Insights of Data

Import libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Import the dataset

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

```
--2023-10-02 01:41:04--  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, 108.157.172.173, 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  7.44MB/s    in 3.0s

2023-10-02 01:41:07 (7.44 MB/s) - ‘Walmart.csv’ saved [23027994/23027994]
```

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

Analysing the structure & characteristics of the dataset

```
df.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Yea
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	

```
df.shape
```

```
(550068, 10)
```

```
df.columns
```

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

```
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
```

Changing the data types 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

```
#Statistical Summary for categorical data
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

```
#Statistical Summary for numerical data
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

```
#Total No. of Users/Customers
df.User_ID.nunique()
```

```
5891
```

```
#No. of Age-groups
df.Age.unique()
```

```
['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+']
```

```
#No. of occupation
df.Occupation.unique()
```

```
[10, 16, 15, 7, 20, ..., 18, 5, 14, 13, 6]
Length: 21
Categories (21, int64): [0, 1, 2, 3, ..., 17, 18, 19, 20]
```

```
#Total no of product category
df.Product_Category.unique()
```

```
[3, 1, 12, 8, 5, ..., 10, 17, 9, 20, 19]
Length: 20
Categories (20, int64): [1, 2, 3, 4, ..., 17, 18, 19, 20]
```

```
#Total no. of Products
df.Product_ID.nunique()
```

```
3631
```

MIssing Values and Duplicates

```
#missing values
df.isnull().sum()
```

```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category              0
Purchase                      0
dtype: int64
```

```
#duplicate values
df.duplicated().sum()
```

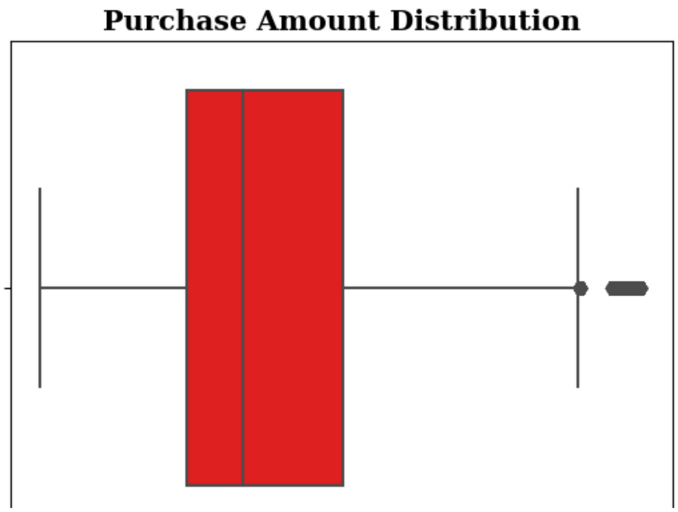
```
0
```

Detect Outliers

```
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

```
sns.boxplot(x= 'Purchase', data= df, color='red')
plt.title("Purchase Amount Distribution",{'font':'serif', 'size':15,'weight':'bold'})
plt.show()
```



Replacing the values of Marital_status column

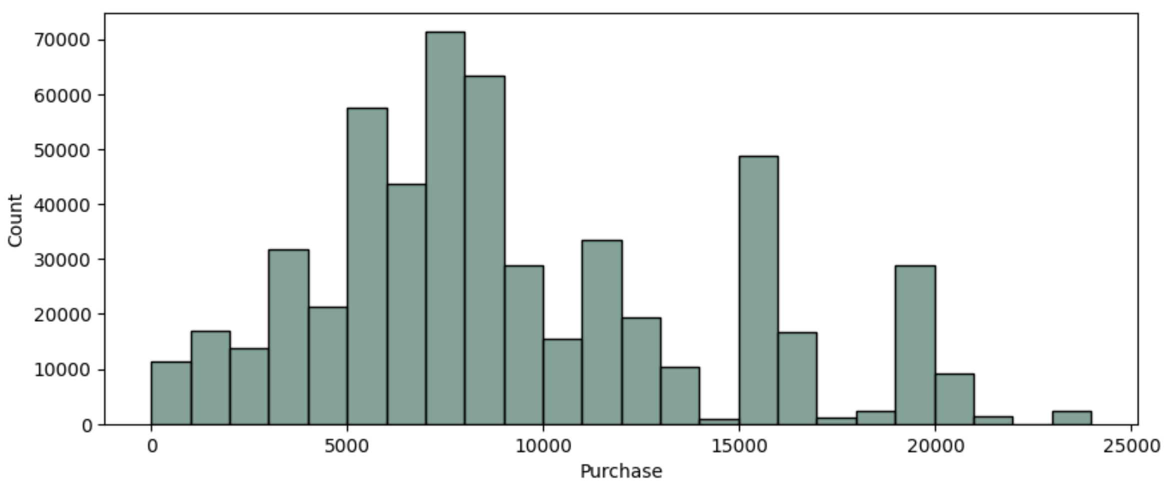
```
df['Marital_Status'] = df['Marital_Status'].replace({0: 'Unmarried',1: 'Married'})
df['Marital_Status'].unique()
```

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

Univariate Analysis

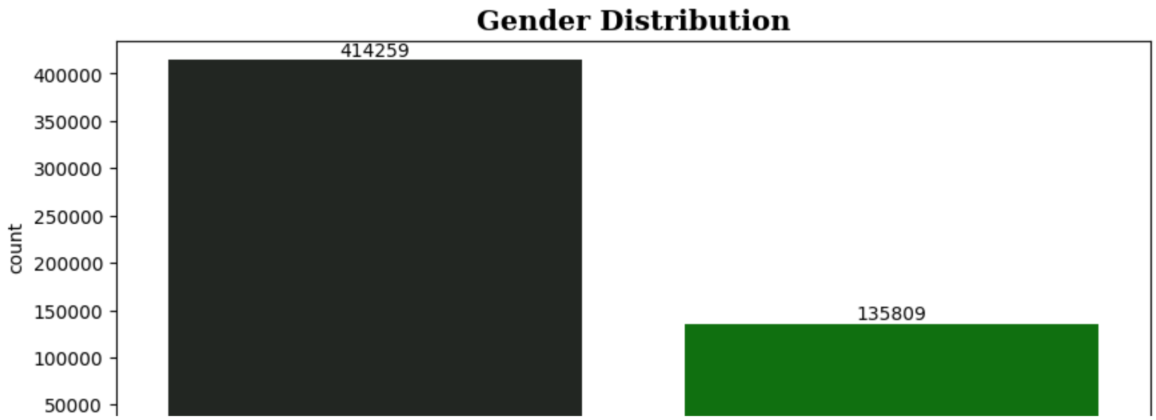
Continuous Variable

```
plt.figure(figsize = (10,4))
sns.histplot(x='Purchase', data=df, bins=24, color='#5C8374')
plt.show()
```

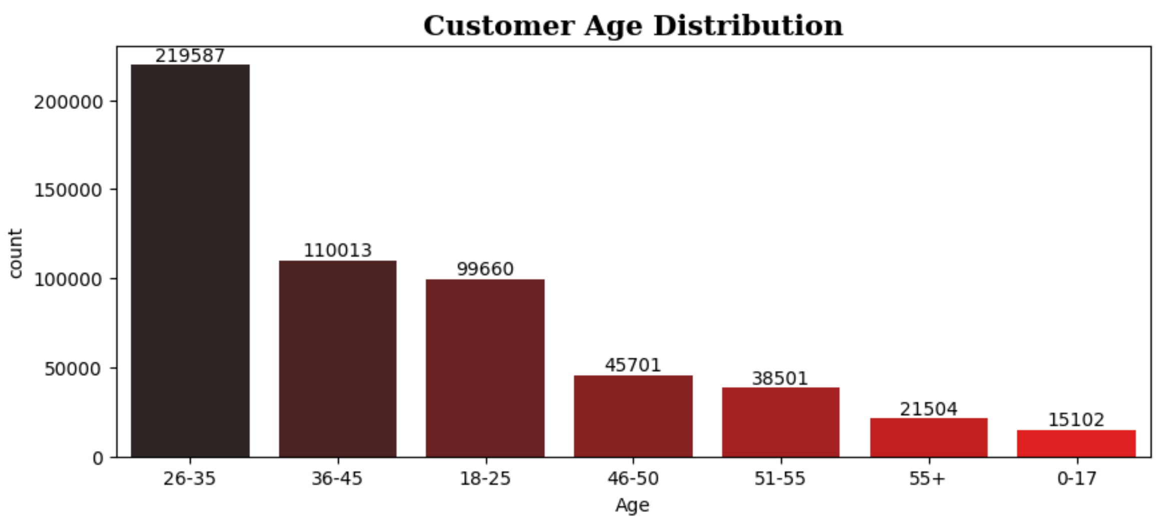


Categorical Variables

```
#Gender
fig = plt.figure(figsize = (10,4))
a = sns.countplot(x = "Gender", order = df['Gender'].value_counts().index, data = df, palette='dark:green')
a_values = df["Gender"].value_counts().values
a.bar_label(container=a.containers[0], labels=a_values)
plt.title('Gender Distribution',{'font':'serif', 'size':15,'weight':'bold'})
plt.show()
```



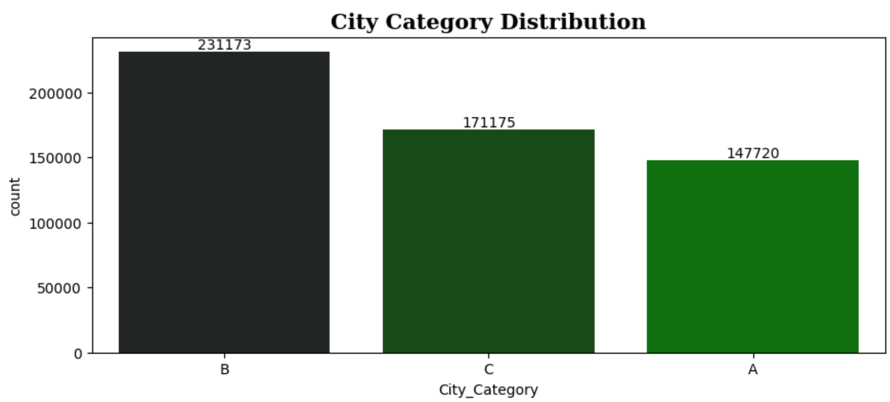

```
#Age
fig = plt.figure(figsize = (10,4))
a = sns.countplot(x = "Age", order = df['Age'].value_counts().index, data = df, palette='dark:red')
a_values = df["Age"].value_counts().values
a.bar_label(container=a.containers[0], labels=a_values)
plt.title('Customer Age Distribution',{'font':'serif', 'size':15,'weight':'bold'})
plt.show()
```



