BUSINESS CASE STUDY: WALMART: Confidence Interval & CLT

INTRODUCTION:

Walmart is an American multinational retail corporation that operates a chain of supercentres, discount departmental stores, and grocery stores from the United States. Walmart has more than 100 million customers worldwide.

PROBLEM STATEMENT:

The intent of this case study is to do a detailed analysation of given dataset from Walmart and thereby understand the factors which are predominantly influencing the purchase pattern at Walmart. To assess on how gender and age determine the purchasing pattern at the company. We also need to analyse between male and female who made maximum purchase on black Friday sale. This study is driven basically on Confidence Interval & CLT concepts where in sample of the population is taken for the study. This in-depth analysation should result in key actionable recommendations to the organization which could potentially increase the sales for the company.

1. **Import the dataset and do usual data analysis steps like checking the structure & characteristics of the dataset.**

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

```
[7] df = pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv")
```

```
[8] 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

```
[13] df.shape
```

```
(550068, 10)
```

```
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
```

[10]
```
df.describe(include='all')
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purch
count	5.500680e+05	550068	550068	550068	550068.000000	550068	550068	550068.000000	550068.000000	550068.000
unique	NaN	3631	2	7	NaN	3	5	NaN	NaN	N
top	NaN	P00265242	M	26-35	NaN	B	1	NaN	NaN	N
freq	NaN	1880	414259	219587	NaN	231173	193821	NaN	NaN	N
mean	1.003029e+06	NaN	NaN	NaN	8.076707	NaN	NaN	0.409653	5.404270	9263.968
std	1.727592e+03	NaN	NaN	NaN	6.522660	NaN	NaN	0.491770	3.936211	5023.065
min	1.000001e+06	NaN	NaN	NaN	0.000000	NaN	NaN	0.000000	1.000000	12.000
25%	1.001516e+06	NaN	NaN	NaN	2.000000	NaN	NaN	0.000000	1.000000	5823.000
50%	1.003077e+06	NaN	NaN	NaN	7.000000	NaN	NaN	0.000000	5.000000	8047.000
75%	1.004478e+06	NaN	NaN	NaN	14.000000	NaN	NaN	1.000000	8.000000	12054.000
max	1.006040e+06	NaN	NaN	NaN	20.000000	NaN	NaN	1.000000	20.000000	23961.000

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

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

[15]
```
df['Marital_Status'].value_counts()
```

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

[16]
```
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['City_Category'].value_counts()
```

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

2. **Detect Null values & Outliers (using boxplot, “describe” method by checking the difference between mean and median, isnull etc.)**

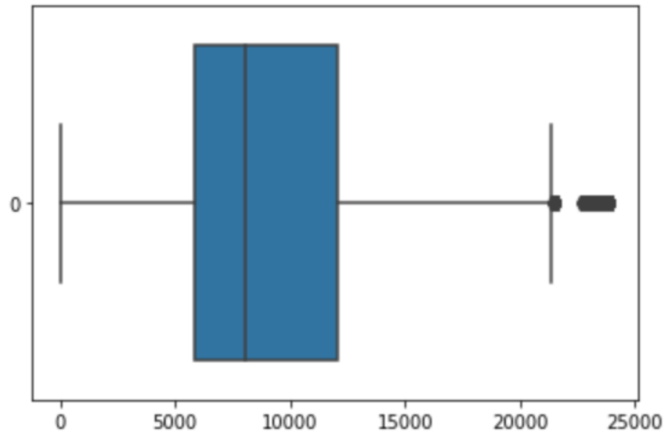
```
df.isnull().sum()
```

```
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
```

There is no null data in the dataset.

```
sns.boxplot(data=df['Purchase'],orient='h')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5f9b6f5fd0>
```



There are outliers in purchase beyond purchase of 22000 rupees

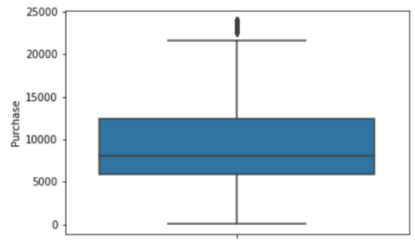
```
df.describe(include='all')
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purch
count	5.500680e+05	550068	550068	550068	550068.000000	550068	550068	550068.000000	550068.000000	550068.000
unique	NaN	3631	2	7	NaN	3	5	NaN	NaN	N
top	NaN	P00265242	M	26-35	NaN	B	1	NaN	NaN	N
freq	NaN	1880	414259	219587	NaN	231173	193821	NaN	NaN	N
mean	1.003029e+06	NaN	NaN	NaN	8.076707	NaN	NaN	0.409653	5.404270	9263.968
std	1.727592e+03	NaN	NaN	NaN	6.522660	NaN	NaN	0.491770	3.936211	5023.065
min	1.000001e+06	NaN	NaN	NaN	0.000000	NaN	NaN	0.000000	1.000000	12.000
25%	1.001516e+06	NaN	NaN	NaN	2.000000	NaN	NaN	0.000000	1.000000	5823.000
50%	1.003077e+06	NaN	NaN	NaN	7.000000	NaN	NaN	0.000000	5.000000	8047.000
75%	1.004478e+06	NaN	NaN	NaN	14.000000	NaN	NaN	1.000000	8.000000	12054.000
max	1.006040e+06	NaN	NaN	NaN	20.000000	NaN	NaN	1.000000	20.000000	23961.000

50% of value gives us the median of the columns in the dataset. Median is more robust to outliers compared to mean

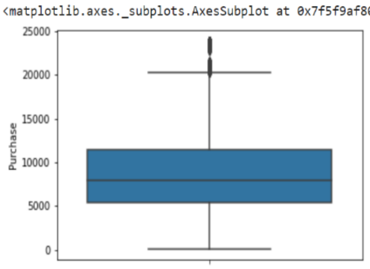
```
sns.boxplot(y='Purchase',data=df[df['Gender']=='M'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f5f9b03ff40>




```
[36] sns.boxplot(y='Purchase',data=df[df['Gender']=='F'])
```

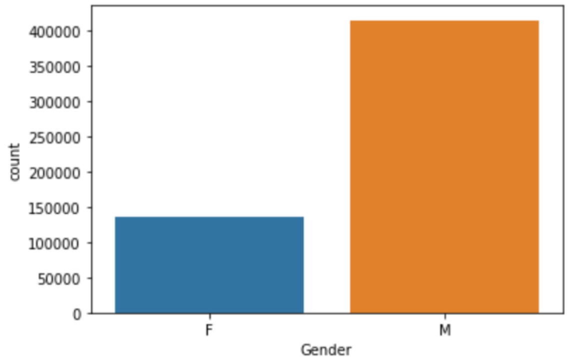
<matplotlib.axes._subplots.AxesSubplot at 0x7f5f9af80f40>



Female customers has more outliers than male customers

```
[37] sns.countplot(x='Gender',data=df)
```

