# walmart

February 25, 2024

[1]:
```
#Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

[2]:
```
# loading the dataset
walmart=pd.read_csv(r"C:\Users\shobh\OneDrive\Desktop\walmart.csv")
```

[3]:
```
#storing walmart dataset as df
df=walmart
df
```

[3]:
```
        User_ID Product_ID Gender    Age  Occupation City_Category  \
0       1000001  P00069042      F   0-17          10             A
1       1000001  P00248942      F   0-17          10             A
2       1000001  P00087842      F   0-17          10             A
3       1000001  P00085442      F   0-17          10             A
4       1000002  P00285442      M    55+          16             C
…           …          …     …     …            …
550063  1006033  P00372445      M  51-55          13             B
550064  1006035  P00375436      F  26-35           1             C
550065  1006036  P00375436      F  26-35          15             B
550066  1006038  P00375436      F    55+           1             C
550067  1006039  P00371644      F  46-50           0             B

       Stay_In_Current_City_Years  Marital_Status  Product_Category  Purchase
0                               2               0                 3      8370
1                               2               0                 1     15200
2                               2               0                12      1422
3                               2               0                12      1057
4                              4+               0                 8      7969
…                            …               …                …       …
550063                          1               1                20       368
550064                          3               0                20       371
550065                         4+               1                20       137
```

```
550066                          2               0                20       365
550067                         4+               1                20       490

[550068 rows x 10 columns]
```

[4]:
```
walmart.describe()
```

[4]:
```
            User_ID     Occupation  Marital_Status  Product_Category  \
count  5.500680e+05  550068.000000   550068.000000     550068.000000
mean   1.003029e+06       8.076707        0.409653          5.404270
std    1.727592e+03       6.522660        0.491770          3.936211
min    1.000001e+06       0.000000        0.000000          1.000000
25%    1.001516e+06       2.000000        0.000000          1.000000
50%    1.003077e+06       7.000000        0.000000          5.000000
75%    1.004478e+06      14.000000        1.000000          8.000000
max    1.006040e+06      20.000000        1.000000         20.000000

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

[5]:
```
walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     550068 non-null  int64
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64
 8   Product_Category            550068 non-null  int64
 9   Purchase                    550068 non-null  int64
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

[6]:
```
walmart.shape
```

[6]:
```
(550068, 10)
```

[7]:
```
walmart.size
```

[7]:
```
5500680
```

[8]:
```
walmart.dtypes
```

[8]:
```
User_ID                        int64
Product_ID                    object
Gender                        object
Age                           object
Occupation                     int64
City_Category                 object
Stay_In_Current_City_Years    object
Marital_Status                 int64
Product_Category               int64
Purchase                       int64
dtype: object
```

# 1 Insights

1. Based on the analysis, the dataset consists of 10 features containing various alphanumeric data types. With the exception of the 'Purchase' column, the remaining columns contain categorical data.
2. To enhance the dataset's clarity and optimize memory usage, we intend to convert all non-numeric columns to the categorical data type. This conversion will streamline data representation and enable more efficient storage and processing.

## 1.1 Conversion of datatypes

[9]:
```
for column_name in walmart.columns[:-1]:
    walmart[column_name]=pd.Categorical(walmart[column_name])
walmart
```

[9]:
```
        User_ID Product_ID Gender    Age Occupation City_Category  \
0       1000001  P00069042      F   0-17         10             A
1       1000001  P00248942      F   0-17         10             A
2       1000001  P00087842      F   0-17         10             A
3       1000001  P00085442      F   0-17         10             A
4       1000002  P00285442      M    55+         16             C
…           …         …      …    …      …            …
550063  1006033  P00372445      M  51-55         13             B
550064  1006035  P00375436      F  26-35          1             C
```

```
550065  1006036  P00375436       F  26-35         15              B
550066  1006038  P00375436       F    55+          1              C
550067  1006039  P00371644       F  46-50          0              B

       Stay_In_Current_City_Years Marital_Status Product_Category  Purchase
0                               2              0                3      8370
1                               2              0                1     15200
2                               2              0               12      1422
3                               2              0               12      1057
4                              4+              0                8      7969
…                           …            …              …       …
550063                          1              1               20       368
550064                          3              0               20       371
550065                         4+              1               20       137
550066                          2              0               20       365
550067                         4+              1               20       490

[550068 rows x 10 columns]
```

[10]:
```
walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     550068 non-null  category
 1   Product_ID                  550068 non-null  category
 2   Gender                      550068 non-null  category
 3   Age                         550068 non-null  category
 4   Occupation                  550068 non-null  category
 5   City_Category               550068 non-null  category
 6   Stay_In_Current_City_Years  550068 non-null  category
 7   Marital_Status              550068 non-null  category
 8   Product_Category            550068 non-null  category
 9   Purchase                    550068 non-null  int64
dtypes: category(9), int64(1)
memory usage: 10.3 MB
```

## 2 Statistical Summary

[11]:
```
walmart.describe(include='category')
```

[11]:
```
       User_ID Product_ID  Gender     Age  Occupation City_Category  \
count   550068     550068  550068  550068      550068        550068
unique    5891       3631       2       7          21             3
```

```
top     1001680  P00265242        M   26-35            4             B
freq       1026       1880   414259  219587        72308        231173

       Stay_In_Current_City_Years  Marital_Status  Product_Category
count                      550068          550068            550068
unique                          5               2                20
top                             1               0                 5
freq                       193821          324731            150933
```

## 3 Insights

1. Among the 550,068 transactions, there are 5,891 unique user IDs, indicating that multiple products were purchased by the same customers.
2. Within the dataset's 550,068 transactions, there are 3,631 unique products. The product with the code P00265242 emerges as the top seller, with a maximum of 1,880 units sold.
3. Gender distribution reveals a notable disparity in purchasing behavior, with approximately 75% (or 414,259 transactions) conducted by male customers.
4. Age groups within the dataset span seven unique categories. The age group of 26-35 accounts for the highest transaction count, totaling 219,587 transactions.
5. Regarding the duration of stay in the current city, customers residing for one year represent the largest group, with 193,821 transactions, surpassing those with longer stay durations (0, 2, 3, 4+ years).
6. Analysis of marital status indicates that unmarried customers contribute to 59% of the total transactions, while married customers account for the remaining 41%.

[12]:
```python
#Summary for numerical datatype

walmart.describe()
```

[12]:
```
            Purchase
count  550068.000000
mean     9263.968713
std      5023.065394
min        12.000000
25%      5823.000000
50%      8047.000000
75%     12054.000000
max     23961.000000
```

[13]:
```python
#Duplicate Values
df.duplicated().sum()
```

[13]: 0

# 4 Insight

- No duplicate values found

## 4.1 Detecting Null Values

[14]:
```
walmart.isnull().sum()
```

[14]:
```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category              0
Purchase                      0
dtype: int64
```

- Dataset doesn't contain any null values.

## 4.2 Univariate Analysis

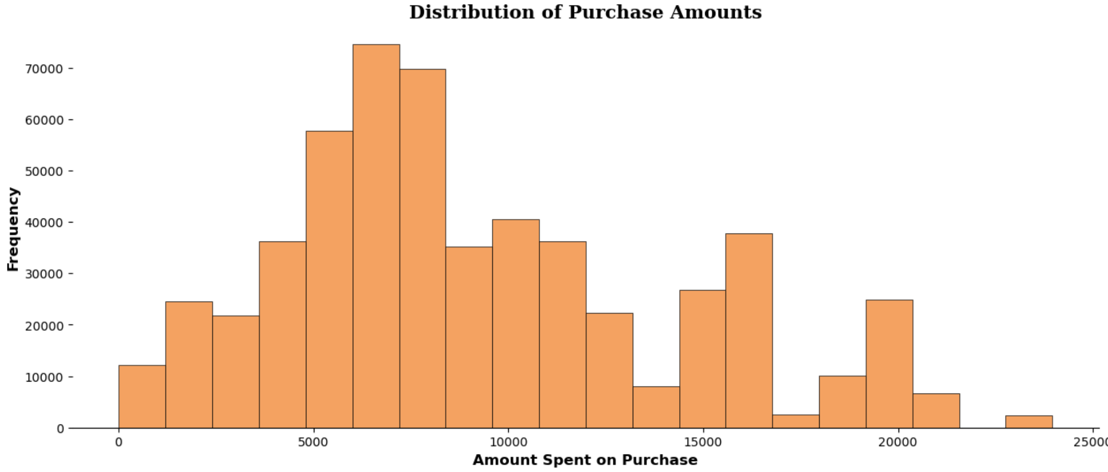
## 4.3 1. Numerical variables

**Searching Outliers for continuous variable**

- Purchase column is the only continuous variable and we will first create a histogram for the Purchase column and then search for outliers using a Boxplot

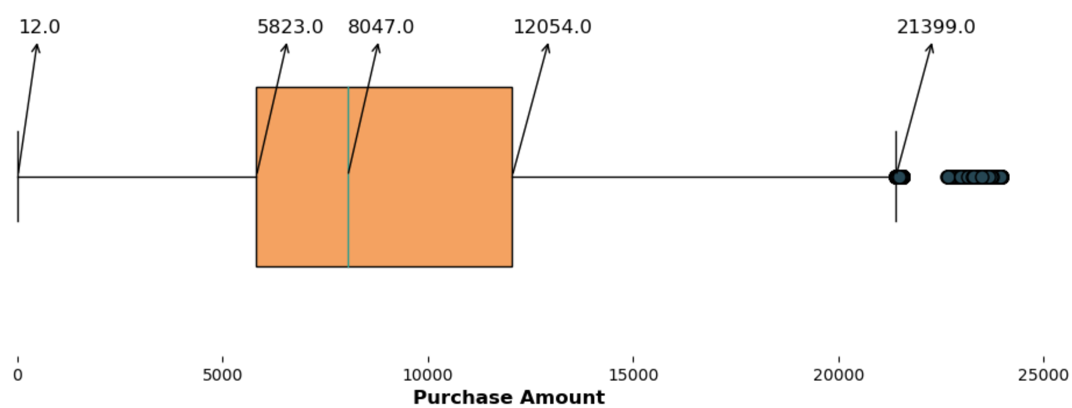
[15]:
```
#Histogram

figure = plt.figure(figsize=(15, 10))

grid_spec = figure.add_gridspec(2, 1, height_ratios=[0.65, 0.35])

purchase_histogram = figure.add_subplot(grid_spec[0, 0])
purchase_histogram.hist(df['Purchase'], color='#F4A261', linewidth=0.5,␣
  ↪edgecolor='black', bins=20)
purchase_histogram.set_xlabel('Amount Spent on Purchase', fontsize=12,␣
  ↪fontweight='bold')
purchase_histogram.set_ylabel('Frequency', fontsize=12, fontweight='bold')

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

purchase_histogram.set_title('Distribution of Purchase Amounts', {'font':␣
  ↪'serif', 'size': 15, 'weight': 'bold'})
```

[15]: Text(0.5, 1.0, 'Distribution of Purchase Amounts')




[16]:
```
#Boxplot
fig, ax = plt.subplots(figsize=(12, 4))

boxplot = ax.boxplot(df['Purchase'], vert=False, patch_artist=True, widths=0.5)

boxplot['boxes'][0].set(facecolor='#F4A261')

# Customize median line
boxplot['medians'][0].set(color='#2A9D8F')

# Customize outlier markers
for flier in boxplot['fliers']:
    flier.set(marker='o', markersize=8, markerfacecolor='#264653')

# Removing the axis lines
for spine in ax.spines.values():
    spine.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 median (Q2)

for i, j in info:
    ax.annotate(text=f"{i:.1f}", xy=(i, 1), xytext=(i, 1.4), fontsize=12,
                arrowprops=dict(arrowstyle="<-", lw=1,␣
 ↪connectionstyle="arc,rad=0"))

    ax.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 annotation
ax.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
ax.set_yticks([])

# Adding x-axis label
ax.set_xlabel('Purchase Amount', fontweight='bold', fontsize=12)

# Show the plot
plt.show()
```




[17]:
```
#Number of Outliers
len(df.loc[df['Purchase']>21399,'Purchase'])
```

[17]: 2677

## 5 Insights

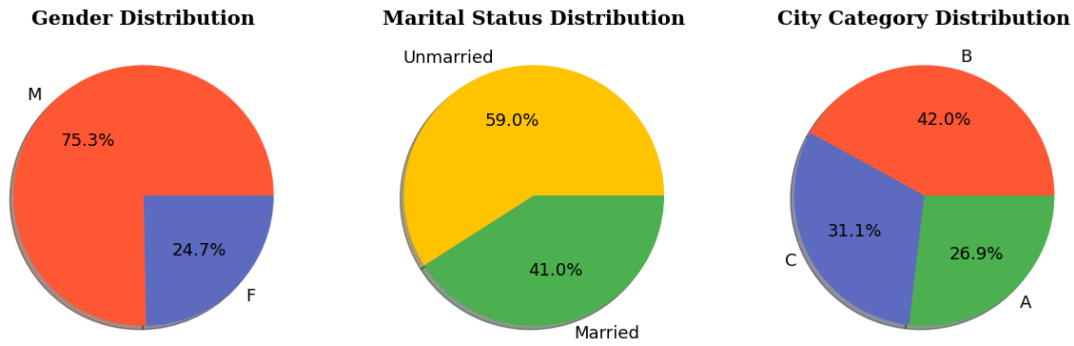
- **Outliers**:
  - A total of 2677 outliers were identified, constituting approximately 0.48% of the total purchase data. These outliers will not be removed, as they represent a diverse range of spending behaviors during the sale. This highlights the importance of tailoring marketing strategies to cater to both regular and high-value customers, thus maximizing revenue potential.

- **Distribution**:
  - The data indicates that the majority of customers made purchases ranging from 5,823 USD to 12,054 USD, with the median purchase amount being 8,047 USD. The lower limit of 12 USD and upper limit of 21,399 USD demonstrate significant variability in customer spending patterns.

### 5.1 2. Categorical variables

**2.1 Distribution of Gender,Marital status and City distribution of Customers**

[18]:
```
df['Marital_Status'] = df['Marital_Status'].map({0: 'Unmarried', 1: 'Married'})

# Set the plot style

plt.figure(figsize=(15, 12))

# Create subplots for gender, marital status, and city category distributions
plt.subplot(1, 3, 1)
gender_colors = ['#FF5733', '#5C6BC0']
plt.pie(df['Gender'].value_counts(), labels=df['Gender'].value_counts().index,␣
↪autopct='%.1f%%',
        shadow=True, colors=gender_colors, textprops={'fontsize': 13, 'color':␣
↪'black'})
plt.title('Gender Distribution', fontdict={'font': 'serif', 'size': 15,␣
↪'weight': 'bold'})

plt.subplot(1, 3, 2)
marital_colors = ['#FFC300', '#4CAF50']
plt.pie(df['Marital_Status'].value_counts(), labels=df['Marital_Status'].
↪value_counts().index, autopct='%.1f%%',
        shadow=True, colors=marital_colors, textprops={'fontsize': 13, 'color':␣
↪'black'})
plt.title('Marital Status Distribution', fontdict={'font': 'serif', 'size': 15,␣
↪'weight': 'bold'})

plt.subplot(1, 3, 3)
city_colors = ['#FF5733', '#5C6BC0', '#4CAF50']
plt.pie(df['City_Category'].value_counts(), labels=df['City_Category'].
↪value_counts().index, autopct='%.1f%%',
        shadow=True, colors=city_colors, textprops={'fontsize': 13, 'color':␣
↪'black'})
plt.title('City Category Distribution', fontdict={'font': 'serif', 'size': 15,␣
↪'weight': 'bold'})

plt.show()
```




# 6 Insights

**1. Gender Distribution** - The data reveals notable differences in purchasing patterns between males and females during the Black Friday event.

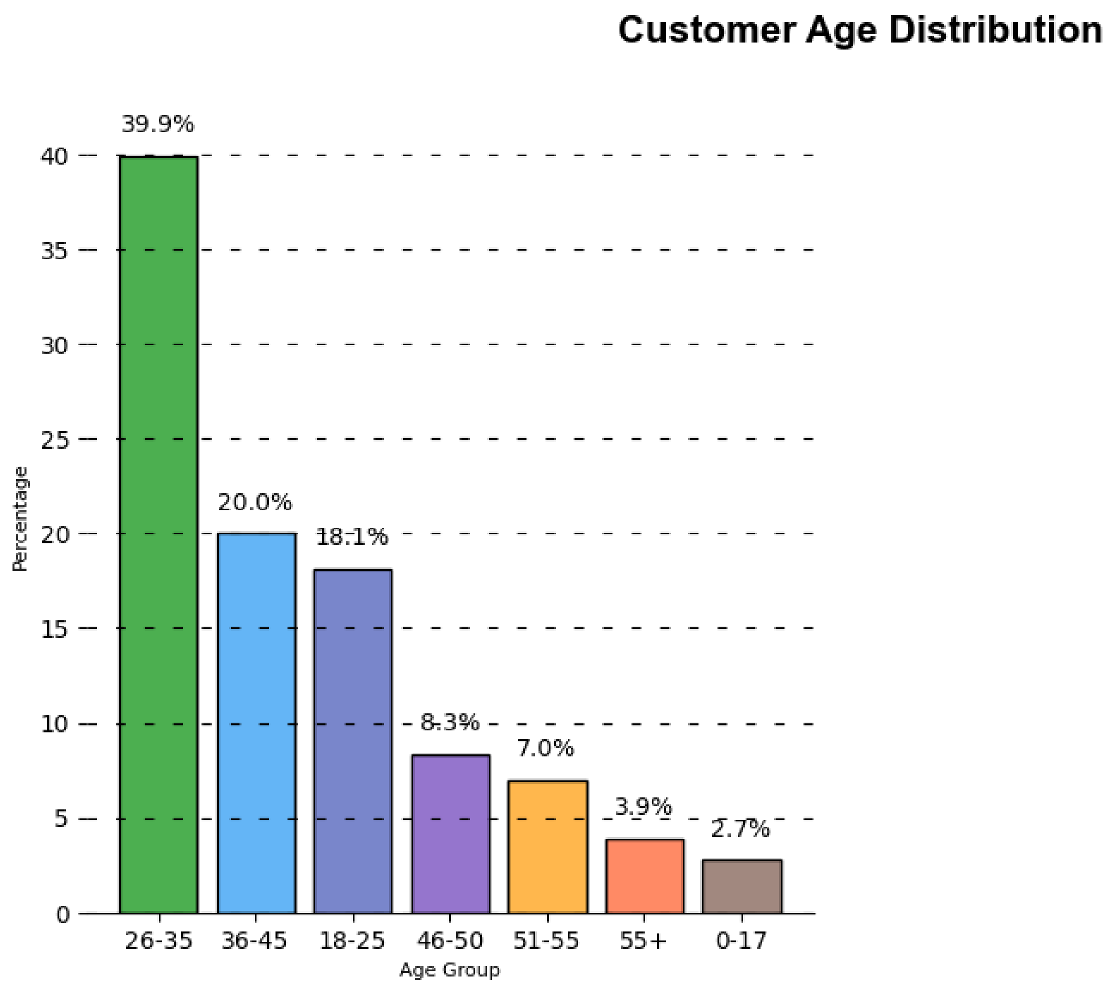
**2. Marital Status** - With unmarried customers constituting a larger portion of transactions, there's an opportunity to explore tailored marketing strategies or promotions targeting this demographic.

**3. City Category** - City B recorded the highest number of transactions, trailed by City C and City A, respectively.

### 2.2 Age Distribution of customers

[19]:
```
import matplotlib.pyplot as plt

# Set the plot style and size
plt.figure(figsize=(12,6))

# Creating a barplot for age distribution
plt.subplot(1, 2, 1)
age_counts = df['Age'].value_counts(normalize=True) * 100
color_map = ["#4CAF50", "#64B5F6", "#7986CB", "#9575CD", "#FFB74D", "#FF8A65",␣
↪"#A1887F"]
plt.bar(x=age_counts.index, height=age_counts.values, color=color_map,␣
↪edgecolor='black')
# Adding percentage values above the bars
for i in range(len(age_counts)):
    plt.text(x=i, y=age_counts[i] + 1, s=f'{age_counts[i]:.1f}%', ha='center',␣
↪va='bottom')
# Adding grid lines
plt.grid(color='black', linestyle='--', axis='y', zorder=0, dashes=(5, 10))
# Removing axis lines
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
```

```
plt.gca().spines['left'].set_visible(False)
# Adding axis labels
plt.ylabel('Percentage', fontsize=8)
plt.xlabel('Age Group',fontsize=8)
plt.xticks()

# Setting title for visual
plt.suptitle('Customer Age Distribution', font='Arial', size=16, weight='bold')

plt.show()
```



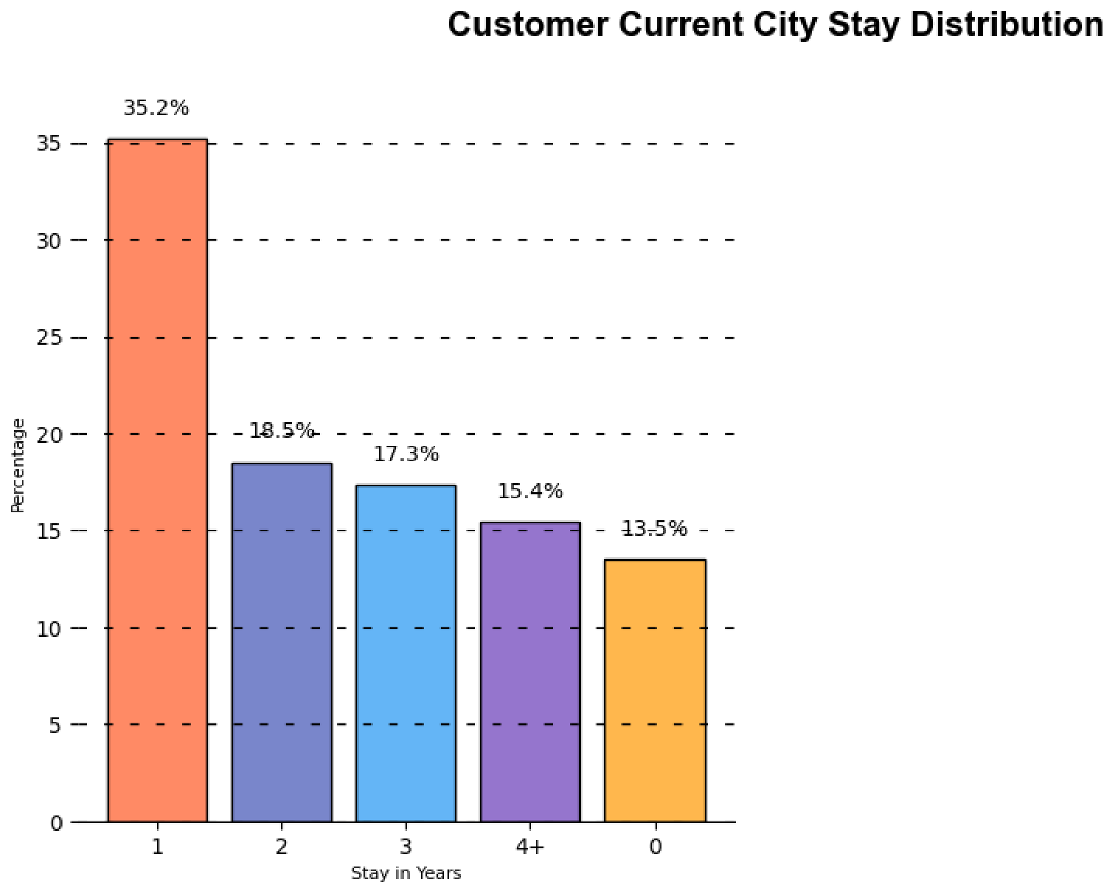

## 7 Insights

- The age group spanning from 26 to 35 years old represents the majority of Walmart's Black Friday sales, comprising 40% of the total sales. This underscores the notion that young and middle-aged adults exhibit the highest level of engagement and interest in seeking out deals and discounts.

- Following closely behind, the age groups of 36-45 and 18-25 constitute the second and third largest segments, respectively, accounting for 20% and 18% of the total sales. This suggests that Walmart caters to a diverse customer base, spanning various life stages and preferences.
- Conversely, the age groups of 46-50, 51-55, 55+, and 0-17 are smaller segments, each contributing less than 10% of the total sales. This indicates potential areas for improvement in Walmart's marketing strategies and product offerings to better attract customers from these age demographics, particularly seniors and children.
- **2.3 Customer stay in current City (in years)**

[20]:

```
plt.figure(figsize=(12,6))

plt.subplot(1, 2, 1)

current_stay_counts = df['Stay_In_Current_City_Years'].
 ↪value_counts(normalize=True) * 100
color_map = ["#FF8A65","#7986CB" , "#64B5F6", "#9575CD", "#FFB74D", "#4CAF50"]
plt.bar(x=current_stay_counts.index, height=current_stay_counts.values,␣
 ↪color=color_map, edgecolor='black')

for i in range(len(current_stay_counts)):
    plt.text(x=i, y=current_stay_counts[i] + 1, s=f'{current_stay_counts[i]:.
 ↪1f}%', ha='center', va='bottom')
# Adding grid lines
plt.grid(color='black', linestyle='--', axis='y', zorder=0, dashes=(5, 10))
# Removing axis lines
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
# Adding axis labels
plt.ylabel('Percentage', fontsize=8)
plt.xlabel('Stay in Years',fontsize=8)
plt.xticks()

# Setting title for visual
plt.suptitle('Customer Current City Stay Distribution', font='Arial', size=16,␣
 ↪weight='bold')

plt.show()
```



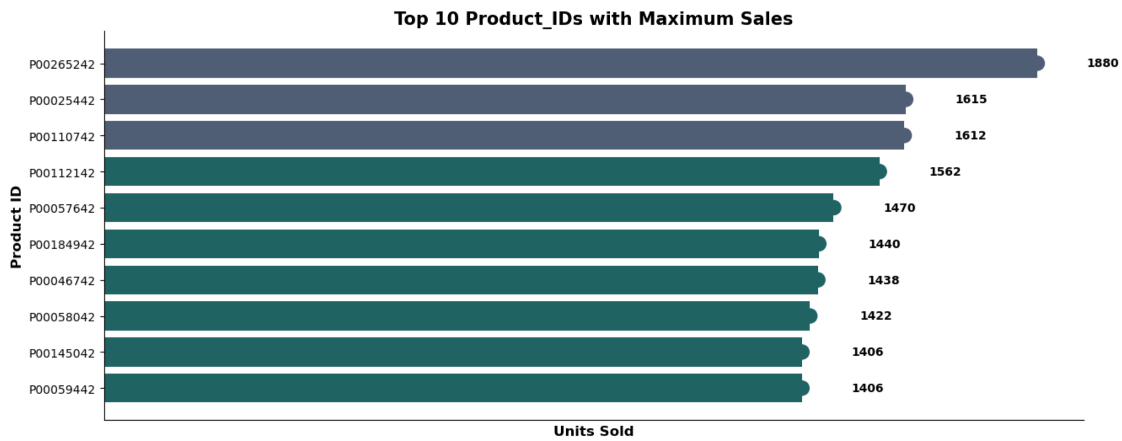

# 8 Insights

- The data indicates a notable proportion of customers who are either new to the city or frequently relocate, suggesting potential differences in preferences and needs compared to long-term residents.
- A significant majority of customers (49%) have resided in the current city for one year or less, indicating Walmart's strong appeal to newcomers seeking affordable and convenient shopping options.
- Customers in the 4+ years category (14%) signify Walmart's loyal customer base consisting of long-time city residents.
- The percentage of customers gradually decreases with longer durations of stay in the current city, implying that Walmart could enhance its efforts in targeting long-term residents through loyalty programs and promotions.

**2.4 Top 10 products with maximum sales**

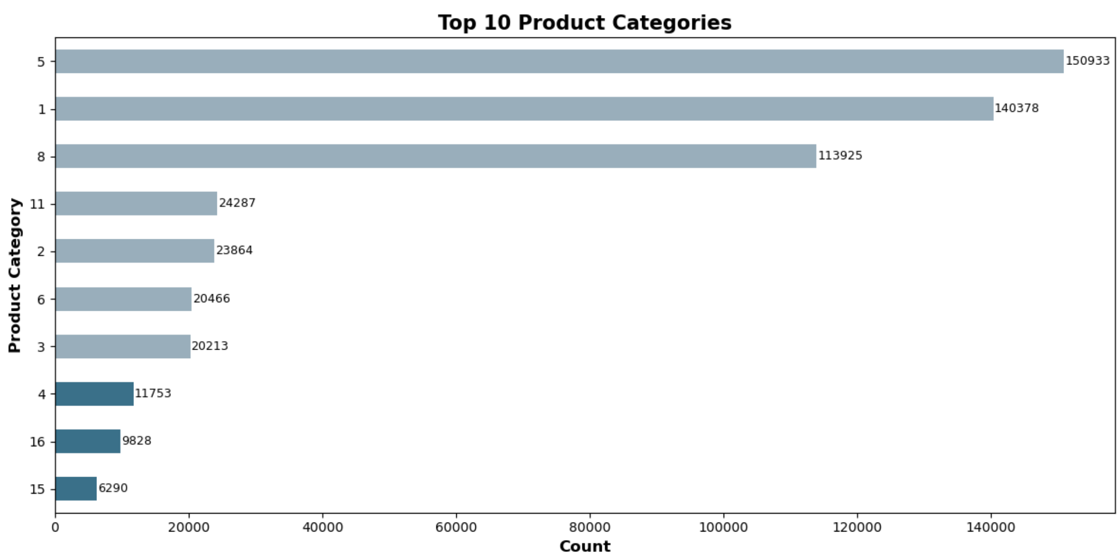
[21]:

```
# Create a new figure and grid spec
fig, ax = plt.subplots(figsize=(15, 6))

# Retrieve top 10 Product_IDs with maximum sales
top_10_products = df['Product_ID'].value_counts()[0:10]
top_10_products=top_10_products.iloc[-1:-11:-1]

color_map =  ['#1F6363' for i in range(7)] + ["#4F5D75" for i in range(3)]
# Plot horizontal bar chart
ax.barh(y=top_10_products.index, width=top_10_products.values, color=color_map)
ax.scatter(y = top_10_products.index, x =top_10_products.values, s = 150 ,␣
↪color = color_map )

# Add labels to each bar
for i, (product_id, sales) in enumerate(top_10_products.items()):
    ax.text(sales + 100, i, f'{sales}', va='center', fontsize=10,␣
↪fontweight='bold')

# Remove x-axis ticks
ax.set_xticks([])

# Set axis labels and title
ax.set_xlabel('Units Sold', fontsize=12, fontweight='bold')
ax.set_ylabel('Product ID', fontsize=12, fontweight='bold')
ax.set_title('Top 10 Product_IDs with Maximum Sales', fontsize=15,␣
↪fontweight='bold')

# Hide spines
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

plt.show()
```



### 2.5 Top 10 product categories

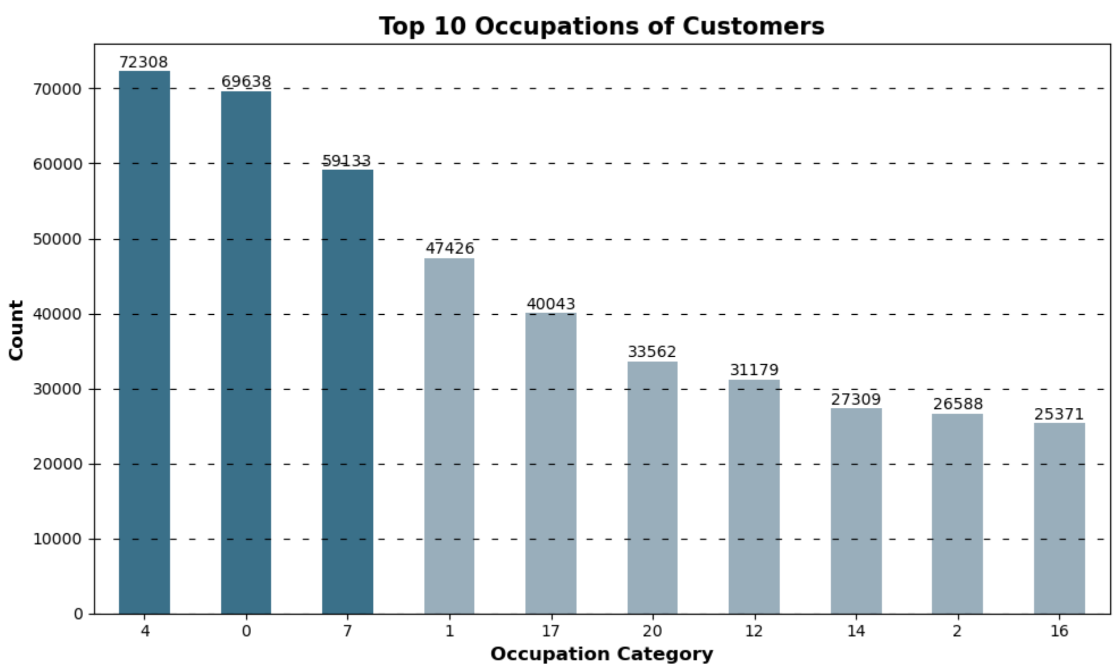
[22]:
```
top_10_product_category = df['Product_Category'].value_counts().nlargest(10)
top_10_product_category=top_10_product_category.iloc[-1:-11:-1]

# Create the horizontal bar plot
plt.figure(figsize=(12, 6))
color_map = ["#3A7089" for i in range(3)] + ["#99AEBB" for i in range(7)]
top_10_product_category.plot(kind='barh', color=color_map)

# Add value counts next to each bar
for i, v in enumerate(top_10_product_category):
    plt.text(v + 100, i, str(v), ha='left', va='center', fontsize=9)


# Add labels and title
plt.xlabel('Count', fontsize=12, fontweight='bold')
plt.ylabel('Product Category', fontsize=12, fontweight='bold')
plt.title('Top 10 Product Categories', fontsize=15, fontweight='bold')

# Show plot
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```




### 2.6 Top 10 Customer's Occupation

[23]:

```
top_10_occupations = df['Occupation'].value_counts().nlargest(10)

# Create the bar plot
plt.figure(figsize=(10, 6))
color_map =  ["#3A7089" for i in range(3)] +  ['#99AEBB' for i in range(7)]
top_10_occupations.plot(kind='bar', color=color_map)

for i, v in enumerate(top_10_occupations):
    plt.text(i, v + 100, str(v), ha='center', va='bottom', fontsize=10)

plt.grid(color = 'black',linestyle = '--',axis = 'y',zorder = 0,dashes = (5,10))

# Add labels and title
plt.xlabel('Occupation Category', fontsize=12, fontweight='bold')
plt.ylabel('Count', fontsize=12, fontweight='bold')
plt.title('Top 10 Occupations of Customers', fontsize=15, fontweight='bold')

# Show plot
plt.xticks(rotation=0)  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```



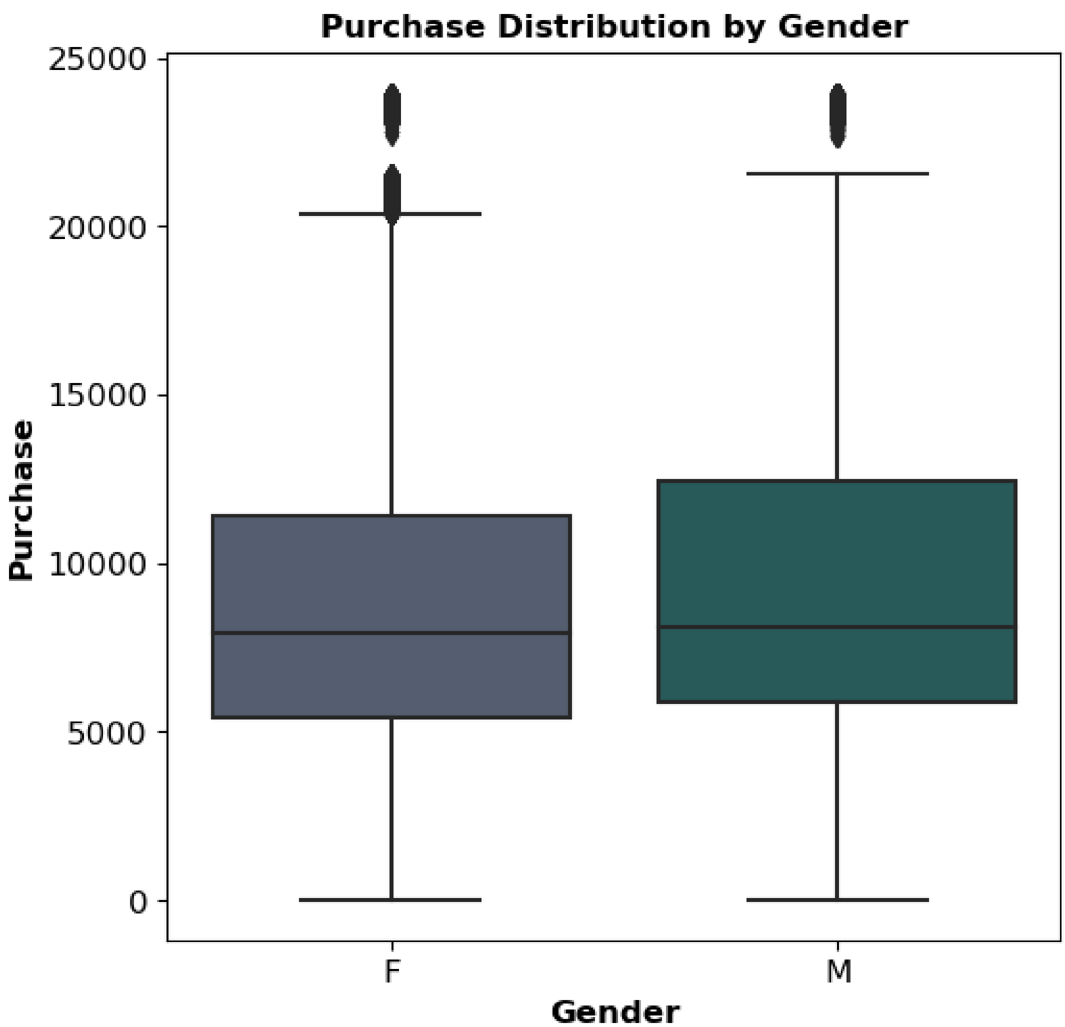
# 9 Insights

1. **Top 10 Products Sold** - The most popular products during Walmart's Black Friday sales exhibit relatively consistent sales figures, indicating a diverse range of products favored by a broad customer base.
2. **Top 10 Product Categories** - Categories 5, 1, and 8 demonstrate exceptional sales performance, collectively constituting approximately 75% of total sales. This underscores a pronounced preference among customers for products within these categories.
3. **Occupation Categories** - Customers within occupation categories 4, 0, and 7 significantly contributed to approximately 37% of total purchases. This suggests either a strong demand for Walmart's offerings among individuals in these occupational groups or a higher disposable income, particularly evident during Black Friday shopping.

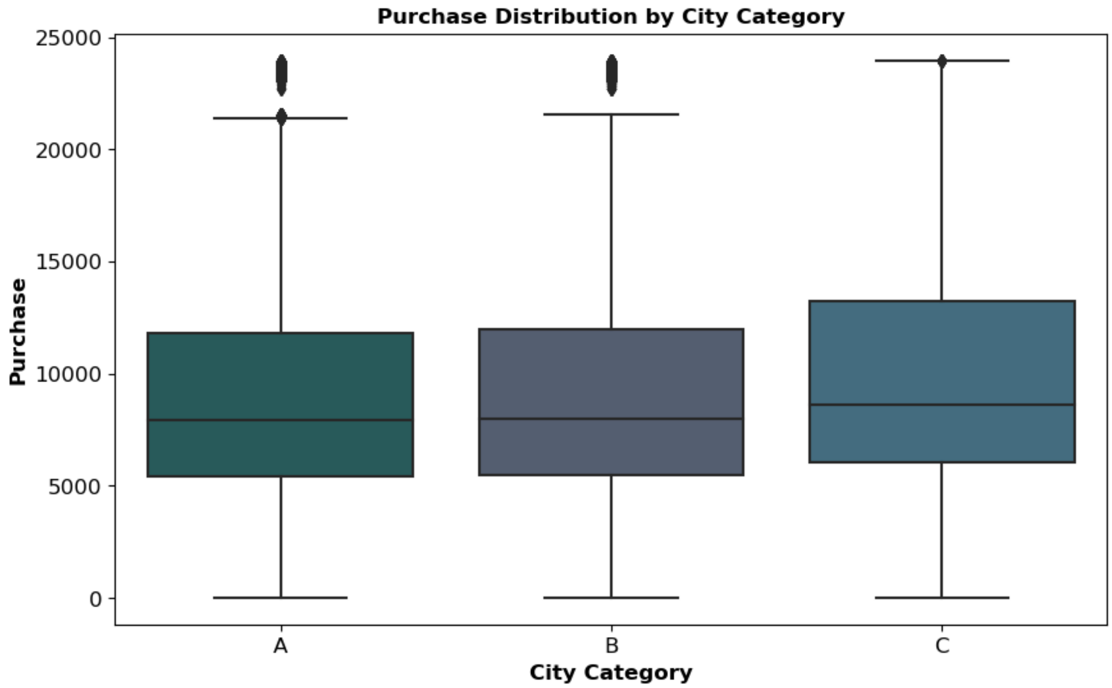
## 9.1 Bivariate Analysis

[24]:
```
## Purchase distribution by Gender
```

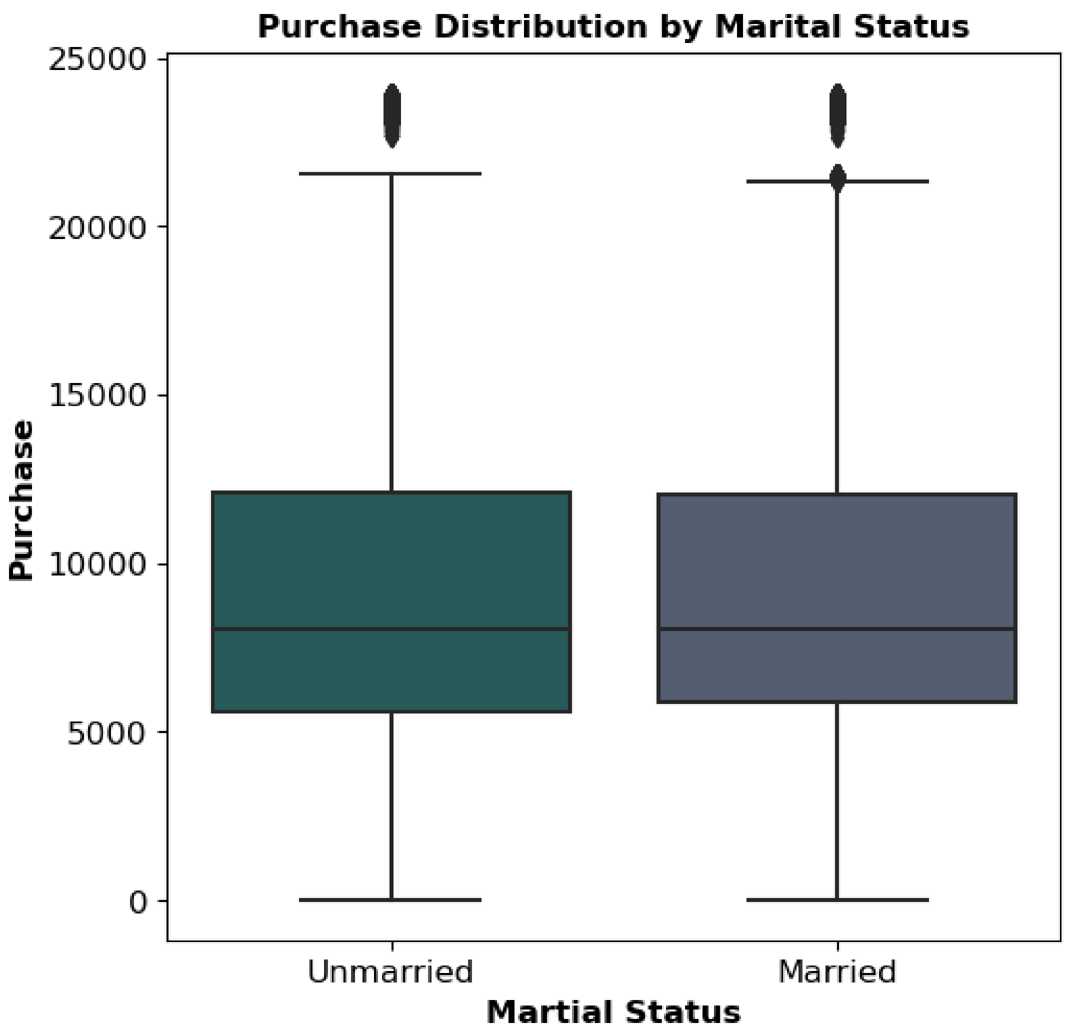
[25]:
```
colors = {"M": "#1F6363", "F": "#4F5D75"}

plt.figure(figsize=(6,6))
sns.boxplot(x='Gender', y='Purchase', data=df, palette=colors)

plt.title('Purchase Distribution by Gender', fontsize=12, fontweight='bold')
plt.xlabel('Gender', fontsize=12, fontweight='bold')
plt.ylabel('Purchase', fontsize=12, fontweight='bold')
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

plt.show()
```




[26]:
```
#Purchase distribution by City Category
colors = {"A": "#1F6363", "B": "#4F5D75","C":"#3A7089"}

plt.figure(figsize=(10,6))
sns.boxplot(x='City_Category', y='Purchase', data=df, palette=colors)

plt.title('Purchase Distribution by City Category', fontsize=12,␣
↪fontweight='bold')
plt.xlabel('City Category', fontsize=12, fontweight='bold')
plt.ylabel('Purchase', fontsize=12, fontweight='bold')
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

plt.show()
```

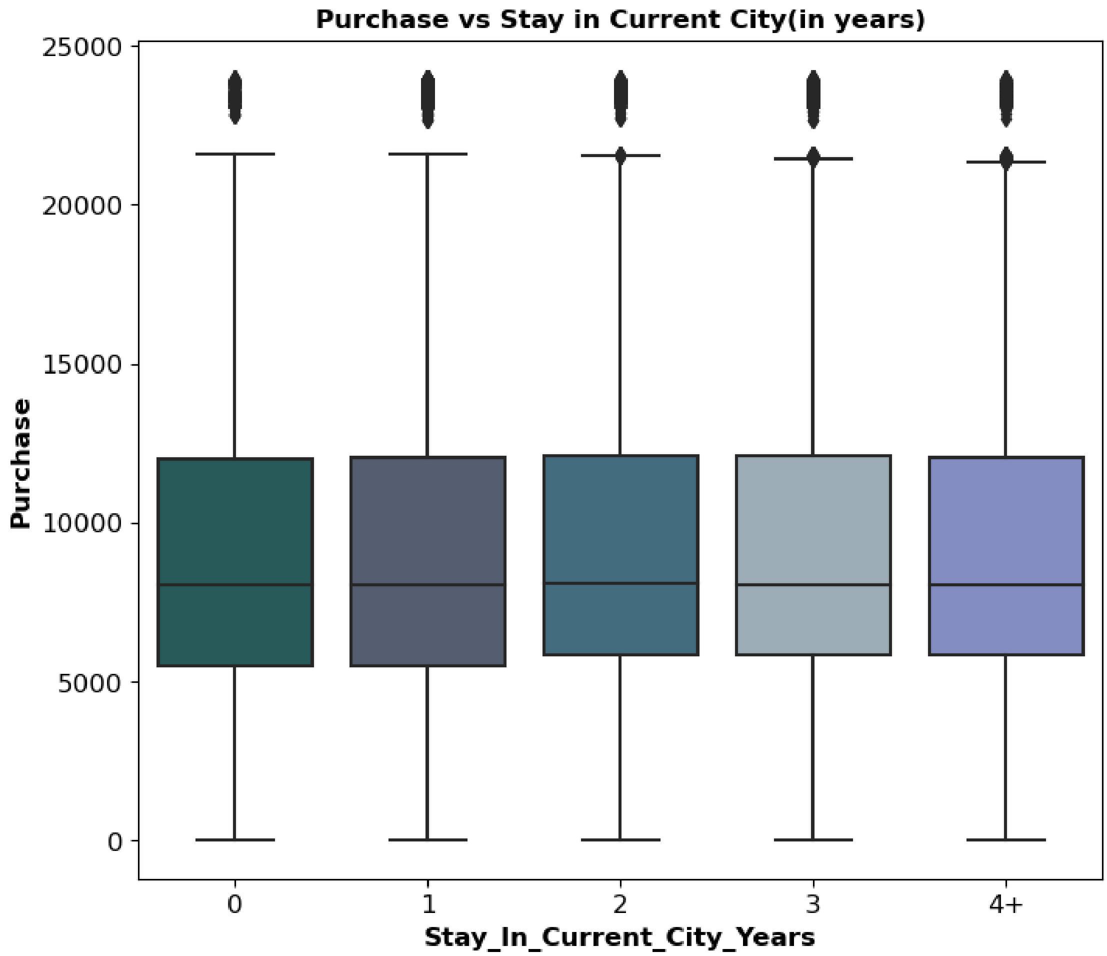



[27]:
```
#Purchase distribution by Marital Status
colors = {"Unmarried": "#1F6363", "Married": "#4F5D75"}

plt.figure(figsize=(6,6))
sns.boxplot(x='Marital_Status', y='Purchase', data=df, palette=colors)

plt.title('Purchase Distribution by Marital Status', fontsize=12,␣
↪fontweight='bold')
plt.xlabel('Martial Status', fontsize=12, fontweight='bold')
plt.ylabel('Purchase', fontsize=12, fontweight='bold')
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

plt.show()
```

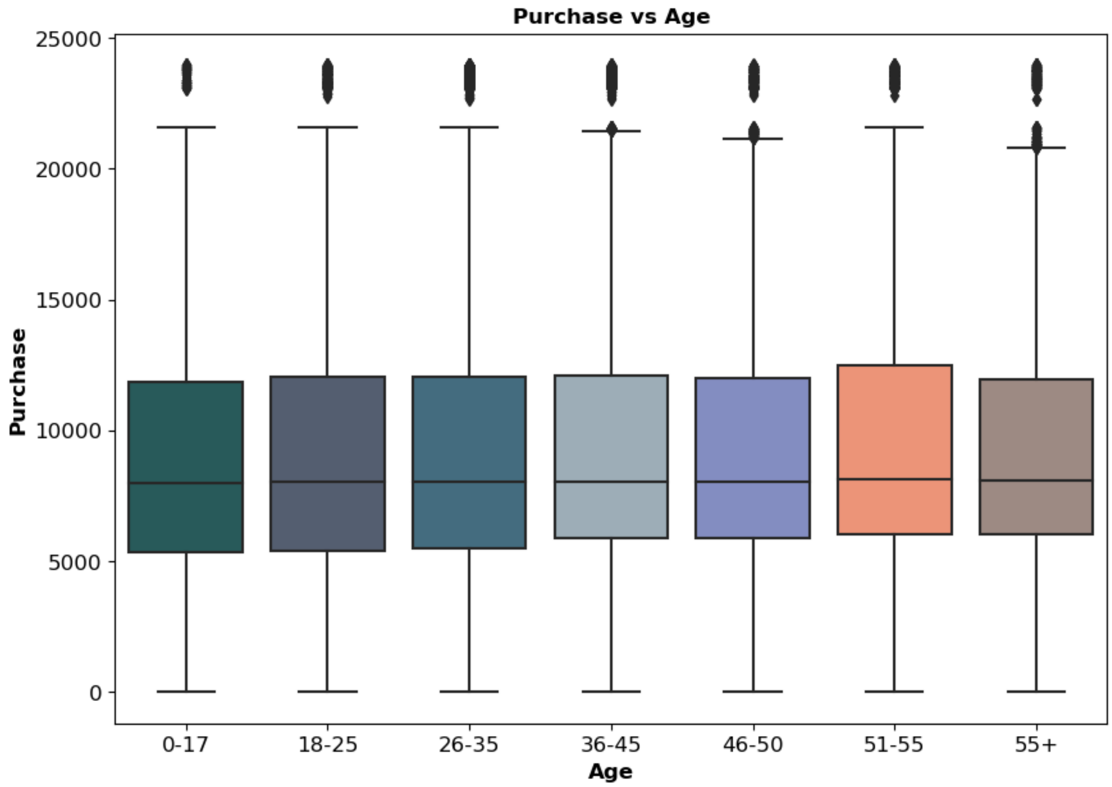



[28]:
```
#Purchase distribution by Stay in Current City
colors = {"0":"#1F6363","1":"#4F5D75","2":"#3A7089","3":"#99AEBB","4+":
 ↪"#7986CB"}

plt.figure(figsize=(8,7))
sns.boxplot(x='Stay_In_Current_City_Years', y='Purchase', data=df,␣
 ↪palette=colors)

plt.title('Purchase vs Stay in Current City(in years)', fontsize=12,␣
 ↪fontweight='bold')
plt.xlabel('Stay_In_Current_City_Years', fontsize=12, fontweight='bold')
plt.ylabel('Purchase', fontsize=12, fontweight='bold')
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
```

```
plt.show()
```




[29]:
```
#Purchase distribution by Age

colors = {"0-17":"#1F6363","18-25":"#4F5D75","26-35":"#3A7089","36-45":
 ↪"#99AEBB","46-50":"#7986CB","51-55":"#FF8A65","55+":"#A1887F"}

plt.figure(figsize=(10,7))
sns.boxplot(x='Age', y='Purchase', data=df, palette=colors)

plt.title('Purchase vs Age', fontsize=12, fontweight='bold')
plt.xlabel('Age', fontsize=12, fontweight='bold')
plt.ylabel('Purchase', fontsize=12, fontweight='bold')
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

plt.show()
```




# 10 Insights

- Across all the variables examined, it is notable that the purchase amount remains relatively consistent. Regardless of the variable being considered, the median purchase amount remains around $8,000 USD. This indicates a stable purchasing pattern across different categories in the dataset.

## 10.1 Gender wise spending per Transaction

[30]:
```
#creating a df(A_G) for purchase amount vs gender
A_G=df.groupby('Gender')['Purchase'].agg(['sum','count']).reset_index()

A_G['amount_in_billions'] = round(A_G['sum']/ 10**9,2)

A_G['sum_percentage']=round(A_G['sum']/A_G['sum'].sum(),2)

A_G['per_purchase']=round(A_G['sum']/A_G['count'])

A_G['Gender']=A_G['Gender'].replace({'F':'Female','M':'Male'})

A_G
```

[30]:

```
   Gender         sum   count  amount_in_billions  sum_percentage  \
0  Female  1186232642  135809                1.19            0.23
1    Male  3909580100  414259                3.91            0.77

   per_purchase
0        8735.0
1        9438.0
```

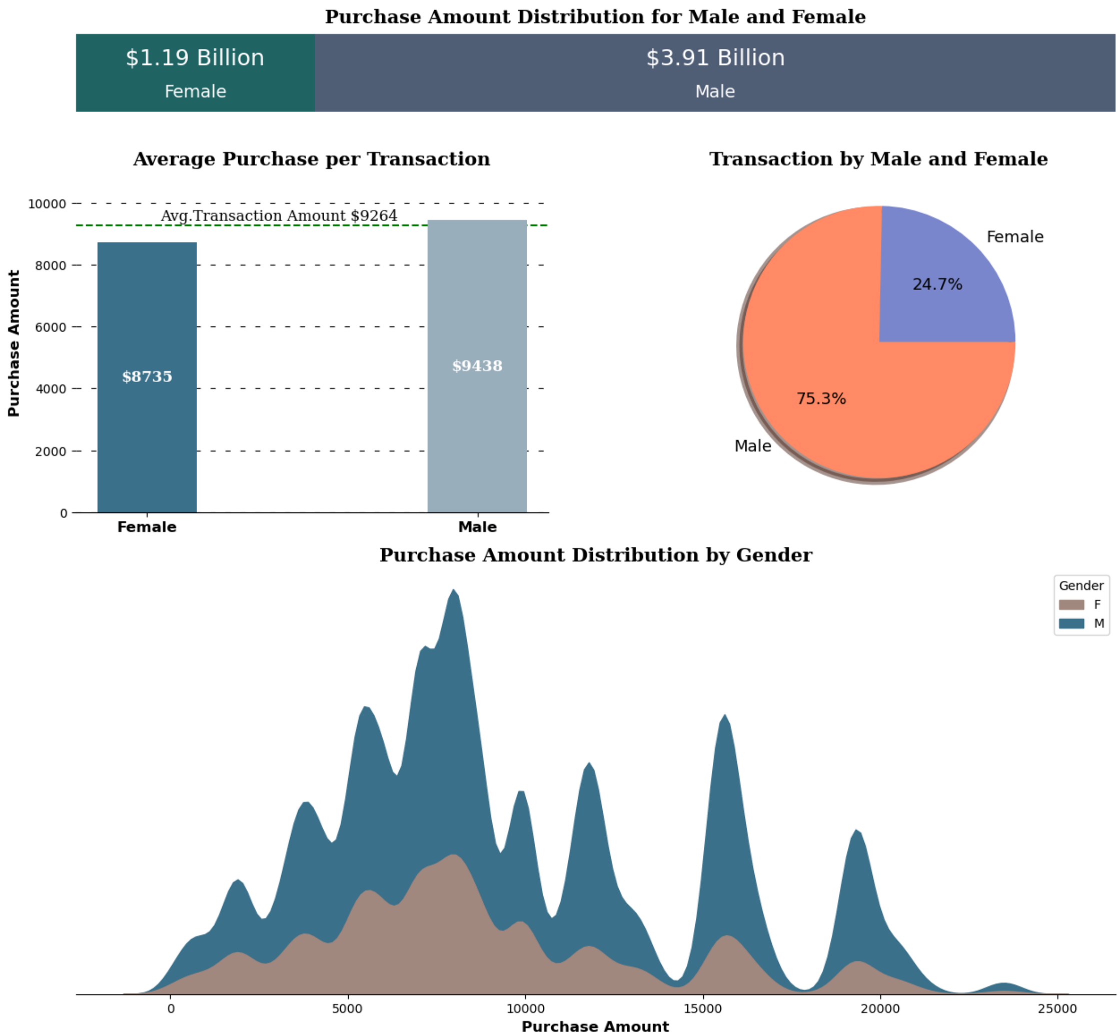
[31]:

```
#setting the plot style
figure=plt.figure(figsize=(15,14))
grid_spec=figure.add_gridspec(3,2,height_ratios=[0.10,0.4,0.5])
                                                        #Distribution of Purchase Amount
plt1=figure.add_subplot(grid_spec[0,:])
plt1.barh(A_G.loc[0,'Gender'],width=A_G.
 ↪loc[0,'sum_percentage'],color="#1F6363",label='Female')
plt1.barh(A_G.loc[0,'Gender'],width=A_G.loc[1,'sum_percentage'],left =A_G.
 ↪loc[0,'sum_percentage'],color="#4F5D75",label='Male')
text=[0.0]
for i in A_G.index:
    plt1.text(A_G.loc[i,'sum_percentage']/2+text[0],0.15,f"${A_G.
 ↪loc[i,'amount_in_billions']} Billion",
              va='center',ha='center',fontsize=18,color='white')

    plt1.text(A_G.loc[i,'sum_percentage']/2+text[0],-0.20,f"{A_G.
 ↪loc[i,'Gender']}",
              va='center',ha='center',fontsize=14,color='white')

    text+=A_G.loc[i,'sum_percentage']


for s in ['top','left','right','bottom']:
    plt1.spines[s].set_visible(False)

plt1.set_xticks([])
plt1.set_yticks([])
plt1.set_xlim(0,1)

plt1.set_title('Purchase Amount Distribution for Male and Female',{'font':
 ↪'serif','size':15,'weight':'bold'})

plt2=figure.add_subplot(grid_spec[1,0])
color_map = ["#3A7089","#99AEBB"]

plt2.bar(A_G['Gender'],A_G['per_purchase'],color=color_map,zorder=2,width=0.3)

avg=round(df['Purchase'].mean())
plt2.axhline(y=avg,color='green',zorder=0,linestyle='--')
```

```
plt2.text(0.4,avg+300,f"Avg.Transaction Amount ${avg:.0f}",
          {'font':'serif','size':12},ha ='center',va ='center')

plt2.set_ylim(0,11000)

for i in A_G.index:
    plt2.text(A_G.loc[i,'Gender'],A_G.loc[i,'per_purchase']/2,f"${A_G.loc[i,'per_purchase']:.0f}",
              {'font':'serif','size':12,'color':'white','weight':'bold' },ha ='center',va ='center')


plt2.grid(color='black',linestyle='--',axis='y',zorder=0,dashes=(5,10))

for j in ['top','left','right']:
    plt2.spines[j].set_visible(False)

plt2.set_ylabel('Purchase Amount',fontweight='bold',fontsize=12)
plt2.set_xticklabels(A_G['Gender'],fontweight='bold',fontsize=12)

plt2.set_title('Average Purchase per Transaction',{'font':'serif','size':15,'weight':'bold'})

plt3=figure.add_subplot(grid_spec[1,1])
color_map=["#7986CB","#FF8A65"]
plt3.pie(A_G['count'],labels=A_G['Gender'],autopct='%.1f%%',
        shadow=True,colors=color_map,wedgeprops={'linewidth':5},textprops={'fontsize': 13,'color':'black'})

plt3.set_title('Transaction by Male and Female',{'font':'serif','size':15,'weight':'bold'})

plt4=figure.add_subplot(grid_spec[2,:])
color_map=["#A1887F","#3A7089"]
sns.kdeplot(data=df,x='Purchase',hue='Gender',palette=color_map,fill=True,alpha=1,ax=plt4)

for k in ['top','left','right']:
    plt4.spines[k].set_visible(False)

plt4.set_yticks([])
plt4.set_ylabel('')
plt4.set_xlabel('Purchase Amount',fontweight='bold',fontsize=12)
```

```
plt4.set_title('Purchase Amount Distribution by Gender',{'font':'serif', 'size':
 ↪15,'weight':'bold'})
plt.show()
```

```
C:\Users\shobh\AppData\Local\Temp\ipykernel_39048\2273816679.py:52: UserWarning:
FixedFormatter should only be used together with FixedLocator
  plt2.set_xticklabels(A_G['Gender'],fontweight='bold',fontsize=12)
```




## 11 Insights

1. In terms of overall sales and transaction volume, male customers significantly outpaced thei

2. Male customers exhibited a slightly higher average transaction value compared to female cust

3. It's worth noting that the distribution of purchase amounts for both genders deviates from t

## 12 Confidence Interval

1. **Step 1 - Constructing the CLT Curve:**
   - Given the non-normal distribution of purchase amounts, we turn to the Central Limit Theorem (CLT) for analysis. This theorem assures that, regardless of the original population distribution, the distribution of sample means will tend towards a normal distribution.
2. **Step 2 - Establishing Confidence Intervals:**
   - Following the construction of the CLT curve, our next objective is to establish confidence intervals to predict population means. These intervals will be computed at confidence levels of 99%, 95%, and 90%.

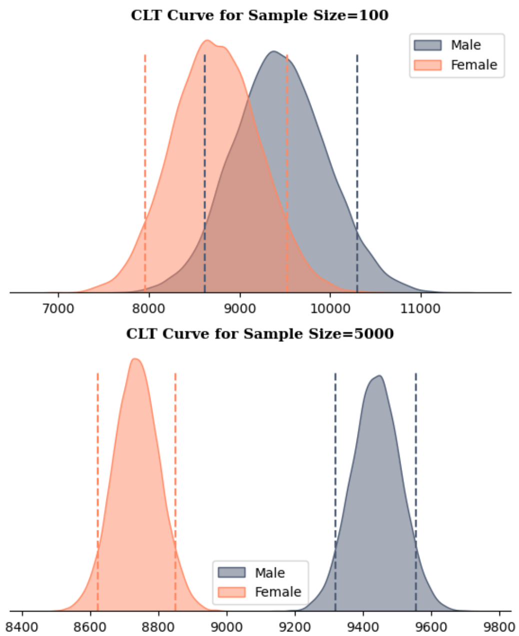
We will use varying sample sizes of 100, 1000, 5000, and 50000.

[32]:
```
def confidence_interval(data,confidence_level):

    lower_ci=(100-confidence_level)/2
    upper_ci=(100+confidence_level)/2


    conf_interval=np.percentile(data,[lower_ci,upper_ci]).round(0)

    return conf_interval
```

[33]:
```
#90 Percent Confidence Level
confidence_level=90

figure=plt.figure(figsize=(15,8))
grid_spec=figure.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=[(100,0,0),(1000,0,1),(5000,1,0),(50000,1,1)]

bootstrap_samples=20000
male_sample_90={}
female_sample_90={}

for i,x,y in sample_sizes:
    male_means=[]
    female_means=[]
    for j in range(bootstrap_samples):
        male_bootstrapped_samples=np.random.choice(df_male,size=i)
        female_bootstrapped_samples=np.random.choice(df_female,size=i)

        male_sample_mean=np.mean(male_bootstrapped_samples)
```

```
        female_sample_mean=np.mean(female_bootstrapped_samples)

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

    male_means,female_means

    male_sample_90[f'{confidence_level}%_{i}']=male_means
    female_sample_90[f'{confidence_level}%_{i}']=female_means

    df1=pd.DataFrame(data={'male_means':male_means,'female_means':female_means})

    plt5=figure.add_subplot(grid_spec[x,y])


    sns.kdeplot(data=df1,x='male_means',color="#4F5D75",fill=True,alpha=0.
 ↪5,ax=plt5,label='Male')
    sns.kdeplot(data=df1,x='female_means',color="#FF8A65",fill=True,alpha=0.
 ↪5,ax=plt5,label='Female')

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

    for k in m_range:
        plt5.axvline(x=k,ymax=0.9,color="#4F5D75",linestyle='--')
    for k in f_range:
        plt5.axvline(x=k,ymax=0.9,color="#FF8A65",linestyle='--')

    for l in ['top','left','right']:
        plt5.spines[l].set_visible(False)

    plt5.set_yticks([])
    plt5.set_ylabel('')
    plt5.set_xlabel('')

    plt5.set_title(f'CLT Curve for Sample Size={i}',{'font':'serif','size':
 ↪11,'weight':'bold'})
    plt.legend()


figure.suptitle(f'{confidence_level}% Confidence␣
 ↪Interval',font='serif',size=18,weight='bold')

male_sample_90,female_sample_90

plt.show()
```

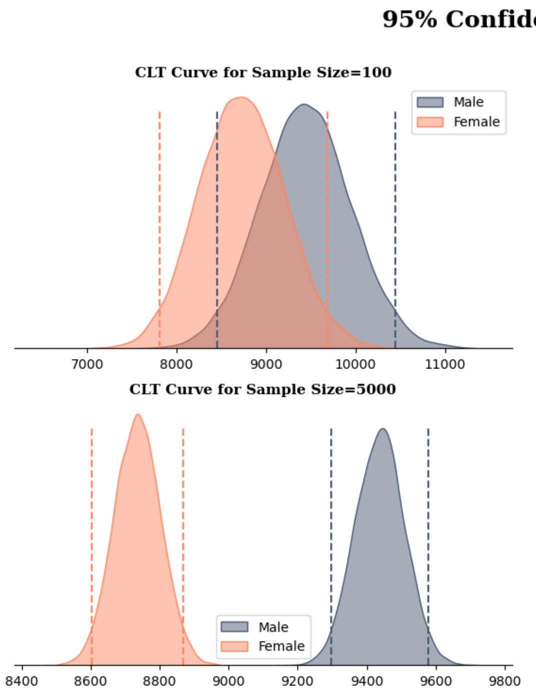
**90% Confidence Interval**




[34]:

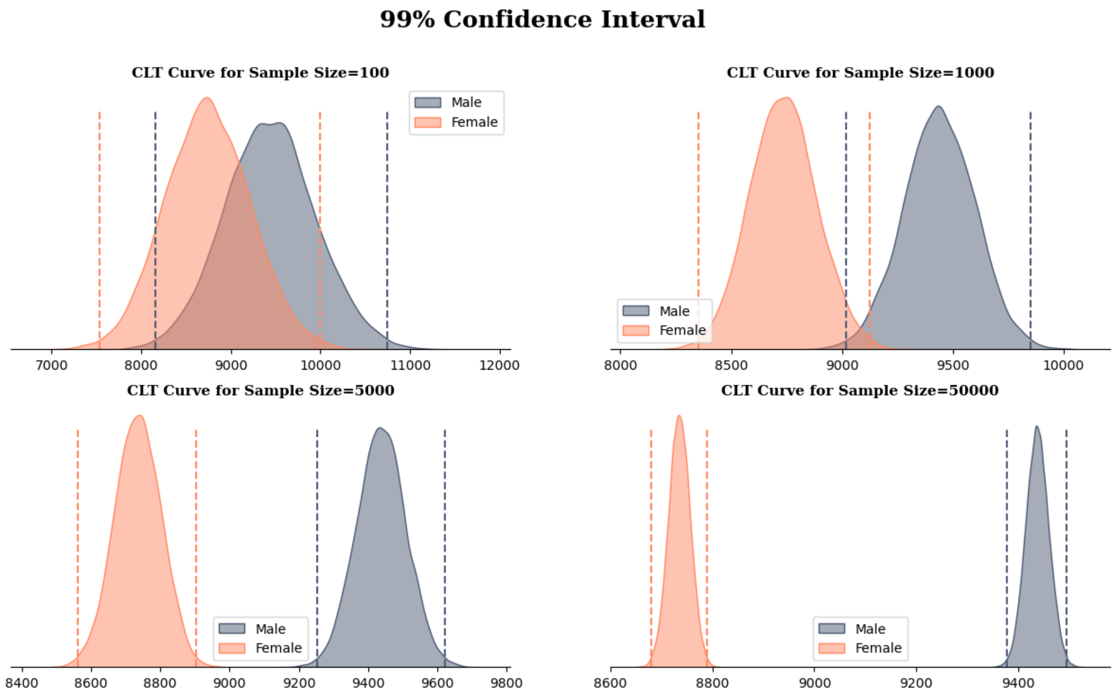
```
#95 Percent Confidence Level
confidence_level=95

figure=plt.figure(figsize=(15,8))
grid_spec=figure.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=[(100,0,0),(1000,0,1),(5000,1,0),(50000,1,1)]

bootstrap_samples=20000
male_sample_95={}
female_sample_95={}

for i,x,y in sample_sizes:
    male_means=[]
    female_means=[]
    for j in range(bootstrap_samples):
        male_bootstrapped_samples=np.random.choice(df_male,size=i)
        female_bootstrapped_samples=np.random.choice(df_female,size=i)

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

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

    male_means,female_means

    male_sample_95[f'{confidence_level}%_{i}']=male_means
    female_sample_95[f'{confidence_level}%_{i}']=female_means

    df1=pd.DataFrame(data={'male_means':male_means,'female_means':female_means})

    plt5=figure.add_subplot(grid_spec[x,y])


    sns.kdeplot(data=df1,x='male_means',color="#4F5D75",fill=True,alpha=0.
 ↪5,ax=plt5,label='Male')
    sns.kdeplot(data=df1,x='female_means',color="#FF8A65",fill=True,alpha=0.
 ↪5,ax=plt5,label='Female')

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

    for k in m_range:
        plt5.axvline(x=k,ymax=0.9,color="#4F5D75",linestyle='--')
    for k in f_range:
        plt5.axvline(x=k,ymax=0.9,color="#FF8A65",linestyle='--')

    for l in ['top','left','right']:
        plt5.spines[l].set_visible(False)

    plt5.set_yticks([])
    plt5.set_ylabel('')
    plt5.set_xlabel('')

    plt5.set_title(f'CLT Curve for Sample Size={i}',{'font':'serif','size':
 ↪11,'weight':'bold'})
    plt.legend()


figure.suptitle(f'{confidence_level}% Confidence␣
 ↪Interval',font='serif',size=18,weight='bold')
male_sample_95,female_sample_95
plt.show()
```




[35]:
```
#99 Percent Confidence Level
confidence_level=99

figure=plt.figure(figsize=(15,8))
grid_spec=figure.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=[(100,0,0),(1000,0,1),(5000,1,0),(50000,1,1)]

bootstrap_samples=20000
male_sample_99={}
female_sample_99={}

for i,x,y in sample_sizes:
    male_means=[]
    female_means=[]
    for j in range(bootstrap_samples):
        male_bootstrapped_samples=np.random.choice(df_male,size=i)
        female_bootstrapped_samples=np.random.choice(df_female,size=i)

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

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

    male_means,female_means

    male_sample_99[f'{confidence_level}%_{i}']=male_means
    female_sample_99[f'{confidence_level}%_{i}']=female_means

    df1=pd.DataFrame(data={'male_means':male_means,'female_means':female_means})

    plt5=figure.add_subplot(grid_spec[x,y])


    sns.kdeplot(data=df1,x='male_means',color="#4F5D75",fill=True,alpha=0.
 ↪5,ax=plt5,label='Male')
    sns.kdeplot(data=df1,x='female_means',color="#FF8A65",fill=True,alpha=0.
 ↪5,ax=plt5,label='Female')

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

    for k in m_range:
        plt5.axvline(x=k,ymax=0.9,color="#4F5D75",linestyle='--')
    for k in f_range:
        plt5.axvline(x=k,ymax=0.9,color="#FF8A65",linestyle='--')

    for l in ['top','left','right']:
        plt5.spines[l].set_visible(False)

    plt5.set_yticks([])
    plt5.set_ylabel('')
    plt5.set_xlabel('')

    plt5.set_title(f'CLT Curve for Sample Size={i}',{'font':'serif','size':
 ↪11,'weight':'bold'})
    plt.legend()


figure.suptitle(f'{confidence_level}% Confidence␣
 ↪Interval',font='serif',size=18,weight='bold')
male_sample_99,female_sample_99
plt.show()
```




# 13 Are confidence intervals of average male and female customer spending overlapping?

[36]:
```
figure=plt.figure(figsize=(20,10))
grid_specs=figure.add_gridspec(3,1)

for i,j,k,l in␣
↪[(male_sample_90,female_sample_90,90,0),(male_sample_95,female_sample_95,95,1),(male_sample
↪
    male_ci=['Male']
    female_ci=['Female']

    for m in i:
        male_range=confidence_interval(i[m],k)
        male_ci.append(f"CI=${male_range[0]:.0f}-${male_range[1]:.0f},␣
↪Range={(male_range[1]-male_range[0]):.0f}")


    for f in j:
        female_range=confidence_interval(j[f],k)
        female_ci.append(f"CI=${female_range[0]:.0f}-${female_range[1]:.0f},␣
↪Range={(female_range[1]-female_range[0]):.0f}")
```

```
    plot=figure.add_subplot(grid_specs[l])


    conf_int_info=[male_ci,female_ci]


    table=plot.table(cellText=conf_int_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
    plot.axis('off')

    #setting title
    plot.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=8617 – 10295, Range=1678 | CI=9170 – 9706, Range=536 | CI=9319 – 9556, Range=237 | CI=9400 – 9475, Range=75 |
| Female | CI=7956 – 9530, Range=1574 | CI=8489 – 8988, Range=499 | CI=8624 – 8849, Range=225 | CI=8699 – 8770, Range=71 |

**95% Confidence Interval Summary**

| Gender | Sample Size=100 | Sample Size=1000 | Sample Size=5000 | Sample Size=50000 |
|---|---|---|---|---|
| Male | CI=8453 – 10444, Range=1991 | CI=9123 – 9752, Range=629 | CI=9298 – 9578, Range=280 | CI=9393 – 9483, Range=90 |
| Female | CI=7814 – 9688, Range=1874 | CI=8443 – 9036, Range=593 | CI=8602 – 8868, Range=266 | CI=8692 – 8777, Range=85 |

**99% Confidence Interval Summary**

| Gender | Sample Size=100 | Sample Size=1000 | Sample Size=5000 | Sample Size=50000 |
|---|---|---|---|---|
| Male | CI=8162 – 10749, Range=2587 | CI=9017 – 9851, Range=834 | CI=9252 – 9623, Range=371 | CI=9378 – 9496, Range=118 |
| Female | CI=7533 – 9995, Range=2462 | CI=8351 – 9123, Range=772 | CI=8561 – 8903, Range=342 | CI=8679 – 8789, Range=110 |

## 14 Insights

- The analysis underscores the pivotal role of sample size in estimating population parameters. It suggests that with larger sample sizes, confidence intervals tend to become narrower and more precise. In business contexts, this implies that utilizing larger samples can yield more dependable insights and estimations.
- As observed in the analysis, barring the sample size of 100, the confidence intervals exhibit non-overlapping ranges as sample sizes increase. This indicates a statistically significant dis-

parity in average transaction spending between men and women within the provided samples.

- With 95% confidence, it is estimated that the true population average for males falls within the range of $9,393-$9,482, while for females, it lies between $8,692-$8,777.
- The findings suggest that men tend to spend more per transaction on average compared to women. This is evident as the upper bounds of confidence intervals consistently surpass those of women across various sample sizes.

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

**Targeted Segmentation Strategies:**

- Walmart has the opportunity to develop tailored marketing initiatives, loyalty programs, or product offerings tailored to the unique spending patterns of male and female customers. Such targeted approaches can potentially optimize revenue generation from each customer segment.

**Pricing Optimization Tactics:**

- Leveraging insights from the average spending per transaction data across genders, Walmart can refine its pricing and discount strategies. This may involve adjusting prices or offering discounts to encourage higher spending among male customers, while ensuring competitive pricing remains for products targeted towards female customers.

## 15.1 Martial Status vs Purchase Amount

[37]:
```
#creating a df(A_G) for purchase amount vs gender
A_M=df.groupby('Marital_Status')['Purchase'].agg(['sum','count']).reset_index()

A_M['amount_in_billions']=round(A_M['sum']/ 10**9,2)

A_M['sum_percentage']=round(A_M['sum']/A_M['sum'].sum(),3)

A_M['per_purchase']=round(A_M['sum']/A_M['count'])

A_M
```

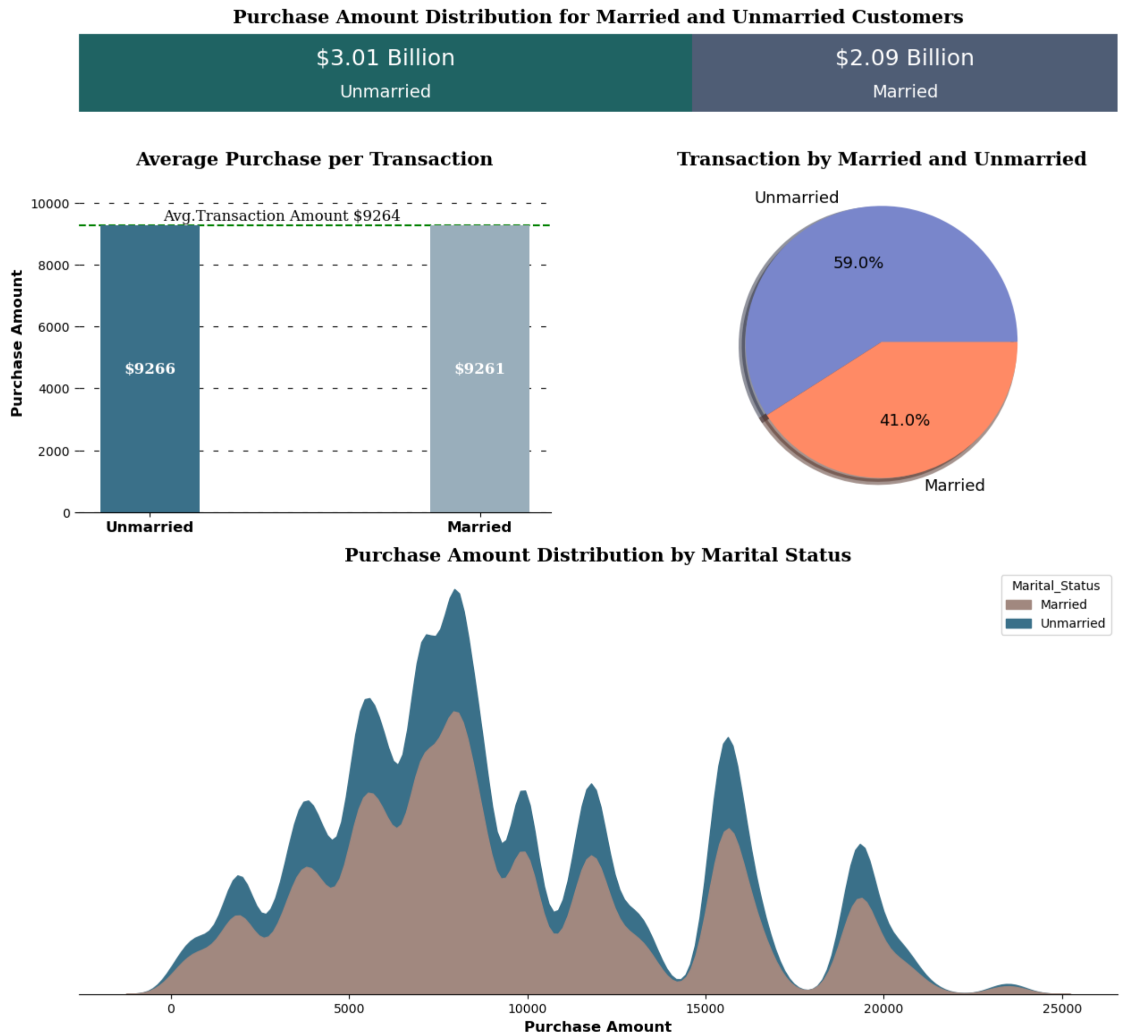
[37]:
```
  Marital_Status         sum   count  amount_in_billions  sum_percentage  \
0      Unmarried  3008927447  324731                3.01            0.59
1        Married  2086885295  225337                2.09            0.41

   per_purchase
0        9266.0
1        9261.0
```

[38]:
```
#setting the plot style
figure=plt.figure(figsize=(15,14))
```

```
grid_spec=figure.add_gridspec(3,2,height_ratios=[0.10,0.4,0.5])
                                                        #Distribution of Purchase Amount
plt1=figure.add_subplot(grid_spec[0,:])
plt1.barh(A_M.loc[0,'Marital_Status'],width=A_M.
 ↪loc[0,'sum_percentage'],color="#1F6363",label='Unmarried')
plt1.barh(A_M.loc[0,'Marital_Status'],width=A_M.loc[1,'sum_percentage'],left␣
 ↪=A_M.loc[0,'sum_percentage'],color="#4F5D75",label='Married')
text=[0.0]
for i in A_M.index:
    plt1.text(A_M.loc[i,'sum_percentage']/2+text[0],0.15,f"${A_M.
 ↪loc[i,'amount_in_billions']} Billion",
              va='center',ha='center',fontsize=18,color='white')

    plt1.text(A_M.loc[i,'sum_percentage']/2+text[0],-0.20,f"{A_M.
 ↪loc[i,'Marital_Status']}",
              va='center',ha='center',fontsize=14,color='white')

    text+=A_M.loc[i,'sum_percentage']


for s in ['top','left','right','bottom']:
    plt1.spines[s].set_visible(False)

plt1.set_xticks([])
plt1.set_yticks([])
plt1.set_xlim(0,1)

plt1.set_title('Purchase Amount Distribution for Married and Unmarried␣
 ↪Customers',{'font':'serif','size':15,'weight':'bold'})

plt2=figure.add_subplot(grid_spec[1,0])
color_map = ["#3A7089","#99AEBB"]

plt2.
 ↪bar(A_M['Marital_Status'],A_M['per_purchase'],color=color_map,zorder=2,width=0.
 ↪3)

avg=round(df['Purchase'].mean())
plt2.axhline(y=avg,color='green',zorder=0,linestyle='--')

plt2.text(0.4,avg+300,f"Avg.Transaction Amount ${avg:.0f}",
            {'font':'serif','size':12},ha ='center',va ='center')

plt2.set_ylim(0,11000)

for i in A_M.index:
```

```
    plt2.text(A_M.loc[i,'Marital_Status'],A_M.loc[i,'per_purchase']/2,f"${A_M.
↪loc[i,'per_purchase']:.0f}",
              {'font':'serif','size':12,'color':'white','weight':'bold' },ha␣
↪='center',va ='center')


plt2.grid(color='black',linestyle='--',axis='y',zorder=0,dashes=(5,10))

for j in ['top','left','right']:
    plt2.spines[j].set_visible(False)

plt2.set_ylabel('Purchase Amount',fontweight='bold',fontsize=12)
plt2.set_xticklabels(A_M['Marital_Status'],fontweight='bold',fontsize=12)

plt2.set_title('Average Purchase per Transaction',{'font':'serif','size':
↪15,'weight':'bold'})

plt3=figure.add_subplot(grid_spec[1,1])
color_map=["#7986CB","#FF8A65"]
plt3.pie(A_M['count'],labels=A_M['Marital_Status'],autopct='%.1f%%',
        shadow=True,colors=color_map,wedgeprops={'linewidth':
↪5},textprops={'fontsize': 13,'color':'black'})

plt3.set_title('Transaction by Married and Unmarried',{'font':'serif','size':
↪15,'weight':'bold'})

plt4=figure.add_subplot(grid_spec[2,:])
color_map=["#A1887F","#3A7089"]
sns.
↪kdeplot(data=df,x='Purchase',hue='Marital_Status',palette=color_map,fill=True,alpha=1,ax=pl

for k in ['top','left','right']:
    plt4.spines[k].set_visible(False)

plt4.set_yticks([])
plt4.set_ylabel('')
plt4.set_xlabel('Purchase Amount',fontweight='bold',fontsize=12)

plt4.set_title('Purchase Amount Distribution by Marital Status',{'font':
↪'serif', 'size':15,'weight':'bold'})
plt.show()
```

```
C:\Users\shobh\AppData\Local\Temp\ipykernel_39048\444023375.py:52: UserWarning:
FixedFormatter should only be used together with FixedLocator
  plt2.set_xticklabels(A_M['Marital_Status'],fontweight='bold',fontsize=12)
```



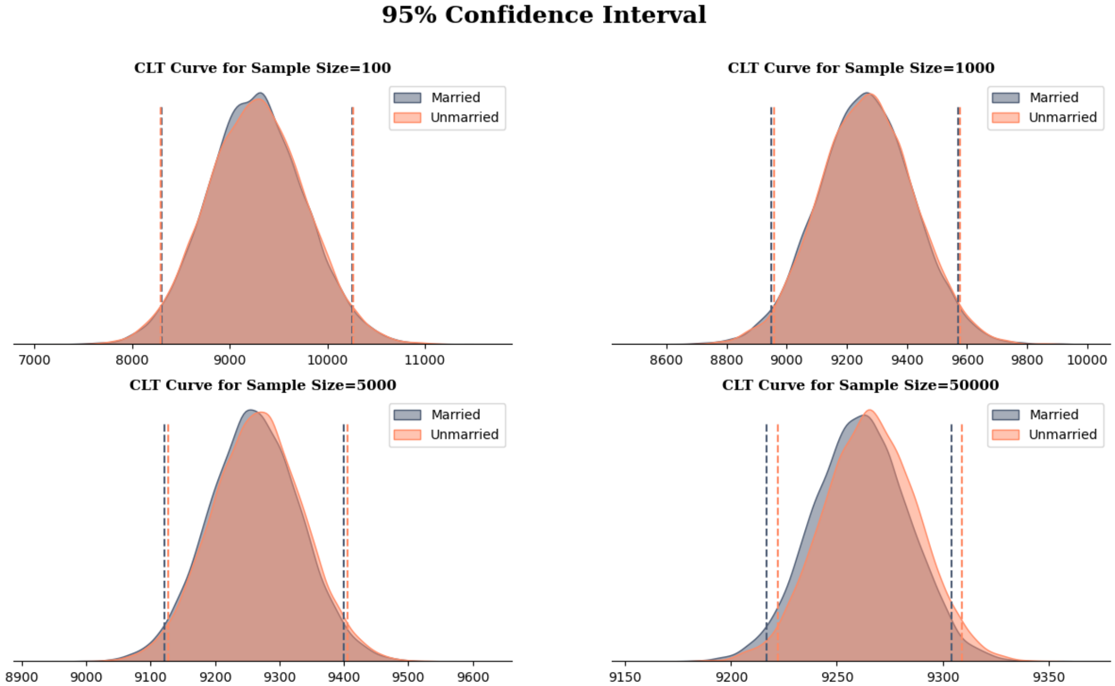

## 16 Confidence Interval

[39]:

```
# 95 Percent Confidence Level
confidence_level = 95

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

# Creating separate data frames for each marital status
A_M_married = df.loc[df['Marital_Status'] == 'Married', 'Purchase']
A_M_unmarried = df.loc[df['Marital_Status'] == 'Unmarried', 'Purchase']

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

```python
bootstrap_samples = 20000
married_sample_95 = {}
unmarried_sample_95 = {}

for i, x, y in sample_sizes:
    married_means = []
    unmarried_means = []
    for j in range(bootstrap_samples):
        married_bootstrapped_samples = np.random.choice(A_M_married, size=i)
        unmarried_bootstrapped_samples = np.random.choice(A_M_unmarried, size=i)

        married_sample_mean = np.mean(married_bootstrapped_samples)
        unmarried_sample_mean = np.mean(unmarried_bootstrapped_samples)

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


    married_sample_95[f'{confidence_level}%_{i}'] = married_means
    unmarried_sample_95[f'{confidence_level}%_{i}'] = unmarried_means

    df1 = pd.DataFrame(data={'married_means': married_means, 'unmarried_means':␣
↪unmarried_means})

    plt5 = figure.add_subplot(grid_spec[x, y])

    sns.kdeplot(data=df1, x='married_means', color="#4F5D75", fill=True,␣
↪alpha=0.5, ax=plt5, label='Married')
    sns.kdeplot(data=df1, x='unmarried_means', color="#FF8A65", fill=True,␣
↪alpha=0.5, ax=plt5, label='Unmarried')

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

    for k in m_range:
        plt5.axvline(x=k, ymax=0.9, color="#4F5D75", linestyle='--')
    for k in u_range:
        plt5.axvline(x=k, ymax=0.9, color="#FF8A65", linestyle='--')

    for l in ['top', 'left', 'right']:
        plt5.spines[l].set_visible(False)

    plt5.set_yticks([])
    plt5.set_ylabel('')
    plt5.set_xlabel('')
```

```
    plt5.set_title(f'CLT Curve for Sample Size={i}', {'font': 'serif', 'size':␣
↪11, 'weight': 'bold'})
    plt.legend()

figure.suptitle(f'{confidence_level}% Confidence Interval', font='serif',␣
↪size=18, weight='bold')
married_sample_95
unmarried_sample_95
plt.show()
```




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

[40]:
```
figure,plot=plt.subplots(figsize=(20,3))

married_ci=['Married']
unmarried_ci=['Unmarried']

for m in married_sample_95:
    married_range=confidence_interval(married_sample_95[m],95)
    married_ci.append(f"CI=${married_range[0]:.0f}-${married_range[1]:.0f},␣
↪Range={(married_range[1]-married_range[0]):.0f}")
```

```python
for u in unmarried_sample_95:
    unmarried_range=confidence_interval(unmarried_sample_95[u],95)
    unmarried_ci.append(f"CI=${unmarried_range[0]:.0f}-${unmarried_range[1]:.0f}, Range={(unmarried_range[1]-unmarried_range[0]):.0f}")


conf_int_info=[married_ci,unmarried_ci]


table=plot.table(cellText=conf_int_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)
plot.axis('off')

plot.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=8300 – 10251, Range=1951 | CI=8948 – 9570, Range=622 | CI=9122 – 9400, Range=278 | CI=9217 – 9304, Range=87 |
| Unmarried | CI=8291 – 10263, Range=1972 | CI=8959 – 9576, Range=617 | CI=9127 – 9406, Range=279 | CI=9222 – 9309, Range=87 |

## 18 Insights

1. The analysis underscores the significance of sample size in estimating population parameters. It indicates that with larger sample sizes, confidence intervals become narrower and offer more precise estimates. In business contexts, this suggests that increasing sample sizes can yield more dependable insights and estimations.
2. Examining the confidence intervals from the analysis reveals overlapping intervals across all sample sizes. This implies that there is no statistically significant disparity in average spending per transaction between married and unmarried customers within the provided samples.
3. With 95% confidence, we ascertain that the genuine population average for married customers lies within the range of $9,217-$9,305, while for unmarried customers, it falls between $9,221-$9,309.
4. The convergence of confidence intervals for average spending among married and unmarried customers suggests comparable spending patterns between the two groups. This indicates a

similarity in spending behavior irrespective of marital status.

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

- Walmart might find it unnecessary to target one group exclusively over the other. Instead, they could concentrate on implementing broader marketing strategies that resonate with both demographics simultaneously.

## 19.1 Different Age Group vs Purchase Amount

[41]:
```python
A_P=df.groupby('Age')['Purchase'].agg(['sum','count']).reset_index()

A_P['amount_in_billions']=round(A_P['sum']/ 10**9,2)

A_P['sum_percentage']=round(A_P['sum']/A_P['sum'].sum(),3)

A_P['per_purchase']=round(A_P['sum']/A_P['count'])

A_P
```

[41]:

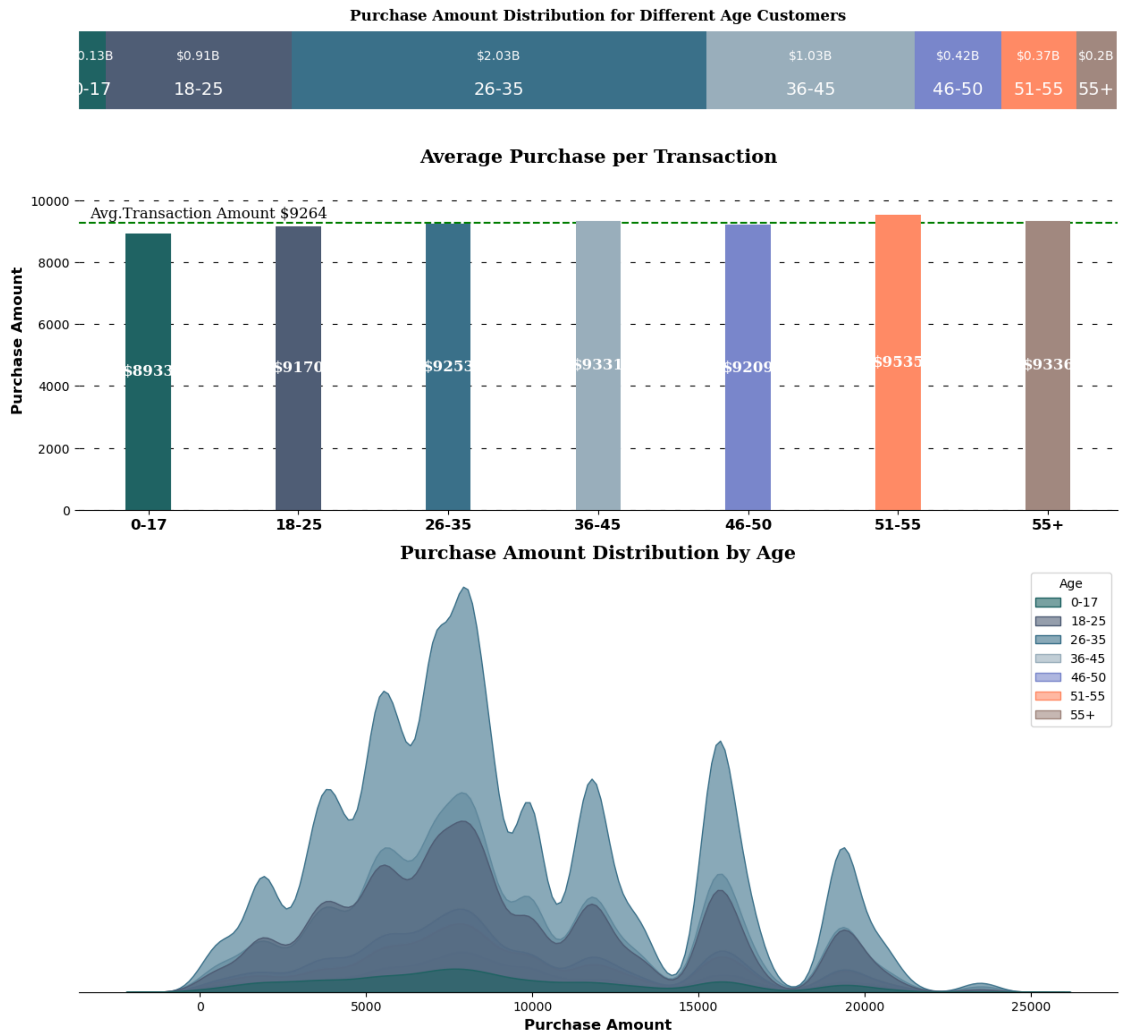
| | Age | sum | count | amount_in_billions | sum_percentage | per_purchase |
|---|---|---|---|---|---|---|
| 0 | 0-17 | 134913183 | 15102 | 0.13 | 0.026 | 8933.0 |
| 1 | 18-25 | 913848675 | 99660 | 0.91 | 0.179 | 9170.0 |
| 2 | 26-35 | 2031770578 | 219587 | 2.03 | 0.399 | 9253.0 |
| 3 | 36-45 | 1026569884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