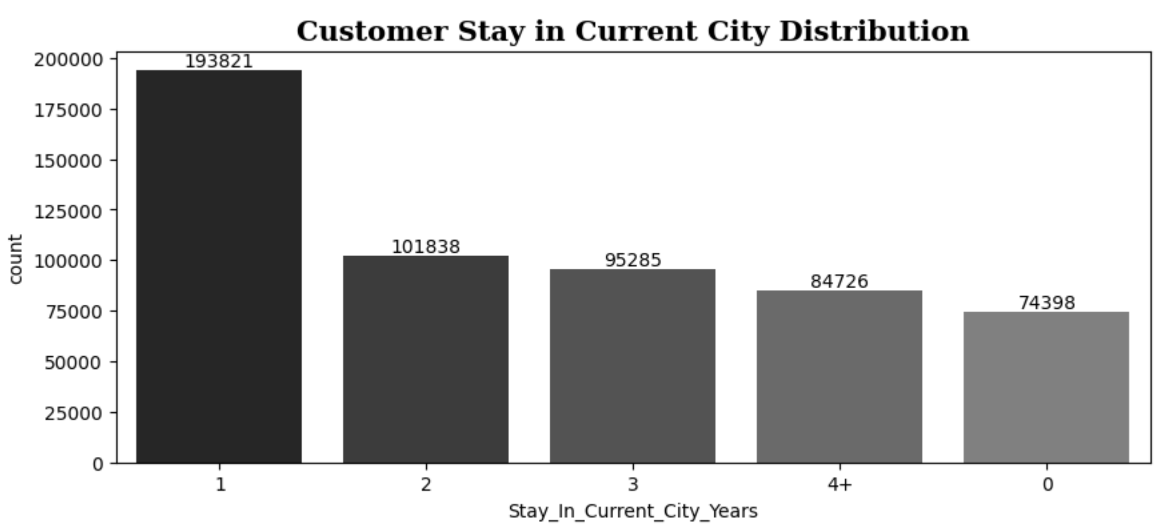

```
#Marital Status Distribution
fig = plt.figure(figsize = (10,4))
a = sns.countplot(x = "Marital_Status", order = df['Marital_Status'].value_counts().index, data = df, palette='dark:yellow')
a_values = df["Marital_Status"].value_counts().values
a.bar_label(container=a.containers[0], labels=a_values)
plt.title('Marital Status Distribution',{'font':'serif', 'size':15,'weight':'bold'})
plt.show()
```

Marital Status Distribution

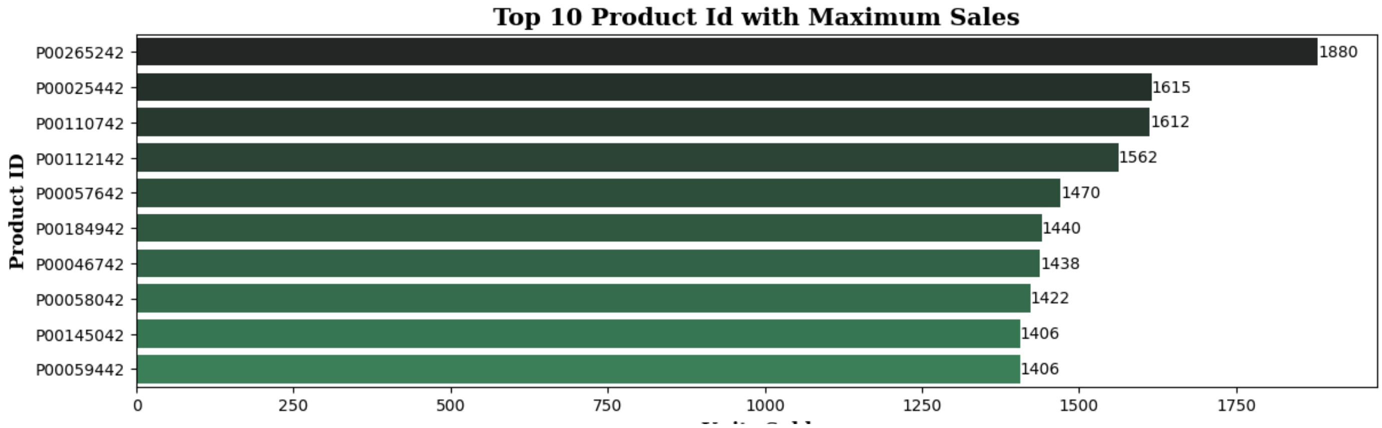
```
#City Category
fig = plt.figure(figsize = (10,4))
a = sns.countplot(x = "City_Category", order = df['City_Category'].value_counts().index, data = df, palette='dark:green')
a_values = df["City_Category"].value_counts().values
a.bar_label(container=a.containers[0], labels=a_values)
plt.title('City Category Distribution',{'font':'serif', 'size':15,'weight':'bold'})
plt.show()
```



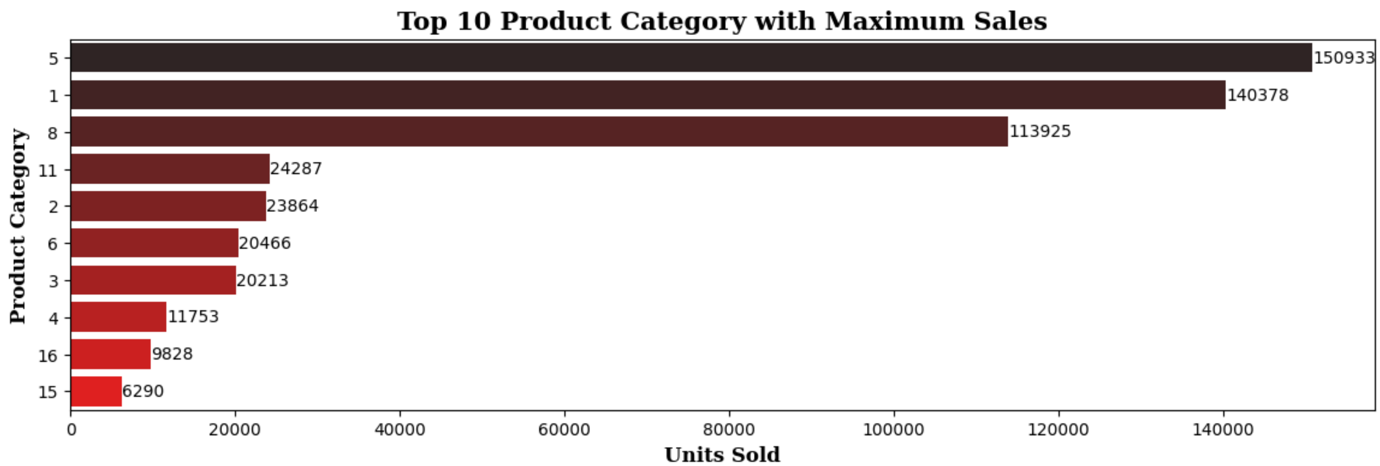

```
#Customer stay in current city
fig = plt.figure(figsize = (10,4))
a = sns.countplot(x = "Stay_In_Current_City_Years", data = df, palette='dark:grey', order = df['Stay_In_Current_City_Years'].value_counts().i
a_values = df["Stay_In_Current_City_Years"].value_counts().values
a.bar_label(container=a.containers[0], labels=a_values)
plt.title('Customer Stay in Current City Distribution',{'font':'serif', 'size':15,'weight':'bold'})
plt.show()
```



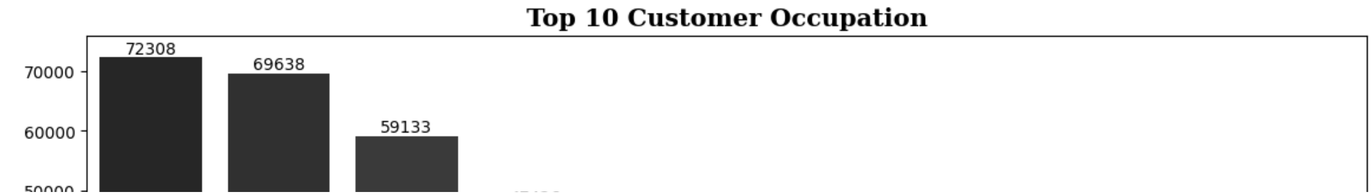

```
#Product_ID
fig = plt.figure(figsize = (14,4))

a=sns.countplot(y='Product_ID', data=df, order=df.Product_ID.value_counts().index[:10], palette='dark:seagreen')
plt.bar_label(container=a.containers[0])
plt.xlabel('Units Sold',{'font':'serif', 'size':12,'weight':'bold'})
plt.ylabel('Product ID',{'font':'serif', 'size':12,'weight':'bold'})
plt.title('Top 10 Product Id with Maximum Sales',{'font':'serif', 'size':15,'weight':'bold'})
plt.show()
```




```
#Product_Category
fig = plt.figure(figsize = (14,4))

b = sns.countplot(y='Product_Category', data=df, order=df.Product_Category.value_counts().index[:10], palette='dark:red')
plt.bar_label(container=b.containers[0])
plt.xlabel('Units Sold',{'font':'serif', 'size':12,'weight':'bold'})
plt.ylabel('Product Category',{'font':'serif', 'size':12,'weight':'bold'})
plt.title('Top 10 Product Category with Maximum Sales',{'font':'serif', 'size':15,'weight':'bold'})
plt.show()
```




```
#Customer_Occupation
fig = plt.figure(figsize = (14,5))

c = sns.countplot(x='Occupation', data=df, order=df.Occupation.value_counts().index[:10], palette='dark:grey')
plt.bar_label(container=c.containers[0])
plt.xlabel('Occupation of Customer',{'font':'serif', 'size':12,'weight':'bold'})
plt.title('Top 10 Customer Occupation',{'font':'serif', 'size':15,'weight':'bold'})
plt.show()
```



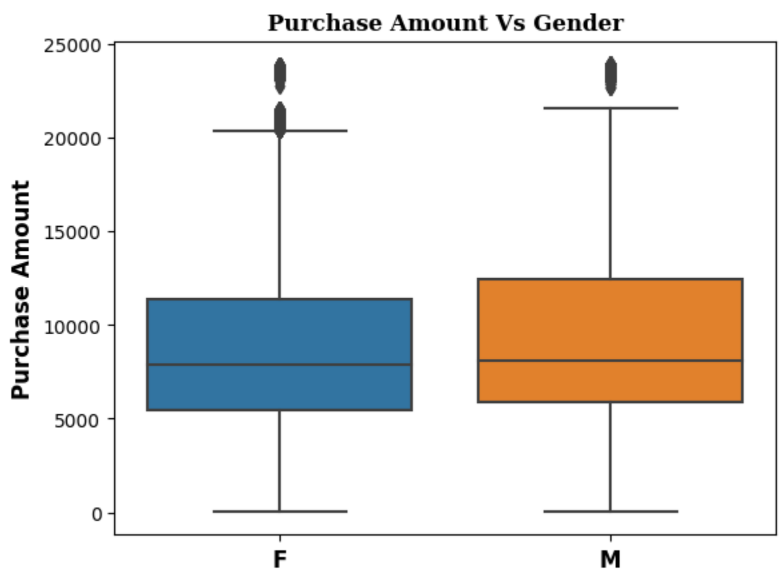
Bivariate Analysis

Defining Function for Boxplots of Purchase Amount Across various Variables

```
def Boxplot(k):
  ax = sns.boxplot(x=k, y='Purchase', data=df)
  plt.title(f'Purchase Amount Vs {k}',{'font':'serif', 'size':12,'weight':'bold'}),
  plt.ylabel('Purchase Amount',fontweight = 'bold',fontsize = 12),
  ax.set_xticklabels(df[k].unique(),fontweight = 'bold',fontsize = 12),
  plt.xlabel(''),
  plt.show()

  return None
```

