ABSTRACT

Walmart, Inc. (WMT) is an American multinational discount store operator and one of the largest corporations in the global retail industry. It is founded in 1962 by Sam Walton, it has more than 12000 stores in 28 countries. Its company headquarters is located in Bentonville, Arkansas.

Walmart operates a vast network of hypermarkets, discount department stores, and grocery stores under various brand names across the United States and in numerous countries around the world. It provides services like Wal-Mart to Wal-Mart, Wal-Mart Pay, Wal-Mart Money Card which makes easier and comfortable for the user point of view. Known for its "Everyday Low Prices" strategy, Walmart has redefined the retail landscape with its commitment to offering a wide range of products at affordable prices.

BUSINESS PROBLEM-

The Management team at Walmart Inc. wants to analyze the customer purchase behaviour (specifically, purchase amount) against the customer's gender and the various other factors to help the business make better decisions. They want to understand if the spending habits differ between male and female customers: Do women spend more on Black Friday than men? (Assume 50 million customers are male and 50 million are female).

EXPLORATORY DATA ANALYSIS-

Import libraries:-

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import t
```

Loading the dataset:--

```
df = pd.read_csv('a.csv')
```

```
df
```

Let's check the first five data :-

```
df.head(5)
```

	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

```
df.shape
```

```
(550068, 10)
```

To get all attributes:-

```
df.columns
```

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

Data types of all the attributes:-

```
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
```

🔍 **Insights-**

- From the above analysis, it is clear that data has total of 10 features with lots of mixed alpha numeric data.

- Apart from Purchase Column, all the other data types are of categorical type. We will change the datatypes of all such columns to category.

Let’s change the datatype of columns:-

```
for i in df.columns[:-1]:
    df[i] = df[i].astype('category')

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  category
 1   Product_ID                  550068 non-null  category
 2   Gender                      550068 non-null  category
 3   Age                         550068 non-null  category
 4   Occupation                  550068 non-null  category
 5   City_Category               550068 non-null  category
 6   Stay_In_Current_City_Years  550068 non-null  category
 7   Marital_Status              550068 non-null  category
 8   Product_Category            550068 non-null  category
 9   Purchase                    550068 non-null  int64
dtypes: category(9), int64(1)
memory usage: 10.3 MB
```

📝 Statistical Summary of object type columns:-

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

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category
count	550068	550068	550068	550068	550068	550068	550068	550068	550068
unique	5891	3631	2	7	21	3	5	2	20
top	1001680	P00265242	M	26-35	4	B	1	0	5
freq	1026	1880	414259	219587	72308	231173	193821	324731	150933

🔍 Insights-

- Among 550068 transactions there are 5891 unique User_ID.
- Among 5,50,068 transactions there are 3631 unique products,with the product having the code P00265242 being the highest seller, with a maximum of 1,880 units sold.
- Out of 5,50,068 transactions, 4,14,259 (nearly 75%) were done by male gender indicating a significant disparity in purchase behaviour between males and females during the Black Friday.
- We have 7 unique age groups in the dataset. 26 - 35 Age group has maximum of 2,19,587 transactions.
- Customers with 1 year of stay in current city accounted to maximum of 1,93,821 transactions.

Statistical summary of numerical data type columns:-

```
df.describe()
```

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

Identify the missing values:-

```
print("columns with missing values-")
print(df.isnull().any())
```

```
columns with missing values-
User_ID                       False
Product_ID                    False
Gender                        False
Age                           False
Occupation                    False
City_Category                 False
Stay_In_Current_City_Years    False
Marital_Status                False
Product_Category              False
Purchase                      False
dtype: bool
```

🔍 **Insights-**

- There are no null values in the dataset.

👥 Duplicate Detection:-

```
df.duplicated().value_counts()
```

```
False    550068
Name: count, dtype: int64
```

🔍 Insights-

- There are no duplicate entries in the dataset.

Unique values:-

```
# checking the unique values for columns
for i in df.columns:
    print('Unique Values in',i,'column are :-')
    print(df[i].unique())
    print('-'*90)
```

```
Unique Values in User_ID column are :-
[1000001, 1000002, 1000003, 1000004, 1000005, ..., 1004588, 1004871, 1004113, 1005391, 1001529]
Length: 5891
Categories (5891, int64): [1000001, 1000002, 1000003, 1000004, ..., 1006037, 1006038, 1006039, 1006040]
------------------------------------------------------------------------------------------
Unique Values in Product_ID column are :-
['P00069042', 'P00248942', 'P00087842', 'P00085442', 'P00285442', ..., 'P00375436', 'P00372445', 'P00370293', 'P00371644', 'P00370853']
Length: 3631
Categories (3631, object): ['P00000142', 'P00000242', 'P00000342', 'P00000442', ..., 'P0099642',
                            'P0099742', 'P0099842', 'P0099942']
------------------------------------------------------------------------------------------
Unique Values in Gender column are :-
['F', 'M']
Categories (2, object): ['F', 'M']
------------------------------------------------------------------------------------------
Unique Values in Age column are :-
['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25']
Categories (7, object): ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
------------------------------------------------------------------------------------------
Unique Values in Occupation column are :-
[10, 16, 15, 7, 20, ..., 18, 5, 14, 13, 6]
Length: 21
Categories (21, int64): [0, 1, 2, 3, ..., 17, 18, 19, 20]
------------------------------------------------------------------------------------------
Unique Values in City_Category column are :-
['A', 'C', 'B']
Categories (3, object): ['A', 'B', 'C']
------------------------------------------------------------------------------------------
Unique Values in Stay_In_Current_City_Years column are :-
['2', '4+', '3', '1', '0']
Categories (5, object): ['0', '1', '2', '3', '4+']
------------------------------------------------------------------------------------------
Unique Values in Marital_Status column are :-
[0, 1]
Categories (2, int64): [0, 1]
------------------------------------------------------------------------------------------
Unique Values in Product_Category column are :-
[3, 1, 12, 8, 5, ..., 10, 17, 9, 20, 19]
Length: 20
Categories (20, int64): [1, 2, 3, 4, ..., 17, 18, 19, 20]
------------------------------------------------------------------------------------------
Unique Values in Purchase column are :-
[ 8370 15200  1422 ...   135   123   613]
------------------------------------------------------------------------------------------
```

🔍 Insights-

- The dataset does not contain any abnormal values.
- We will convert the 0,1 in Marital Status column as married and unmarried.

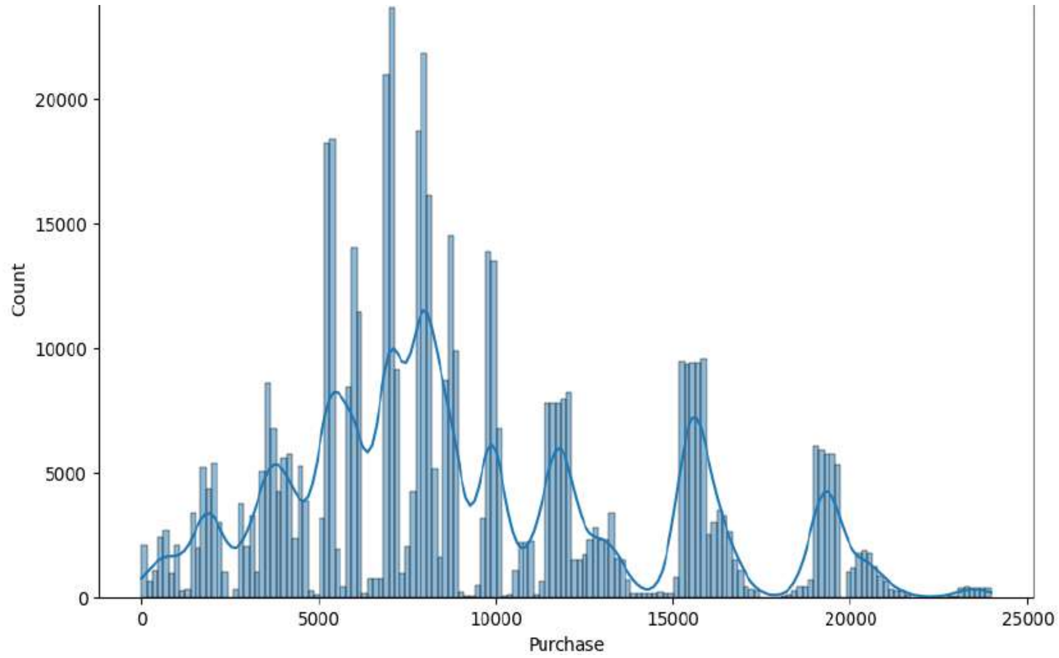
```
df['Marital_Status'] = df['Marital_Status'].replace({0:'Unmarried',1:'Married'})
df['Marital_Status'].unique()
```

```
['Unmarried', 'Married']
Categories (2, object): ['Unmarried', 'Married']
```

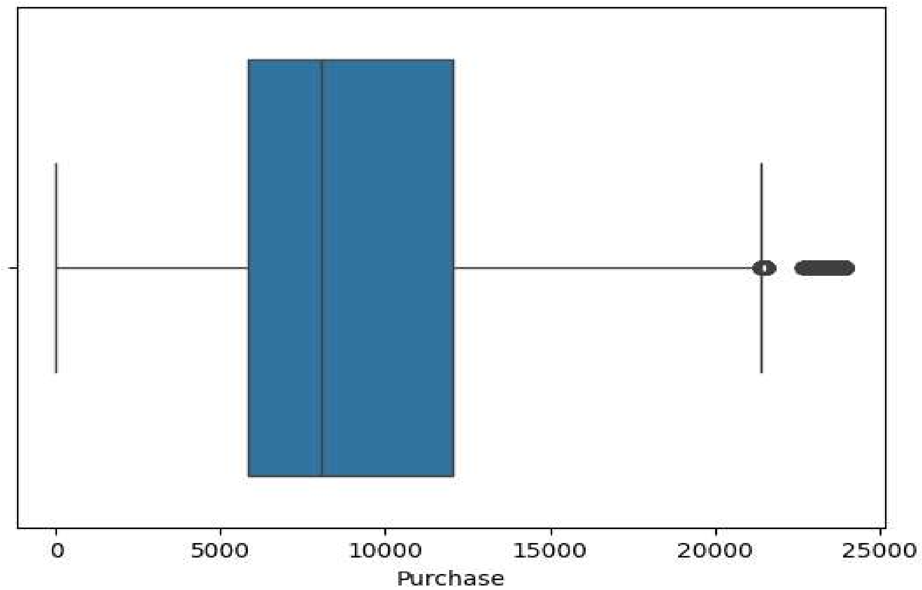
UNIVARIATE ANALYSIS-

Understanding the distribution of data and detecting outlies for continuous variables-

```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Purchase', kde=True)
plt.show()
```




```
sns.boxplot(data=df, x='Purchase', orient='h')
plt.show()
```

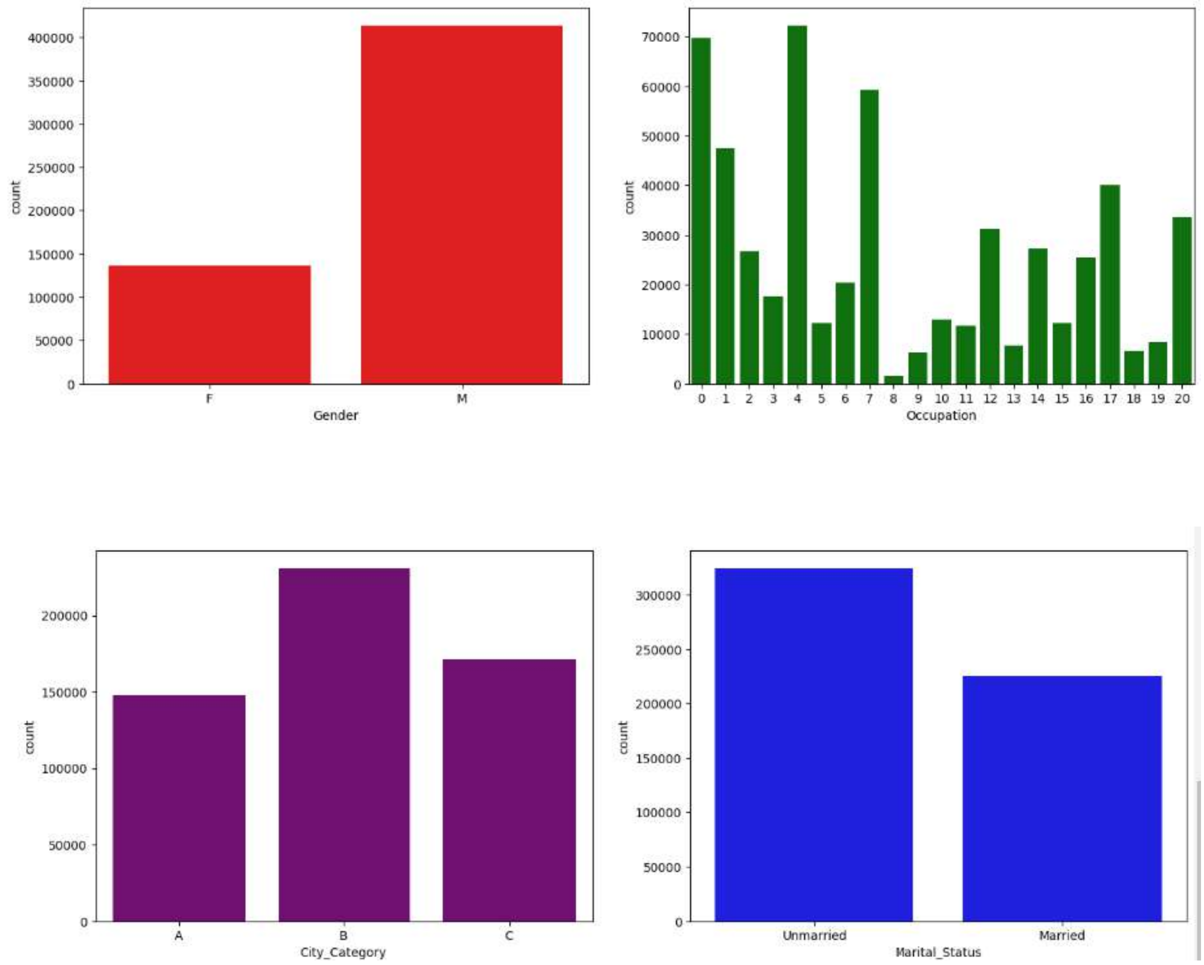


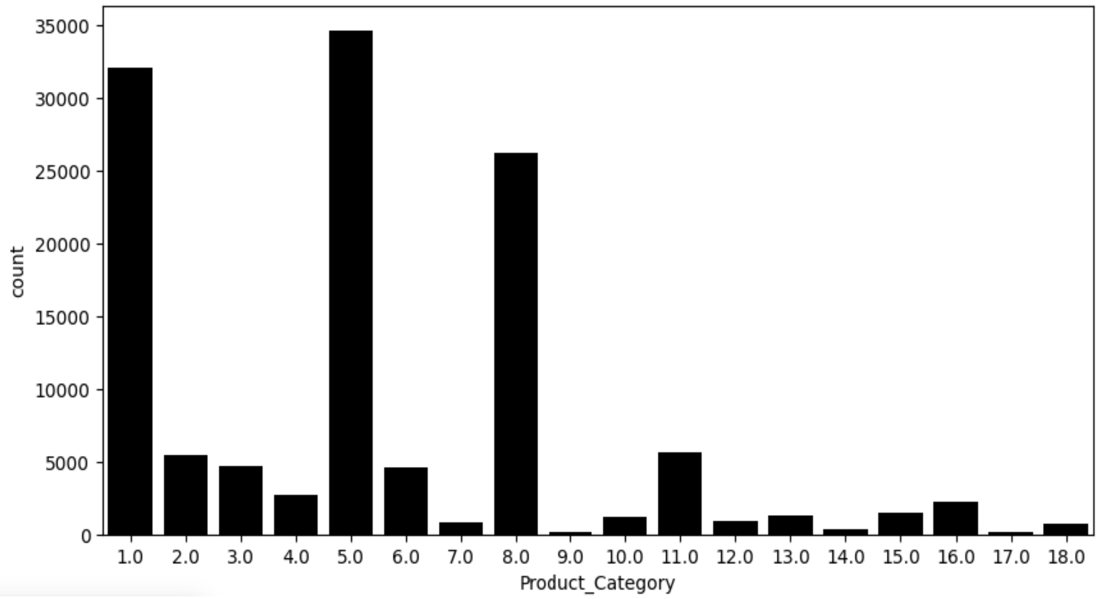
🔍 **Insights-**

- Purchase is having outliers.

Understanding the distribution of data for the categorical variables-

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

fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(16, 12))
sns.countplot(data=df, x='Gender',ax=axs[0,0],color = 'red')
sns.countplot(data=df, x='Occupation', ax=axs[0,1],color = 'green')
sns.countplot(data=df, x='City_Category', ax=axs[1,0],color = 'purple')
sns.countplot(data=df, x='Marital_Status', ax=axs[1,1],color = 'blue')
plt.show()

plt.figure(figsize=(10, 8))
sns.countplot(data=df, x='Product_Category',color = 'black')
plt.show()
```



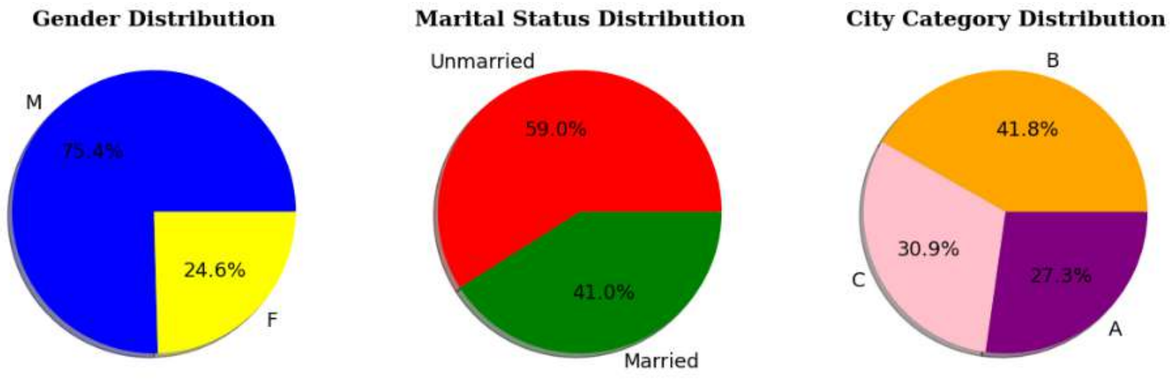


🔍 **Insights-**

- Most of the users are male.
- There are 20 different types of Occupation.
- More users belong to B City_Category.
- More users are Single as compared to Married.
- Product_Category - 1, 5 & 8 have highest purchasing frequency.

👩🏻 👨🏻 Gender, 👫 Marital Status and 🌍 City Category Distribution-

```
fig = plt.figure(figsize = (14,12))
gs = fig.add_gridspec(1,3)
a = fig.add_subplot(gs[0,0])
color_map = ['blue','yellow']
a.pie(df['Gender'].value_counts().values,labels = df['Gender'].value_counts().index,autopct = '%.1f%%',
        shadow = True,colors = color_map,textprops={'fontsize': 13, 'color': 'black'})
a.set_title('Gender Distribution',{'font':'serif', 'size':14,'weight':'bold'})

                                        # creating pie chart for marital status
b = fig.add_subplot(gs[0,1])
color_map = ['red','green']
b.pie(df['Marital_Status'].value_counts().values,labels = df['Marital_Status'].value_counts().index,autopct = '%.1f%%',
        shadow = True,colors = color_map,textprops={'fontsize': 13, 'color': 'black'})
b.set_title('Marital Status Distribution',{'font':'serif', 'size':14,'weight':'bold'})

                                        # creating pie chart for city category
c = fig.add_subplot(gs[0,2])
color_map = ['orange','pink','purple']
c.pie(df['City_Category'].value_counts().values,labels = df['City_Category'].value_counts().index,autopct = '%.1f%%',
        shadow = True,colors = color_map,textprops={'fontsize': 13, 'color': 'black'})
c.set_title('City Category Distribution',{'font':'serif', 'size':14,'weight':'bold'})
plt.show()
```

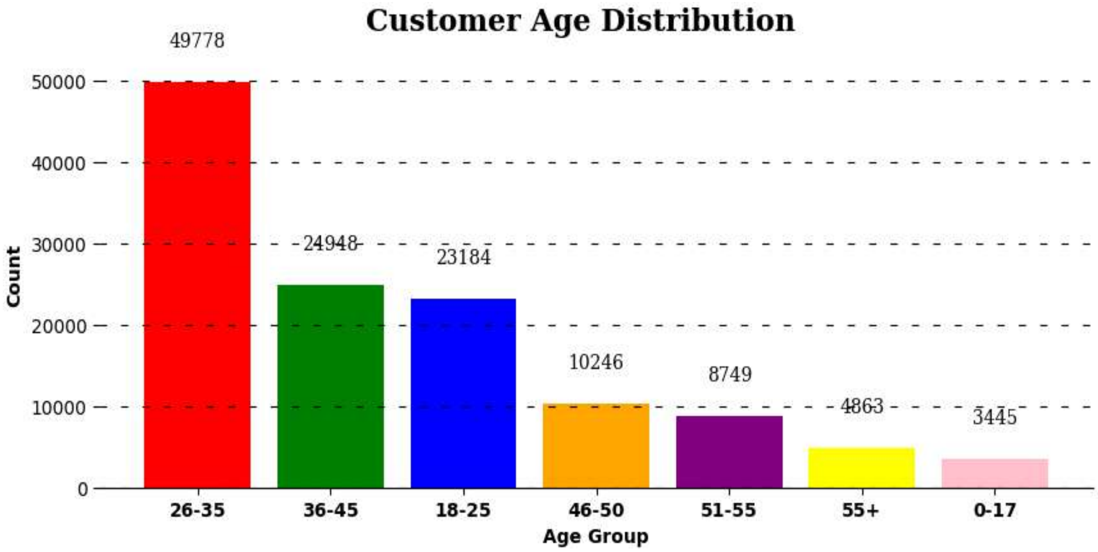


🔍 **Insights-**

- There is significant disparity in purchase behaviour between males and females during the Black Friday.
- Unmarried customers account for a higher percentage of transactions.
- Most number of transactions followed by City C and City A respectively- City B

📅 <u>Customer Age Distribution-</u>

```
fig,a = plt.subplots(figsize = (10,4))
a0= df['Age'].value_counts()
color_map = ['red','green','blue','orange','purple','yellow','pink']
a.bar(x=a0.index,height = a0.values,color = color_map)
for i in a0.index:
   a.text(i,a0[i]+5000,a0[i],{'font':'serif','size' : 10},ha = 'center',va = 'center')
a.grid(color = 'black',linestyle = '--', axis = 'y', dashes = (5,10))
for s in ['top','left','right']:
    a.spines[s].set_visible(False)
a.set_ylabel('Count',fontweight = 'bold',fontsize = 10)
a.set_xlabel('Age Group',fontweight = 'bold',fontsize = 10)
a.set_xticklabels(a0.index,fontweight = 'bold')
fig.suptitle('Customer Age Distribution',font = 'serif', size = 16, weight = 'bold')
plt.show()
```



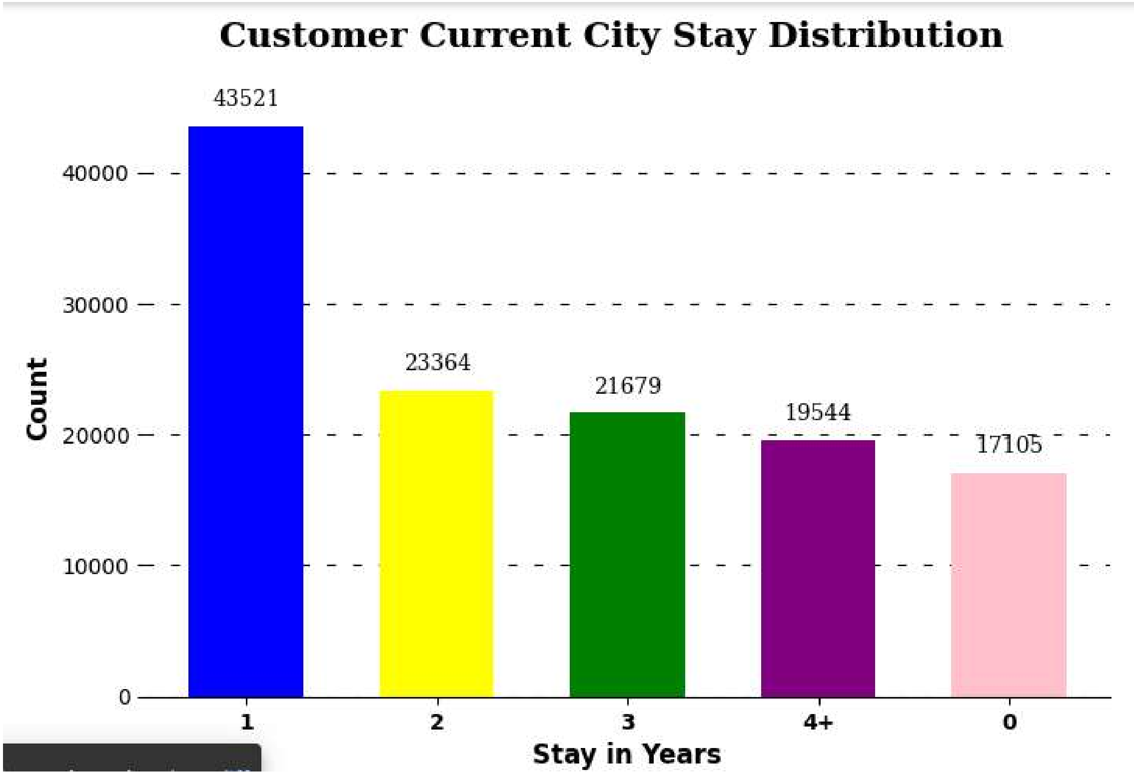
🔍 **Insights-**

- The age group of 26-35 represents the largest share of Walmart's Black Friday sales. This suggests that the young and middle-aged adults are the most active and interested in shopping for deals and discounts.

- The 36-45 and 18-25 age groups are the second and third largest segments.
- The 46-50, 51-55, 55+, and 0-17 age groups are the smallest customer segments.

🌆 <u>Customer Stay In current City Distribution:-</u>

```
fig,b = plt.subplots(figsize = (8,5))
b0 = df['Stay_In_Current_City_Years'].value_counts()
color_map = ['blue','yellow','green','purple','pink']
b.bar(x=b0.index,height = b0.values,color = color_map,zorder = 2,width = 0.6)
for i in b0.index:
    b.text(i,b0[i]+2000,b0[i],{'font':'serif','size' : 10},ha = 'center',va = 'center')
b.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes = (5,10))
for s in ['top','left','right']:
    b.spines[s].set_visible(False)
b.set_ylabel('Count',fontweight = 'bold',fontsize = 12)
b.set_xlabel('Stay in Years',fontweight = 'bold',fontsize = 12)
b.set_xticklabels(b0.index,fontweight = 'bold')
fig.suptitle('Customer Current City Stay Distribution',font = 'serif', size = 16, weight = 'bold')

plt.show()
```



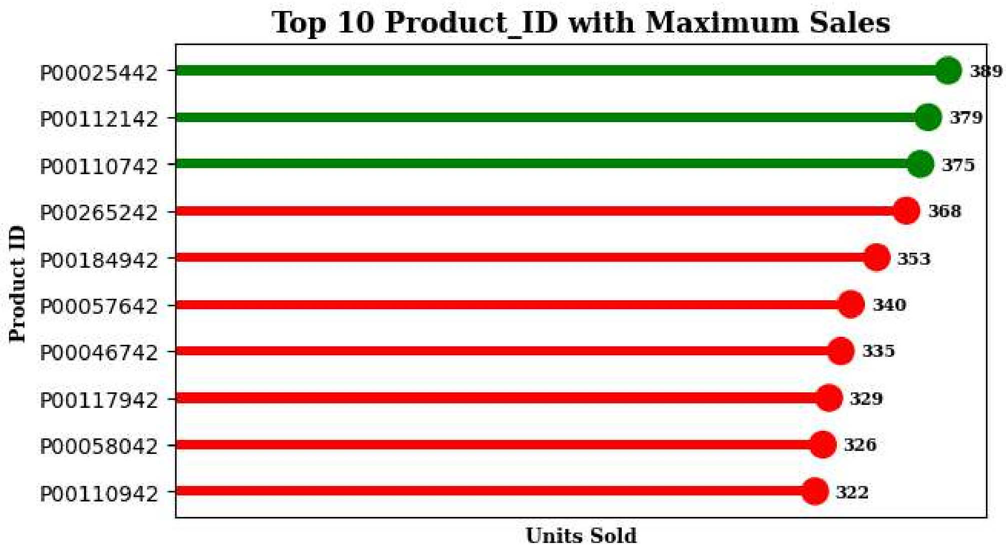
🔍 **Insights-**

- The majority of the customers (49%) have stayed in the current city for one year or less. This suggests that Walmart has a strong appeal to newcomers who may be looking for affordable and convenient shopping options.

- 4+ years category customers indicates that Walmart has a loyal customer base who have been living in the same city for a long time.

<u>Top 10 Products and Categories</u>:

```
fig = plt.figure(figsize = (14,6))
gs = fig.add_gridspec(1,2)

                                        #Top 10 Product_ID Sales

c = fig.add_subplot(gs[0,0])
ca = df['Product_ID'].value_counts()[0:10]
ca = ca.iloc[-1:-11:-1]

color_map =  ["red" for i in range(7)] + ["green" for i in range(3)]
c.barh(y = ca.index,width = ca.values,height = 0.2,color = color_map)
c.scatter(y = ca.index, x = ca.values, s = 150 , color = color_map )
c.set_xticks([])
for y,x in zip(ca.index,ca.values):
  c.text( x + 50 , y , x,{'font':'serif', 'size':8,'weight':'bold'},va='center')
c.set_xlabel('Units Sold',{'font':'serif', 'size':9,'weight':'bold'})
c.set_ylabel('Product ID',{'font':'serif', 'size':9,'weight':'bold'})
c.set_title('Top 10 Product_ID with Maximum Sales',
            {'font':'serif', 'size':13,'weight':'bold'})
plt.show()
```



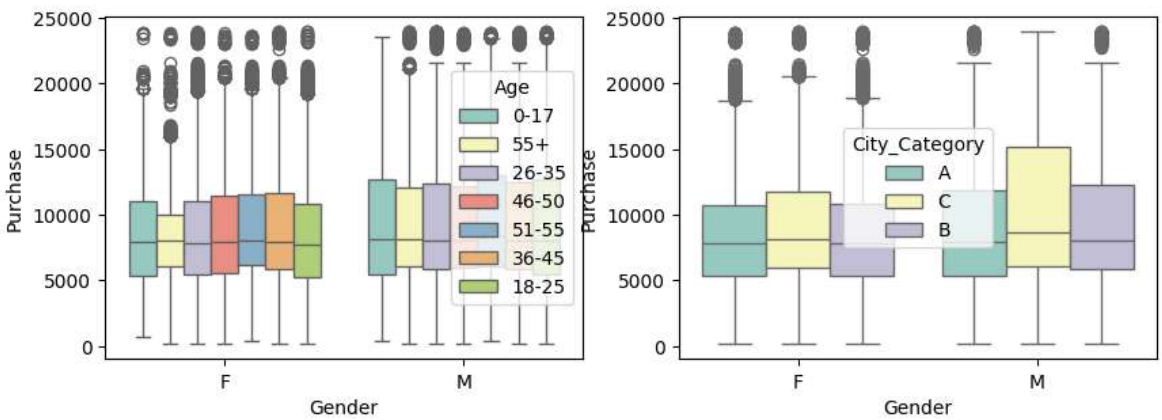
🔍 **Insights-**

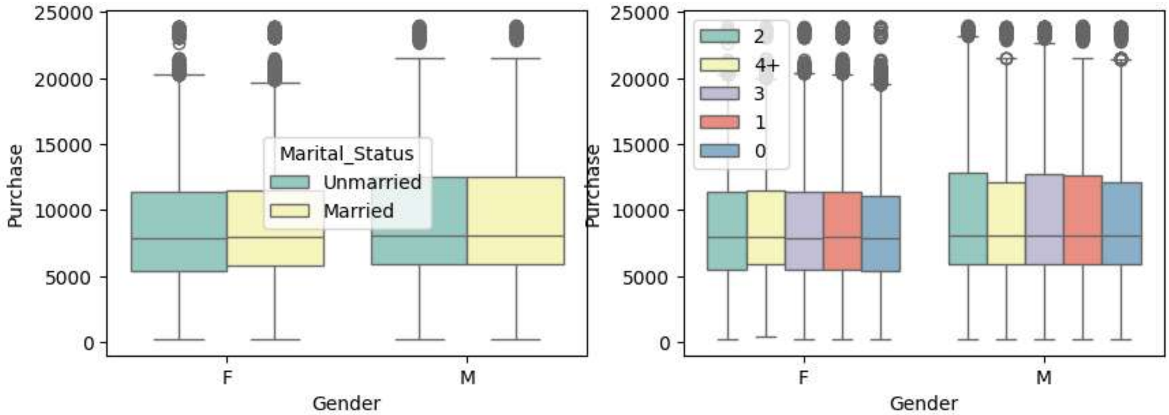
The top-selling products during Walmart's Black Friday sales are characterized by a relatively small variation in sales numbers, suggesting that Walmart offers a variety of products that many different customers like to buy.

Bivariate Analysis

📊 Exploring Purchase Patterns:-

```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(10,4))
fig.subplots_adjust(top=1.5)
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Age', palette='Set3', ax=axs[0,0])
sns.boxplot(data=df, y='Purchase', x='Gender', hue='City_Category', palette='Set3', ax=axs[0,1])

sns.boxplot(data=df, y='Purchase', x='Gender', hue='Marital_Status', palette='Set3', ax=axs[1,0])
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Stay_In_Current_City_Years', palette='Set3', ax=axs[1,1])
axs[1,1].legend(loc='upper left')

plt.show()
```



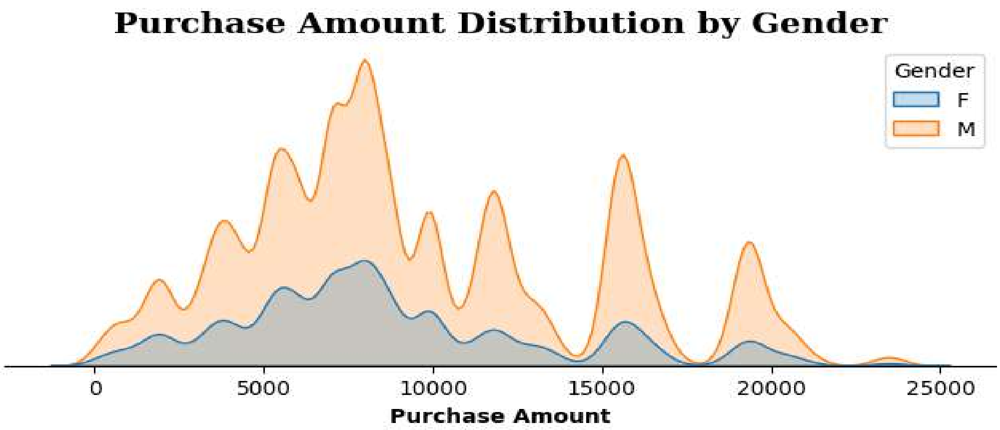


👨‍💼 👩‍💼 <u>Gender VS 💰 Purchase Amount:-</u>

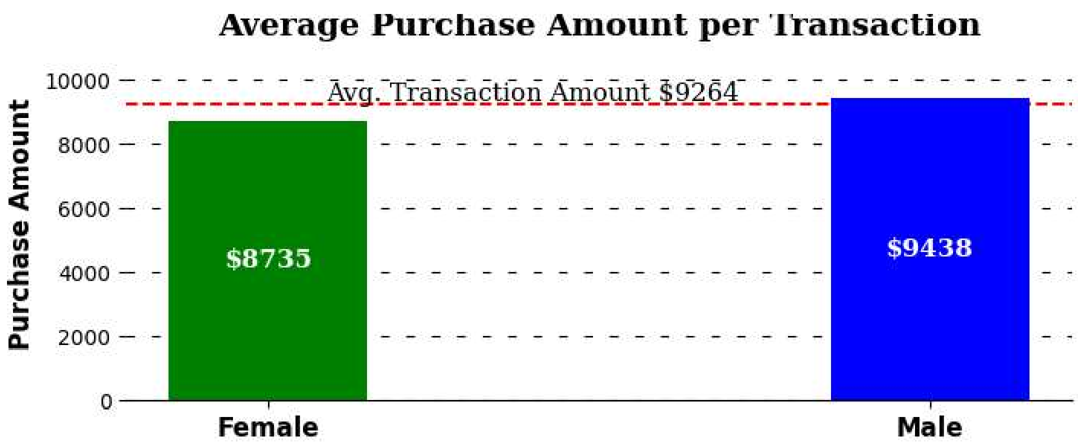
```
#creating a df for purchase amount vs gender
df1= df.groupby('Gender')['Purchase'].agg(['sum','count']).reset_index()

#calculating the amount in billions
df1['sum_in_billions'] = round(df1['sum'] / 10**9,2)

#calculationg percentage distribution of purchase amount
df1['%sum'] = round(df1['sum']/df1['sum'].sum(),3)

#calculationg per purchase amount
df1['per_purchase'] = round(df1['sum']/df1['count'])

#renaming the gender
df1['Gender'] = df1['Gender'].replace({'F':'Female','M':'Male'})

df1
```

	Gender	sum	count	sum_in_billions	%sum	per_purchase
0	Female	1186232642	135809	1.19	0.233	8735.0
1	Male	3909580100	414259	3.91	0.767	9438.0

```
fig,ax3 = plt.subplots(figsize = (8,3))

#plotting the kdeplot
sns.kdeplot(data = df, x = 'Purchase', hue = 'Gender',fill = True)

#removing the axis lines
for s in ['top','left','right']:
    ax3.spines[s].set_visible(False)

# adjusting axis labels
ax3.set_yticks([])
ax3.set_ylabel('')
ax3.set_xlabel('Purchase Amount',fontweight = 'bold',fontsize = 10)

#setting title for visual
ax3.set_title('Purchase Amount Distribution by Gender',{'font':'serif', 'size':15,'weight':'bold'})

plt.show()
```




```
fig,ax1= plt.subplots(figsize = (8,3))
color_map = ["green","blue"]
ax1.bar(df1['Gender'],df1['per_purchase'],color = color_map,zorder = 2,width = 0.3)
avg = round(df['Purchase'].mean())
ax1.axhline(y = avg, color ='red', zorder = 0,linestyle = '--')
ax1.text(0.4,avg + 300, f"Avg. Transaction Amount ${avg:.0f}",
         {'font':'serif','size' : 12},ha = 'center',va = 'center')
ax1.set_ylim(0,11000)
for i in df1.index:
    ax1.text(df1.loc[i,'Gender'],df1.loc[i,'per_purchase']/2,f"${df1.loc[i,'per_purchase']:.0f}",
             {'font':'serif','size' : 12,'color':'white','weight':'bold' },ha = 'center',va = 'center')
ax1.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes = (5,10))
for s in ['top','left','right']:
    ax1.spines[s].set_visible(False)
ax1.set_ylabel('Purchase Amount',fontweight = 'bold',fontsize = 12)
ax1.set_xticklabels(df1['Gender'],fontweight = 'bold',fontsize = 12)
ax1.set_title('Average Purchase Amount per Transaction',{'font':'serif', 'size':15,'weight':'bold'})
```



🔍 Insights-

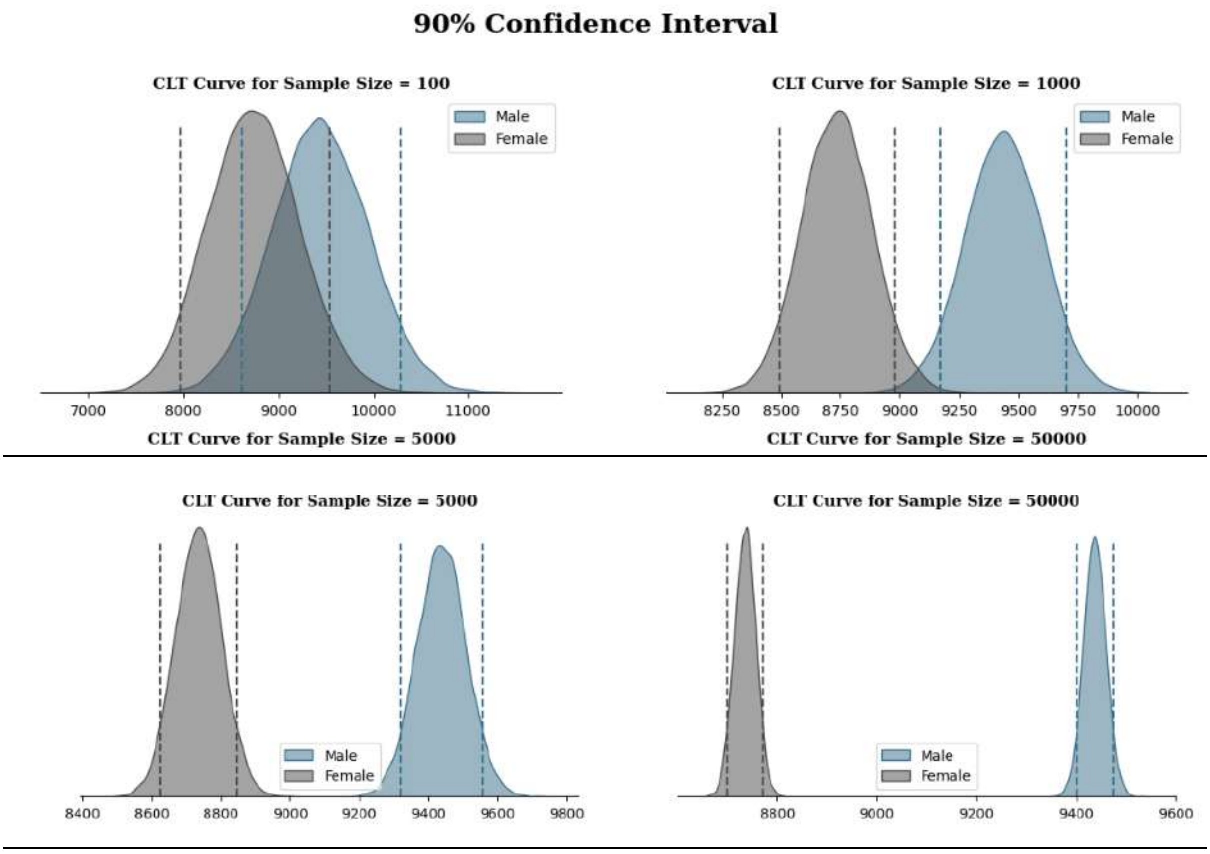
- The total purchase amount and number of transactions by male customers was more than three times the amount and transactions by female customers indicating that they had a more significant impact on the Black Friday sales.
- The average purchase amount per transaction was slightly higher for male customers than female customers ($9438 vs $8735).
- As seen above, the purchase amount for both the genders is not normally distributed.

Confidence Interval Construction: Estimating Average Purchase Amount per Transaction:-

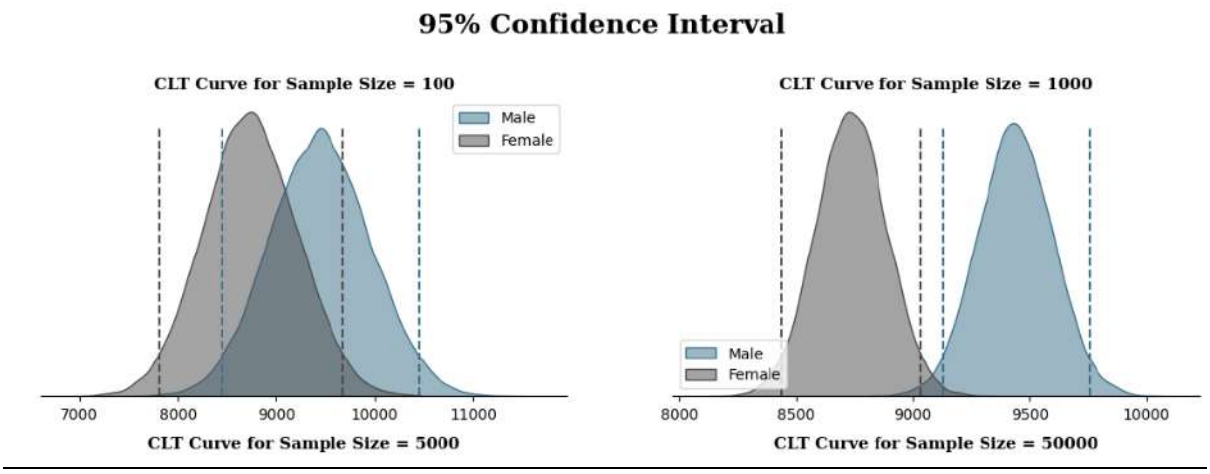
```
def confidence_interval(data,ci):
    l_ci = (100-ci)/2
    u_ci = (100+ci)/2
    interval = np.percentile(data,[l_ci,u_ci]).round(0)
    return interval
```

