almart-confidence-interval-and-clt

March 25, 2024

#Business Case: Walmart - Confidence Interval and CLT

#About Walmart

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

#Business Problem

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

##Exploratory Data Analysis

[1]:
```
#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
```

[4]:
```
# loading the dataset
df = pd.read_csv('walmart_data.txt')
```

[5]:
```
df.head()
```

[5]:
```
   User_ID Product_ID Gender   Age  Occupation City_Category  \
0  1000001  P00069042      F  0-17        10.0             A
1  1000001  P00248942      F  0-17        10.0             A
2  1000001  P00087842      F  0-17        10.0             A
3  1000001  P00085442      F  0-17        10.0             A
4  1000002  P00285442      M   55+        16.0             C
```

```
  Stay_In_Current_City_Years  Marital_Status  Product_Category  Purchase
0                          2             0.0               3.0    8370.0
1                          2             0.0               1.0   15200.0
2                          2             0.0              12.0    1422.0
3                          2             0.0              12.0    1057.0
4                         4+             0.0               8.0    7969.0
```

[6]:
```
df.tail()
```

[6]:
```
        User_ID Product_ID Gender    Age  Occupation City_Category  \
225385  1004715  P00187342      M  26-35         2.0             B
225386  1004715  P00181342      M  26-35         2.0             B
225387  1004715  P00157642      M  26-35         2.0             B
225388  1004715  P00014842      M  26-35         2.0             B
225389      100        NaN    NaN    NaN         NaN           NaN

       Stay_In_Current_City_Years  Marital_Status  Product_Category  Purchase
225385                          3             0.0               4.0    2753.0
225386                          3             0.0              11.0    1695.0
225387                          3             0.0               1.0   15346.0
225388                          3             0.0               1.0   11773.0
225389                        NaN             NaN               NaN       NaN
```

[7]:
```
df.shape
```

[7]:
```
(225390, 10)
```

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 225390 entries, 0 to 225389
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     225390 non-null  int64
 1   Product_ID                  225389 non-null  object
 2   Gender                      225389 non-null  object
 3   Age                         225389 non-null  object
 4   Occupation                  225389 non-null  float64
 5   City_Category               225389 non-null  object
 6   Stay_In_Current_City_Years  225389 non-null  object
 7   Marital_Status              225389 non-null  float64
 8   Product_Category            225389 non-null  float64
 9   Purchase                    225389 non-null  float64
dtypes: float64(4), int64(1), object(5)
memory usage: 17.2+ MB
```

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

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

##Changing the Datatype of Columns

[9]:
```python
for i in df.columns[:-1]:
    df[i] = df[i].astype('category')

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 225390 entries, 0 to 225389
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     225390 non-null  category
 1   Product_ID                  225389 non-null  category
 2   Gender                      225389 non-null  category
 3   Age                         225389 non-null  category
 4   Occupation                  225389 non-null  category
 5   City_Category               225389 non-null  category
 6   Stay_In_Current_City_Years  225389 non-null  category
 7   Marital_Status              225389 non-null  category
 8   Product_Category            225389 non-null  category
 9   Purchase                    225389 non-null  float64
dtypes: category(9), float64(1)
memory usage: 4.4 MB
```

##Satistical summary of object type columns

[10]:
```python
df.describe(include = 'category')
```

[10]:
```
        User_ID Product_ID  Gender     Age  Occupation City_Category  \
count    225390     225389  225389  225389    225389.0        225389
unique     5890       3484       2       7        21.0             3
top     1001680  P00265242       M   26-35         4.0             B
freq        464        709  170197   89894     29827.0         95245

       Stay_In_Current_City_Years  Marital_Status  Product_Category
count                      225389        225389.0          225389.0
unique                          5             2.0              18.0
top                             1             0.0               5.0
freq                        79205        133306.0           62451.0
```

1. User_ID - Among 5,50,068 transactions there are 5891 unique user_id, indicating same customers buying multiple products.

2. Product_ID - Among 5,50,068 transactions there are 3631 unique products,with the product having the code P00265242 being the highest seller, with a maximum of 1,880 units sold.

3. Gender - Out of 5,50,068 transactions, 4,14,259 (nearly 75%) were done by male gender indicating a significant disparity in purchase behavior between males and females during the Black Friday event.

4. Age - We have 7 unique age groups in the dataset. 26 - 35 Age group has maximum of 2,19,587 transactions. We will analyse this feature in detail in future

5. Stay_In_Current_City_Years - Customers with 1 year of stay in current city accounted to maximum of 1,93,821 transactions among all the other customers with (0,2,3,4+) years of stay in current city

6. Marital_Status - 59% of the total transactions were done by Unmarried Customers and 41% by Married Customers.

0.0.1 Satistical summary of numerical data type columns

[11]:
```
df.describe()
```

[11]:
```
             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
```

The purchase amounts vary widely, with the minimum recorded purchase being $12 and the maximum reaching $23961. The median purchase amount of $8047 is notably lower than the mean purchase amount of $9264, indicating a right-skewed distribution where a few high-value purchases pull up the mean

###Duplicate Detection

[12]:
```
df.duplicated().value_counts()
```

[12]:
```
False    225390
dtype: int64
```

- There are no duplicate entries in the dataset

0.1 Sanity Check for columns

[13]:
```
# checking the unique values for columns
for i in df.columns:
    print('Unique Values in',i,'column are :-')
    print(df[i].unique())
```

```python
    print('-'*70)
```

```
Unique Values in User_ID column are :-
[1000001, 1000002, 1000003, 1000004, 1000005, …, 1004293, 1004588, 1004871,
1004113, 100]
Length: 5890
Categories (5890, int64): [100, 1000001, 1000002, 1000003, …, 1006037,
1006038, 1006039, 1006040]
----------------------------------------------------------------------
Unique Values in Product_ID column are :-
['P00069042', 'P00248942', 'P00087842', 'P00085442', 'P00285442', …,
'P00301342', 'P00301142', 'P00038042', 'P00341142', NaN]
Length: 3485
Categories (3484, object): ['P00000142', 'P00000242', 'P00000342', 'P00000442',
…, 'P0099642',
                            'P0099742', 'P0099842', 'P0099942']
----------------------------------------------------------------------
Unique Values in Gender column are :-
['F', 'M', NaN]
Categories (2, object): ['F', 'M']
----------------------------------------------------------------------
Unique Values in Age column are :-
['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25', NaN]
Categories (7, object): ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55',
'55+']
----------------------------------------------------------------------
Unique Values in Occupation column are :-
[10.0, 16.0, 15.0, 7.0, 20.0, …, 5.0, 14.0, 13.0, 6.0, NaN]
Length: 22
Categories (21, float64): [0.0, 1.0, 2.0, 3.0, …, 17.0, 18.0, 19.0, 20.0]
----------------------------------------------------------------------
Unique Values in City_Category column are :-
['A', 'C', 'B', NaN]
Categories (3, object): ['A', 'B', 'C']
----------------------------------------------------------------------
Unique Values in Stay_In_Current_City_Years column are :-
['2', '4+', '3', '1', '0', NaN]
Categories (5, object): ['0', '1', '2', '3', '4+']
----------------------------------------------------------------------
Unique Values in Marital_Status column are :-
[0.0, 1.0, NaN]
Categories (2, float64): [0.0, 1.0]
----------------------------------------------------------------------
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. … 23310. 21248.    nan]
-----------------------------------------------------------------------
```

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

[14]:
```
#replacing the values in marital_status column

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

[14]:
```
['Unmarried', 'Married', NaN]
Categories (2, object): ['Unmarried', 'Married']
```

0.2 Missing Value Analysis

[15]:
```
df.isnull().sum()
```

[15]:
```
User_ID                       0
Product_ID                    1
Gender                        1
Age                           1
Occupation                    1
City_Category                 1
Stay_In_Current_City_Years    1
Marital_Status                1
Product_Category              1
Purchase                      1
dtype: int64
```

0.3 - The dataset does not contain any missing values.

1 Univariate Analysis

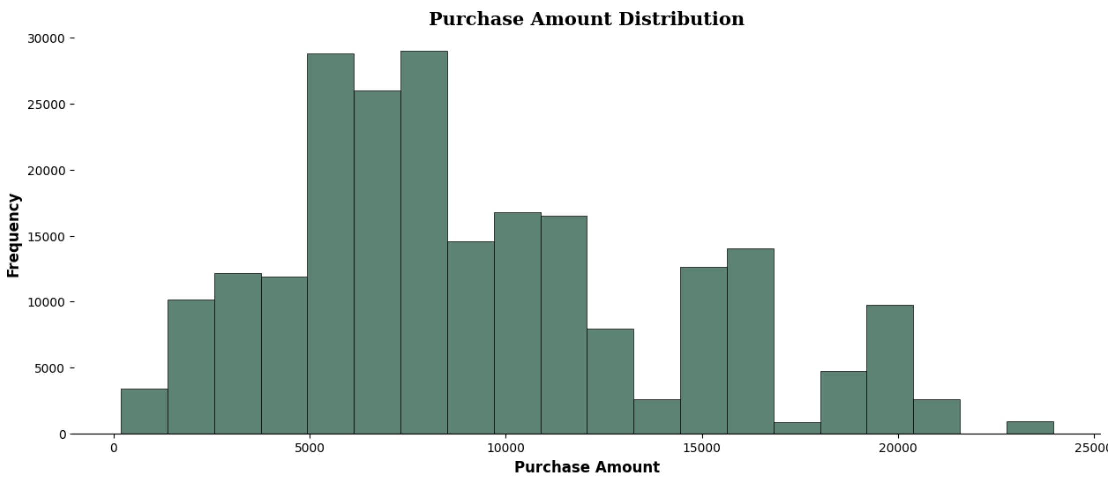
1.1 Numerical Variables

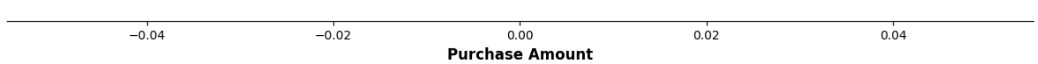
1.1.1 Purchase Amount Distribution

[17]:
```
#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 lowerlimit

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 = 12,
            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

- As seen above, Purchase amount over 21399 is considered as outlier. We will count the number of outliers as below

[18]:
```python
len(df.loc[df['Purchase'] > 21399,'Purchase'])
```

[18]: 1069

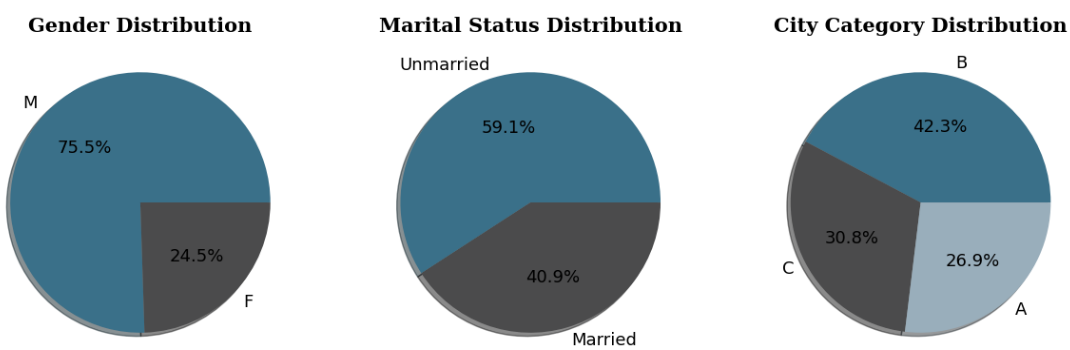
- **Outliers**
  - There are total of `2677 outliers` which is roughly `0.48%` of the total data present in purchase amount. We will not remove them as it indicates a broad range of spending behaviors during the sale, highlighting the importance of tailoring marketing strategies to both regular and high-value customers to maximize revenue.
- **Distribution**
  - Data suggests that the majority of customers spent between `5,823 USD` and `12,054 USD`, with the `median` purchase amount being `8,047 USD`.
  - The lower limit of `12 USD` while the upper limit of `21,399 USD` reveal significant variability in customer spending

Categorical Variables

1.2 Gender, Marital Status and City Category Distribution

[19]:
```python
#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,autopct = '%.1f%%',
        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_counts().index,autopct = '%.1f%%',
        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_counts().index,autopct = '%.1f%%',
        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()
```



1. Gender Distribution - Data indicates a significant disparity in purchase behavior between males and females during the Black Friday event.

2. Marital Status - Given that unmarried customers account for a higher percentage of transactions, it may be worthwhile to consider specific marketing campaigns or promotions that appeal to this group.

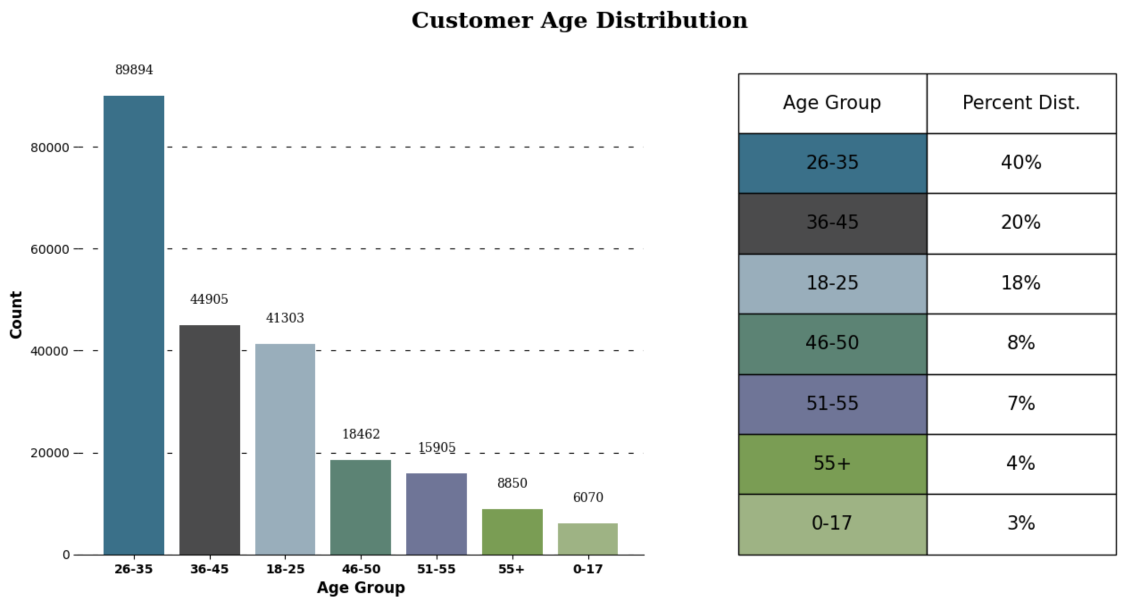
3. City Category - City B saw the most number of transactions followed by City C and City A respectively

1.3 Customer Age Distribution

[20]:
```
#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%'],['51-55','7%'],['55+','4%']
                       ['0-17','3%']]
color_2d =␣
↪[["#3A7089",'#FFFFFF'],["#4b4b4c",'#FFFFFF'],['#99AEBB','#FFFFFF'],['#5C8374','#FFFFFF'],['
            ['#7A9D54','#FFFFFF'],['#9EB384','#FFFFFF']]

table = ax1.table(cellText = age_info, cellColours=color_2d,␣
↪cellLoc='center',colLabels =['Age Group','Percent Dist.'],
                  colLoc = 'center',bbox =[0, 0, 1, 1])

table.set_fontsize(15)