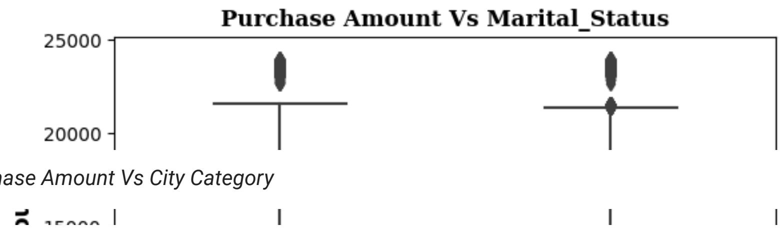
Purchase Amount Vs Gender

```
Boxplot('Gender')
```



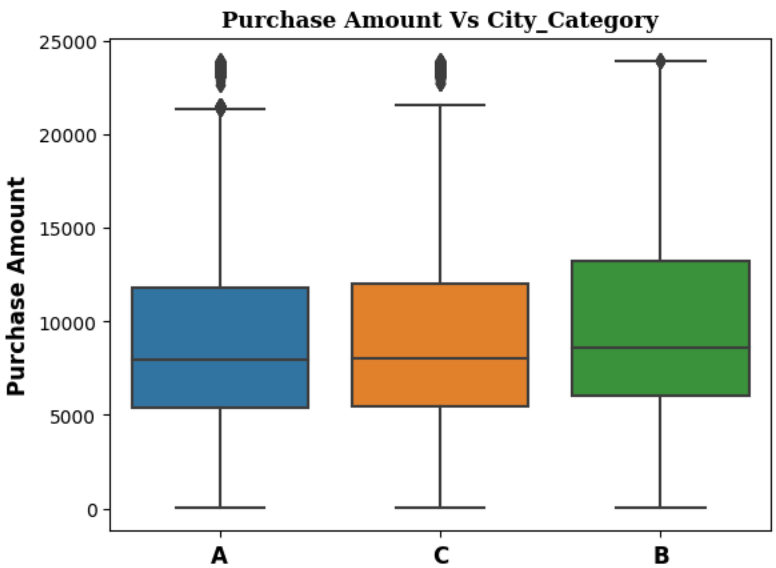
Purchase Amount Vs Marital Status

```
Boxplot('Marital_Status')
```



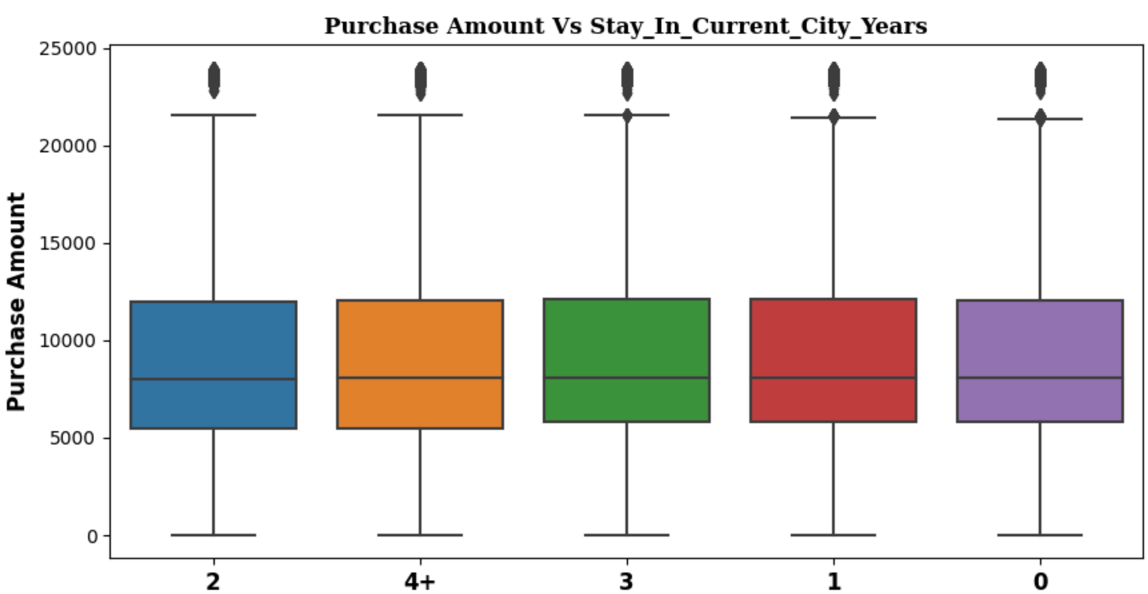
Purchase Amount Vs City Category

```
Boxplot('City_Category')
```



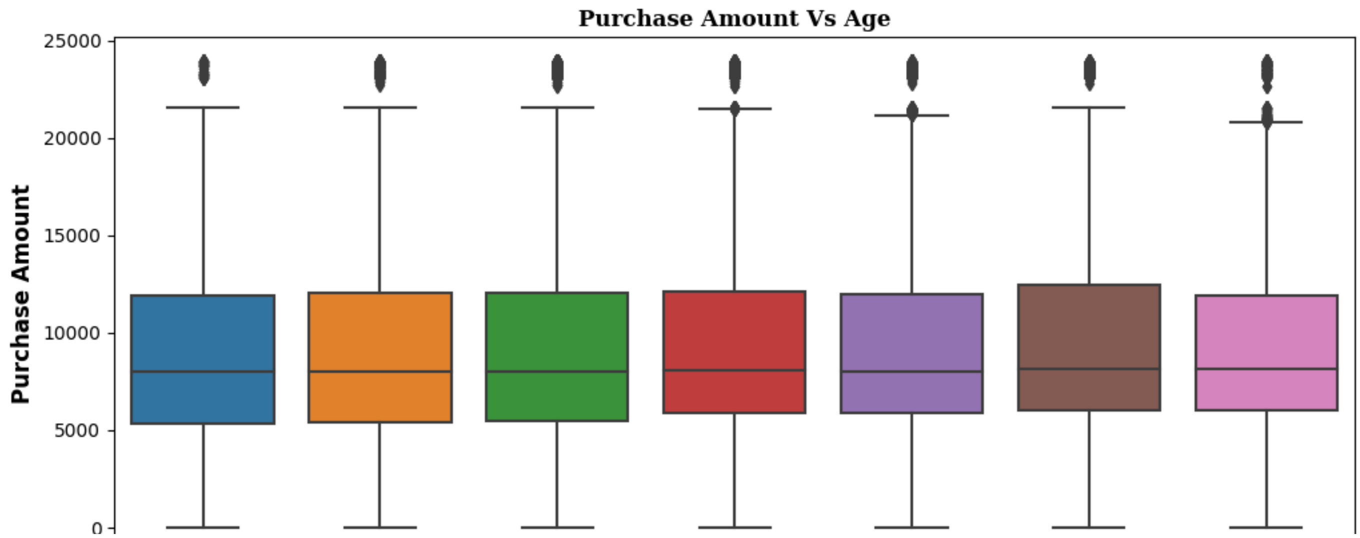
Purchase Amount Vs Stay in Current City Years

```
fig = plt.figure(figsize = (10,5))
Boxplot('Stay_In_Current_City_Years')
```



Purchase Amount Vs Age

```
fig = plt.figure(figsize = (12,5))
Boxplot('Age')
```



Gender VS Purchase Amount

Data Visualization

Sample Checking Gender-wise

```
sample_data = df.sample(50000,replace=True).groupby(['Gender'])['Purchase'].agg(['sum','count','mean']).reset_index()

sample_data['%sum'] = round(sample_data['sum']/sample_data['sum'].sum(),3)

sample_data
```

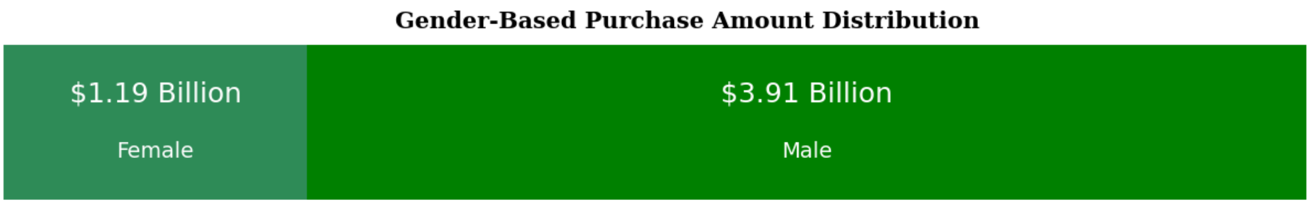
	Gender	sum	count	mean	%sum
0	F	110036147	12675	8681.352821	0.237
1	M	353323907	37325	9466.146202	0.763

```
#creating separate dataframe for Gender and Purchase Amount

Gdf = df.groupby('Gender')['Purchase'].agg(['sum','count']).reset_index()

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

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

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

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

Gdf
```

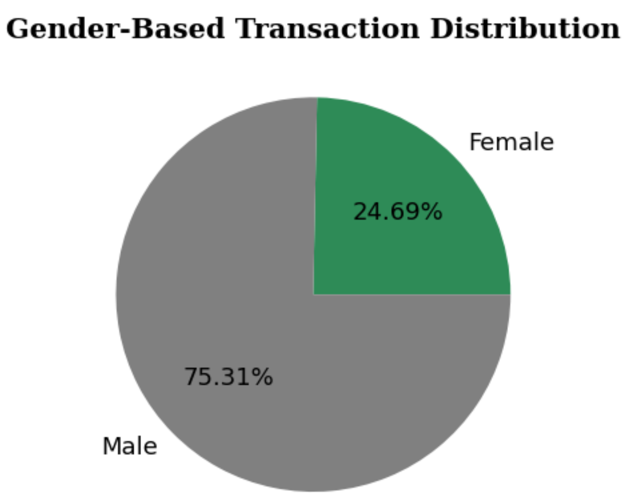
	Gender	sum	count	sum(billions)	%sum	per_purchase
0	Female	1186232642	135809	1.19	0.233	8735.0
1	Male	3909580100	414259	3.91	0.767	9438.0

```
#Distribution of Purchase Amount

fig = plt.figure(figsize=(16,2))

#plotting the visual
plt.barh(Gdf.loc[0,'Gender'],width = Gdf.loc[0,'%sum'],color = "seagreen",label = 'Female')
```

```
plt.barh(Gdf.loc[0,'Gender'],width = Gdf.loc[1,'%sum'],left =Gdf.loc[0,'%sum'], color =  "green",label = 'Male' )

#inserting the text
txt = [0.0]

for i in Gdf.index:
    #for amount
    plt.text(Gdf.loc[i,'%sum']/2 + txt[0],0.15,f"${Gdf.loc[i,'sum(billions)']} Billion",
           va = 'center', ha='center',fontsize=18, color='white')

    #for gender
    plt.text(Gdf.loc[i,'%sum']/2 + txt[0],- 0.15 ,f"{Gdf.loc[i,'Gender']}",
           va = 'center', ha='center',fontsize=14, color='white')

    txt += Gdf.loc[i,'%sum']

#removing the axis lines
plt.axis('off')

#plot title
plt.title('Gender-Based Purchase Amount Distribution',{'font':'serif', 'size':15,'weight':'bold'})

plt.show()
```

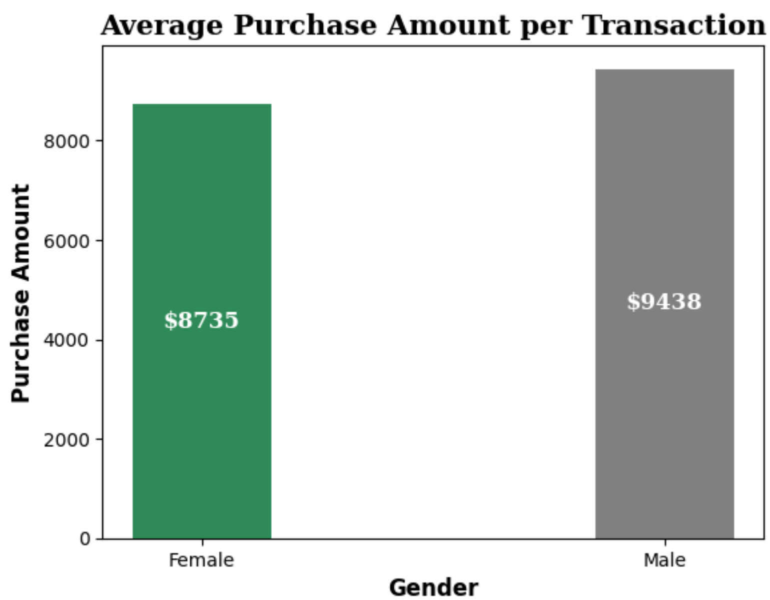



```
# creating pie chart for gender disribution

color_map = ["seagreen","grey"]
plt.pie(Gdf['count'],autopct = '%.2f%%', labels= Gdf['Gender'], colors=color_map, wedgeprops = {'linewidth': 5},textprops={'fontsize': 13, 'c

#setting title for visual
plt.title('Gender-Based Transaction Distribution',{'font':'serif', 'size':15,'weight':'bold'})

plt.show()
```




```
#Average perchase amount per transaction

plt.bar(Gdf['Gender'],Gdf['per_purchase'],color = color_map,width = 0.3)

for i in Gdf.index:
    plt.text(Gdf.loc[i,'Gender'],Gdf.loc[i,'per_purchase']/2,f"${Gdf.loc[i,'per_purchase']:.0f}",
             {'font':'serif','size' : 12,'color':'white','weight':'bold' },ha = 'center',va = 'center')

plt.ylabel('Purchase Amount',fontweight = 'bold',fontsize = 12)
```

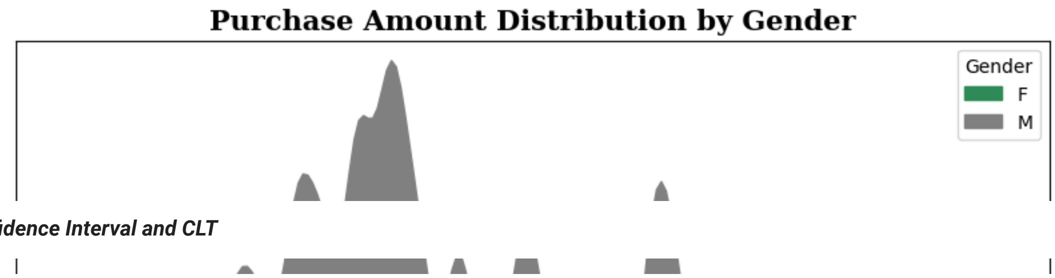
```
plt.xlabel('Gender', fontweight = 'bold',fontsize = 12)

plt.title('Average Purchase Amount per Transaction',{'font':'serif', 'size':15,'weight':'bold'})

plt.show()
```




```
#Question: Are women spending more money per transaction than men?

dd= df.groupby(['Gender','Marital_Status'])['Purchase'].agg(['sum','count']).reset_index()
dd['avg_per_person'] = round(dd['sum']/dd['count'],0)

dd
```

	Gender	Marital_Status	sum	count	avg_per_person
0	F	Unmarried	684154127	78821	8680.0
1	F	Married	502078515	56988	8810.0
2	M	Unmarried	2324773320	245910	9454.0
3	M	Married	1584806780	168349	9414.0

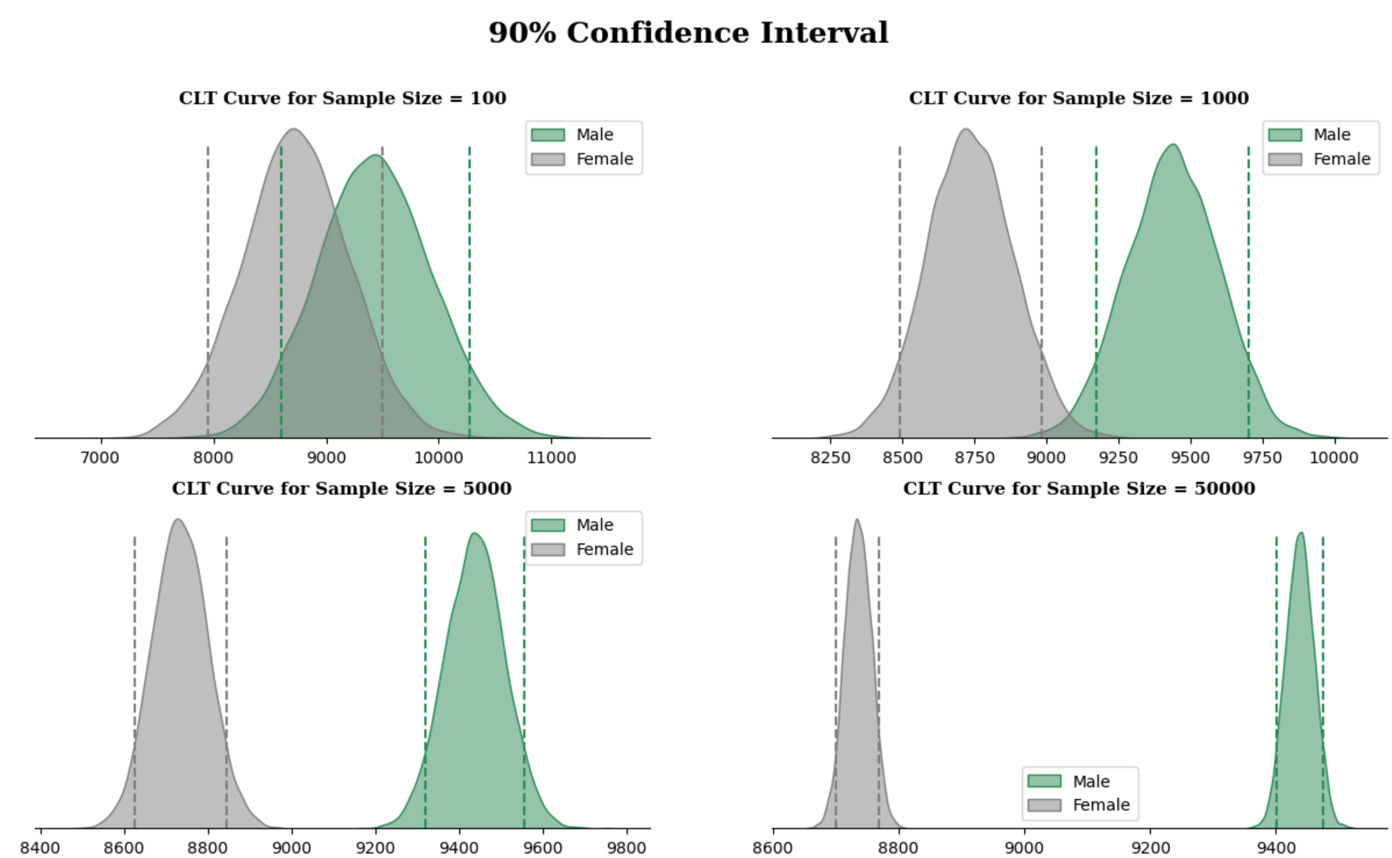
```
# creating kdeplot for purchase amount distribution

fig = plt.figure(figsize=(10,4))
sns.kdeplot(data = df, x = 'Purchase', hue = 'Gender', palette = color_map, fill = True, alpha = 1)

# adjusting axis labels
plt.yticks([])
plt.ylabel('')
plt.xlabel('Purchase Amount',fontweight = 'bold',fontsize = 12)

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

plt.show()
```



