```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import t
import warnings
warnings.filterwarnings('ignore')
import copy
```

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

```
df.head()
```

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

Next steps: View recommended plots

```
df.tail()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_C
50075	1001667	P00145942	M	51-55	16	B	
50076	1001667	P00044442	M	51-55	16	B	
50077	1001667	P00036842	M	51-55	16	B	

```
df.shape
```

```
(50080, 10)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50080 entries, 0 to 50079
Data columns (total 10 columns):
 #   Column                      Non-Null Count  Dtype
 ---  ------                      --------------  -----  
  0   User_ID                     50080 non-null  int64  
  1   Product_ID                  50080 non-null  object 
  2   Gender                      50080 non-null  object 
  3   Age                         50080 non-null  object 
  4   Occupation                  50080 non-null  int64  
  5   City_Category               50080 non-null  object 
  6   Stay_In_Current_City_Years  50080 non-null  object 
  7   Marital_Status              50080 non-null  int64  
  8   Product_Category            50079 non-null  float64
  9   Purchase                    50079 non-null  float64
 dtypes: float64(2), int64(3), object(5)
 memory usage: 3.8+ MB
```

Changing the Datatype of Columns

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

```
df.info()
```

```
 <class 'pandas.core.frame.DataFrame'>
 RangeIndex: 50080 entries, 0 to 50079
 Data columns (total 10 columns):
  #   Column                      Non-Null Count  Dtype   
 ---  ------                      --------------  -----   
  0   User_ID                     50080 non-null  category
  1   Product_ID                  50080 non-null  category
  2   Gender                      50080 non-null  category
  3   Age                         50080 non-null  category
  4   Occupation                  50080 non-null  category
  5   City_Category               50080 non-null  category
  6   Stay_In_Current_City_Years  50080 non-null  category
  7   Marital_Status              50080 non-null  category
  8   Product_Category            50079 non-null  category
  9   Purchase                    50079 non-null  float64 
 dtypes: category(9), float64(1)
 memory usage: 1.2 MB
```

Satistical summary of object type columns

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

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Curren
count	50080	50080	50080	50080	50080	50080	
unique	5426	3098	2	7	21	3	
top	1000889	P00265242	M	26-35	4	B	
freq	137	159	37868	19755	6543	20777	

Satistical summary of numerical data type columns

```
df.describe()
```

	Purchase
count	225389.000000
mean	9318.194331
std	4971.776715
min	185.000000
25%	5860.000000
50%	8059.000000
75%	12061.000000
max	23961.000000

Duplicate Detection

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

```
False    50080
dtype: int64
```

Double-click (or enter) to edit

Sanity Check for columns

```
for i in df.columns:
 print('Unique Values in',i,'column are :-')
 print(df[i].unique())
 print('-'*70)
```

```
Unique Values in User_ID column are :-
[1000001, 1000002, 1000003, 1000004, 1000005, ..., 1001621, 1001633, 1001638, 100165
Length: 5426
Categories (5426, int64): [1000001, 1000002, 1000003, 1000004, ..., 1006036, 1006037,
----------------------------------------------------------------------
Unique Values in Product_ID column are :-
['P00069042', 'P00248942', 'P00087842', 'P00085442', 'P00285442', ..., 'P00262042', '
Length: 3098
Categories (3098, object): ['P00000142', 'P00000242', 'P00000342', 'P00000442', ...,
                            '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 :-
[3.0, 1.0, 12.0, 8.0, 5.0, ..., 18.0, 10.0, 17.0, 9.0, NaN]
Length: 19
Categories (18, float64): [1.0, 2.0, 3.0, 4.0, ..., 15.0, 16.0, 17.0, 18.0]
----------------------------------------------------------------------
Unique Values in Purchase column are :-
[ 8370. 15200.  1422. ... 15927. 20056.    nan]
----------------------------------------------------------------------
<ipython-input-15-5317717e4240>:3: FutureWarning: Index.ravel returning ndarray is de
  print(df[i].unique())
```

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

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

Missing Value Analysis

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

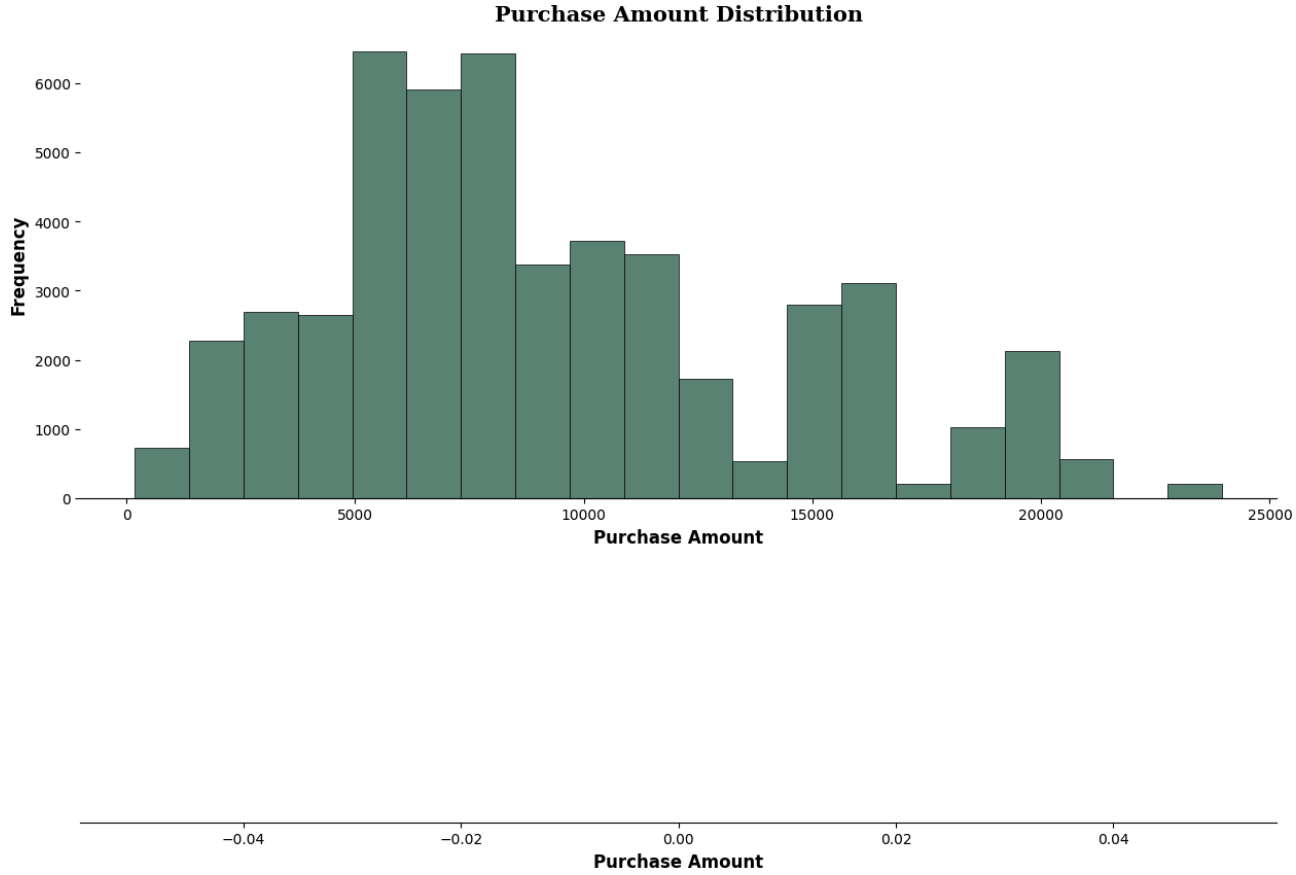
```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category              1
Purchase                      1
dtype: int64
```

Univariate Analysis

Numerical Variables

Purchase Amount Distribution

```
#setting the plot style
fig = plt.figure(figsize = (15,10))
gs = fig.add_gridspec(2,1,height_ratios=[0.65, 0.35])
 #creating purchase amount histogram

ax0 = fig.add_subplot(gs[0,0])
ax0.hist(df['Purchase'],color= '#5C8374',linewidth=0.5,edgecolor='black',bins = 20)
ax0.set_xlabel('Purchase Amount',fontsize = 12,fontweight = 'bold')
ax0.set_ylabel('Frequency',fontsize = 12,fontweight = 'bold')
#removing the axis lines
for s in ['top','left','right']:
 ax0.spines[s].set_visible(False)

#setting title for visual
ax0.set_title('Purchase Amount Distribution',{'font':'serif', 'size':15,'weight':'bold'})

 #creating box plot for purchase amount

ax1 = fig.add_subplot(gs[1,0])
boxplot = ax1.boxplot(x = df['Purchase'],vert = False,patch_artist = True,widths = 0.5)
# Customize box and whisker colors
boxplot['boxes'][0].set(facecolor='#5C8374')
# Customize median line
boxplot['medians'][0].set(color='red')
# Customize outlier markers
for flier in boxplot['fliers']:
 flier.set(marker='o', markersize=8, markerfacecolor= "#4b4b4c")

#removing the axis lines
for s in ['top','left','right']:
 ax1.spines[s].set_visible(False)
#adding 5 point summary annotations
info = [i.get_xdata() for i in boxplot['whiskers']] #getting the upperlimit,Q1,Q3 and low
median = df['Purchase'].quantile(0.5) #getting Q2
for i,j in info: #using i,j here because of the output type of info list comprehension

 ax1.annotate(text = f"{i:.1f}", xy = (i,1), xytext = (i,1.4),fontsize = 12,
 arrowprops= dict(arrowstyle="<-", lw=1, connectionstyle="arc,rad=0"))

 ax1.annotate(text = f"{j:.1f}", xy = (j,1), xytext = (j,1.4),fontsize = 12,
 arrowprops= dict(arrowstyle="<-", lw=1, connectionstyle="arc,rad=0"))
#adding the median separately because it was included in info list
ax1.annotate(text = f"{median:.1f}",xy = (median,1),xytext = (median + 1,1.4),fontsize =
 arrowprops= dict(arrowstyle="<-", lw=1, connectionstyle="arc,rad=0"))
#removing y-axis ticks
ax1.set_yticks([])
#adding axis label
ax1.set_xlabel('Purchase Amount',fontweight = 'bold',fontsize = 12)
plt.show()
```



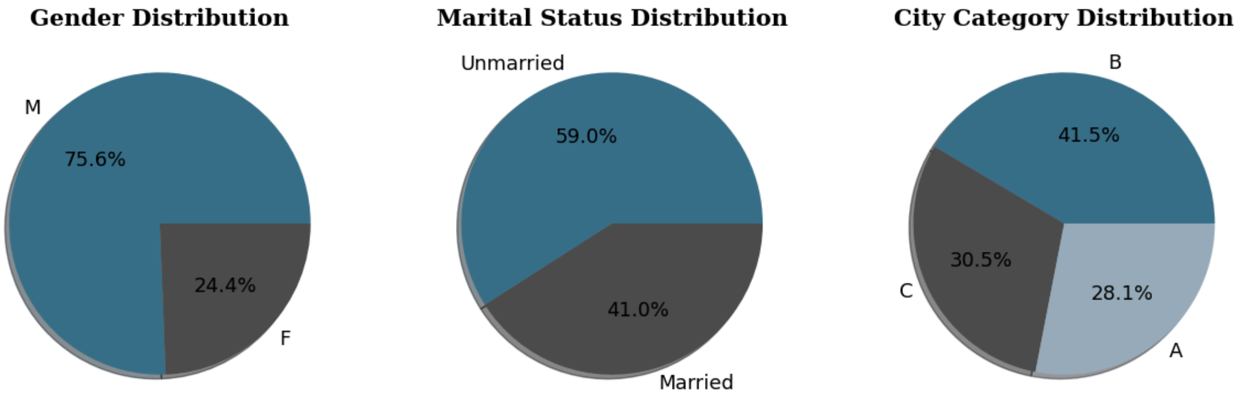
Calculating the Number of Outliers