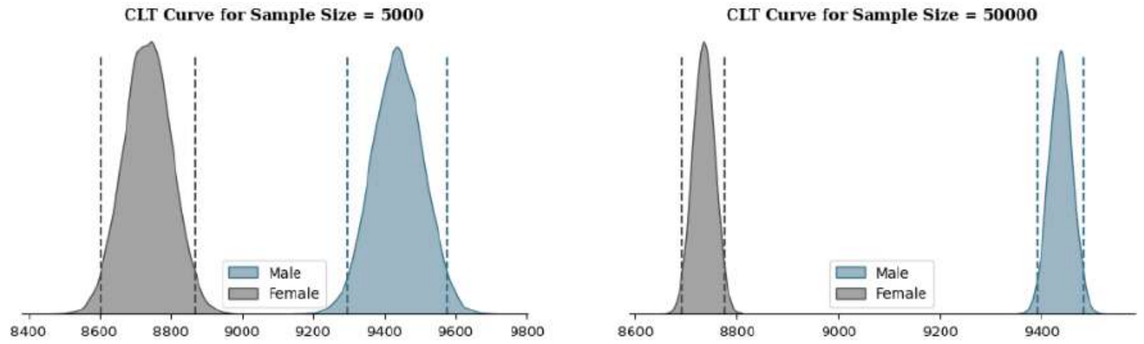
```
def plot(ci):

    fig = plt.figure(figsize = (14,8))
    gs = fig.add_gridspec(2,2)

    df_male = df.loc[df['Gender'] == 'M','Purchase']
    df_female = df.loc[df['Gender'] == 'F','Purchase']

    sample_sizes = [(100,0,0),(1000,0,1),(5000,1,0),(50000,1,1)]
    bootstrap_samples = 20000

    male_samples = {}
    female_samples = {}

    for i,x,y in sample_sizes:
        male_means = []
        female_means = []

        for j in range(bootstrap_samples):

            male_bootstrapped_samples = np.random.choice(df_male,size = i)
            female_bootstrapped_samples = np.random.choice(df_female,size = i)

            male_sample_mean = np.mean(male_bootstrapped_samples)
            female_sample_mean = np.mean(female_bootstrapped_samples)

            male_means.append(male_sample_mean)
            female_means.append(female_sample_mean)

        male_samples[f'{ci}%_{i}'] = male_means
        female_samples[f'{ci}%_{i}'] = female_means

        temp_df = pd.DataFrame(data = {'male_means':male_means,'female_means':female_means})

                                                #plotting kdeplots

        ax = fig.add_subplot(gs[x,y])

        sns.kdeplot(data = temp_df,x = 'male_means',color ="#3A7089" ,fill = True, alpha = 0.5,ax = ax,label = 'Male')
        sns.kdeplot(data = temp_df,x = 'female_means',color ="#4b4b4c" ,fill = True, alpha = 0.5,ax = ax,label = 'Female'

        m_range = confidence_interval(male_means,ci)
        f_range = confidence_interval(female_means,ci)

        for k in m_range:
            ax.axvline(x = k,ymax = 0.9, color ="#3A7089",linestyle = '--')

        for k in f_range:
            ax.axvline(x = k,ymax = 0.9, color ="#4b4b4c",linestyle = '--')

        for s in ['top','left','right']:
            ax.spines[s].set_visible(False)


        ax.set_yticks([])
        ax.set_ylabel('')
        ax.set_xlabel('')

        ax.set_title(f'CLT Curve for Sample Size = {i}',{'font':'serif', 'size':11,'weight':'bold'})

        plt.legend()

    fig.suptitle(f'{ci}% Confidence Interval',font = 'serif', size = 18, weight = 'bold')

    plt.show()

    return male_samples,female_samples
```

```
m_samp_90,f_samp_90 = plot(90)
```



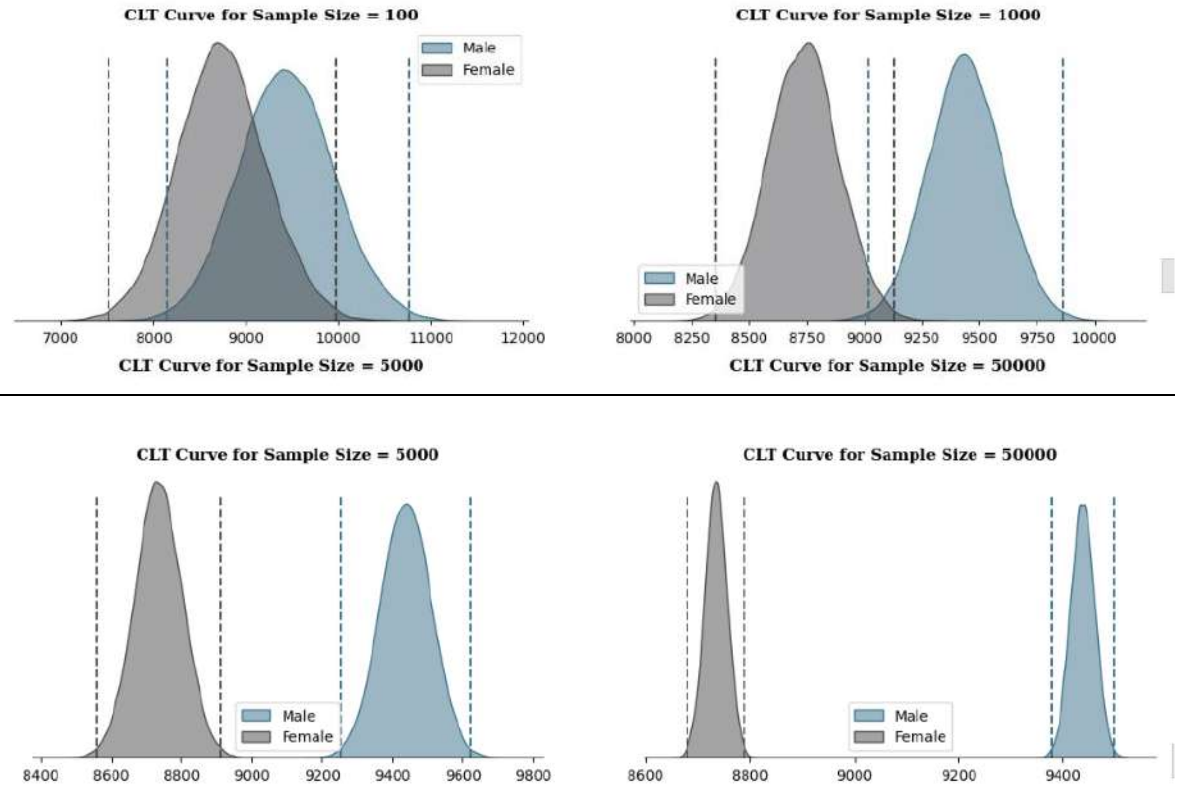

```
m_samp_95,f_samp_95 = plot(95)
```






```
m_samp_99,f_samp_99 = plot(99)
```

99% Confidence Interval



90% Confidence Interval Summary

Gender	Sample Size = 100	Sample Size = 1000	Sample Size = 5000	Sample Size = 50000
Male	CI = 8612 – 10277, Range = 1665	CI = 9171 – 9703, Range = 532	CI = 9320 – 9555, Range = 235	CI = 9400 – 9475, Range = 75
Female	CI = 7964 – 9538, Range = 1574	CI = 8489 – 8982, Range = 493	CI = 8625 – 8847, Range = 222	CI = 8699 – 8770, Range = 71

95% Confidence Interval Summary

Gender	Sample Size = 100	Sample Size = 1000	Sample Size = 5000	Sample Size = 50000
Male	CI = 8456 – 10449, Range = 1993	CI = 9128 – 9755, Range = 627	CI = 9297 – 9578, Range = 281	CI = 9392 – 9482, Range = 90
Female	CI = 7818 – 9673, Range = 1855	CI = 8438 – 9032, Range = 594	CI = 8604 – 8867, Range = 263	CI = 8692 – 8776, Range = 84

99% Confidence Interval Summary

Gender	Sample Size = 100	Sample Size = 1000	Sample Size = 5000	Sample Size = 50000
Male	CI = 8151 – 10760, Range = 2609	CI = 9020 – 9861, Range = 841	CI = 9253 – 9622, Range = 369	CI = 9377 – 9496, Range = 119
Female	CI = 7519 – 9976, Range = 2457	CI = 8353 – 9128, Range = 775	CI = 8560 – 8911, Range = 351	CI = 8680 – 8789, Range = 109

👫 Marital Status VS 💰 Purchase Amount

```
#creating a df for purchase amount vs marital status
temp = df.groupby('Marital_Status')['Purchase'].agg(['sum','count']).reset_index()

#calculating the amount in billions
temp['sum_in_billions'] = round(temp['sum'] / 10**9,2)

#calculationg percentage distribution of purchase amount
temp['%sum'] = round(temp['sum']/temp['sum'].sum(),3)

#calculationg per purchase amount
temp['per_purchase'] = round(temp['sum']/temp['count'])

temp
```

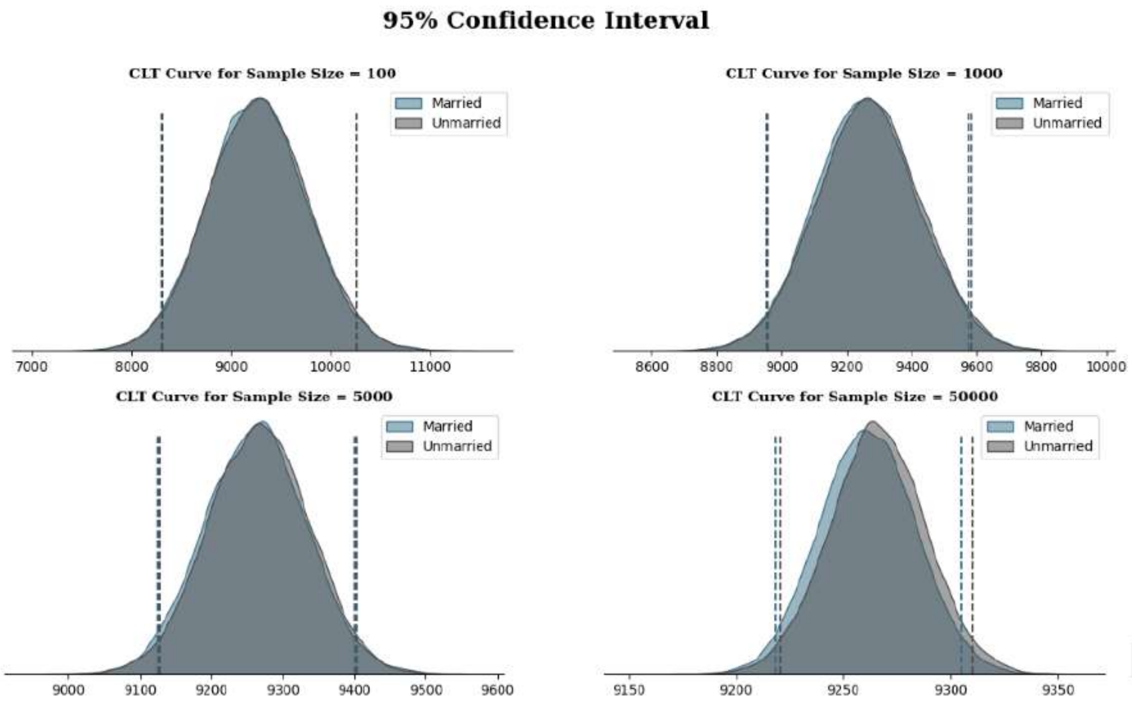
	Marital_Status	sum	count	sum_in_billions	%sum	per_purchase
0	Unmarried	3008927447	324731	3.01	0.59	9266.0
1	Married	2086885295	225337	2.09	0.41	9261.0

```
def plot(ci):

    fig = plt.figure(figsize = (15,8))
    gs = fig.add_gridspec(2,2)

    df_married = df.loc[df['Marital_Status'] == 'Married','Purchase']
    df_unmarried = df.loc[df['Marital_Status'] == 'Unmarried','Purchase']

    sample_sizes = [(100,0,0),(1000,0,1),(5000,1,0),(50000,1,1)]
    bootstrap_samples = 20000

    married_samples = {}
    unmarried_samples = {}

    for i,x,y in sample_sizes:
        married_means = []
        unmarried_means = []

        for j in range(bootstrap_samples):
          married_bootstrapped_samples = np.random.choice(df_married,size = i)
          unmarried_bootstrapped_samples = np.random.choice(df_unmarried,size = i)
          married_sample_mean = np.mean(married_bootstrapped_samples)
          unmarried_sample_mean = np.mean(unmarried_bootstrapped_samples)
```

```
          married_means.append(married_sample_mean)
          unmarried_means.append(unmarried_sample_mean)

        married_samples[f'{ci}%_{i}'] = married_means
        unmarried_samples[f'{ci}%_{i}'] = unmarried_means


        temp_df = pd.DataFrame(data = {'married_means':married_means,'unmarried_means':unmarried_means})

                                                #plotting kdeplots

        ax = fig.add_subplot(gs[x,y])

        sns.kdeplot(data = temp_df,x = 'married_means',color ="#3A7089" ,fill = True, alpha = 0.5,ax = ax,label = 'Married
        sns.kdeplot(data = temp_df,x = 'unmarried_means',color ="#4b4b4c" ,fill = True, alpha = 0.5,ax = ax,label = 'Unmar

        m_range = confidence_interval(married_means,ci)
        u_range = confidence_interval(unmarried_means,ci)

        for k in m_range:
            ax.axvline(x = k,ymax = 0.9, color ="#3A7089",linestyle = '--')

        for k in u_range:
            ax.axvline(x = k,ymax = 0.9, color ="#4b4b4c",linestyle = '--')
```

```
        for s in ['top','left','right']:
            ax.spines[s].set_visible(False)

        ax.set_yticks([])
        ax.set_ylabel('')
        ax.set_xlabel('')
        ax.set_title(f'CLT Curve for Sample Size = {i}',{'font':'serif', 'size':11,'weight':'bold'})

        plt.legend()

    fig.suptitle(f'{ci}% Confidence Interval',font = 'serif', size = 18, weight = 'bold')
    plt.show()
    return married_samples,unmarried_samples
```

```python
m_samp_95,u_samp_95 = plot(95)
```




```python
fig,ax = plt.subplots(figsize = (20,3))

m_ci = ['Married']
u_ci = ['Unmarried']

for m in m_samp_95:
    m_range = confidence_interval(m_samp_95[m],95)
    m_ci.append(f"CI = ${m_range[0]:.0f} - ${m_range[1]:.0f}, Range = {(m_range[1] - m_range[0]):.0f}")

for u in u_samp_95:
    u_range = confidence_interval(u_samp_95[u],95)
    u_ci.append(f"CI = ${u_range[0]:.0f} - ${u_range[1]:.0f}, Range = {(u_range[1] - u_range[0]):.0f}")

                                    #plotting the summary
ci_info = [m_ci,u_ci]

table = ax.table(cellText = ci_info, cellLoc='center',
             colLabels =['Marital_Status','Sample Size = 100','Sample Size = 1000','Sample Size = 5000','Sample Size = 50
             colLoc = 'center',colWidths = [0.1,0.225,0.225,0.225,0.225],bbox =[0, 0, 1, 1])

table.set_fontsize(13)

ax.axis('off')

   ax.set_title(f"95% Confidence Interval Summary",{'font':'serif', 'size':14,'weight':'bold'})

   plt.show()
```

95% Confidence Interval Summary

Marital_Status	Sample Size = 100	Sample Size = 1000	Sample Size = 5000	Sample Size = 50000
Married	CI = 8295 – 10261, Range = 1966	CI = 8953 – 9573, Range = 620	CI = 9124 – 9400, Range = 276	CI = 9218 – 9305, Range = 87
Unmarried	CI = 8304 – 10265, Range = 1961	CI = 8956 – 9582, Range = 626	CI = 9127 – 9403, Range = 276	CI = 9221 – 9310, Range = 89

```
[ ] #creating a df for purchase amount vs age group
    temp = df.groupby('Age')['Purchase'].agg(['sum','count']).reset_index()

    #calculating the amount in billions
    temp['sum_in_billions'] = round(temp['sum'] / 10**9,2)

    #calculationg percentage distribution of purchase amount
    temp['%sum'] = round(temp['sum']/temp['sum'].sum(),3)

    #calculationg per purchase amount
    temp['per_purchase'] = round(temp['sum']/temp['count'])

    temp
]
```

	Age	sum	count	sum_in_billions	%sum	per_purchase
0	0-17	134913183	15102	0.13	0.026	8933.0
1	18-25	913848675	99660	0.91	0.179	9170.0
2	26-35	2031770578	219587	2.03	0.399	9253.0
3	36-45	1026569884	110013	1.03	0.201	9331.0
4	46-50	420843403	45701	0.42	0.083	9209.0
5	51-55	367099644	38501	0.37	0.072	9535.0
6	55+	200767375	21504	0.20	0.039	9336.0

Start coding or generate with AI.

+ Code + Text

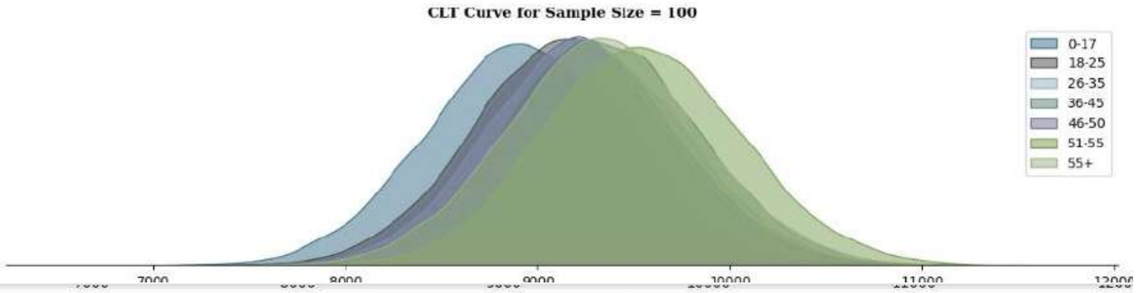
```
[ ] def plot(ci):
        fig = plt.figure(figsize = (15,15))
        gs = fig.add_gridspec(4,1)

        df_1 = df.loc[df['Age'] == '0-17','Purchase']
        df_2 = df.loc[df['Age'] == '18-25','Purchase']
        df_3 = df.loc[df['Age'] == '26-35','Purchase']
        df_4 = df.loc[df['Age'] == '36-45','Purchase']
        df_5 = df.loc[df['Age'] == '46-50','Purchase']
        df_6 = df.loc[df['Age'] == '51-55','Purchase']
        df_7 = df.loc[df['Age'] == '55+','Purchase']

        sample_sizes = [(100,0),(1000,1),(5000,2),(50000,3)]

        bootstrap_samples = 20000

        samples1,samples2,samples3,samples4,samples5,samples6,samples7 = {},{},{},{},{},{},{}

        for i,x in sample_sizes:
            l1,l2,l3,l4,l5,l6,l7 = [],[],[],[],[],[],[]

            for j in range(bootstrap_samples):
```

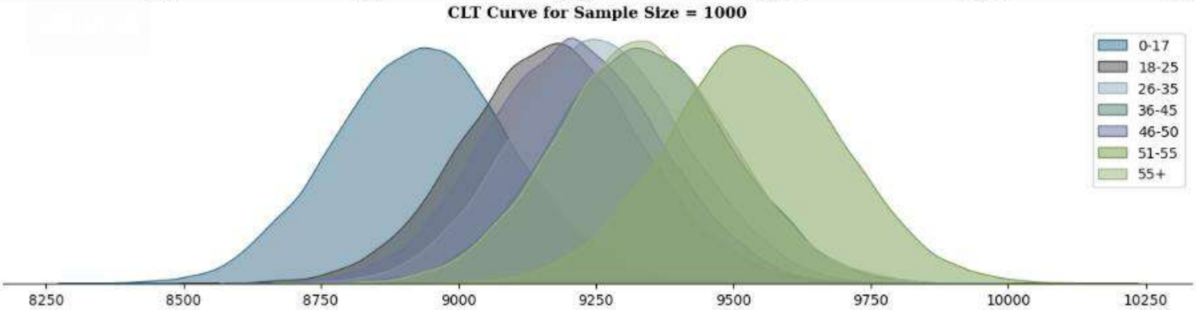
```
bootstrapped_samples_1 = np.random.choice(df_1,size = i)
bootstrapped_samples_2 = np.random.choice(df_2,size = i)
bootstrapped_samples_3 = np.random.choice(df_3,size = i)
bootstrapped_samples_4 = np.random.choice(df_4,size = i)
bootstrapped_samples_5 = np.random.choice(df_5,size = i)
bootstrapped_samples_6 = np.random.choice(df_6,size = i)
bootstrapped_samples_7 = np.random.choice(df_7,size = i)

sample_mean_1 = np.mean(bootstrapped_samples_1)
sample_mean_2 = np.mean(bootstrapped_samples_2)
sample_mean_3 = np.mean(bootstrapped_samples_3)
sample_mean_4 = np.mean(bootstrapped_samples_4)
sample_mean_5 = np.mean(bootstrapped_samples_5)
sample_mean_6 = np.mean(bootstrapped_samples_6)
sample_mean_7 = np.mean(bootstrapped_samples_7)

l1.append(sample_mean_1)
l2.append(sample_mean_2)
l3.append(sample_mean_3)
l4.append(sample_mean_4)
l5.append(sample_mean_5)
l6.append(sample_mean_6)
l7.append(sample_mean_7)
```

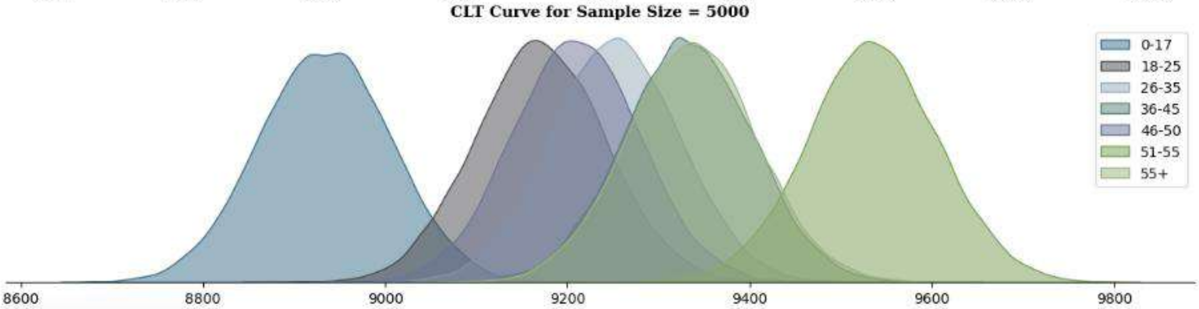
```
    samples1[f'{ci}%_{i}'] = l1
    samples2[f'{ci}%_{i}'] = l2
    samples3[f'{ci}%_{i}'] = l3
    samples4[f'{ci}%_{i}'] = l4
    samples5[f'{ci}%_{i}'] = l5
    samples6[f'{ci}%_{i}'] = l6
    samples7[f'{ci}%_{i}'] = l7

    #creating a temporary dataframe for creating kdeplot
    temp_df = pd.DataFrame(data = {'0-17':l1,'18-25':l2,'26-35':l3,'36-45':l4,'46-50':l5,'51-55':l6,'55+':l7})

                                                #plotting kdeplots

    ax = fig.add_subplot(gs[x])
    for p,q in [('#3A7089', '0-17'),('#4b4b4c', '18-25'),('#99AEBB', '26-35'),('#5C8374', '36-45'),('#6F7597', '46-50
             ('#7A9D54', '51-55'),('#9EB384', '55+')]:

        sns.kdeplot(data = temp_df,x = q,color =p ,fill = True, alpha = 0.5,ax = ax,label = q)
    for s in ['top','left','right']:
        ax.spines[s].set_visible(False)
    ax.set_yticks([])
    ax.set_ylabel('')
    ax.set_xlabel('')
    ax.set_title(f'CLT Curve for Sample Size = {i}',{'font':'serif', 'size':11,'weight':'bold'})
    plt.legend()
```

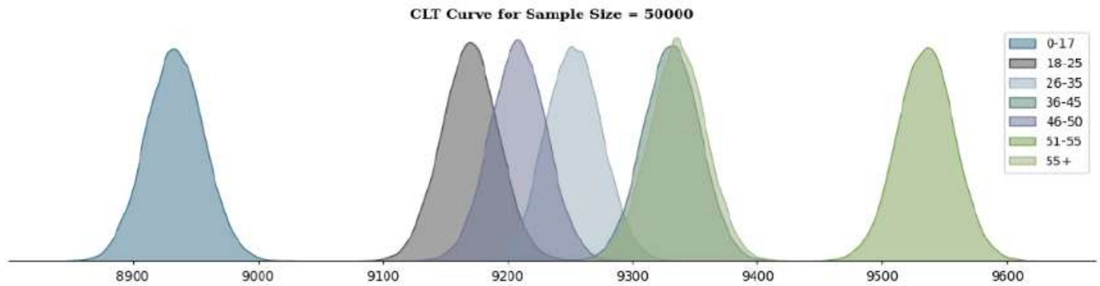
```
fig.suptitle(f'{ci}% Confidence Interval',font = 'serif', size = 18, weight = 'bold')
plt.show()
return samples1,samples2,samples3,samples4,samples5,samples6,samples7
```

```
[ ] samples1,samples2,samples3,samples4,samples5,samples6,samples7 = plot(95)
```










```
[ ] fig,ax = plt.subplots(figsize = (20,5))
    ci_1,ci_2,ci_3,ci_4,ci_5,ci_6,ci_7 = ['0-17'],['18-25'],['26-35'],['36-45'],['46-50'],['51-55'],['55+']

    samples = [(samples1,ci_1),(samples2,ci_2),(samples3,ci_3),(samples4,ci_4),(samples5,ci_5),(samples6,ci_6),(samples7,ci_7
    for s,c in samples:
        for i in s:
            s_range = confidence_interval(s[i],95)
            c.append(f"CI = ${s_range[0]:.0f} - ${s_range[1]:.0f}, Range = {(s_range[1] - s_range[0]):.0f}")
```

```python
                    #plotting the summary

ci_info = [ci_1,ci_2,ci_3,ci_4,ci_5,ci_6,ci_7]

table = ax.table(cellText = ci_info, cellLoc='center',
            colLabels =['Age Group','Sample Size = 100','Sample Size = 1000','Sample Size = 5000','Sample Size = 50000']
            colLoc = 'center',colWidths = [0.1,0.225,0.225,0.225,0.225],bbox =[0, 0, 1, 1])
table.set_fontsize(13)

ax.axis('off')
ax.set_title(f"95% Confidence Interval Summary",{'font':'serif', 'size':14,'weight':'bold'})
plt.show()
```

95% Confidence Interval Summary

Age Group	Sample Size = 100	Sample Size = 1000	Sample Size = 5000	Sample Size = 50000
0-17	CI = 7948 – 9941, Range = 1993	CI = 8624 – 9252, Range = 628	CI = 8792 – 9074, Range = 282	CI = 8889 – 8979, Range = 90
18-25	CI = 8197 – 10167, Range = 1970	CI = 8855 – 9488, Range = 633	CI = 9031 – 9312, Range = 281	CI = 9126 – 9214, Range = 88
26-35	CI = 8290 – 10251, Range = 1961	CI = 8946 – 9564, Range = 618	CI = 9114 – 9390, Range = 276	CI = 9209 – 9296, Range = 87
36-45	CI = 8364 – 10318, Range = 1954	CI = 9018 – 9646, Range = 628	CI = 9193 – 9470, Range = 277	CI = 9287 – 9375, Range = 88
46-50	CI = 8256 – 10211, Range = 1955	CI = 8896 – 9514, Range = 618	CI = 9070 – 9348, Range = 278	CI = 9165 – 9252, Range = 87
51-55	CI = 8562 – 10566, Range = 2004	CI = 9218 – 9852, Range = 634	CI = 9393 – 9676, Range = 283	CI = 9490 – 9579, Range = 89
55+	CI = 8353 – 10341, Range = 1988	CI = 9027 – 9650, Range = 623	CI = 9195 – 9477, Range = 282	CI = 9292 – 9380, Range = 88

🔍 Insights-

- The analysis highlights the importance of sample size in estimating population parameters. It suggests that as the sample size increases, the confidence intervals become narrower and more precise. In business, this implies that larger sample sizes can provide more reliable insights and estimates.

<u>Recommendations: -</u>

- Men spent more money than women, So company should focus on retaining the male customers and getting more male customers.
- 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.
- 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.
- Walmart may need to improve its marketing strategies and product offering to attract more customers from these age groups, especially the seniors and the children.
- Male customers living in City Category C spend more money than other male customers living in B or C, Selling more products in the City Category C will help the company increase the revenue.
- Company can focus on selling more of these products or selling more of the products which are purchased less.