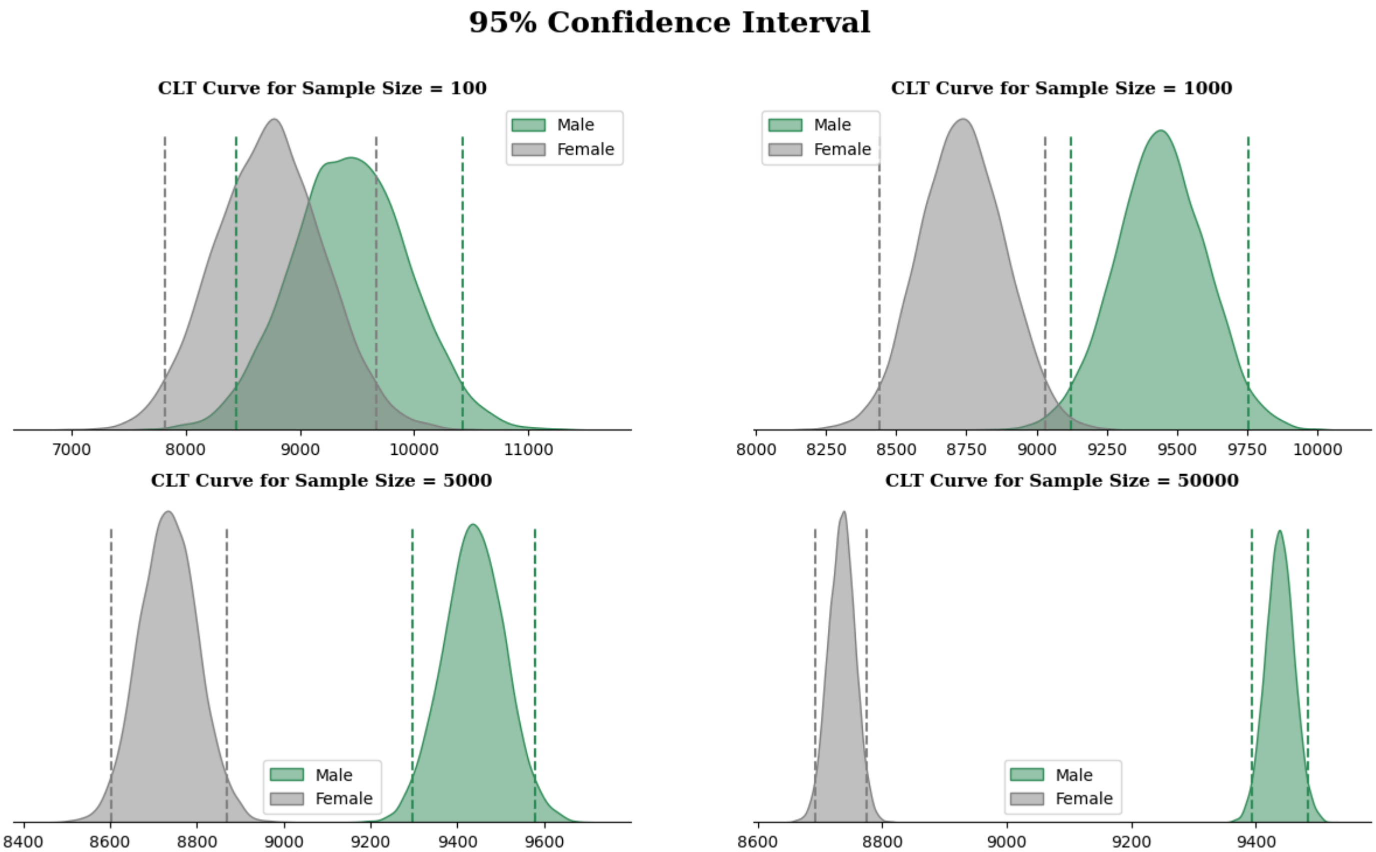
Confidence Interval and CLT

```
#creating a function to calculate confidence interval

def confidence_interval(data,ci):
    #converting the list to series
    lower_ci = (100-ci)/2
    upper_ci = (100+ci)/2

    #calculating lower limit and upper limit of confidence interval
    interval = np.percentile(data,[lower_ci,upper_ci]).round(0)

    return interval


#defining a function for plotting the visual for given confidence interval

def plot(ci):

    #setting the plot style
    fig = plt.figure(figsize = (15,8))
    gs = fig.add_gridspec(2,2)

    #creating separate data frames for each gender
    df_male = df.loc[df['Gender'] == 'M','Purchase']
    df_female = df.loc[df['Gender'] == 'F','Purchase']

    #sample sizes and corresponding plot positions
    sample_sizes = [(100,0,0),(1000,0,1),(5000,1,0),(50000,1,1)]

    #number of samples to be taken from purchase amount
    boot_samples = 10000

    male_samples = {}
    female_samples = {}

    for i,x,y in sample_sizes:
        male_means = [] #list for collecting the means of male sample
        female_means = [] #list for collecting the means of female sample

        for j in range(boot_samples):

            #creating random 5000 samples of i sample size
            male_boot_samples = np.random.choice(df_male,size = i)
            female_boot_samples = np.random.choice(df_female,size = i)

            #calculating mean of those samples
            male_sample_mean = np.mean(male_boot_samples)
            female_sample_mean = np.mean(female_boot_samples)

            #appending the mean to the list
            male_means.append(male_sample_mean)
            female_means.append(female_sample_mean)

        #storing the above sample generated
        male_samples[f'{ci}%_{i}'] = male_means
        female_samples[f'{ci}%_{i}'] = female_means

        #creating a temporary dataframe for creating kdeplot
        temp_df = pd.DataFrame(data = {'male_means':male_means,'female_means':female_means})

                                                        #plotting kdeplots
        #plot position
        ax = fig.add_subplot(gs[x,y])

        #plots for male and female
        sns.kdeplot(data = temp_df,x = 'male_means',color ="seagreen" ,fill = True, alpha = 0.5,label = 'Male')
```

```
        sns.kdeplot(data = temp_df,x = 'female_means',color ="grey" ,fill = True, alpha = 0.5,label = 'Female')

        #calculating confidence intervals for given confidence level(ci)
        m_range = confidence_interval(male_means,ci)
        f_range = confidence_interval(female_means,ci)

        #plotting confidence interval on the distribution
        for k in m_range:
            ax.axvline(x = k,ymax = 0.9, color ="seagreen",linestyle = '--')

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


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

        # adjusting axis labels
        ax.set_yticks([])
        ax.set_ylabel('')
        ax.set_xlabel('')

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

        plt.legend()

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

    plt.show()

    return male_samples,female_samples

male_sample_90,female_sample_90 = plot(90)
```



```
male_sample_95,female_sample_95 = plot(95)
```




```
male_sample_99,female_sample_99 = plot(99)
```


```
fig = plt.figure(figsize = (20,10))
gs = fig.add_gridspec(3,1)

for i,j,k,l in [(male_sample_90,female_sample_90,90,0),(male_sample_95,female_sample_95,95,1),(male_sample_99,female_sample_99,99,2)]:
    #list for collecting ci for given cl
    m_ci = ['Male']
    f_ci = ['Female']

    #finding ci for each sample size (males)
    for m in i:
        m_range = confidence_interval(i[m],k)
        m_ci.append(f"CI = ${m_range[0]:.0f} - ${m_range[1]:.0f}, Range = {(m_range[1] - m_range[0]):.0f}")

    #finding ci for each sample size (females)
    for f in j:
        f_range = confidence_interval(j[f],k)
        f_ci.append(f"CI = ${f_range[0]:.0f} - ${f_range[1]:.0f}, Range = {(f_range[1] - f_range[0]):.0f}")

                                    #plotting the summary
    ax = fig.add_subplot(gs[l])

    #contents of the table
    ci_info = [m_ci,f_ci]

    #plotting the table
    table = ax.table(cellText = ci_info, cellLoc='center',
                     colLabels =['Gender','Sample Size = 100','Sample Size = 1000','Sample Size = 5000','Sample Size = 50000'],
                     colLoc = 'center',colWidths = [0.05,0.2375,0.2375,0.2375,0.2375],bbox =[0, 0, 1, 1])

    table.set_fontsize(13)

    #removing axis
    ax.axis('off')

    #setting title
    ax.set_title(f"{k}% Confidence Interval Summary",{'font':'serif', 'size':14,'weight':'bold'})
```

90% Confidence Interval Summary

Male	CI = 8603 – 10275, Range = 1672	CI = 9172 – 9701, Range = 529	CI = 9319 – 9557, Range = 238	CI = 9401 – 9475, Range = 74
Female	CI = 7951 – 9500, Range = 1549	CI = 8490 – 8985, Range = 495	CI = 8626 – 8846, Range = 220	CI = 8700 – 8769, Range = 69

Marital Status Vs Purchase Amount

Data Visualization

```
#creating separate dataframe for Marital Status and Purchase Amount

Gdf_MS = df.groupby('Marital_Status')['Purchase'].agg(['sum','count']).reset_index()

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

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

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


Gdf_MS
```

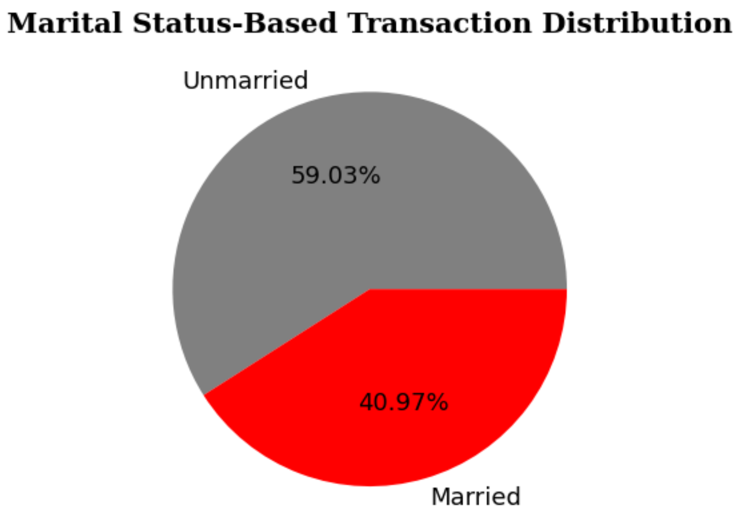
	Marital_Status	sum	count	sum(billions)	%sum	per_purchase
0	Unmarried	3008927447	324731	3.01	0.59	9266.0
1	Married	2086885295	225337	2.09	0.41	9261.0

```
#Distribution of Purchase Amount

fig = plt.figure(figsize=(16,2))

#plotting the visual
plt.barh(Gdf_MS.loc[0,'Marital_Status'],width = Gdf_MS.loc[0,'%sum'],color = "grey",label = 'Female')
plt.barh(Gdf_MS.loc[0,'Marital_Status'],width = Gdf_MS.loc[1,'%sum'],left =Gdf_MS.loc[0,'%sum'], color =  "red",label = 'Male' )

#inserting the text
txt = [0.0]

for i in Gdf_MS.index:
    #for amount
    plt.text(Gdf_MS.loc[i,'%sum']/2 + txt[0],0.15,f"${Gdf_MS.loc[i,'sum(billions)']} Billion",
           va = 'center', ha='center',fontsize=18, color='white')

    #for gender
    plt.text(Gdf_MS.loc[i,'%sum']/2 + txt[0],- 0.15 ,f"{Gdf_MS.loc[i,'Marital_Status']}",
           va = 'center', ha='center',fontsize=14, color='white')

    txt += Gdf_MS.loc[i,'%sum']

#removing the axis lines
plt.axis('off')

#plot title
plt.title('Marital Status-Based Purchase Amount Distribution',{'font':'serif', 'size':15,'weight':'bold'})

plt.show()
```

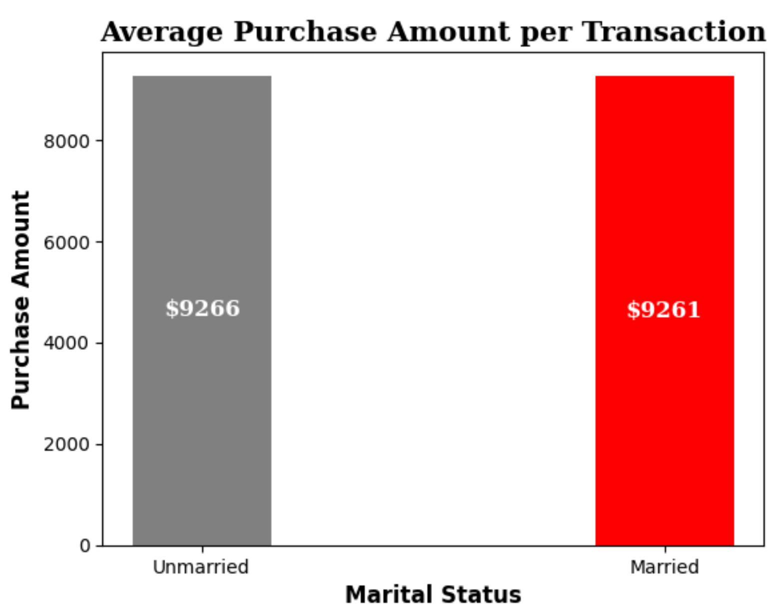
Marital Status-Based Purchase Amount Distribution


```
# creating pie chart for gender disribution

color_map = ["grey","red"]
plt.pie(Gdf_MS['count'],autopct = '%.2f%%', labels= Gdf_MS['Marital_Status'], colors=color_map, wedgeprops = {'linewidth': 5},textprops={'fon
```

```
#setting title for visual
plt.title('Marital Status-Based Transaction Distribution',{'font':'serif', 'size':15,'weight':'bold'})

plt.show()
```