```
len(df.loc[df['Purchase'] > 21399,'Purchase'])
```

```
241
```

Categorical Variables

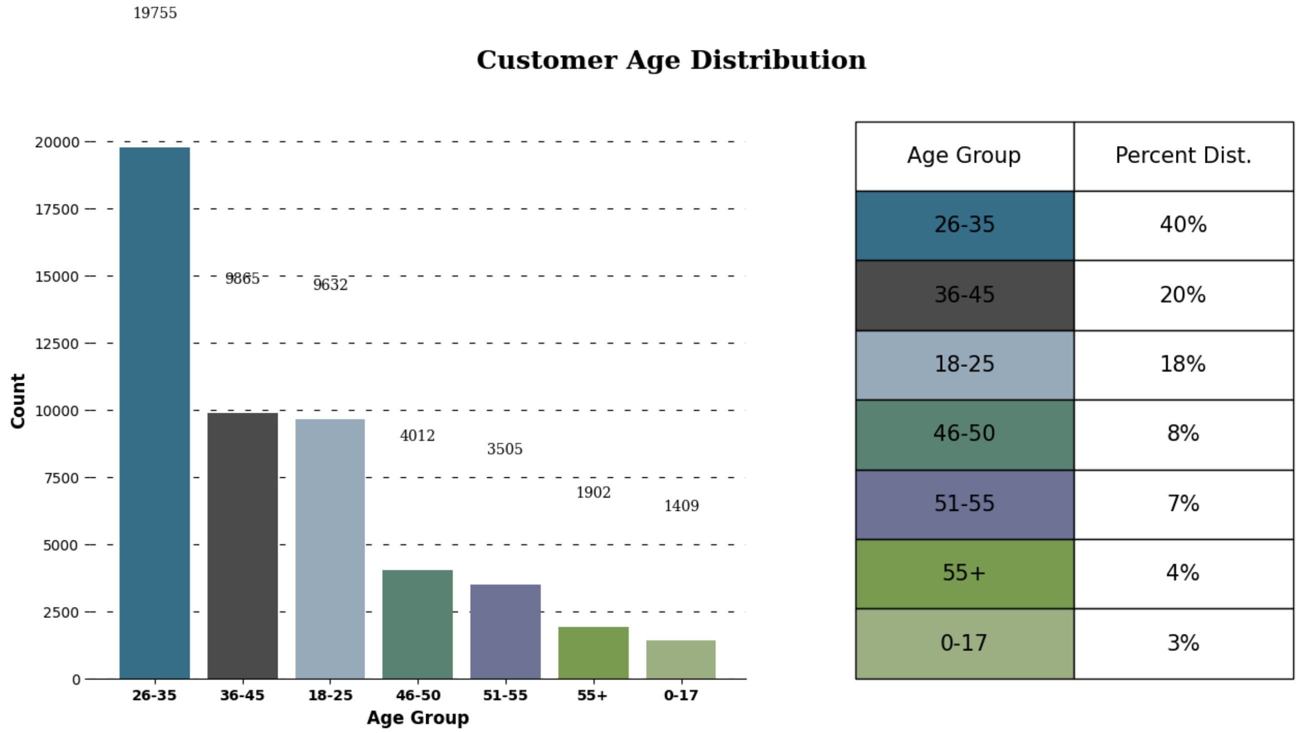
```
#setting the plot style
fig = plt.figure(figsize = (15,12))
gs = fig.add_gridspec(1,3)
 # creating pie chart for gender disribution
ax0 = fig.add_subplot(gs[0,0])
color_map = ["#3A7089", "#4b4b4c"]
ax0.pie(df['Gender'].value_counts().values,labels = df['Gender'].value_counts().index,aut
 shadow = True,colors = color_map,textprops={'fontsize': 13, 'color': 'black'})
#setting title for visual
ax0.set_title('Gender Distribution',{'font':'serif', 'size':15,'weight':'bold'})
 # creating pie chart for marital status
ax1 = fig.add_subplot(gs[0,1])
color_map = ["#3A7089", "#4b4b4c"]
ax1.pie(df['Marital_Status'].value_counts().values,labels = df['Marital_Status'].value_cc
 shadow = True,colors = color_map,textprops={'fontsize': 13, 'color': 'black'})
#setting title for visual
ax1.set_title('Marital Status Distribution',{'font':'serif', 'size':15,'weight':'bold'})
 # creating pie chart for city category
ax1 = fig.add_subplot(gs[0,2])
color_map = ["#3A7089", "#4b4b4c",'#99AEBB']
ax1.pie(df['City_Category'].value_counts().values,labels = df['City_Category'].value_coun
 shadow = True,colors = color_map,textprops={'fontsize': 13, 'color': 'black'})
#setting title for visual
ax1.set_title('City Category Distribution',{'font':'serif', 'size':15,'weight':'bold'})
plt.show()
```



Customer Age Distribution

```
#setting the plot style
fig = plt.figure(figsize = (15,7))
gs = fig.add_gridspec(1,2,width_ratios=[0.6, 0.4])
 # creating bar chart for age disribution

ax0 = fig.add_subplot(gs[0,0])
temp = df['Age'].value_counts()
color_map = ["#3A7089", "#4b4b4c",'#99AEBB','#5C8374','#6F7597','#7A9D54','#9EB384']
ax0.bar(x=temp.index,height = temp.values,color = color_map,zorder = 2)
#adding the value_counts
for i in temp.index:
 ax0.text(i,temp[i]+5000,temp[i],{'font':'serif','size' : 10},ha = 'center',va = 'center'
#adding grid lines
ax0.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes = (5,10))
#removing the axis lines
for s in ['top','left','right']:
 ax0.spines[s].set_visible(False)

#adding axis label
ax0.set_ylabel('Count',fontweight = 'bold',fontsize = 12)
ax0.set_xlabel('Age Group',fontweight = 'bold',fontsize = 12)
ax0.set_xticklabels(temp.index,fontweight = 'bold')
 #creating a info table for age

ax1 = fig.add_subplot(gs[0,1])
age_info = age_info = [['26-35','40%'],['36-45','20%'],['18-25','18%'],['46-50','8%'],['5
 ['0-17','3%']]
color_2d = [["#3A7089",'#FFFFFF'],["#4b4b4c",'#FFFFFF'],['#99AEBB','#FFFFFF'],['#5C8374',
 ['#7A9D54','#FFFFFF'],['#9EB384','#FFFFFF']]
table = ax1.table(cellText = age_info, cellColours=color_2d, cellLoc='center',colLabels =
 colLoc = 'center',bbox =[0, 0, 1, 1])
table.set_fontsize(15)
#removing axis
ax1.axis('off')
#setting title for visual
fig.suptitle('Customer Age Distribution',font = 'serif', size = 18, weight = 'bold')
plt.show()
```

```
<ipython-input-21-39a9eb623ad6>:22: UserWarning: FixedFormatter should only be used t
  ax0.set_xticklabels(temp.index,fontweight = 'bold')
```

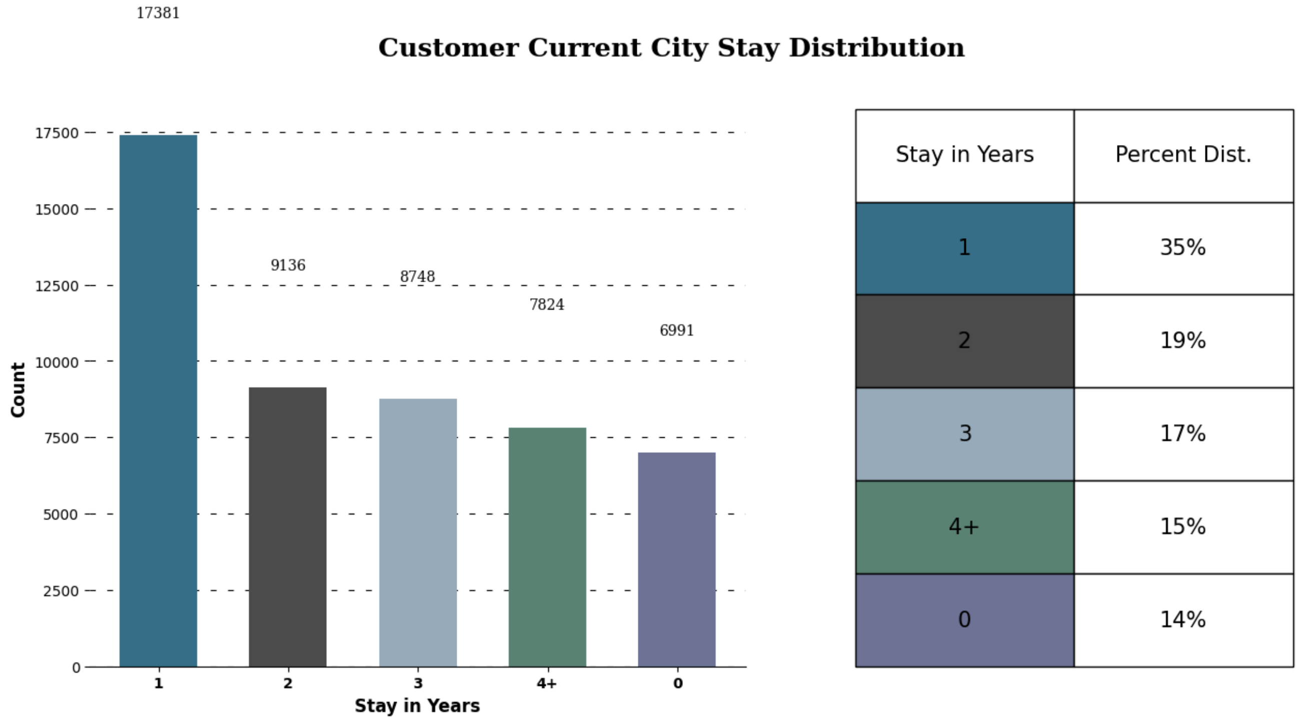


Age Group	Percent Dist.
26-35	40%
36-45	20%
18-25	18%
46-50	8%
51-55	7%
55+	4%
0-17	3%

Customer Stay In current City Distribution

```
#setting the plot style
fig = plt.figure(figsize = (15,7))
gs = fig.add_gridspec(1,2,width_ratios=[0.6, 0.4])
 # creating bar chart for Customer Stay In current City

ax1 = fig.add_subplot(gs[0,0])
temp = df['Stay_In_Current_City_Years'].value_counts()
color_map = ["#3A7089", "#4b4b4c",'#99AEBB','#5C8374','#6F7597']
ax1.bar(x=temp.index,height = temp.values,color = color_map,zorder = 2,width = 0.6)
#adding the value_counts
for i in temp.index:
 ax1.text(i,temp[i]+4000,temp[i],{'font':'serif','size' : 10},ha = 'center',va = 'center'
#adding grid lines
ax1.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes = (5,10))
#removing the axis lines
for s in ['top','left','right']:
 ax1.spines[s].set_visible(False)

#adding axis label
ax1.set_ylabel('Count',fontweight = 'bold',fontsize = 12)
ax1.set_xlabel('Stay in Years',fontweight = 'bold',fontsize = 12)
ax1.set_xticklabels(temp.index,fontweight = 'bold')
 #creating a info table for Customer Stay In current City