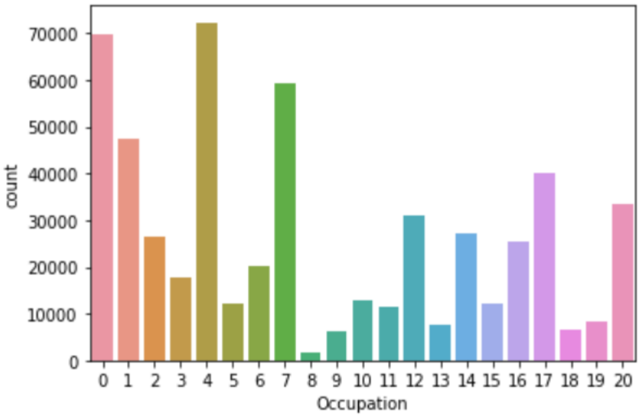
<matplotlib.axes._subplots.AxesSubplot at 0x7f5f9aeef100>



Male customers are higher in general when compared to female customers

```
[38] sns.countplot(x='Occupation',data=df)
```

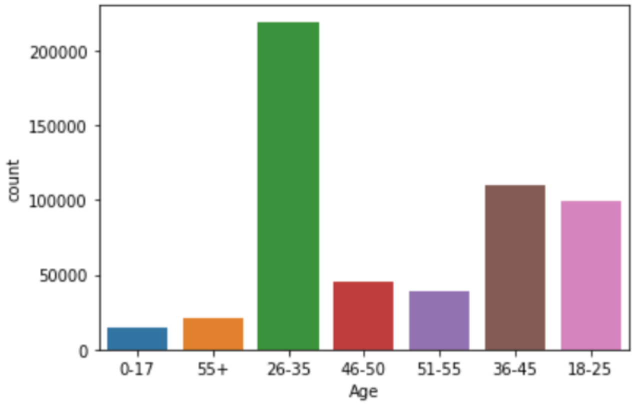
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5f9aeeb880>
```



People with occupation 4 has more purchase rate when compared to other occupation groups

```
[39] sns.countplot(x='Age',data=df)
```

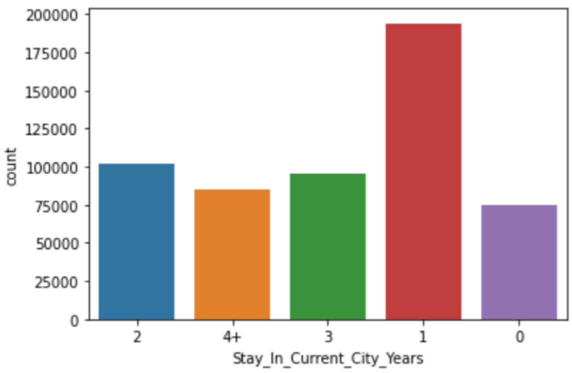
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5f9ae01460>
```



People in age group between 26-35 group has higher purchase count compared to other age groups.

```
[40] sns.countplot(x='Stay_In_Current_City_Years',data=df)
```

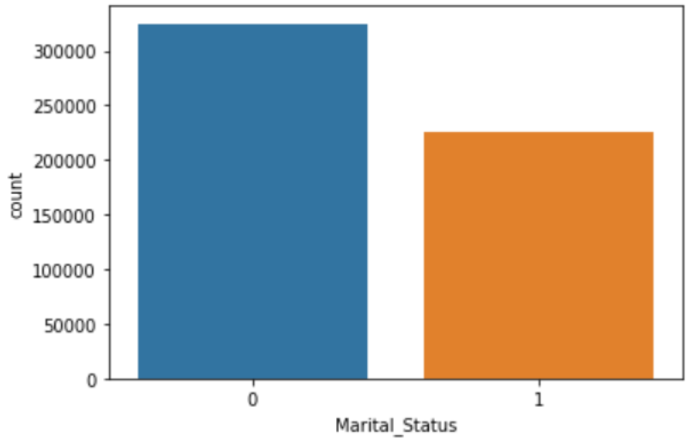
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5f9add58e0>
```



People who stayed in one year in the current city has higher purchase count with Walmart.

```
sns.countplot(x='Marital_Status',data=df)
```

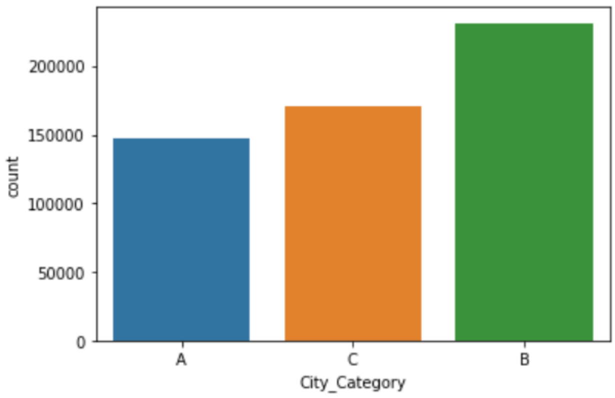
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5f9ac78a00>
```



Unmarried people count is higher than married people with respect to purchase.

```
sns.countplot(x='City_Category',data=df)
```

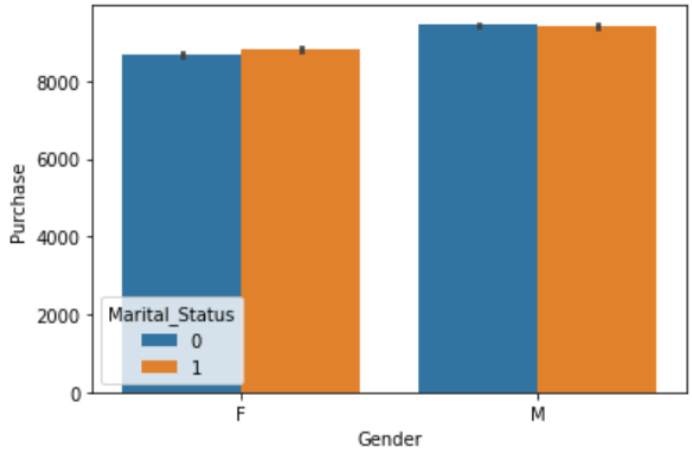
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5f9ad3e970>
```



Category B has higher purchase count than other 2 categories.

```
sns.barplot(x='Gender',y='Purchase',data=df,hue='Marital_Status')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5f9ad06fd0>
```

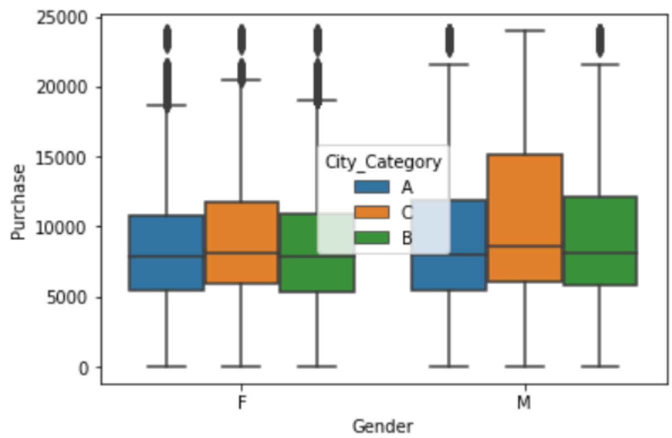


In female, married people have slightly higher purchase rate than unmarried.

In male, unmarried people have slightly higher purchase rate than married.

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e15840fd0>
```



City Category C has more purchase in both male and female groups.

