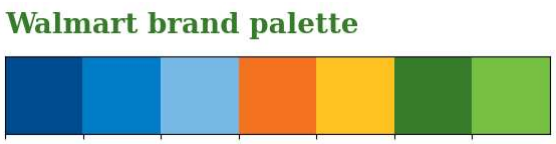
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import gdown as gd
from scipy.stats import norm,stats
from scipy.stats import chi2,chisquare,chi2_contingency
```

## About Walmart and Black Friday


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

```
sns.palplot(['#004c91', '#007dc6', '#78b9e7','#f47321', '#ffc220','#367c2b','#76c043'])
font1 = {'family':'serif','color':'#367c2b','size':20, 'weight': "bold"}
plt.title("Walmart brand palette ",loc='left',fontfamily='serif',fontsize=20,y=1.2, fontdict = font1)
plt.show()
```




### Objectives of the Project

- Perform EDA on the given dataset and find insights.
- Provide Useful Insights and Business recommendations that can help the business to grow.

## 1. Understading the Data

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

```
!wget https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094 -O walmart.csv
```

```
--2024-05-12 09:05:46--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 108.157.172.183, 108.157.172.176, 108.157.172.10, ...
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|108.157.172.183|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 23027994 (22M) [text/plain]
Saving to: ‘walmart.csv’

walmart.csv         100%[===================>]  21.96M  82.8MB/s    in 0.3s

2024-05-12 09:05:46 (82.8 MB/s) - ‘walmart.csv’ saved [23027994/23027994]
```

```
df = pd.read_csv("walmart.csv")
```

```
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 |

## Analysis to be done on following Critera

1. Average Purchase Amount by Age Group
2. Average Purchase Amount by Gender
3. Average purchase Amount by City_Category
4. Average purchase Amount by Occupation
5. Average purchase Amount by Marital_Status
6. Product prefered by Age Group
7. Product prefered by Occupation
8. Product prefered by City_Category
9. Is there any relaiton between Stay_In_Current_City_Years and City_Category
10. Is there any relaiton between Maritual Status and Age
11. Is there any relation between Product category and Purchase Amount

Categorical Variable : Product_ID, Gender, Age, Occupation, City_Category, Matrial_Status

Numerical Variable:- Purchase

## Univariant Analysis

A. Categorical Variables

1. Gender
2. Age
3. Occupation
4. City_category
5. Maritial Status
6. Product_ID

B. Between Numerical Variable

1. Purchase

## Bivariant Analysis

A. Between 1 Categorical Variable and 1 Numerical Variable

1. Purchase Amount and Gender
2. Purchase Amount and Age
3. Purchase Amount and Occupation
4. Purchase Amount and City_category
5. Purchase Amount and Maritial Status

B. Between 2 Categorical and Categorical Variables

1. Gender and Age
2. Gender and Occupation
3. Gender and City_category
4. Gender and Maritial Status
5. Age and Occupation
6. Age and City_category
7. Age and Maritial Status
8. Occupation and Occupation
9. Occupation and City_category

10. Occupation and Maritial Status
11. City_category and Maritial Status

C. Between 2 Numerical Variables

1. Product ID and Purchase

### Multivariant Analysis

A. Between 2 Categorical Variables and 1 Numerical Variable

1. Product ID and Gender and Age
2. Product ID and Gender and Occupation
3. Product ID and Gender and City_category
4. Product ID and Gender and Maritial Status
5. Product ID and Age and Occupation
6. Product ID and Age and City_category
7. Product ID and Age and Maritial Status
8. Product ID and Occupation and Occupation
9. Product ID and Occupation and City_category
10. Product ID and Occupation and Maritial Status
11. Product ID and City_category and Maritial Status
12. Purchase and Gender and Occupation
13. Purchase and Gender and City_category
14. Purchase and Gender and Maritial Status
15. Purchase and Age and Occupation
16. Purchase and Age and City_category
17. Purchase and Age and Maritial Status
18. Purchase and Occupation and Occupation
19. Purchase and Occupation and City_category
20. Purchase and Occupation and Maritial Status
21. Purchase and City_category and Maritial Status
22. Purchase and Gender and Age

B. Between 1 Catergorical and 2 Numerical Variable

1. Prduct_ID and Purchase and Gender
2. Prduct_ID and Purchase and Age
3. Prduct_ID and Purchase and Occupation
4. Prduct_ID and Purchase and City_category
5. Prduct_ID and Purchase and Maritial Status
6. Prduct_ID and Purchase and Product_ID

Double-click (or enter) to edit

## 2. Cleaning the Data

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

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

```
0
```

The data has no null values.

## 3. Non - Visual Analysis

Understaindg the data

```
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
```

```
#getting uniques values in the column
for i in df.columns :
  print(f"{i} :{df[i].nunique()} ")
```

```
User_ID :5891
Product_ID :3631
Gender :2
Age :7
Occupation :21
City_Category :3
Stay_In_Current_City_Years :5
Marital_Status :2
Product_Category :20
Purchase :18105
```

There are 58 uniquie customers. 3631 different Products. 20 differenet product category. 21 different types of Occupation.

```
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 |

```
for i in df.columns :
  print(f"{i} :{df[i].unique()} ")
```

```
User_ID :[1000001 1000002 1000003 ... 1004113 1005391 1001529]
Product_ID :['P00069042' 'P00248942' 'P00087842' ... 'P00370293' 'P00371644'
 'P00370853']
Gender :['F' 'M']
Age :['0-17' '55+' '26-35' '46-50' '51-55' '36-45' '18-25']
Occupation :[10 16 15  7 20  9  1 12 17  0  3  4 11  8 19  2 18  5 14 13  6]
City_Category :['A' 'C' 'B']
Stay_In_Current_City_Years :['2' '4+' '3' '1' '0']
Marital_Status :[0 1]
Product_Category :[ 3  1 12  8  5  4  2  6 14 11 13 15  7 16 18 10 17  9 20 19]
Purchase :[ 8370 15200  1422 ...   135   123   613]
```

```
categorical_col = ["Gender", "Age", "Occupation", "City_Category","Stay_In_Current_City_Years", "Marital_Status", "Product_Category"]
df[["Age"]].melt().groupby(["variable", "value"])[["value"]].count()/len(df)
```

| | | value |
|---|---|---|
| variable | value | |
| Age | 0-17 | 0.027455 |
| | 18-25 | 0.181178 |
| | 26-35 | 0.399200 |
| | 36-45 | 0.199999 |
| | 46-50 | 0.083082 |
| | 51-55 | 0.069993 |
| | 55+ | 0.039093 |

Observation : 80% of people are between 18-50 ( 18 % -> 18-25 , 40 % -> 26-35, 20% -> 36-45)

```
df[["Gender"]].melt().groupby(["variable", "value"])[["value"]].count()/len(df)
```

| | | value |
|---|---|---|
| variable | value | |
| Gender | F | 0.246895 |
| | M | 0.753105 |

75 Percent of people are Male and 25 percent are Female

```
df[["Marital_Status"]].melt().groupby(["variable", "value"])[["value"]].count()/len(df)
```

| | | value |
|---|---|---|
| variable | value | |
| Marital_Status | 0 | 0.590347 |
| | 1 | 0.409653 |

60 percent of people are married and 40 percent are unmarried

```
df[["City_Category"]].melt().groupby(["variable", "value"])[["value"]].count()/len(df)
```

| | | value |
|---|---|---|
| variable | value | |
| City_Category | A | 0.268549 |
| | B | 0.420263 |
| | C | 0.311189 |

42 percent are from Category B City, 26.85 percent are from Category A, 31 percent are from Category C

```
df[["Stay_In_Current_City_Years"]].melt().groupby(["variable", "value"])[["value"]].count()/len(df)
```

| | | value |
|---|---|---|
| variable | value | |
| Stay_In_Current_City_Years | 0 | 0.135252 |
| | 1 | 0.352358 |
| | 2 | 0.185137 |
| | 3 | 0.173224 |
| | 4+ | 0.154028 |

35 % Users lived in the same city for 1 year. 18 % for 2 years and 17 % for 3 years and 15 percent for 4 years and more.

## 4. Visual Analysis

### Univariant Analysis

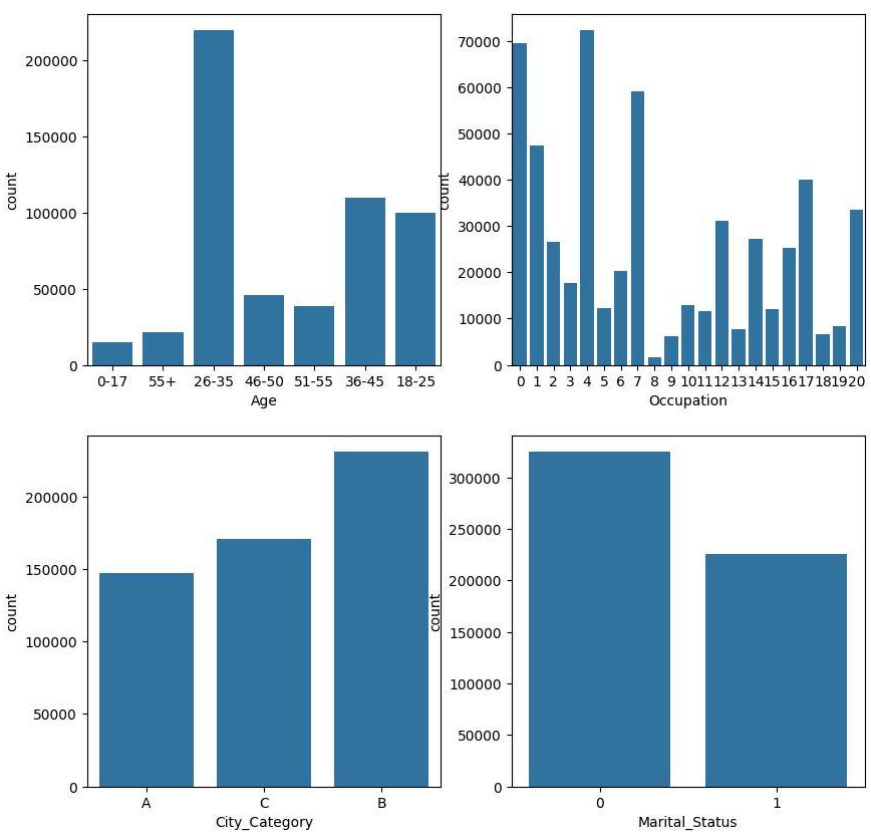
A. Categorical Variables

1. Gender
2. Age
3. Occupation
4. City_category
5. Maritial Status
6. Product_ID

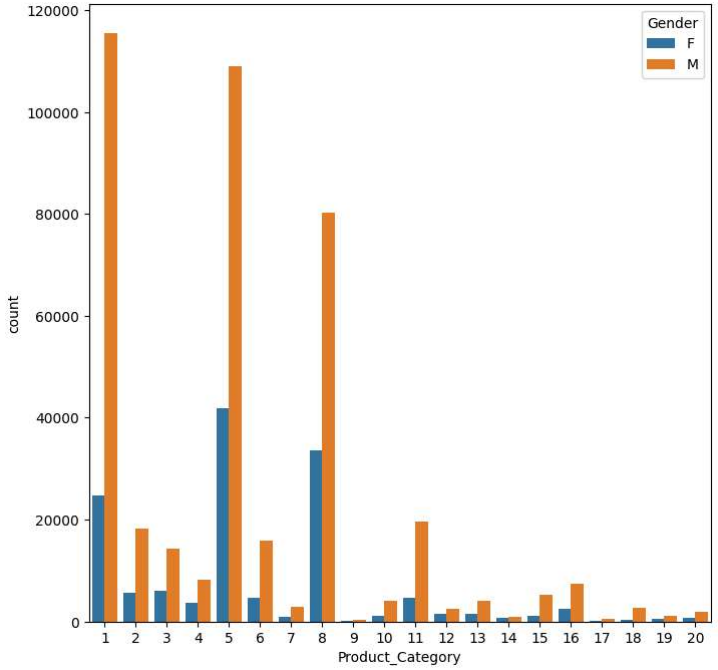
```
df.head()
```

| | 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 |

```
categorical_cols = ["Gender", "Occupation", "City_Category", "Marital_Status","Product_Category"]

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




```
plt.figure(figsize=(8,8))
sns.countplot(data=df,x ="Product_Category", hue = "Gender")
plt.show()
```

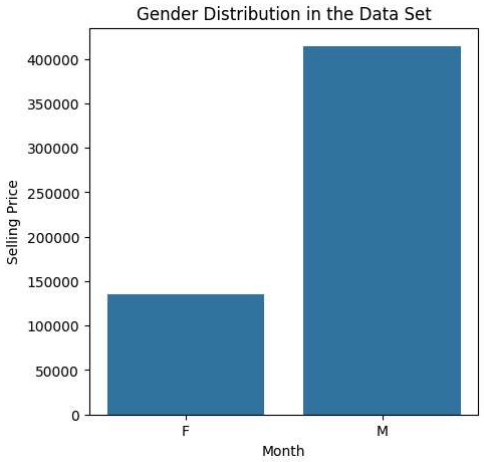



```
x = df.Gender.value_counts()
sizes = (x*100/(x.sum())).tolist()
labels = x.keys().tolist()
labels
```

```
['M', 'F']
```

```
plt.figure(figsize = (5,5))
# plt.pie(sizes, labels = labels)
sns.countplot(data = df, x = df["Gender"])
plt.xlabel("Month")
plt.ylabel("Selling Price")
plt.title("Gender Distribution in the Data Set")
plt.plot()
```

[]

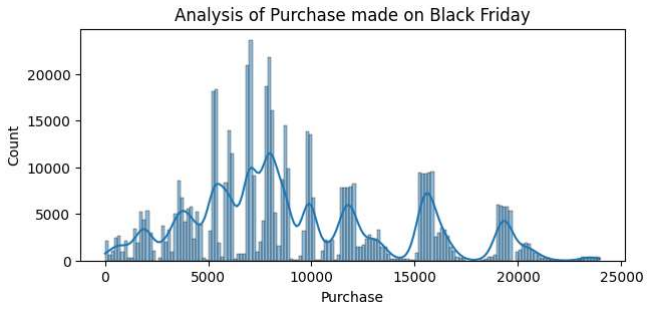



## Observations

1. Most of the users are Male (75 Percent of people are Male and 25 percent are Female)
2. There are 20 different types of Occupation and Product Cateogory
3. Most of the users belong to B City Category (42 percent are from Category B City, 26.85 percent are from Category A, 31 percent are from Category C)
4. Product Category - 1, 5, 8 and 11 have highest purchasing frequency
5. Most buyers are single (60 percent of people are married and 40 percent are unmarried)
6. 80% of people are between 18-50 ( 18 % -> 18-25 , 40 % -> 26-35, 20% -> 36-45)
7. 35 % Users lived in the same city for 1 year. 18 % for 2 years and 17 % for 3 years and 15 percent for 4 years and more.

B. For contineous variables, Boxplot and Histogram for univariate Analysis : Understanding the distribution of data and detecting outliers for contineous variables

```
plt.figure(figsize= (7,3))
sns.histplot(data = df, x = 'Purchase', kde = True)
plt.title("Analysis of Purchase made on Black Friday")
plt.show()
```



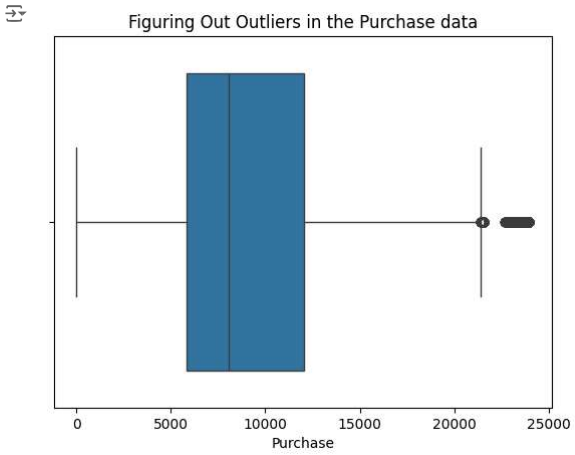

```
df["Purchase"].describe()
```

```
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
```

We can see the Purchae range from 12 to 23961 . There is a purchase spike at certain intervals.

```
plt.title("Figuring Out Outliers in the Purchase data")
sns.boxplot(x = df["Purchase"])
plt.show()
```




Start coding or generate with AI.