ax2 = fig.add_subplot(gs[0,1])
stay_info = [['1','35%'],['2','19%'],['3','17%'],['4+','15%'],['0','14%']]
color_2d = [["#3A7089",'#FFFFFF'],["#4b4b4c",'#FFFFFF'],['#99AEBB','#FFFFFF'],['#5C8374',
table = ax2.table(cellText = stay_info, cellColours=color_2d, cellLoc='center',colLabels
 colLoc = 'center',bbox =[0, 0, 1, 1])
table.set_fontsize(15)
#removing axis
ax2.axis('off')
#setting title for visual
fig.suptitle('Customer Current City Stay Distribution',font = 'serif', size = 18, weight
plt.show()
```

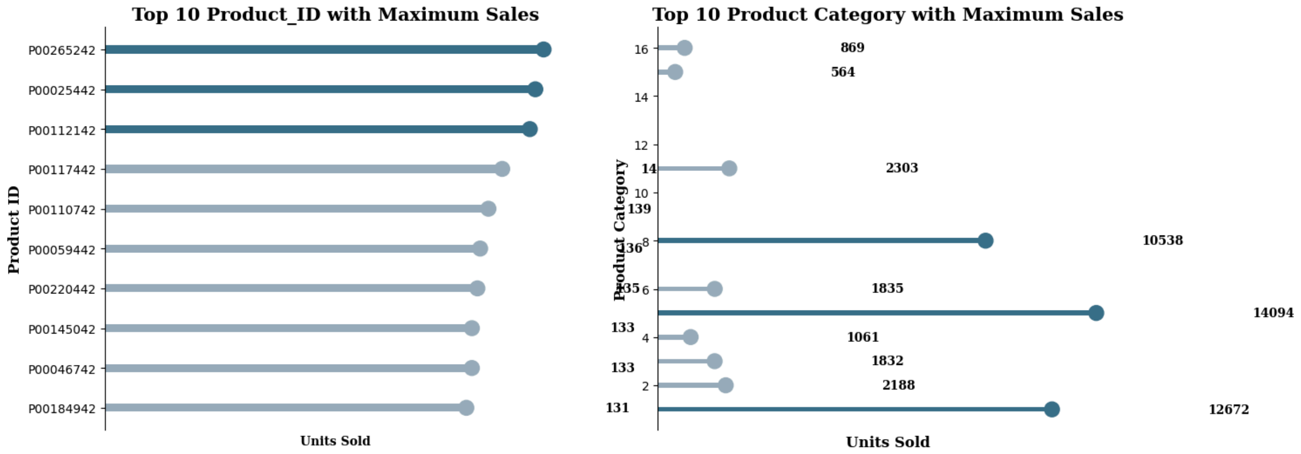
```
<ipython-input-22-3f7de575671a>:22: UserWarning: FixedFormatter should only be used t
  ax1.set_xticklabels(temp.index,fontweight = 'bold')
```



Stay in Years	Percent Dist.
1	35%
2	19%
3	17%
4+	15%
0	14%

Top 10 Products and Categories: Sales Snapshot

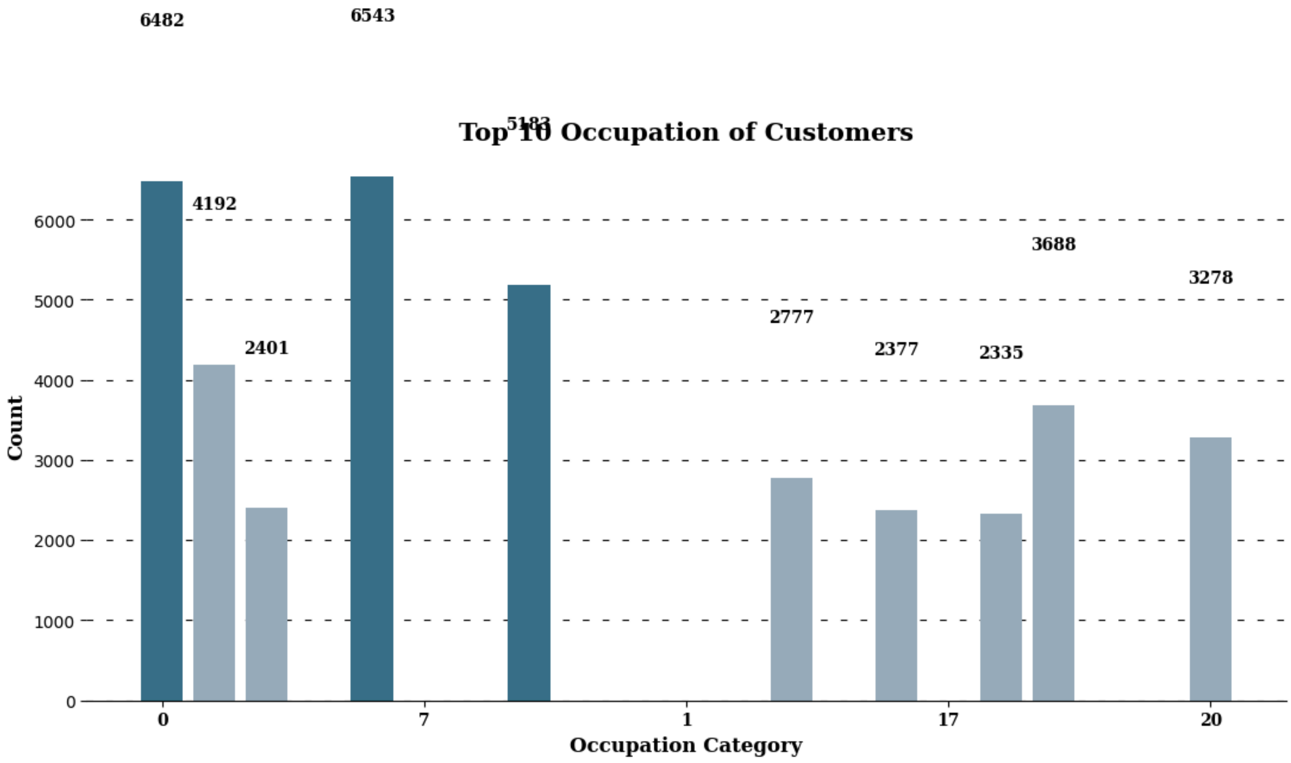
```
#setting the plot style
fig = plt.figure(figsize = (15,6))
gs = fig.add_gridspec(1,2)
 #Top 10 Product_ID Sales
ax = fig.add_subplot(gs[0,0])
temp = df['Product_ID'].value_counts()[0:10]
# reversing the list
temp = temp.iloc[-1:-11:-1]
color_map = ['#99AEBB' for i in range(7)] + ["#3A7089" for i in range(3)]
#creating the plot
ax.barh(y = temp.index,width = temp.values,height = 0.2,color = color_map)
ax.scatter(y = temp.index, x = temp.values, s = 150 , color = color_map )
#removing x-axis
ax.set_xticks([])
#adding label to each bar
for y,x in zip(temp.index,temp.values):
 ax.text( x + 50 , y , x,{'font':'serif', 'size':10,'weight':'bold'},va='center')
#removing the axis lines
for s in ['top','bottom','right']:
 ax.spines[s].set_visible(False)

#adding axis labels
ax.set_xlabel('Units Sold',{'font':'serif', 'size':10,'weight':'bold'})
ax.set_ylabel('Product ID',{'font':'serif', 'size':12,'weight':'bold'})
#creating the title
ax.set_title('Top 10 Product_ID with Maximum Sales',
 {'font':'serif', 'size':15,'weight':'bold'})
 #Top 10 Product Category Sales
ax = fig.add_subplot(gs[0,1])
temp = df['Product_Category'].value_counts()[0:10]
# reversing the list
temp = temp.iloc[-1:-11:-1]
#creating the plot
ax.barh(y = temp.index,width = temp.values,height = 0.2,color = color_map)
ax.scatter(y = temp.index, x = temp.values, s = 150 , color = color_map )
#removing x-axis
ax.set_xticks([])
#adding label to each bar
for y,x in zip(temp.index,temp.values):
 ax.text( x + 5000 , y , x,{'font':'serif', 'size':10,'weight':'bold'},va='center')
#removing the axis lines
for s in ['top','bottom','right']:
 ax.spines[s].set_visible(False)

#adding axis labels
ax.set_xlabel('Units Sold',{'font':'serif', 'size':12,'weight':'bold'})
ax.set_ylabel('Product Category',{'font':'serif', 'size':12,'weight':'bold'})
#creating the title
ax.set_title('Top 10 Product Category with Maximum Sales',
 {'font':'serif', 'size':15,'weight':'bold'})
plt.show()
```



Top 10 Customer Occupation

```
temp = df['Occupation'].value_counts()[0:10]
#setting the plot style
fig,ax = plt.subplots(figsize = (13,6))
color_map = ["#3A7089" for i in range(3)] + ['#99AEBB' for i in range(7)]
#creating the plot
ax.bar(temp.index,temp.values,color = color_map,zorder = 2)
#adding valuecounts
for x,y in zip(temp.index,temp.values):
 ax.text(x, y + 2000, y,{'font':'serif', 'size':10,'weight':'bold'},va='center',ha = 'cen

#setting grid style
ax.grid(color = 'black',linestyle = '--',axis = 'y',zorder = 0,dashes = (5,10))
#customizing the axis labels
ax.set_xticklabels(temp.index,fontweight = 'bold',fontfamily='serif')
ax.set_xlabel('Occupation Category',{'font':'serif', 'size':12,'weight':'bold'})
ax.set_ylabel('Count',{'font':'serif', 'size':12,'weight':'bold'})
#removing the axis lines
for s in ['top','left','right']:
 ax.spines[s].set_visible(False)

#adding title to the visual
ax.set_title('Top 10 Occupation of Customers',
 {'font':'serif', 'size':15,'weight':'bold'})
plt.show()
```

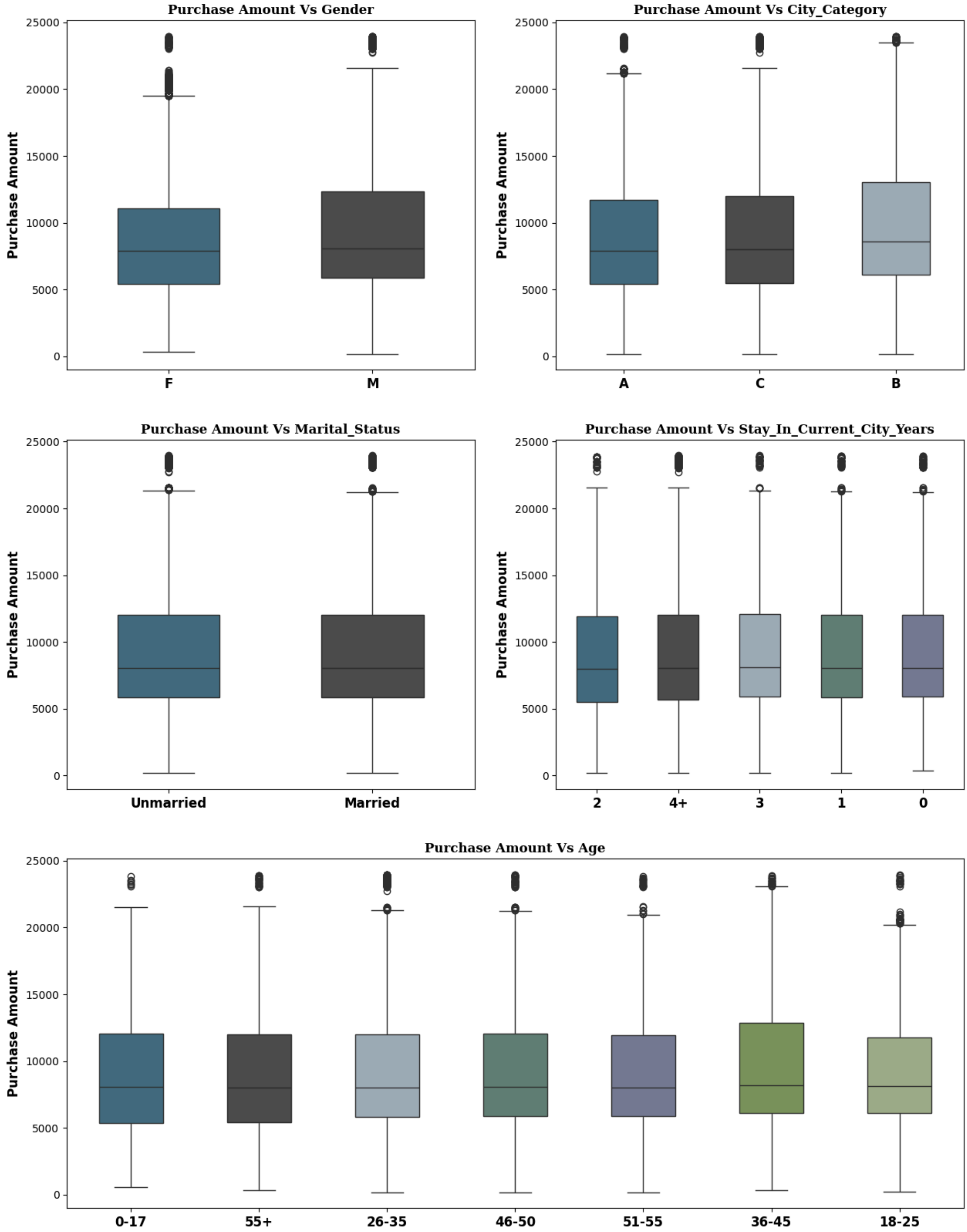
```
<ipython-input-24-856f48c44637>:14: UserWarning: FixedFormatter should only be used t
  ax.set_xticklabels(temp.index,fontweight = 'bold',fontfamily='serif')
```



Bivariate Analysis

Exploring Purchase Patterns

```
#setting the plot style
fig = plt.figure(figsize = (15,20))
gs = fig.add_gridspec(3,2)
for i,j,k in [(0,0,'Gender'),(0,1,'City_Category'),(1,0,'Marital_Status'),(1,1,'Stay_In_C

        #plot position
        if i <= 1:
          ax0 = fig.add_subplot(gs[i,j])
        else:
          ax0 = fig.add_subplot(gs[i,:])

        #plot
        color_map = ["#3A7089", "#4b4b4c",'#99AEBB','#5C8374','#6F7597','#7A9D54','#9EB38
        sns.boxplot(data = df, x = k, y = 'Purchase' ,ax = ax0,width = 0.5, palette =colo
        #plot title
        ax0.set_title(f'Purchase Amount Vs {k}',{'font':'serif', 'size':12,'weight':'bold

        #customizing axis
        ax0.set_xticklabels(df[k].unique(),fontweight = 'bold',fontsize = 12)
        ax0.set_ylabel('Purchase Amount',fontweight = 'bold',fontsize = 12)
        ax0.set_xlabel('')

plt.show()
```



Purchase Amount Vs Gender

Purchase Amount Vs City_Category

Purchase Amount Vs Marital_Status

Purchase Amount Vs Stay_In_Current_City_Years

Purchase Amount Vs Age

Gender VS Purchase Amount