3. **Do some data exploration steps like:**
    - **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.**

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

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

```
[29] df.groupby('Gender')[['Purchase']].sum()
```

Gender	Purchase
F	1186232642
M	3909580100

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

Gender	Purchase
F	8734.565765
M	9437.526040

```
[25] pd.crosstab(index=df['Gender'],columns=df['Age'],values=df['Purchase'],aggfunc='mean')
```

Gender \ Age	0-17	18-25	26-35	36-45	46-50	51-55	55+
F	8338.771985	8343.180201	8728.251754	8959.844056	8842.098947	9042.449666	9007.036199
M	9235.173670	9440.942971	9410.337578	9453.193643	9357.471509	9705.094802	9438.195603

```
pd.crosstab(index=df['Gender'],columns=df['City_Category'],values=df['Purchase'],aggfunc='mean')
```

Gender \ City_Category	A	B	C
F	8579.708576	8540.677694	9130.107518
M	9017.834470	9354.854433	9913.567248

```
[27] pd.crosstab(index=df['Gender'],columns=df['Marital_Status'],values=df['Purchase'],aggfunc='mean')
```

Gender \ Marital_Status	0	1
F	8679.845815	8810.249789
M	9453.756740	9413.817605

- **Inference after computing the average female and male expenses.**
  1. Mean purchase based on gender wise for different age, city and marital status are listed above.
  2. Its observed that both in male and female, people between age group 51-55 has highest average purchase value.
  3. Married female has highest average purchase when compared to unmarried female where unmarried male has slightly higher average purchase value compared to married male.

4. **Use the Central limit theorem to compute the interval. Change the sample size to observe the distribution of the mean of the expenses by female and male customers**
    - **Use the sample average to find out an interval within which the population average will lie. Using the sample of female customers, you will calculate the interval within which the average spending of 50 million male and female customers may lie.**
    - **The interval that you calculated is called Confidence Interval. The width of the interval is mostly decided by the business: Typically 90%, 95%, or 99%. Play around with the width parameter and report the observations.**

MALE:

MALE

```
[32] df_male=df[df['Gender']=='M']
```

```
[36] male_pop_mean = round(df_male["Purchase"].mean(), 2)
```

```
[34] male_pop_std = round(df_male["Purchase"].std(), 2)
```

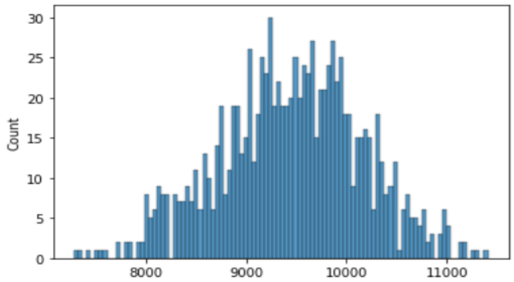
```
[37] male_pop_mean

9437.53
```

```
[38] male_pop_std

5092.19
```

SAMPLE SIZE = 50

```
n = 50
male_sample_mean = []
male_sample_std = []
for i in range(1000):
 sample = df_male["Purchase"].sample(n)
 male_sample_mean.append(round(sample.mean(), 2))
 male_sample_std.append(round(sample.std(), 2))
```

```
[104] sns.histplot(male_sample_mean, bins = 100)

<matplotlib.axes._subplots.AxesSubplot at 0x7f5e13c88d30>
```



```
[105] male_sample_mean50=np.mean(male_sample_mean)
```

```
male_sample_mean50
```

```
9450.487919999998
```

```
[107] male_sample_mean50,male_pop_mean
```

```
(9450.487919999998, 9437.53)
```

```
[108] male_std_sample_50=np.std(male_sample_std)
      male_std_sample_50
```

```
446.2890702127265
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[109] (male_sample_mean50-1.645*(male_pop_std/np.sqrt(n)),male_sample_mean50+1.645*(male_pop_std/np.sqrt(n)))
```

```
(8265.85035565028, 10635.125484349715)
```

```
male_sample_mean50,male_pop_mean
```

```
(9450.487919999998, 9437.53)
```

CASE2: 95%

```
[111] (male_sample_mean50-1.96*(male_pop_std/np.sqrt(n)),male_sample_mean50+1.96*(male_pop_std/np.sqrt(n)))
```

```
(8039.004864604591, 10861.970975395405)
```

CASE3 : 99%

```
[112] (male_sample_mean50-2.575*(male_pop_std/np.sqrt(n)),male_sample_mean50+2.575*(male_pop_std/np.sqrt(n)))
```

```
(7596.11604875348, 11304.859791246516)
```

SAMPLE SIZE = 500

```
[113] n = 500
      male_sample_mean = []
      male_sample_std = []
      for i in range(1000):
       sample = df_male["Purchase"].sample(n)
       male_sample_mean.append(round(sample.mean(), 2))
       male_sample_std.append(round(sample.std(), 2))
```

```
male_sample_mean500=np.mean(male_sample_mean)
male_sample_mean500
```

```
9439.35564
```

```
[116] male_sample_mean500,male_pop_mean
```

```
(9439.35564, 9437.53)
```

```
[117] male_std_sample_500=np.std(male_sample_std)
      male_std_sample_500
```

```
140.71621103762567
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[118] (male_sample_mean500-1.645*(male_pop_std/np.sqrt(n)),male_sample_mean500+1.645*(male_pop_std/np.sqrt(n)))
```

```
(9064.740349486061, 9813.970930513939)
```

CASE2: 95%

```
[119] (male_sample_mean500-1.96*(male_pop_std/np.sqrt(n)),male_sample_mean500+1.96*(male_pop_std/np.sqrt(n)))
```

```
(8993.00550662169, 9885.70577337831)
```

CASE3: 99%

```
[120] (male_sample_mean500-2.575*(male_pop_std/np.sqrt(n)),male_sample_mean500+2.575*(male_pop_std/np.sqrt(n)))
```

```
(8852.951765791251, 10025.759514208748)
```

SAMPLE SIZE = 1000

```
n = 1000
male_sample_mean = []
male_sample_std = []
for i in range(1000):
 sample = df_male["Purchase"].sample(n)
 male_sample_mean.append(round(sample.mean(), 2))
 male_sample_std.append(round(sample.std(), 2))
```

```
male_sample_mean1000=np.mean(male_sample_mean)
male_sample_mean1000
```

```
9435.600330000001
```

```
[124] male_sample_mean1000,male_pop_mean
```

```
(9435.600330000001, 9437.53)
```

```
[126] male_std_sample_1000=np.std(male_sample_std)
male_std_sample_1000
```

```
101.54664074655744
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[127] (male_sample_mean1000-1.645*(male_pop_std/np.sqrt(n)),male_sample_mean1000+1.645*(male_pop_std/np.sqrt(n)))
```

```
(9170.707317741426, 9700.493342258576)
```

CASE2: 95%

```
[128] (male_sample_mean1000-1.96*(male_pop_std/np.sqrt(n)),male_sample_mean1000+1.96*(male_pop_std/np.sqrt(n)))
```

```
(9119.983123904678, 9751.217536095324)
```

CASE3: 99%

```
[129] (male_sample_mean1000-2.575*(male_pop_std/np.sqrt(n)),male_sample_mean1000+2.575*(male_pop_std/np.sqrt(n)))
```

```
(9020.95017403293, 9850.250485967072)
```