[42]:
```python
# Setting the plot style
figure=plt.figure(figsize=(15,14))
grid_spec=figure.add_gridspec(3,1,height_ratios=[0.10,0.4,0.5])

# Distribution of Purchase Amount
plt1=figure.add_subplot(grid_spec[0])
color_map=["#1F6363","#4F5D75","#3A7089","#99AEBB","#7986CB","#FF8A65","#A1887F"]

left=0

for i in A_P.index:
    plt1.barh(A_P.loc[0,'Age'],width=A_P.
 ↪loc[i,'sum_percentage'],left=left,color=color_map[i],label=A_P.loc[i,'Age'])
    left+=A_P.loc[i,'sum_percentage']

text=0.0
```

```
for i in A_P.index:
    plt1.text(A_P.loc[i,'sum_percentage']/2+text,0.15,f"${A_P.
↪loc[i,'amount_in_billions']}B",
           va='center',ha='center',fontsize=10,color='white')

    plt1.text(A_P.loc[i,'sum_percentage']/2+text,-0.20,f"{A_P.loc[i,'Age']}",
           va='center',ha='center',fontsize=14,color='white')

    text+=A_P.loc[i,'sum_percentage']


for s in ['top','left','right','bottom']:
    plt1.spines[s].set_visible(False)

plt1.set_xticks([])
plt1.set_yticks([])
plt1.set_xlim(0,1)

plt1.set_title('Purchase Amount Distribution for Different Age␣
↪Customers',{'font':'serif','size':12,'weight':'bold'})

plt2=figure.add_subplot(grid_spec[1])

plt2.bar(A_P['Age'],A_P['per_purchase'],color=color_map,zorder=2,width=0.3)

avg=round(df['Purchase'].mean())
plt2.axhline(y=avg,color='green',zorder=0,linestyle='--')

plt2.text(0.4,avg+300,f"Avg.Transaction Amount ${avg:.0f}",
         {'font':'serif','size':12},ha ='center',va ='center')

plt2.set_ylim(0,11000)

for i in A_P.index:
    plt2.text(A_P.loc[i,'Age'],A_P.loc[i,'per_purchase']/2,f"${A_P.
↪loc[i,'per_purchase']:.0f}",
              {'font':'serif','size':12,'color':'white','weight':'bold' },ha␣
↪='center',va ='center')


plt2.grid(color='black',linestyle='--',axis='y',zorder=0,dashes=(5,10))

for j in ['top','left','right']:
    plt2.spines[j].set_visible(False)

plt2.set_ylabel('Purchase Amount',fontweight='bold',fontsize=12)
plt2.set_xticklabels(A_P['Age'],fontweight='bold',fontsize=12)
```

```
plt2.set_title('Average Purchase per Transaction',{'font':'serif','size':
 ↪15,'weight':'bold'})

plt3=figure.add_subplot(grid_spec[2,:])

sns.kdeplot(data=df,x='Purchase',hue='Age',palette=color_map,fill=True,alpha=0.
 ↪6,ax=plt3)

for k in ['top','left','right']:
    plt3.spines[k].set_visible(False)

plt3.set_yticks([])
plt3.set_ylabel('')
plt3.set_xlabel('Purchase Amount',fontweight='bold',fontsize=12)

plt3.set_title('Purchase Amount Distribution by Age',{'font':'serif', 'size':
 ↪15,'weight':'bold'})
plt.show()
```

```
C:\Users\shobh\AppData\Local\Temp\ipykernel_39048\3184049153.py:59: UserWarning:
FixedFormatter should only be used together with FixedLocator
  plt2.set_xticklabels(A_P['Age'],fontweight='bold',fontsize=12)
```




# 20 Insights

**1. Total Sales Breakdown:** The age bracket spanning from 26 to 45 years constitutes nearly 60% of the total sales, indicating a strong preference for Walmart's Black Friday deals within these demographics. Conversely, the age group 0-17 contributes the lowest percentage (2.6%) to sales, which aligns with their expected lower purchasing power. Tailoring offerings to their preferences and providing special incentives could foster customer loyalty as they mature.

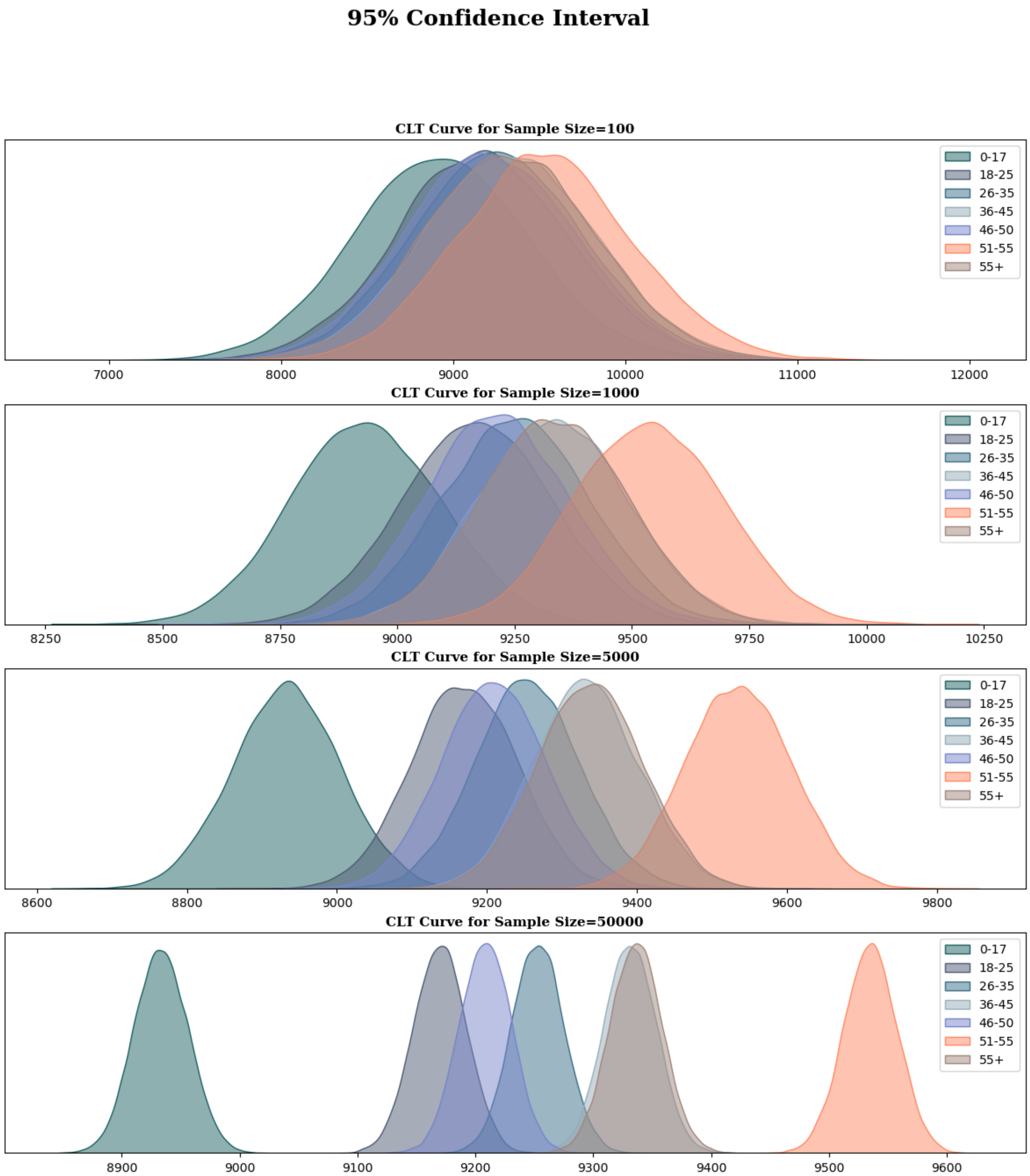
**2. Average Transaction Value:** While there isn't a notable discrepancy in per-purchase spending across age groups, the 51-55 age range stands out with a comparatively low sales percentage (7.2%) but the highest average transaction value at $9535. Walmart might explore targeted strategies to engage and retain this demographic with its higher spending potential.

**3. Purchase Amount Distribution:** The distribution of purchase amounts across age groups exhibits non-normal patterns, as illustrated above.

## 21 Confidence Interval

[43]:
```
confidence_level = 95

figure=plt.figure(figsize=(15, 15))
grid_specs=figure.add_gridspec(4,1)

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

sample_sizes = [(100,0),(1000,1),(5000,2),(50000,3)]
bootstrap_samples = 20000
samples1,samples2,samples3,samples4,samples5,samples6,samples7 =␣
↪{},{},{},{},{},{},{}

for i,x in sample_sizes:
    l1,l2,l3,l4,l5,l6,l7 = [],[],[],[],[],[],[]
    for j in range(bootstrap_samples):
        #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)

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

    age_df= pd.DataFrame(data = {'0-17':l1,'18-25':l2,'26-35':l3,'36-45':
↪l4,'46-50':l5,'51-55':l6,'55+':l7})

    plt1=figure.add_subplot(grid_specs[x])

    for a,b in [('#1F6363', '0-17'),('#4F5D75', '18-25'),('#3A7089',␣
↪'26-35'),('#99AEBB', '36-45'),('#7986CB', '46-50'),
                ('#FF8A65', '51-55'),('#A1887F', '55+')]:
        sns.kdeplot(data=age_df,x=b,color=a,fill=True,alpha=0.5,ax␣
↪=plt1,label=b)
    for l in ['top', 'left', 'right']:
        plt5.spines[l].set_visible(False)

    plt1.set_yticks([])
    plt1.set_ylabel('')
    plt1.set_xlabel('')

    plt1.set_title(f'CLT Curve for Sample Size={i}', {'font': 'serif', 'size':␣
↪11, 'weight': 'bold'})
    plt.legend()

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




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

[44]:
```
figure,plt1=plt.subplots(figsize=(20,5))
conf_1,conf_2,conf_3,conf_4,conf_5,conf_6,conf_7=['0-17'],['18-25'],['26-35'],['36-45'],['46-5
```

```
samples =
↪[(samples1,conf_1),(samples2,conf_2),(samples3,conf_3),(samples4,conf_4),(samples5,conf_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_range[0]):.0f}")

ci_info=[conf_1,conf_2,conf_3,conf_4,conf_5,conf_6,conf_7]
table=plt1.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)

plt1.axis('off')

plt1.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=7953 – 9943,Range=1990 | CI=8619 – 9250,Range=631 | CI=8791 – 9076,Range=285 | CI=8889 – 8979,Range=90 |
| 18-25 | CI=8188 – 10172,Range=1984 | CI=8856 – 9484,Range=628 | CI=9031 – 9308,Range=277 | CI=9125 – 9214,Range=89 |
| 26-35 | CI=8298 – 10248,Range=1950 | CI=8944 – 9561,Range=617 | CI=9115 – 9393,Range=278 | CI=9209 – 9297,Range=88 |
| 36-45 | CI=8358 – 10327,Range=1969 | CI=9026 – 9648,Range=622 | CI=9195 – 9471,Range=276 | CI=9287 – 9375,Range=88 |
| 46-50 | CI=8253 – 10190,Range=1937 | CI=8903 – 9519,Range=616 | CI=9072 – 9347,Range=275 | CI=9165 – 9252,Range=87 |
| 51-55 | CI=8558 – 10550,Range=1992 | CI=9223 – 9850,Range=627 | CI=9396 – 9673,Range=277 | CI=9490 – 9579,Range=89 |
| 55+ | CI=8363 – 10313,Range=1950 | CI=9028 – 9641,Range=613 | CI=9198 – 9473,Range=275 | CI=9292 – 9380,Range=88 |

# 23 Insights

**1. Sample Size Importance:** This analysis underscores the critical role of sample size in accurately estimating population parameters. It reveals that as the sample size increases, the confidence intervals narrow, leading to more precise estimates. In a business context, this emphasizes the significance of larger sample sizes in generating reliable insights and estimates.

**2. Confidence Intervals and Spending Patterns:** Observing the confidence interval overlaps across various age groups, we can categorize average spending into distinct age brackets: - 0 - 17: Customers in this age range exhibit the lowest spending per transaction. - 18 - 25, 26 - 35, 46 - 50: These age groups demonstrate overlapping confidence intervals, suggesting similar purchasing behaviors. - 36 - 45, 55+: Customers within these age brackets also exhibit overlapping confidence intervals, indicating comparable spending patterns. - 51 - 55: Notably, customers in this age group display the highest spending per transaction.

**3. Population Average Estimation:** With 95% confidence, we estimate the true population average for the following age groups to fall within the specified ranges: - 0 - 17: $8,888 to $ 8,978 - 18 - 25: $9,125 to $ 9,214 - 26 - 35: $9,208 to $ 9,296 - 36 - 45: $9,287 to $ 9,375 - 46 - 50: $9,165 to $ 9,253 - 51 - 55: $9,491 to $ 9,579 - 55+ : $9,292 to $ 9,380

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

**1. Targeted Marketing Approach:** Understanding that customers aged 0 - 17 have the lowest spending per transaction, Walmart can implement strategies to boost their transaction value. This could involve enticing discounts, coupons, or rewards programs tailored specifically to this demographic. Additionally, adapting product offerings and marketing initiatives to align with the preferences and interests of this age group can further enhance their engagement with Walmart's offerings.

**2. Customer Segmentation Strategy:** Given the similarities in purchasing behavior among customers aged 18 - 25, 26 - 35, and 46 - 50, as well as between those aged 36 - 45 and 55+, Walmart can optimize its product assortment to cater effectively to the preferences of these demographics. Moreover, leveraging this insight, Walmart can refine its pricing strategies to better resonate with the distinct age groups within its customer base.

**3. Premium Service Enhancement:** Recognizing the robust spending behavior of customers aged 51 - 55, Walmart can explore avenues to elevate their shopping experience. This may entail introducing premium services, such as personalized recommendations or exclusive loyalty programs tailored to the unique preferences and spending patterns of this demographic. Such initiatives can further solidify Walmart's appeal and foster greater customer loyalty within this age group.

## 25 Recommendations

**1. Targeting Male Shoppers:** - Given their significant contribution to Black Friday sales and tendency to spend more per transaction, Walmart should tailor its marketing strategies and product offerings to encourage higher spending among male customers. Additionally, it's crucial to maintain competitive pricing for products targeted towards female shoppers.

**2. Focusing on the 26 - 45 Age Group:** - With this age bracket accounting for the majority of sales, Walmart should prioritize meeting the preferences and requirements of customers aged between 26 and 45. This may involve offering exclusive deals on products popular within this demographic segment.

**3. Engagement with Younger Shoppers:** - Recognizing the lower spending per transaction among customers aged 0 - 17, Walmart can stimulate their spending by providing enticing discounts, coupons, or rewards programs. Initiating efforts to foster brand loyalty among younger consumers is also essential.

**4. Customer Segmentation Strategies:** - Given the analogous buying behaviors observed among customers aged 18 - 25, 26 - 35, and 46 - 50, as well as between those aged 36 - 45 and 55+, Walmart can optimize its product assortment to align with the preferences of these age groups.

Furthermore, leveraging this data can inform adjustments to pricing strategies tailored to distinct age demographics.

**5. Enhancing the Shopping Experience for the 51 - 55 Age Group:** - Considering the higher spending per transaction among customers aged 51 - 55, Walmart could offer exclusive pre-sale access, special discounts, or personalized product recommendations tailored specifically for this demographic. Introducing loyalty programs designed to reward and retain customers within this age bracket could also be beneficial.

**6. Post-Black Friday Engagement:** - Following Black Friday, Walmart should maintain engagement with customers who made purchases by deploying follow-up emails or offering related product promotions. This proactive approach can bolster customer retention and encourage repeat business throughout the holiday season and beyond.

[ ]:
```

```