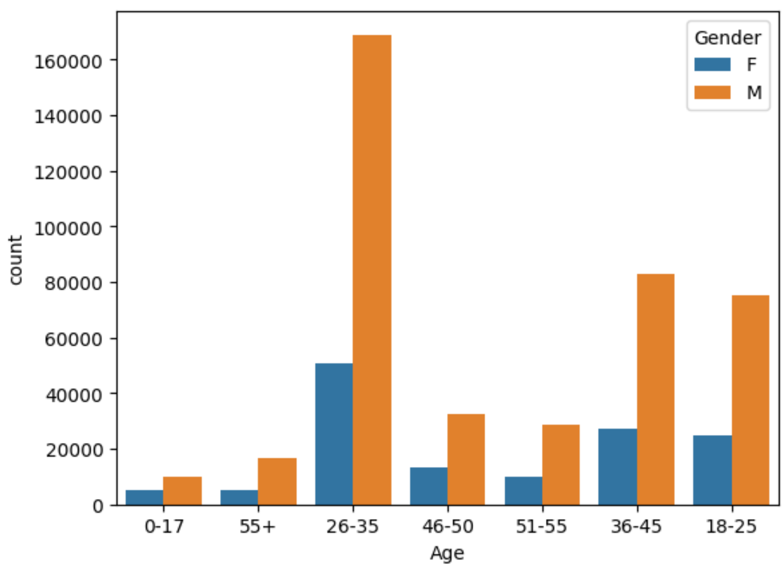
	User_ID	Occupation	Marital_Status	Product_Category	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9263.968713
std	1.727592e+03	6.522660	0.491770	3.936211	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	12.000000
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

```
pd.crosstab(index=df["Gender"],columns=df["Age"],margins=True)
```

Age	0-17	18-25	26-35	36-45	46-50	51-55	55+	All
Gender								
F	5083	24628	50752	27170	13199	9894	5083	135809
M	10019	75032	168835	82843	32502	28607	16421	414259
All	15102	99660	219587	110013	45701	38501	21504	550068

```
sns.countplot(data=df,x="Age",hue="Gender")  #The age group 26-35 are most occuring in the given dataset
```

```
<Axes: xlabel='Age', ylabel='count'>
```




```
df2=df[df["Gender"]=="M"]["Purchase"]
df2
```

```
4          7969
5         15227
6         19215
7         15854
8         15686
          ...
550057       61
550058      121
550060      494
550062      473
550063      368
Name: Purchase, Length: 414259, dtype: int64
```

```
df3=df[df["Gender"]=="F"]["Purchase"]
df3
```

```
0          8370
1         15200
2          1422
3          1057
14         5378
          ...
550061      599
550064      371
550065      137
550066      365
550067      490
Name: Purchase, Length: 135809, dtype: int64
```

```
f_oneway(df2,df3)
```

```
F_onewayResult(statistic=2010.4424717228953, pvalue=0.0)
```

```
 #Here the p-value is almost 0 so we can conclude that Gender affects the Purchase (statistically Significant)
```

```
#Marital Status Comparison
df.groupby('Marital_Status')["Purchase"].mean()
```

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

```
#Analysis about the population mean
df2.describe()
```

```
count    414259.00000
mean       9437.52604
std        5092.18621
min          12.00000
25%        5863.00000
50%        8098.00000
75%       12454.00000
max       23961.00000
Name: Purchase, dtype: float64
```

```
a=df2.mean()
b=df2.std()
n=len(df2)
n
```

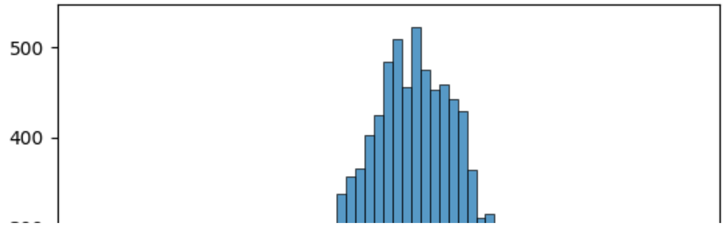
```
414259
```

```
boostrapped_mean_arr=[]
for i in range(10000):
  boostrapped_mean=np.random.choice(df2,size=n).mean()
  boostrapped_mean_arr.append(boostrapped_mean)
print(np.mean(boostrapped_mean_arr))
```

```
9437.37892818792
```

```
sns.histplot(boostrapped_mean_arr)
```

```
<Axes: ylabel='Count'>
```




```
#Calculating for 95 % CI
x1=np.percentile(boostrapped_mean_arr,2.5)
x2=np.percentile(boostrapped_mean_arr,97.5)
(x1,x2) #Lower and Upper limit
#So the average spending of 50 million male customers will lie in the range 9421 to 9452
```