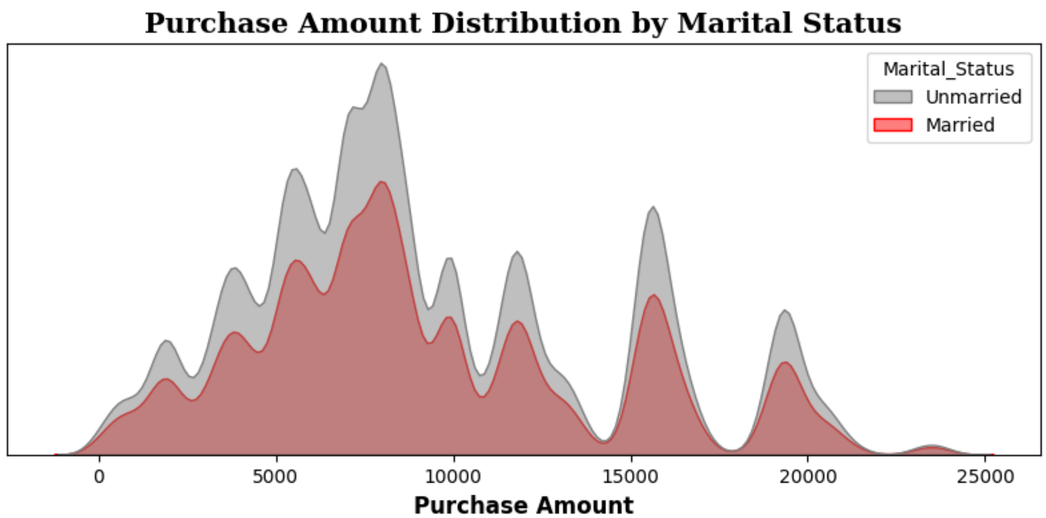



```
#Average perchase amount per transaction

plt.bar(Gdf_MS['Marital_Status'],Gdf_MS['per_purchase'],color = ['grey','red'],width = 0.3)

for i in Gdf_MS.index:
    plt.text(Gdf_MS.loc[i,'Marital_Status'],Gdf_MS.loc[i,'per_purchase']/2,f"${Gdf_MS.loc[i,'per_purchase']:.0f}",
             {'font':'serif','size' : 12,'color':'white','weight':'bold' },ha = 'center',va = 'center')

plt.ylabel('Purchase Amount',fontweight = 'bold',fontsize = 12)
plt.xlabel('Marital Status', fontweight = 'bold',fontsize = 12)

plt.title('Average Purchase Amount per Transaction',{'font':'serif', 'size':15,'weight':'bold'})

plt.show()
```




```
# creating kdeplot for purchase amount distribution

fig = plt.figure(figsize=(10,4))
sns.kdeplot(data = df, x = 'Purchase', hue = 'Marital_Status', palette = color_map, fill = True, alpha = 0.5)

# adjusting axis labels
plt.yticks([])
```

```
plt.ylabel('')
plt.xlabel('Purchase Amount',fontweight = 'bold',fontsize = 12)

#setting title for visual
plt.title('Purchase Amount Distribution by Marital Status',{'font':'serif', 'size':15,'weight':'bold'})

plt.show()
```

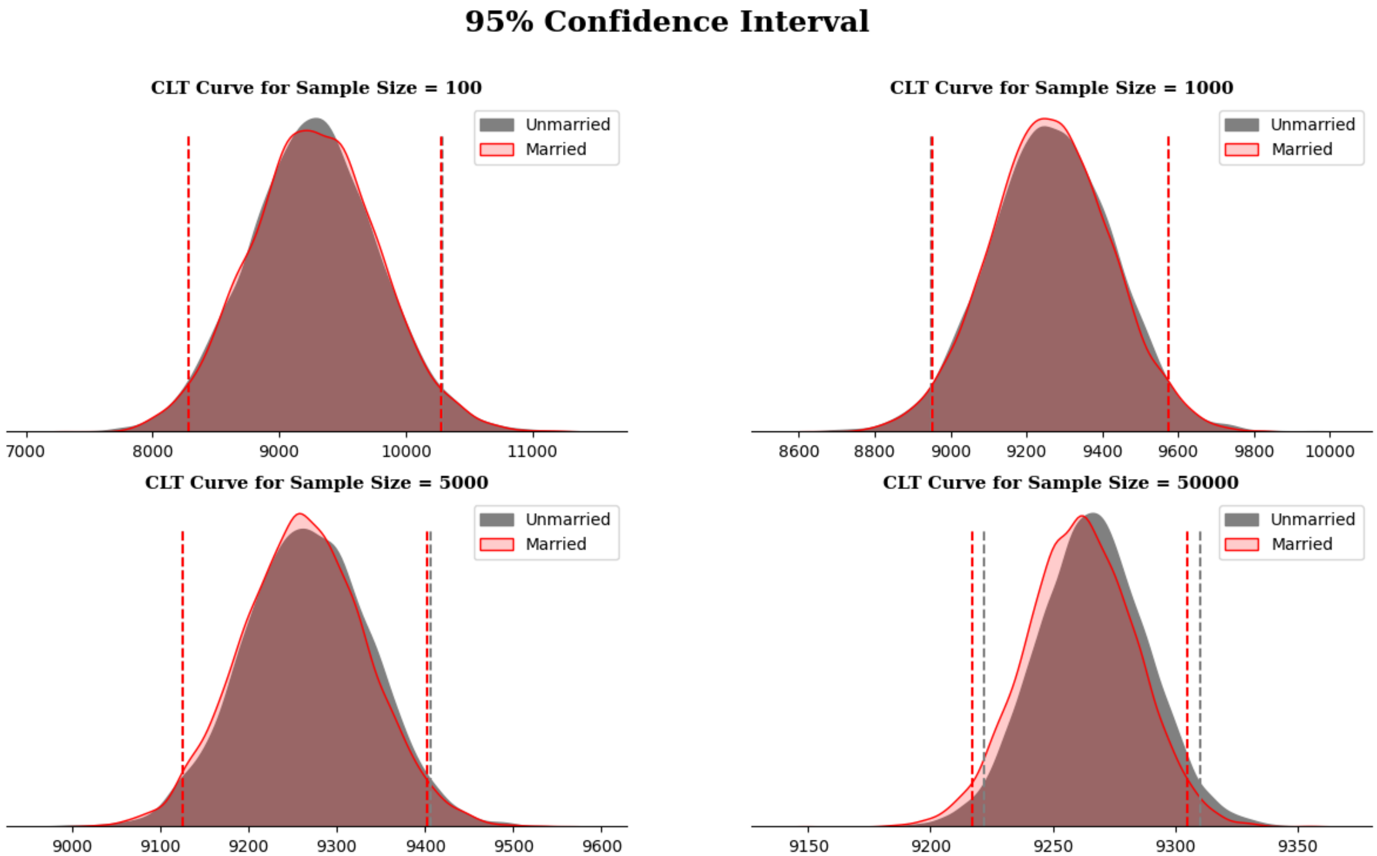


Confidence Interval and CLT

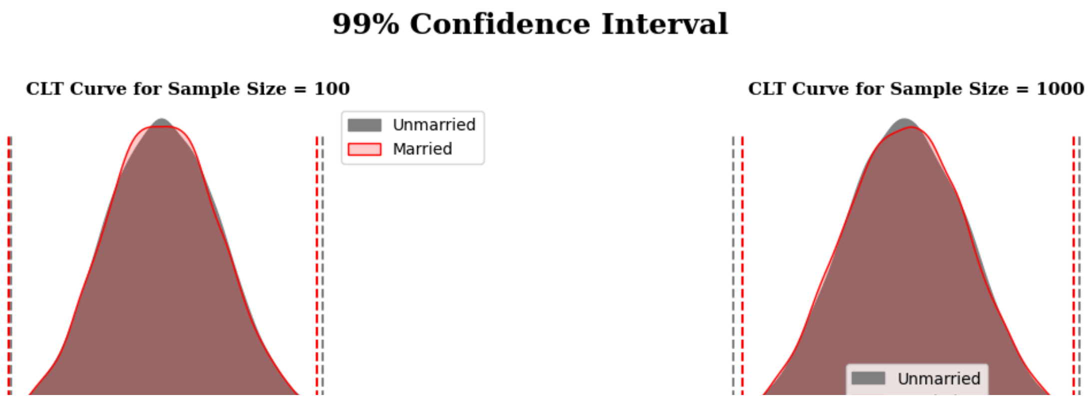
```
#defining a function for plotting the visual for given confidence interval

def plot(ci):

    #setting the plot style
    fig = plt.figure(figsize = (15,8))
    gs = fig.add_gridspec(2,2)

    #creating separate data frames for each gender
    df_unmarried = df.loc[df['Marital_Status'] == 'Unmarried','Purchase']
    df_married = df.loc[df['Marital_Status'] == 'Married','Purchase']

    #sample sizes and corresponding plot positions
    sample_sizes = [(100,0,0),(1000,0,1),(5000,1,0),(50000,1,1)]

    #number of samples to be taken from purchase amount
    boot_samples = 10000

    unmarried_samples = {}
    married_samples = {}

    for i,x,y in sample_sizes:
        unmarried_means = [] #list for collecting the means of male sample
        married_means = [] #list for collecting the means of female sample

        for j in range(boot_samples):

            #creating random 5000 samples of i sample size
            unmarried_boot_samples = np.random.choice(df_unmarried,size = i)
            married_boot_samples = np.random.choice(df_married,size = i)

            #calculating mean of those samples
            unmarried_sample_mean = np.mean(unmarried_boot_samples)
            married_sample_mean = np.mean(married_boot_samples)

            #appending the mean to the list
            unmarried_means.append(unmarried_sample_mean)
            married_means.append(married_sample_mean)

        #storing the above sample generated
        unmarried_samples[f'{ci}%_{i}'] = unmarried_means
        married_samples[f'{ci}%_{i}'] = married_means
```

```
        #creating a temporary dataframe for creating kdeplot
        temp_df_MS = pd.DataFrame(data = {'unmarried_means':unmarried_means,'married_means':married_means})

                                                #plotting kdeplots
        #plot position
        ax = fig.add_subplot(gs[x,y])

        #plots for male and female
        sns.kdeplot(data = temp_df_MS,x = 'unmarried_means',color ="grey" ,fill = True, alpha = 1,label = 'Unmarried')
        sns.kdeplot(data = temp_df_MS,x = 'married_means',color ="red" ,fill = True, alpha = 0.2,label = 'Married')

        #calculating confidence intervals for given confidence level(ci)
        un_range = confidence_interval(unmarried_means,ci)
        ma_range = confidence_interval(married_means,ci)

        #plotting confidence interval on the distribution
        for k in un_range:
            ax.axvline(x = k,ymax = 0.9, color ="grey",linestyle = '--')

        for k in ma_range:
            ax.axvline(x = k,ymax = 0.9, color ="red",linestyle = '--')


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

        # adjusting axis labels
        ax.set_yticks([])
        ax.set_ylabel('')
        ax.set_xlabel('')

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

        plt.legend()

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

    plt.show()

    return unmarried_samples,married_samples

unmarried_samples_90,married_samples_90 = plot(90)
```


```
unmarried_samples_95,married_samples_95 = plot(95)
```




```
unmarried_samples_99,married_samples_99 = plot(99)
```




```
fig = plt.figure(figsize = (20,12))
gs = fig.add_gridspec(3,1)

for i,j,k,l in [(unmarried_samples_90,married_samples_90,90,0),(unmarried_samples_95,married_samples_95,95,1),(unmarried_samples_99,married_s
    #list for collecting ci for given cl
    un_ci = ['Unmarried']
    ma_ci = ['Married']

    #finding ci for each sample size (unmarried)
    for m in i:
        un_range = confidence_interval(i[m],k)
        un_ci.append(f"CI = ${un_range[0]:.0f} - ${un_range[1]:.0f}, Range = {(un_range[1] - un_range[0]):.0f}")

    #finding ci for each sample size (married)
    for f in j:
        ma_range = confidence_interval(j[f],k)
        ma_ci.append(f"CI = ${ma_range[0]:.0f} - ${ma_range[1]:.0f}, Range = {(ma_range[1] - ma_range[0]):.0f}")

                                    #plotting the summary
    ax = fig.add_subplot(gs[l])

    #contents of the table
    ci_info = [un_ci,ma_ci]

    #plotting the table
    table = ax.table(cellText = ci_info, cellLoc='center',
                     colLabels =['Marital_Status','Sample Size = 100','Sample Size = 1000','Sample Size = 5000','Sample Size = 50000'],
                     colLoc = 'center',colWidths = [0.05,0.2375,0.2375,0.2375,0.2375],bbox =[0, 0, 1, 1])

    table.set_fontsize(25)

    #removing axis
    ax.axis('off')

    #setting title
    ax.set_title(f"{k}% Confidence Interval Summary",{'font':'serif', 'size':14,'weight':'bold'})
```