#removing axis
ax1.axis('off')
```

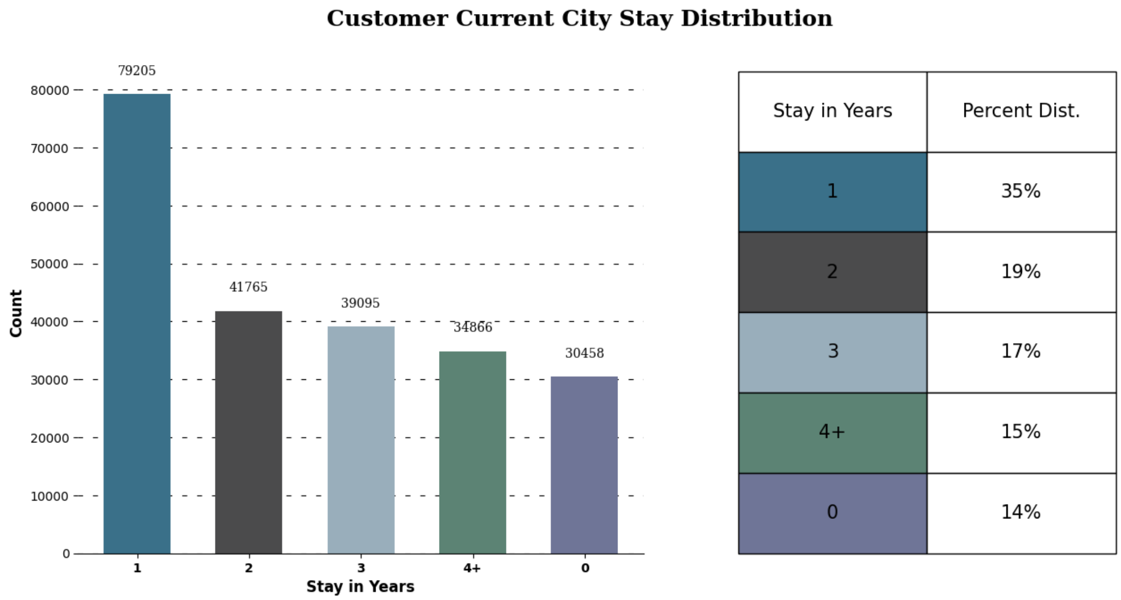
```
#setting title for visual
fig.suptitle('Customer Age Distribution',font = 'serif', size = 18, weight = 'bold')

plt.show()
```



1.4 Customer Stay In current City Distribution

[21]:
```
#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','#FFFFFF'],['

table = ax2.table(cellText = stay_info, cellColours=color_2d,␣
↪cellLoc='center',colLabels =['Stay in Years','Percent Dist.'],
                  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 = 'bold')

plt.show()
```



- The data suggests that the customers are either new to the city or move frequently, and may have different preferences and needs than long-term residents.
- The majority of the customers `(49%)` have stayed in the current city for `one year or less`. This suggests that Walmart has a strong appeal to newcomers who may be looking for affordable and convenient shopping options.
- `4+ years` category `(14%)` customers indicates that Walmart has a `loyal customer base` who have been living in the same city for a long time.
- The percentage of customers `decreases` as the stay in the current city `increases` which suggests that Walmart may benefit from targeting long-term residents for `loyalty programs and promotions`.

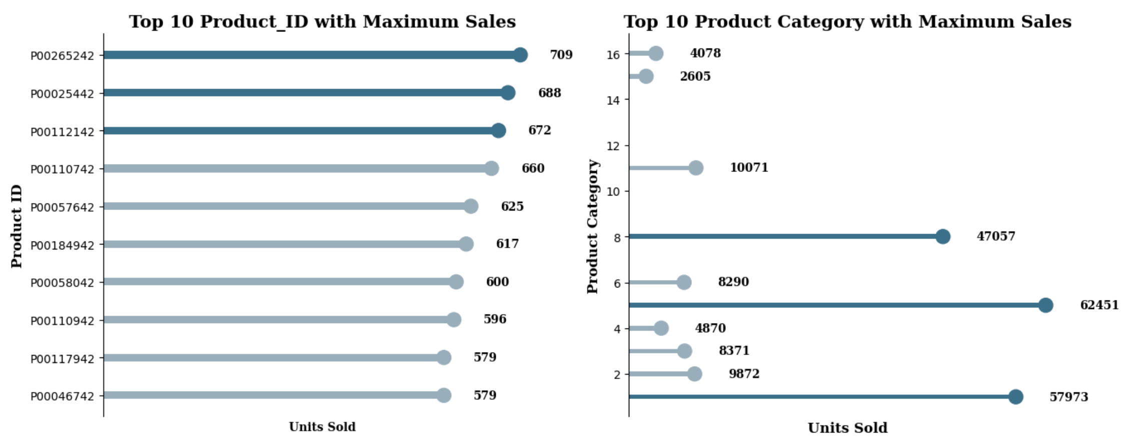
1.4.1 Top 10 Products and Categories: Sales Snapshot

- **`Top 10 Products and Product Categories`** which has sold most during Black Friday Sales

[22]:
```
#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')
```

```python
#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()
```



1. Top 10 Products Sold - The top-selling products during Walmart's Black Friday sales are characterized by a `relatively small variation` in sales numbers, suggesting that Walmart offers a variety of products that many different customers like to buy.

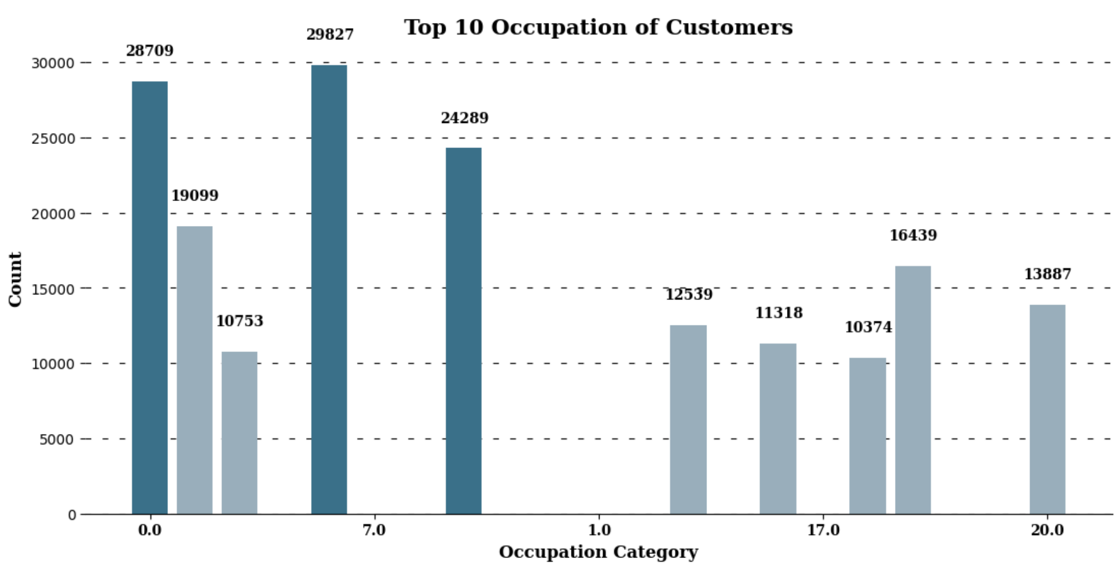
2. Top 10 Product Categories - Categories `5,1 and 8` have significantly outperformed other categories with combined Sales of nearly `75%` of the total sales suggesting a strong preference for these products among customers.

Top 10 Customer Occupation

- **`Top 10 Occupation of Customer`** in Black Friday Sales

[23]:
```python
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 = 'center')

#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()
```



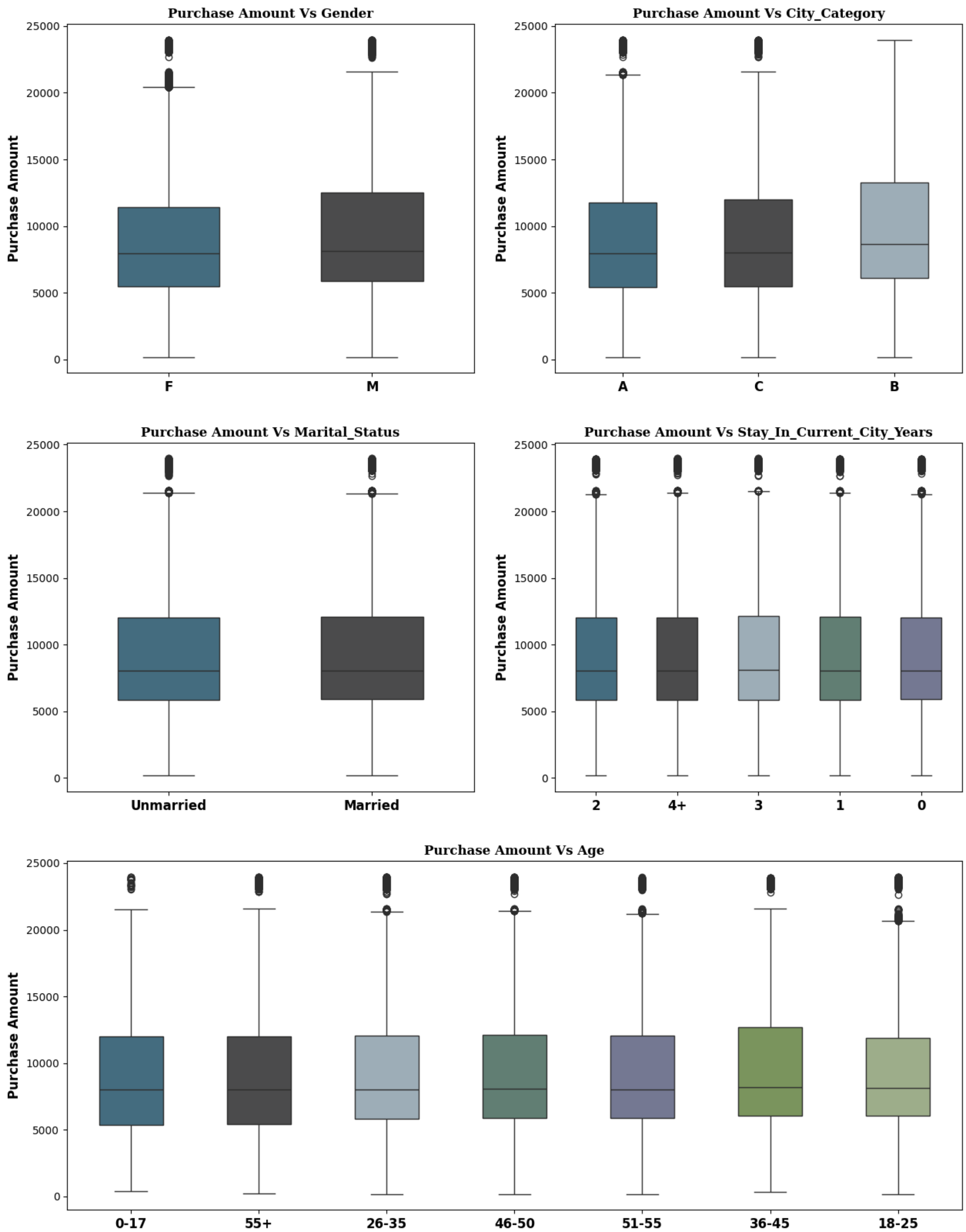
- Customers with Occupation category `4,0 and 7` contributed significantly i.e. almost 37% of the total purchases suggesting that these occupations have a high demand for Walmart products or services, or that they have more disposable income to spend on Black Friday.

1.5 Bivariate Analysis

Exploring Purchase Patterns

- Boxplots of Purchase Amount Across various Variables

[24]:
```
#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_Current_City_Yea
↪

    #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','#9EB384']
    sns.boxplot(data = df, x = k, y  = 'Purchase' ,ax = ax0,width = 0.5,␣
↪palette =color_map)

    #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()
```



- Out of all the variables analysed above, it's noteworthy that the purchase amount remains `relatively stable` regardless of the variable under consideration. As indicated in the data, the `median purchase amount` consistently hovers around `8,000 USD`, regardless of the specific variable being examined.

Gender VS Purchase Amount

1.6 Data Visualization

[25]:
```
#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
```

[25]:

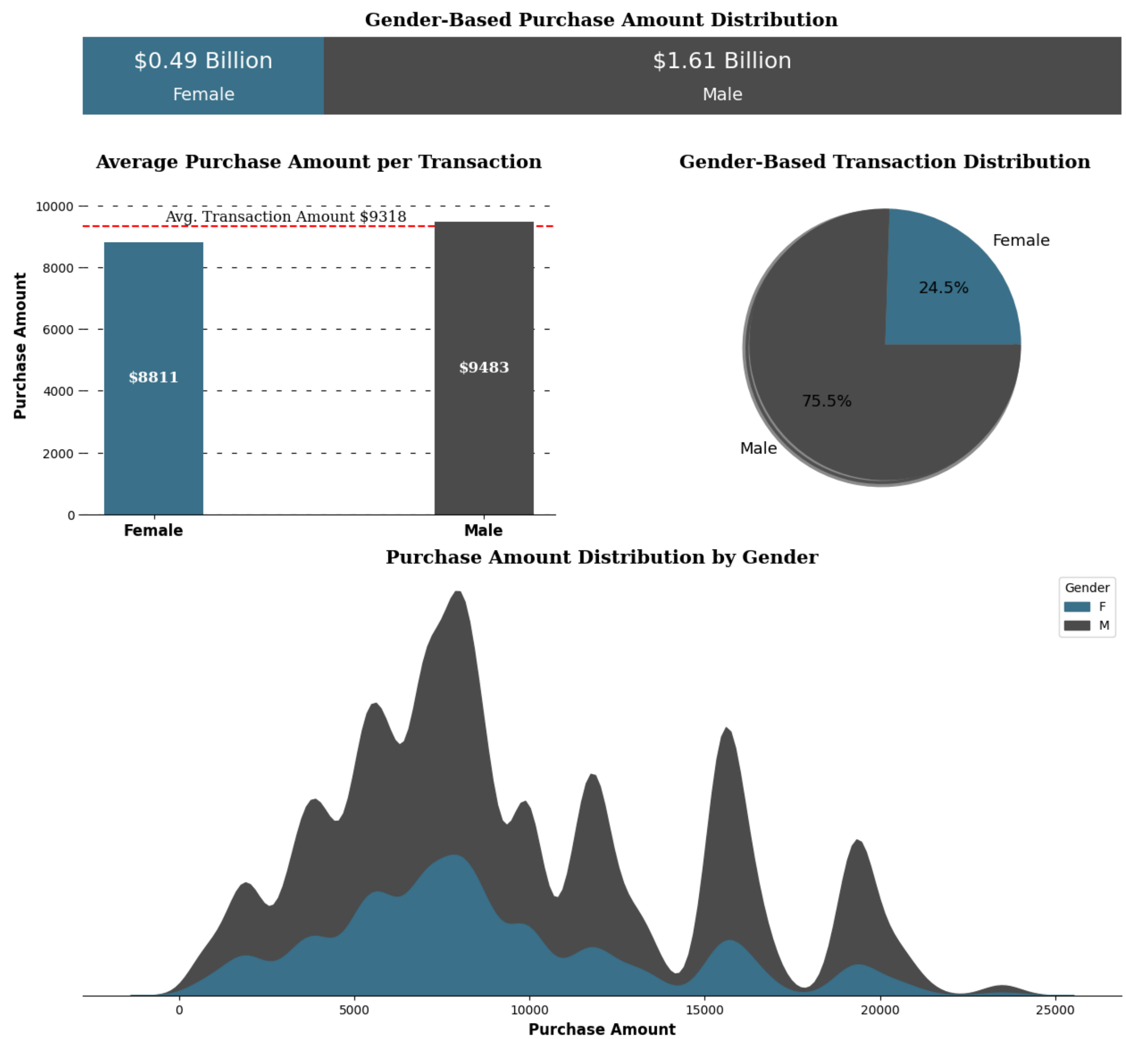
	Gender	sum	count	sum_in_billions	%sum	per_purchase
0	Female	4.863131e+08	55192	0.49	0.232	8811.0
1	Male	1.613905e+09	170197	1.61	0.768	9483.0

[26]:
```
#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 =  "#4b4b4c",label = 'Male' )

#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,'weight':'bold'})


                                            #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']:.0f}",
             {'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,'weight':'bold'})

                                        # 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, 'color': 'black'})

#setting title for visual
ax2.set_title('Gender-Based Transaction Distribution',{'font':'serif', 'size':
↪15,'weight':'bold'})

                                        # 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, alpha = 1,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 Gender',{'font':'serif', 'size':
 ↪15,'weight':'bold'})

plt.show()
```



1. Total Sales and Transactions Comparison - The total purchase amount and number of transactions by male customers was `more` than `three times` the amount and transactions by female customers indicating that they had a more significant impact on the Black Friday sales.

2. Average Transaction Value - The average purchase amount per transaction was slightly higher for male customers than female customers `($9438 vs $8735)`.

3. Distribution of Purchase Amount - As seen above, the purchase amount for both the genders is `not normally distributed`.

1.7 Confidence Interval Construction: Estimating Average Purchase Amount per Transaction

1. Step 1 - Building CLT Curve - As seen above, the purchase amount distribution is not Normal. So we need to use `Central Limit Theorem`. It states the distribution of sample means will approximate a normal distribution, regardless of the underlying population distribution

2. Step 2 - Building Confidence Interval - After building CLT curve, we will create a confidence interval predicting population mean at `99%,95% and 90% Confidence level`.

Note - We will use different sample sizes of [100,1000,5000,50000]

[27]:
```
#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
```

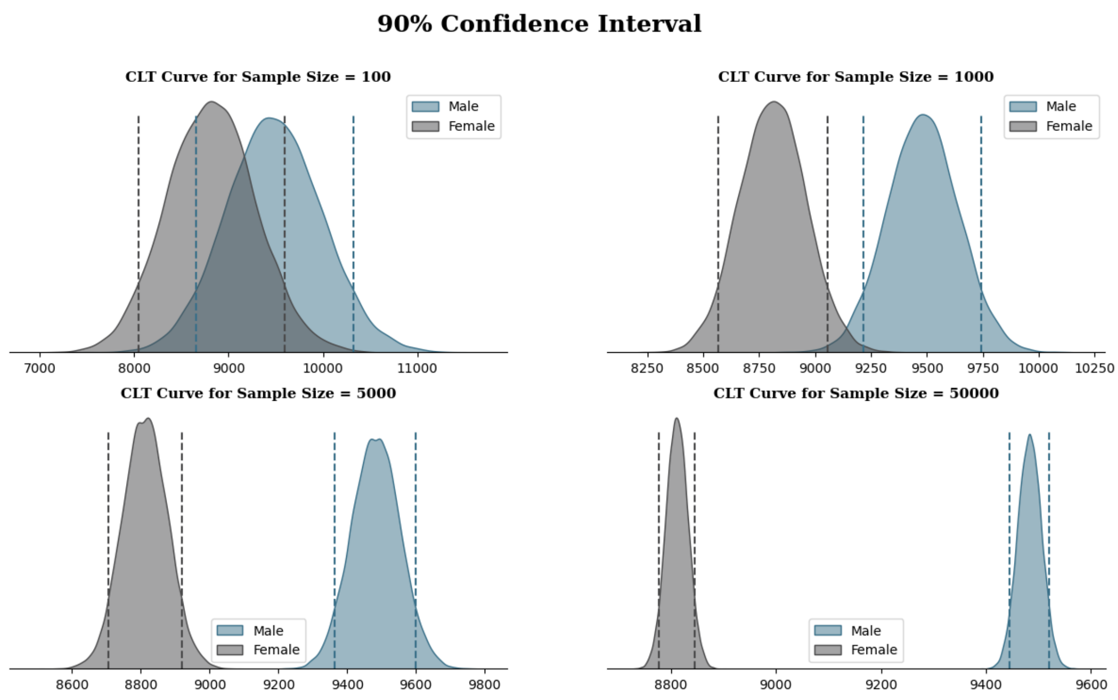
[28]:
```
#defining a function for plotting the visual for given confidence interval

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

        #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:
```

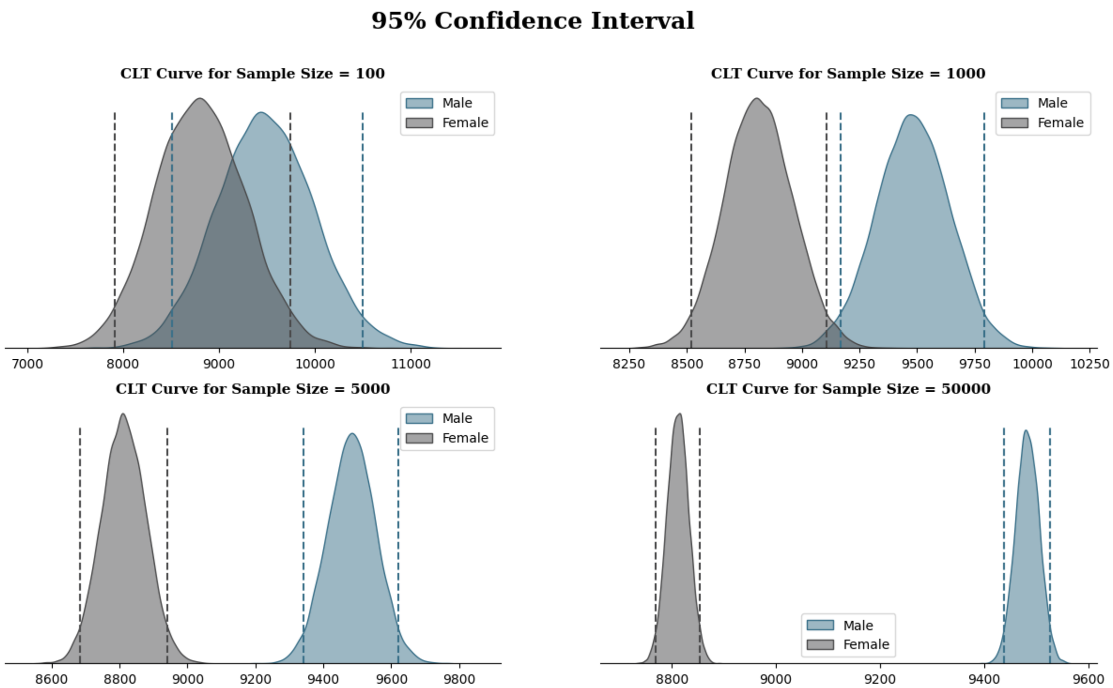
```
            ax.axvline(x = k,ymax = 0.9, color ="#4b4b4c",linestyle = '--')


        #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,'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
```

[29]:

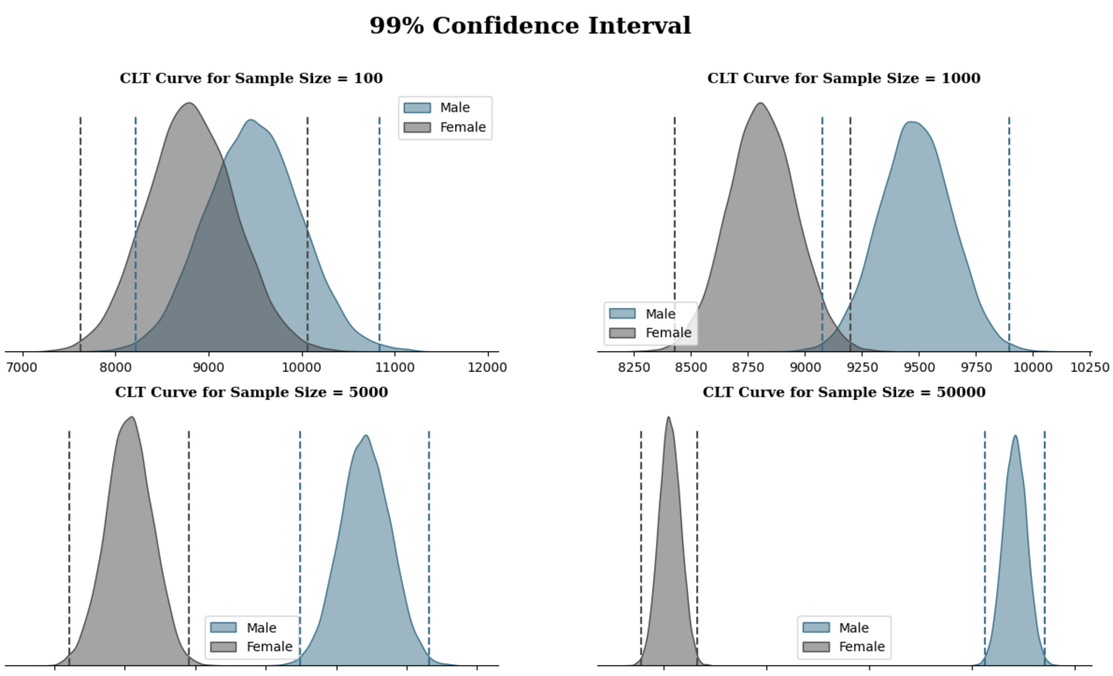
```
m_samp_90,f_samp_90 = plot(90)
```



[30]:
```
m_samp_95,f_samp_95 = plot(95)
```



[31]:
```
m_samp_99,f_samp_99 = plot(99)
```



[32]:

```
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_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