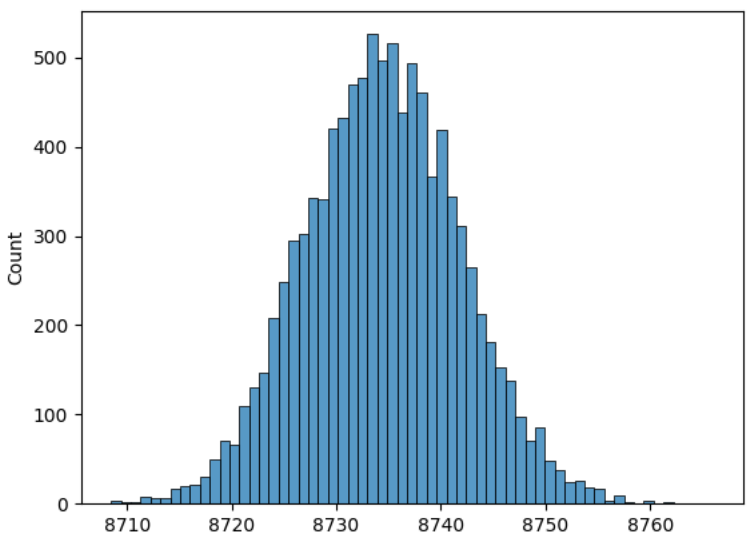
```
(9421.948457184999, 9452.927967165468)
```

```
#For Females(analysing the population mean)
boostrapped_mean_arr=[]
m=len(df3)
for i in range(10000):
  boostrapped_mean=np.random.choice(df3,size=m).mean()
  boostrapped_mean_arr.append(boostrapped_mean)
print(np.mean(boostrapped_mean_arr))
```

```
8734.790964545795
```

```
sns.histplot(boostrapped_mean_arr)
```

```
<Axes: ylabel='Count'>
```




```
#For 95% CI
x1=np.percentile(boostrapped_mean_arr,2.5)
x2=np.percentile(boostrapped_mean_arr,97.5)
(x1,x2) #Lower and Upper limit
#So the average spending of 50 million female customers will lie in the range 8709 to 8760
```

```
(8709.911389524996, 8760.16391789204)
```

```
#As we can conclude that the confidence interval of the two samples do not coincides so we can say that the two populations are different fr
```

```
#Calculting the average spending for men and women for 90% and 99% CI:
#For men
x1=np.percentile(boostrapped_mean_arr,5)
x2=np.percentile(boostrapped_mean_arr,95)
(x1,x2)
```

```
(9424.182298272337, 9450.499850335176)
```

```
#For women
boostrapped_mean_arr1=[]
m=len(df3)
for i in range(10000):
  boostrapped_mean=np.random.choice(df3,size=m).mean()
  boostrapped_mean_arr1.append(boostrapped_mean)
x1=np.percentile(boostrapped_mean_arr1,5)
x2=np.percentile(boostrapped_mean_arr1,95)
(x1,x2)
 #As we can conclude that the confidence interval of the two samples do not coincides so we can say that the two populations are different fr
```

```
(8713.382685609937, 8755.921422733398)
```

```
#For 99% CI:For male
x1=np.percentile(boostrapped_mean_arr,0.5)
x2=np.percentile(boostrapped_mean_arr,99.5)
(x1,x2)
```

```
(9417.1555629932, 9458.02190094844)
```

```
#For Female
x1=np.percentile(boostrapped_mean_arr1,0.5)
x2=np.percentile(boostrapped_mean_arr1,99.5)
(x1,x2)
 #As we can conclude that the confidence interval of the two samples do not coincides so we can say that the two populations are different fr
```

```
(8700.411154047228, 8768.775329028269)
```

```
ttest_ind(df2,df3)
```

```
Ttest_indResult(statistic=44.837957934353966, pvalue=0.0)
```

```
 #Checking the distributions of male and female average spendind using KS Test with 95% CI
kstest(df2,df3)
#Here the p value is almost 0.0 so the two distributions are very different from each other
```

```
KstestResult(statistic=0.08316914937650632, pvalue=0.0, statistic_location=11369, statistic_sign=-1)
```

```
#Let us calculate the Average age of the population mean with 95% CI
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
```

```
x1=df[df['Age']=="0-17"][['User_ID','Age']]
x1['Rand_Age']=np.random.choice(np.arange(18),size=len(x1))
x1
```

	User_ID	Age	Rand_Age
0	1000001	0-17	14

```
x2=df[df['Age']=="18-25"][['User_ID','Age']]
x2['Rand_Age']=np.random.choice(np.arange(18,26),size=len(x2))
x2
```

	User_ID	Age	Rand_Age
70	1000018	18-25	25
71	1000018	18-25	22
72	1000018	18-25	22
73	1000018	18-25	20
74	1000018	18-25	20
...	...	...	...
550000	1005936	18-25	23
550015	1005957	18-25	19
550017	1005959	18-25	18
550020	1005964	18-25	22
550032	1005985	18-25	21

99660 rows × 3 columns

```
x3=df[df['Age']=="26-35"][['User_ID','Age']]
x3['Rand_Age']=np.random.choice(np.arange(26,36),size=len(x3))
x3
```

	User_ID	Age	Rand_Age
5	1000003	26-35	34
9	1000005	26-35	27
10	1000005	26-35	27
11	1000005	26-35	33
12	1000005	26-35	31
...	...	...	...
550058	1006024	26-35	26
550059	1006025	26-35	33
550061	1006029	26-35	34
550064	1006035	26-35	30
550065	1006036	26-35	31

219587 rows × 3 columns

```
x4=df[df['Age']=="36-45"][['User_ID','Age']]
x4['Rand_Age']=np.random.choice(np.arange(36,46),size=len(x4))
x4
```

	User_ID	Age	Rand_Age
18	1000007	36-45	43
29	1000010	36-45	41
30	1000010	36-45	44

```
x5=df[df['Age']=="46-50"][['User_ID','Age']]
x5['Rand_Age']=np.random.choice(np.arange(46,51),size=len(x5))
x5
```

	User_ID	Age	Rand_Age
6	1000004	46-50	50
7	1000004	46-50	49
8	1000004	46-50	49
52	1000013	46-50	48
53	1000013	46-50	48
...	...	...	...
550041	1006000	46-50	50
550043	1006003	46-50	49
550052	1006016	46-50	50
550062	1006032	46-50	48
550067	1006039	46-50	46

45701 rows × 3 columns

```
x6=df[df['Age']=="51-55"][['User_ID','Age']]
x6['Rand_Age']=np.random.choice(np.arange(51,56),size=len(x6))
x6
```

	User_ID	Age	Rand_Age
14	1000006	51-55	51
15	1000006	51-55	54
16	1000006	51-55	54
17	1000006	51-55	54
67	1000017	51-55	52
...	...	...	...
549985	1005916	51-55	53
550004	1005940	51-55	53
550037	1005993	51-55	52
550042	1006002	51-55	54
550063	1006033	51-55	52

38501 rows × 3 columns

```
x7=df[df['Age']=="55+"][['User_ID','Age']]
x7['Rand_Age']=np.random.choice(np.arange(55,81),size=len(x7))
x7
```

	User_ID	Age	Rand_Age
4	1000002	55+	60
159	1000031	55+	58
160	1000031	55+	76
161	1000031	55+	73
162	1000031	55+	77

```
x8=pd.concat([x1,x2])
x9=pd.concat([x8,x3])
x10=pd.concat([x9,x4])
x11=pd.concat([x10,x5])
x12=pd.concat([x11,x6])
x13=pd.concat([x12,x7])
x13.drop(['Age'],inplace=True,axis=1)
x13
```

	User_ID	Rand_Age
0	1000001	14
1	1000001	12
2	1000001	16
3	1000001	6
85	1000019	12
...	...	...
549925	1005834	58
549989	1005922	78
550008	1005946	77
550030	1005980	67
550066	1006038	67

550068 rows × 2 columns

```
boostrapped_mean_arr2=[]
m=len(x13)
for i in range(1000):
  boostrapped_mean=np.random.choice(x13["Rand_Age"],size=m).mean()
  boostrapped_mean_arr2.append(boostrapped_mean)
x1=np.percentile(boostrapped_mean_arr2,5)
x2=np.percentile(boostrapped_mean_arr2,95)
(x1,x2) #Calculating the age mean of sample data and analyzying the population mean of 100m customers(the approx range for 90% CI)
```

```
(34.71280378062349, 34.764562744969716)
```

```
sns.histplot(boostrapped_mean_arr2) #The mean of the population will lie between the range 34.71280378062349, 34.764562744969716
```

```
<Axes: ylabel='Count'>
```

```
#Analyzing the average spending for Unmarried and married people
xa=df[df['Marital_Status']==0]['Purchase']
xa
```

```
0          8370
1         15200
2          1422
3          1057
4          7969
          ...
550056      254
550059       48
550062      473
550064      371
550066      365
Name: Purchase, Length: 324731, dtype: int64
```

```
boostrapped_mean_arr3=[]
m=len(xa)
for i in range(1000):
  boostrapped_mean=np.random.choice(xa,size=m).mean()
  boostrapped_mean_arr3.append(boostrapped_mean)
x1=np.percentile(boostrapped_mean_arr3,2.5)
x2=np.percentile(boostrapped_mean_arr3,97.5)
(x1,x2) #The range in which the population mean will lie for 95% CI
```

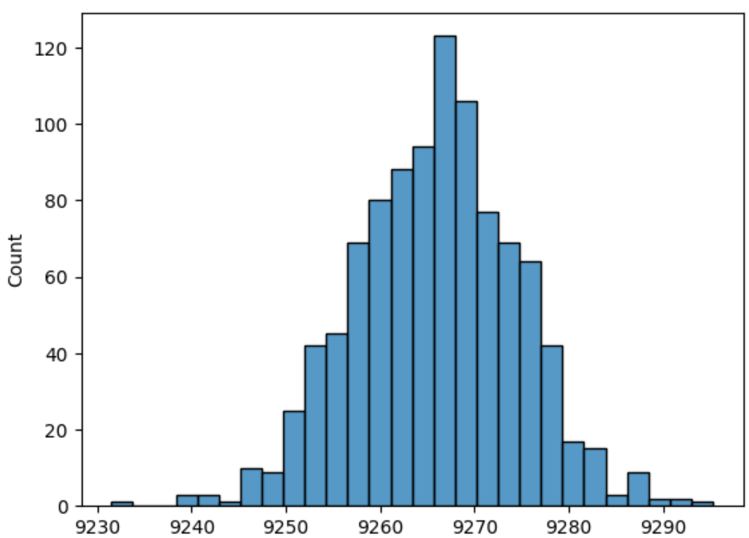
```
(9249.91737176309, 9283.881345713837)
```

```
np.mean(boostrapped_mean_arr3)
```

```
9265.907014886167
```

```
sns.histplot(boostrapped_mean_arr3)
```

```
<Axes: ylabel='Count'>
```




```
#For married people
xb=df[df['Marital_Status']==1]['Purchase']
xb
```

```
6         19215
7         15854
8         15686
9          7871
10         5254
          ...
550060      494
550061      599
550063      368
550065      137
```

```
550067      490
Name: Purchase, Length: 225337, dtype: int64
```

```
boostrapped_mean_arr4=[]
m=len(xb)
for i in range(1000):
  boostrapped_mean=np.random.choice(xb,size=m).mean()
  boostrapped_mean_arr4.append(boostrapped_mean)
x1=np.percentile(boostrapped_mean_arr4,2.5)
x2=np.percentile(boostrapped_mean_arr4,97.5)
(x1,x2) #The range in which the population mean will lie for 95% CI
```

```
(9240.283551857883, 9282.238252816891)
```

```
np.mean(boostrapped_mean_arr4)
```

```
9260.693103098914
```

```
#We can analyze that for the above two diustribution related to marriage the  ,population range overlaps with each other so we can say that t
#We can also see that the means of the two samples are quite similar .Let us statiscally prove that the means are similar or not.
```