90% Confidence Interval Summary

Marital_Status	Sample Size = 100	Sample Size = 1000	Sample Size = 5000	Sample Size = 50000
Unmarried	CI = 8443 – 10090, Range = 1647	CI = 9006 – 9528, Range = 522	CI = 9150 – 9384, Range = 234	CI = 9229 – 9303, Range = 74
Married	CI = 8441 – 10103, Range = 1662	CI = 9001 – 9523, Range = 522	CI = 9145 – 9379, Range = 234	CI = 9225 – 9298, Range = 73

95% Confidence Interval Summary

Marital_Status	Sample Size = 100	Sample Size = 1000	Sample Size = 5000	Sample Size = 50000

Customer Age VS Purchase Amount

Data Visualization

```
#creating separate dataframe for Customer Age and Purchase Amount

Gdf_A = df.groupby('Age')['Purchase'].agg(['sum','count']).reset_index()

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

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

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

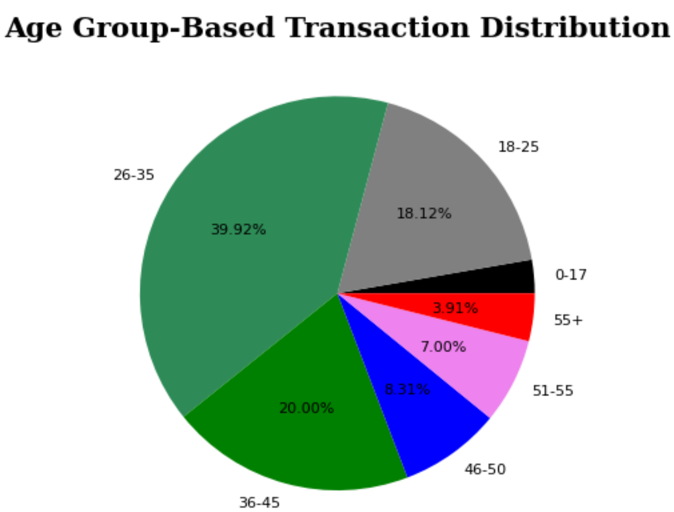
```
Gdf_A
```

	Age	sum	count	sum(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

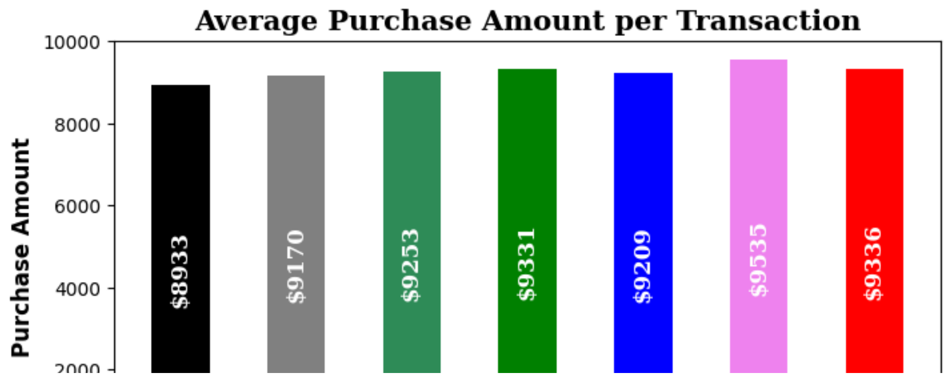
```
#Distribution of Purchase Amount

fig = plt.figure(figsize=(16,2))
color_map = ['black','grey','seagreen','green','blue','violet','red']

#plotting the visual
left = 0

for i in Gdf_A.index:
  plt.barh(Gdf_A.loc[0,'Age'],width = Gdf_A.loc[i,'%sum'],left=left, color = color_map[i], label = Gdf_A.loc[i,'Age'])
  left += Gdf_A.loc[i,'%sum']

#inserting the text
txt = [0.0]

for i in Gdf_A.index:
    #for amount
    plt.text(Gdf_A.loc[i,'%sum']/2 + txt[0],0.15,f"${Gdf_A.loc[i,'sum(billions)']} B",
          va = 'center', ha='center',fontsize=12, color='white',rotation=90)

    #for age grp
    plt.text(Gdf_A.loc[i,'%sum']/2 + txt,- 0.20 ,f"{Gdf_A.loc[i,'Age']}",
```

```
             va = 'center', ha='center',fontsize=12, color='white')
        txt += Gdf_A.loc[i,'%sum']

#removing the axis lines
plt.axis('off')

#plot title
plt.title('Age Group Purchase Amount Distribution',{'font':'serif', 'size':15,'weight':'bold'})

plt.show()
```

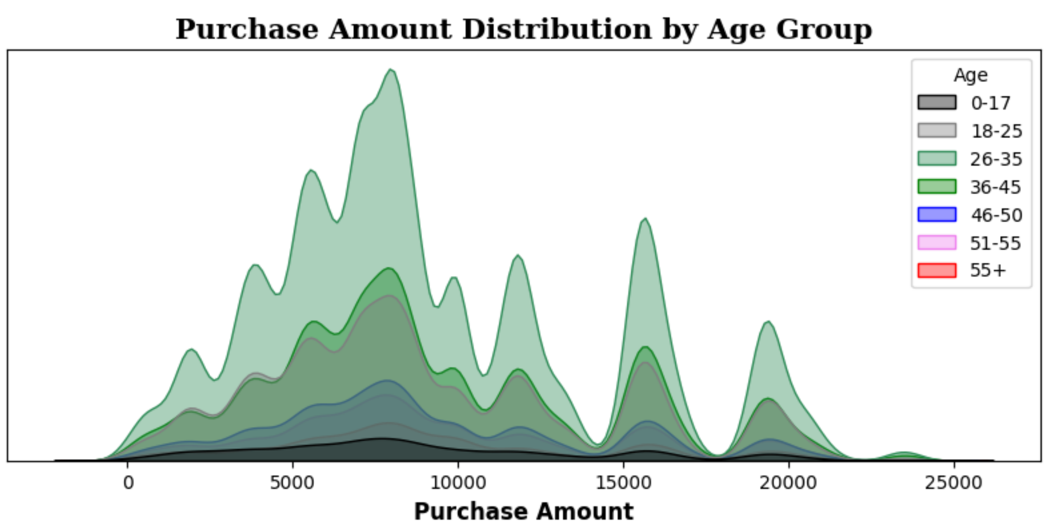

```
# creating pie chart for Age disribution

plt.pie(Gdf_A['count'],autopct = '%.2f%%', labels= Gdf_A['Age'], colors=color_map, wedgeprops = {'linewidth': 5},textprops={'fontsize': 8, 'c

#setting title for visual
plt.title('Age Group-Based Transaction Distribution',{'font':'serif', 'size':15,'weight':'bold'})

plt.show()
```




```
#Average perchase amount per transaction
fig = plt.figure(figsize=(8,4))

plt.bar(Gdf_A['Age'],Gdf_A['per_purchase'],color = color_map,zorder = 2,width = 0.5)

for i in Gdf_A.index:
    plt.text(Gdf_A.loc[i,'Age'],Gdf_A.loc[i,'per_purchase']/2,f"${Gdf_A.loc[i,'per_purchase']:.0f}",
             {'font':'serif','size' : 12,'color':'white','weight':'bold' },ha = 'center',va = 'center',rotation=90)

plt.ylabel('Purchase Amount',fontweight = 'bold',fontsize = 12)
plt.xlabel('Age Group', fontweight = 'bold',fontsize = 12)

plt.title('Average Purchase Amount per Transaction',{'font':'serif', 'size':15,'weight':'bold'})

plt.show()
```




```
# creating kdeplot for purchase amount distribution

fig = plt.figure(figsize=(10,4))

sns.kdeplot(data = df, x = 'Purchase', hue = 'Age', palette = color_map, fill = True, alpha = 0.4)

# adjusting axis labels
plt.yticks([])
plt.ylabel('')
plt.xlabel('Purchase Amount',fontweight = 'bold',fontsize = 12)

#setting title for visual
plt.title('Purchase Amount Distribution by Age Group',{'font':'serif', 'size':15,'weight':'bold'})

plt.show()
```



Confidence Interval And CLT

```
def plot(ci):

    #setting the plot style
    fig = plt.figure(figsize = (12,15))
    gs = fig.add_gridspec(4,1)

    #creating separate data frames

    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 and corresponding plot positions
    sample_sizes = [(100,0),(1000,1),(5000,2),(50000,3)]
```

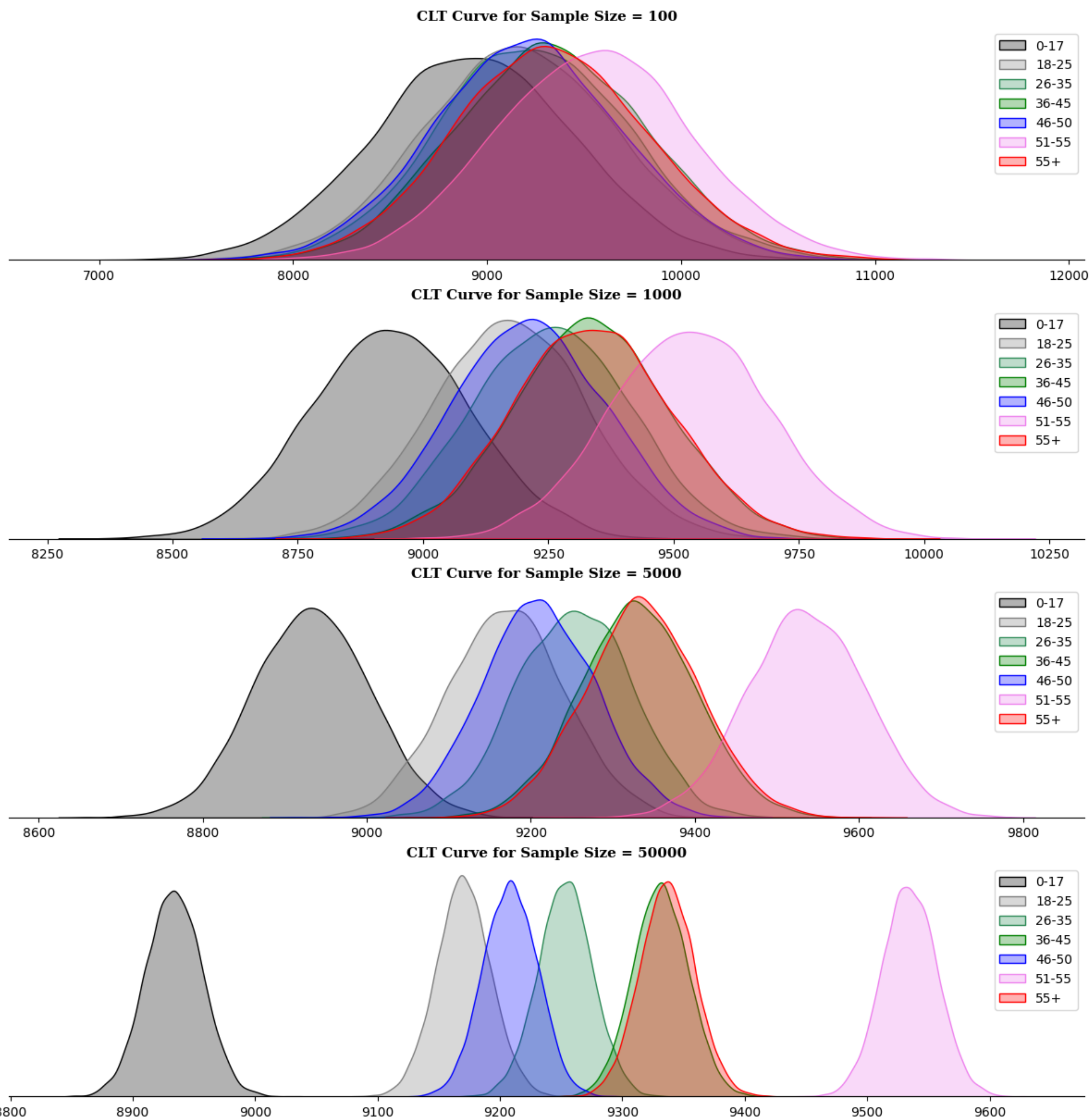
```
#number of samples to be taken from purchase amount
bootstrap_samples = 10000

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):

        #creating random 5000 samples of i sample size
        boot_samples_1 = np.random.choice(df_1,size = i)
        boot_samples_2 = np.random.choice(df_2,size = i)
        boot_samples_3 = np.random.choice(df_3,size = i)
        boot_samples_4 = np.random.choice(df_4,size = i)
        boot_samples_5 = np.random.choice(df_5,size = i)
        boot_samples_6 = np.random.choice(df_6,size = i)
        boot_samples_7 = np.random.choice(df_7,size = i)

        #calculating mean of those samples
        sample_mean_1 = np.mean(boot_samples_1)
        sample_mean_2 = np.mean(boot_samples_2)
        sample_mean_3 = np.mean(boot_samples_3)
        sample_mean_4 = np.mean(boot_samples_4)
        sample_mean_5 = np.mean(boot_samples_5)
        sample_mean_6 = np.mean(boot_samples_6)
        sample_mean_7 = np.mean(boot_samples_7)

        #appending the mean to the list
        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)

    #storing the above sample generated
    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_A = 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
    #plot position
    ax = fig.add_subplot(gs[x])

    #plots
    for p,q in [('black', '0-17'),('grey', '18-25'),('seagreen', '26-35'),('green', '36-45'),('blue', '46-50'),
             ('violet', '51-55'),('red', '55+')]:

        sns.kdeplot(data = temp_df_A,x = q,color =p ,fill = True, alpha = 0.3,ax = ax,label = q)


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

    # adjusting axis labels
    ax.set_yticks([])
    ax.set_ylabel('')
    ax.set_xlabel('')

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

    plt.legend()

#setting title for visual
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(90)
```

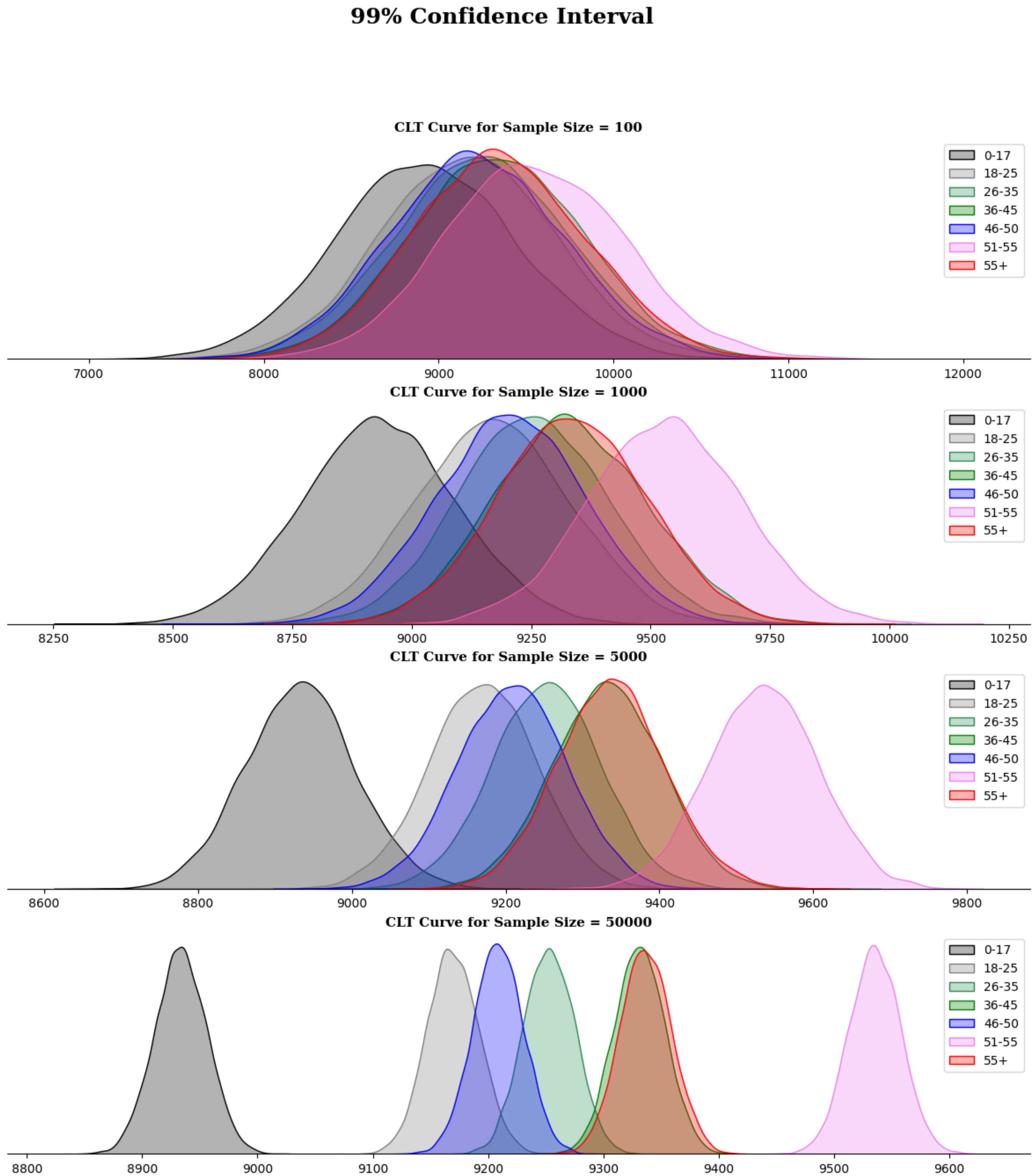
90% Confidence Interval

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

95% Confidence Interval



```
samples1,samples2,samples3,samples4,samples5,samples6,samples7 = plot(99)
```




```
def confidence_interval_summary(x):
```

```
    fig = plt.figure(figsize = (18,12))

    s1_ci,s2_ci,s3_ci,s4_ci,s5_ci,s6_ci,s7_ci = ['0-17'],['18-25'],['26-35'],['36-45'],['46-50'],['51-55'],['55+']

    samples = [(samples1,s1_ci),(samples2,s2_ci),(samples3,s3_ci),(samples4,s4_ci),(samples5,s5_ci),(samples6,s6_ci),(samples7,s7_ci)]

    #finding ci for each sample size
    for s,c in samples:
      for i in s:
        s_range = confidence_interval(s[i],x)
        c.append(f"CI = ${s_range[0]:.0f} - ${s_range[1]:.0f}, Range = {(s_range[1] - s_range[0]):.0f}")

                                        #plotting the summary
    ax = fig.add_subplot(gs[1])

    #contents of the table
    ci_info = [s1_ci,s2_ci,s3_ci,s4_ci,s5_ci,s6_ci,s7_ci]

    #plotting the table
    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)

    #removing axis
    ax.axis('off')

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

    return None

  for i in [90,95,99]:
    confidence_interval_summary(i)
```

90% Confidence Interval Summary

Age Group	Sample Size = 100	Sample Size = 1000	Sample Size = 5000	Sample Size = 50000
0-17	CI = 8114 − 9782, Range = 1668	CI = 8664 − 9200, Range = 536	CI = 8815 − 9052, Range = 237	CI = 8896 − 8971, Range = 75
18-25	CI = 8341 − 9999, Range = 1658	CI = 8910 − 9431, Range = 521	CI = 9052 − 9288, Range = 236	CI = 9131 − 9207, Range = 76
26-35	CI = 8420 − 10083, Range = 1663	CI = 8991 − 9515, Range = 524	CI = 9139 − 9370, Range = 231	CI = 9216 − 9289, Range = 73
36-45	CI = 8506 − 10176, Range = 1670	CI = 9076 − 9587, Range = 511	CI = 9215 − 9448, Range = 233	CI = 9294 − 9369, Range = 75
46-50	CI = 8417 − 10026, Range = 1609	CI = 8948 − 9469, Range = 521	CI = 9093 − 9325, Range = 232	CI = 9172 − 9246, Range = 74
51-55	CI = 8708 − 10390, Range = 1682	CI = 9268 − 9800, Range = 532	CI = 9416 − 9652, Range = 236	CI = 9498 − 9572, Range = 74
55+	CI = 8518 − 10176, Range = 1658	CI = 9078 − 9597, Range = 519	CI = 9221 − 9453, Range = 232	CI = 9299 − 9374, Range = 75

95% Confidence Interval Summary

Age Group	Sample Size = 100	Sample Size = 1000	Sample Size = 5000	Sample Size = 50000
0-17	CI = 7946 − 9949, Range = 2003	CI = 8614 − 9248, Range = 634	CI = 8793 − 9077, Range = 284	CI = 8889 − 8979, Range = 90
18-25	CI = 8186 − 10178, Range = 1992	CI = 8856 − 9481, Range = 625	CI = 9030 − 9311, Range = 281	CI = 9125 − 9214, Range = 89
26-35	CI = 8282 − 10264, Range = 1982	CI = 8945 − 9568, Range = 623	CI = 9116 − 9394, Range = 278	CI = 9209 − 9297, Range = 88
36-45	CI = 8358 − 10339, Range = 1981	CI = 9030 − 9643, Range = 613	CI = 9194 − 9471, Range = 277	CI = 9287 − 9376, Range = 89