SAMPLE SIZE = 5000

```
n = 5000
male_sample_mean = []
male_sample_std = []
for i in range(1000):
 sample = df_male["Purchase"].sample(n)
 male_sample_mean.append(round(sample.mean(), 2))
 male_sample_std.append(round(sample.std(), 2))
```

```
male_sample_mean5000=np.mean(male_sample_mean)
male_sample_mean5000
```

```
9436.2099
```

```
[141] male_sample_mean5000,male_pop_mean
```

```
(9436.2099, 9437.53)
```

```
[142] male_std_sample_5000=np.std(male_sample_std)
male_std_sample_5000
```

```
31.13061849452883
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[143] (male_sample_mean5000-1.645*(male_pop_std/np.sqrt(n)),male_sample_mean5000+1.645*(male_pop_std/np.sqrt(n)))
```

```
(9352.4433745, 9519.9764255)
```

CASE2: 95%

```
[144] (male_sample_mean5000-1.96*(male_pop_std/np.sqrt(n)),male_sample_mean5000+1.96*(male_pop_std/np.sqrt(n)))
```

```
(9336.402976, 9536.016824)
```

CASE3: 99%

```
[145] (male_sample_mean5000-2.575*(male_pop_std/np.sqrt(n)),male_sample_mean5000+2.575*(male_pop_std/np.sqrt(n)))
```

```
(9305.0860075, 9567.3337925)
```

SAMPLE SIZE = 10000

```
n = 10000
male_sample_mean = []
male_sample_std = []
for i in range(1000):
 sample = df_male["Purchase"].sample(n)
 male_sample_mean.append(round(sample.mean(), 2))
 male_sample_std.append(round(sample.std(), 2))
```

```
male_sample_mean10000=np.mean(male_sample_mean)
male_sample_mean10000
```

```
9435.41938
```

```
[149] male_sample_mean10000,male_pop_mean
```

```
(9435.41938, 9437.53)
```

```
[150] male_std_sample_10000=np.std(male_sample_std)
      male_std_sample_10000
```

```
32.0742231113631
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[151] (male_sample_mean10000-1.645*(male_pop_std/np.sqrt(n)),male_sample_mean10000+1.645*(male_pop_std/np.sqrt(n)))
```

```
(9351.6528545, 9519.185905499999)
```

CASE2: 95%

```
[152] (male_sample_mean10000-1.96*(male_pop_std/np.sqrt(n)),male_sample_mean10000+1.96*(male_pop_std/np.sqrt(n)))
```

```
(9335.612455999999, 9535.226304)
```

CASE3: 99%

```
[153] (male_sample_mean10000-2.575*(male_pop_std/np.sqrt(n)),male_sample_mean10000+2.575*(male_pop_std/np.sqrt(n)))
```

```
(9304.2954875, 9566.5432725)
```

SAMPLE SIZE = 35000

```
[154] n = 35000
      male_sample_mean = []
      male_sample_std = []
      for i in range(1000):
       sample = df_male["Purchase"].sample(n)
       male_sample_mean.append(round(sample.mean(), 2))
       male_sample_std.append(round(sample.std(), 2))
```

```
[155] male_sample_mean35000=np.mean(male_sample_mean)
      male_sample_mean35000
```

```
9437.57486
```

```
[156] male_sample_mean35000,male_pop_mean
```

```
(9437.57486, 9437.53)
```

```
[157] male_std_sample_35000=np.std(male_sample_std)
      male_std_sample_35000
```

```
16.137904145259995
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[158] (male_sample_mean35000-1.645*(male_pop_std/np.sqrt(n)),male_sample_mean35000+1.645*(male_pop_std/np.sqrt(n)))
```

```
(9392.799768728362, 9482.349951271639)
```

CASE2: 95%

```
[159] (male_sample_mean35000-1.96*(male_pop_std/np.sqrt(n)),male_sample_mean35000+1.96*(male_pop_std/np.sqrt(n)))
```

```
(9384.225815080603, 9490.923904919398)
```

CASE3: 99%

```
[160] (male_sample_mean35000-2.575*(male_pop_std/np.sqrt(n)),male_sample_mean35000+2.575*(male_pop_std/np.sqrt(n)))
```

```
(9367.486191292119, 9507.663528707883)
```

FEMALE:

FEMALE

[3]
```
df_female=df[df['Gender']=='F']
```

[6]
```
female_pop_mean = round(df_female["Purchase"].mean(), 2)
female_pop_mean
```

```
8734.57
```

[7]
```
female_pop_std = round(df_female["Purchase"].std(), 2)
female_pop_std
```

```
4767.23
```

SAMPLE SIZE = 50

[8]
```
n = 50
female_sample_mean = []
female_sample_std = []
for i in range(1000):
 sample = df_female["Purchase"].sample(n)
 female_sample_mean.append(round(sample.mean(), 2))
 female_sample_std.append(round(sample.std(), 2))
```

```
female_sample_mean50=np.mean(female_sample_mean)
```

[12]
```
female_sample_mean50,female_pop_mean
```

```
(8741.11184, 8734.57)
```

[13]
```
female_std_sample_50=np.std(female_sample_std)
female_std_sample_50
```

```
509.8225918860403
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[14] (female_sample_mean50-1.645*(female_pop_std/np.sqrt(n)),female_sample_mean50+1.645*(female_pop_std/np.sqrt(n)))
```

```
(7632.072362703414, 9850.151317296586)
```

```
[15] female_sample_mean50,female_pop_mean
```

```
(8741.11184, 8734.57)
```

CASE2: 95%

```
[16] (female_sample_mean50-1.96*(female_pop_std/np.sqrt(n)),female_sample_mean50+1.96*(female_pop_std/np.sqrt(n)))
```

```
(7419.70310109343, 10062.52057890657)
```

CASE3: 99%

```
[17] (female_sample_mean50-2.575*(female_pop_std/np.sqrt(n)),female_sample_mean50+2.575*(female_pop_std/np.sqrt(n)))
```

```
(7005.077399854888, 10477.14628014511)
```

SAMPLE SIZE = 1000

```
[19] n = 1000
     female_sample_mean = []
     female_sample_std = []
     for i in range(1000):
      sample = df_female["Purchase"].sample(n)
      female_sample_mean.append(round(sample.mean(), 2))
      female_sample_std.append(round(sample.std(), 2))
```

```
female_sample_mean1000=np.mean(female_sample_mean)
```

```
[22] female_sample_mean1000,female_pop_mean
```

```
(8738.563370000002, 8734.57)
```

```
[23] female_std_sample_1000=np.std(female_sample_std)
     female_std_sample_1000
```

```
109.63080373694977
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[24] (female_sample_mean1000-1.645*(female_pop_std/np.sqrt(n)),female_sample_mean1000+1.645*(female_pop_std/np.sqrt(n)))
```

```
(8490.574603903402, 8986.552136096601)
```

```
[25] female_sample_mean1000,female_pop_mean
```

```
(8738.563370000002, 8734.57)
```

CASE2: 95%

```
[26] (female_sample_mean1000-1.96*(female_pop_std/np.sqrt(n)),female_sample_mean1000+1.96*(female_pop_std/np.sqrt(n)))
```

```
(8443.087393374266, 9034.039346625737)
```

CASE3: 99%

```
[27] (female_sample_mean1000-2.575*(female_pop_std/np.sqrt(n)),female_sample_mean1000+2.575*(female_pop_std/np.sqrt(n)))
```

```
(8350.374268055475, 9126.752471944528)
```