Gender	Sample Size = 100	Sample Size = 1000	Sample Size = 5000	Sample Size = 50000
Male	CI = 8656 – 10328, Range = 1672	CI = 9217 – 9745, Range = 528	CI = 9365 – 9600, Range = 235	CI = 9445 – 9520, Range = 75
Female	CI = 8048 – 9595, Range = 1547	CI = 8567 – 9055, Range = 488	CI = 8705 – 8920, Range = 215	CI = 8777 – 8846, Range = 69

95% Confidence Interval Summary

Gender	Sample Size = 100	Sample Size = 1000	Sample Size = 5000	Sample Size = 50000
Male	CI = 8511 – 10496, Range = 1985	CI = 9167 – 9792, Range = 625	CI = 9342 – 9622, Range = 280	CI = 9438 – 9526, Range = 88
Female	CI = 7910 – 9744, Range = 1834	CI = 8521 – 9106, Range = 585	CI = 8683 – 8942, Range = 259	CI = 8770 – 8853, Range = 83

99% Confidence Interval Summary

Gender	Sample Size = 100	Sample Size = 1000	Sample Size = 5000	Sample Size = 50000
Male	CI = 8219 – 10840, Range = 2621	CI = 9078 – 9896, Range = 818	CI = 9298 – 9663, Range = 365	CI = 9425 – 9541, Range = 116
Female	CI = 7625 – 10063, Range = 2438	CI = 8429 – 9199, Range = 770	CI = 8642 – 8982, Range = 340	CI = 8757 – 8865, Range = 108

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

2. Confidence Intervals - From the above analysis, we can see that except for the Sample Size of 100, `the confidence interval do not overlap` as the sample size increases. This means that there is a statistically significant difference between the average spending per transaction for men and women within the given samples.

3. Population Average - We are `95% confident` that the true population average for `males` falls between `$9,393 and $9,483`, and for `females`, it falls between `$8,692 and $8,777`.

4. Women spend less - `Men tend to spend more money per transaction on average than women`, as the upper bounds of the confidence intervals for men are consistently higher than those for women across different sample sizes.

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

5.1. Segmentation Opportunities - Walmart can create targeted marketing campaigns, loyalty programs, or product bundles to cater to the distinct spending behaviors of male and female customers. This approach may help maximize revenue from each customer segment.

5.2. Pricing Strategies - Based on the above data of average spending per transaction by gender, they might adjust pricing or discount strategies to incentivize higher spending among male customers while ensuring competitive pricing for female-oriented products.

1.7.2 `Note`

- Moving forward in our analysis, we will use `95% Confidence Level` only.

##Marital Status VS Purchase Amount

1.8 Data Visualization

[33]:
```
#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
```

[33]:

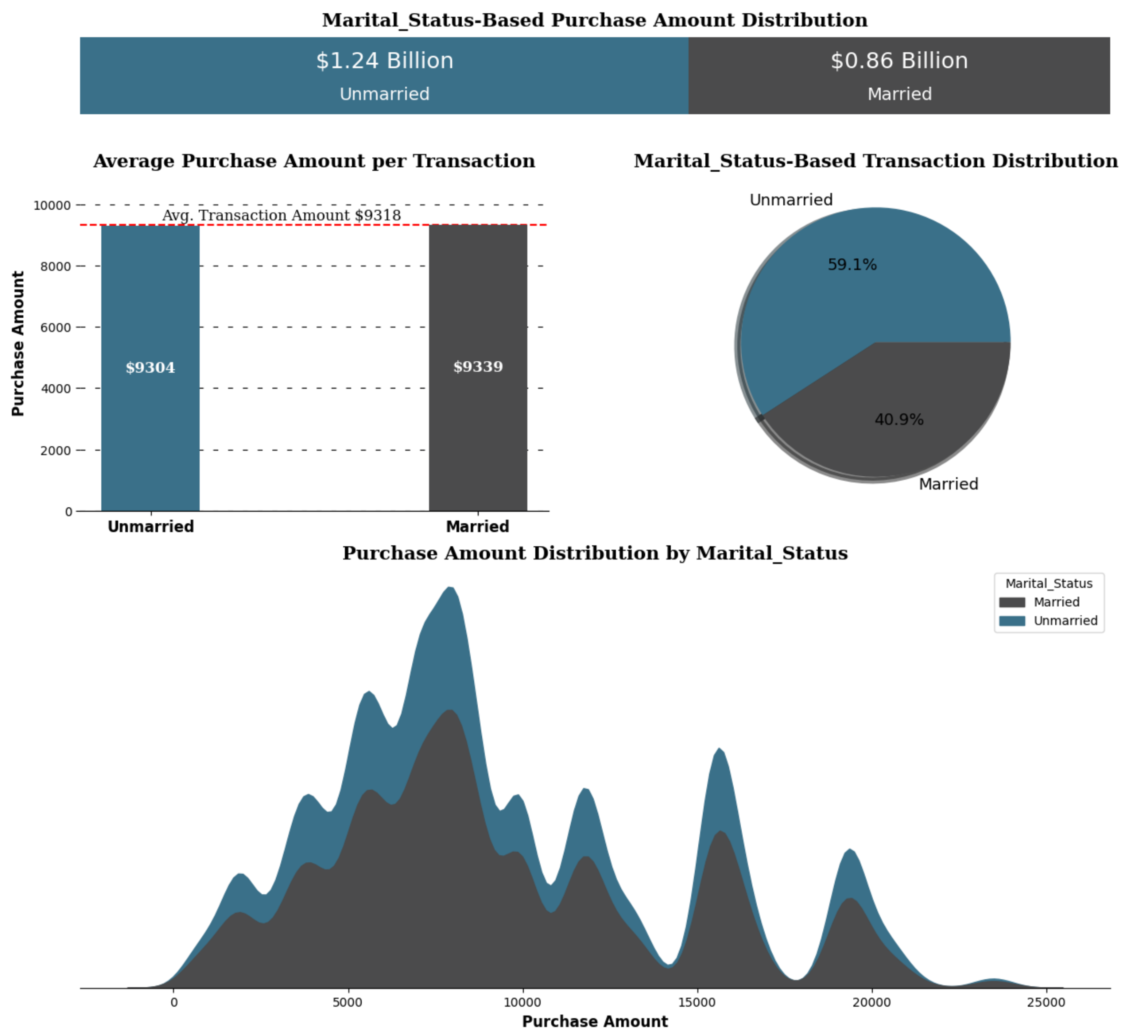
	Marital_Status	sum	count	sum_in_billions	%sum	per_purchase
0	Unmarried	1.240269e+09	133306	1.24	0.591	9304.0
1	Married	8.599497e+08	92083	0.86	0.409	9339.0

[34]:
```
#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 = 'Unmarried')
ax.barh(temp.loc[0,'Marital_Status'],width = temp.loc[1,'%sum'],left =temp.
 ↪loc[0,'%sum'], color =  "#4b4b4c",label = 'Married')

#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':15,'weight':'bold'})


                                            #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 = 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,'Marital_Status'],temp.loc[i,'per_purchase']/2,f"${temp.
 ↪loc[i,'per_purchase']:.0f}",
```

```
            {'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,'weight':'bold'})

                                        # 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, 'color': 'black'})

#setting title for visual
ax2.set_title('Marital_Status-Based Transaction Distribution',{'font':'serif',␣
↪'size':15,'weight':'bold'})

                                        # 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 = True, alpha = 1,
            ax = ax3,hue_order = ['Married','Unmarried'])

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

```python
# 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,'weight':'bold'})

plt.show()
```



1. Total Sales and Transactions Comparison - The total purchase amount and number of transactions by Unmarried customers was `more` than `20%` the amount and transactions by married customers indicating that they had a more significant impact on the Black Friday sales.

2. Average Transaction Value - The average purchase amount per transaction was almost similar for married and unmarried customers `($9261 vs $9266)`.

3. Distribution of Purchase Amount - As seen above, the purchase amount for both married and unmarried customers is `not normally distributed`."

Confidence Interval Construction: Estimating Average Purchase Amount per Transaction

1. Step 1 - Building CLT Curve - As seen above, the purchase amount distribution is not Normal. So we need to use `Central Limit Theorem`. It states the distribution of sample means will approximate a normal distribution, regardless of the underlying population distribution

2. Step 2 - Building Confidence Interval - After building CLT curve, we will create a confidence interval predicting population mean at `95% Confidence level`.

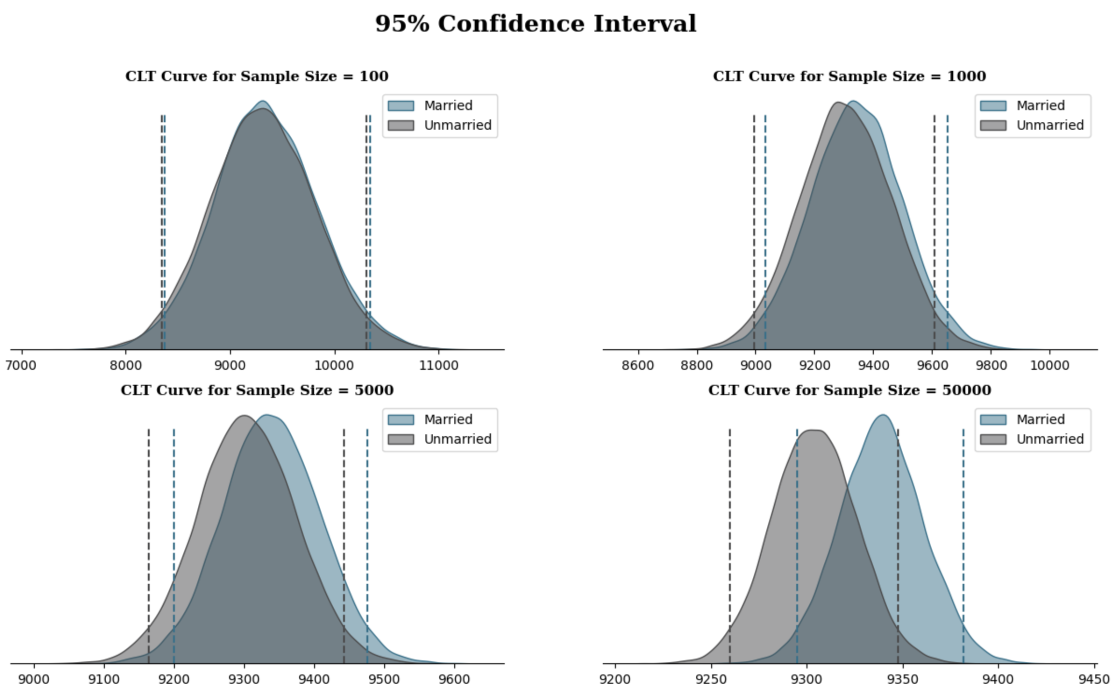
Note - We will use different sample sizes of [100,1000,5000,50000]

[35]:
```python
#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_means':unmarried_means})

                                                                   #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 =␣
 ↪True, alpha = 0.5,ax = ax,label = 'Married')
        sns.kdeplot(data = temp_df,x = 'unmarried_means',color ="#4b4b4c" ,fill␣
 ↪= True, alpha = 0.5,ax = ax,label = 'Unmarried')

        #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,'weight':'bold'})

        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
```

[36]:
```
m_samp_95,u_samp_95 = plot(95)
```



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

[37]:
```
#setting the plot style
fig,ax = plt.subplots(figsize = (20,3))

#list for collecting ci for given cl
m_ci = ['Married']
u_ci = ['Unmarried']
```

```python
#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[0]):.0f}")

#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[0]):.0f}")

                                    #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 = 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"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 = 8377 – 10343, Range = 1966	CI = 9032 – 9654, Range = 622	CI = 9201 – 9476, Range = 275	CI = 9295 – 9382, Range = 87
Unmarried	CI = 8344 – 10308, Range = 1964	CI = 8995 – 9610, Range = 615	CI = 9164 – 9443, Range = 279	CI = 9260 – 9348, Range = 88

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

2. Confidence Intervals - From the above analysis, we can see that `the confidence interval overlap` for all the sample sizes. This means that there is `no statistically significant difference` between the average spending per transaction for married and unmarried customers within the given samples.

3. Population Average - We are `95% confident` that the true population average for `married customers` falls between `$9,217 and $9,305`, and for `unmarried customers`, it falls between `$9,222 and $9,311`.

4. Both the customers spend equal - The overlapping confidence intervals of average spending for married and unmarried customers indicate that `both married and unmarried customers spend a similar amount per transaction`. This implies a resemblance in spending behavior between the two groups.

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

5.1. Marketing Resources - Walmart may not need to allocate marketing resources specifically targeting one group over the other. Instead, they can focus on broader marketing strategies that appeal to both groups.

Customer Age VS Purchase Amount

Data Visualization

[38]:
```
#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
```

[38]:

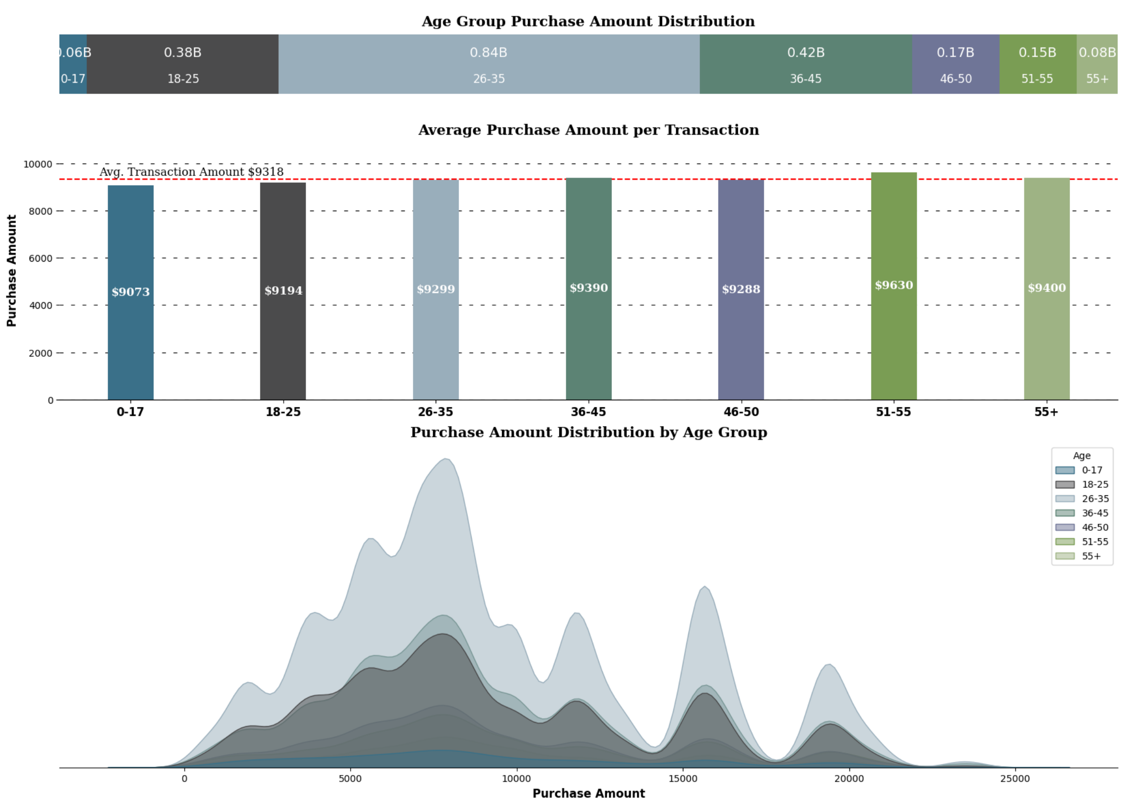
	Age	sum	count	sum_in_billions	%sum	per_purchase
0	0-17	55072934.0	6070	0.06	0.026	9073.0
1	18-25	379734589.0	41303	0.38	0.181	9194.0
2	26-35	835929160.0	89894	0.84	0.398	9299.0
3	36-45	421645254.0	44905	0.42	0.201	9390.0
4	46-50	171476032.0	18462	0.17	0.082	9288.0
5	51-55	153166393.0	15905	0.15	0.073	9630.0
6	55+	83194140.0	8850	0.08	0.040	9400.0

[39]:

```
#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],label = temp.loc[i,'Age'])
    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':'bold'})
```

```
                                              #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']:.0f}",
             {'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,'weight':'bold'})
```

```
                                        # 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, alpha = 0.5,
            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,'weight':'bold'})

plt.show()
```



1. Total Sales Comparison - Age group between `26 - 45` accounts to almost `60%` of the total sales suggesting that Walmart's Black Friday sales are most popular among these age groups.

- The age group `0-17` has the lowest sales percentage `(2.6%)`, which is expected as they may not have as much purchasing power. Understanding their preferences and providing special offers could be beneficial, especially considering the potential for building customer loyalty as they age.

2. Average Transaction Value - While there is not a significant difference in per purchase spending among the age groups, the `51-55` age group has a relatively low sales percentage `(7.2%)` but they have the highest per purchase spending at `9535`. Walmart could consider strategies to attract and retain this high-spending demographic.

3. Distribution of Purchase Amount - As seen above, the purchase amount for all age groups is `not normally distributed`.

1.9 Confidence Interval Construction: Estimating Average Purchase Amount per Transaction

1. Step 1 - Building CLT Curve - As seen above, the purchase amount distribution is not Normal. So we need to use `Central Limit Theorem`. It states the distribution of sample means will approximate a normal distribution, regardless of the underlying population distribution

2. Step 2 - Building Confidence Interval - After building CLT curve, we will create a confidence interval predicting population mean at `95% Confidence level`.

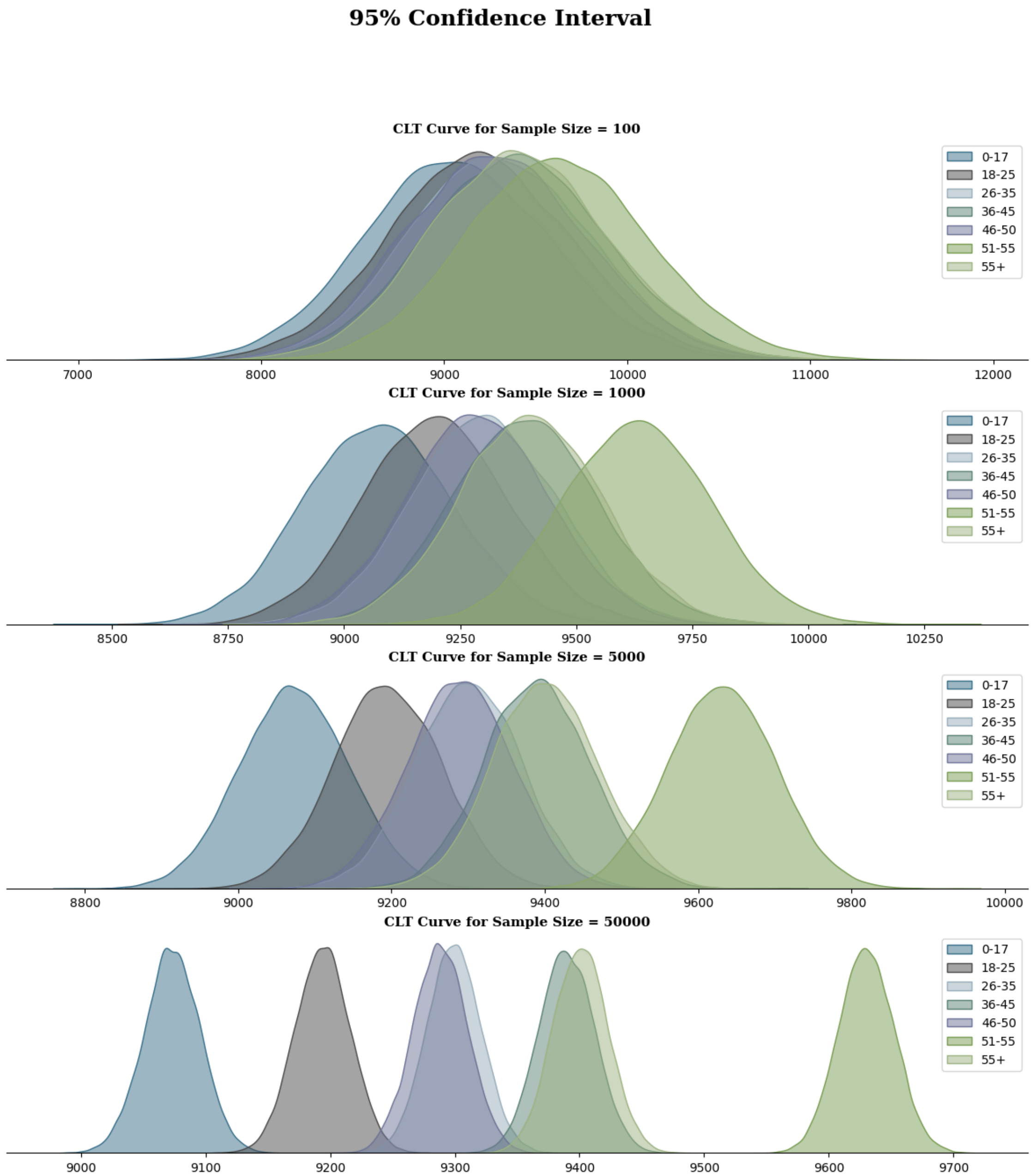
Note - We will use different sample sizes of [100,1000,5000,50000]

[40]:
```python
#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':
↪l4,'46-50':l5,'51-55':l6,'55+':l7})

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

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

            sns.kdeplot(data = temp_df,x = q,color =p ,fill = True, alpha = 0.
↪5,ax = ax,label = q)


        #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
```

[41]:
```
samples1,samples2,samples3,samples4,samples5,samples6,samples7 = plot(95)
```



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

[42]:
```
#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'],['51-55'],['55+']
```

```python
#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),(samples6,

for s,c in samples:
    for i in s:
        s_range = confidence_interval(s[i],95)
        c.append(f"CI = ${s_range[0]:.0f} - ${s_range[1]:.0f}, Range = {(s_range[1] - s_range[0]):.0f}")



                                    #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','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"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 = 8091 – 10104, Range = 2013	CI = 8759 – 9391, Range = 632	CI = 8933 – 9214, Range = 281	CI = 9028 – 9117, Range = 89
18-25	CI = 8220 – 10181, Range = 1961	CI = 8886 – 9505, Range = 619	CI = 9056 – 9333, Range = 277	CI = 9150 – 9237, Range = 87
26-35	CI = 8347 – 10297, Range = 1950	CI = 8995 – 9609, Range = 614	CI = 9161 – 9434, Range = 273	CI = 9256 – 9343, Range = 87
36-45	CI = 8430 – 10394, Range = 1964	CI = 9085 – 9701, Range = 616	CI = 9254 – 9529, Range = 275	CI = 9345 – 9434, Range = 89
46-50	CI = 8338 – 10269, Range = 1931	CI = 8982 – 9597, Range = 615	CI = 9153 – 9422, Range = 269	CI = 9245 – 9332, Range = 87
51-55	CI = 8649 – 10640, Range = 1991	CI = 9312 – 9945, Range = 633	CI = 9492 – 9769, Range = 277	CI = 9587 – 9674, Range = 87
55+	CI = 8468 – 10384, Range = 1916	CI = 9091 – 9709, Range = 618	CI = 9265 – 9540, Range = 275	CI = 9357 – 9444, Range = 87

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

2. Confidence Intervals and customer spending patterns - From the above analysis, we can see that `the confidence interval overlap` for some of the age groups. We can club the average spending into following age groups - - 0 - 17 - Customers in this age group have the lowest spending per transaction - 18 - 25, 26 - 35, 46 - 50 - Customers in these age groups have overlapping confidence intervals indicating similar buying characteristics - 36 - 45, 55+ - Customers in these age groups have overlapping confidence intervals indicating and similar spending patterns - 51 - 55 - Customers in this age group have the highest spending per transaction

3. Population Average - We are `95% confident` that the true population average for following age groups falls between the below range -

```
- 0 - 17 = \$ 8,888 to 8,979
- 18 - 25 = \$ 9,125 to 9,213
- 26 - 35 = \$ 9,209 to 9,297
- 36 - 45 = \$ 9,288 to 9,376
- 46 - 50 = \$ 9,165 to 9,253
- 51 - 55 = \$ 9,490 to 9,579
- 55+ = \$ 9,292 to 9,381
```

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

4.1. Targeted Marketing - 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. Walmart can also tailor their product selection and marketing strategies to appeal to the preferences and needs of this age group

4.2. 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.

4.3 Premium Services - Recognizing that customers in the 51 - 55 age group have the highest spending per transaction, Walmart can explore opportunities to enhance the shopping experience for this demographic. This might involve offering premium services, personalized recommendations, or loyalty programs that cater to the preferences and spending habits of this age group.

1.10 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.