```
#creating a df for purchase amount vs gender
temp = df.groupby('Gender')['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'])
#renaming the gender
temp['Gender'] = temp['Gender'].replace({'F':'Female','M':'Male'})
temp
```

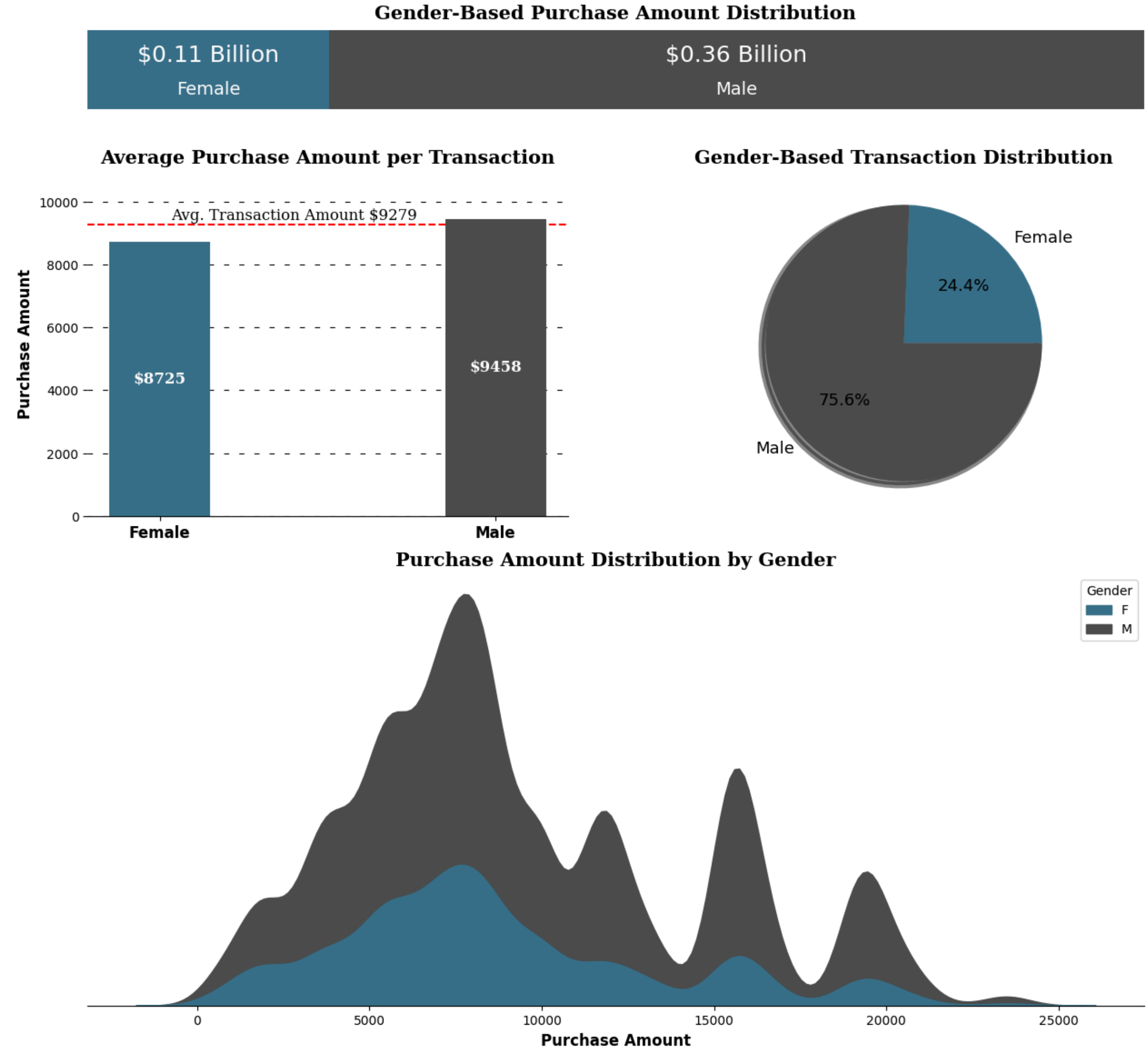
```
   Gender          sum  count  sum_in_billions   %sum  per_purchase
0  Female  106555490.0  12212             0.11  0.229        8725.0
1    Male  358152116.0  37867             0.36  0.771        9458.0
```

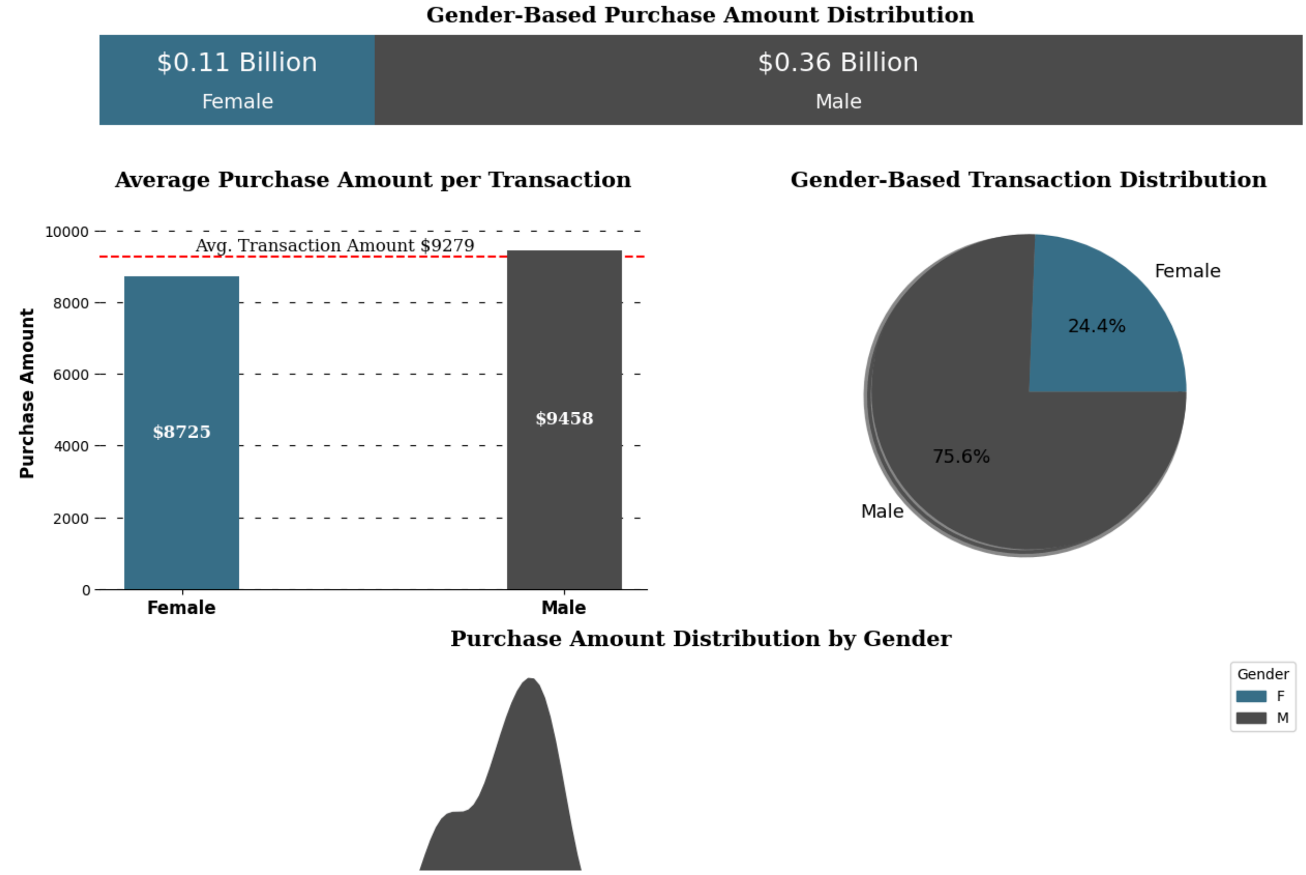
Next steps: View recommended plots

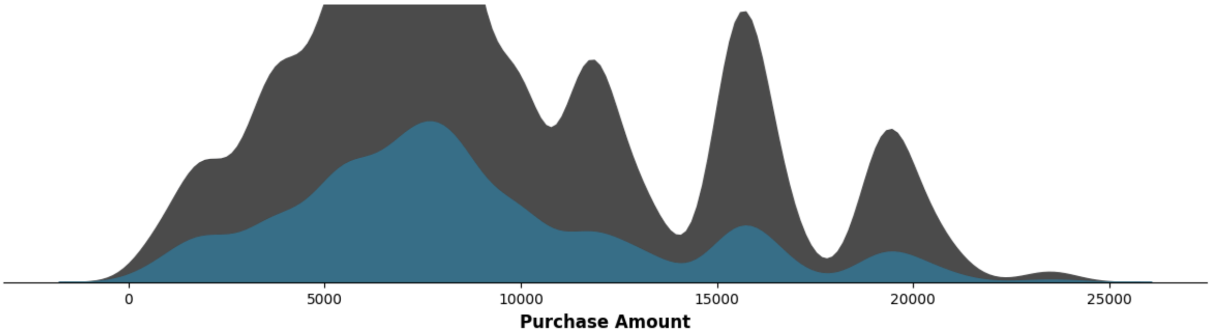
```
#setting the plot style
fig = plt.figure(figsize = (15,14))
gs = fig.add_gridspec(3,2,height_ratios =[0.10,0.4,0.5])
 #Distribution of Purchase Amount
ax = fig.add_subplot(gs[0,:])
#plotting the visual
ax.barh(temp.loc[0,'Gender'],width = temp.loc[0,'%sum'],color = "#3A7089",label = 'Female
ax.barh(temp.loc[0,'Gender'],width = temp.loc[1,'%sum'],left =temp.loc[0,'%sum'], color =
#inserting the text
txt = [0.0] #for left parameter in ax.text()
for i in temp.index:
 #for amount
 ax.text(temp.loc[i,'%sum']/2 + txt[0],0.15,f"${temp.loc[i,'sum_in_billions']} Billion",
 va = 'center', ha='center',fontsize=18, color='white')

 #for gender
 ax.text(temp.loc[i,'%sum']/2 + txt[0],- 0.20 ,f"{temp.loc[i,'Gender']}",
 va = 'center', ha='center',fontsize=14, color='white')

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

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

#customizing ticks
ax.set_xticks([])
ax.set_yticks([])
ax.set_xlim(0,1)
#plot title
ax.set_title('Gender-Based Purchase Amount Distribution',{'font':'serif', 'size':15,'weig
 #Distribution of Purchase Amount per Transaction

ax1 = fig.add_subplot(gs[1,0])
color_map = ["#3A7089", "#4b4b4c"]
#plotting the visual
ax1.bar(temp['Gender'],temp['per_purchase'],color = color_map,zorder = 2,width = 0.3)
#adding average transaction line
avg = round(df['Purchase'].mean())
ax1.axhline(y = avg, color ='red', zorder = 0,linestyle = '--')
#adding text for the line
ax1.text(0.4,avg + 300, f"Avg. Transaction Amount ${avg:.0f}",
 {'font':'serif','size' : 12},ha = 'center',va = 'center')
#adjusting the ylimits
ax1.set_ylim(0,11000)
#adding the value_counts
for i in temp.index:
 ax1.text(temp.loc[i,'Gender'],temp.loc[i,'per_purchase']/2,f"${temp.loc[i,'per_purchase'
 {'font':'serif','size' : 12,'color':'white','weight':'bold' },ha = 'center',va = 'center

#adding grid lines
ax1.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes = (5,10))
#removing the axis lines
for s in ['top','left','right']:
 ax1.spines[s].set_visible(False)
```

```
#adding axis label
ax1.set_ylabel('Purchase Amount',fontweight = 'bold',fontsize = 12)
ax1.set_xticklabels(temp['Gender'],fontweight = 'bold',fontsize = 12)
#setting title for visual
ax1.set_title('Average Purchase Amount per Transaction',{'font':'serif', 'size':15,'weigh
 # creating pie chart for gender disribution
ax2 = fig.add_subplot(gs[1,1])
color_map = ["#3A7089", "#4b4b4c"]
ax2.pie(temp['count'],labels = temp['Gender'],autopct = '%.1f%%',
 shadow = True,colors = color_map,wedgeprops = {'linewidth': 5},textprops={'fontsize': 13
#setting title for visual
ax2.set_title('Gender-Based Transaction Distribution',{'font':'serif', 'size':15,'weight'
 # creating kdeplot for purchase amount distribution
ax3 = fig.add_subplot(gs[2,:])
#plotting the kdeplot
sns.kdeplot(data = df, x = 'Purchase', hue = 'Gender', palette = color_map,fill = True, a
#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 = 12)
#setting title for visual
ax3.set_title('Purchase Amount Distribution by Gender',{'font':'serif', 'size':15,'weight
plt.show()#setting the plot style
fig = plt.figure(figsize = (15,14))
gs = fig.add_gridspec(3,2,height_ratios =[0.10,0.4,0.5])
 #Distribution of Purchase Amount
ax = fig.add_subplot(gs[0,:])
#plotting the visual
ax.barh(temp.loc[0,'Gender'],width = temp.loc[0,'%sum'],color = "#3A7089",label = 'Female
ax.barh(temp.loc[0,'Gender'],width = temp.loc[1,'%sum'],left =temp.loc[0,'%sum'], color =
#inserting the text
txt = [0.0] #for left parameter in ax.text()
for i in temp.index:
 #for amount
 ax.text(temp.loc[i,'%sum']/2 + txt[0],0.15,f"${temp.loc[i,'sum_in_billions']} Billion",
 va = 'center', ha='center',fontsize=18, color='white')

 #for gender
 ax.text(temp.loc[i,'%sum']/2 + txt[0],- 0.20 ,f"{temp.loc[i,'Gender']}",
 va = 'center', ha='center',fontsize=14, color='white')