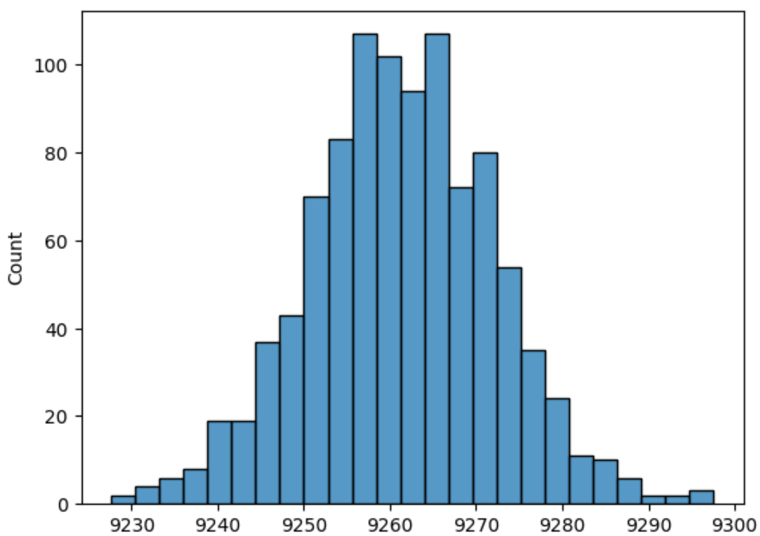
```
ttest_ind(boostrapped_mean_arr3,boostrapped_mean_arr4)
```

```
Ttest_indResult(statistic=11.428315278234258, pvalue=2.4099882686348846e-29)
```

```
#As the p value is very less we can conclude that thes two graphs are not similar.
```

```
sns.histplot(boostrapped_mean_arr4)
```

```
<Axes: ylabel='Count'>
```




```
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

```
m=pd.crosstab(index=df['Gender'],columns=df['City_Category'])
m
```

City_Category	A	B	C
Gender			

```
xstat,p_value,dof,expected=chi2_contingency(m)
```

M	112016	173377	128866

```
alpha=0.05
if p_value<alpha:
  print("Gender depends on City category")
else:
  print("Gender doesnt depend on City category")
```

    Gender depends on City category

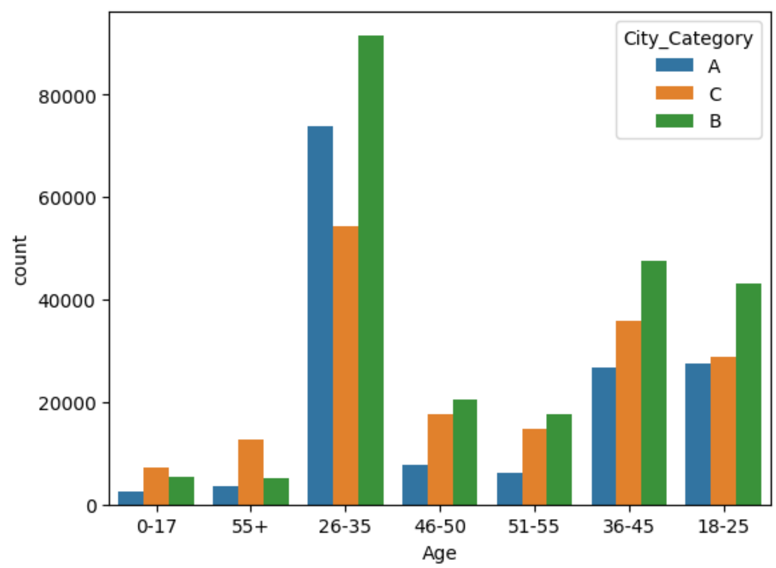
```
#It is statistically sigbificant that both city category and gender is dependent on each other
```

```
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

```
sns.countplot(data=df,x="Age",hue="City_Category") #City Category B is densely populated with more no of people of age groups 26-35
```

    <Axes: xlabel='Age', ylabel='count'>




```
"""
Insights and Recommendations:
Insight 1: The average spending of Male and Female customers after analysing for different confidence intervals of the population differs in
Insight 2:After performing hypothesis testing of the two different distributions of male and female we get a p value (0.0) which is less than
the buying patterns and average spending of the two genders are different.
Insight3:After performing CLT based on customers who are single or married we analysed by performing hypothesis testing that the two distribu
Insight4:After performing column wise dependency we can say that gender has a influence of city category and city_category('B') is highly pop
Insight5:After perofrming CLT on the sampled dataset we analysed the mean age of the population lies in the range( 34.71 to 34.76).
Insight6:

Recommendations:

If Men Spend More:

Targeted Marketing: Design specific marketing campaigns targeting men customers with tailored offers.
Product Placement: Place products with higher margins that are popular among men strategically.
Loyalty Programs: Consider loyalty programs to incentivize frequent men shoppers.
If No Significant Difference:
```

```
Universal Promotions: Implement promotions that appeal to both genders.
Balanced Inventory: Ensure inventory reflects the preferences of both genders.
"""
```

✓ 0s completed at 10:43 PM