WALLMART CASE STUDY

In [308…

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import binom,geom,norm,chi2,chisquare,chi2_contingency,ttest_1samp,t
import math
import scipy.stats as stats
import statistics
```

In [5]:

```
df= pd.read_csv("C:\\Users\\mayank.khanduja\\Desktop\\wallmart.csv")
df
```

Out[5]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	M
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+	
...	...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	1	
550064	1006035	P00375436	F	26-35	1	C	3	
550065	1006036	P00375436	F	26-35	15	B	4+	
550066	1006038	P00375436	F	55+	1	C	2	
550067	1006039	P00371644	F	46-50	0	B	4+	

550068 rows × 10 columns

In [6]:

```
df.info()
# there are no null values in the dataset
```

```
<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 [7]:

```
df.describe()
# 40% of the sample population is married
# Median purchase amount is 8047
```

Out[7]:

	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

In [8]:

```
df.describe(include="object")
# Majority of sample belongs to 26-35 age group which shows high shopping capacity und
```

Out[8]:

	Product_ID	Gender	Age	City_Category	Stay_In_Current_City_Years
count	550068	550068	550068	550068	550068
unique	3631	2	7	3	5
top	P00265242	M	26-35	B	1
freq	1880	414259	219587	231173	193821

In [21]:

```
# identifying unique values in some important dataset
col= ['Gender','Age','City_Category','Stay_In_Current_City_Years','Marital_Status','Pr
for i in col:
    print("unique values in",i,":",df[i].unique())
```

```
unique values in Gender : ['F' 'M']
unique values in Age : ['0-17' '55+' '26-35' '46-50' '51-55' '36-45' '18-25']
unique values in City_Category : ['A' 'C' 'B']
unique values in Stay_In_Current_City_Years : ['2' '4+' '3' '1' '0']
unique values in Marital_Status : [0 1]
unique values in Product_Category : [ 3  1 12  8  5  4  2  6 14 11 13 15  7 16 18 10
17  9 20 19]
unique values in Occupation : [10 16 15  7 20  9  1 12 17  0  3  4 11  8 19  2 18  5
14 13  6]
```

In [25]:
```python
# range
col= ["Product_Category","Occupation","Age"]
for i in col:
    print("range of",i,":",df[i].min(),"-",df[i].max())
```

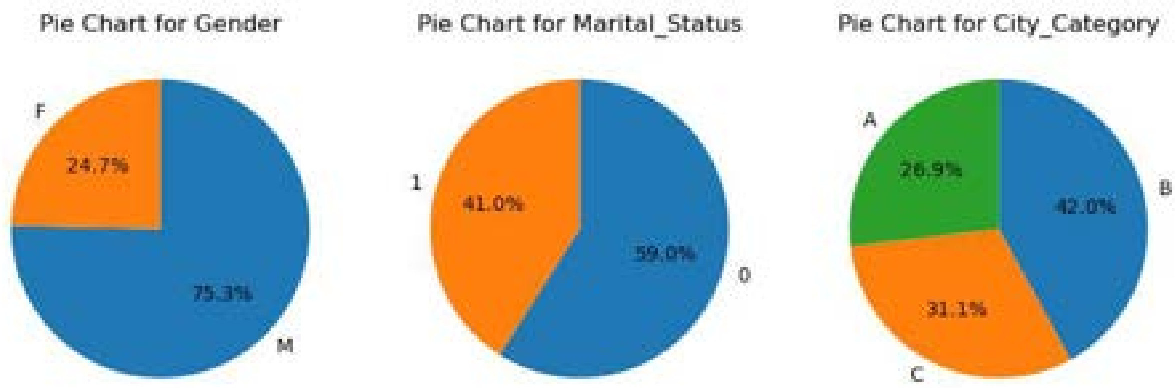
```
range of Product_Category : 1 - 20
range of Occupation : 0 - 20
range of Age : 0-17 - 55+
```

Univariate Analysis

SUMMARY

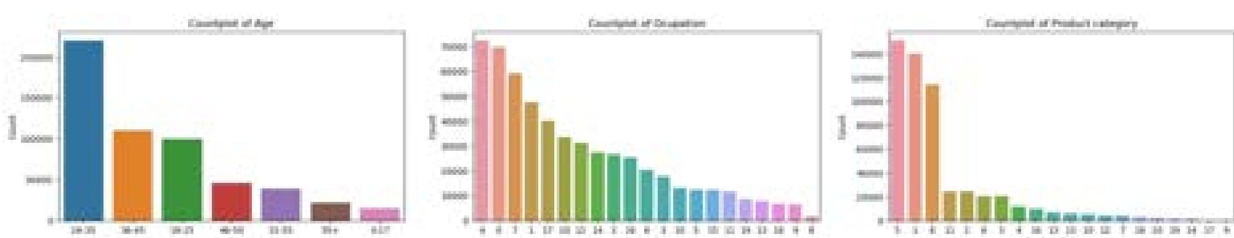
1. Gender - 75.3% of the population are male
2. Martial Status - 59% of the population are unmarried
3. City Category - Majority of buyers belong to category B
4. Age - Majority of buyers belong to 26-35 years of age 5.Occupation - Majority of buyers belong to 4,0,7 occupation category
5. Product Category - 5,1 and 8 are the highest selling product categories

In [60]:
```python
plt.figure(figsize=(9, 3))
#Gender
plt.subplot(131)
value_counts= df["Gender"].value_counts()
plt.pie(value_counts, labels=value_counts.index,autopct='%1.1f%%',startangle=90, count
plt.title('Pie Chart for Gender')

#Marital_Status
plt.subplot(132)
value_counts= df["Marital_Status"].value_counts()
plt.pie(value_counts, labels=value_counts.index,autopct='%1.1f%%',startangle=90, count
plt.title('Pie Chart for Marital_Status')

#City_Category
plt.subplot(133)
value_counts= df["City_Category"].value_counts()
plt.pie(value_counts, labels=value_counts.index,autopct='%1.1f%%',startangle=90, count
plt.title('Pie Chart for City_Category')

plt.tight_layout()
plt.show()
```



In [83]:
```
plt.figure(figsize=(25, 4))
#Age
plt.subplot(131)
sns.countplot(data= df,x="Age",order=df["Age"].value_counts().index)
plt.title("Countplot of Age")
plt.xlabel("Age")
plt.ylabel("Count")
# Occupation
plt.subplot(132)
sns.countplot(data= df,x="Occupation",order=df["Occupation"].value_counts().index)
plt.title("Countplot of Ocupation")
plt.xlabel("Occupation")
plt.ylabel("Count")
# Product Category
plt.subplot(133)
sns.countplot(data= df,x="Product_Category",order=df["Product_Category"].value_counts(
plt.title("Countplot of Product category")
plt.xlabel("Product Category")
plt.ylabel("Count")
```

Out[83]:
```
Text(0, 0.5, 'Count')
```



Bivariate Analysis

SUMMARY

---------------------------------MALE BUYERS--

1. The age group of 26-35 has the highest number of male buyers.
2. Males predominantly belong to occupation categories 0, 4, and 7.
3. The majority of male buyers prefer products from categories 1, 5, and 8.
4. Most male buyers are from "B" city category.

---------------------------------FEMALE BUYERS--

1. The age group of 26-35 has the highest number of female buyers.
2. Females predominantly belong to occupation categories 0, 4, and 1.
3. The majority of female buyers prefer products from categories 1, 5, and 8.
4. Most female buyers are from "B" city category.