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

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

#customizing ticks
ax.set_xticks([])
ax.set_yticks([])
```

```
ax.set_xlim(0,1)
#plot title
ax.set_title('Gender-Based Purchase Amount Distribution',{'font':'serif', 'size':15,'weig
 #Distribution of Purchase Amount per Transaction

ax1 = fig.add_subplot(gs[1,0])
color_map = ["#3A7089", "#4b4b4c"]
#plotting the visual
ax1.bar(temp['Gender'],temp['per_purchase'],color = color_map,zorder = 2,width = 0.3)
#adding average transaction line
avg = round(df['Purchase'].mean())
ax1.axhline(y = avg, color ='red', zorder = 0,linestyle = '--')
#adding text for the line
ax1.text(0.4,avg + 300, f"Avg. Transaction Amount ${avg:.0f}",
 {'font':'serif','size' : 12},ha = 'center',va = 'center')
#adjusting the ylimits
ax1.set_ylim(0,11000)
#adding the value_counts
for i in temp.index:
 ax1.text(temp.loc[i,'Gender'],temp.loc[i,'per_purchase']/2,f"${temp.loc[i,'per_purchase'
 {'font':'serif','size' : 12,'color':'white','weight':'bold' },ha = 'center',va = 'center

#adding grid lines
ax1.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes = (5,10))
#removing the axis lines
for s in ['top','left','right']:
 ax1.spines[s].set_visible(False)

#adding axis label
ax1.set_ylabel('Purchase Amount',fontweight = 'bold',fontsize = 12)
ax1.set_xticklabels(temp['Gender'],fontweight = 'bold',fontsize = 12)
#setting title for visual
ax1.set_title('Average Purchase Amount per Transaction',{'font':'serif', 'size':15,'weigh
 # creating pie chart for gender disribution
ax2 = fig.add_subplot(gs[1,1])
color_map = ["#3A7089", "#4b4b4c"]
ax2.pie(temp['count'],labels = temp['Gender'],autopct = '%.1f%%',
 shadow = True,colors = color_map,wedgeprops = {'linewidth': 5},textprops={'fontsize': 13
#setting title for visual
ax2.set_title('Gender-Based Transaction Distribution',{'font':'serif', 'size':15,'weight'
 # creating kdeplot for purchase amount distribution
ax3 = fig.add_subplot(gs[2,:])
#plotting the kdeplot
sns.kdeplot(data = df, x = 'Purchase', hue = 'Gender', palette = color_map,fill = True, a
#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 = 12)
#setting title for visual
ax3.set_title('Purchase Amount Distribution by Gender',{'font':'serif', 'size':15,'weight
plt.show()
```


Gender-Based Purchase Amount Distribution
$0.11 Billion
Female
$0.36 Billion
Male
Average Purchase Amount per Transaction
Purchase Amount
10000
8000
6000
4000
2000
0
Avg. Transaction Amount $9279
$8725
$9458
Female
Male
Gender-Based Transaction Distribution
Female
24.4%
75.6%
Male
Purchase Amount Distribution by Gender
Gender
F
M
0
5000
10000
15000
20000
25000
Purchase Amount


Gender-Based Purchase Amount Distribution
$0.11 Billion
Female
$0.36 Billion
Male
Average Purchase Amount per Transaction
Purchase Amount
10000
8000
6000
4000
2000
0
Avg. Transaction Amount $9279
$8725
$9458
Female
Male
Gender-Based Transaction Distribution
Female
24.4%
75.6%
Male
Purchase Amount Distribution by Gender
Gender
F
M

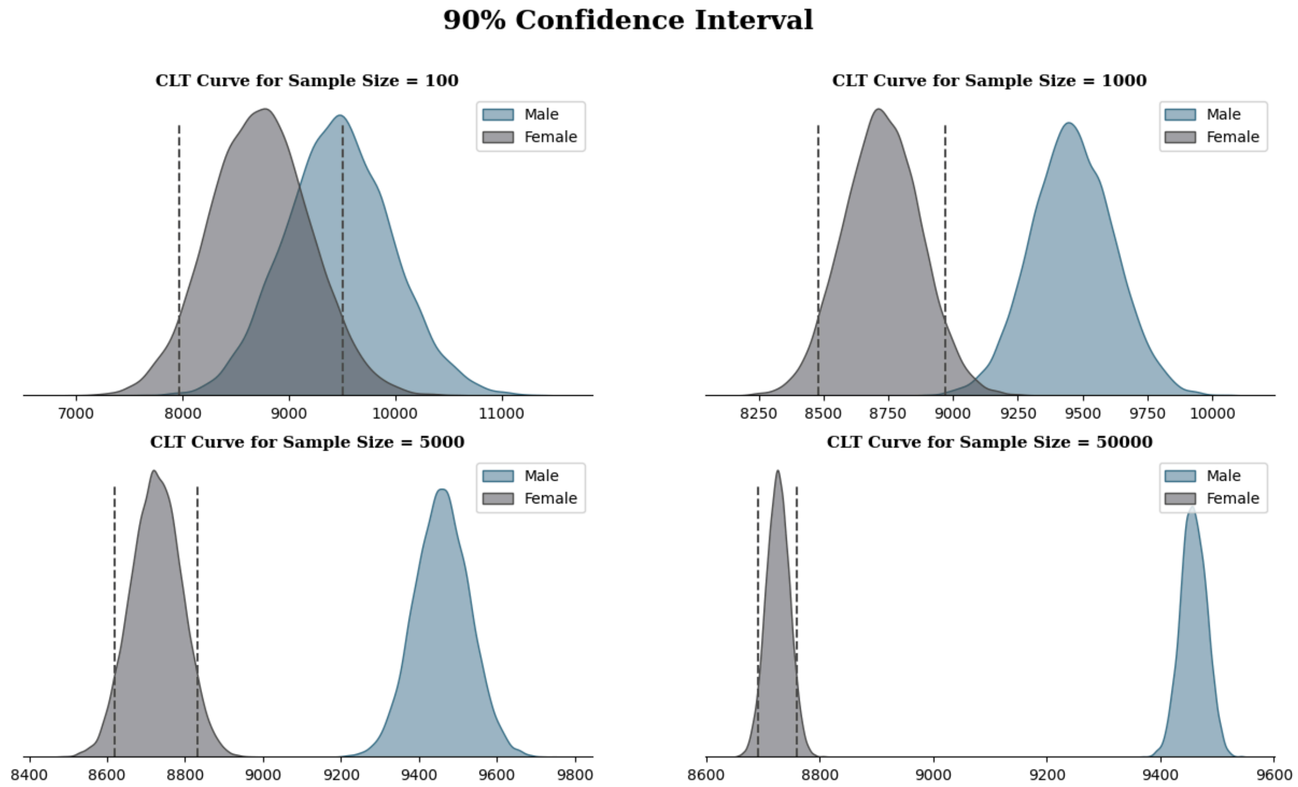

0
5000
10000
15000
20000
25000
Purchase Amount

Confidence Interval Construction: Estimating Average Purchase Amount per Transaction

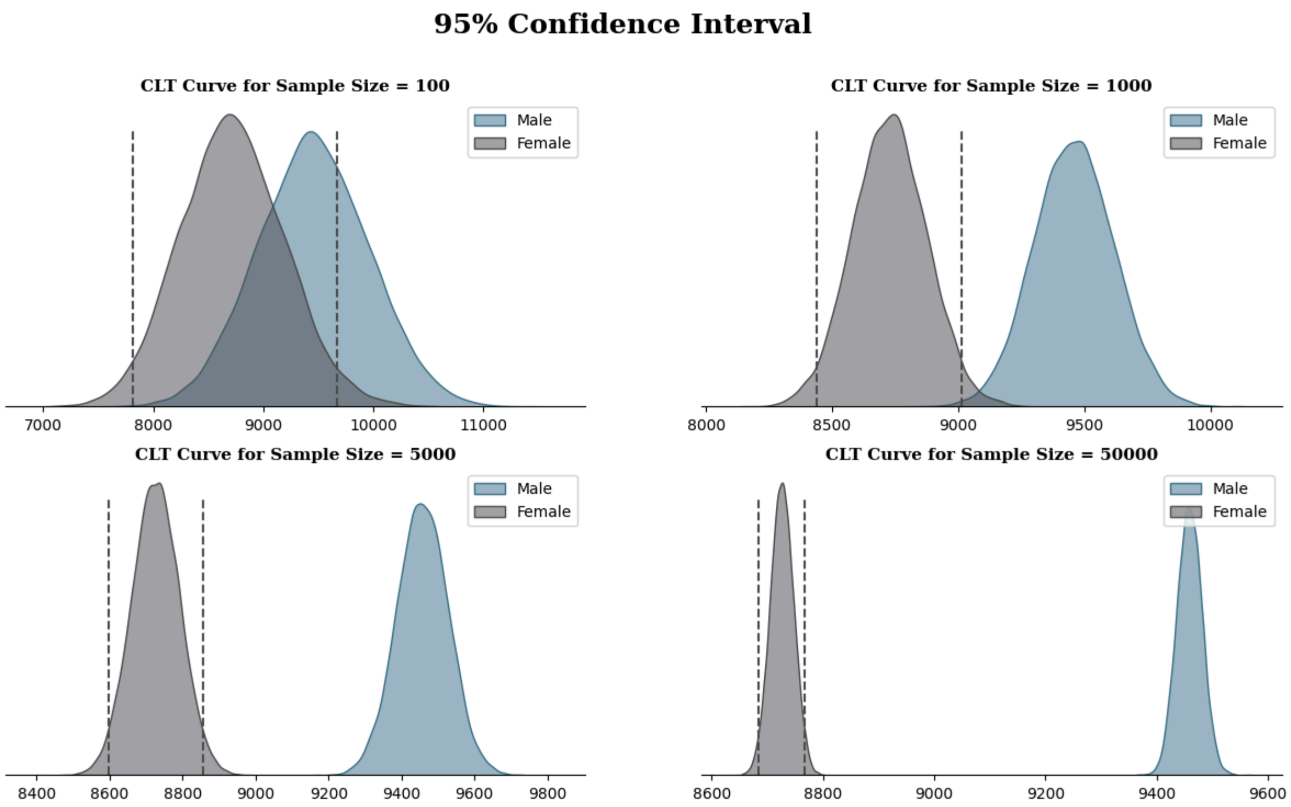
```
#creating a function to calculate confidence interval
def confidence_interval(data,ci):
 #converting the list to series
 l_ci = (100-ci)/2
 u_ci = (100+ci)/2

