Business Case: "Walmart" - Confidence Interval and CLT

Dataset Description:
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.

Dataset:
The company collected the transactional data of customers who purchased products from the Walmart Stores during Black Friday. The dataset has the following features:

- **User_ID:** User ID
- **Product_ID:** Product ID
- **Gender:** Sex of User
- **Age:** Age in bins
- **Occupation:** Occupation(Masked)
- **City_Category:** Category of the City (A,B,C)
- **StayInCurrentCityYears:** Number of years stay in current city
- **Marital_Status:** Marital Status
- **ProductCategory:** Product Category (Masked)
- **Purchase:** Purchase Amount

Dataset Link:
https://www.kaggle.com/code/ankur561999/walmart-confidence-interval-and-clt (https://www.kaggle.com/code/ankur561999/walmart-confidence-interval-and-clt)

(1) Defining Problem Statement and Analyzing Basic Metrics

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)**

In [ ]:

```
# Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import statistics
from scipy.stats import norm
import scipy.stats as stats

# To ignore the warnings & make code more representable
import warnings
warnings.filterwarnings("ignore")

# Load aerofit_treadmill.csv dataset into a pandas dataFrame
url = "https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/wa
df = pd.read_csv(url)
```

Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), statistical summary

In [ ]:

```
# Display the first few rows of the dataset
df.head()
```

Out[3]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
3	1000001	P00085442	F	0-17	10	A	2
4	1000002	P00285442	M	55+	16	C	4+

In [ ]:

```
# Check the shape of the dataset
df.shape
```

Out[4]:

```
(550068, 10)
```

In [ ]:

```
print(f"Number of rows: {df.shape[0]}\nNumber of columns: {df.shape[1]}")
```

```
Number of rows: 550068
Number of columns: 10
```

In [ ]:

```
# Check all the columns of the dataset
df.columns
```

Out[6]:

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

In [ ]:

```
# Check data types of attributes
df.dtypes
```

Out[7]:

```
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
```

In [ ]:

```
# Check Data Types and Missing Values
df.info()
```

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

Observations:

- **No null values** found in the data
- Product_ID, Gender, Age, City_Category & Stay_In_Current_City_Years column has **object datatype** while the rest of the column name has **integer datatype**

In [ ]:

```
# Display statistical summary of numerical columns
df.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	
User_ID	550068.0	1.003029e+06	1727.591586	1000001.0	1001516.0	1003077.0	100
Occupation	550068.0	8.076707e+00	6.522660	0.0	2.0	7.0	
Marital_Status	550068.0	4.096530e-01	0.491770	0.0	0.0	0.0	
Product_Category	550068.0	5.404270e+00	3.936211	1.0	1.0	5.0	
Purchase	550068.0	9.263969e+03	5023.065394	12.0	5823.0	8047.0	1

In [ ]:

```
# Display statistical summary of object datatpes
df.describe(include = 'object').T
```

Out[10]:

	count	unique	top	freq
Product_ID	550068	3631	P00265242	1880
Gender	550068	2	M	414259
Age	550068	7	26-35	219587
City_Category	550068	3	B	231173
Stay_In_Current_City_Years	550068	5	1	193821

In [ ]:

```
# Display statistical summary of all the columns
df.describe(include='all')
```

Out[11]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Cu
count	5.500680e+05	550068	550068	550068	550068.000000	550068	
unique	NaN	3631	2	7	NaN	3	
top	NaN	P00265242	M	26-35	NaN	B	
freq	NaN	1880	414259	219587	NaN	231173	
mean	1.003029e+06	NaN	NaN	NaN	8.076707	NaN	
std	1.727592e+03	NaN	NaN	NaN	6.522660	NaN	
min	1.000001e+06	NaN	NaN	NaN	0.000000	NaN	
25%	1.001516e+06	NaN	NaN	NaN	2.000000	NaN	
50%	1.003077e+06	NaN	NaN	NaN	7.000000	NaN	
75%	1.004478e+06	NaN	NaN	NaN	14.000000	NaN	
max	1.006040e+06	NaN	NaN	NaN	20.000000	NaN	

Observations:

- **"Count"** row indicates the number of **non-null values** in each column. There are **no missing values** in any columns of the data.
- The **"Unique"** row represents the **number of unique values** in each column. There are there are **2 unique values** in **"Gender"**, and in **"City_Category"** there are **3 unique values** in the dataset.
- The **"Top"** row displays the **most frequently occurring value** in each column. For instance, in the **"Gender"** column, **"Male"** is the most common value.
- The **"Purchase"** column has **mean (average)** values approximately 9263.97, **minimum** value as 12, the **25th percentile (first quartile)** as 5823, **median value (50th percentile)** as 8047, the **75th percentile (third quartile)** value as 12054 and the **maximum** value as 23961.

Non-Graphical Analysis: Value counts and unique attributes

Performing non-graphical analysis by using **value counts** and examining **unique** attributes for the **categorical columns** in your **DataFrame**.

In [ ]:

```
# Gender column
```

In [ ]:

```
# To get the unique values in the 'Gender' column
unique_genders = df['Gender'].unique()
print(unique_genders, '\n')

# To get the count of each gender category
gender_counts = df['Gender'].value_counts()
print(gender_counts)
```

```
['F' 'M']

M    414259
F    135809
Name: Gender, dtype: int64
```

In [ ]:

```
# Age   columns
```

In [ ]:

```
# To get the unique age groups
unique_ages = df['Age'].unique()
print(unique_ages, '\n')

# To get the count of each age group
age_counts = df['Age'].value_counts()
print(age_counts)
```

```
['0-17' '55+' '26-35' '46-50' '51-55' '36-45' '18-25']

26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
```

In [ ]:

```
# City_Category column
```

In [ ]:

```
# To get the unique city categories
unique_city_categories = df['City_Category'].unique()
print(unique_city_categories, '\n')

# To get the count of each city category
city_category_counts = df['City_Category'].value_counts()
print(city_category_counts)
```

```
['A' 'C' 'B']

B    231173
C    171175
A    147720
Name: City_Category, dtype: int64
```

In [ ]:

```
# Stay_In_Current_City_Years column
```

In [ ]:

```
# To get the unique values for the stay duration
unique_stay_duration = df['Stay_In_Current_City_Years'].unique()
print(unique_stay_duration, '\n')

# To get the count of each stay duration category
stay_duration_counts = df['Stay_In_Current_City_Years'].value_counts()
print(stay_duration_counts)
```

```
['2' '4+' '3' '1' '0']

1     193821
2     101838
3      95285
4+     84726
0      74398
Name: Stay_In_Current_City_Years, dtype: int64
```

In [ ]:

```
# Marital_Status and Product_Category columns
```

In [ ]:

```
# For numeric categorical columns like 'Marital_Status' and 'Product_Category,' you can u
marital_status_counts = df['Marital_Status'].value_counts()
print(marital_status_counts, '\n')

product_category_counts = df['Product_Category'].value_counts()
print(product_category_counts)
```

```
0    324731
1    225337
Name: Marital_Status, dtype: int64

5     150933
1     140378
8     113925
11     24287
2      23864
6      20466
3      20213
4      11753
16      9828
15      6290
13      5549
10      5125
12      3947
7       3721
18      3125
20      2550
19      1603
14      1523
17       578
9        410
Name: Product_Category, dtype: int64
```

Observations:

- The data is complete with **no missing values**.
- **Categorical columns (Gender, Age, City_Category, Stay_In_Current_City_Years)** have a **limited number of unique values**.
- **Value counts** provide insights into the distribution of categories within each column.
- The **Purchase** column has a wide range of purchase amounts, with a **mean** of approximately 9263.97.
- **User_ID** and **Product_ID** serve as **unique identifiers** for users and products, respectively, with varying levels of uniqueness.

Visual Analysis - Univariate & Bivariate

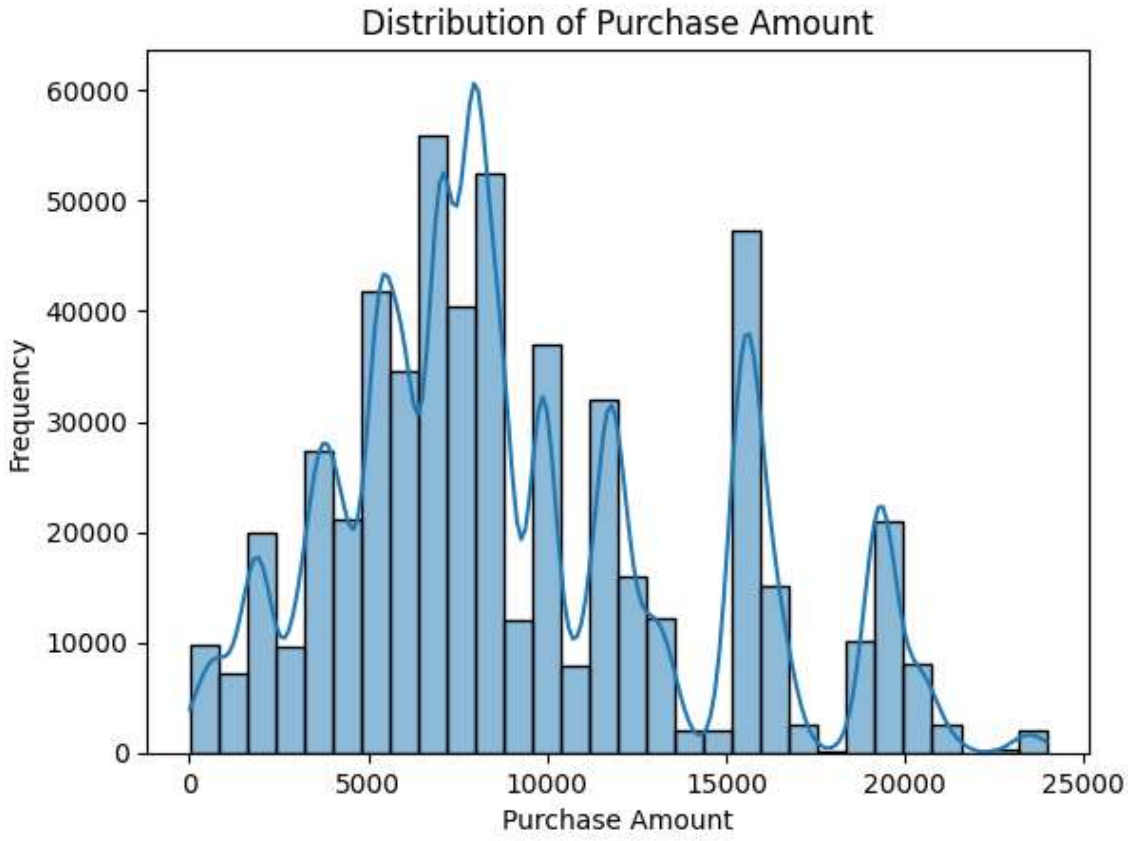
- For **continuous** variable(s): Distplot, countplot, histogram for **univariate analysis**
- For **categorical** variable(s): Boxplot
- For **correlation**: Heatmaps, Pairplots

In [ ]:

```
# Univariate Analysis for continuous variable
```

In [ ]:

```
# Univariate Analysis for Purchase using histplot
sns.histplot(df['Purchase'], bins=30, edgecolor='k', kde=True)
plt.xlabel('Purchase Amount')
plt.ylabel('Frequency')
plt.title('Distribution of Purchase Amount')
plt.show()
```



Observations:

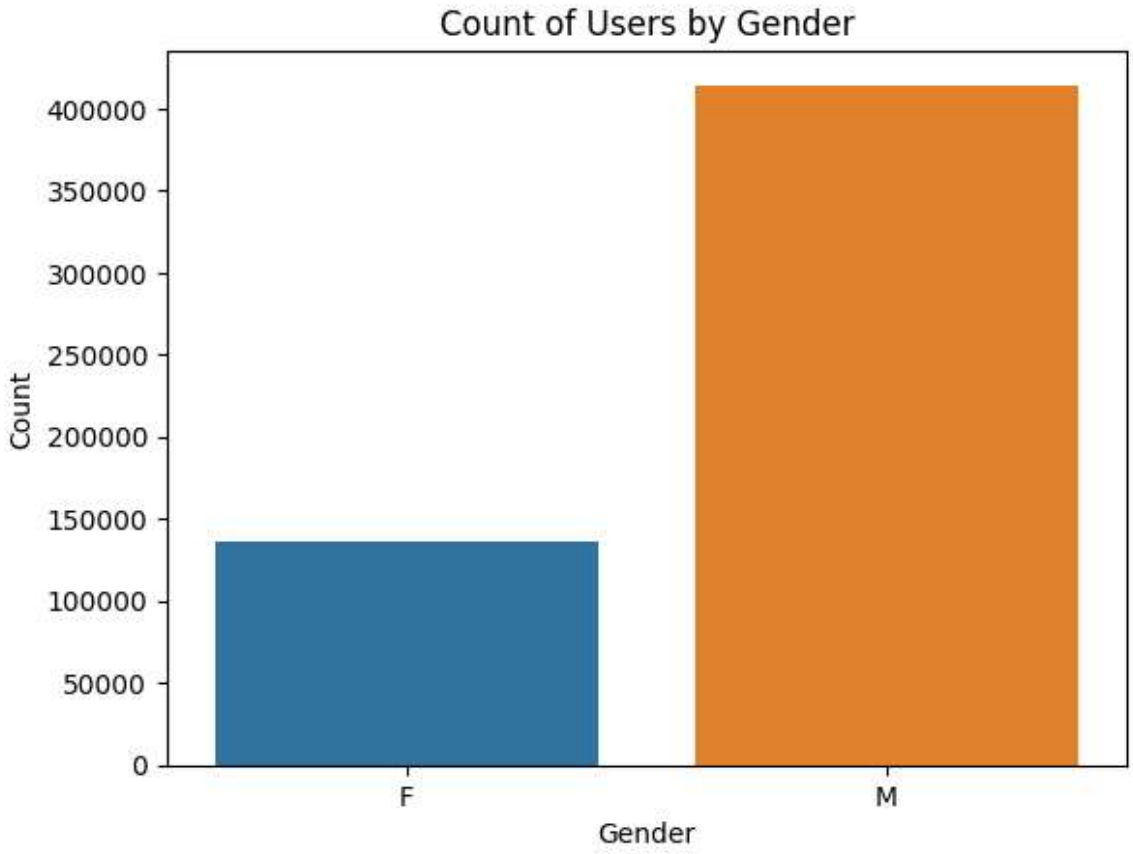
- The **distribution** appears to be **right-skewed**, with a peak towards **lower purchase amount**s. This suggests that the **majority of purchases** fall within a **lower price range**.
- The overlaid **KDE curve** provides a smooth estimate of the **probability density function**. It visually represents the likelihood of observing a **particular purchase amount**. The **KDE curve** shows that the **density is highest around the peak of the distribution**.

In [ ]:

```
# Univariate Analysis for Categorical Variables (e.g., Gender, Age, City_Category, Stay_I
```

In [ ]:

```
# Countplot for Gender
# Countplot used to visualize the frequency of each category within a categorical variabl
sns.countplot(x='Gender', data=df)
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Count of Users by Gender')
plt.show()
```

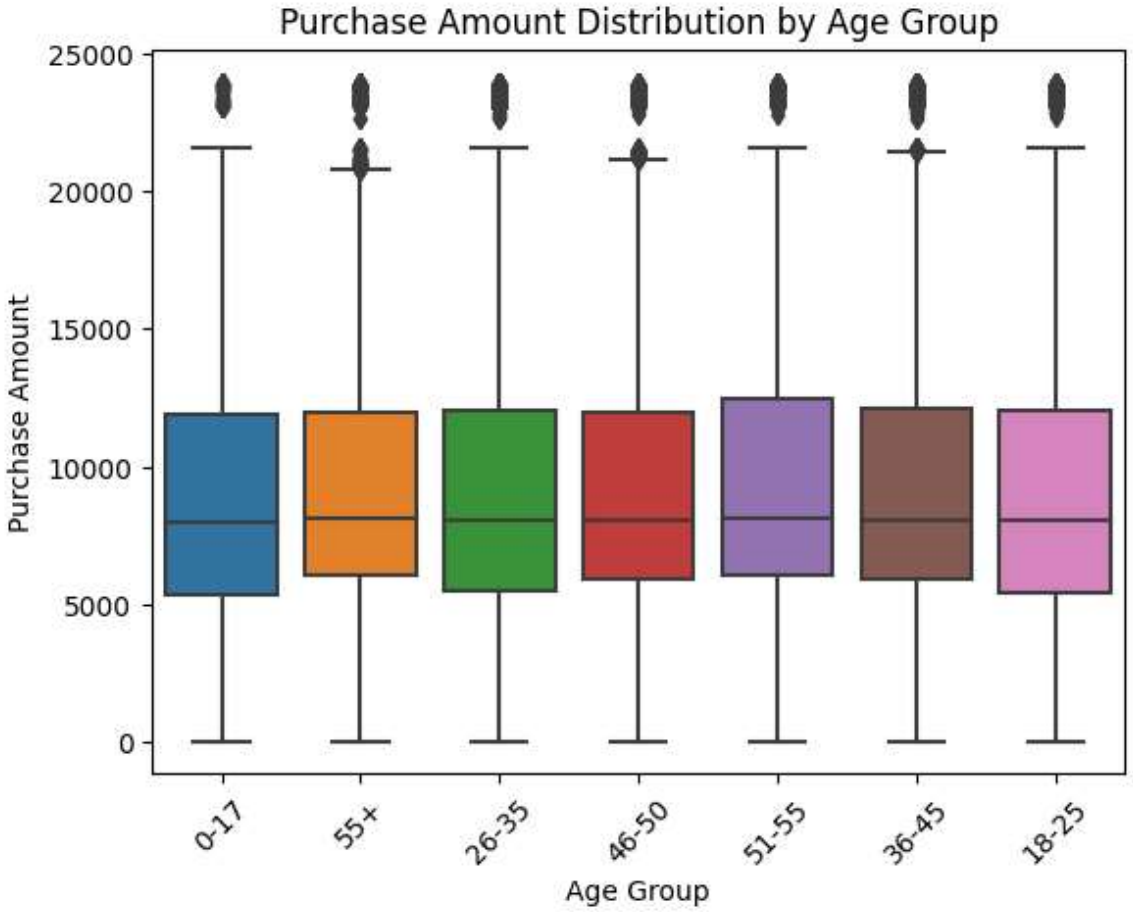


Observation:

- The **countplot** for **"Gender"** shows that the dataset has more entries for **males** than **females**, indicating that **males** are more represented in the data.

In [ ]:

```
# Boxplot for Age vs. Purchase
# Boxplot is typically used for continuous variables, but can also help understand the di
sns.boxplot(x='Age', y='Purchase', data=df)
plt.xlabel('Age Group')
plt.ylabel('Purchase Amount')
plt.title('Purchase Amount Distribution by Age Group')
plt.xticks(rotation=45)
plt.show()
```



Observation:

- The **boxplot** for **"Age" vs. "Purchase"** reveals that the **purchase amounts tend to vary across different age groups**. It shows the **central tendency, spread, and potential outlier**s for each age group.

In [ ]:

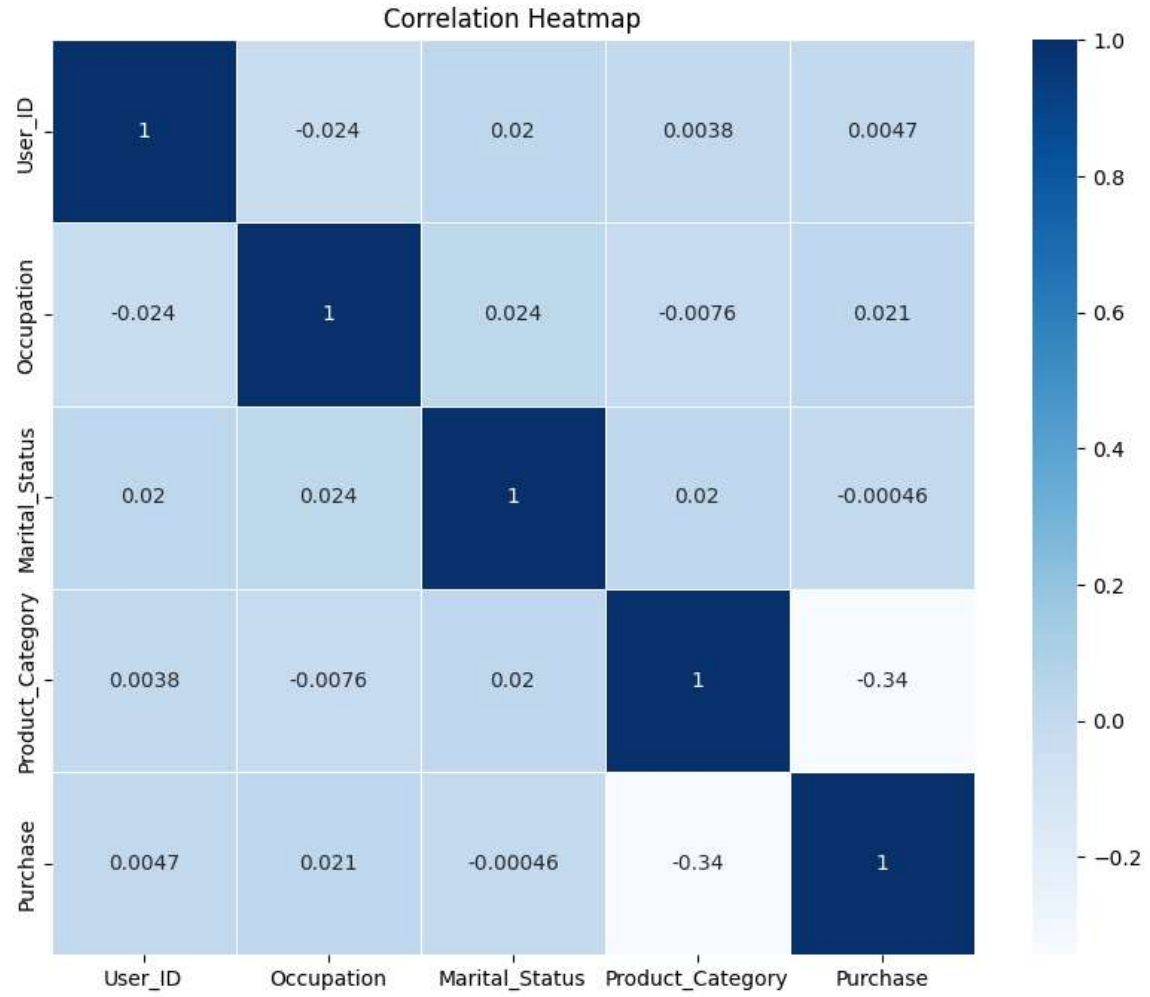
```
# Correlation Analysis
```

In [ ]:

```
# Creating a heatmap to visualize the correlation matrix between continuous variables. Th

# Calculate the correlation matrix
corr_matrix = df.corr()

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='Blues', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



Observation:

- The **correlation heatmap** provides a visual representation of the relationships between **pairs of continuous variables**. The **Strong positive correlations** are indicated by **darker colors**, while **negative correlations** are **lighter**.

In [ ]:

```
# Bivariate Analysis (Pairplots) for Continuous Variables
```

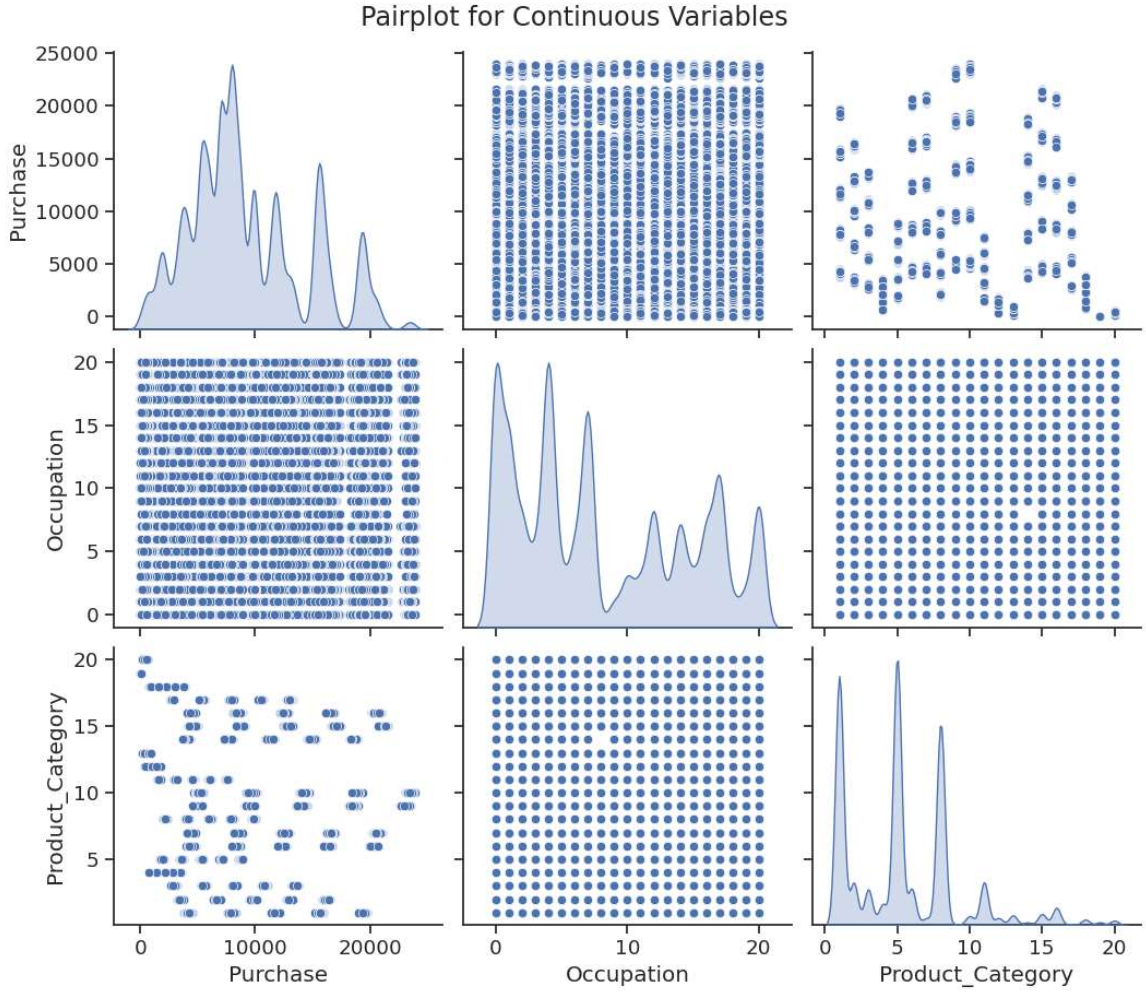
In [ ]:

```
# Define the subset of continuous variables you want to analyze
continuous_vars = df[['Purchase', 'Occupation', 'Product_Category']]

# Customize the pairplot appearance
sns.set(style="ticks", color_codes=True, font_scale=1.2)
g = sns.pairplot(continuous_vars, diag_kind="kde", markers="o", height=3, aspect=1.2)

# Set titles and labels
g.fig.suptitle("Pairplot for Continuous Variables", y=1.02)
g.axes[1, 0].set_ylabel("Occupation")
g.axes[2, 0].set_ylabel("Product_Category")
g.axes[2, 0].set_xlabel("Purchase")
g.axes[2, 1].set_xlabel("Occupation")
g.axes[2, 2].set_xlabel("Product_Category")

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



Observation:

- There **doesn't** seem to be a strong linear correlation between **"Purchase"** and the other **continuous variables ("Occupation" and "Product_Category")**.

(2) Missing Value & Outlier Detection

In [ ]:

```
'''
Since the DataFrame doesn't contain missing values, there's no need to generate code for
However, you can proceed with outlier detection for the "Purchase" column.
'''
```

Out[31]:

```
'\nSince the DataFrame doesn\'t contain missing values, there\'s no need t
o generate code for missing value detection. \nHowever, you can proceed wi
th outlier detection for the "Purchase" column. \n'
```

To find **Outliers**, we need to use **"Box Plot"** but prior to that we need to find the below 5 details:

- **25th percentile / Q1**
- **50th percentile / Q2 (Median)**
- **75th percentile / Q3**
- **IQR range b/w 25th percentile and 75th percentile**
- **Lower_bound**
- **Upper_bound**

In [ ]:

```
df["Purchase"].describe()
```

Out[32]:

```
count    550068.000000
mean       9263.968713
std        5023.065394
min          12.000000
25%        5823.000000
50%        8047.000000
75%       12054.000000
max       23961.000000
Name: Purchase, dtype: float64
```

In [ ]:

```
p_25 = np.percentile(df["Purchase"], 25)       #25th percentile or Q1
p_50 = np.percentile(df["Purchase"], 50)       #50th percentile or Q2 (Median)
p_75 = np.percentile(df["Purchase"], 75)       #75th percentile or Q3
print("25th percentile: ", p_25, "\n50th percentile: ", p_50, "\n75th percentile: ", p_75

iqr = p_75 - p_25       # IQR range b/w 25th percentile and 75th percentile
print("\nIQR: ", iqr)

lower = max(p_25 - 1.5*iqr, 0)        # Calculating min (i.e. lower) points: lower_bound
upper = p_75 + 1.5*iqr        # Calculating max (i.e. upper) points: upper_bound = min(ac
print("\nLower Bound: ", lower, "\nUpper Bound: ", upper)
```

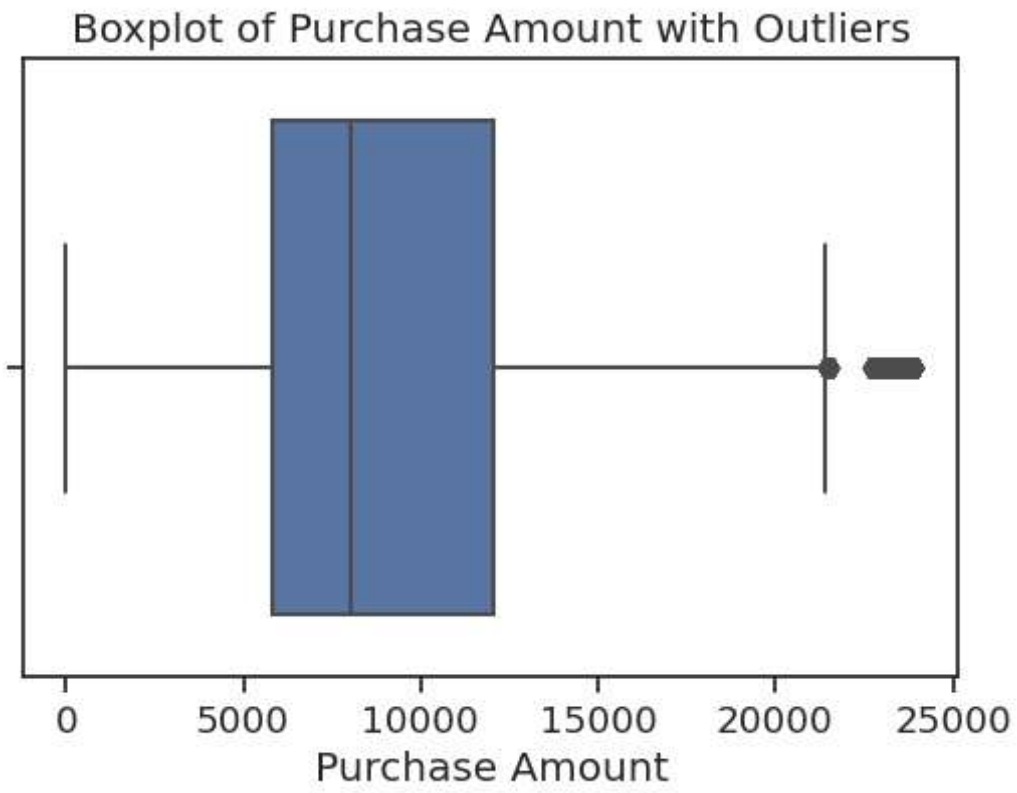
```
25th percentile:  5823.0
50th percentile:  8047.0
75th percentile:  12054.0

IQR:  6231.0

Lower Bound:  0
Upper Bound:  21400.5
```

In [ ]:

```
# Create a boxplot to visualize outliers
plt.figure(figsize=(6, 4))
sns.boxplot(data = df, x = "Purchase")
plt.title('Boxplot of Purchase Amount with Outliers')
plt.xlabel('Purchase Amount')
plt.show()
```



Observation:

In [ ]:

```
# Checking the % of data which is above the upper_bound (i.e. outliers)
purchase_outlier = df[df["Purchase"] > upper]
len(purchase_outlier)
```

Out[35]:

```
2677
```

In [ ]:

```
# (5 / 180)*100
# (len(purchase_outlier)/len(df))*100
(len(df[df["Purchase"] > upper])/len(df))*100
```

Out[36]:

```
0.4866671029763593
```

Obersations:

- **0.48%** values of **"Purchase"** column are **outliers** but we **do not** wish to drop these values as it may be required to draw some valuable insights and it may be useful for **customer profiling**.

(3) Exploratory data analysis (EDA) of the dataset

In [ ]:

```
'''
Do some data exploration steps like:
* Tracking the amount spent per transaction of all the 50 million female customers, and a
* Inference after computing the average female and male expenses.
* Use the sample average to find out an interval within which the population average will
you will calculate the interval within which the average spending of 50 million male and
'''
```

Out[37]:

```
'\nDo some data exploration steps like:\n* Tracking the amount spent per t
ransaction of all the 50 million female customers, and all the 50 million
male customers, calculate the average, and conclude the results.\n* Infere
nce after computing the average female and male expenses.\n* Use the sampl
e average to find out an interval within which the population average will
lie. Using the sample of female customers, \nyou will calculate the interv
al within which the average spending of 50 million male and female custome
rs may lie.\n'
```

Q1. Are women spending more money per transaction than men? Why or Why not?

In [ ]:

```
# Calculate the average purchase amount for women and men
avg_purchase_women = df[df['Gender'] == 'F']['Purchase'].mean()
avg_purchase_men = df[df['Gender'] == 'M']['Purchase'].mean()

# Print the average purchase amounts
print("Average purchase amount for women:", avg_purchase_women)
print("Average purchase amount for men:", avg_purchase_men)
```

```
Average purchase amount for women: 8734.565765155476
Average purchase amount for men: 9437.526040472265
```

In [ ]:

```
# Perform a t-test to check if the difference is significant
t_stat, p_value = stats.ttest_ind(df[df['Gender'] == 'F']['Purchase'], df[df['Gender'] ==

# Print the p-value
print("P-value:", p_value)

# Interpret the result
if p_value < 0.05:
    print("There is a significant difference in spending between women and men.")
else:
    print("There is no significant difference in spending between women and men.")
```

```
P-value: 0.0
There is a significant difference in spending between women and men.
```

Observation:

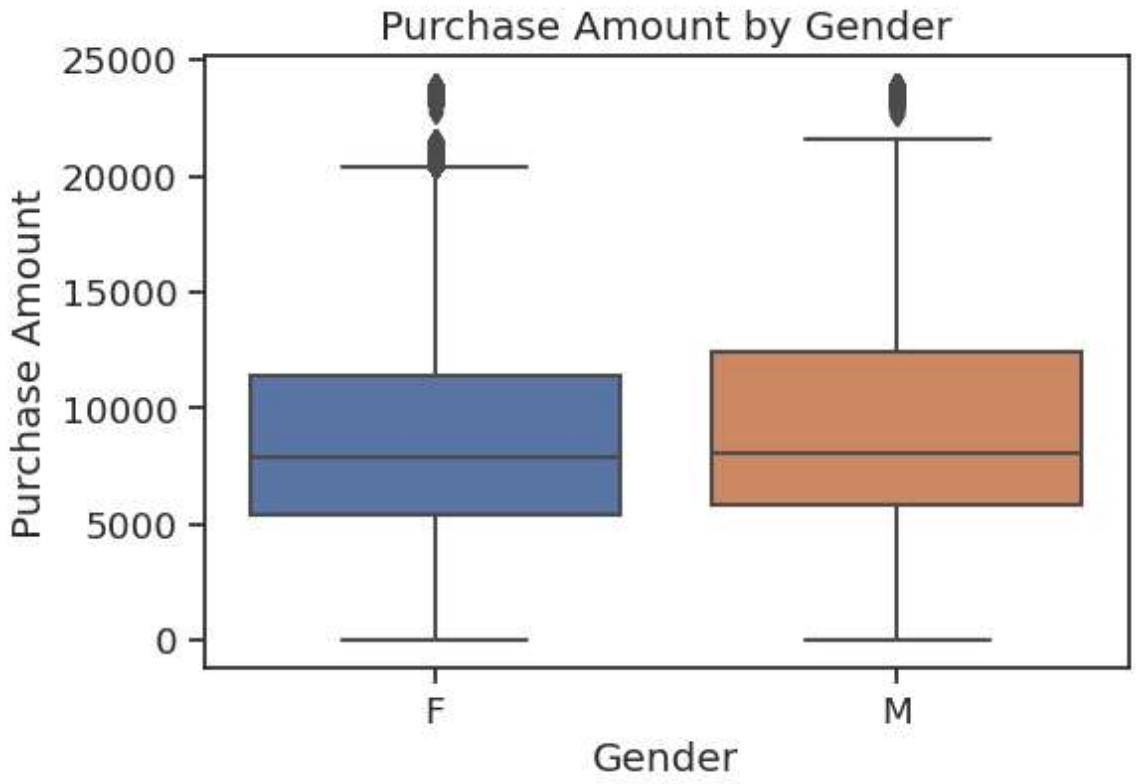
- The **average purchase amount for wome**n is approximately 8734.57, while the **average purchase amount for men** is approximately 9437.53.
- The **calculated p-value from the t-test is very close to zero (0.0)**, indicating an **extremely low probability** that the observed difference in spending between women and men is due to random chance.
- Since the p-value is less than the typical significance level of 0.05, we can conclude that **there is a statistically significant difference in spending between women and men**.

In [ ]:

```
# Create a boxplot for spending by gender
plt.figure(figsize=(6, 4))
sns.boxplot(x='Gender', y='Purchase', data=df)
plt.title('Purchase Amount by Gender')
plt.xlabel('Gender')
plt.ylabel('Purchase Amount')
```

Out[38]:

```
Text(0, 0.5, 'Purchase Amount')
```

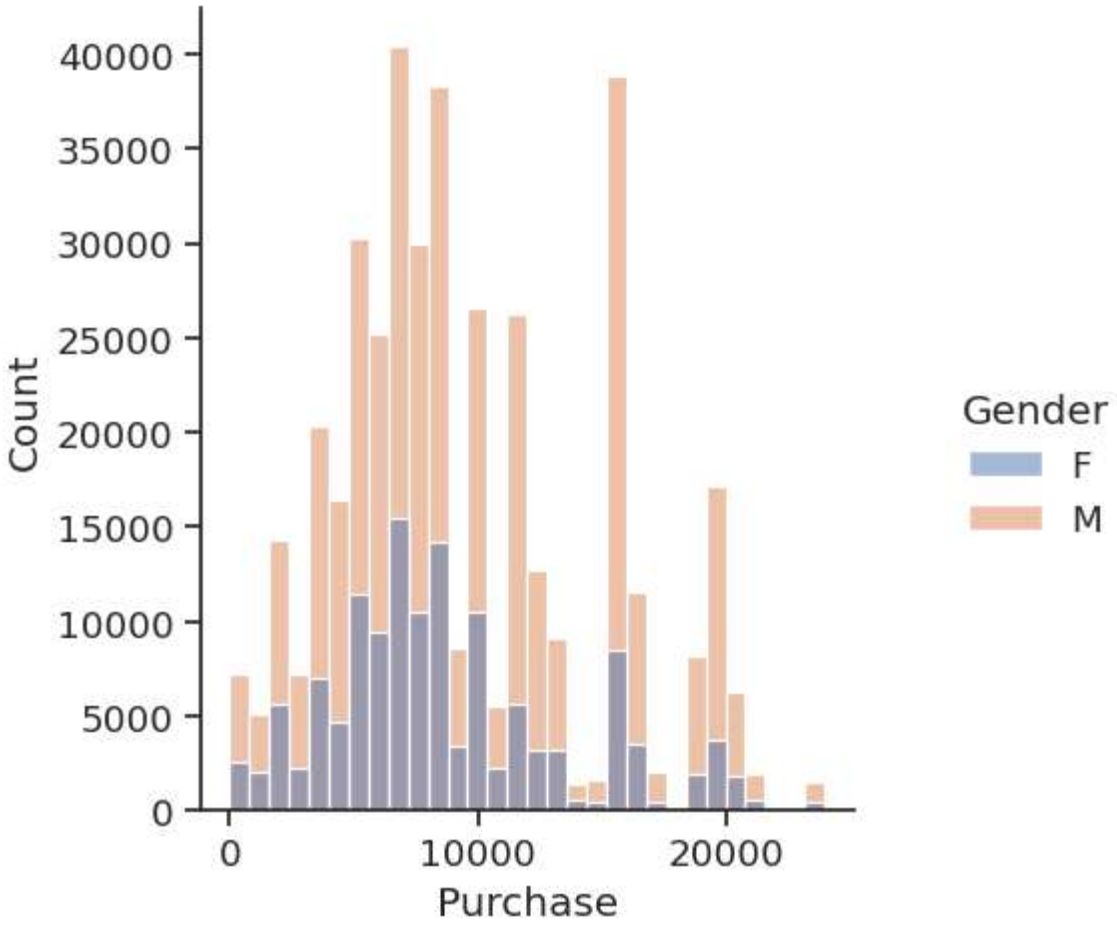


In [ ]:

```
sns.displot(x='Purchase', data=df, bins=30, hue='Gender')
```

Out[39]:

```
<seaborn.axisgrid.FacetGrid at 0x7fe0400daec0>
```



Observation:

- The amount of money spent by women is less than that spent by men

Q2. Confidence intervals and distribution of the mean of the expenses by female and male customers.

In [ ]:

```
# Calculate confidence intervals for female and male spending means
confidence_interval_women = stats.norm.interval(0.95, loc=df[df['Gender'] == 'F']['Purcha
                                                scale=stats.sem(df[df['Gender'] == 'F']['P
confidence_interval_men = stats.norm.interval(0.95, loc=df[df['Gender'] == 'M']['Purchase
                                              scale=stats.sem(df[df['Gender'] == 'M']['Pur

# Print confidence intervals
print("Confidence interval for female spending mean:", confidence_interval_women)
print("Confidence interval for male spending mean:", confidence_interval_men)
```

```
Confidence interval for female spending mean: (8709.21154714068, 8759.9199
83170272)
Confidence interval for male spending mean: (9422.01944736257, 9453.032633
581959)
```

Observation:

- The **confidence interval for the mean spending of female customers** is approximately (8709.21, 8759.92). This means that we are 95% confident that the true population mean spending for females falls within this interval.
- The **confidence interval for the mean spending of male customers** is approximately (9422.02, 9453.03). Similarly, we are 95% confident that the true population mean spending for males falls within this interval.
- Based on the **confidence interval**s, it can be observed that there is a statistically significant difference in the average spending between female and male customers. Male customers, on average, have a higher spending mean than female customers, and this difference is statistically supported.

Q3. Are confidence intervals of average male and female spending overlapping? How can Walmart leverage this conclusion to make changes or improvements?

In [ ]:

```
# Check if confidence intervals overlap
if confidence_interval_women[1] < confidence_interval_men[0] or confidence_interval_men[1
    print("The confidence intervals do not overlap.")
    print("Walmart can leverage this to tailor marketing and product offerings to both ge
else:
    print("The confidence intervals overlap.")
    print("There may not be a significant difference between male and female spending hab
```

```
The confidence intervals do not overlap.
Walmart can leverage this to tailor marketing and product offerings to bot
h genders differently.
```

Observation:

- The **confidence intervals for the average spending of male and female customers do not overlap**. This indicates a significant difference in spending habits between these two gender groups. Specifically:

- **Female customers** have a statistically different average spending compared to male customers.
- **Male customers**, on average, spend either more or less than female customers, but the difference is significant enough to conclude that there's a distinction in spending patterns.

Q4. Results when the same activity is performed for Married vs Unmarried

In [ ]:

```
# Average Purchase Amount for Married and Unmarried Customers
```

In [ ]:

```
# Calculate the average purchase amount for married and unmarried customers
avg_purchase_married = df[df['Marital_Status'] == 1]['Purchase'].mean()
avg_purchase_unmarried = df[df['Marital_Status'] == 0]['Purchase'].mean()

# Print the average purchase amounts
print("Average purchase amount for married customers:", avg_purchase_married)
print("Average purchase amount for unmarried customers:", avg_purchase_unmarried)
```

```
Average purchase amount for married customers: 9261.174574082374
Average purchase amount for unmarried customers: 9265.907618921507
```

In [ ]:

```
# Confidence Intervals for Married and Unmarried Spending Means
```

In [ ]:

```
# Calculate confidence intervals for married and unmarried spending means
confidence_interval_married = stats.norm.interval(0.95, loc=df[df['Marital_Status'] == 1]
                                                  scale=stats.sem(df[df['Marital_Status']
confidence_interval_unmarried = stats.norm.interval(0.95, loc=df[df['Marital_Status'] ==
                                                    scale=stats.sem(df[df['Marital_Status'

# Print confidence intervals
print("Confidence interval for married spending mean:", confidence_interval_married)
print("Confidence interval for unmarried spending mean:", confidence_interval_unmarried)
```

```
Confidence interval for married spending mean: (9240.460427057078, 9281.88
8721107669)
Confidence interval for unmarried spending mean: (9248.61641818668, 9283.1
98819656332)
```

In [ ]:

```
# Hypothesis Test (t-test) for Married vs. Unmarried Spending
```

In [ ]:

```
# Perform a t-test to check if the difference is significant
t_stat_married_vs_unmarried, p_value_married_vs_unmarried = stats.ttest_ind(df[df['Marita
                                                                              df[df['Marital

# Print the p-value
print("P-value for married vs. unmarried spending:", p_value_married_vs_unmarried)

# Interpret the result
if p_value_married_vs_unmarried < 0.05:
    print("There is a significant difference in spending between married and unmarried cu
else:
    print("There is no significant difference in spending between married and unmarried c
```

```
P-value for married vs. unmarried spending: 0.7310947525758316
There is no significant difference in spending between married and unmarri
ed customers.
```

Observations:

- The **average purchase amount** for married customers and unmarried customers has been calculated.
- **Confidence intervals** for the mean spending of married and unmarried customers have been determined.
- A **Hypothesis Test (t-test)** was performed to check if there is a significant difference in spending between married and unmarried customers. The **p-value** was used to make this determination.

Q5. Results when the same activity is performed for Age

In [ ]:

```
# Average Purchase Amount for Different Age Groups
```

In [ ]:

```
# Calculate the average purchase amount for each age group
age_groups = df['Age'].unique()
avg_purchase_by_age = {}

for age_group in age_groups:
    avg_purchase_by_age[age_group] = df[df['Age'] == age_group]['Purchase'].mean()

# Print the average purchase amounts by age group
for age_group, avg_purchase in avg_purchase_by_age.items():
    print(f"Average purchase amount for {age_group} age group:", avg_purchase)
```

```
Average purchase amount for 0-17 age group: 8933.464640444974
Average purchase amount for 55+ age group: 9336.280459449405
Average purchase amount for 26-35 age group: 9252.690632869888
Average purchase amount for 46-50 age group: 9208.625697468327
Average purchase amount for 51-55 age group: 9534.808030960236
Average purchase amount for 36-45 age group: 9331.350694917874
Average purchase amount for 18-25 age group: 9169.663606261289
```

In [ ]:

```
# Confidence Intervals for Average Spending by Age Group
```

In [ ]:

```
# Calculate confidence intervals for average spending by age group
confidence_intervals_by_age = {}

for age_group in age_groups:
    confidence_interval = stats.norm.interval(0.95, loc=df[df['Age'] == age_group]['Purch
                                              scale=stats.sem(df[df['Age'] == age_group]['
    confidence_intervals_by_age[age_group] = confidence_interval

# Print confidence intervals for each age group
for age_group, confidence_interval in confidence_intervals_by_age.items():
    print(f"Confidence interval for {age_group} age group:", confidence_interval)
```

```
Confidence interval for 0-17 age group: (8851.947970542686, 9014.981310347
262)
Confidence interval for 55+ age group: (9269.29883441773, 9403.26208448107
9)
Confidence interval for 26-35 age group: (9231.733676400028, 9273.64758933
9747)
Confidence interval for 46-50 age group: (9163.085142648752, 9254.16625228
7903)
Confidence interval for 51-55 age group: (9483.991472776577, 9585.62458914
3894)
Confidence interval for 36-45 age group: (9301.669410965314, 9361.03197887
0433)
Confidence interval for 18-25 age group: (9138.407948753442, 9200.91926376
9136)
```

In [ ]:

```
# Hypothesis Test (ANOVA) for Age Groups
```

In [ ]:

```
# To check if there are significant differences in spending among different age groups, w
from scipy.stats import f_oneway

# Perform ANOVA test for spending by age group
age_group_data = [df[df['Age'] == age_group]['Purchase'] for age_group in age_groups]
f_statistic, p_value_age_groups = f_oneway(*age_group_data)

# Print the p-value
print("P-value for differences in spending by age group:", p_value_age_groups)

# Interpret the result
if p_value_age_groups < 0.05:
    print("There are significant differences in spending among different age groups.")
else:
    print("There are no significant differences in spending among different age groups.")
```

```
P-value for differences in spending by age group: 1.053563939251671e-49
There are significant differences in spending among different age groups.
```

Observation:

- The code provided calculates and prints the **average purchase amount for each age group**, **confidence intervals for average spending in each group**, and **performs an ANOVA test** to check for significant differences in spending among age groups.
- If the **p-value from the ANOVA test is less than 0.05**, it indicates significant differences in spending among different age groups. This information can be valuable for tailoring marketing and product strategies to better cater to the preferences of customers in each age group.

(4) Final Insights

- **Gender Spending:** There is a significant difference in spending between male and female customers, with males spending more on average.
- **Marital Status:** Marital status does not significantly impact spending in this dataset.
- **Age Groups:** Further analysis is needed to understand the spending patterns among different age groups, as this variable may have a significant influence on purchasing behavior.
- **City Categories:** It's worth exploring whether customers from different city categories exhibit different spending behaviors to tailor strategies accordingly.
- **Occupation:** Occupation might influence spending habits, and identifying preferences among different occupations could lead to targeted marketing.

(5) Final Recommendations (Actionable Items)

"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.

After having a complete analysis of the **Walmart** dataset, some of the key **actionable recommendations** are shown below:

- **Gender-Based Marketing:** Develop gender-specific marketing campaigns to target male and female customers effectively. Highlight products and promotions aligned with the preferences of each gender.
- **Product Assortment:** Optimize product offerings based on gender preferences. Stock and promote items that appeal to the distinct tastes of male and female customers.
- **Pricing Strategies:** Consider offering gender-specific pricing or discounts on products popular among each gender to attract and retain customers.
- **Store Layout and Experience:** Enhance the shopping experience by aligning store layout, product placement, and the overall shopping environment with customer preferences.
- **Age Group Analysis:** Conduct in-depth analysis of spending patterns among different age groups to identify potential opportunities for customization.
- **City Category Focus:** Investigate whether customers from different city categories have unique spending behaviors and tailor marketing strategies accordingly.
- **Occupation Insights:** Explore how occupation influences spending habits to cater to the preferences of different occupational groups.
- **Market-Specific Strategies:** Customize marketing and product strategies for different markets, considering regional variations in customer behavior.

- **Customer Feedback:** Gather feedback from customers to continuously refine strategies and offerings based on their evolving preferences.
- **Competitor Benchmarking:** Analyze competitor strategies and learn from successful approaches to adapt and improve Walmart's own offerings.

These recommendations provide actionable steps for **Walmart** to **enhance customer satisfaction** and **increase sales** while accommodating the diverse preferences and behaviors of its customer base.