## Bivariant Analysis

A. Between 1 Categorical Variable and 1 Numerical Variable

1. Product ID and Gender
2. Product ID and Age
3. Product ID and Occupation
4. Product ID and City_category
5. Product ID and Maritial Status

B. Between 2 Categorical and Categorical Variables

1. Gender and Age
2. Gender and Occupation
3. Gender and City_category
4. Gender and Maritial Status
5. Age and Occupation
6. Age and City_category
7. Age and Maritial Status
8. Occupation and Occupation
9. Occupation and City_category
10. Occupation and Maritial Status
11. City_category and Maritial Status

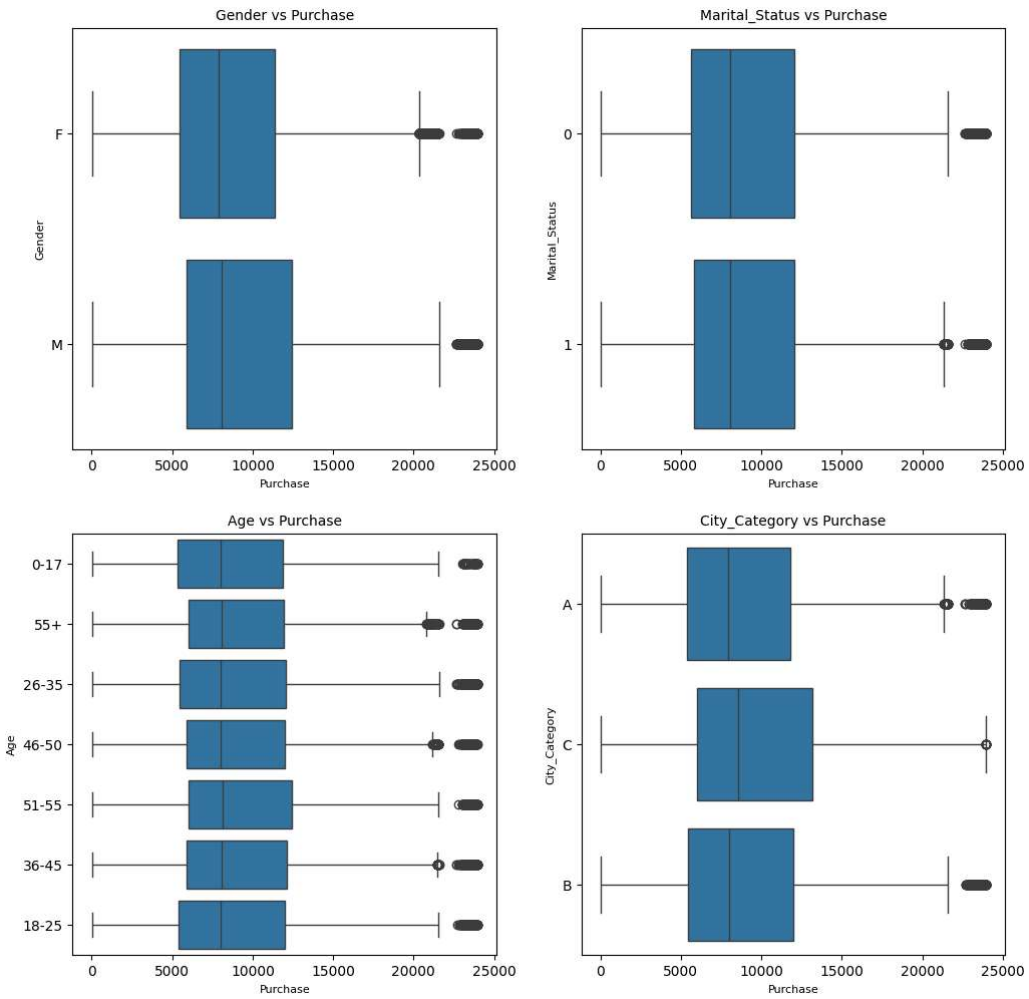
C. Between 2 Numerical Variables

1. Product ID and Purchase

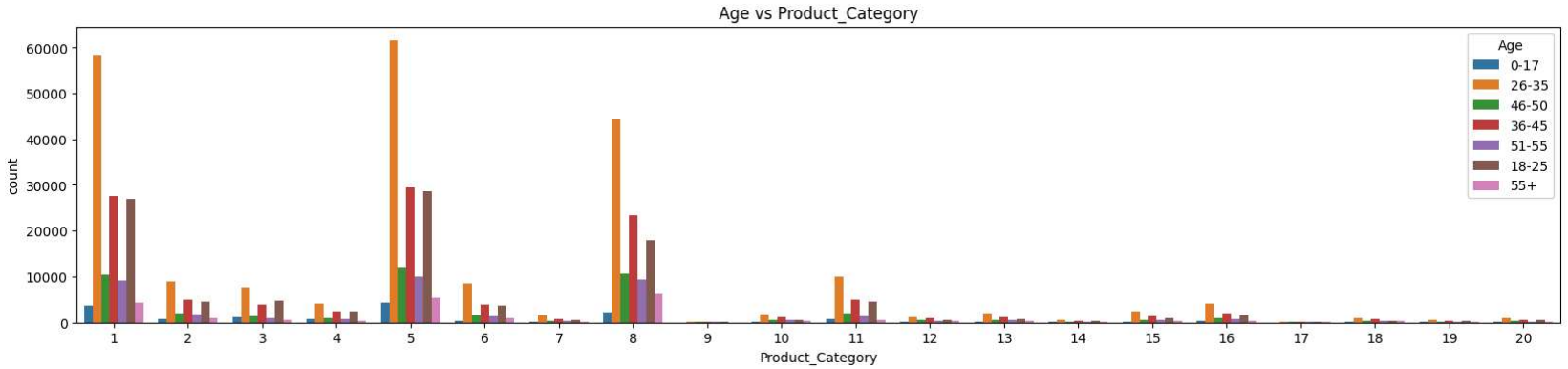
A. Between 1 Categorical Variable and 1 Numerical Variable

1. Purchase and Gender
2. Purchase and Age
3. Purchase and Occupation
4. Purchase and City_category
5. Purchase and Maritial Status

```
fig1, axs=plt.subplots(nrows=2,ncols=2, figsize=(12,12))

sns.boxplot(data=df, y='Gender',x ='Purchase',orient='h',ax=axs[0,0])
axs[0,0].set_title("Gender vs Purchase", fontsize=10)
axs[0,0].set_xlabel("Purchase", fontsize=8)
axs[0,0].set_ylabel("Gender", fontsize=8)

sns.boxplot(data=df, y='Marital_Status',x ='Purchase',orient='h',ax=axs[0,1])
axs[0,1].set_title("Marital_Status vs Purchase", fontsize=10)
axs[0,1].set_xlabel("Purchase", fontsize=8)
axs[0,1].set_ylabel("Marital_Status", fontsize=8)

sns.boxplot(data=df, y='Age',x ='Purchase',orient='h',ax=axs[1,0])
axs[1,0].set_title("Age vs Purchase", fontsize=10)
axs[1,0].set_xlabel("Purchase", fontsize=8)
axs[1,0].set_ylabel("Age", fontsize=8)

sns.boxplot(data=df, y='City_Category',x ='Purchase',orient='h',ax=axs[1,1])
axs[1,1].set_title("City_Category vs Purchase", fontsize=10)
axs[1,1].set_xlabel("Purchase", fontsize=8)
axs[1,1].set_ylabel("City_Category", fontsize=8)

plt.show()
```

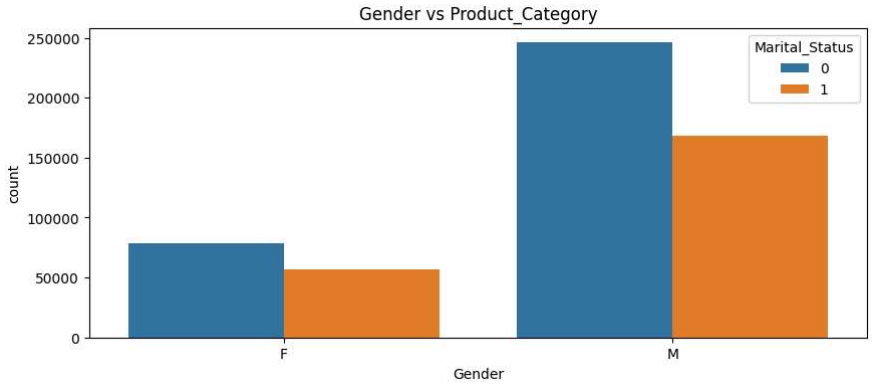



```
plt.figure(figsize=(20,4))
sns.countplot(data=df, x=df['Product_Category'], hue=df['Age'])
plt.title('Age vs Product_Category')
plt.show()
```

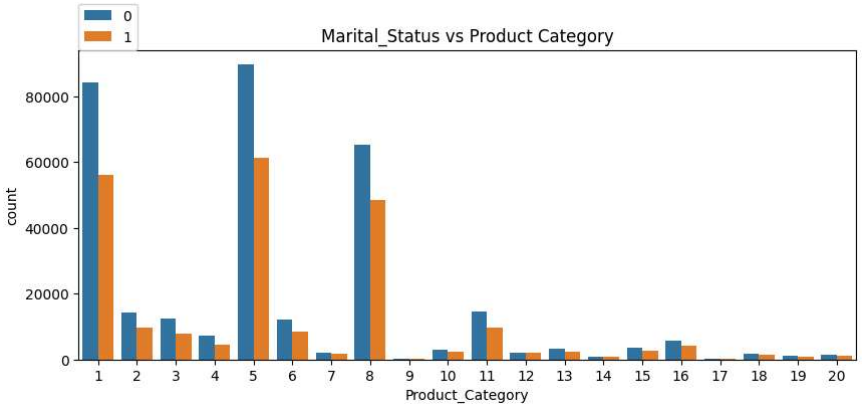



1,5,8 are most purchaed products

```
plt.figure(figsize=(10,4))
sns.countplot(data=df, x=df['Gender'], hue=df['Marital_Status'])
plt.title('Gender vs Product_Category')
plt.show()
```



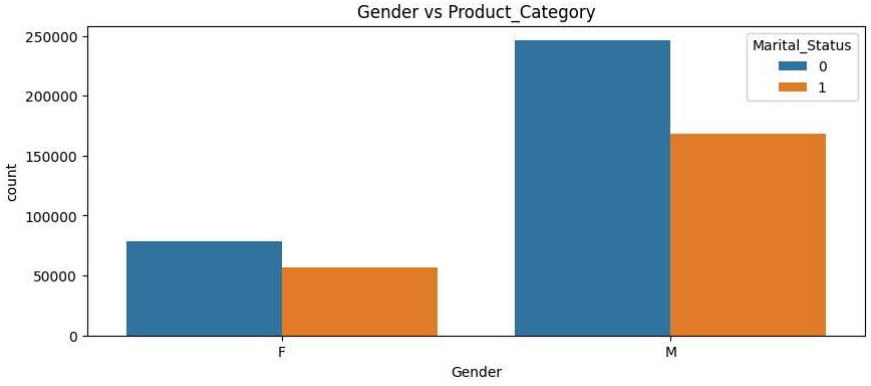

```
plt.figure(figsize=(10,4))
sns.countplot(data=df, x=df['Product_Category'],hue=df['Marital_Status'])
plt.legend(['0','1'],loc=[0,1])
plt.title('Marital_Status vs Product Category')
plt.show()
```



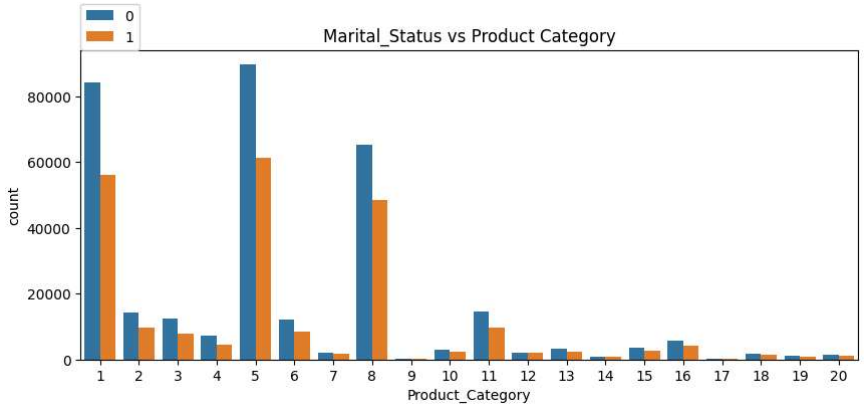

Start coding or generate with AI.

Insights: The users aged between 26 and 35 are purchasing more. The most purchased product categories are 1,5 and 8

```
plt.figure(figsize=(10,4))
sns.countplot(data=df, x=df['Gender'], hue=df['Marital_Status'])
plt.title('Gender vs Product_Category')
plt.show()
```

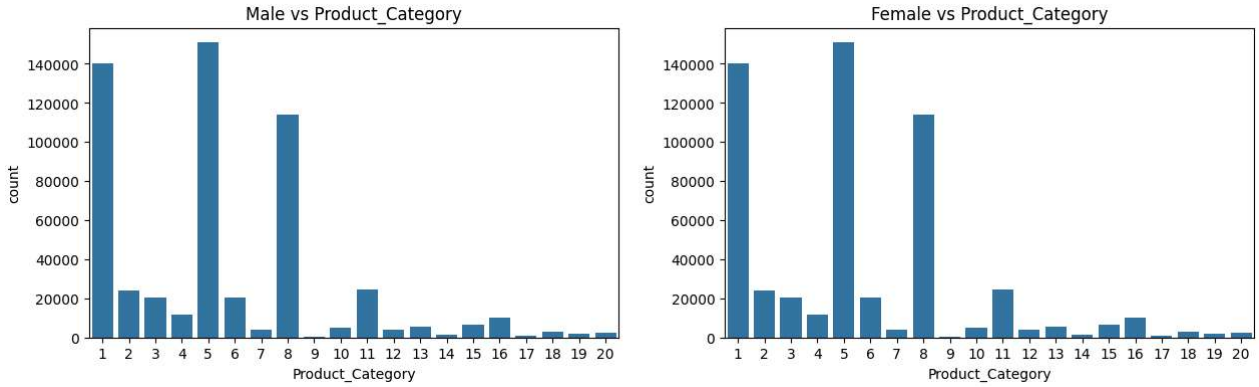



```
plt.figure(figsize=(10,4))
sns.countplot(data=df, x=df['Product_Category'],hue=df['Marital_Status'])
plt.legend(['0','1'],loc=[0,1])
plt.title('Marital_Status vs Product Category')
plt.show()
```




Insights: In all age groups Unmarried users are spending more than the Married users.

```
plt.figure(figsize=(15,4))
plt.subplot(1,2,1)
sns.countplot(data=df.loc[df['Gender']=='M'], x=df['Product_Category'])
plt.title('Male vs Product_Category')
plt.subplot(1,2,2)
sns.countplot(data=df.loc[df['Gender']=='F'], x=df['Product_Category'])
plt.title('Female vs Product_Category')
plt.show()
```




Insights: Both Male and Female customers purchased the products 1,5,8 more than the others

**Mean and Standard deviation for whole dataset**

```
m,sigma=df['Purchase'].agg(['mean','std'])
m,sigma
print('Mean:',m,'  Std Dev:',sigma)
r1,r2=(m+(sigma * norm.ppf(0.025)), m-(sigma * norm.ppf(0.025)))
print('Range: (',r1,' , ',r2,')')
```

```
Mean: 9263.968712959126   Std Dev: 5023.065393820582
Range: ( -581.0585509187204  ,  19108.995976836974 )
```

**Data for Gender Male**

```
df_mpur= df.loc[df['Gender']=='M']
df_mpur.head()
```

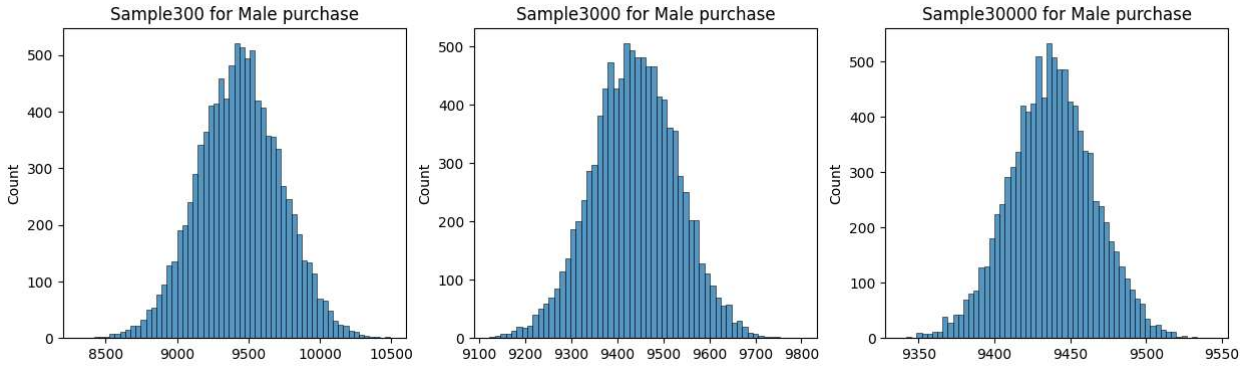
| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 1000002 | P00285442 | M | 55+ | 16 | C | 4+ | 0 | 8 | 7969 |
| 5 | 1000003 | P00193542 | M | 26-35 | 15 | A | 3 | 0 | 1 | 15227 |
| 6 | 1000004 | P00184942 | M | 46-50 | 7 | B | 2 | 1 | 1 | 19215 |
| 7 | 1000004 | P00346142 | M | 46-50 | 7 | B | 2 | 1 | 1 | 15854 |
| 8 | 1000004 | P0097242 | M | 46-50 | 7 | B | 2 | 1 | 1 | 15686 |

```
mu,sig = df_mpur['Purchase'].agg(['mean','std']) # Mean and std of Purchase amount by male from actual data
mu,sig
```

```
(9437.526040472265, 5092.18620977797)
```

```
sample300=[np.mean(df_mpur['Purchase'].sample(300)) for i in range(10000)]
sample3000=[np.mean(df_mpur['Purchase'].sample(3000)) for i in range(10000)]
sample30000=[np.mean(df_mpur['Purchase'].sample(30000)) for i in range(10000)]
```

```
plt.figure(figsize=(15,4))
plt.subplot(1,3,1)
sns.histplot(sample300)
plt.title('Sample300 for Male purchase')
plt.subplot(1,3,2)
sns.histplot(sample3000)
plt.title('Sample3000 for Male purchase')
plt.subplot(1,3,3)
sns.histplot(sample30000)
plt.title('Sample30000 for Male purchase')
plt.show()
```




```
m_mu300=np.mean(sample300)
m_sig300=np.std(sample300)
m_mu3000=np.mean(sample3000)
m_sig3000=np.std(sample3000)
m_mu30000=np.mean(sample30000)
m_sig30000=np.std(sample30000)
print('Mean,Std of Actual Male purchase:',mu,' , ',sig)
print('Mean,Std of Sample300:',m_mu300,' , ',m_sig300)
print('Mean,Std of Sample3000:',m_mu3000,' , ',m_sig3000)
print('Mean,Std of Sample30000:',m_mu30000,' , ',m_sig30000)
```

```
Mean,Std of Actual Male purchase: 9437.526040472265  ,  5092.18620977797
Mean,Std of Sample300: 9435.891392333335  ,  292.4044334381297
Mean,Std of Sample3000: 9437.646418933333  ,  92.58188541668784
Mean,Std of Sample30000: 9437.68795027  ,  28.029869429278563
```

Insights: The higher the sample size the sample30000 mean is equal to the actual mean

**95% Confidence Interval for the actual data and samples**

```
a,b=(mu+(sig * norm.ppf(0.025)), mu-(sig * norm.ppf(0.025)))
a,b
```

```
(-542.9755332640834, 19418.027614208615)
```

```
a300,b300 = (m_mu300+(m_sig300 * norm.ppf(0.025)), m_mu300-(m_sig300 * norm.ppf(0.025)))
a3000,b3000 = (m_mu3000+(m_sig3000 * norm.ppf(0.025)), m_mu3000-(m_sig3000 * norm.ppf(0.025)))
a30000,b30000 = (m_mu30000+(m_sig30000 * norm.ppf(0.025)), m_mu30000-(m_sig30000 * norm.ppf(0.025)))
print("95% Confidence Interval Ranges according to the size for Male data")
print("CI range for whole dataset: (", r1,' , ',r2,')')
print("Actual data CI Range: (", a,' , ',b,')')
print("Sample300 data CI Range: (", a300,' , ',b300,')')
print("Sample3000 data CI Range: (", a3000,' , ',b3000,')')
print("Sample30000 data CI Range: (", a30000,' , ',b30000,')')
```

```
95% Confidence Interval Ranges according to the size for Male data
CI range for whole dataset: ( -581.0585509187204  ,  19108.995976836974 )
Actual data CI Range: ( -542.9755332640834  ,  19418.027614208615 )
Sample300 data CI Range: ( 8862.78923387476  ,  10008.993550791909 )
Sample3000 data CI Range: ( 9256.18925789581  ,  9619.103579970855 )
Sample30000 data CI Range: ( 9382.750415697254  ,  9492.625484842745 )
```

**Insights:**

- i. Is the confidence interval computed using the entire dataset wider for one of the gender Male? Why is this the case?

  > No, total dataset CI lies between -260 to 18774 and CI for Makle purchase data is between 227 to 19081

- ii. How is the width of the confidence interval affected by the sample size?

  > The larger the sample size the width is getting decreased.

- iii. Do the confidence intervals for different sample sizes overlap?

  > All the samples CI's overlap with each other

- iv. How does the sample size affect the shape of the distributions of the means?

  > The larger the sample size the distribution is more likely to be normal distribution and the mean of sample is closer to mean of population.

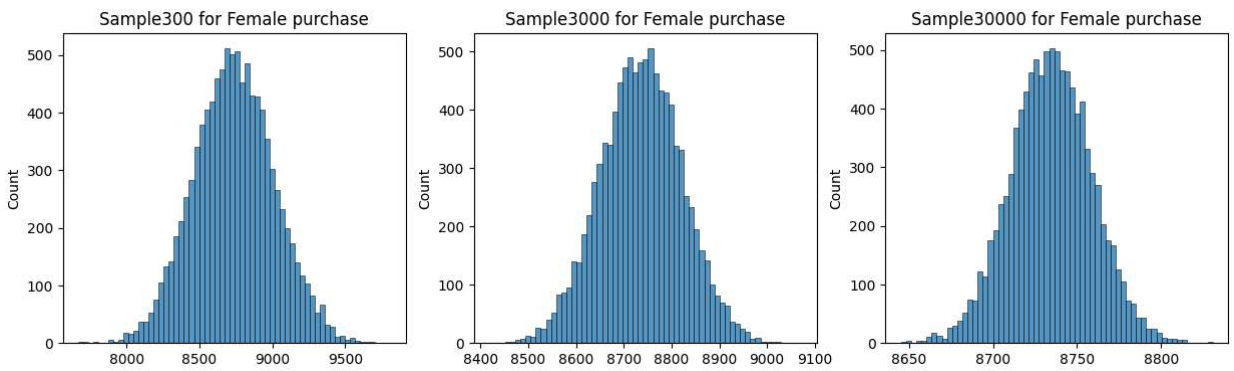
**Data Analysis for Gender Female**

```
df_fpur= df.loc[df['Gender']=='F'] # Mean and std of Purchase amount by male
fmu,fsig = df_fpur['Purchase'].agg(['mean','std']) # Mean and std of Purchase amount by male from actual data
fmu,fsig
```

```
(8734.565765155476, 4767.233289291458)
```

```
fsample300=[np.mean(df_fpur['Purchase'].sample(300)) for i in range(10000)]
fsample3000=[np.mean(df_fpur['Purchase'].sample(3000)) for i in range(10000)]
fsample30000=[np.mean(df_fpur['Purchase'].sample(30000)) for i in range(10000)]
```

```
plt.figure(figsize=(15,4))
plt.subplot(1,3,1)
sns.histplot(fsample300)
plt.title('Sample300 for Female purchase')
plt.subplot(1,3,2)
sns.histplot(fsample3000)
plt.title('Sample3000 for Female purchase')
plt.subplot(1,3,3)
sns.histplot(fsample30000)
plt.title('Sample30000 for Female purchase')
plt.show()
```




```
fmu300=np.mean(fsample300)
fsig300=np.std(fsample300)
fmu3000=np.mean(fsample3000)
fsig3000=np.std(fsample3000)
fmu30000=np.mean(fsample30000)
fsig30000=np.std(fsample30000)
print('Mean,Std of Actual Female purchase:',fmu,' , ',fsig)
print('Mean,Std of Female Sample300:',fmu300,' , ',fsig300)
print('Mean,Std of Female Sample3000:',fmu3000,' , ',fsig3000)
print('Mean,Std of Female Sample30000:',fmu30000,' , ',fsig30000)
```

```
Mean,Std of Actual Female purchase: 8734.565765155476  ,  4767.233289291458
Mean,Std of Female Sample300: 8736.056577666666  ,  275.02621736330656
Mean,Std of Female Sample3000: 8735.924182166667  ,  85.36161977460377
Mean,Std of Female Sample30000: 8734.382345996666  ,  24.388485375122762
```

```
a,b=(fmu+(fsig * norm.ppf(0.025)), fmu-(fsig * norm.ppf(0.025)))
a300,b300 = (fmu300+(fsig300 * norm.ppf(0.025)), fmu300-(fsig300 * norm.ppf(0.025)))
a3000,b3000 = (fmu3000+(fsig3000 * norm.ppf(0.025)), fmu3000-(fsig3000 * norm.ppf(0.025)))
a30000,b30000 = (fmu30000+(fsig30000 * norm.ppf(0.025)), fmu30000-(fsig30000 * norm.ppf(0.025)))
print("95% Confidence Interval Ranges according to the size for Female purchase data")
print("Actual data CI Range: (", a,' , ',b,')')
print("Sample300 data CI Range: (", a300,' , ',b300,')')
print("Sample3000 data CI Range: (", a3000,' , ',b3000,')')
print("Sample30000 data CI Range: (", a30000,' , ',b30000,')')
```

```
95% Confidence Interval Ranges according to the size for Female purchase data
Actual data CI Range: ( -609.0397877562009  ,  18078.17131806715 )
Sample300 data CI Range: ( 8197.0150968303  ,  9275.098058503032 )
Sample3000 data CI Range: ( 8568.618481746442  ,  8903.229882586893 )
Sample30000 data CI Range: ( 8686.581793023945  ,  8782.182898969388 )
```

**95% Confidence Interval Ranges according to the size for Male purchase:**

- Whole Data Range: ( -260.77106809139514 , 18774.192045134234 )
- Actual Male data CI Range: ( -227.45043782258836 , 19081.9324309718 )
- Sample300 data CI Range: ( 8869.07350686749 , 9990.676806465844 )
- Sample3000 data CI Range: ( 9252.328414253736 , 9602.4362692796 )
- Sample30000 data CI Range: ( 9373.402943539179 , 9481.097680220817 )

**95% Confidence Interval Ranges according to the size for Female purchase:**

- Whole Data Range: ( -260.77106809139514 , 18774.192045134234 )
- Actual Female data CI Range: ( -273.3840138802334 , 17746.464546098276 )
- Sample300 data CI Range: ( 8218.759821391945 , 9253.136445274726 )
- Sample3000 data CI Range: ( 8574.074650651304 , 8901.129941815363 )
- Sample30000 data CI Range: ( 8690.087213283003 , 8782.713687170335 )

**Insights:**

- i. Is the confidence interval computed using the entire dataset wider for one of the gender Female? Why is this the case?

> No, total dataset CI lies between -260 to 18774 and CI for female purchase data is between -273 to 17746

- ii. How is the width of the confidence interval affected by the sample size?

> The larger the sample size the width is getting decreased.

- iii. Do the confidence intervals for different sample sizes overlap?

> All the samples CI's overlap with each other

- iv. How does the sample size affect the shape of the distributions of the means?

> The larger the sample size the distribution is more likely to be normal distribution and the mean of sample is closer to mean of population.

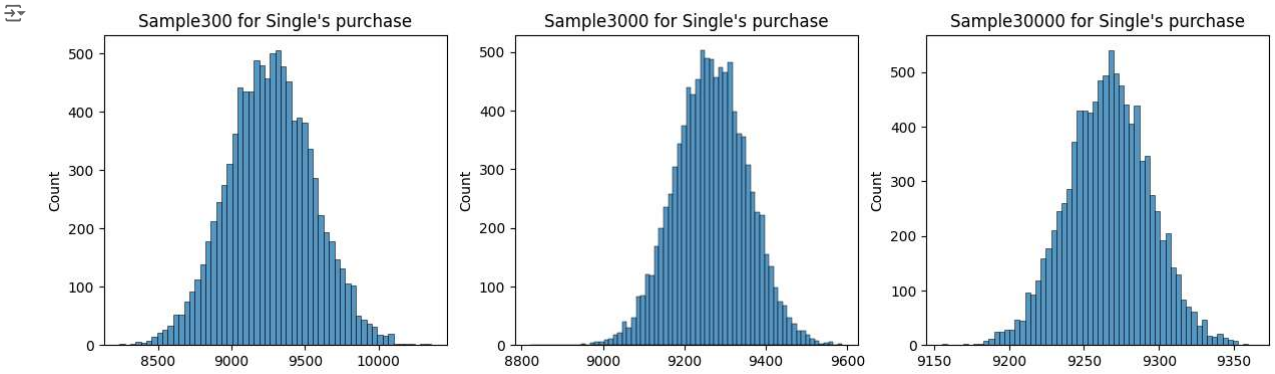
**Customer's purchase data analysis whose Marital Status is Single**

```
df_si_pur= df.loc[df['Marital_Status']==0]
si_mu,si_sig = df_si_pur['Purchase'].agg(['mean','std'])
si_mu,si_sig
```

```
(9265.907618921507, 5027.347858674449)
```

```
si_sample300=[np.mean(df_si_pur['Purchase'].sample(300)) for i in range(10000)]
si_sample3000=[np.mean(df_si_pur['Purchase'].sample(3000)) for i in range(10000)]
si_sample30000=[np.mean(df_si_pur['Purchase'].sample(30000)) for i in range(10000)]
```

```
plt.figure(figsize=(15,4))
plt.subplot(1,3,1)
sns.histplot(si_sample300)
plt.title("Sample300 for Single's purchase")
plt.subplot(1,3,2)
sns.histplot(si_sample3000)
plt.title("Sample3000 for Single's purchase")
plt.subplot(1,3,3)
sns.histplot(si_sample30000)
plt.title("Sample30000 for Single's purchase")
plt.show()
```




```
si_mu300=np.mean(si_sample300)
si_sig300=np.std(si_sample300)
si_mu3000=np.mean(si_sample3000)
si_sig3000=np.std(si_sample3000)
si_mu30000=np.mean(si_sample30000)
si_sig30000=np.std(si_sample30000)
print("Mean,Std of Actual Single's purchase:",si_mu,' , ',si_sig)
print("Mean,Std of Single's Sample300:",si_mu300,' , ',si_sig300)
print("Mean,Std of Single's Sample3000:",si_mu3000,' , ',si_sig3000)
print("Mean,Std of Single's Sample30000:",si_mu30000,' , ',si_sig30000)
```

```
Mean,Std of Actual Single's purchase: 9265.907618921507  ,  5027.347858674449
Mean,Std of Single's Sample300: 9264.752088333333  ,  293.4197601565526
Mean,Std of Single's Sample3000: 9266.155450533333  ,  89.93486829690966
Mean,Std of Single's Sample30000: 9266.102251453334  ,  27.786682942676713
```

```
a,b=(si_mu+(si_sig * norm.ppf(0.025)), si_mu-(si_sig * norm.ppf(0.025)))
a300,b300 = (si_mu300+(si_sig300 * norm.ppf(0.025)), si_mu300-(si_sig300 * norm.ppf(0.025)))
a3000,b3000 = (si_mu3000+(si_sig3000 * norm.ppf(0.025)), si_mu3000-(si_sig3000 * norm.ppf(0.025)))
a30000,b30000 = (si_mu30000+(si_sig30000 * norm.ppf(0.025)), si_mu30000-(si_sig30000 * norm.ppf(0.025)))
print("95% Confidence Interval Ranges according to the size for Single's purchase data")
print("Actual Single's data CI Range: (", a,' , ',b,')')
print("Sample300 data CI Range: (", a300,' , ',b300,')')
print("Sample3000 data CI Range: (", a3000,' , ',b3000,')')
print("Sample30000 data CI Range: (", a30000,' , ',b30000,')')
```

```
95% Confidence Interval Ranges according to the size for Single's purchase data
Actual Single's data CI Range: ( -587.5131218349761  ,  19119.328359677987 )
Sample300 data CI Range: ( 8689.65992607411  ,  9839.844250592556 )
Sample3000 data CI Range: ( 9089.886347717038  ,  9442.424553349629 )
Sample30000 data CI Range: ( 9211.641353635854  ,  9320.563149270814 )
```

**Insights:**

- i. Is the confidence interval computed using the entire dataset wider for Marital Status Single? Why is this the case?

  > Yes, total dataset CI lies between -260 to 18774 and CI for Unmarried customer purchase data is between -275 to 18793

- ii. How is the width of the confidence interval affected by the sample size?

  > The larger the sample size the width is getting decreased.

- iii. Do the confidence intervals for different sample sizes overlap?

  > All the samples CI's overlap with each other

- iv. How does the sample size affect the shape of the distributions of the means?

  > The larger the sample size the distribution is more likely to be normal distribution and the mean of sample is closer to mean of population.

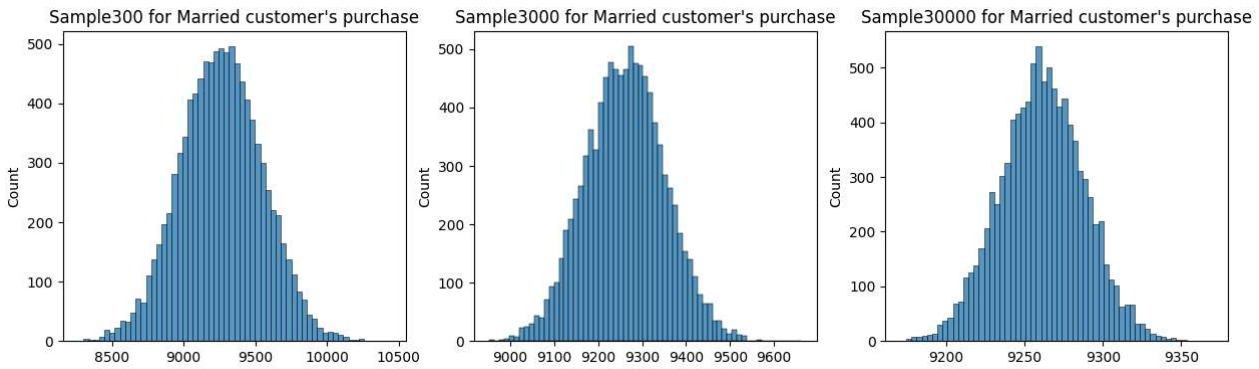
**Customer's purchase data analysis whose Marital Status is Married**

```
df_ma_pur= df.loc[df['Marital_Status']==1]
ma_mu,ma_sig = df_ma_pur['Purchase'].agg(['mean','std'])
ma_mu,ma_sig
```

```
(9261.174574082374, 5016.897377793055)
```

```
ma_sample300=[np.mean(df_ma_pur['Purchase'].sample(300)) for i in range(10000)]
ma_sample3000=[np.mean(df_ma_pur['Purchase'].sample(3000)) for i in range(10000)]
ma_sample30000=[np.mean(df_ma_pur['Purchase'].sample(30000)) for i in range(10000)]
```

```
plt.figure(figsize=(15,4))
plt.subplot(1,3,1)
sns.histplot(ma_sample300)
plt.title("Sample300 for Married customer's purchase")
plt.subplot(1,3,2)
sns.histplot(ma_sample3000)
plt.title("Sample3000 for Married customer's purchase")
plt.subplot(1,3,3)
sns.histplot(ma_sample30000)
plt.title("Sample30000 for Married customer's purchase")
plt.show()
```




```
ma_mu300=np.mean(ma_sample300)
ma_sig300=np.std(ma_sample300)
ma_mu3000=np.mean(ma_sample3000)
ma_sig3000=np.std(ma_sample3000)
ma_mu30000=np.mean(ma_sample30000)
ma_sig30000=np.std(ma_sample30000)
print("Mean,Std of Actual Married customer's purchase:",ma_mu,' , ',ma_sig)
print("Mean,Std of Married customer's Sample300:",ma_mu300,' , ',ma_sig300)
print("Mean,Std of Married customer's Sample3000:",ma_mu3000,' , ',ma_sig3000)
print("Mean,Std of Married customer's Sample30000:",ma_mu30000,' , ',ma_sig30000)
```

```
Mean,Std of Actual Married customer's purchase: 9261.174574082374  ,  5016.897377793055
Mean,Std of Married customer's Sample300: 9261.570837000001  ,  288.07387878045466
Mean,Std of Married customer's Sample3000: 9261.831846466666  ,  90.65425154454138
Mean,Std of Married customer's Sample30000: 9261.179055713335  ,  26.780150330087103
```

```
a,b=(ma_mu+(ma_sig * norm.ppf(0.025)), ma_mu-(ma_sig * norm.ppf(0.025)))
a300,b300 = (ma_mu300+(ma_sig300 * norm.ppf(0.025)), ma_mu300-(ma_sig300 * norm.ppf(0.025)))
a3000,b3000 = (ma_mu3000+(ma_sig3000 * norm.ppf(0.025)), ma_mu3000-(ma_sig3000 * norm.ppf(0.025)))
a30000,b30000 = (ma_mu30000+(ma_sig30000 * norm.ppf(0.025)), ma_mu30000-(ma_sig30000 * norm.ppf(0.025)))
print("95% Confidence Interval Ranges according to the size for Married customer's purchase data")
print("CI range for whole dataset: (", r1,' , ',r2,')')
print("Actual data CI Range: (", a,' , ',b,')')
print("Sample300 data CI Range: (", a300,' , ',b300,')')
print("Sample3000 data CI Range: (", a3000,' , ',b3000,')')
print("Sample30000 data CI Range: (", a30000,' , ',b30000,')')
```

```
95% Confidence Interval Ranges according to the size for Married customer's purchase data
CI range for whole dataset: ( -581.0585509187204  ,  19108.995976836974 )
Actual data CI Range: ( -571.7636005254535  ,  19094.1127486902 )
Sample300 data CI Range: ( 8696.956409703553  ,  9826.18526429645 )
Sample3000 data CI Range: ( 9084.152778393931  ,  9439.510914539402 )
Sample30000 data CI Range: ( 9208.690925565796  ,  9313.667185860875 )
```

**Insights:**

- i. Is the confidence interval computed using the entire dataset wider for Marital Status Married? Why is this the case?

  > No, total dataset CI lies between -260 to 18774 and CI for Married customer purchase data is between -239 to 18746

- ii. How is the width of the confidence interval affected by the sample size?

  > The larger the sample size the width is getting decreased.

- iii. Do the confidence intervals for different sample sizes overlap?

  > All the samples CI's overlap with each other

- iv. How does the sample size affect the shape of the distributions of the means?

  > The larger the sample size the distribution is more likely to be normal distribution and the mean of sample is closer to mean of population.

**Customer's purchase data analysis for users of different age groups**

Analysis for customers whose age <=25

```
df['Age'].unique()
```

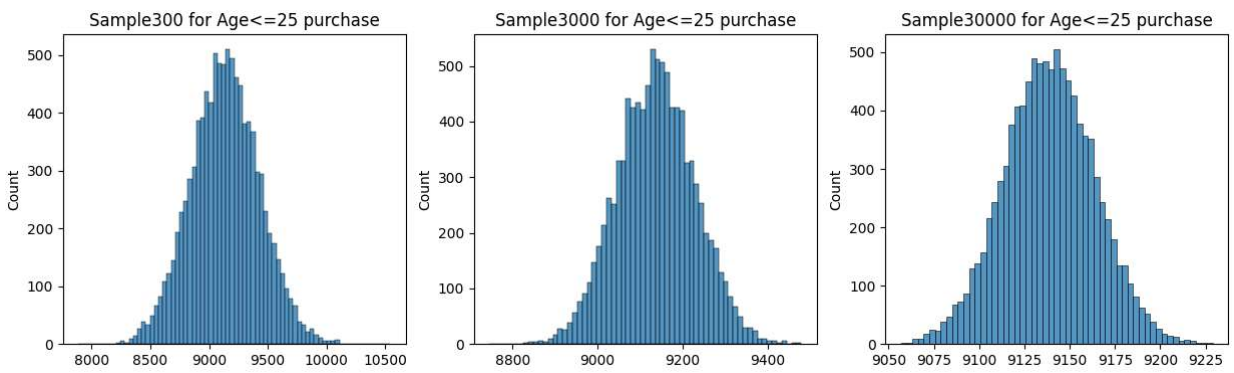
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

```
df_age_25= df.loc[(df.Age=='0-17') | (df.Age=='18-25')]
age_25_mu,age_25_sig = df_age_25['Purchase'].agg(['mean','std'])
age_25_mu,age_25_sig
```

```
(9138.581220264548, 5045.103594263333)
```

```
age_25_sample300=[np.mean(df_age_25['Purchase'].sample(300)) for i in range(10000)]
age_25_sample3000=[np.mean(df_age_25['Purchase'].sample(3000)) for i in range(10000)]
age_25_sample30000=[np.mean(df_age_25['Purchase'].sample(30000)) for i in range(10000)]
```

```
plt.figure(figsize=(15,4))
plt.subplot(1,3,1)
sns.histplot(age_25_sample300)
plt.title("Sample300 for Age<=25 purchase")
plt.subplot(1,3,2)
sns.histplot(age_25_sample3000)
plt.title("Sample3000 for Age<=25 purchase")
plt.subplot(1,3,3)
sns.histplot(age_25_sample30000)
plt.title("Sample30000 for Age<=25 purchase")
plt.show()
```




```
age_25_mu300=np.mean(age_25_sample300)
age_25_sig300=np.std(age_25_sample300)
age_25_mu3000=np.mean(age_25_sample3000)
age_25_sig3000=np.std(age_25_sample3000)
age_25_mu30000=np.mean(age_25_sample30000)
age_25_sig30000=np.std(age_25_sample30000)
print("Mean,Std of Actual data Age<=25 purchase:",age_25_mu,' , ',age_25_sig)
print("Mean,Std of Age<=25 Sample300:",age_25_mu300,' , ',age_25_sig300)
print("Mean,Std of Age<=25 Sample3000:",age_25_mu3000,' , ',age_25_sig3000)
print("Mean,Std of Age<=25 Sample30000:",age_25_mu30000,' , ',age_25_sig30000)
```

```
Mean,Std of Actual data Age<=25 purchase: 9138.581220264548  ,  5045.103594263333
Mean,Std of Age<=25 Sample300: 9136.193037333334  ,  291.4344271431514
Mean,Std of Age<=25 Sample3000: 9138.241122299998  ,  91.7702783881238
Mean,Std of Age<=25 Sample30000: 9138.510645710001  ,  25.183948655806937
```

```
a,b=(age_25_mu+(age_25_sig * norm.ppf(0.025)), age_25_mu-(age_25_sig * norm.ppf(0.025)))
a300,b300 = (age_25_mu300+(age_25_sig300 * norm.ppf(0.025)), age_25_mu300-(age_25_sig300 * norm.ppf(0.025)))
a3000,b3000 = (age_25_mu3000+(age_25_sig3000 * norm.ppf(0.025)), age_25_mu3000-(age_25_sig3000 * norm.ppf(0.025)))
a30000,b30000 = (age_25_mu30000+(age_25_sig30000 * norm.ppf(0.025)), age_25_mu30000-(age_25_sig30000 * norm.ppf(0.025)))
print("95% Confidence Interval Ranges according to the size for Age<=25 purchase data")
print("CI range for whole dataset: (", r1,' , ',r2,')')
print("Actual data CI Range: (", a,' , ',b,')')
print("Sample300 data CI Range: (", a300,' , ',b300,')')
print("Sample3000 data CI Range: (", a3000,' , ',b3000,')')
print("Sample30000 data CI Range: (", a30000,' , ',b30000,')')
```

```
95% Confidence Interval Ranges according to the size for Age<=25 purchase data
CI range for whole dataset: ( -581.0585509187204  ,  19108.995976836974 )
Actual data CI Range: ( -749.6401227651659  ,  19026.802563294263 )
Sample300 data CI Range: ( 8564.992056277695  ,  9707.394018388974 )
Sample3000 data CI Range: ( 8958.374681808062  ,  9318.107562791934 )
Sample30000 data CI Range: ( 9089.151013356113  ,  9187.870278063889 )
```

**Insights:**

- i. Is the confidence interval computed using the entire dataset wider for data with age <=25? Why is this the case?

  > Yes, total dataset CI lies between -260 to 18774 and CI for customer with Age <=25 purchase data is between -458 to 18736

- ii. How is the width of the confidence interval affected by the sample size?

  > The larger the sample size the width is getting decreased.

- iii. Do the confidence intervals for different sample sizes overlap?

  > All the samples CI's overlap with each other

- iv. How does the sample size affect the shape of the distributions of the means?

  > The larger the sample size the distribution is more likely to be normal distribution and the mean of sample is closer to mean of population.

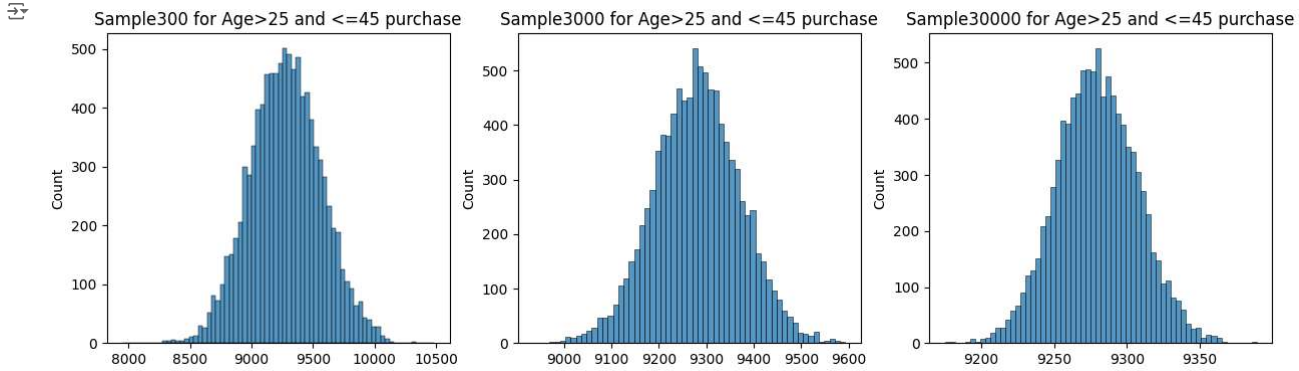
Analysis for customers whose age>25 and <=45

```
df_age_45= df.loc[(df.Age=='26-35') | (df.Age=='36-45')]
age_45_mu,age_45_sig = df_age_45['Purchase'].agg(['mean','std'])
age_45_mu,age_45_sig
```

```
(9278.94557645631, 5014.797971975373)
```

```
age_45_sample300=[np.mean(df_age_45['Purchase'].sample(300)) for i in range(10000)]
age_45_sample3000=[np.mean(df_age_45['Purchase'].sample(3000)) for i in range(10000)]
age_45_sample30000=[np.mean(df_age_45['Purchase'].sample(30000)) for i in range(10000)]
```

```
plt.figure(figsize=(15,4))
plt.subplot(1,3,1)
sns.histplot(age_45_sample300)
plt.title("Sample300 for Age>25 and <=45 purchase")
plt.subplot(1,3,2)
sns.histplot(age_45_sample3000)
plt.title("Sample3000 for Age>25 and <=45 purchase")
plt.subplot(1,3,3)
sns.histplot(age_45_sample30000)
plt.title("Sample30000 for Age>25 and <=45 purchase")
plt.show()
```




```
age_45_mu300=np.mean(age_45_sample300)
age_45_sig300=np.std(age_45_sample300)
age_45_mu3000=np.mean(age_45_sample3000)
age_45_sig3000=np.std(age_45_sample3000)
age_45_mu30000=np.mean(age_45_sample30000)
age_45_sig30000=np.std(age_45_sample30000)
print("Mean,Std of Actual data Age>25 and <=45 purchase:",age_45_mu,' , ',age_45_sig)
print("Mean,Std of Age>25 and <=45 Sample300:",age_45_mu300,' , ',age_45_sig300)
print("Mean,Std of Age>25 and <=45 Sample3000:",age_45_mu3000,' , ',age_45_sig3000)
print("Mean,Std of Age>25 and <=45 Sample30000:",age_45_mu30000,' , ',age_45_sig30000)
```

```
Mean,Std of Actual data Age>25 and <=45 purchase: 9278.94557645631  ,  5014.797971975373
Mean,Std of Age>25 and <=45 Sample300: 9279.257480000002  ,  289.81933374839
Mean,Std of Age>25 and <=45 Sample3000: 9279.888566833333  ,  90.51542590662199
Mean,Std of Age>25 and <=45 Sample30000: 9279.371957416666  ,  27.654469432065035
```

```
a,b=(age_45_mu+(age_45_sig * norm.ppf(0.025)), age_45_mu-(age_45_sig * norm.ppf(0.025)))
a300,b300 = (age_45_mu300+(age_45_sig300 * norm.ppf(0.025)), age_45_mu300-(age_45_sig300 * norm.ppf(0.025)))
a3000,b3000 = (age_45_mu3000+(age_45_sig3000 * norm.ppf(0.025)), age_45_mu3000-(age_45_sig3000 * norm.ppf(0.025)))
a30000,b30000 = (age_45_mu30000+(age_45_sig30000 * norm.ppf(0.025)), age_45_mu30000-(age_45_sig30000 * norm.ppf(0.025)))
print("95% Confidence Interval Ranges according to the size for Age<>25 and <=45 purchase data")
print("CI range for whole dataset: (", r1,' , ',r2,')')
print("Actual data CI Range: (", a,' , ',b,')')
print("Sample300 data CI Range: (", a300,' , ',b300,')')
print("Sample3000 data CI Range: (", a3000,' , ',b3000,')')
print("Sample30000 data CI Range: (", a30000,' , ',b30000,')')
```

```
95% Confidence Interval Ranges according to the size for Age<>25 and <=45 purchase data
CI range for whole dataset: ( -581.0585509187204  ,  19108.995976836974 )
Actual data CI Range: ( -549.8778383599274  ,  19107.768991272547 )
Sample300 data CI Range: ( 8711.222023829763  ,  9847.29293617024 )
Sample3000 data CI Range: ( 9102.48159201105  ,  9457.295541655616 )
Sample30000 data CI Range: ( 9225.170193318256  ,  9333.573721515077 )
```

**Insights:**

- i. Is the confidence interval computed using the entire dataset wider for data with age > 25 and <=45? Why is this the case?

  > No, total dataset CI lies between -581 to 19108 and CI for customer with age>25 and <=45 purchase data is between -581 to 19108

- ii. How is the width of the confidence interval affected by the sample size?

  > The larger the sample size the width is getting decreased.

- iii. Do the confidence intervals for different sample sizes overlap?

  > All the samples CI's overlap with each other

- iv. How does the sample size affect the shape of the distributions of the means?

  > The larger the sample size the distribution is more likely to be normal distribution and the mean of sample is closer to mean of population.

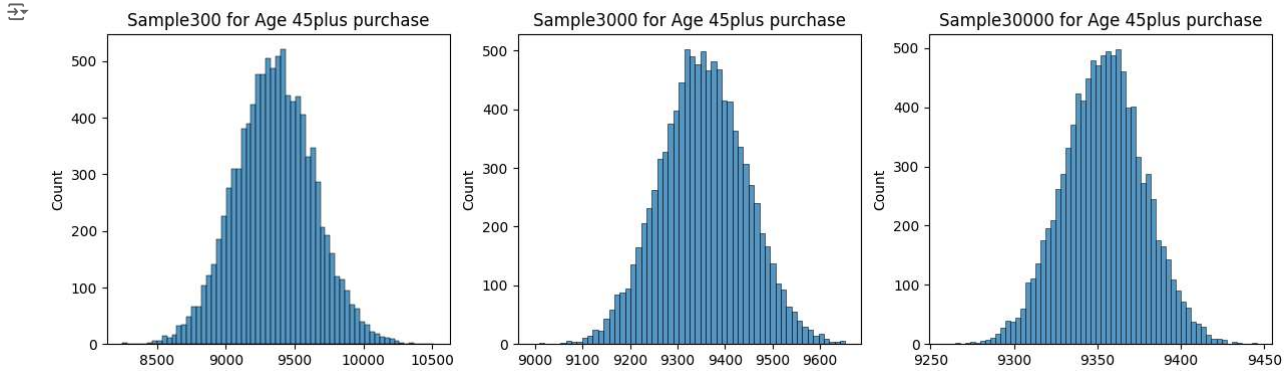
Analysis for customers whose age>45

```
df_age_45plus= df.loc[(df.Age=='46-50') | (df.Age=='51-55') | (df.Age=='55+')]
age_45plus_mu,age_45plus_sig = df_age_45plus['Purchase'].agg(['mean','std'])
age_45plus_mu,age_45plus_sig
```

```
(9353.399258320247, 5022.32674777519)
```

```
age_45plus_sample300=[np.mean(df_age_45plus['Purchase'].sample(300)) for i in range(10000)]
age_45plus_sample3000=[np.mean(df_age_45plus['Purchase'].sample(3000)) for i in range(10000)]
age_45plus_sample30000=[np.mean(df_age_45plus['Purchase'].sample(30000)) for i in range(10000)]
```

```
plt.figure(figsize=(15,4))
plt.subplot(1,3,1)
sns.histplot(age_45plus_sample300)
plt.title("Sample300 for Age 45plus purchase")
plt.subplot(1,3,2)
sns.histplot(age_45plus_sample3000)
plt.title("Sample3000 for Age 45plus purchase")
plt.subplot(1,3,3)
sns.histplot(age_45plus_sample30000)
plt.title("Sample30000 for Age 45plus purchase")
plt.show()
```




```
age_45plus_mu300=np.mean(age_45plus_sample300)
age_45plus_sig300=np.std(age_45plus_sample300)
age_45plus_mu3000=np.mean(age_45plus_sample3000)
age_45plus_sig3000=np.std(age_45plus_sample3000)
age_45plus_mu30000=np.mean(age_45plus_sample30000)
age_45plus_sig30000=np.std(age_45plus_sample30000)
print("Mean,Std of Actual Age 45plus purchase:",age_45plus_mu,' , ',age_45plus_sig)
print("Mean,Std of Age 45plus Sample300:",age_45plus_mu300,' , ',age_45plus_sig300)
print("Mean,Std of Age45 Sample3000:",age_45plus_mu3000,' , ',age_45plus_sig3000)
print("Mean,Std of Age 45plus Sample30000:",age_45plus_mu30000,' , ',age_45plus_sig30000)
```

```
Mean,Std of Actual Age 45plus purchase: 9353.399258320247  ,  5022.32674777519
Mean,Std of Age 45plus Sample300: 9354.930338666669  ,  285.9136091250968
Mean,Std of Age45 Sample3000: 9353.4762238  ,  90.97844832220468
Mean,Std of Age 45plus Sample30000: 9353.731718590001  ,  24.438056721696203
```

```
a,b=(age_45plus_mu+(age_45plus_sig * norm.ppf(0.025)), age_45plus_mu-(age_45plus_sig * norm.ppf(0.025)))
a300,b300 = (age_45plus_mu300+(age_45plus_sig300 * norm.ppf(0.025)), age_45plus_mu300-(age_45plus_sig300 * norm.ppf(0.025)))
a3000,b3000 = (age_45plus_mu3000+(age_45plus_sig3000 * norm.ppf(0.025)), age_45plus_mu3000-(age_45plus_sig3000 * norm.ppf(0.025)))
a30000,b30000 = (age_45plus_mu30000+(age_45plus_sig30000 * norm.ppf(0.025)), age_45plus_mu30000-(age_45plus_sig30000 * norm.ppf(0.025)))
print("95% Confidence Interval Ranges according to the size for Age 45plus purchase data")
print("CI range for whole dataset: (", r1,' , ',r2,')')
print("Actual data CI Range: (", a,' , ',b,')')
print("Sample300 data CI Range: (", a300,' , ',b300,')')
print("Sample3000 data CI Range: (", a3000,' , ',b3000,')')
print("Sample30000 data CI Range: (", a30000,' , ',b30000,')')
```

```
95% Confidence Interval Ranges according to the size for Age 45plus purchase data
CI range for whole dataset: ( -581.0585509187204  ,  19108.995976836974 )
Actual data CI Range: ( -490.1802859113068  ,  19196.9788025518 )
Sample300 data CI Range: ( 8794.549962091616  ,  9915.310715241721 )
Sample3000 data CI Range: ( 9175.161741719141  ,  9531.79070588086 )
Sample30000 data CI Range: ( 9305.83400756333  ,  9401.629429616672 )
```

**Insights:**

- i. Is the confidence interval computed using the entire dataset wider for data with age >45? Why is this the case?

  > No, total dataset CI lies between -581 to 19108 and CI for customer with age>45 purchase data is between -581 to 19108

- ii. How is the width of the confidence interval affected by the sample size?

  > The larger the sample size the width is getting decreased.

- iii. Do the confidence intervals for different sample sizes overlap?

  > All the samples CI's overlap with each other

- iv. How does the sample size affect the shape of the distributions of the means?

  > The larger the sample size the distribution is more likely to be normal distribution and the mean of sample is closer to mean of population.

Double-click (or enter) to edit

B. Between 2 Categorical and Categorical Variables

1. Gender and Age
2. Gender and Occupation
3. Gender and City_category
4. Gender and Maritial Status
5. Age and Occupation
6. Age and City_category
7. Age and Maritial Status
8. Occupation and Occupation
9. Occupation and City_category
10. Occupation and Maritial Status
11. City_category and Maritial Status

```
contingency_table = pd.crosstab(df['Gender'], df['Age'])

# Perform the Chi-Square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Print the results
print(f"Chi-Square statistic: {chi2}")
print(f"P-value: {p_value}")
print(f"Degrees of freedom: {dof}")
if p_value < 0.05:
    print("Reject H0")
    print("there is a significant association between the 'Gender' and 'Age' variables.")
else:
    print("Fail to reject H0")
    print("there is not significant association between the 'Gender' and 'Age' variables.")
```

```
Chi-Square statistic: 1412.7156576195039
P-value: 4.2750569064909935e-302
Degrees of freedom: 6
Reject H0
there is a significant association between the 'Gender' and 'Age' variables.
```

```
contingency_table = pd.crosstab(df['Gender'], df['Occupation'])

# Perform the Chi-Square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)
```

```
# Print the results
print(f"Chi-Square statistic: {chi2}")
print(f"P-value: {p_value}")
print(f"Degrees of freedom: {dof}")
if p_value < 0.05:
    print("Reject H0")
    print("there is a significant association between the 'Gender' and 'Occupation' variables.")
else:
    print("Fail to reject H0")
    print("there is not significant association between the 'Gender' and 'City_Category' variables.")
```

```
Chi-Square statistic: 41664.592979596964
P-value: 0.0
Degrees of freedom: 20
Reject H0
there is a significant association between the 'Gender' and 'Occupation' variables.
```

```
contingency_table = pd.crosstab(df['Gender'], df['City_Category'])

# Perform the Chi-Square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Print the results
print(f"Chi-Square statistic: {chi2}")
print(f"P-value: {p_value}")
print(f"Degrees of freedom: {dof}")
if p_value < 0.05:
    print("Reject H0")
    print("There is a significant association between the 'Gender' and 'City_Category' variables.")
else:
    print("Fail to reject H0")
    print("There is not significant association between the 'Gender' and 'City_Category' variables.")
```

```
Chi-Square statistic: 33.58382571304351
P-value: 5.097590042852447e-08
Degrees of freedom: 2
Reject H0
There is a significant association between the 'Gender' and 'City_Category' variables.
```

```
contingency_table = pd.crosstab(df['Gender'], df['Marital_Status'])

# Perform the Chi-Square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Print the results
print(f"Chi-Square statistic: {chi2}")
print(f"P-value: {p_value}")
print(f"Degrees of freedom: {dof}")
if p_value < 0.05:
    print("Reject H0")
    print("there is a significant association between the 'Gender' and 'City_Category' variables.")
else:
    print("Fail to reject H0")
    print("there is not significant association between the 'Gender' and 'City_Category' variables.")
```

```
Chi-Square statistic: 74.00272697523472
P-value: 7.80091894540745e-18
Degrees of freedom: 1
Reject H0
there is a significant association between the 'Gender' and 'City_Category' variables.
```

```
contingency_table = pd.crosstab(df['Age'], df['City_Category'])

# Perform the Chi-Square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Print the results
print(f"Chi-Square statistic: {chi2}")
print(f"P-value: {p_value}")
print(f"Degrees of freedom: {dof}")
if p_value < 0.05:
    print("Reject H0")
    print("There is a significant association between the 'Age' and 'City_Category' variables.")
else:
    print("Fail to reject H0")
    print("There is not significant association between the 'Age' and 'City_Category' variables.")
```

```
Chi-Square statistic: 22368.805059695536
P-value: 0.0
Degrees of freedom: 12
Reject H0
There is a significant association between the 'Age' and 'City_Category' variables.
```

```
contingency_table = pd.crosstab(df['Age'], df['Marital_Status'])

# Perform the Chi-Square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Print the results
print(f"Chi-Square statistic: {chi2}")
print(f"P-value: {p_value}")
print(f"Degrees of freedom: {dof}")
if p_value < 0.05:
    print("Reject H0")
    print("There is a significant association between the 'Age' and 'Marital_Status' variables.")
else:
    print("Fail to reject H0")
    print("There is not significant association between the 'Age' and 'Marital_Status' variables.")
```

```
Chi-Square statistic: 65038.31034963805
P-value: 0.0
Degrees of freedom: 6
Reject H0
There is a significant association between the 'Age' and 'Marital_Status' variables.
```

```
contingency_table = pd.crosstab(df['Occupation'], df['Marital_Status'])

# Perform the Chi-Square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Print the results
print(f"Chi-Square statistic: {chi2}")
print(f"P-value: {p_value}")
print(f"Degrees of freedom: {dof}")
if p_value < 0.05:
    print("Reject H0")
    print("There is a significant association between the 'Occupation' and 'Marital_Status' variables.")
else:
    print("Fail to reject H0")
    print("There is not significant association between the 'Occupation' and 'Marital_Status' variables.")
```

```
Chi-Square statistic: 18710.023962434694
P-value: 0.0
Degrees of freedom: 20
Reject H0
There is a significant association between the 'Occupation' and 'Marital_Status' variables.
```

```
contingency_table = pd.crosstab(df['City_Category'], df['Marital_Status'])

# Perform the Chi-Square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Print the results
print(f"Chi-Square statistic: {chi2}")
print(f"P-value: {p_value}")
print(f"Degrees of freedom: {dof}")
if p_value < 0.05:
    print("Reject H0")
    print("There is a significant association between the 'City_Category' and 'Marital_Status' variables.")
else:
    print("Fail to reject H0")
    print("There is not significant association between the 'City_Category' and 'Marital_Status' variables.")
```

```
Chi-Square statistic: 870.9131695144557
P-value: 7.649061125706496e-190
Degrees of freedom: 2
Reject H0
There is a significant association between the 'City_Category' and 'Marital_Status' variables.
```

C. Between 2 Numerical Variables

Product ID and Purchase

## Multivariant Analysis

A. Between 2 Categorical Variables and 1 Numerical Variable

1. Product ID and Gender and Age
2. Product ID and Gender and Occupation
3. Product ID and Gender and City_category
4. Product ID and Gender and Maritial Status
5. Product ID and Age and Occupation
6. Product ID and Age and City_category
7. Product ID and Age and Maritial Status
8. Product ID and Occupation and Occupation
9. Product ID and Occupation and City_category
10. Product ID and Occupation and Maritial Status
11. Product ID and City_category and Maritial Status
12. Purchase and Gender and Occupation
13. Purchase and Gender and City_category
14. Purchase and Gender and Maritial Status
15. Purchase and Age and Occupation
16. Purchase and Age and City_category
17. Purchase and Age and Maritial Status
18. Purchase and Occupation and Occupation
19. Purchase and Occupation and City_category
20. Purchase and Occupation and Maritial Status
21. Purchase and City_category and Maritial Status
22. Purchase and Gender and Age

B. Between 1 Catergorical and 2 Numerical Variable

1. Prduct_ID and Purchase and Gender
2. Prduct_ID and Purchase and Age
3. Prduct_ID and Purchase and Occupation
4. Prduct_ID and Purchase and City_category
5. Prduct_ID and Purchase and Maritial Status
6. Prduct_ID and Purchase and Product_ID

```
avgamt_gender = df.groupby(['User_ID', 'Gender'])[['Purchase']].sum()
avgamt_gender = avgamt_gender.reset_index()
avgamt_gender
```

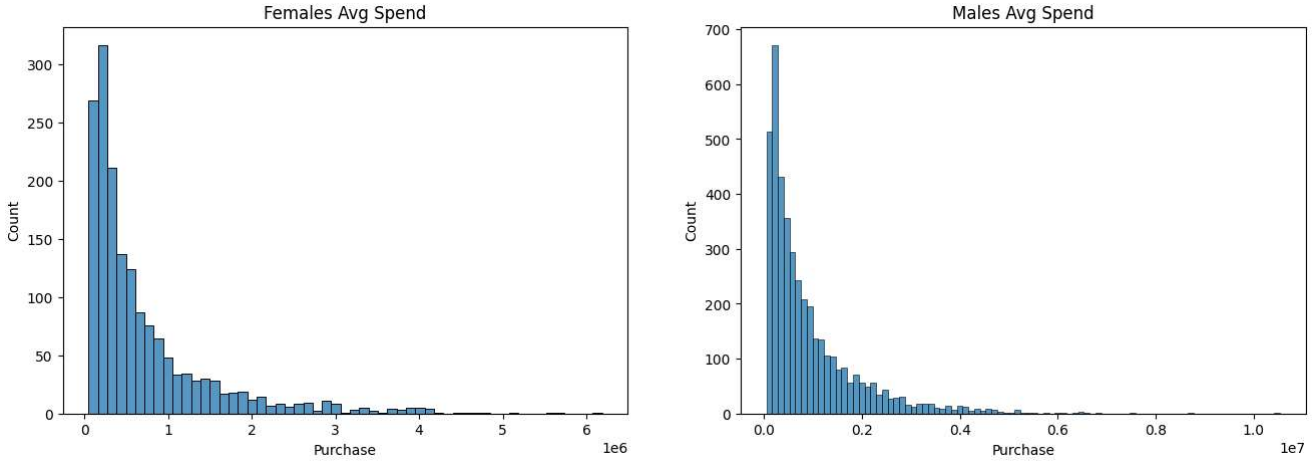
| | User_ID | Gender | Purchase |
|---|---|---|---|
| 0 | 1000001 | F | 334093 |
| 1 | 1000002 | M | 810472 |
| 2 | 1000003 | M | 341635 |
| 3 | 1000004 | M | 206468 |
| 4 | 1000005 | M | 821001 |
| ... | ... | ... | ... |
| 5886 | 1006036 | F | 4116058 |
| 5887 | 1006037 | F | 1119538 |
| 5888 | 1006038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

5891 rows × 3 columns

Next steps: Generate code with avgamt_gender View recommended plots

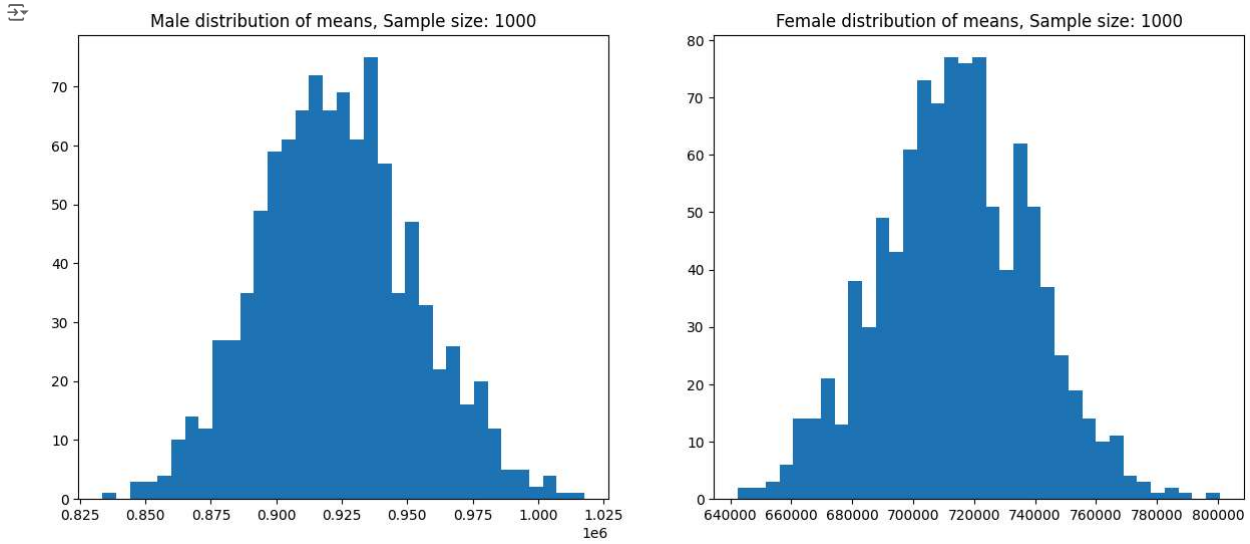
```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(16,5))
sns.histplot(data=avgamt_gender[avgamt_gender['Gender']=='F']['Purchase'], ax=axs[0]).set_title("Females Avg Spend")
sns.histplot(avgamt_gender[avgamt_gender['Gender']=='M']['Purchase'], ax=axs[1]).set_title("Males Avg Spend")
```

```
Text(0.5, 1.0, 'Males Avg Spend')
```




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


#Finding the sample(sample size=1000) for avg purchase amount for males and females
genders = ["M", "F"]

sample_size = 1000

num_repitions = 1000
male_means = []
female_means = []

for i in range(num_repitions):
    male_mean = avgamt_male.sample(sample_size, replace=True)['Purchase'].mean()
    female_mean = avgamt_female.sample(sample_size, replace=True)['Purchase'].mean()

    male_means.append(male_mean)
    female_means.append(female_mean)
```

```
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(15, 6))

axis[0].hist(male_means, bins=35)
axis[1].hist(female_means, bins=35)
axis[0].set_title("Male distribution of means, Sample size: 1000")
axis[1].set_title("Female distribution of means, Sample size: 1000")

plt.show()
```




```
#Taking the value for z at 90% confidence interval as:
z90=1.645 #90% Confidence Interval

print("Population avg spend amount for Male: {:.2f}".format(avgamt_male['Purchase'].mean()))
print("Population avg spend amount for Female: {:.2f}\n".format(avgamt_female['Purchase'].mean()))

print("Sample avg spend amount for Male: {:.2f}".format(np.mean(male_means)))
print("Sample avg spend amount for Female: {:.2f}\n".format(np.mean(female_means)))

print("Sample std for Male: {:.2f}".format(pd.Series(male_means).std()))
print("Sample std for Female: {:.2f}\n".format(pd.Series(female_means).std()))

print("Sample std error for Male: {:.2f}".format(pd.Series(male_means).std()/np.sqrt(1000)))
print("Sample std error for Female: {:.2f}\n".format(pd.Series(female_means).std()/np.sqrt(1000)))

sample_mean_male=np.mean(male_means)
sample_mean_female=np.mean(female_means)

sample_std_male=pd.Series(male_means).std()
sample_std_female=pd.Series(female_means).std()

sample_std_error_male=sample_std_male/np.sqrt(1000)
sample_std_error_female=sample_std_female/np.sqrt(1000)

Upper_Limit_male=z90*sample_std_error_male + sample_mean_male
Lower_Limit_male=sample_mean_male - z90*sample_std_error_male

Upper_Limit_female=z90*sample_std_error_female + sample_mean_female
Lower_Limit_female=sample_mean_female - z90*sample_std_error_female

print("Male_CI: ",[Lower_Limit_male,Upper_Limit_male])
print("Female_CI: ",[Lower_Limit_female,Upper_Limit_female])
```

```
Population avg spend amount for Male: 925344.40
Population avg spend amount for Female: 712024.39

Sample avg spend amount for Male: 923242.04
Sample avg spend amount for Female: 714043.83
```

```
Sample std for Male: 29635.32
Sample std for Female: 24804.32

Sample std error for Male: 937.15
Sample std error for Female: 784.38

Male_CI:  [921700.4246697414, 924783.6517702586]
Female_CI:  [712753.5263983972, 715334.141639603]
```

Start coding or generate with AI.

## Analysis from the Data

1. Target Male Shoppers Since male customers account for a significant portion of Black Friday sales and tend to spend more per transaction on average, Walmart should tailor its marketing strategies and product offerings to incentivize higher spending among male customers while ensuring competitive pricing for female-oriented products.
2. Focus on 26 - 45 Age Group With the age group between 26 and 45 contributing to the majority of sales, Walmart should specifically cater to the preferences and needs of this demographic.This could include offering exclusive deals on products that are popular among this age group.
3. Engage Younger Shoppers Knowing that customers in the 0 - 17 age group have the lowest spending per transaction, Walmart can try to increase their spending per transaction by offering them more attractive discounts, coupons, or rewards programs. It's essential to start building brand loyalty among younger consumers.
4. Customer Segmentation Since customers in the 18 - 25, 26 - 35, and 46 - 50 age groups exhibit similar buying characteristics, and so do the customers in 36 - 45 and 55+, Walmart can optimize its product selection to cater to the preferences of these age groups. Also, Walmart can use this information to adjust their pricing strategies for different age groups.
5. Enhance the 51 - 55 Age Group Shopping Experience Considering that customers aged 51 - 55 have the highest spending per transaction, Walmart offer them exclusive pre-sale access, special discount or provide personalized product recommendations for this age group. Walmart can also introduce loyalty programs specifically designed to reward and retain customers in the 51 - 55 age group.
6. Post-Black Friday Engagement After Black Friday, walmart should engage with customers who made purchases by sending follow-up emails or offers for related products. This can help increase customer retention and encourage repeat business throughout the holiday season and beyond.

Start coding or generate with AI.

## Use the Central Theorm to compute the interval. Change the sample size to observe the distribution of the mean of the expenses by female and male customers

```
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 |

```
male_customers = df[df["Gender"] == "M"]
female_customers = df[df["Gender"] == "F"]
```

```
male_customers["Purchase"].mean() , male_customers["Purchase"].std()
```

```
(9437.526040472265, 5092.18620977797)
```

```
female_customers["Purchase"].mean() , female_customers["Purchase"].std()
```

```
(8734.565765155476, 4767.233289291458)
```

```
mSample_300 = [male_customers["Purchase"].sample(300).mean() for i in range(10000)]
mSample_300 = np.array(mSample_300)
print(f"Std : {mSample_300.std()}, Mean : {mSample_300.mean()}")
norm(loc =mSample_300.mean() , scale = mSample_300.std() ).interval(0.95)
```

```
Std : 293.1683272838221, Mean : 9432.133646333332
(8857.53428344919, 10006.733009217474)
```

```
fSample_300 = [female_customers["Purchase"].sample(300).mean() for i in range(10000)]
fSample_300 = np.array(fSample_300)
# mSample_300.mean(), mSample_300.std()
print(f"Std : {fSample_300.std()}, Mean : {fSample_300.mean()}")
norm(loc =fSample_300.mean() , scale = fSample_300.std() ).interval(0.95)
```

```
Std : 272.55628950568064, Mean : 8732.792969666665
(8198.59245847566, 9266.99348085767)
```

**Observation for 300 sample**

1. The average purchase made by a sample of 300 male customers is 9436.69 with a standard deviation of 5092.18
2. The average purchaese made by a sample of 300 female customers is 8732.79 with a standard deviation of 272.55
3. The interval estimate of 300 male sample is 8857 and 10006
4. The interval estimate of 300 female sample is 8198 and 9266
5. 95 percent interval for male customer is tighter than female customers.

```
mSample_3000 = [male_customers["Purchase"].sample(3000).mean() for i in range(10000)]
mSample_3000 = np.array(mSample_3000)
print(f"Std : {mSample_3000.std()}, Mean : {mSample_3000.mean()}")
norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
```

```
Std : 91.79765915169874, Mean : 9436.749161166665
(9256.829055364251, 9616.669266969078)
```

```
fSample_3000 = [female_customers["Purchase"].sample(3000).mean() for i in range(10000)]
fSample_3000 = np.array(fSample_3000)
# mSample_300.mean(), mSample_300.std()
print(f"Std : {fSample_3000.std()}, Mean : {fSample_3000.mean()}")
norm(loc =fSample_3000.mean() , scale = fSample_3000.std() ).interval(0.95)
```

```
Std : 85.19928256197923, Mean : 8733.925333266667
(8566.937807936536, 8900.912858596797)
```

**Observation for 3000 sample**

The average purchase made by a sample of 3000 male customers is 9436.74 with a standard deviation of 91.79

The average purchaese made by a sample of 3000 female customers is 8733.92 with a standard deviation of 85.19

The interval estimate of 3000 male sample is 9256 and 9616

The interval estimate of 3000 female sample is 8566 and 8900

95 percent interval for male customer is tighter than female customers.

```
mSample_30000 = [male_customers["Purchase"].sample(30000).mean() for i in range(10000)]
mSample_30000 = np.array(mSample_30000)
print(f"Std : {mSample_30000.std()}, Mean : {mSample_30000.mean()}")
norm(loc =mSample_30000.mean() , scale = mSample_30000.std() ).interval(0.95)
```

```
Std : 28.116855096884304, Mean : 9437.615977266667
(9382.507953918242, 9492.724000615091)
```

```
fSample_30000 = [female_customers["Purchase"].sample(30000).mean() for i in range(10000)]
fSample_30000 = np.array(fSample_30000)
# mSample_300.mean(), mSample_300.std()
print(f"Std : {fSample_30000.std()}, Mean : {fSample_30000.mean()}")
norm(loc =fSample_30000.mean() , scale = fSample_30000.std() ).interval(0.95)
```

```
Std : 24.38640497937937, Mean : 8735.170268133334
(8687.373792661341, 8782.966743605326)
```

**Observation for 30000 sample**

The average purchase made by a sample of 3000 male customers is 9437.61 with a standard deviation of 28.11

The average purchaese made by a sample of 3000 female customers is 8735.17 with a standard deviation of 24.38

The interval estimate of 3000 male sample is 9382.50 and 9492.72

The interval estimate of 3000 female sample is 8687.37 and 8782.96

95 percent interval for male customer is tighter than female customers.

## Conclude the result 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?

1. 95 percent confidence interval of Male and Female spends are overlapping, it suggest that there may be a **statistically significant difference in the spending of Males and Females.**
2. The average purchases made by Male is more than the average purchase made by female customers. Walmart might consider focusing on Male customers since they tend to spend more in comparision to female customers.

**Actionalbe Insights:**

1. Introducing more Male specific products.
2. Introducting some credit to Female customers. This could be in the form of credit cards espically to Female.
3. Discount on Products on Womens Day.

Start coding or generate with AI.

## Use the Central Limit Theorm to compute the interval . Change the sample size to Observe the distribution of mean of the expense by married and unmarried customers.

```
m_customers = df[df["Marital_Status"] == 1]
um_customers = df[df["Marital_Status"] == 0]
```

```
m_customers["Purchase"].mean() , m_customers["Purchase"].std()
```

```
(9261.174574082374, 5016.897377793055)
```

```
um_customers["Purchase"].mean() , um_customers["Purchase"].std()
```

```
(9265.907618921507, 5027.347858674449)
```

```
mSample_300 = [m_customers["Purchase"].sample(300).mean() for i in range(10000)]
mSample_300 = np.array(mSample_300)
print(f"Std : {mSample_300.std()}, Mean : {mSample_300.mean()}")
norm(loc =mSample_300.mean() , scale = mSample_300.std() ).interval(0.95)
```

```
Std : 291.88525124512284, Mean : 9263.752818666666
(8691.668238607801, 9835.83739872553)
```

```
fSample_300 = [um_customers["Purchase"].sample(300).mean() for i in range(10000)]
fSample_300 = np.array(fSample_300)
# mSample_300.mean(), mSample_300.std()
print(f"Std : {fSample_300.std()}, Mean : {fSample_300.mean()}")
norm(loc =fSample_300.mean() , scale = fSample_300.std() ).interval(0.95)
```

```
Std : 286.07461813729225, Mean : 9263.959605333332
(8703.26365689319, 9824.655553773473)
```

**Observation for 300 sample**

1. The average purchase made by a sample of 300 unmarried customers is 9263.75 with a standard deviation of 291.88

2. The average purchaese made by a sample of 300 married customers is 9263.95 with a standard deviation of 286.07
3. The interval estimate of 300 married sample is 8691.66 and 9835
4. The interval estimate of 300 unmarried sample is 8703.26 and 9824.65
5. 95 percent interval for married customer is tighter than female customers.

```
mSample_3000 = [m_customers["Purchase"].sample(3000).mean() for i in range(10000)]
mSample_3000 = np.array(mSample_3000)
print(f"Std : {mSample_3000.std()}, Mean : {mSample_3000.mean()}")
norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
```

```
Std : 90.89635831484134, Mean : 9261.113028433332
(9082.959439810395, 9439.26661705627)
```

```
fSample_3000 = [um_customers["Purchase"].sample(3000).mean() for i in range(10000)]
fSample_3000 = np.array(fSample_3000)
# mSample_300.mean(), mSample_300.std()
print(f"Std : {fSample_3000.std()}, Mean : {fSample_3000.mean()}")
norm(loc =fSample_3000.mean() , scale = fSample_3000.std() ).interval(0.95)
```

```
Std : 91.590540459542, Mean : 9265.255718333334
(9085.741557708072, 9444.769878958596)
```

**Observation for 3000 sample**

The average purchase made by a sample of 3000 married customers is 9261.11 with a standard deviation of 90.89

The average purchaese made by a sample of 3000 unmarried customers is 9265.25 with a standard deviation of 91.59

The interval estimate of 3000 unmarried sample is 9085.74 and 9444

The interval estimate of 3000 married sample is 9082.95 and 9439.26

95 percent interval for married customer is tighter than married customers.

```
mSample_30000 = [m_customers["Purchase"].sample(30000).mean() for i in range(10000)]
mSample_30000 = np.array(mSample_30000)
print(f"Std : {mSample_30000.std()}, Mean : {mSample_30000.mean()}")
norm(loc =mSample_30000.mean() , scale = mSample_30000.std() ).interval(0.95)
```

```
Std : 26.973739557122062, Mean : 9260.542687473335
(9207.675129413012, 9313.410245533658)
```

```
fSample_30000 = [um_customers["Purchase"].sample(30000).mean() for i in range(10000)]
fSample_30000 = np.array(fSample_30000)
# mSample_300.mean(), mSample_300.std()
print(f"Std : {fSample_30000.std()}, Mean : {fSample_30000.mean()}")
norm(loc =fSample_30000.mean() , scale = fSample_30000.std() ).interval(0.95)
```

```
Std : 28.11236774825027, Mean : 9266.23743965
(9211.138211343285, 9321.336667956715)
```

**Observation for 30000 sample**

The average purchase made by a sample of 30000 male customers is 9260.5 with a standard deviation of 26.97

The average purchaese made by a sample of 30000 unmarried customers is 8735.17 with a standard deviation of 24.38

The interval estimate of 30000 married sample is 9207.67 and 9313.41

The interval estimate of 30000 unmarried sample is 9211.13 and 9321.33

95 percent interval for male customer is tighter than female customers.

## Conclude the result and check if the confidence intervals of average married and unmarried spends are overlapping or not overlapping.

How can Walmart leverage this conclusion to make changes or improvements?

1. 95 percent confidence interval of Married and Unmarried spends are overlapping, it suggest that there may be a **statistically significant difference in the spending of Married and Unmarried.**

2. The average purchases made by Married is more than the average purchase made by umarried customers. Walmart might consider focusing on Married customers since they tend to spend more in comparision to unmarried customers.

**Actionalbe Insights:**

1. Introducing more Married people specific products.
2. Introducting some credit to Unmarried customers focus segment could be Unmarried working professionals. This could be in the form of credit cards espically to Young Bachelor Professionals.
3. Discount on Products on Valentines Day.

## Use the Central Limit Theorm to compute the interval. Change the sample size to observe the distribution of the mean of the expenses according to their age.

```
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 |

```
age_0_17 = df[df["Age"] == "0-17"]
age_18_25 = df[df["Age"] == "18-25"]
age_26_35 = df[df["Age"] == "26-35"]
age_36_45 = df[df["Age"] == "36-45"]
age_46_50 = df[df["Age"] == "46-50"]
age_51_55 = df[df["Age"] == "51-55"]
age_55plus = df[df["Age"] == "55+"]
```

```
mSample_300 = [age_0_17["Purchase"].sample(300).mean() for i in range(10000)]
mSample_300 = np.array(mSample_3000)
print("For the Age group :- 0-17 for 300 sample")
print(f"Total value count for Age 0-17 is : {age_0_17.shape[0]}")
print(f"Std : {mSample_3000.std()}, Mean : {mSample_3000.mean()}")
(a,b)= norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
print(f"The CI range for 300 sample for Age 0-17 is :{a} - {b}")
print("*************")
mSample_3000 = [age_0_17["Purchase"].sample(3000).mean() for i in range(10000)]
mSample_3000 = np.array(mSample_3000)
print("For the Age group :- 0-17 for 3000 sample")
print(f"Total value count for Age 0-17 is : {age_0_17.shape[0]}")
print(f"Std : {mSample_3000.std()}, Mean : {mSample_30000.mean()}")
(a,b)= norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
print(f"The CI range for 3000 sample for Age 0-17 is :{a} - {b}")
print("*************")
mSample_30000 = [age_0_17["Purchase"].sample(10000).mean() for i in range(10000)]
mSample_30000 = np.array(mSample_3000)
print("For the Age group :- 0-17 for 10000 sample")
print(f"Total value count for Age 0-17 is : {age_0_17.shape[0]}")
print(f"Std : {mSample_30000.std()}, Mean : {mSample_30000.mean()}")
(a,b)= norm(loc =mSample_30000.mean() , scale = mSample_30000.std() ).interval(0.95)
print(f"The CI range for 10000 sample for Age 0-17 is :{a} - {b}")
print("*************")
```

```
For the Age group :- 0-17 for 300 sample
Total value count for Age 0-17 is : 15102
Std : 82.83264411009274, Mean : 8934.173781399999
The CI range for 300 sample for Age 0-17 is :8771.824782199994 - 9096.522780600004
*************
For the Age group :- 0-17 for 3000 sample
Total value count for Age 0-17 is : 15102
Std : 82.75534404520766, Mean : 9260.542687473335
The CI range for 3000 sample for Age 0-17 is :8772.423573476502 - 9096.81856119016
*************
For the Age group :- 0-17 for 10000 sample
Total value count for Age 0-17 is : 15102
Std : 82.75534404520766, Mean : 8934.621067333332
```

```
The CI range for 10000 sample for Age 0-17 is :8772.423573476502 - 9096.81856119016
*************
```

```
mSample_300 = [age_18_25["Purchase"].sample(300).mean() for i in range(10000)]
mSample_300 = np.array(mSample_3000)
print("For the Age group :- 18-25 for 300 sample")
print(f"Total value count for Age 18-25 is : {age_0_17.shape[0]}")
print(f"Std : {mSample_3000.std()}, Mean : {mSample_3000.mean()}")
(a,b)= norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
print(f"The CI range for 300 sample for Age 18-25 is :{a} - {b}")
print("*************")
mSample_3000 = [age_18_25["Purchase"].sample(3000).mean() for i in range(10000)]
mSample_3000 = np.array(mSample_3000)
print("For the Age group :- 18-25 for 3000 sample")
print(f"Total value count for Age 18-25 is : {age_0_17.shape[0]}")
print(f"Std : {mSample_3000.std()}, Mean : {mSample_30000.mean()}")
(a,b)= norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
print(f"The CI range for 3000 sample for Age 18-25 is :{a} - {b}")
print("*************")
mSample_30000 = [age_18_25["Purchase"].sample(10000).mean() for i in range(10000)]
mSample_30000 = np.array(mSample_3000)
print("For the Age group :- 18-25 for 10000 sample")
print(f"Total value count for Age 18-25 is : {age_0_17.shape[0]}")
print(f"Std : {mSample_30000.std()}, Mean : {mSample_30000.mean()}")
(a,b)= norm(loc =mSample_30000.mean() , scale = mSample_30000.std() ).interval(0.95)
print(f"The CI range for 10000 sample for Age 18-25 is :{a} - {b}")
print("*************")
```

```
For the Age group :- 18-25 for 300 sample
Total value count for Age 18-25 is : 15102
Std : 90.57332066540845, Mean : 9252.087014333334
The CI range for 300 sample for Age 18-25 is :9074.566567868937 - 9429.607460797732
*************
For the Age group :- 18-25 for 3000 sample
Total value count for Age 18-25 is : 15102
Std : 89.83609172444022, Mean : 9252.087014333334
The CI range for 3000 sample for Age 18-25 is :8993.46839577493 - 9345.619404358407
*************
For the Age group :- 18-25 for 10000 sample
Total value count for Age 18-25 is : 15102
Std : 89.83609172444022, Mean : 9169.543900066668
The CI range for 10000 sample for Age 18-25 is :8993.46839577493 - 9345.619404358407
*************
```

```
mSample_300 = [age_26_35["Purchase"].sample(300).mean() for i in range(10000)]
mSample_300 = np.array(mSample_3000)
print("For the Age group :- 26-35 for 300 sample")
print(f"Total value count for Age 26-35 is : {age_26_35.shape[0]}")
print(f"Std : {mSample_3000.std()}, Mean : {mSample_3000.mean()}")
(a,b)= norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
print(f"The CI range for 300 sample for Age 26-35 is :{a} - {b}")
print("*************")
mSample_3000 = [age_26_35["Purchase"].sample(3000).mean() for i in range(10000)]
mSample_3000 = np.array(mSample_3000)
print("For the Age group :- 26-35 for 3000 sample")
print(f"Total value count for Age 26-35 is : {age_26_35.shape[0]}")
print(f"Std : {mSample_3000.std()}, Mean : {mSample_30000.mean()}")
(a,b)= norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
print(f"The CI range for 3000 sample for Age 26-35 is :{a} - {b}")
print("*************")
mSample_30000 = [age_26_35["Purchase"].sample(10000).mean() for i in range(10000)]
mSample_30000 = np.array(mSample_3000)
print("For the Age group :- 26-35 for 10000 sample")
print(f"Total value count for Age 26-35 is : {age_26_35.shape[0]}")
print(f"Std : {mSample_30000.std()}, Mean : {mSample_30000.mean()}")
(a,b)= norm(loc =mSample_30000.mean() , scale = mSample_30000.std() ).interval(0.95)
print(f"The CI range for 10000 sample for Age 26-35 is :{a} - {b}")
print("*************")
```

```
For the Age group :- 26-35 for 300 sample
Total value count for Age 26-35 is : 15102
Std : 90.35376868572851, Mean : 9170.165004733333
The CI range for 300 sample for Age 26-35 is :8993.074872241843 - 9347.255137224824
*************
For the Age group :- 26-35 for 3000 sample
Total value count for Age 26-35 is : 15102
Std : 90.57332066540845, Mean : 9170.165004733333
The CI range for 3000 sample for Age 26-35 is :9074.566567868937 - 9429.607460797732
*************
```

```
For the Age group :- 26-35 for 10000 sample
Total value count for Age 26-35 is : 15102
Std : 90.57332066540845, Mean : 9252.087014333334
The CI range for 10000 sample for Age 26-35 is :9074.566567868937 - 9429.607460797732
************
```

```
mSample_300 = [age_36_45["Purchase"].sample(300).mean() for i in range(10000)]
mSample_300 = np.array(mSample_3000)
print("For the Age group :- 36-45 for 300 sample")
print(f"Total value count for Age 36-45 is : {age_36_45.shape[0]}")
print(f"Std : {mSample_3000.std()}, Mean : {mSample_3000.mean()}")
(a,b)= norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
print(f"The CI range for 300 sample for Age 36-45 is :{a} - {b}")
print("**************")
mSample_3000 = [age_36_45["Purchase"].sample(3000).mean() for i in range(10000)]
mSample_3000 = np.array(mSample_3000)
print("For the Age group :- 36-45 for 3000 sample")
print(f"Total value count for Age 36-45 is : {age_36_45.shape[0]}")
print(f"Std : {mSample_3000.std()}, Mean : {mSample_30000.mean()}")
(a,b)= norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
print(f"The CI range for 3000 sample for Age 36-45 is :{a} - {b}")
print("**************")
mSample_30000 = [age_36_45["Purchase"].sample(10000).mean() for i in range(10000)]
mSample_30000 = np.array(mSample_3000)
print("For the Age group :- 36-45 for 10000 sample")
print(f"Total value count for Age 36-45 is : {age_36_45.shape[0]}")
print(f"Std : {mSample_30000.std()}, Mean : {mSample_30000.mean()}")
(a,b)= norm(loc =mSample_30000.mean() , scale = mSample_30000.std() ).interval(0.95)
print(f"The CI range for 10000 sample for Age 36-45 is :{a} - {b}")
print("**************")
```

```
For the Age group :- 36-45 for 300 sample
Total value count for Age 36-45 is : 110013
Std : 89.83609172444022, Mean : 9169.543900066668
The CI range for 300 sample for Age 36-45 is :8993.46839577493 - 9345.619404358407
************
For the Age group :- 36-45 for 3000 sample
Total value count for Age 36-45 is : 110013
Std : 90.71376750222441, Mean : 9169.543900066668
The CI range for 3000 sample for Age 36-45 is :9151.9249376937 - 9507.5163721063
************
For the Age group :- 36-45 for 10000 sample
Total value count for Age 36-45 is : 110013
Std : 90.71376750222441, Mean : 9329.7206549
The CI range for 10000 sample for Age 36-45 is :9151.9249376937 - 9507.5163721063
************
```

```
mSample_300 = [age_46_50["Purchase"].sample(300).mean() for i in range(10000)]
mSample_300 = np.array(mSample_3000)
print("For the Age group :- 46-50 for 300 sample")
print(f"Total value count for Age 46-50 is : {age_46_50.shape[0]}")
print(f"Std : {mSample_3000.std()}, Mean : {mSample_3000.mean()}")
(a,b)= norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
print(f"The CI range for 300 sample for Age 46-50 is :{a} - {b}")
print("**************")
mSample_3000 = [age_46_50["Purchase"].sample(3000).mean() for i in range(10000)]
mSample_3000 = np.array(mSample_3000)
print("For the Age group :- 46-50 for 3000 sample")
print(f"Total value count for Age 46-50 is : {age_46_50.shape[0]}")
print(f"Std : {mSample_3000.std()}, Mean : {mSample_30000.mean()}")
(a,b)= norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
print(f"The CI range for 3000 sample for Age 46-50 is :{a} - {b}")
print("**************")
mSample_30000 = [age_46_50["Purchase"].sample(10000).mean() for i in range(10000)]
mSample_30000 = np.array(mSample_3000)
print("For the Age group :- 46-50 for 10000 sample")
print(f"Total value count for Age 46-50 is : {age_46_50.shape[0]}")
print(f"Std : {mSample_30000.std()}, Mean : {mSample_30000.mean()}")
(a,b)= norm(loc =mSample_30000.mean() , scale = mSample_30000.std() ).interval(0.95)
print(f"The CI range for 10000 sample for Age 46-50 is :{a} - {b}")
print("**************")
```

```
For the Age group :- 46-50 for 300 sample
Total value count for Age 46-50 is : 45701
```

```
Std : 90.71376750222441, Mean : 9329.7206549
The CI range for 300 sample for Age 46-50 is :9151.9249376937 - 9507.5163721063
*************
For the Age group :- 46-50 for 3000 sample
Total value count for Age 46-50 is : 45701
Std : 86.72442480567042, Mean : 9329.7206549
The CI range for 3000 sample for Age 46-50 is :9038.148294034267 - 9378.101792432399
*************
For the Age group :- 46-50 for 10000 sample
Total value count for Age 46-50 is : 45701
Std : 86.72442480567042, Mean : 9208.125043233333
The CI range for 10000 sample for Age 46-50 is :9038.148294034267 - 9378.101792432399
*************
```

```
mSample_300 = [age_51_55["Purchase"].sample(300).mean() for i in range(10000)]
mSample_300 = np.array(mSample_3000)
print("For the Age group :- 51-55 for 300 sample")
print(f"Total value count for Age 51-55 is : {age_51_55.shape[0]}")
print(f"Std : {mSample_3000.std()}, Mean : {mSample_3000.mean()}")
(a,b)= norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
print(f"The CI range for 300 sample for Age 51-55 is :{a} - {b}")
print("*************")
mSample_3000 = [age_51_55["Purchase"].sample(3000).mean() for i in range(10000)]
mSample_3000 = np.array(mSample_3000)
print("For the Age group :- 51-55 for 3000 sample")
print(f"Total value count for Age 51-55 is : {age_51_55.shape[0]}")
print(f"Std : {mSample_3000.std()}, Mean : {mSample_30000.mean()}")
(a,b)= norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
print(f"The CI range for 3000 sample for Age 51-55 is :{a} - {b}")
print("*************")
mSample_30000 = [age_51_55["Purchase"].sample(10000).mean() for i in range(10000)]
mSample_30000 = np.array(mSample_3000)
print("For the Age group :- 51-55 for 10000 sample")
print(f"Total value count for Age 51-55 is : {age_51_55.shape[0]}")
print(f"Std : {mSample_30000.std()}, Mean : {mSample_30000.mean()}")
(a,b)= norm(loc =mSample_30000.mean() , scale = mSample_30000.std() ).interval(0.95)
print(f"The CI range for 10000 sample for Age 51-55 is :{a} - {b}")
print("*************")
```

```
For the Age group :- 51-55 for 300 sample
Total value count for Age 51-55 is : 38501
Std : 86.72442480567042, Mean : 9208.125043233333
The CI range for 300 sample for Age 51-55 is :9038.148294034267 - 9378.101792432399
*************
For the Age group :- 51-55 for 3000 sample
Total value count for Age 51-55 is : 38501
Std : 90.69769311446387, Mean : 9208.125043233333
The CI range for 3000 sample for Age 51-55 is :9356.125826614785 - 9711.654250585216
*************
For the Age group :- 51-55 for 10000 sample
Total value count for Age 51-55 is : 38501
Std : 90.69769311446387, Mean : 9533.8900386
The CI range for 10000 sample for Age 51-55 is :9356.125826614785 - 9711.654250585216
*************
```

```
mSample_300 = [age_55plus["Purchase"].sample(300).mean() for i in range(10000)]
mSample_300 = np.array(mSample_3000)
print("For the Age group :- 55plus for 300 sample")
print(f"Total value count for Age 55plus is : {age_55plus.shape[0]}")
print(f"Std : {mSample_3000.std()}, Mean : {mSample_3000.mean()}")
(a,b)= norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
print(f"The CI range for 300 sample for Age 55plus is :{a} - {b}")
print("*************")
mSample_3000 = [age_55plus["Purchase"].sample(3000).mean() for i in range(10000)]
mSample_3000 = np.array(mSample_3000)
print("For the Age group :- 55plus for 3000 sample")
print(f"Total value count for Age 55plus is : {age_55plus.shape[0]}")
print(f"Std : {mSample_3000.std()}, Mean : {mSample_30000.mean()}")
(a,b)= norm(loc =mSample_3000.mean() , scale = mSample_3000.std() ).interval(0.95)
print(f"The CI range for 3000 sample for Age 55plus is :{a} - {b}")
print("*************")
mSample_30000 = [age_55plus["Purchase"].sample(10000).mean() for i in range(10000)]
mSample_30000 = np.array(mSample_3000)
print("For the Age group :- 55plus for 10000 sample")
print(f"Total value count for Age 55plus is : {age_55plus.shape[0]}")
print(f"Std : {mSample_30000.std()}, Mean : {mSample_30000.mean()}")
(a,b)= norm(loc =mSample_30000.mean() , scale = mSample_30000.std() ).interval(0.95)
print(f"The CI range for 10000 sample for Age 55plus is :{a} - {b}")
print("*************")
```

```
For the Age group :- 55plus for 300 sample
Total value count for Age 55plus is : 21504
Std : 90.69769311446387, Mean : 9533.8900386
The CI range for 300 sample for Age 55plus is :9356.125826614785 - 9711.654250585216
*************
For the Age group :- 55plus for 3000 sample
Total value count for Age 55plus is : 21504
Std : 85.17430019688379, Mean : 9533.8900386
The CI range for 3000 sample for Age 55plus is :9168.616348472371 - 9502.493470060961
*************
For the Age group :- 55plus for 10000 sample
Total value count for Age 55plus is : 21504
Std : 85.17430019688379, Mean : 9335.554909266666
The CI range for 10000 sample for Age 55plus is :9168.616348472371 - 9502.493470060961
*************
```

## Observation

For the Age group :- 0-17 for 300 sample Total value count for Age 0-17 is : 15102 Std : 82.83264411009274, Mean : 8934.173781399999 The CI range for 300 sample for Age 0-17 is :8771.824782199994 - 9096.522780600004

---

For the Age group :- 0-17 for 3000 sample Total value count for Age 0-17 is : 15102 Std : 82.75534404520766, Mean : 9260.542687473335 The CI range for 3000 sample for Age 0-17 is :8772.423573476502 - 9096.81856119016

---

For the Age group :- 0-17 for 10000 sample Total value count for Age 0-17 is : 15102 Std : 82.75534404520766, Mean : 8934.621067333332 The CI range for 10000 sample for Age 0-17 is :8772.423573476502 - 9096.81856119016

For the Age group :- 18-25 for 300 sample Total value count for Age 18-25 is : 15102 Std : 90.57332066540845, Mean : 9252.087014333334 The CI range for 300 sample for Age 18-25 is :9074.566567868937 - 9429.607460797732

---

For the Age group :- 18-25 for 3000 sample Total value count for Age 18-25 is : 15102 Std : 89.83609172444022, Mean : 9252.087014333334 The CI range for 3000 sample for Age 18-25 is :8993.46839577493 - 9345.619404358407

---

For the Age group :- 18-25 for 10000 sample Total value count for Age 18-25 is : 15102 Std : 89.83609172444022, Mean : 9169.543900066668 The CI range for 10000 sample for Age 18-25 is :8993.46839577493 - 9345.619404358407

For the Age group :- 26-35 for 300 sample Total value count for Age 26-35 is : 15102 Std : 90.35376868572851, Mean : 9170.165004733333 The CI range for 300 sample for Age 26-35 is :8993.074872241843 - 9347.255137224824

---

For the Age group :- 26-35 for 3000 sample Total value count for Age 26-35 is : 15102 Std : 90.57332066540845, Mean : 9170.165004733333 The CI range for 3000 sample for Age 26-35 is :9074.566567868937 - 9429.607460797732

---

For the Age group :- 26-35 for 10000 sample Total value count for Age 26-35 is : 15102 Std : 90.57332066540845, Mean : 9252.087014333334 The CI range for 10000 sample for Age 26-35 is :9074.566567868937 - 9429.607460797732

For the Age group :- 36-45 for 300 sample Total value count for Age 36-45 is : 110013 Std : 89.83609172444022, Mean : 9169.543900066668 The CI range for 300 sample for Age 36-45 is :8993.46839577493 - 9345.619404358407

---

For the Age group :- 36-45 for 3000 sample Total value count for Age 36-45 is : 110013 Std : 90.71376750222441, Mean : 9169.543900066668 The CI range for 3000 sample for Age 36-45 is :9151.9249376937 - 9507.5163721063

---

For the Age group :- 36-45 for 10000 sample Total value count for Age 36-45 is : 110013 Std : 90.71376750222441, Mean : 9329.7206549 The CI range for 10000 sample for Age 36-45 is :9151.9249376937 - 9507.5163721063

For the Age group :- 46-50 for 300 sample Total value count for Age 46-50 is : 45701 Std : 90.71376750222441, Mean : 9329.7206549 The CI range for 300 sample for Age 46-50 is :9151.9249376937 - 9507.5163721063

---

For the Age group :- 46-50 for 3000 sample Total value count for Age 46-50 is : 45701 Std : 86.72442480567042, Mean : 9329.7206549 The CI range for 3000 sample for Age 46-50 is :9038.148294034267 - 9378.101792432399

---

For the Age group :- 46-50 for 10000 sample Total value count for Age 46-50 is : 45701 Std : 86.72442480567042, Mean : 9208.125043233333 The CI range for 10000 sample for Age 46-50 is :9038.148294034267 - 9378.101792432399

For the Age group :- 51-55 for 300 sample Total value count for Age 51-55 is : 38501 Std : 86.72442480567042, Mean : 9208.125043233333 The CI range for 300 sample for Age 51-55 is :9038.148294034267 - 9378.101792432399

---

For the Age group :- 51-55 for 3000 sample Total value count for Age 51-55 is : 38501 Std : 90.69769311446387, Mean : 9208.125043233333 The CI range for 3000 sample for Age 51-55 is :9356.125826614785 - 9711.654250585216

---

For the Age group :- 51-55 for 10000 sample Total value count for Age 51-55 is : 38501 Std : 90.69769311446387, Mean : 9533.8900386 The CI range for 10000 sample for Age 51-55 is :9356.125826614785 - 9711.654250585216

For the Age group :- 55plus for 300 sample Total value count for Age 55plus is : 21504 Std : 90.69769311446387, Mean : 9533.8900386 The CI range for 300 sample for Age 55plus is :9356.125826614785 - 9711.654250585216

---

For the Age group :- 55plus for 3000 sample Total value count for Age 55plus is : 21504 Std : 85.17430019688379, Mean : 9533.8900386 The CI range for 3000 sample for Age 55plus is :9168.616348472371 - 9502.493470060961

---

For the Age group :- 55plus for 10000 sample Total value count for Age 55plus is : 21504 Std : 85.17430019688379, Mean : 9335.554909266666 The CI range for 10000 sample for Age 55plus is :9168.616348472371 - 9502.493470060961

95 % Confidence Interval of average 26-35 and 36-45 spends are not overalpping, it suggest that there might be statistically significant difference.

The average purchase made by married customer are more as compared to average purchase made by unmarried customers. Walmart mayconcentrating on customers who belongs to age group 26-35 and 36-45, as they exhibit higher spending patterns compared to other age groups.

## Actionable Insights

1. The average purchase made by married customer are more as compared to average purchase made by unmarried customers. Walmart mayconcentrating on customers who belongs to age group 26-35 and 36-45, as they exhibit higher spending patterns compared to other age groups.
2. Introducing more Married people specific products.
3. Introducting some credit to Unmarried customers focus segment could be Unmarried working professionals. This could be in the form of credit cards espically to Young Bachelor Professionals.
4. Discount on Products on Valentines Day.
5. Target Male Shoppers Since male customers account for a significant portion of Black Friday sales and tend to spend more per transaction on average, Walmart should tailor its marketing strategies and product offerings to incentivize higher spending among male customers while ensuring competitive pricing for female-oriented products.
6. Focus on 26 - 45 Age Group With the age group between 26 and 45 contributing to the majority of sales, Walmart should specifically cater to the preferences and needs of this demographic.This could include offering exclusive deals on products that are popular among this age group.
7. Engage Younger Shoppers Knowing that customers in the 0 - 17 age group have the lowest spending per transaction, Walmart can try to increase their spending per transaction by offering them more attractive discounts, coupons, or rewards programs. It's essential to start

building brand loyalty among younger consumers.

8. Customer Segmentation Since customers in the 18 - 25, 26 - 35, and 46 - 50 age groups exhibit similar buying characteristics, and so do the customers in 36 - 45 and 55+, Walmart can optimize its product selection to cater to the preferences of these age groups. Also, Walmart can use this information to adjust their pricing strategies for different age groups.
9. Enhance the 51 - 55 Age Group Shopping Experience Considering that customers aged 51 - 55 have the highest spending per transaction, Walmart offer them exclusive pre-sale access, special discount or provide personalized product recommendations for this age group. Walmart can also introduce loyalty programs specifically designed to reward and retain customers in the 51 - 55 age group.
10. Post-Black Friday Engagement After Black Friday, walmart should engage with customers who made purchases by sending follow-up emails or offers for related products. This can help increase customer retention and encourage repeat business throughout the holiday season and beyond.

```
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 |

Actionable Insights The average purchase made by married customer are more as compared to average purchase made by unmarried customers. Walmart mayconcentrating on customers who belongs to age group 26-35 and 36-45, as they exhibit higher spending patterns compared to other age groups.

Introducing more Married people specific products.

Introducting some credit to Unmarried customers focus segment could be Unmarried working professionals. This could be in the form of credit cards espically to Young Bachelor Professionals.

Discount on Products on Valentines Day.

Target Male Shoppers Since male customers account for a significant portion of Black Friday sales and tend to spend more per transaction on average, Walmart should tailor its marketing strategies and product offerings to incentivize higher spending among male customers while ensuring competitive pricing for female-oriented products.

Focus on 26 - 45 Age Group With the age group between 26 and 45 contributing to the majority of sales, Walmart should specifically cater to the preferences and needs of this demographic.This could include offering exclusive deals on products that are popular among this age group.

Engage Younger Shoppers Knowing that customers in the 0 - 17 age group have the lowest spending per transaction, Walmart can try to increase their spending per transaction by offering them more attractive discounts, coupons, or rewards programs. It's essential to start building brand loyalty among younger consumers.

Customer Segmentation Since customers in the 18 - 25, 26 - 35, and 46 - 50 age groups exhibit similar buying characteristics, and so do the customers in 36 - 45 and 55+, Walmart can optimize its product selection to cater to the preferences of these age groups. Also, Walmart can use this information to adjust their pricing strategies for different age groups.

Enhance the 51 - 55 Age Group Shopping Experience Considering that customers aged 51 - 55 have the highest spending per transaction, Walmart offer them exclusive pre-sale access, special discount or provide personalized product recommendations for this age group. Walmart can also introduce loyalty programs specifically designed to reward and retain customers in the 51 - 55 age group.

Post-Black Friday Engagement After Black Friday, walmart should engage with customers who made purchases by sending follow-up emails or offers for related products. This can help increase customer retention and encourage repeat business throughout the holiday season and beyond.

```
df.head()
```