 #calculating lower limit and upper limit of confidence interval
 interval = np.percentile(data,[l_ci,u_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
    bootstrap_samples = 20000
    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(bootstrap_samples):
          #creating random 5000 samples of i sample size
          male_bootstrapped_samples = np.random.choice(df_male,size = i)
          female_bootstrapped_samples = np.random.choice(df_female,size = i)
          #calculating mean of those samples
          male_sample_mean = np.mean(male_bootstrapped_samples)
          female_sample_mean = np.mean(female_bootstrapped_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_mean
        #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 ="#3A7089" ,fill = True, alpha
        sns.kdeplot(data = temp_df,x = 'female_means',color ="#4b4b4c" ,fill = True, alph
        #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 ="#3A7089",linestyle = '--')
        for k in f_range:
          ax.axvline(x = k,ymax = 0.9, color ="#4b4b4c",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,'weigh
        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
```

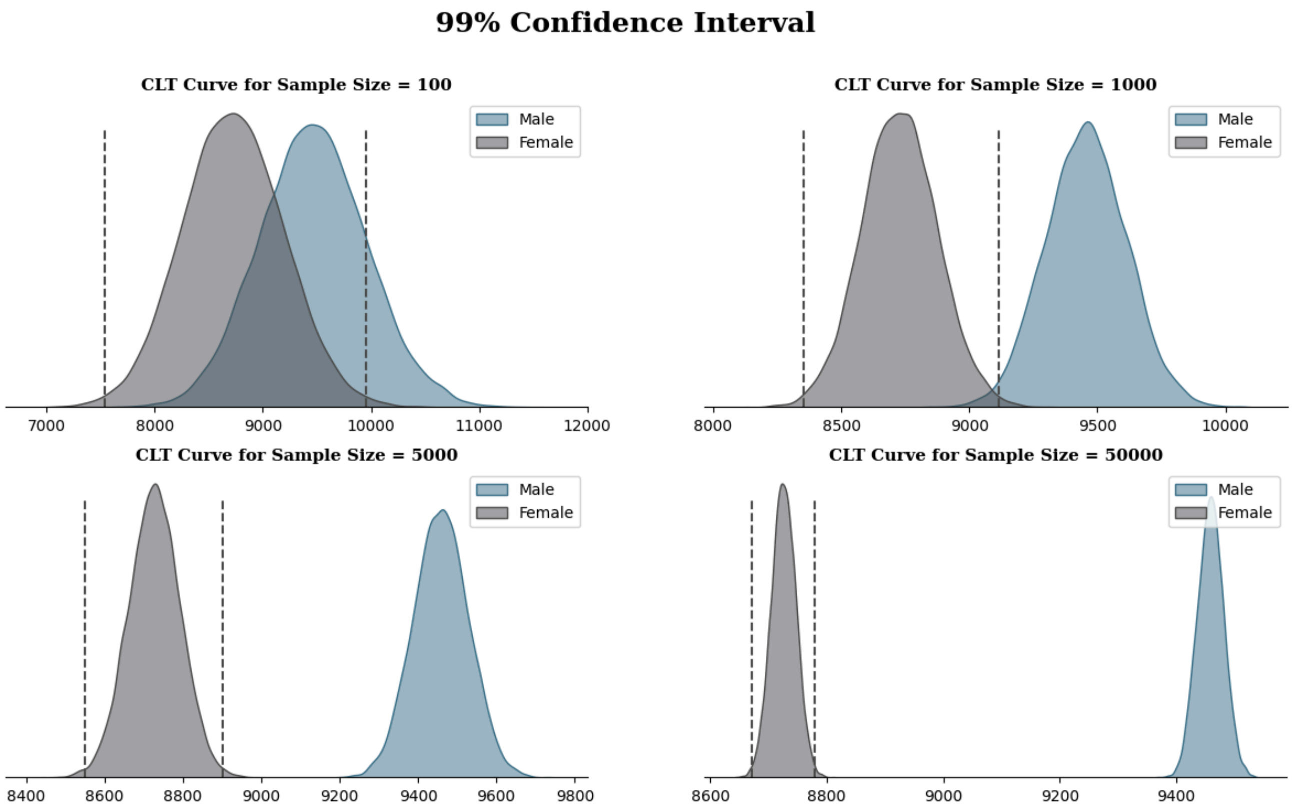
```
m_samp_90,f_samp_90 = plot(90)
```



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

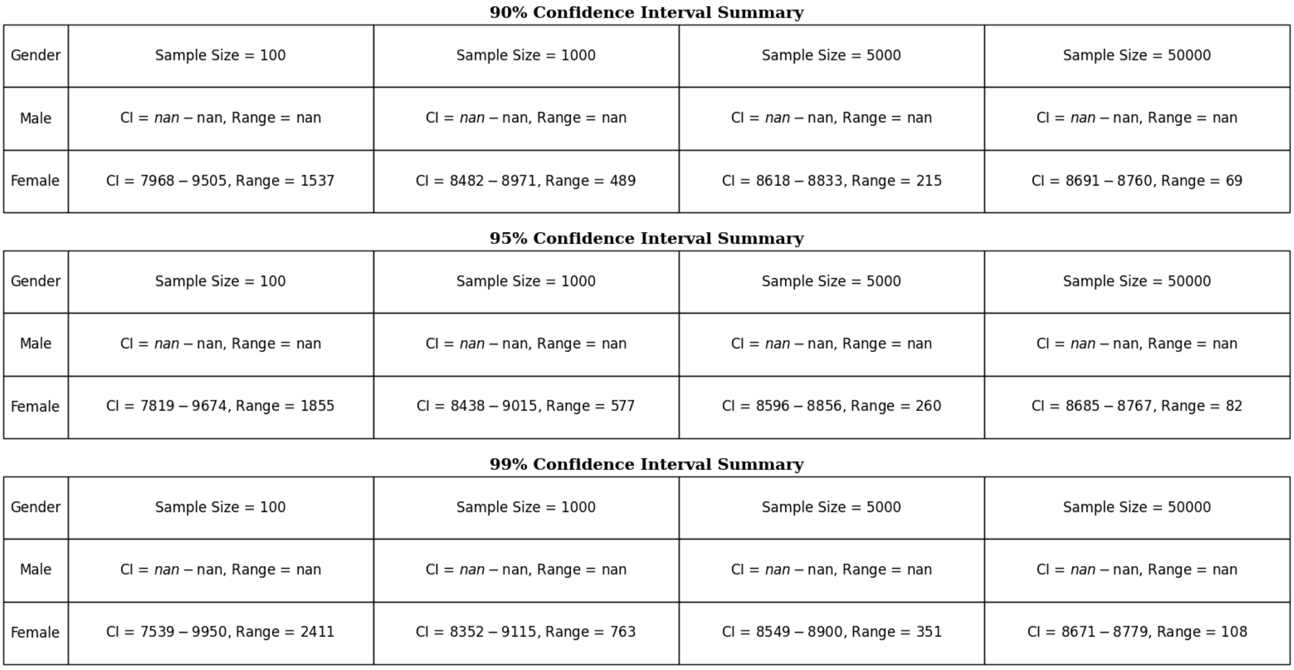


```
m_samp_99,f_samp_99 = plot(99)
```



Are confidence intervals of average male and female spending overlapping?

```
fig = plt.figure(figsize = (20,10))
gs = fig.add_gridspec(3,1)
for i,j,k,l in [(m_samp_90,f_samp_90,90,0),(m_samp_95,f_samp_95,95,1),(m_samp_99,f_samp_9
      #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]

      #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]

      #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',
      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
```



90% Confidence Interval Summary

Gender	Sample Size = 100	Sample Size = 1000	Sample Size = 5000	Sample Size = 50000
Male	CI = *nan* − nan, Range = nan	CI = *nan* − nan, Range = nan	CI = *nan* − nan, Range = nan	CI = *nan* − nan, Range = nan
Female	CI = 7968 − 9505, Range = 1537	CI = 8482 − 8971, Range = 489	CI = 8618 − 8833, Range = 215	CI = 8691 − 8760, Range = 69

95% Confidence Interval Summary

Gender	Sample Size = 100	Sample Size = 1000	Sample Size = 5000	Sample Size = 50000
Male	CI = *nan* − nan, Range = nan	CI = *nan* − nan, Range = nan	CI = *nan* − nan, Range = nan	CI = *nan* − nan, Range = nan
Female	CI = 7819 − 9674, Range = 1855	CI = 8438 − 9015, Range = 577	CI = 8596 − 8856, Range = 260	CI = 8685 − 8767, Range = 82

99% Confidence Interval Summary

Gender	Sample Size = 100	Sample Size = 1000	Sample Size = 5000	Sample Size = 50000
Male	CI = *nan* − nan, Range = nan	CI = *nan* − nan, Range = nan	CI = *nan* − nan, Range = nan	CI = *nan* − nan, Range = nan
Female	CI = 7539 − 9950, Range = 2411	CI = 8352 − 9115, Range = 763	CI = 8549 − 8900, Range = 351	CI = 8671 − 8779, Range = 108

Marital Status VS Purchase Amount

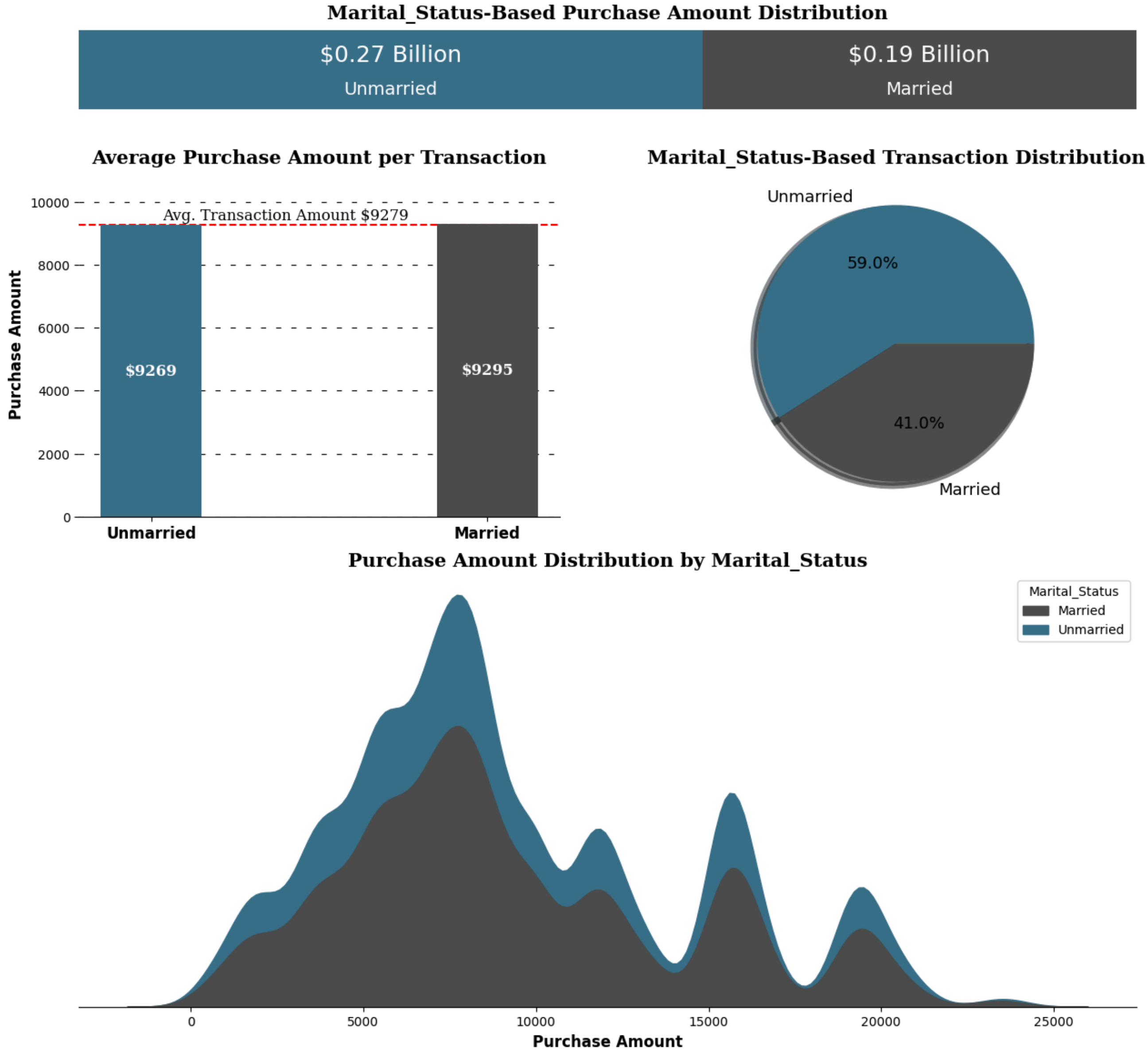
```
#creating a df for purchase amount vs marital status
temp = df.groupby('Marital_Status')['Purchase'].agg(['sum','count']).reset_index()
#calculating the amount in billions
temp['sum_in_billions'] = round(temp['sum'] / 10**9,2)
#calculationg percentage distribution of purchase amount
temp['%sum'] = 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	274027182.0	29564	0.27	0.59	9269.0
1	Married	190680424.0	20515	0.19	0.41	9295.0

Next steps: View recommended plots

```
#setting the plot style
fig = plt.figure(figsize = (15,14))
gs = fig.add_gridspec(3,2,height_ratios =[0.10,0.4,0.5])
 #Distribution of Purchase Amount
ax = fig.add_subplot(gs[0,:])
#plotting the visual
ax.barh(temp.loc[0,'Marital_Status'],width = temp.loc[0,'%sum'],color = "#3A7089",label =
ax.barh(temp.loc[0,'Marital_Status'],width = temp.loc[1,'%sum'],left =temp.loc[0,'%sum'],
#inserting the text
txt = [0.0] #for left parameter in ax.text()
for i in temp.index:
 #for amount
 ax.text(temp.loc[i,'%sum']/2 + txt[0],0.15,f"${temp.loc[i,'sum_in_billions']} Billion",
 va = 'center', ha='center',fontsize=18, color='white')

 #for marital status
 ax.text(temp.loc[i,'%sum']/2 + txt[0],- 0.20 ,f"{temp.loc[i,'Marital_Status']}",
 va = 'center', ha='center',fontsize=14, color='white')

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

#removing the axis lines
for s in ['top','left','right','bottom']:
 ax.spines[s].set_visible(False)
#customizing ticks
ax.set_xticks([])
ax.set_yticks([])
ax.set_xlim(0,1)
#plot title
ax.set_title('Marital_Status-Based Purchase Amount Distribution',{'font':'serif', 'size':
 #Distribution of Purchase Amount per Transaction

ax1 = fig.add_subplot(gs[1,0])
color_map = ["#3A7089", "#4b4b4c"]
#plotting the visual
ax1.bar(temp['Marital_Status'],temp['per_purchase'],color = color_map,zorder = 2,width =
#adding average transaction line
avg = round(df['Purchase'].mean())
ax1.axhline(y = avg, color ='red', zorder = 0,linestyle = '--')
#adding text for the line
ax1.text(0.4,avg + 300, f"Avg. Transaction Amount ${avg:.0f}",
 {'font':'serif','size' : 12},ha = 'center',va = 'center')
#adjusting the ylimits
ax1.set_ylim(0,11000)
#adding the value_counts
for i in temp.index:
 ax1.text(temp.loc[i,'Marital_Status'],temp.loc[i,'per_purchase']/2,f"${temp.loc[i,'per_p
 {'font':'serif','size' : 12,'color':'white','weight':'bold' },ha = 'center',va = 'center

#adding grid lines
ax1.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes = (5,10))
#removing the axis lines
for s in ['top','left','right']:
 ax1.spines[s].set_visible(False)
```

```
#adding axis label
ax1.set_ylabel('Purchase Amount',fontweight = 'bold',fontsize = 12)
ax1.set_xticklabels(temp['Marital_Status'],fontweight = 'bold',fontsize = 12)
#setting title for visual
ax1.set_title('Average Purchase Amount per Transaction',{'font':'serif', 'size':15,'weigh
 # creating pie chart for Marital_Status disribution
ax2 = fig.add_subplot(gs[1,1])
color_map = ["#3A7089", "#4b4b4c"]
ax2.pie(temp['count'],labels = temp['Marital_Status'],autopct = '%.1f%%',
 shadow = True,colors = color_map,wedgeprops = {'linewidth': 5},textprops={'fontsize': 13
#setting title for visual
ax2.set_title('Marital_Status-Based Transaction Distribution',{'font':'serif', 'size':15,
 # creating kdeplot for purchase amount distribution
ax3 = fig.add_subplot(gs[2,:])
color_map = [ "#4b4b4c","#3A7089"]
#plotting the kdeplot
sns.kdeplot(data = df, x = 'Purchase', hue = 'Marital_Status', palette = color_map,fill =
 ax = ax3,hue_order = ['Married','Unmarried'])
#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 = 12)
#setting title for visual
ax3.set_title('Purchase Amount Distribution by Marital_Status',{'font':'serif', 'size':15
plt.show()
```