SAMPLE SIZE = 10000

```
[28] n = 10000
     female_sample_mean = []
     female_sample_std = []
     for i in range(1000):
      sample = df_female["Purchase"].sample(n)
      female_sample_mean.append(round(sample.mean(), 2))
      female_sample_std.append(round(sample.std(), 2))
```

```
female_sample_mean10000=np.mean(female_sample_mean)
```

```
[32] female_sample_mean10000,female_pop_mean
```

```
(8736.22249, 8734.57)
```

```
[33] female_std_sample_10000=np.std(female_sample_std)
     female_std_sample_10000
```

```
33.25474337594413
```

CONFIDENCE INTERVAL

CASE1: 90%

```
(female_sample_mean10000-1.645*(female_pop_std/np.sqrt(n)),female_sample_mean10000+1.645*(female_pop_std/np.sqrt(n)))
```

```
(8657.8015565, 8814.6434235)
```

CASE2: 95%

```
[35] (female_sample_mean10000-1.96*(female_pop_std/np.sqrt(n)),female_sample_mean10000+1.96*(female_pop_std/np.sqrt(n)))
```

```
(8642.784782, 8829.660198)
```

CASE3: 99%

```
[36] (female_sample_mean10000-2.575*(female_pop_std/np.sqrt(n)),female_sample_mean10000+2.575*(female_pop_std/np.sqrt(n)))
```

```
(8613.4663175, 8858.9786625)
```

SAMPLE SIZE = 25000

```
[37] n = 25000
     female_sample_mean = []
     female_sample_std = []
     for i in range(1000):
      sample = df_female["Purchase"].sample(n)
      female_sample_mean.append(round(sample.mean(), 2))
      female_sample_std.append(round(sample.std(), 2))
```

```
female_sample_mean25000=np.mean(female_sample_mean)
```

```
[40] female_sample_mean25000,female_pop_mean

(8734.45647, 8734.57)
```

```
[41] female_std_sample_25000=np.std(female_sample_std)
     female_std_sample_25000

19.828961480622237
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[42] (female_sample_mean25000-1.645*(female_pop_std/np.sqrt(n)),female_sample_mean25000+1.645*(female_pop_std/np.sqrt(n)))

(8684.85871678068, 8784.054223219318)
```

CASE2: 95%

```
[43] (female_sample_mean25000-1.96*(female_pop_std/np.sqrt(n)),female_sample_mean25000+1.96*(female_pop_std/np.sqrt(n)))

(8675.361274674851, 8793.551665325147)
```

CASE3: 99%

```
[44] (female_sample_mean25000-2.575*(female_pop_std/np.sqrt(n)),female_sample_mean25000+2.575*(female_pop_std/np.sqrt(n)))

(8656.818649611094, 8812.094290388904)
```

- As the number of sample increases, the sample mean will be nearer to the population mean and it follows a normal distribution.
- Also, as the number of sample increases, the standard deviation decreases.
- As the number of sample increases, error also decreases.

5. Conclude the results and check if the confidence intervals of average male and female spends are overlapping or not overlapping. How can Walmart leverage this conclusion to make changes or improvements?

SAMPLE NUMBERS	Width of the interval	MALE - CI	FEMALE - CI
50	95%	(8039.004, 10861.970)	(7419.703, 10062.52)
1000	95%	(9119.983, 9751.217)	(8443.087, 9034.039)
10000	95%	(9335.612, 9535.226)	(8642.784, 8829.660)

- The overlapping is predominant for lesser samples and gradually decreases as sample size increases.
- In general, mean of male purchases is higher than female customers.
- So, Walmart can focus on additional initiatives to attract more female customers.
- Special discounts can be rolled out on women’s day, mother’s day on female products to increase the number of female customers

6. **Perform the same activity for Married vs Unmarried and Age**
   - **For Age, you can try bins based on life stages: 0-17, 18-25, 26-35, 36-50, 51+ years.**

MARRIED VS UNMARIIED:

UNMARRIED:

UNMARRIED

```
df_unmarried=df[df['Marital_Status']==0]
```

```
[52] pop_mean_unmarried = round(df_unmarried["Purchase"].mean(), 2)
     pop_mean_unmarried
```

```
9265.91
```

```
[53] pop_std_unmarried = round(df_unmarried["Purchase"].std(), 2)
     pop_std_unmarried
```

```
5027.35
```

SAMPLES = 500

```
[62] n = 500
     sample_mean_unmarried = []
     sample_std_unmarried = []
     for i in range(1000):
      sample = df_unmarried["Purchase"].sample(n)
      sample_mean_unmarried.append(round(sample.mean(), 2))
      sample_std_unmarried.append(round(sample.std(), 2))
```

```
[63] mean_sample_500=np.mean(sample_mean_unmarried)
     mean_sample_500
```

```
9268.630780000001
```

```
[64] std_sample_500=np.std(sample_std_unmarried)
     std_sample_500
```

```
147.42132788697842
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[65] (mean_sample_500-1.645*(pop_std_unmarried/np.sqrt(n)),mean_sample_500+1.645*(pop_std_unmarried/np.sqrt(n)))
```

```
(8898.785550194112, 9638.47600980589)
```

CASE2: 95%

```
[67] (mean_sample_500-1.96*(pop_std_unmarried/np.sqrt(n)),mean_sample_500+1.96*(pop_std_unmarried/np.sqrt(n)))
```

```
(8827.964123210006, 9709.297436789997)
```

CASE3: 99%

```
[66] (mean_sample_500-2.575*(pop_std_unmarried/np.sqrt(n)),mean_sample_500+2.575*(pop_std_unmarried/np.sqrt(n)))
```

```
(8689.693718145798, 9847.567841854205)
```

SAMPLES = 1000

```
[69] n = 1000
     sample_mean_unmarried = []
     sample_std_unmarried = []
     for i in range(1000):
      sample = df_unmarried["Purchase"].sample(n)
      sample_mean_unmarried.append(round(sample.mean(), 2))
      sample_std_unmarried.append(round(sample.std(), 2))
```

```
[70] mean_sample_1000=np.mean(sample_mean_unmarried)
     mean_sample_1000
```

```
9264.85166
```

```
[71] std_sample_1000=np.std(sample_std_unmarried)
     std_sample_1000
```

```
103.06991752316289
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[72] (mean_sample_1000-1.645*(pop_std_unmarried/np.sqrt(n)),mean_sample_1000+1.645*(pop_std_unmarried/np.sqrt(n)))
```

```
(9003.331590014759, 9526.371729985241)
```

CASE2: 95%

```
[73] (mean_sample_1000-1.96*(pop_std_unmarried/np.sqrt(n)),mean_sample_1000+1.96*(pop_std_unmarried/np.sqrt(n)))
```

```
(8953.25327874099, 9576.450041259011)
```

CASE3: 99%

```
[74] (mean_sample_1000-2.575*(pop_std_unmarried/np.sqrt(n)),mean_sample_1000+2.575*(pop_std_unmarried/np.sqrt(n)))
```

```
(8855.481337682677, 9674.221982317324)
```

SAMPLES = 5000

[75]
```
n = 5000
sample_mean_unmarried = []
sample_std_unmarried = []
for i in range(1000):
 sample = df_unmarried["Purchase"].sample(n)
 sample_mean_unmarried.append(round(sample.mean(), 2))
 sample_std_unmarried.append(round(sample.std(), 2))
```

[76]
```
mean_sample_5000=np.mean(sample_mean_unmarried)
mean_sample_5000
```

```
9262.99936
```

[77]
```
std_sample_5000=np.std(sample_std_unmarried)
std_sample_5000
```

```
44.974002493661814
```

CONFIDENCE INTERVAL

CASE1: 90%

[78]
```
(mean_sample_5000-1.645*(pop_std_unmarried/np.sqrt(n)),mean_sample_5000+1.645*(pop_std_unmarried/np.sqrt(n)))
```

```
(9146.0440292065, 9379.9546907935)
```

CASE2: 95%

[79]
```
(mean_sample_5000-1.96*(pop_std_unmarried/np.sqrt(n)),mean_sample_5000+1.96*(pop_std_unmarried/np.sqrt(n)))
```

```
(9123.64832756519, 9402.35039243481)
```

CASE3: 99%

[80]
```
(mean_sample_5000-2.575*(pop_std_unmarried/np.sqrt(n)),mean_sample_5000+2.575*(pop_std_unmarried/np.sqrt(n)))
```

```
(9079.923386265493, 9446.075333734507)
```

SAMPLES = 10000

```python
n = 10000
sample_mean_unmarried = []
sample_std_unmarried = []
for i in range(1000):
 sample = df_unmarried["Purchase"].sample(n)
 sample_mean_unmarried.append(round(sample.mean(), 2))
 sample_std_unmarried.append(round(sample.std(), 2))
```

```python
[82] mean_sample_10000=np.mean(sample_mean_unmarried)
mean_sample_10000
```

9268.12788

```python
[83] std_sample_10000=np.std(sample_std_unmarried)
std_sample_10000
```

32.17402683432554

CONFIDENCE INTERVAL

CASE1: 90%

```python
[84] (mean_sample_10000-1.645*(pop_std_unmarried/np.sqrt(n)),mean_sample_10000+1.645*(pop_std_unmarried/np.sqrt(n)))
```

(9185.4279725, 9350.8277875)

CASE2: 95%

```python
[85] (mean_sample_10000-1.96*(pop_std_unmarried/np.sqrt(n)),mean_sample_10000+1.96*(pop_std_unmarried/np.sqrt(n)))
```

(9169.59182, 9366.66394)

CASE3: 99%

```python
[86] (mean_sample_10000-2.575*(pop_std_unmarried/np.sqrt(n)),mean_sample_10000+2.575*(pop_std_unmarried/np.sqrt(n)))
```

(9138.6736175, 9397.5821425)

MARRIED:

MARRIED

```
[3] df_married=df[df['Marital_Status']==1]
```

```
[4] pop_mean_married = round(df_married["Purchase"].mean(), 2)
    pop_mean_married
```

```
9261.17
```

```
[5] pop_std_married = round(df_married["Purchase"].std(), 2)
    pop_std_married
```

```
5016.9
```

SAMPLES = 500

```
[6] n = 500
    sample_mean_married = []
    sample_std_married = []
    for i in range(1000):
     sample = df_married["Purchase"].sample(n)
     sample_mean_married.append(round(sample.mean(), 2))
     sample_std_married.append(round(sample.std(), 2))
```

```
[7] mean_sample_500=np.mean(sample_mean_married)
    mean_sample_500
```

```
9252.348940000002
```

```
[8] std_sample_500=np.std(sample_std_married)
    std_sample_500
```

```
143.23281367632734
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[9] (mean_sample_500-1.645*(pop_std_married/np.sqrt(n)),mean_sample_500+1.645*(pop_std_married/np.sqrt(n)))
```

```
(8883.272481545117, 9621.425398454887)
```

CASE2: 95%

```
[10] (mean_sample_500-1.96*(pop_std_married/np.sqrt(n)),mean_sample_500+1.96*(pop_std_married/np.sqrt(n)))
```

```
(8812.598266096309, 9692.099613903694)
```

CASE3: 99%

```
[11] (mean_sample_500-2.575*(pop_std_married/np.sqrt(n)),mean_sample_500+2.575*(pop_std_married/np.sqrt(n)))
```

```
(8674.61527402959, 9830.082605970414)
```

SAMPLES = 1000

```
n = 1000
sample_mean_married = []
sample_std_married = []
for i in range(1000):
 sample = df_married["Purchase"].sample(n)
 sample_mean_married.append(round(sample.mean(), 2))
 sample_std_married.append(round(sample.std(), 2))
```

[13]
```
mean_sample_1000=np.mean(sample_mean_married)
mean_sample_1000
```

```
9268.06463
```

[14]
```
std_sample_1000=np.std(sample_std_married)
std_sample_1000
```

```
99.22942524423085
```

CONFIDENCE INTERVAL

CASE1: 90%

[15]
```
(mean_sample_1000-1.645*(pop_std_married/np.sqrt(n)),mean_sample_1000+1.645*(pop_std_married/np.sqrt(n)))
```

```
(9007.088163450237, 9529.041096549765)
```

CASE2: 95%

[16]
```
(mean_sample_1000-1.96*(pop_std_married/np.sqrt(n)),mean_sample_1000+1.96*(pop_std_married/np.sqrt(n)))
```

```
(8957.113946451345, 9579.015313548656)
```

CASE3: 99%

[17]
```
(mean_sample_1000-2.575*(pop_std_married/np.sqrt(n)),mean_sample_1000+2.575*(pop_std_married/np.sqrt(n)))
```

```
(8859.545237072558, 9676.584022927444)
```

SAMPLES = 5000

```
[18] n = 5000
     sample_mean_married = []
     sample_std_married = []
     for i in range(1000):
      sample = df_married["Purchase"].sample(n)
      sample_mean_married.append(round(sample.mean(), 2))
      sample_std_married.append(round(sample.std(), 2))
```

```
[19] mean_sample_5000=np.mean(sample_mean_married)
     mean_sample_5000
```

```
9261.69259
```

```
[20] std_sample_5000=np.std(sample_std_married)
     std_sample_5000
```

```
46.10959493145759
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[ ] (mean_sample_5000-1.645*(pop_std_married/np.sqrt(n)),mean_sample_5000+1.645*(pop_std_married/np.sqrt(n)))
```

```
(9144.980366053405, 9378.404813946596)
```

CASE2: 95%

```
[ ] (mean_sample_5000-1.96*(pop_std_married/np.sqrt(n)),mean_sample_5000+1.96*(pop_std_married/np.sqrt(n)))
```

```
(9122.631216787036, 9400.753963212965)
```

CASE3: 99%

```
[ ] (mean_sample_5000-2.575*(pop_std_married/np.sqrt(n)),mean_sample_5000+2.575*(pop_std_married/np.sqrt(n)))
```

```
(9078.99716345746, 9444.388016542542)
```

SAMPLES = 10000

```
[24] n = 10000
     sample_mean_married = []
     sample_std_married = []
     for i in range(1000):
      sample = df_married["Purchase"].sample(n)
      sample_mean_married.append(round(sample.mean(), 2))
      sample_std_married.append(round(sample.std(), 2))
```

```
[25] mean_sample_10000=np.mean(sample_mean_married)
     mean_sample_10000
```

```
9260.44422
```

```
[26] std_sample_10000=np.std(sample_std_married)
     std_sample_10000
```

```
33.21620389956535
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[27] (mean_sample_10000-1.645*(pop_std_married/np.sqrt(n)),mean_sample_10000+1.645*(pop_std_married/np.sqrt(n)))
```

```
(9177.916215, 9342.972225)
```

CASE2: 95%

```
[28] (mean_sample_10000-1.96*(pop_std_married/np.sqrt(n)),mean_sample_10000+1.96*(pop_std_married/np.sqrt(n)))
```

```
(9162.11298, 9358.775459999999)
```

CASE3: 99%

```
[29] (mean_sample_10000-2.575*(pop_std_married/np.sqrt(n)),mean_sample_10000+2.575*(pop_std_married/np.sqrt(n)))
```

```
(9131.259044999999, 9389.629395)
```

- For married customers, the population mean lies between 95% confidence interval.
- The above clearly shows that as the sample size increases, the mean of the sample size approaches to the population mean
- Also, As sample size increases the interval gets narrower.
- Unmarried customers buys more than married customers. Focus can be made to improve sales among married customers

SAMPLE NUMBERS	Width of the interval	UNMARRIED - CI	MARRIED - CI
500	95%	(8827.96,9709.29)	(8812.59,9692.09)
1000	95%	(8953.25,9576.45)	(8957.11,9579.01)
10000	95%	(9169.59,9366.66)	(9162.11,9358.77)

AGE GROUP:

```
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
```

AGE (0-17)

```
[3] df_age_0_17=df[df['Age']=='0-17']
```

```
[4] pop_mean_0_17 = round(df_age_0_17["Purchase"].mean(), 2)
    pop_mean_0_17
```

```
8933.46
```

```
[5] pop_std_0_17 = round(df_age_0_17["Purchase"].std(), 2)
    pop_std_0_17
```

```
5111.11
```

SAMPLES = 10000

```
[6] n = 10000
    sample_mean_0_17 = []
    sample_std_0_17 = []
    for i in range(1000):
     sample = df_age_0_17["Purchase"].sample(n)
     sample_mean_0_17.append(round(sample.mean(), 2))
     sample_std_0_17.append(round(sample.std(), 2))
```

```
[7] mean_sample_10000=np.mean(sample_mean_0_17)
    mean_sample_10000
```

```
8934.047480000001
```

```
[8] std_sample_10000=np.std(sample_std_0_17)
    std_sample_10000
```

```
19.41586094695777
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[9] (mean_sample_10000-1.645*(pop_std_0_17/np.sqrt(n)),mean_sample_10000+1.645*(pop_std_0_17/np.sqrt(n)))
```

```
(8849.969720500001, 9018.125239500001)
```

CASE2: 95%

```
[10] (mean_sample_10000-1.96*(pop_std_0_17/np.sqrt(n)),mean_sample_10000+1.96*(pop_std_0_17/np.sqrt(n)))
```

```
(8833.869724000002, 9034.225236)
```

CASE3: 99%

```
[11] (mean_sample_10000-2.575*(pop_std_0_17/np.sqrt(n)),mean_sample_10000+2.575*(pop_std_0_17/np.sqrt(n)))
```

```
(8802.436397500001, 9065.6585625)
```

AGE (18-25)

```
[13] df_age_18_25=df[df['Age']=='18-25']
```

```
[15] pop_mean_18_25 = round(df_age_18_25["Purchase"].mean(), 2)
     pop_mean_18_25
```

```
9169.66
```

```
[16] pop_std_18_25 = round(df_age_18_25["Purchase"].std(), 2)
     pop_std_18_25
```

```
5034.32
```

SAMPLES = 10000

```
[17] n = 10000
     sample_mean_18_25 = []
     sample_std_18_25 = []
     for i in range(1000):
      sample = df_age_18_25["Purchase"].sample(n)
      sample_mean_18_25.append(round(sample.mean(), 2))
      sample_std_18_25.append(round(sample.std(), 2))
```

```
[18] mean_sample_10000=np.mean(sample_mean_18_25)
     mean_sample_10000
```

```
9169.248059999998
```

```
[19] std_sample_10000=np.std(sample_std_18_25)
     std_sample_10000
```

```
30.580586815795414
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[20] (mean_sample_10000-1.645*(pop_std_18_25/np.sqrt(n)),mean_sample_10000+1.645*(pop_std_18_25/np.sqrt(n)))
```

```
(9086.433495999998, 9252.062623999998)
```

CASE2: 95%

```
[21] (mean_sample_10000-1.96*(pop_std_18_25/np.sqrt(n)),mean_sample_10000+1.96*(pop_std_18_25/np.sqrt(n)))
```

```
(9070.575387999997, 9267.920731999999)
```

CASE3: 99%

```
[22] (mean_sample_10000-2.575*(pop_std_18_25/np.sqrt(n)),mean_sample_10000+2.575*(pop_std_18_25/np.sqrt(n)))
```

```
(9039.614319999999, 9298.881799999997)
```

AGE (26-35)

```
[23] df_age_26_35=df[df['Age']=='26-35']
```

```
pop_mean_26_35 = round(df_age_26_35["Purchase"].mean(), 2)
pop_mean_26_35
```

```
9252.69
```

```
[25] pop_std_26_35 = round(df_age_26_35["Purchase"].std(), 2)
pop_std_26_35
```

```
5010.53
```

SAMPLES = 10000

```
[26] n = 10000
sample_mean_26_35 = []
sample_std_26_35 = []
for i in range(1000):
 sample = df_age_26_35["Purchase"].sample(n)
 sample_mean_26_35.append(round(sample.mean(), 2))
 sample_std_26_35.append(round(sample.std(), 2))
```

```
[27] mean_sample_10000=np.mean(sample_mean_26_35)
mean_sample_10000
```

```
9252.92228
```

```
[28] std_sample_10000=np.std(sample_std_26_35)
std_sample_10000
```

```
30.595113326214353
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[29] (mean_sample_10000-1.645*(pop_std_26_35/np.sqrt(n)),mean_sample_10000+1.645*(pop_std_26_35/np.sqrt(n)))

(9170.4990615, 9335.3454985)
```

CASE2: 95%

```
[30] (mean_sample_10000-1.96*(pop_std_26_35/np.sqrt(n)),mean_sample_10000+1.96*(pop_std_26_35/np.sqrt(n)))

(9154.715892, 9351.128668000001)
```

CASE3: 99%

```
[31] (mean_sample_10000-2.575*(pop_std_26_35/np.sqrt(n)),mean_sample_10000+2.575*(pop_std_26_35/np.sqrt(n)))

(9123.901132500001, 9381.9434275)
```

AGE (36-45)

```
[32] df_age_36_45=df[df['Age']=='36-45']
```

```
[33] pop_mean_36_45 = round(df_age_36_45["Purchase"].mean(), 2)
     pop_mean_36_45

9331.35
```

```
[34] pop_std_36_45 = round(df_age_36_45["Purchase"].std(), 2)
     pop_std_36_45

5022.92
```

SAMPLES = 10000

```
[35] n = 10000
     sample_mean_36_45 = []
     sample_std_36_45 = []
     for i in range(1000):
       sample = df_age_36_45["Purchase"].sample(n)
       sample_mean_36_45.append(round(sample.mean(), 2))
       sample_std_36_45.append(round(sample.std(), 2))
```

```
[36] mean_sample_10000=np.mean(sample_mean_36_45)
     mean_sample_10000

9331.40149
```

```
[37] std_sample_10000=np.std(sample_std_36_45)
     std_sample_10000

30.75026807367863
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[38] (mean_sample_10000-1.645*(pop_std_36_45/np.sqrt(n)),mean_sample_10000+1.645*(pop_std_36_45/np.sqrt(n)))
```

```
(9248.774456000001, 9414.028524)
```

CASE2: 95%

```
[39] (mean_sample_10000-1.96*(pop_std_36_45/np.sqrt(n)),mean_sample_10000+1.96*(pop_std_36_45/np.sqrt(n)))
```

```
(9232.952258, 9429.850722000001)
```

CASE3: 99%

```
[40] (mean_sample_10000-2.575*(pop_std_36_45/np.sqrt(n)),mean_sample_10000+2.575*(pop_std_36_45/np.sqrt(n)))
```

```
(9202.0613, 9460.741680000001)
```

AGE (46-50)

```
[41] df_age_46_50=df[df['Age']=='46-50']
```

```
[42] pop_mean_46_50 = round(df_age_46_50["Purchase"].mean(), 2)
     pop_mean_46_50
```

```
9208.63
```

```
[43] pop_std_46_50 = round(df_age_46_50["Purchase"].std(), 2)
     pop_std_46_50
```

```
4967.22
```

SAMPLES = 10000

```
[44] n = 10000
     sample_mean_46_50 = []
     sample_std_46_50 = []
     for i in range(1000):
      sample = df_age_46_50["Purchase"].sample(n)
      sample_mean_46_50.append(round(sample.mean(), 2))
      sample_std_46_50.append(round(sample.std(), 2))
```

```
[45] mean_sample_10000=np.mean(sample_mean_46_50)
     mean_sample_10000
```

```
9208.49862
```

```
[46] std_sample_10000=np.std(sample_std_46_50)
     std_sample_10000
```

```
29.182039506365896
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[47] (mean_sample_10000-1.645*(pop_std_46_50/np.sqrt(n)),mean_sample_10000+1.645*(pop_std_46_50/np.sqrt(n)))
```

```
(9126.787851000001, 9290.209389)
```

CASE2: 95%

```
[48] (mean_sample_10000-1.96*(pop_std_46_50/np.sqrt(n)),mean_sample_10000+1.96*(pop_std_46_50/np.sqrt(n)))
```

```
(9111.141108, 9305.856132)
```

CASE3: 99%

```
[49] (mean_sample_10000-2.575*(pop_std_46_50/np.sqrt(n)),mean_sample_10000+2.575*(pop_std_46_50/np.sqrt(n)))
```

```
(9080.592705000001, 9336.404535)
```

AGE (51-55)

```
[50] df_age_51_55=df[df['Age']=='51-55']
```

```
[51] pop_mean_51_55 = round(df_age_51_55["Purchase"].mean(), 2)
     pop_mean_51_55
```

```
9534.81
```

```
[52] pop_std_51_55 = round(df_age_51_55["Purchase"].std(), 2)
     pop_std_51_55
```

```
5087.37
```

SAMPLES = 10000

```
[53] n = 10000
     sample_mean_51_55 = []
     sample_std_51_55 = []
     for i in range(1000):
      sample = df_age_51_55["Purchase"].sample(n)
      sample_mean_51_55.append(round(sample.mean(), 2))
      sample_std_51_55.append(round(sample.std(), 2))
```

```
[54] mean_sample_10000=np.mean(sample_mean_51_55)
     mean_sample_10000
```

```
9534.01151
```

```
[55] std_sample_10000=np.std(sample_std_51_55)
     std_sample_10000
```

```
28.912932685549205
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[58] (mean_sample_10000-1.645*(pop_std_51_55/np.sqrt(n)),mean_sample_10000+1.645*(pop_std_51_55/np.sqrt(n)))
```

```
(9450.3242735, 9617.6987465)
```

CASE2: 95%

```
[57] (mean_sample_10000-1.96*(pop_std_51_55/np.sqrt(n)),mean_sample_10000+1.96*(pop_std_51_55/np.sqrt(n)))
```

```
(9434.299058, 9633.723962)
```

CASE3: 99%

```
[59] (mean_sample_10000-2.575*(pop_std_51_55/np.sqrt(n)),mean_sample_10000+2.575*(pop_std_51_55/np.sqrt(n)))
```

```
(9403.011732500001, 9665.0112875)
```

AGE (55+)

```
[61] df_age_55=df[df['Age']=='55+']
```

```
[62] pop_mean_55 = round(df_age_55["Purchase"].mean(), 2)
     pop_mean_55
```

```
9336.28
```

```
[63] pop_std_55 = round(df_age_55["Purchase"].std(), 2)
     pop_std_55
```

```
5011.49
```

SAMPLES = 10000

```
[64] n = 10000
     sample_mean_55 = []
     sample_std_55 = []
     for i in range(1000):
      sample = df_age_55["Purchase"].sample(n)
      sample_mean_55.append(round(sample.mean(), 2))
      sample_std_55.append(round(sample.std(), 2))
```

```
[65] mean_sample_10000=np.mean(sample_mean_55)
     mean_sample_10000
```

```
9336.790949999999
```

```
[66] std_sample_10000=np.std(sample_std_55)
     std_sample_10000
```

```
23.9293718549213
```

CONFIDENCE INTERVAL

CASE1: 90%

```
[67] (mean_sample_10000-1.645*(pop_std_55/np.sqrt(n)),mean_sample_10000+1.645*(pop_std_55/np.sqrt(n)))
```

```
(9254.351939499998, 9419.229960499999)
```

CASE2: 95%

```
[68] (mean_sample_10000-1.96*(pop_std_55/np.sqrt(n)),mean_sample_10000+1.96*(pop_std_55/np.sqrt(n)))
```

```
(9238.565745999998, 9435.016153999999)
```

CASE3: 99%

```
[69] (mean_sample_10000-2.575*(pop_std_55/np.sqrt(n)),mean_sample_10000+2.575*(pop_std_55/np.sqrt(n)))
```

```
(9207.745082499998, 9465.8368175)
```

AGE GROUP	Width of the interval	SAMPLE NUMBER	CI
0-17	95%	10,000	(8833.86,9034.22)
18-25	95%	10,000	(9070.57,9267.92)
26-35	95%	10,000	(9154.71,9351.12)
36-45	95%	10,000	(9232.95,9429.85)
51-55	95%	10,000	(9434.29,9633.72)
55+	95%	10,000	(9238.56,9435.01)

- 36-45 age group has the highest mean of purchases

- 0-17 age group has the least mean of purchases

7. **Give recommendations and action items to Walmart.**
    - The mean purchase of female is less when to compared to mean purchase of male. Various products targeting female customers can be increased. Discounts can be provided on special day like women's day and on Mother's Day to attract more female customers.
    - With respect to age, children between age group 0-17 years have lesser purchase rate compared to others. Products targeting this age group can be increased to increase the sales. Stores can be made more attractive like conducting game shows relevant to children so more kids may come to the store which may eventually increase the sales. School items should be added to make this group more attractive to purchase.
    - The potential for buyers with mean less than 2000 should be explored. More incentive schemes/discount should be offered to this group. Maybe they are from low-income group.
    - City Category C has highest mean purchase value. A detailed study can be made the reasons for having better purchase rate at city category C when compared to other cities.

```
df.groupby('City_Category')[['Purchase']].mean()
```

City_Category	Purchase
A	8911.939216
B	9151.300563
C	9719.920993

    - Married customers are relatively less in number when compared to unmarried customers. Couple related products can be introduced which can increase the number of married customers.