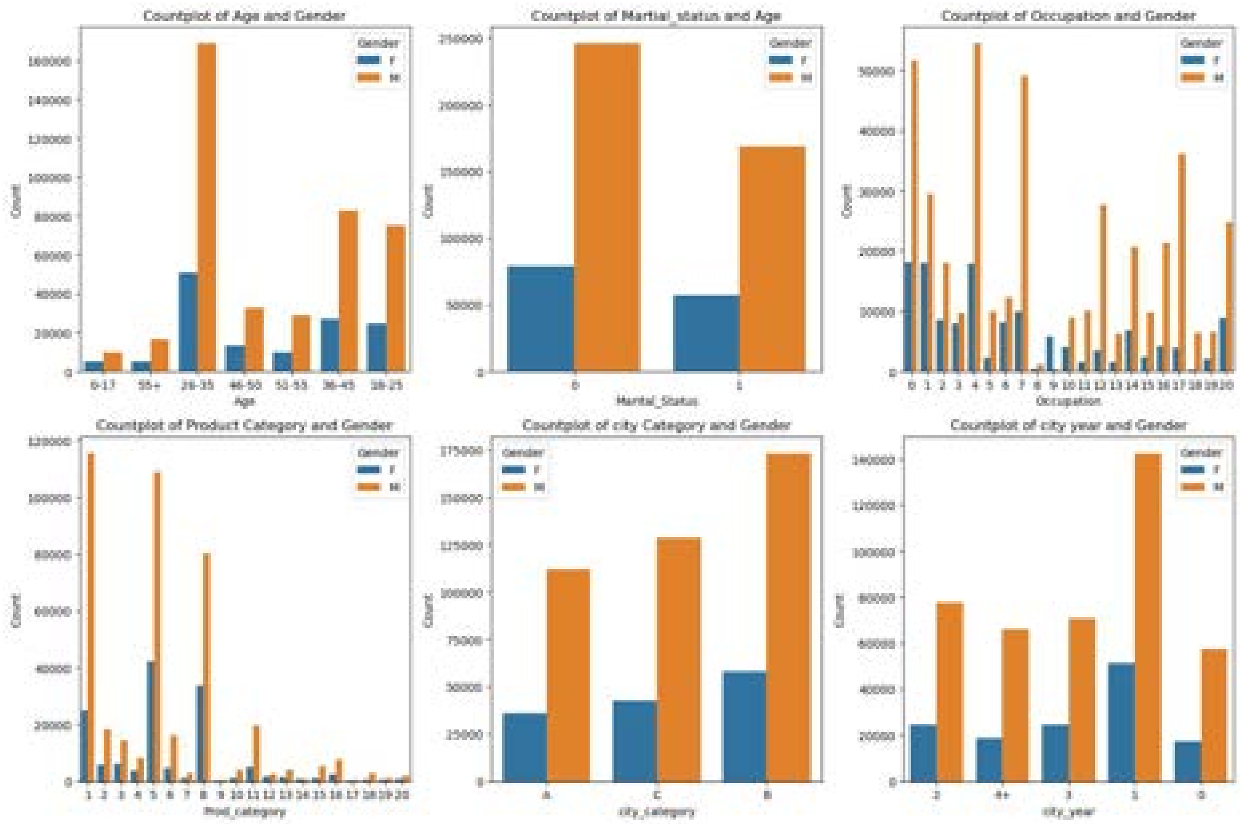
---------------------- COMMON INFERENCE---

1. Majority of Buyers are from 4,0,7,1 occupation categories
2. Product category 1,5 and 8 are the highest selling products
3. A significant portion of the population has a one-year stay in their current city.
4. Majority of Buyers are from "B" city category.
5. Males have a mean of 9437.52 while female have mean value of 9265.91

In [97]:

```
plt.figure(figsize=(15, 10))
#Gender and Age
plt.subplot(2,3,1)
sns.countplot(data= df,x="Age",hue="Gender")
plt.title("Countplot of Age and Gender")
plt.xlabel("Age")
plt.ylabel("Count")

#Gender and Martial_Status
plt.subplot(2,3,2)
sns.countplot(data= df,x="Marital_Status",hue="Gender")
plt.title("Countplot of Martial_status and Age")
plt.xlabel("Marital_Status")
plt.ylabel("Count")

#Gender and Occupation
plt.subplot(2,3,3)
sns.countplot(data= df,x="Occupation",hue="Gender")
plt.title("Countplot of Occupation and Gender")
plt.xlabel("Occupation")
plt.ylabel("Count")

#Gender and Product_Category
plt.subplot(2,3,4)
sns.countplot(data= df,x="Product_Category",hue="Gender")
plt.title("Countplot of Product Category and Gender")
plt.xlabel("Prod_category")
plt.ylabel("Count")

#Gender and City_Category
plt.subplot(2,3,5)
sns.countplot(data= df,x="City_Category",hue="Gender")
plt.title("Countplot of city Category and Gender")
plt.xlabel("city_category")
plt.ylabel("Count")

#Gender and Stay_In_Current_City_Years
plt.subplot(2,3,6)
sns.countplot(data= df,x="Stay_In_Current_City_Years",hue="Gender")
plt.title("Countplot of city year and Gender")
plt.xlabel("city_year")
plt.ylabel("Count")
```

```python
plt.tight_layout()
plt.show()
```



In [119…

```python
# Purchase and Gender
df.groupby(["Gender"])["Purchase"].describe()
```

Out[119]:

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

In [120…

```python
# Purchase and Marital_status
df.groupby(["Marital_Status"])["Purchase"].describe()
```

Out[120]:

Marital_Status	count	mean	std	min	25%	50%	75%	max
0	324731.0	9265.907619	5027.347859	12.0	5605.0	8044.0	12061.0	23961.0
1	225337.0	9261.174574	5016.897378	12.0	5843.0	8051.0	12042.0	23961.0

In [139…

```python
# Purchase and Occupation
df2= pd.DataFrame(df.groupby(["Occupation"])["Purchase"].sum())
df2.sort_values(by="Purchase",ascending=False).head(5)
```

Out[139]:

	Purchase
Occupation	
4	666244484
0	635406958
7	557371587
1	424614144
17	393281453

In [125…

```
# purchase and city category
df.groupby(["City_Category"])["Purchase"].describe()
```

Out[125]:

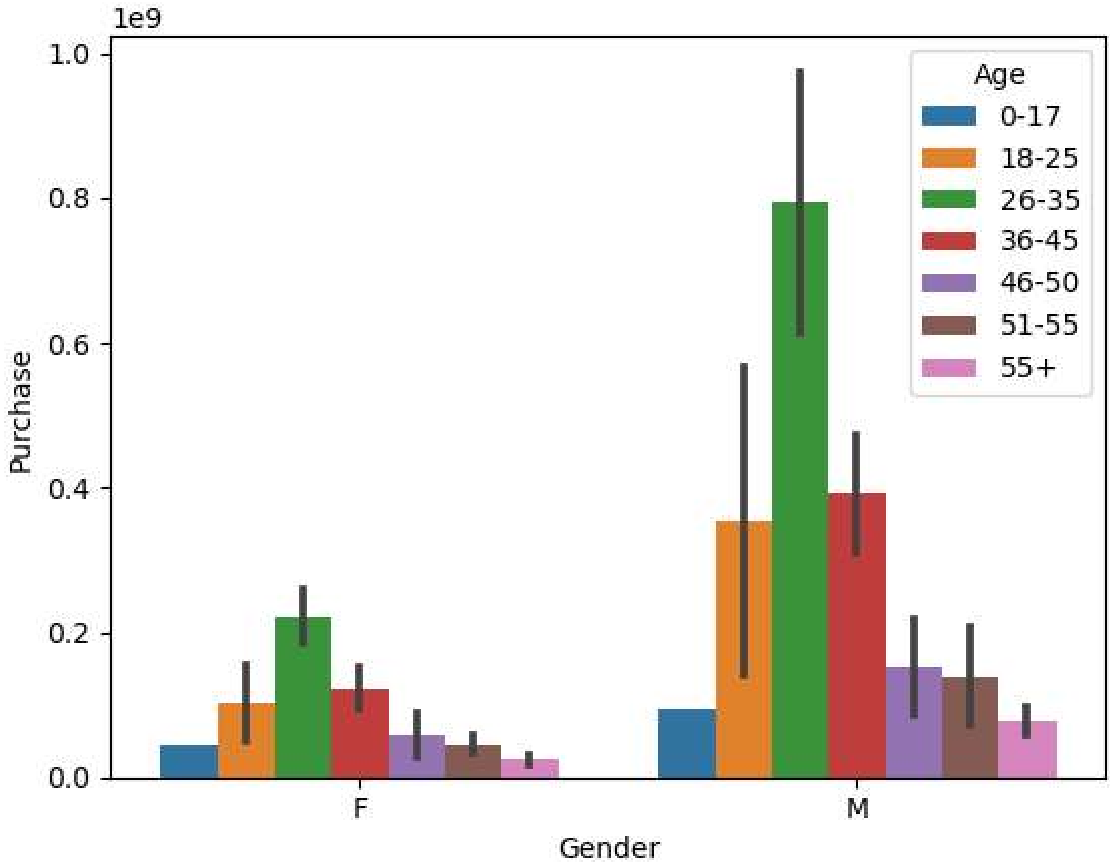
	count	mean	std	min	25%	50%	75%	max
City_Category								
A	147720.0	8911.939216	4892.115238	12.0	5403.0	7931.0	11786.0	23961.0
B	231173.0	9151.300563	4955.496566	12.0	5460.0	8005.0	11986.0	23960.0
C	171175.0	9719.920993	5189.465121	12.0	6031.5	8585.0	13197.0	23961.0

In [168…

```
#Gender and Age and marital_status
grouped_data = df.groupby(['Gender', 'Marital_Status', 'Age'])['Purchase'].sum().reset
sns.barplot(data=grouped_data, x='Gender', y='Purchase', hue='Age')
```

Out[168]:

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



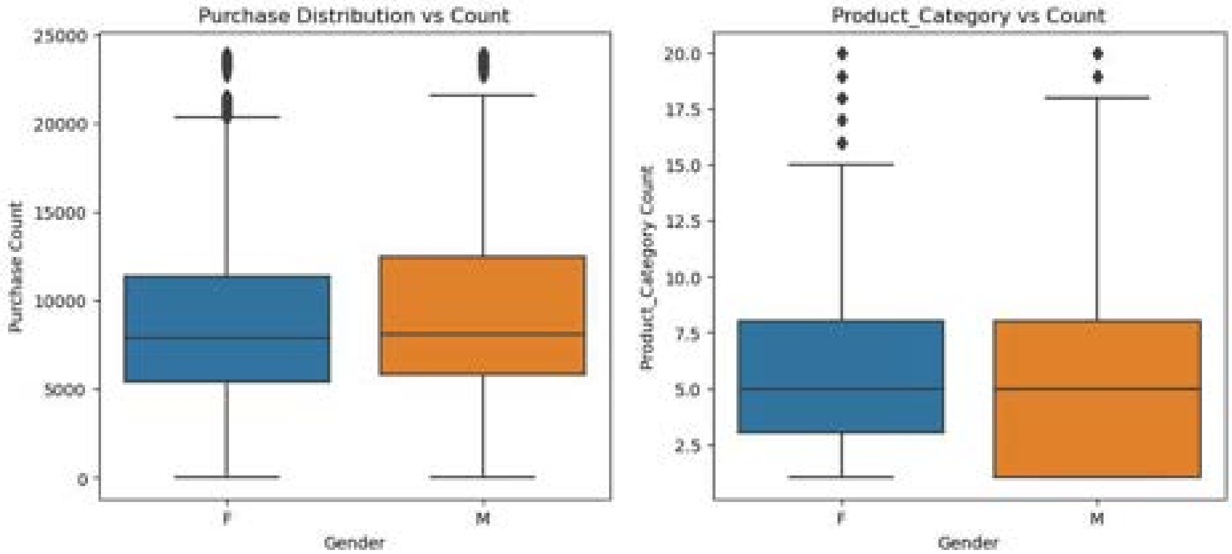
Outliers

SUMMARY

1. The Median purchase for both male and female is similar
2. However, the purchase transactions for males is more than females, hence having higher upper and lower limit.
3. Females have more outliers than males buyers
4. The Median product category for both male and female is product caegory 5
5. Females have more outliers than males buyers in product category

In [180…

```
plt.figure(figsize=(12, 5))
#purchase and Gender
plt.subplot(121)
sns.boxplot(data=df, y="Purchase",x='Gender')
plt.title("Purchase Distribution vs Count")
plt.ylabel('Purchase Count')

# purchase and product category
plt.subplot(122)
sns.boxplot(data=df, y="Product_Category",orient='v',x='Gender')
plt.title("Product_Category vs Count")
plt.ylabel('Product_Category Count')

plt.show()
```



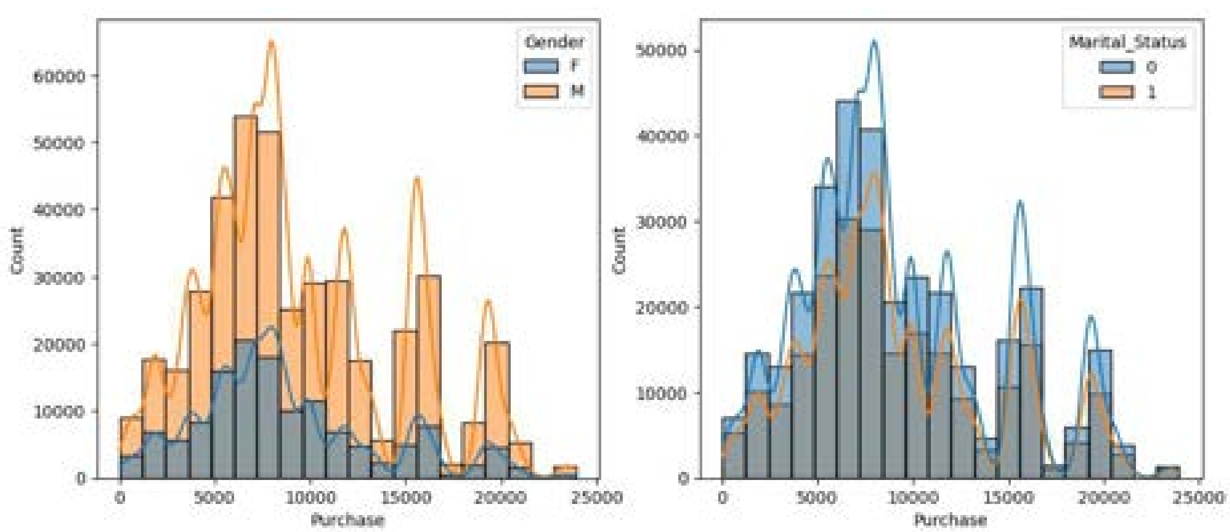
Distribution

SUMMARY

1. Most of the items bought are in the range of 5000-8000
2. Unmarried have purchased mroe items than married in each range

In [183…

```
plt.figure(figsize=(12, 5))
# Purchase and Gender
plt.subplot(121)
sns.histplot(data = df, x='Purchase',bins=20, hue='Gender', kde=True)

# Purchase and Marital_status
plt.subplot(122)
sns.histplot(data = df, x='Purchase',bins=20, hue='Marital_Status', kde=True)

plt.show()
```



Interval Calculation

As per CLT theorm my sample data will follow Normal Distribution with mean = sample mean and standard deviation = population standard deviation/ sqrt(n) Since population standard deviation is unknown we will calculate sample standard deviation using Bassel Correction

---------SUMMARY-----------

Considering All samples for 95% CI CI for Male = 9422.01944736257, 9453.032633581959 CI for Female = 8709.21154714068, 8759.919983170272

Considering mean of 1000 samples of sample size of 500 each we observed

----------------> For MALE

Mean_of_sample_male: 9429.04958 Standard_Deviation_of_sample_male: 236.71522352590785 CI for sample: (9411.636786744646, 9446.462373255355)

----------------> For FEMALE

Mean_of_sample_female: 8738.517 Standard_Deviation_of_sample_female: 215.6922318250281 CI for sample: (8722.65065991586, 8754.38334008414)

In [191...

```python
# Creating Groups for Male and Female
df_f = df.loc[df["Gender"]=="F"]
df_m = df.loc[df["Gender"]=="M"]
```

In [207...

```python
# Calculation of standard Deviation using Bassel Correction for sample
std_f = df_f["Purchase"].std(ddof=1)
std_m = df_m["Purchase"].std(ddof=1)
print("standard deviation for Females:",std_f)
print("standard deviation for Males:",std_m)

# Calculation of Mean for sample
mean_f = df_f["Purchase"].mean()
mean_m = df_m["Purchase"].mean()
print("Mean value for Females:",mean_f)
print("Mean value for males:",mean_m)
```

```
standard deviation for Females: 4767.233289291444
standard deviation for Males: 5092.186209777949
Mean value for Females: 8734.565765155476
Mean value for males: 9437.526040472265
```

In [202...

```python
# CI for Male
norm.interval(0.95,loc=mean_m,scale=std_m/(math.sqrt(df_m.shape[0])))
```

Out[202]:

```
(9422.01944736257, 9453.032633581959)
```

In [203...

```python
# CI for Female
norm.interval(0.95,loc=mean_f,scale=std_f/(math.sqrt(df_f.shape[0])))
```

Out[203]:

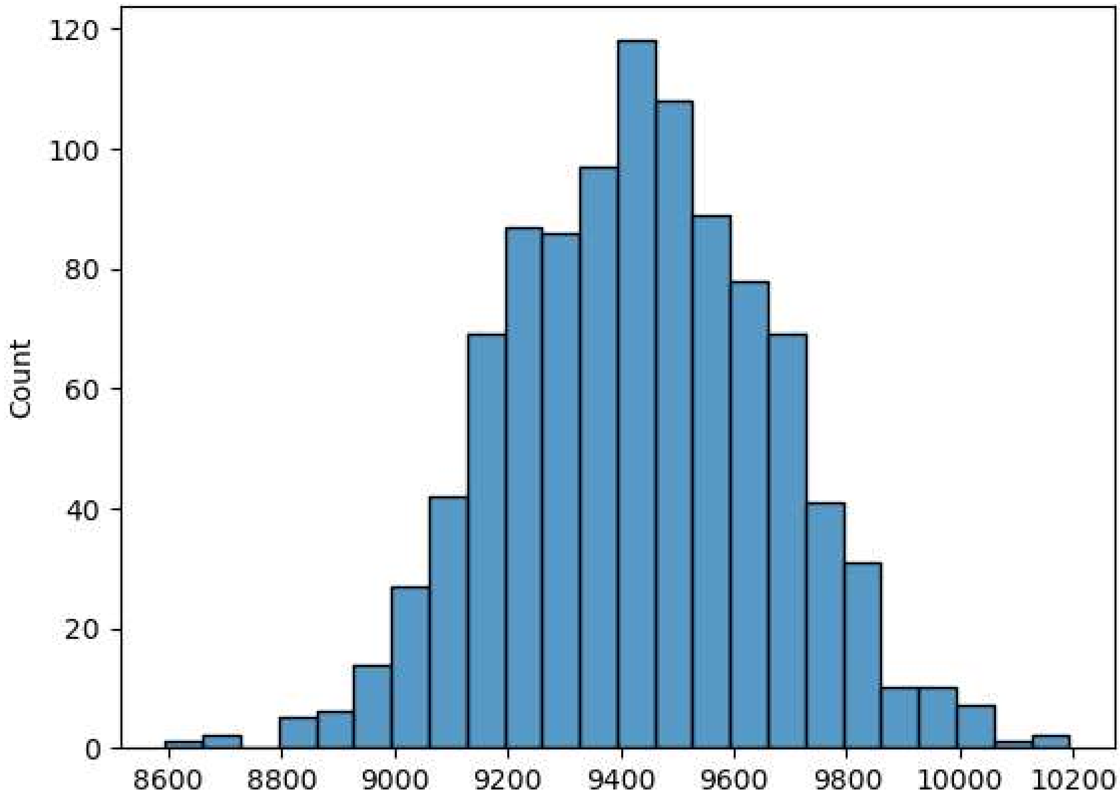
```
(8709.21154714068, 8759.919983170272)
```

In [205...

```python
# Taking 1000 samples each of sample size 500 to see the difference in the Mean values
sample1= [np.mean(df_m["Purchase"].sample(500)) for i in range(1000)]
```

```
sns.histplot(sample1)
plt.show()
```

Out[205]: <Axes: ylabel='Count'>



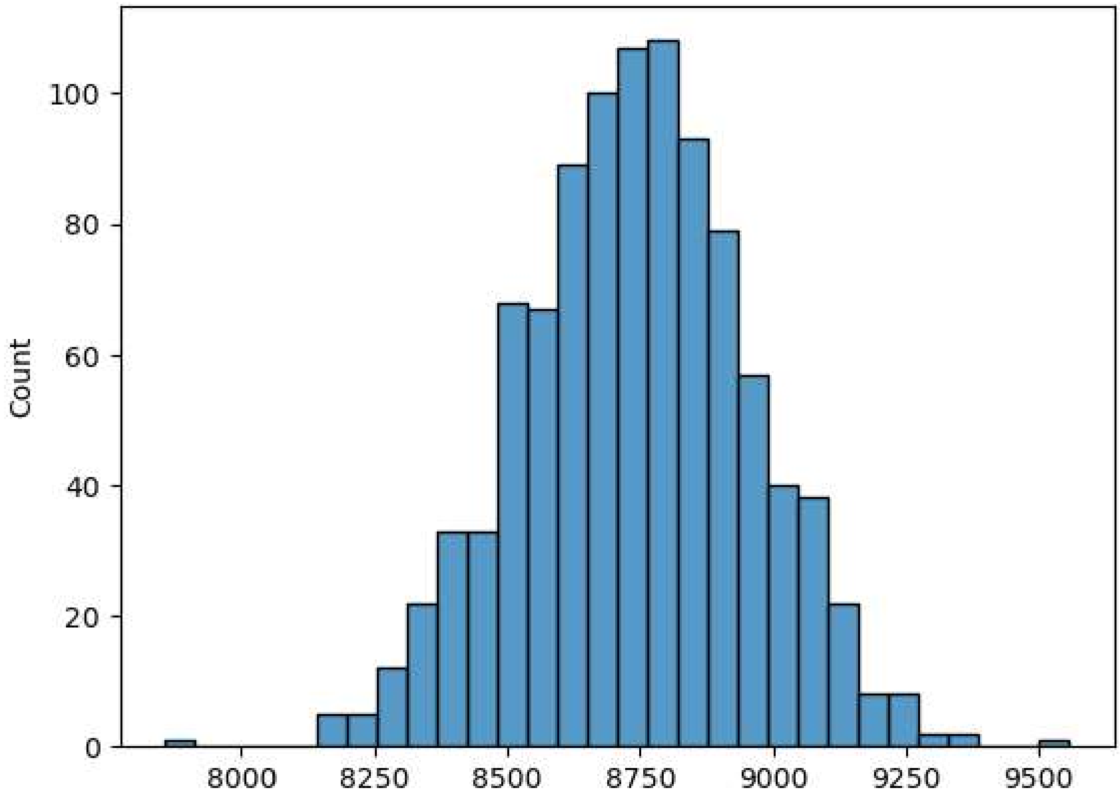
In [217…

```
Mean_of_sample = np.mean(sample1)
Standard_Deviation_of_sample = statistics.stdev(sample1)
CI = Mean_of_sample + (norm.ppf(0.05)*Standard_Deviation_of_sample/math.sqrt(500)),Mea
print("Mean_of_sample_male:",Mean_of_sample)
print("Standard_Deviation_of_sample_male:",Standard_Deviation_of_sample)
print("CI for sample:",CI)
```

```
Mean_of_sample_male: 9429.04958
Standard_Deviation_of_sample_male: 236.71522352590785
CI for sample: (9411.636786744646, 9446.462373255355)
```

In [218…

```
# Taking 1000 samples each of sample size 500 to see the difference in the Mean values
sample2= [np.mean(df_f["Purchase"].sample(500)) for i in range(1000)]
sns.histplot(sample2)
plt.show()
```



In [219…

```
Mean_of_sample = np.mean(sample2)
Standard_Deviation_of_sample = statistics.stdev(sample2)
CI = Mean_of_sample + (norm.ppf(0.05)*Standard_Deviation_of_sample/math.sqrt(500)),Mea
print("Mean_of_sample_female:",Mean_of_sample)
print("Standard_Deviation_of_sample_female:",Standard_Deviation_of_sample)
print("CI for sample:",CI)
```

```
Mean_of_sample_female: 8738.517
Standard_Deviation_of_sample_female: 215.6922318250281
CI for sample: (8722.65065991586, 8754.38334008414)
```

CLT Analysis for average Male and Female spend function

In [243…

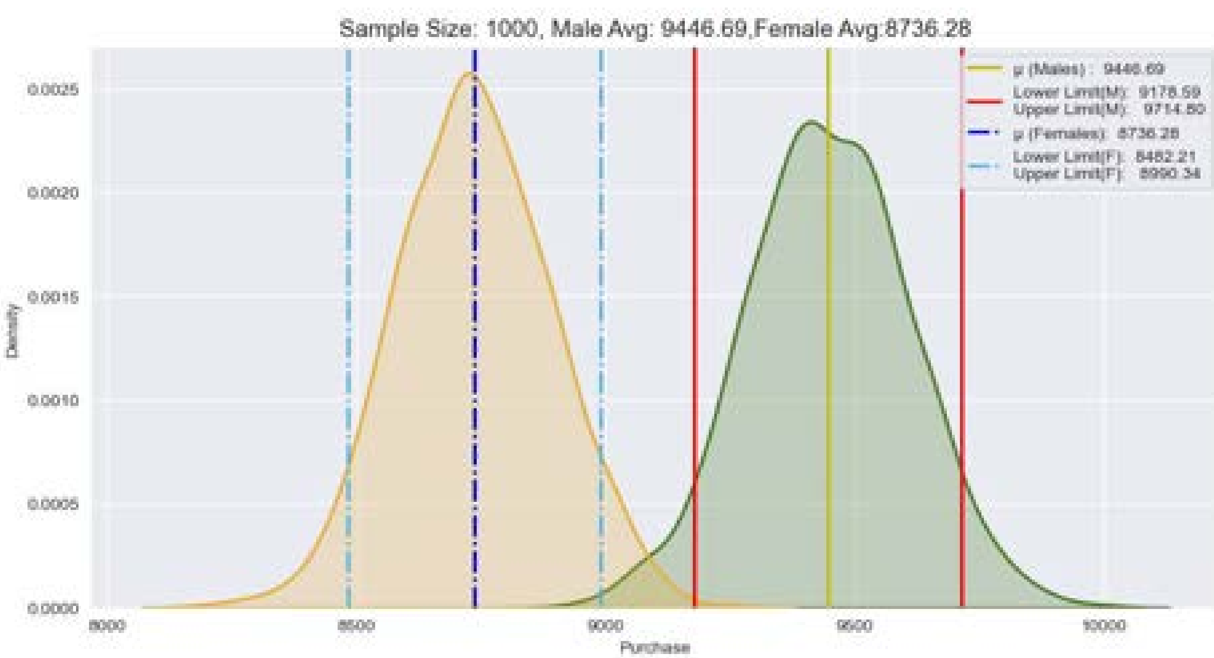
```
def meanPurchase(malesample,femalesample,size,ci,iteration=1000):
    #confidence Interval
    ci = ci/100

    #Sample means
    sampleM = [np.mean(malesample.sample(size)) for i in range(iteration)]
    sampleF = [np.mean(femalesample.sample(size)) for i in range(iteration)]

    #mean,std for sample of males
    meanM = np.mean(sampleM)
    sigmaM = np.std(sampleM,ddof=1)

    #ci for males
    lowerM= meanM + norm.ppf((1-ci)/2)*sigmaM
    upperM = meanM + norm.ppf(ci+ (1-ci)/2)*sigmaM

    #mean,std for sample of females
```

```
    meanF = np.mean(sampleF)
    sigmaF = np.std(sampleF,ddof=1)

    #ci for females
    lowerF= meanF + norm.ppf((1-ci)/2)*sigmaF
    upperF = meanF + norm.ppf(ci+ (1-ci)/2)*sigmaF

    #graph
    fig, ax = plt.subplots(figsize=(12,6))
    sns.set_style("darkgrid")
    sns.kdeplot(data = sampleM, color="#467821", fill = True, linewidth = 2)
    sns.kdeplot(data = sampleF, color='#e5ae38', fill = True, linewidth = 2)

    label_mean1=("μ (Males) :  {:.2f}".format(meanM))
    label_ult1=("Lower Limit(M):  {:.2f}\nUpper Limit(M):   {:.2f}".format(lowerM,uppe
    label_mean2=("μ (Females):  {:.2f}".format(meanF))
    label_ult2=("Lower Limit(F):  {:.2f}\nUpper Limit(F):   {:.2f}".format(lowerF,uppe

    plt.title(f"Sample Size: {size}, Male Avg: {np.round(meanM, 2)},Female Avg:{np.rou
              fontsize=14,family = "sans-serif")
    plt.xlabel('Purchase')
    plt.axvline(meanM, color = 'y', linestyle = 'solid', linewidth = 2,label=label_mea
    plt.axvline(upperM, color = 'r', linestyle = 'solid', linewidth = 2,label=label_ul
    plt.axvline(lowerM, color = 'r', linestyle = 'solid', linewidth = 2)
    plt.axvline(meanF, color = 'b', linestyle = 'dashdot', linewidth = 2,label=label_m
    plt.axvline(upperF, color = '#56B4E9', linestyle = 'dashdot', linewidth = 2,label=
    plt.axvline(lowerF, color = '#56B4E9', linestyle = 'dashdot', linewidth = 2)
    plt.legend(loc='upper right')

    return None
```

90% CI

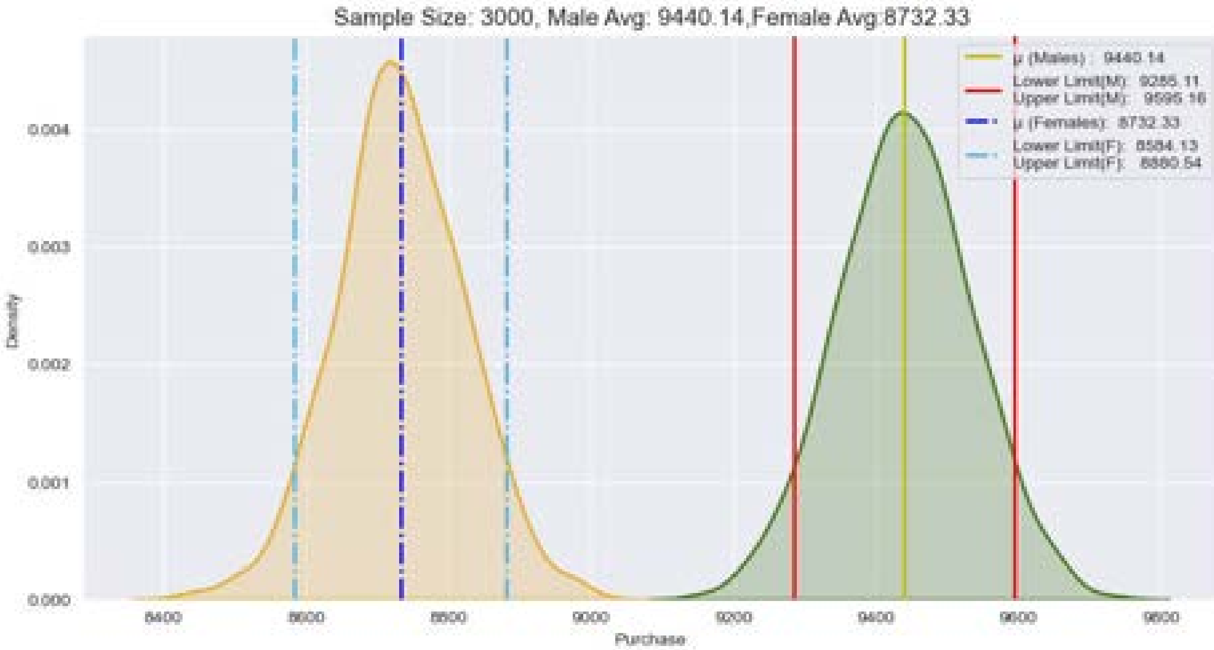
In [245…

```
# 1000 SAMPLES
meanPurchase(df_m["Purchase"],df_f["Purchase"],1000,90)
```



In [253...

```
# 3000 SAMPLES
meanPurchase(df_m["Purchase"],df_f["Purchase"],3000,90)
```

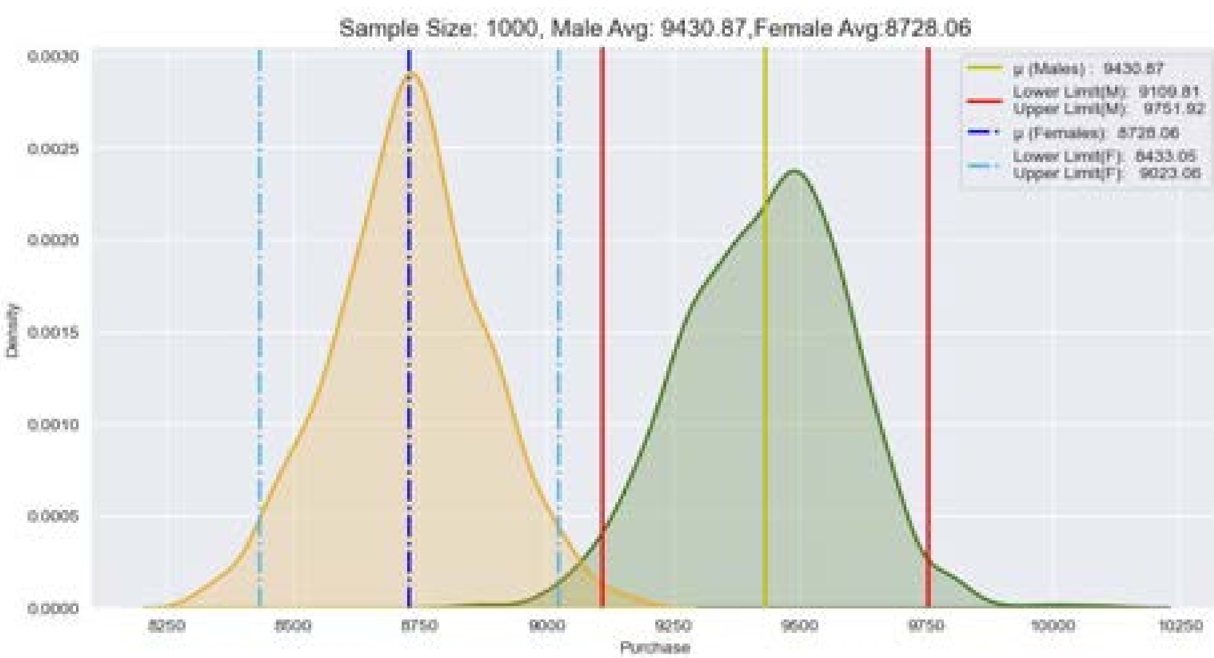


Considering 90% CI, as we enlarged the sample size, the distinction between the two graphs became more pronounced. When the sample size reached 2500, both graphs showed clear distinctions with CI of (9270.90-9270.90) in Males and (8576-8889) in Females

95% CI

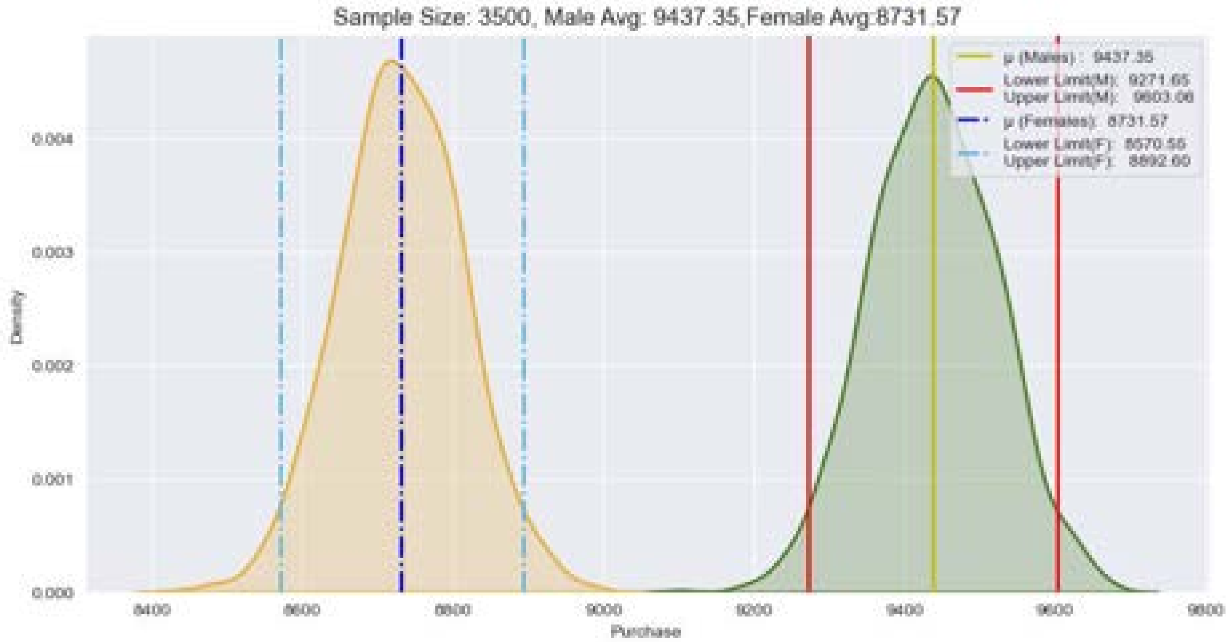
In [247...

```
# sample of 1000
meanPurchase(df_m["Purchase"],df_f["Purchase"],1000,95)
```



In [254...

```
meanPurchase(df_m["Purchase"],df_f["Purchase"],3500,95)
```

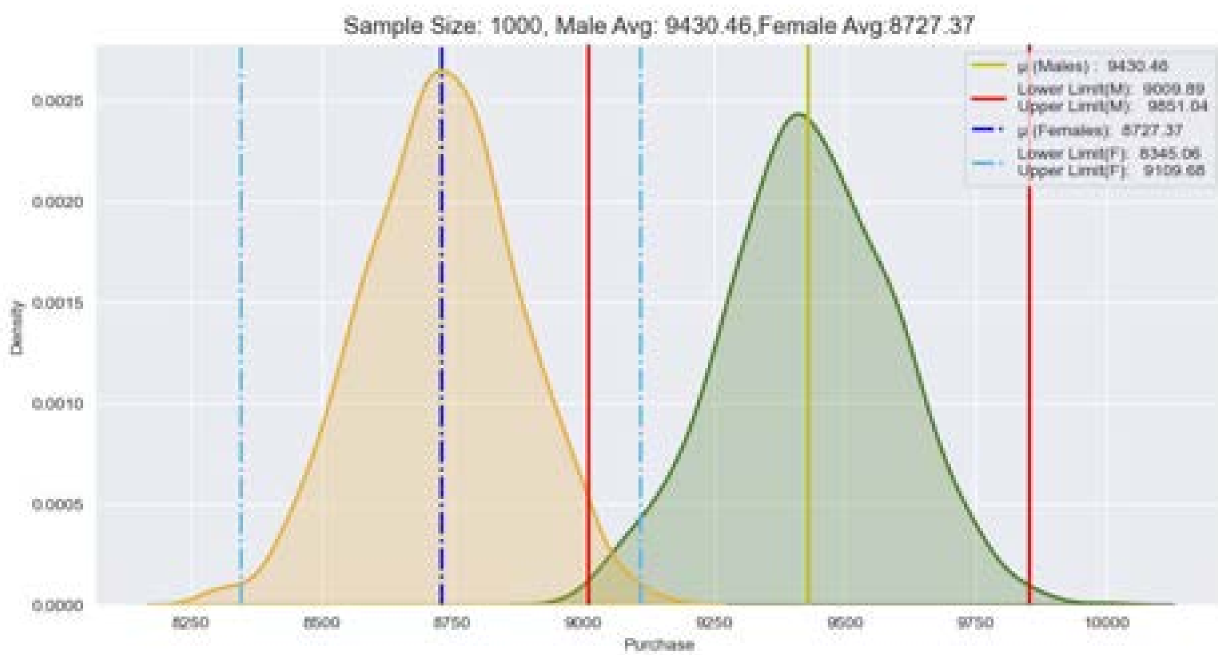


Considering 95% CI, as we enlarged the sample size, the distinction between the two graphs became more pronounced. When the sample size reached 3500, both graphs showed clear distinctions with CI of (9250.76-9624.42) in Males and (8566-8898) in Females

99% CI

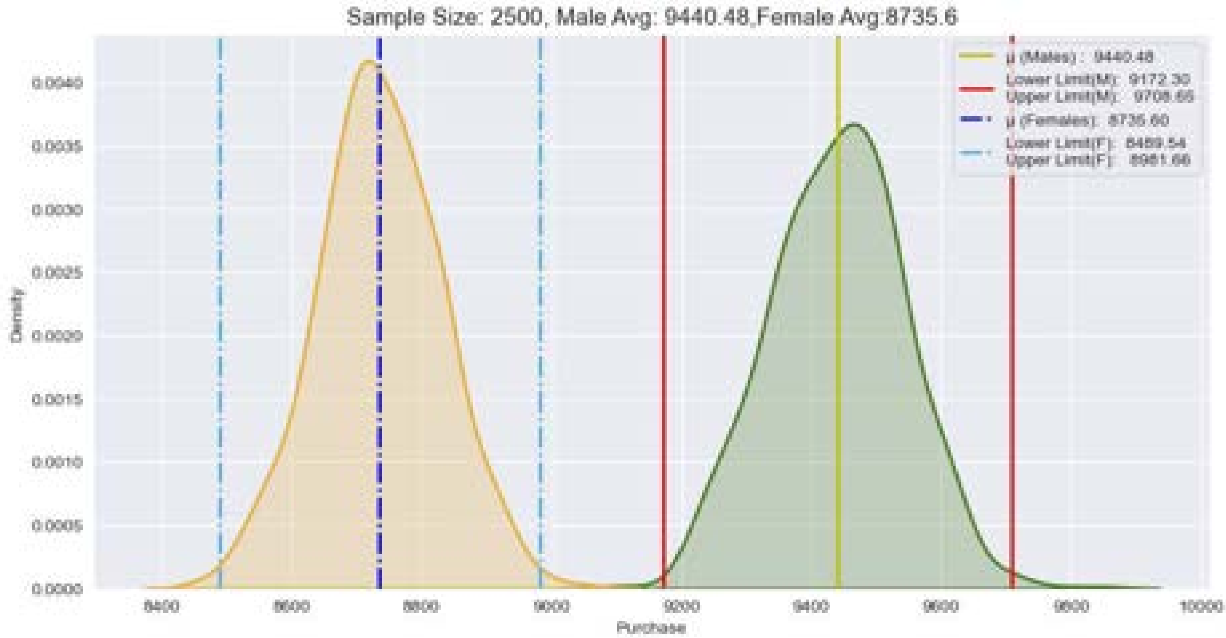
In [250…

```
meanPurchase(df_m["Purchase"],df_f["Purchase"],1000,99)
```



In [251…

```
meanPurchase(df_m["Purchase"],df_f["Purchase"],2500,99)
```



Considering 99% CI, as we enlarged the sample size, the distinction between the two graphs became more pronounced. When the sample size reached 2500, both graphs showed clear distinctions with CI of (9172-9708) in Males and (8489-8981) in Females

INFERENCE

1. Considering 90% CI, as we enlarged the sample size, the distinction between the two graphs became more pronounced. When the sample size reached 2500, both graphs showed clear distinctions with CI of (9270.90-9270.90) in Males and (8576-8889) in Females

2. Considering 95% CI, as we enlarged the sample size, the distinction between the two graphs became more pronounced. When the sample size reached 3500, both graphs showed clear distinctions with CI of (9250.76-9624.42) in Males and (8566-8898) in Females

3. Considering 99% CI, as we enlarged the sample size, the distinction between the two graphs became more pronounced. When the sample size reached 2500, both graphs showed clear distinctions with CI of (9172-9708) in Males and (8489-8981) in Females

CLT Analysis for average spend for Married and Unmarried Buyers

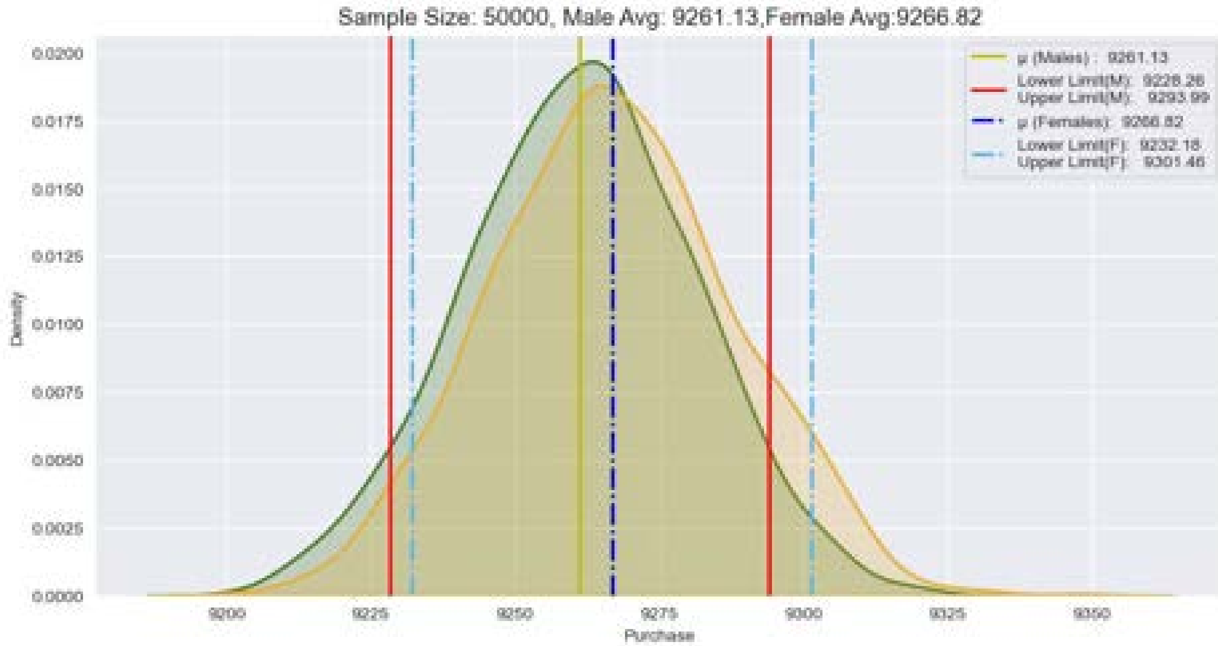
90% CI

In [268…

```
df_married = df.loc[df["Marital_Status"]==1]
df_unmarried = df.loc[df["Marital_Status"]==0]
```

In [270…

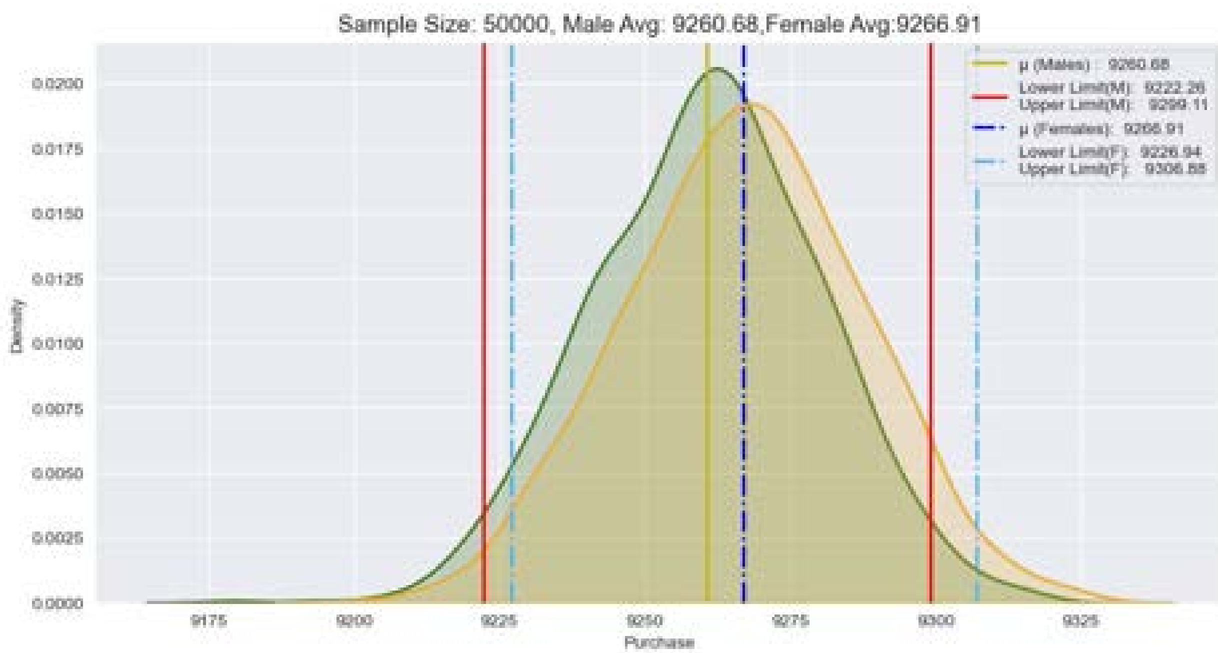
```
meanPurchase(df_married["Purchase"],df_unmarried["Purchase"],50000,90)
```



95% CI

In [271...

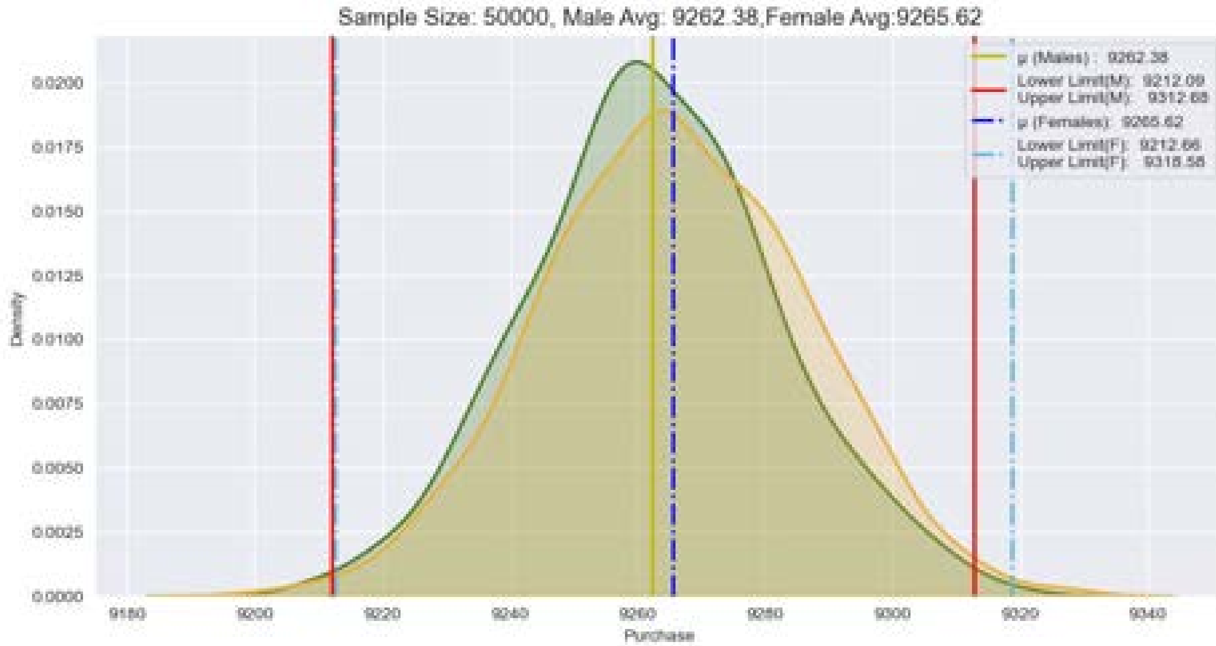
```
meanPurchase(df_married["Purchase"],df_unmarried["Purchase"],50000,95)
```



99% CI

In [272...

```
meanPurchase(df_married["Purchase"],df_unmarried["Purchase"],50000,99)
```



INFERENCE

Overlapping is evident despite increasing the sample size from 1000 to 50,000 for married vs single customer spend ,which indicates that customers spend the same regardless of whether they are single or married.

For 90% CI for sample size of 50k the range lies 9228-9293 for Married and 9232-9301 for Unmarried users.

For 95% CI for sample size of 50k the range lies 9222-9299 for Married and 9226-9306 for Unmarried users.

For 99% CI for sample size of 50k the range lies 9212-9312 for Married and 9212-9318 for Unmarried users.

CLT Analysis for average spend on the basis of Age Group

In [295…

```
# Whole Data
Age =['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
result_df = pd.DataFrame(columns=['Age', 'Mean', 'CI', 'Lower_Limit', 'Upper_Limit'])
for i in Age:
    mean = np.mean(df[df["Age"]== i]["Purchase"])
    ci = 95
    alpha = (1 - (ci/100))/2
    z = stats.norm.ppf(1 - alpha)
    sigma = np.std(df[df["Age"]== i]["Purchase"])
    n = len(df[df["Age"] == i])
    standard_error = sigma / np.sqrt(n)
    margin_of_error = z * standard_error
    lower = mean - margin_of_error
    upper = mean + margin_of_error
    result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': low
```

```
result_df
```

```
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3430750186.py:14: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lower, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3430750186.py:14: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lower, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3430750186.py:14: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lower, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3430750186.py:14: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lower, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3430750186.py:14: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lower, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3430750186.py:14: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lower, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3430750186.py:14: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lower, 'Upper_Limit': upper}, ignore_index=True)
```

Out[295]:

	Age	Mean	CI	Lower_Limit	Upper_Limit
0	0-17	8933.464640	95	8851.950669	9014.978611
1	18-25	9169.663606	95	9138.408106	9200.919107
2	26-35	9252.690633	95	9231.733724	9273.647542
3	36-45	9331.350695	95	9301.669546	9361.031844
4	46-50	9208.625697	95	9163.085641	9254.165754
5	51-55	9534.808031	95	9483.992133	9585.623929
6	55+	9336.280459	95	9269.300392	9403.260527

In [304…

```
def ci_cal(sample_size,ci):
    Age = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
    result_df = pd.DataFrame(columns=['Age', 'Mean', 'CI', 'Lower_Limit', 'Upper_Limit
    sample_size = sample_size
```

```python
    for i in Age:
        sample_data = df[df["Age"] == i]["Purchase"].sample(sample_size)
        mean = sample_data.mean()
        ci = ci
        alpha = (1 - (ci/100))/2
        z = stats.norm.ppf(1 - alpha)
        sigma = sample_data.std()
        n = len(sample_data)
        standard_error = sigma / np.sqrt(n)
        margin_of_error = z * standard_error
        lower = mean - margin_of_error
        upper = mean + margin_of_error

        result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit':

    return result_df
```

90% CI

In [305…

```python
ci_cal(1000,90)
```

```
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
```

Out[305]:

	Age	Mean	CI	Lower_Limit	Upper_Limit
0	0-17	8825.696	90	8559.791214	9091.600786
1	18-25	9249.920	90	8991.221154	9508.618846
2	26-35	9053.529	90	8786.132115	9320.925885
3	36-45	9429.312	90	9165.910675	9692.713325
4	46-50	9196.532	90	8941.624868	9451.439132
5	51-55	9435.737	90	9167.707154	9703.766846
6	55+	9136.189	90	8880.770773	9391.607227

95% CI

In [306…

```
ci_cal(1000,95)
```

```
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
```

Out[306]:

	Age	Mean	CI	Lower_Limit	Upper_Limit
0	0-17	8779.243	95	8465.572948	9092.913052
1	18-25	9138.322	95	8828.117804	9448.526196
2	26-35	9249.147	95	8932.052048	9566.241952
3	36-45	9079.332	95	8777.724793	9380.939207
4	46-50	9300.464	95	8994.955905	9605.972095
5	51-55	9134.634	95	8818.790985	9450.477015
6	55+	9519.040	95	9207.067322	9831.012678

99% CI

In [307…

```
ci_cal(1000,99)
```

```
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_5724\3586372632.py:18: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  result_df = result_df.append({'Age': i, 'Mean': mean, 'CI': ci, 'Lower_Limit': lowe
r, 'Upper_Limit': upper}, ignore_index=True)
```

Out[307]:

	Age	Mean	CI	Lower_Limit	Upper_Limit
0	0-17	8850.506	99	8437.870969	9263.141031
1	18-25	9031.600	99	8630.287667	9432.912333
2	26-35	9048.253	99	8648.152652	9448.353348
3	36-45	9287.027	99	8876.420341	9697.633659
4	46-50	9057.378	99	8650.363887	9464.392113
5	51-55	9525.880	99	9121.766338	9929.993662
6	55+	9268.461	99	8851.998181	9684.923819

Final Inferences

DEMOGRAPHIC INSIGHTS

1. The majority of your sample population belongs to the 26-35 age group, which is a key demographic segment with high shopping capacity
2. The gender distribution shows that 75.3% of the population are male, which means that males are the primary customer base.
3. Approximately 40% of the sample population is married but we didn't find any differences in purchase behavior between married and unmarried individuals
4. The "B" city category has the highest number of buyers.
5. A significant portion of the population has a one-year stay in their current city

PRODUCT CATEGORY INSIGHTS

1. Product categories 1, 5, and 8 are the highest-selling categories
2. Product category 5 is common product category loved by both males and females

OCCUPATION INSIGHTS

1. The majority of buyers belong to occupation categories 4, 0, and 7.
2. Majority of Males are associated with these occupation categories.

PURCHASE BEHAVIOUR

1. Males tend to have a higher mean purchase amount compared to females. While the median purchase amount is similar, but they still have higher transactions compared to females.

2. There are more outliers in female purchase transactions
3. The majority of items are bought in the 5000-8000 range

CLT ANALYSIS

1. The CLT analysis suggests that as sample size increases, distinctions between different groups become more pronounced in case of gender. With increasing sample size, Standard error of the mean in the samples decreases
2. For average spend by gender, the distinction between male and female customers becomes more evident as the sample size increases. The male population revolve around CI of [9250.76-9624.42] where as female mean revolve around [8566-8898]
3. For average spend by marital status, there is overlap, indicating that customers spend similarly, regardless of marital status.
4. For average spend by age group,spending by Age_group 0-17 is low compared to other age groups.
5. Customers in Age_group 51-55 spend the most between [9119.96, 9939.6]

Recommendations

1. Leverage customer segmentation - Since the 26-35 age group has the highest shopping capacity, focus on creating appealing product lines and experiences for these age group can help in generating more sales. Given that the average spending of females is lower than that of males, it presents an opportunity to implement gender-specific strategies. By introducing tailored promotions and incentives for female customers, there is a potential to enhance their spending during the Black Friday sale.
2. Occupation-Centric Campaign combined with Product Category - Developing marketing campaings tailored to occupation categories. This will help align the product offerings specific to occupation.
3. Purchase Range Promotions - Consider offering special promotions, discounts, or bundles for products in the 5000-8000 price range, as this is where the majority of items are bought. Attract customers with enticing offers within this price range.
4. Social Media Engagement - Leverage social media platforms to engage with audience of 18-45 age group can help leveraging the sales.