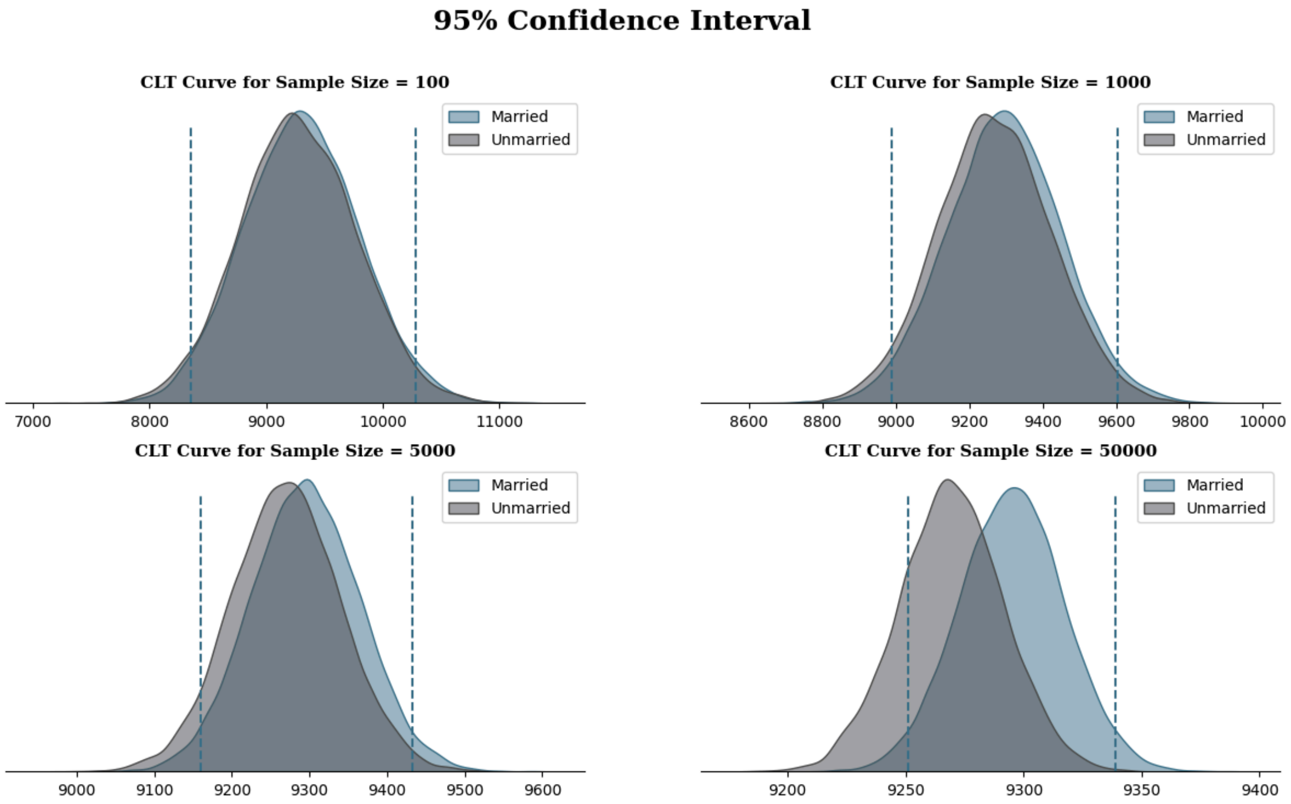

Marital_Status-Based Purchase Amount Distribution
$0.27 Billion
Unmarried
$0.19 Billion
Married
Average Purchase Amount per Transaction
10000
8000
6000
4000
2000
0
Purchase Amount
Avg. Transaction Amount $9279
$9269
$9295
Unmarried
Married
Marital_Status-Based Transaction Distribution
Unmarried
59.0%
41.0%
Married
Purchase Amount Distribution by Marital_Status
Marital_Status
Married
Unmarried
0
5000
10000
15000
20000
25000
Purchase Amount

Confidence Interval Construction: Estimating Average Purchase Amount per Transaction

```
#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
      df_married = df.loc[df['Marital_Status'] == 'Married','Purchase']
      df_unmarried = df.loc[df['Marital_Status'] == 'Unmarried','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
      bootstrap_samples = 20000
      married_samples = {}
      unmarried_samples = {}

      for i,x,y in sample_sizes:
              married_means = [] #list for collecting the means of married sample
              unmarried_means = [] #list for collecting the means of unmarried sample
              for j in range(bootstrap_samples):
                  #creating random 5000 samples of i sample size
                  married_bootstrapped_samples = np.random.choice(df_married,size = i)
                  unmarried_bootstrapped_samples = np.random.choice(df_unmarried,size = i
                  #calculating mean of those samples
                  married_sample_mean = np.mean(married_bootstrapped_samples)
                  unmarried_sample_mean = np.mean(unmarried_bootstrapped_samples)
                  #appending the mean to the list
                  married_means.append(married_sample_mean)
                  unmarried_means.append(unmarried_sample_mean)

              #storing the above sample generated
              married_samples[f'{ci}%_{i}'] = married_means
              unmarried_samples[f'{ci}%_{i}'] = unmarried_means
              #creating a temporary dataframe for creating kdeplot
              temp_df = pd.DataFrame(data = {'married_means':married_means,'unmarried_mea
              #plotting kdeplots
              #plot position
              ax = fig.add_subplot(gs[x,y])

              #plots for married and unmarried
              sns.kdeplot(data = temp_df,x = 'married_means',color ="#3A7089" ,fill = Tru
              sns.kdeplot(data = temp_df,x = 'unmarried_means',color ="#4b4b4c" ,fill = T
              #calculating confidence intervals for given confidence level(ci)
              m_range = confidence_interval(married_means,ci)
              u_range = confidence_interval(unmarried_means,ci)
              #plotting confidence interval on the distribution
              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 = '--')
              #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('')
```

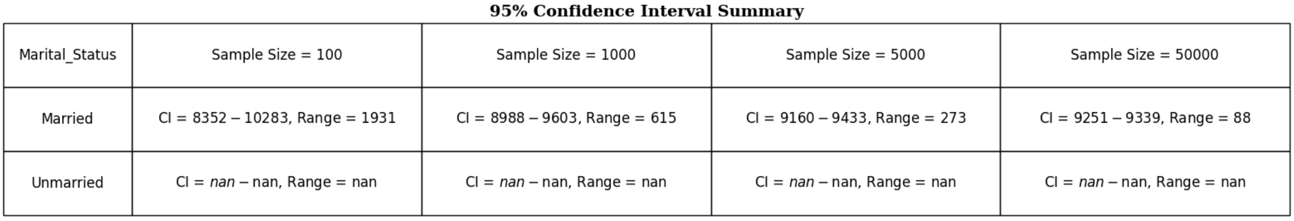
```
            ax.set_xlabel('')
            #setting title for visual
            ax.set_title(f'CLT Curve for Sample Size = {i}',{'font':'serif', 'size':11,
            plt.legend()

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

    return married_samples,unmarried_samples
```

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



Are confidence intervals of average married and unmarried customer spending overlapping?

```
#setting the plot style
fig,ax = plt.subplots(figsize = (20,3))
#list for collecting ci for given cl
m_ci = ['Married']
u_ci = ['Unmarried']
#finding ci for each sample size (married)
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
#finding ci for each sample size (unmarried)
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
 #plotting the summary
#contents of the table
ci_info = [m_ci,u_ci]
#plotting the table
table = ax.table(cellText = ci_info, cellLoc='center',
 colLabels =['Marital_Status','Sample Size = 100','Sample Size = 1000','Sample Size = 500
 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"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 = 8352 – 10283, Range = 1931	CI = 8988 – 9603, Range = 615	CI = 9160 – 9433, Range = 273	CI = 9251 – 9339, Range = 88
Unmarried	CI = *nan* – nan, Range = nan	CI = *nan* – nan, Range = nan	CI = *nan* – nan, Range = nan	CI = *nan* – nan, Range = nan

Customer Age VS Purchase Amount

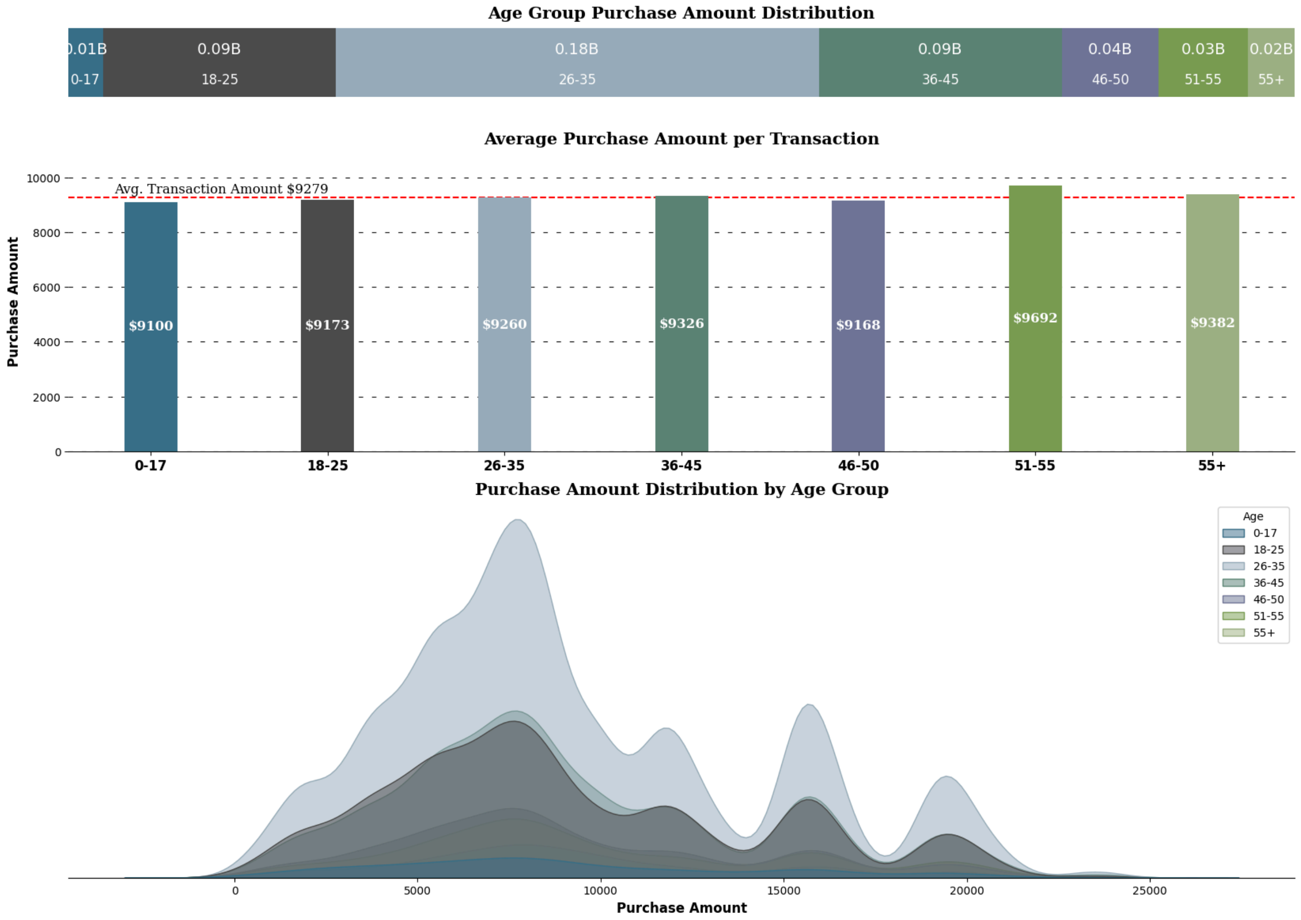
```
#creating a df for purchase amount vs age group
temp = df.groupby('Age')['Purchase'].agg(['sum','count']).reset_index()
#calculating the amount in billions
temp['sum_in_billions'] = round(temp['sum'] / 10**9,2)
#calculationg percentage distribution of purchase amount
temp['%sum'] = round(temp['sum']/temp['sum'].sum(),3)
#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	12821475.0	1409	0.01	0.028	9100.0
1	18-25	88358225.0	9632	0.09	0.190	9173.0
2	26-35	182936918.0	19755	0.18	0.394	9260.0
3	36-45	92004232.0	9865	0.09	0.198	9326.0
4	46-50	36781312.0	4012	0.04	0.079	9168.0
5	51-55	33961570.0	3504	0.03	0.073	9692.0
6	55+	17843874.0	1902	0.02	0.038	9382.0

Next steps: View recommended plots

```
#setting the plot style
fig = plt.figure(figsize = (20,14))
gs = fig.add_gridspec(3,1,height_ratios =[0.10,0.4,0.5])
 #Distribution of Purchase Amount
ax = fig.add_subplot(gs[0])
color_map = ["#3A7089", "#4b4b4c",'#99AEBB','#5C8374','#6F7597','#7A9D54','#9EB384']
#plotting the visual
left = 0
for i in temp.index:
 ax.barh(temp.loc[0,'Age'],width = temp.loc[i,'%sum'],left = left,color = color_map[i],la
 left += temp.loc[i,'%sum']
#inserting the text
txt = 0.0 #for left parameter in ax.text()
for i in temp.index:
 #for amount
 ax.text(temp.loc[i,'%sum']/2 + txt,0.15,f"{temp.loc[i,'sum_in_billions']}B",
 va = 'center', ha='center',fontsize=14, color='white')

 #for age grp
 ax.text(temp.loc[i,'%sum']/2 + txt,- 0.20 ,f"{temp.loc[i,'Age']}",
 va = 'center', ha='center',fontsize=12, color='white')

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

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

#customizing ticks
ax.set_xticks([])
ax.set_yticks([])
ax.set_xlim(0,1)
#plot title
ax.set_title('Age Group Purchase Amount Distribution',{'font':'serif', 'size':15,'weight'
 #Distribution of Purchase Amount per Transaction

ax1 = fig.add_subplot(gs[1])
#plotting the visual
ax1.bar(temp['Age'],temp['per_purchase'],color = color_map,zorder = 2,width = 0.3)
#adding average transaction line
avg = round(df['Purchase'].mean())
ax1.axhline(y = avg, color ='red', zorder = 0,linestyle = '--')
#adding text for the line
ax1.text(0.4,avg + 300, f"Avg. Transaction Amount ${avg:.0f}",
 {'font':'serif','size' : 12},ha = 'center',va = 'center')
#adjusting the ylimits
ax1.set_ylim(0,11000)
#adding the value_counts
for i in temp.index:
 ax1.text(temp.loc[i,'Age'],temp.loc[i,'per_purchase']/2,f"${temp.loc[i,'per_purchase']:.
 {'font':'serif','size' : 12,'color':'white','weight':'bold' },ha = 'center',va = 'center

#adding grid lines
ax1.grid(color = 'black',linestyle = '--', axis = 'y', zorder = 0, dashes = (5,10))
#removing the axis lines
```

```
for s in ['top','left','right']:
 ax1.spines[s].set_visible(False)

#adding axis label
ax1.set_ylabel('Purchase Amount',fontweight = 'bold',fontsize = 12)
ax1.set_xticklabels(temp['Age'],fontweight = 'bold',fontsize = 12)
#setting title for visual
ax1.set_title('Average Purchase Amount per Transaction',{'font':'serif', 'size':15,'weigh
 # creating kdeplot for purchase amount distribution
ax3 = fig.add_subplot(gs[2,:])
#plotting the kdeplot
sns.kdeplot(data = df, x = 'Purchase', hue = 'Age', palette = color_map,fill = True, alph
 ax = ax3)
#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 = 12)
#setting title for visual

ax3.set_title('Purchase Amount Distribution by Age Group',{'font':'serif', 'size':15,'wei
plt.show()
```



Age Group Purchase Amount Distribution

Average Purchase Amount per Transaction

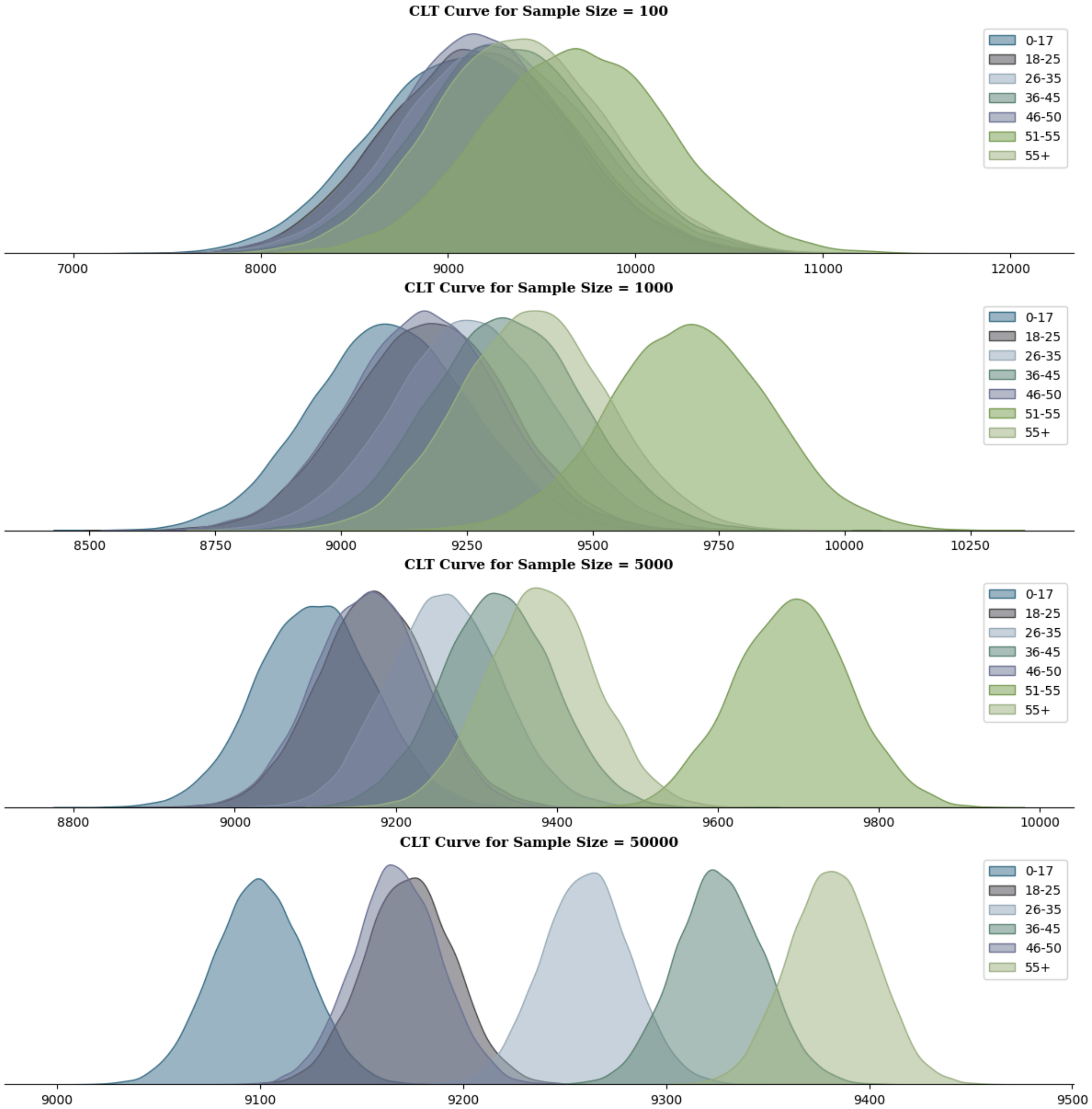
Purchase Amount Distribution by Age Group

Confidence Interval Construction: Estimating Average Purchase Amount per Transaction

```
#defining a function for plotting the visual for given confidence interval
def plot(ci):
          #setting the plot style
          fig = plt.figure(figsize = (15,15))
          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 = 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):
                        #creating random 5000 samples of i sample size
                        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)
                        #calculating mean of those samples
                        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)

                        #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 = pd.DataFrame(data = {'0-17':l1,'18-25':l2,'26-35':l3,'36-45':
                  #plotting kdeplots
                  #plot position
                  ax = fig.add_subplot(gs[x])

                  #plots
                  for p,q in [('#3A7089', '0-17'),('#4b4b4c', '18-25'),('#99AEBB', '26-35
                    ('#7A9D54', '51-55'),('#9EB384', '55+')]:

                    sns.kdeplot(data = temp_df,x = q,color =p ,fill = True, alpha = 0.5,a
                  #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'
                  plt.legend()

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

          return samples1,samples2,samples3,samples4,samples5,samples6,samples7
```

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

95% Confidence Interval



CLT Curve for Sample Size = 100

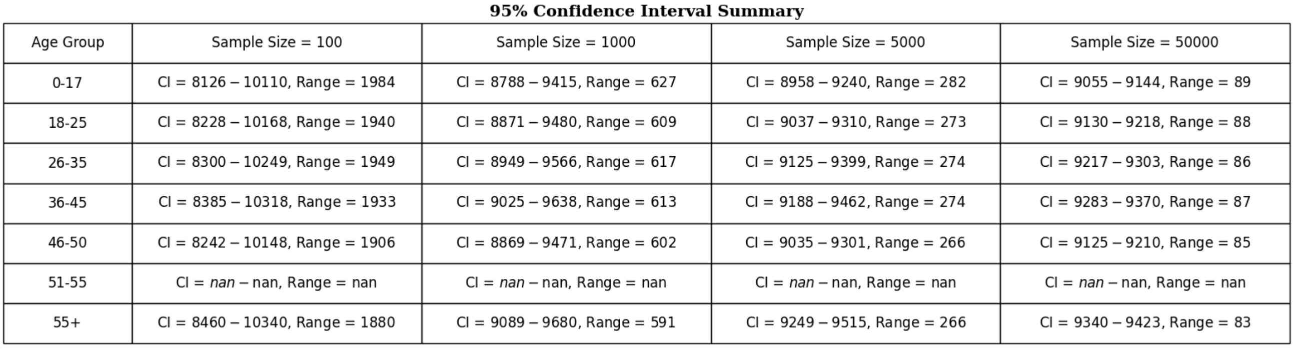
CLT Curve for Sample Size = 1000

CLT Curve for Sample Size = 5000

CLT Curve for Sample Size = 50000

Are confidence intervals of customer's age-group spending overlapping?

```
#setting the plot style
fig,ax = plt.subplots(figsize = (20,5))
#list for collecting ci for given cl
ci_1,ci_2,ci_3,ci_4,ci_5,ci_6,ci_7 = ['0-17'],['18-25'],['26-35'],['36-45'],['46-50'],['5
#finding ci for each sample size
#samples = [samples1,samples2,samples3,samples4,samples5,samples6,samples7]
samples = [(samples1,ci_1),(samples2,ci_2),(samples3,ci_3),(samples4,ci_4),(samples5,ci_5
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_ran
 #plotting the summary
#contents of the table
ci_info = [ci_1,ci_2,ci_3,ci_4,ci_5,ci_6,ci_7]
#plotting the table
table = ax.table(cellText = ci_info, cellLoc='center',
 colLabels =['Age Group','Sample Size = 100','Sample Size = 1000','Sample Size = 5000','S
 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"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 = 8126 – 10110, Range = 1984	CI = 8788 – 9415, Range = 627	CI = 8958 – 9240, Range = 282	CI = 9055 – 9144, Range = 89
18-25	CI = 8228 – 10168, Range = 1940	CI = 8871 – 9480, Range = 609	CI = 9037 – 9310, Range = 273	CI = 9130 – 9218, Range = 88
26-35	CI = 8300 – 10249, Range = 1949	CI = 8949 – 9566, Range = 617	CI = 9125 – 9399, Range = 274	CI = 9217 – 9303, Range = 86
36-45	CI = 8385 – 10318, Range = 1933	CI = 9025 – 9638, Range = 613	CI = 9188 – 9462, Range = 274	CI = 9283 – 9370, Range = 87
46-50	CI = 8242 – 10148, Range = 1906	CI = 8869 – 9471, Range = 602	CI = 9035 – 9301, Range = 266	CI = 9125 – 9210, Range = 85
51-55	CI = *nan* – nan, Range = nan	CI = *nan* – nan, Range = nan	CI = *nan* – nan, Range = nan	CI = *nan* – nan, Range = nan
55+	CI = 8460 – 10340, Range = 1880	CI = 9089 – 9680, Range = 591	CI = 9249 – 9515, Range = 266	CI = 9340 – 9423, Range = 83

Recommendations

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.

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit