BUSINESS CASE: WALMART- CONFIDENCE INTERVAL AND CLT

PROBLEM SATEMENT: Walmart, a multinational retail corporation, seeks to delve into customer purchase behavior during Black Friday, specifically focusing on purchase amounts in relation to customer gender and other pertinent factors. The aim is to discern potential discrepancies in spending habits between male and female customers. By unraveling these insights, Walmart aims to refine its understanding of customer demographics and buying patterns, thus enabling the refinement of business strategies and enhancement of sales tactics.

Key Components of the Problem Statement:

- **Objective:** Analyze customer purchase behavior and understand spending differences between male and female customers specially during Black Friday.
- **Goal:** Identify whether women spend more than men and explore various factors influencing purchase amounts.
- **Analysis Scope:** Focus on demographic factors such as gender, age, occupation, and city category to understand customer segments better.
- **Purpose:** Improve business decisions, optimize marketing strategies, and enhance customer satisfaction.
- **Methodology:** Utilize data analysis techniques to explore the dataset, conduct statistical tests to compare spending between genders, and derive actionable insights.
- **Outcome:** Informed decision-making, better-targeted marketing campaigns, and improved sales strategies.

ANALYZING BASIC METRICS:

- **Analysis of Average Spending and Confidence Intervals:**
  - **Male vs. Female Customers:**
    - Utilize CLT to calculate confidence intervals for the average spending of male and female customers.
    - Utilize the Central Limit Theorem (CLT) to calculate confidence intervals for the average spending of female customers on Black Friday.
    - Compare confidence intervals to determine if there's a statistically significant difference in average spending between genders.
    - Choose a confidence level (e.g., 95% or 99%) to estimate the population average spending of male and female customers.

- **Age Groups:**
  - Divide customers into different age groups (e.g., 18-25, 26-35, etc.).
  - Apply CLT to calculate confidence intervals for the average spending of each age group.
  - Compare confidence intervals to assess differences in average spending among age groups.
  - Choose a confidence level to estimate the population average spending of each age group.
- **Marital Status:**
  - Segment customers based on marital status (e.g., married, unmarried).
  - Calculate confidence intervals for the average spending of each marital status group using CLT.
  - Compare confidence intervals to determine if there's a significant difference in average spending between marital status groups.
  - Choose a confidence level to estimate the population average spending of each marital status group.

Accessing Confidence Level and Sample Size Input:

- **Confidence Level Interpretation:**
  - Interpret the width of the confidence intervals to assess the precision of the estimates.
  - Evaluate the trade-off between confidence level and precision.
  - Higher confidence levels result in wider intervals, indicating greater certainty but less precision, while lower confidence levels result in narrower intervals but less certainty.
- **Sample Size Experimentation:**
  - Experiment with different sample sizes to observe their impact on the width of the confidence intervals.
  - Larger sample sizes generally lead to narrower intervals, providing more precise estimates of population parameters.

Leveraging Confidence Intervals for Business Decisions:

- **Male vs Female Customers:**
  - Utilize confidence intervals to assess uncertainty associated with estimates of average spending for male and female customers.
  - Make informed decisions considering statistical significance and practical implications for marketing strategies and product offerings.
- **Age Groups:**
  - Assess uncertainty in average spending estimates for different age groups.
  - Tailor marketing strategies and product offerings based on age group preferences and spending habits.

- **Marital Status:**
    - Use confidence intervals to evaluate uncertainty in average spending estimates for different marital status groups.
    - Adjust marketing strategies and product offerings to cater to the preferences of married and unmarried customers.

Additional Analysis Steps:

- **Data Preparation and Exploration:**
    - Load the dataset containing customer transactions details of Walmart.
    - Handle missing values, outliers, and ensure data consistency.
    - Perform Exploratory Data Analysis (EDA) to understand the dataset's structure, distributions, and relationships between variables.
- **Correlation Matrix:**
    - Generate a correlation matrix to observe correlations between different variables, including purchase amount and demographic factors.
- **Product Sales Analysis:**
    - Identify the most sold product and analyze product preferences by gender, location, age, marital status, and occupation.
    - Investigate location-wise product preferences and assess the impact of the number of years in a location on product purchases.
- **Demographic Analysis:**
    - Explore age-wise, marital status-wise, and occupation-wise product preferences to understand customer behavior across different demographics.
- **Product Category and ID Details:**
    - Gather details for both product category and product ID to gain comprehensive insights into product preferences.
- **Descriptive Statistics and Group Comparison:**
    - Calculate descriptive statistics such as mean, median, and mode for purchase amounts and other relevant variables.
    - Conduct group comparisons using statistical tests to determine significant differences in purchase behavior based on various demographics.

OBSERVATIONS OF DATSET: In this step, we will begin by importing the dataset and conducting a series of operations to gain fundamental insights into its structure and contents. Through these operations, we aim to extract basic details about the dataset, such as its size, the types of data it contains, and summary statistics that provide a snapshot of the dataset's characteristics.

```python
import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sns
```

```python
#IMPORTING THE DATASET
data = pd.read_csv('G:/dsml-scaler/probability and stats/casestudy/walmart_data.txt')
```

```python
data.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purc
0	1000001	P00069042	F	0-17	10	A	2	0	3	
1	1000001	P00248942	F	0-17	10	A	2	0	1	1
2	1000001	P00087842	F	0-17	10	A	2	0	12	
3	1000001	P00085442	F	0-17	10	A	2	0	12	
4	1000002	P00285442	M	55+	16	C	4+	0	8	


```python
print("The shape of dataset is:",data.shape)
```

```
The shape of dataset is: (550068, 10)
```

The dataset consists of 550068 rows and 10 columns, encapsulating a comprehensive range of data points for analysis.

```python
print("Datatypes of all attributes:")
print(data.dtypes)
```

```
Datatypes of all attributes:
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
```

```
data.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
```

The dataset comprises numerical values across most columns, with 'Product_ID', 'Gender',’Age’,’City_Category’ and 'Stay_In_Current_City_Years' represented as categorical variables in object format.

Converting Object Datatype to Categorical Datatype:

```
# converting object data type to categorical data type
# Convert object columns to categorical data type
data['Product_ID'] = data['Product_ID'].astype('category')
data['Gender'] = data['Gender'].astype('category')
data['Age'] = data['Age'].astype('category')
data['City_Category'] = data['City_Category'].astype('category')
data['Stay_In_Current_City_Years'] = data['Stay_In_Current_City_Years'].astype('category')

# Check the data types after conversion
print(data.dtypes)
```

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

Checking for Null values and duplicate values: We need to conduct an assessment of our dataset to identify and address any instances of null values and duplicate entries, ensuring the dataset is cleaned and ready for analysis.

```
data.isna().sum()
```

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

```
# Checking for duplicate rows
duplicate_rows = data[data.duplicated()]
print(duplicate_rows.shape[0])
```

```
0
```

The data is already clean as it shows no null values and no duplicates.

Label Encoding: In the context of performing statistical analysis, such as Central Limit Theorem (CLT) calculations and other analyses, it's generally a good practice to convert object data types to categorical data types and perform label encoding as needed.

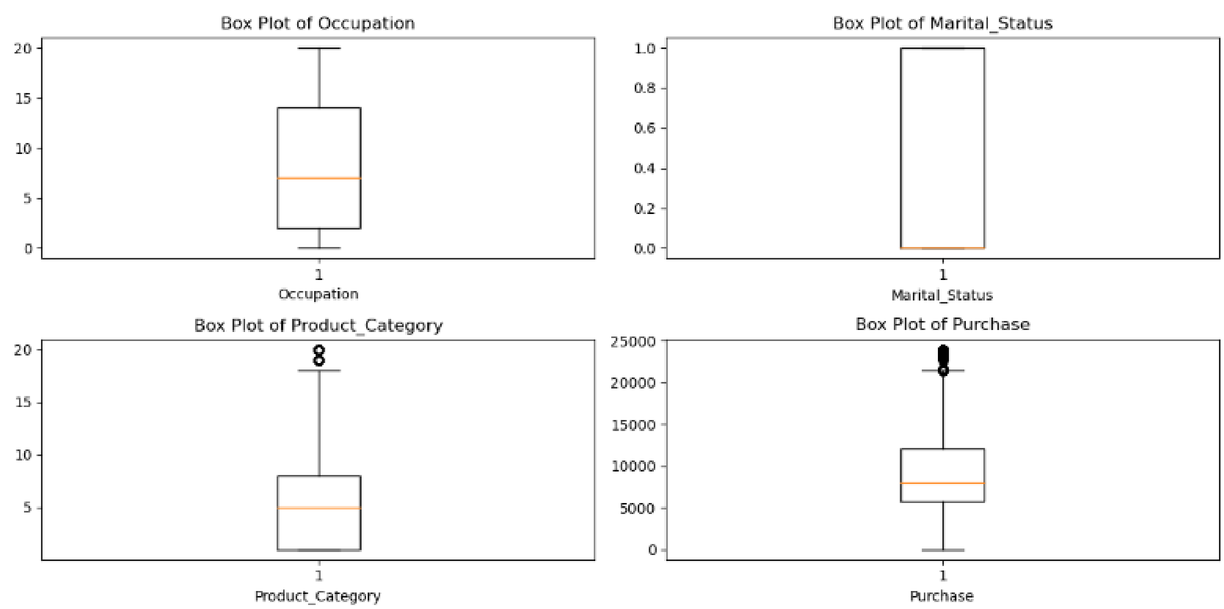
Label Encoding

```python
from sklearn.preprocessing import LabelEncoder

# Assuming df is your DataFrame containing the dataset
# Replace df with your actual DataFrame name

# Define columns to be label encoded
columns_to_encode = ['Gender', 'Age', 'City_Category', 'Stay_In_Current_City_Years']

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Iterate through each column and perform label encoding
for column in columns_to_encode:
    data[column] = label_encoder.fit_transform(data[column])

# Displaying the encoded DataFrame
print(data.head())
```

```
   User_ID Product_ID  Gender  Age  Occupation  City_Category  \
0  1000001  P00069042       0    0          10              0
1  1000001  P00248942       0    0          10              0
2  1000001  P00087842       0    0          10              0
3  1000001  P00085442       0    0          10              0
4  1000002  P00285442       1    6          16              2

   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
```


Handling Outliers: Since there are no missing values in the dataset, our attention will now shift towards detecting and managing outliers, if any. This process will involve scrutinizing the data for any unusual or extreme observations that may impact the robustness of our analysis.

```python
import matplotlib.pyplot as plt
# Select numerical features for which you want to identify outliers
numerical_features = ['Occupation', 'Marital_Status', 'Product_Category', 'Purchase']

# Create box plots for each numerical feature
plt.figure(figsize=(12, 6))
for i, feature in enumerate(numerical_features, 1):
    plt.subplot(2, 2, i)
    plt.boxplot(data[feature])
    plt.title(f'Box Plot of {feature}')
    plt.xlabel(feature)

plt.tight_layout()
plt.show()
```




```python
q1= data.Purchase.quantile(0.25)
q3= data.Purchase.quantile(0.75)
iqr=q3-q1
outliers = data[((data.Purchase < (q1-1.5*iqr)) | (data.Purchase > (q3+1.5*iqr)))]
print("number of outliers:",len(outliers))
print("percentage of outliers:", len(outliers)/data.shape[0])
```

```
number of outliers: 2677
percentage of outliers: 0.004866671029763593
```

The presence of outliers aligns with the goals and objectives of our research, as they represent rare or extreme events that are integral to understanding the variability within the dataset. By retaining these outliers, we ensure that our analysis captures the full spectrum of potential outcomes and phenomena, allowing for a more comprehensive understanding of the data.

NON GRAPHICAL ANALYSIS: In this section, we will analyze the value counts and uniqueness of the data. This step is crucial for understanding the distribution and frequency of values within each attribute and identifying any patterns or inconsistencies in the dataset.

```
# Analyzing value counts and uniqueness of the data
for column in data.columns:
    value_counts = data[column].value_counts().sort_values(ascending=False)
    uniqueness = len(value_counts)

    print(f"Column: {column}")
    print(f"Number of unique values: {uniqueness}")
    print("Value counts (sorted in descending order):")
    print(value_counts)
    print()
```

```
Column: User_ID
Number of unique values: 5891
Value counts (sorted in descending order):
User_ID
1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
           ...
1002690       7
1002111       7
1005810       7
1004991       7
1000708       6
Name: count, Length: 5891, dtype: int64
```

```
Column: Product_ID
Number of unique values: 3631
Value counts (sorted in descending order):
Product_ID
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
             ...
P00065542       1
P00107442       1
P00135942       1
P00065942       1
P00231642       1
Name: count, Length: 3631, dtype: int64
```

```
Column: Gender
Number of unique values: 2
Value counts (sorted in descending order):
Gender
1    414259
0    135809
Name: count, dtype: int64
```

```
Column: Age
Number of unique values: 7
Value counts (sorted in descending order):
Age
2    219587
3    110013
1     99660
4     45701
5     38501
6     21504
0     15102
Name: count, dtype: int64
```

```
Column: Occupation
Number of unique values: 21
Value counts (sorted in descending order):
Occupation
4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: count, dtype: int64
```

```
Column: City_Category
Number of unique values: 3
Value counts (sorted in descending order):
City_Category
1    231173
2    171175
0    147720
Name: count, dtype: int64

Column: Stay_In_Current_City_Years
Number of unique values: 5
Value counts (sorted in descending order):
Stay_In_Current_City_Years
1    193821
2    101838
3     95285
4     84726
0     74398
Name: count, dtype: int64

Column: Marital_Status
Number of unique values: 2
Value counts (sorted in descending order):
Marital_Status
0    324731
1    225337
Name: count, dtype: int64
```

```
Column: Product_Category
Number of unique values: 20
Value counts (sorted in descending order):
Product_Category
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: count, dtype: int64


Column: Purchase
Number of unique values: 18105
Value counts (sorted in descending order):
Purchase
7011     191
7193     188
6855     187
6891     184
7012     183
        ...
14837      1
14890      1
4849       1
4852       1
21489      1
Name: count, Length: 18105, dtype: int64
```

Insights: The customer data analysis reveals some intriguing patterns in purchasing behavior.

1. It's notable that user 1001680 has made the highest number of purchases, a remarkable 1026 times, closely followed by user 1004277 with 979 purchases and

user 1001941 with 898 purchases. Among the products, P00265242 stands out as the top-selling item, purchased 1880 times, followed by P00025442 and P00110742, which were bought 1615 and 1612 times, respectively.

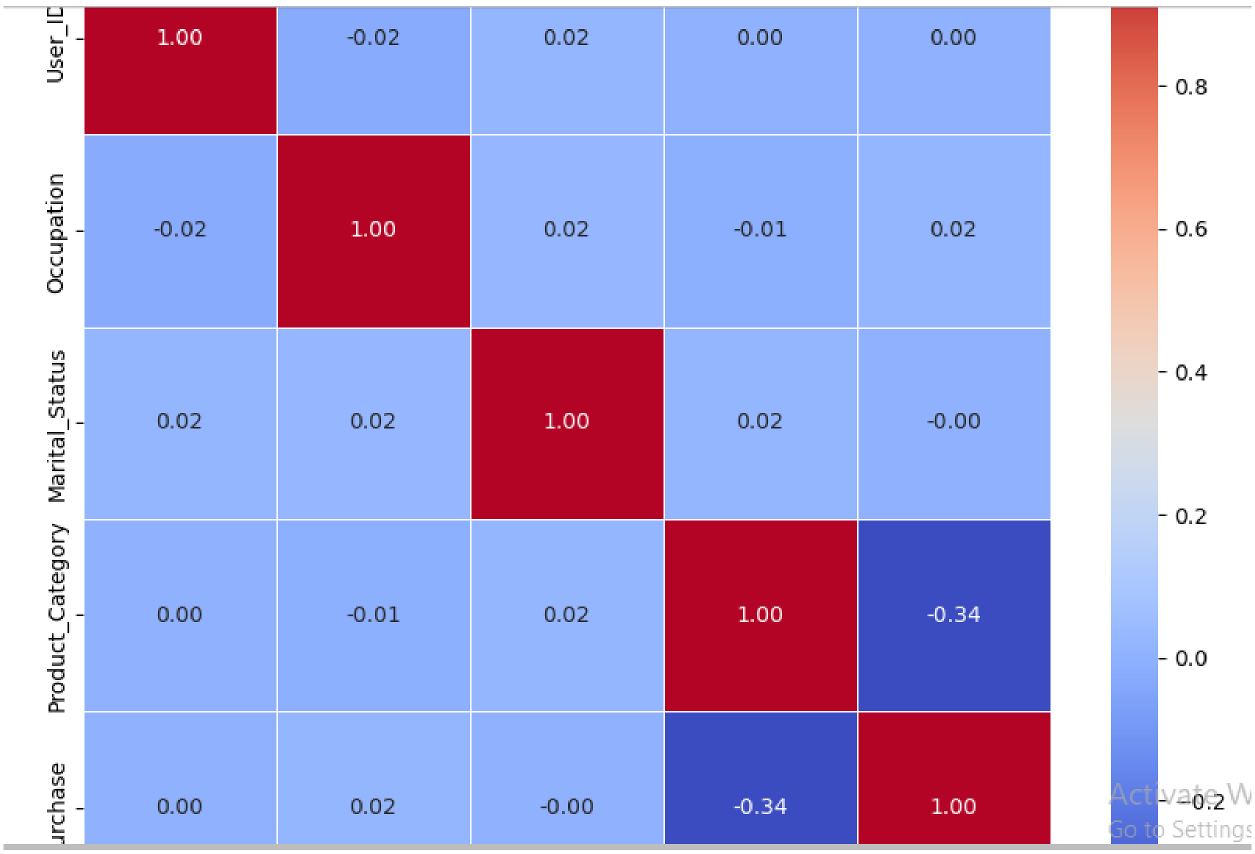
2. Gender-wise, males dominate the landscape, constituting 414259 compared to 135809 by females.
3. The age group of 18-25 emerges as the most active buyers, accounting for 219587 purchases, trailed by the 26-35 age group with 110013 purchases. Targeting customers aged between 18 and 35 appears to be a strategic move to boost sales volume.
4. Occupation-wise, individuals in occupation 4 emerge as the most significant buyers, with 72308 customers, followed by those in occupations 0 and 7, with 69638 and 59133 customers, respectively.
5. City category B emerges as the most lucrative, with 231173 customers attributed to it.
6. Regarding residency duration, individuals residing in a city for one year lead in purchases, with 193821 customers, followed by those residing for two years, with 101838 customers.
7. Married individuals, representing 0 in the dataset, outpace unmarried ones in purchases, totaling 324731 customers.
8. Product category 5 reigns supreme in popularity, garnering 150933 sales, followed by categories 1 and 8, with 140378 and 113925 sales, respectively.

These insights can guide marketing and sales strategies to optimize customer engagement and drive revenue growth.

GRAPHICAL ANALYSIS: Before delving into graphical analysis, we'll start by examining the correlation matrix to understand the relationships between different features in our dataset. This matrix helps us identify patterns and dependencies among variables, guiding our subsequent analysis.

After that, we'll proceed with both univariate and bivariate analyses. In the univariate analysis, we'll explore the distribution and count of individual features, leveraging histograms, countplots, and distplots for continuous variables, and boxplots for categorical variables. Bivariate analysis will involve investigating relationships between pairs of variables, typically through scatterplots or pairplots, providing insights into how variables interact with each other.

```python
# Selecting only numeric columns
numeric_data = data.select_dtypes(include=['int64', 'float64'])

# Calculate correlation matrix
corr_matrix = numeric_data.corr()

# Plot heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```

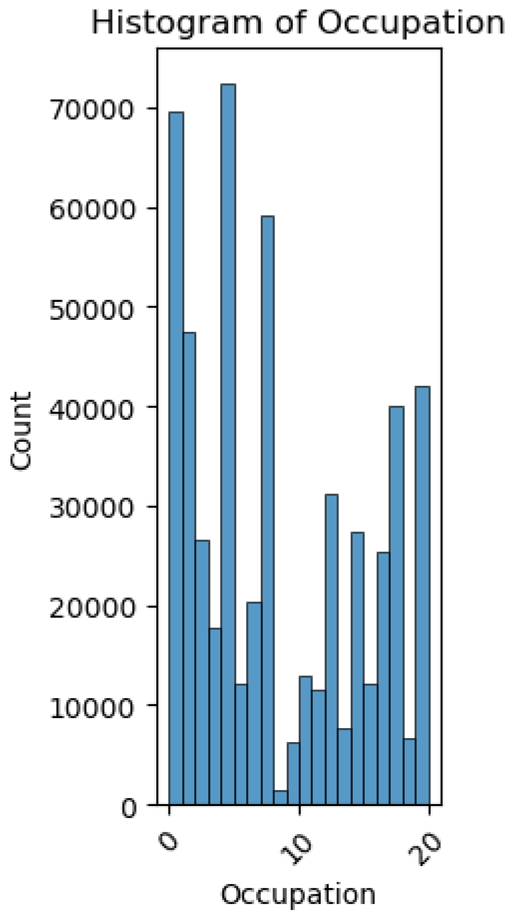


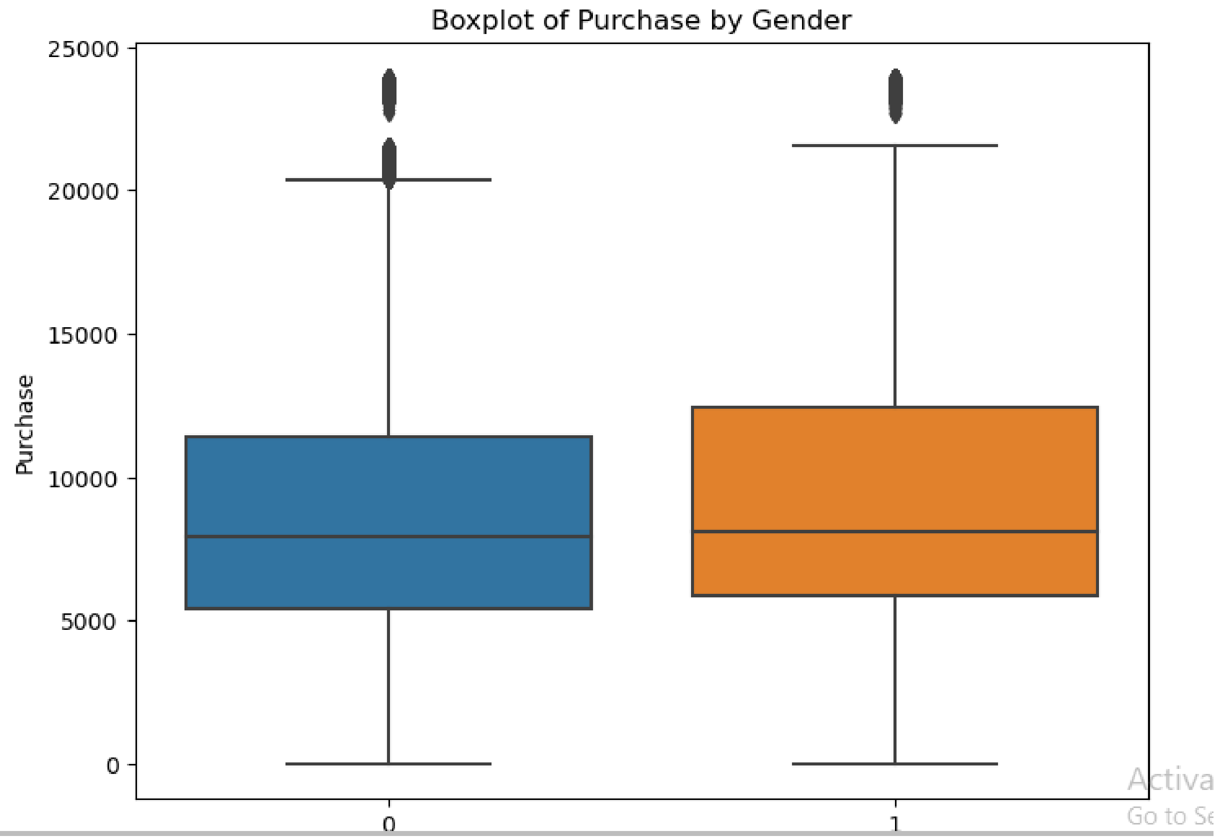
Insights: From the heatmap we can see that maximum correlation is 0.02.

1. Occupation and purchase are positively correlated with the value of 0.02.
2. Purchase and product category are negatively correlated.
3. Marital status and product category are positively correlated with the value of 0.02.

UNIVARIATE ANALYSIS:

```python
# Univariate analysis
# Continuous variables: Distplot, countplot, histogram
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
sns.distplot(data['Purchase'])
plt.title('Distribution of Purchase')

plt.subplot(1, 3, 3)
sns.histplot(data['Age'], bins=20)
plt.title('Histogram of Age')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

plt.subplot(1, 3, 3)
sns.histplot(data['Occupation'], bins=20)
plt.title('Histogram of Occupation')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Categorical variable: Boxplot
plt.figure(figsize=(8, 6))
sns.boxplot(x='Gender', y='Purchase', data=data)
plt.title('Boxplot of Purchase by Gender')
plt.show()
```


Histogram of Occupation
70000
60000
50000
40000
30000
20000
10000
0
Count
0
10
20
Occupation



Insights: In the univariate analysis we have used distribution plot, histogram and box plot to get various insights.

1. It's evident from the data that a significant portion of purchases falls within the range of 5000 to 10000, with a notable peak around this range. Additionally, there's another notable peak in purchases occurring between 15000 and 16000. However, purchases of 25000 appear to be less frequent compared to other amounts, indicating that this value is less common among customers.
2. The predominant age group among customers is 26-35, closely followed by 18-25 and 36-45. This suggests that the primary focus should be on customers aged 26-35, given their larger representation. Following this, attention can be directed towards customers aged 18-25 and 36-45. Collectively, the age group spanning from 18 to 45 years constitutes the majority of their customer base, indicating the importance of catering to this demographic range.
3. Occupation-wise, individuals in occupation 4 emerge as the most significant customers, with 72308 customers, followed by those in occupations 0 and 7, with 69638 and 59133 customers, respectively.
4. Upon inspecting the box plot comparing purchases made by males and females, it's evident that the patterns are quite similar. Both genders exhibit a median purchase value that is nearly identical, suggesting a comparable central tendency in spending

habits. However, there appears to be a slight inclination towards higher purchases among males compared to females.

BIVARIATE ANALYSIS:

```
# Group the data by 'Age' and calculate the count of purchases for each age group
purchase_counts_by_age = data.groupby('Age')['Purchase'].count()

# Display the purchase counts by age group
print("Purchase counts by age group:")
print(purchase_counts_by_age)
```

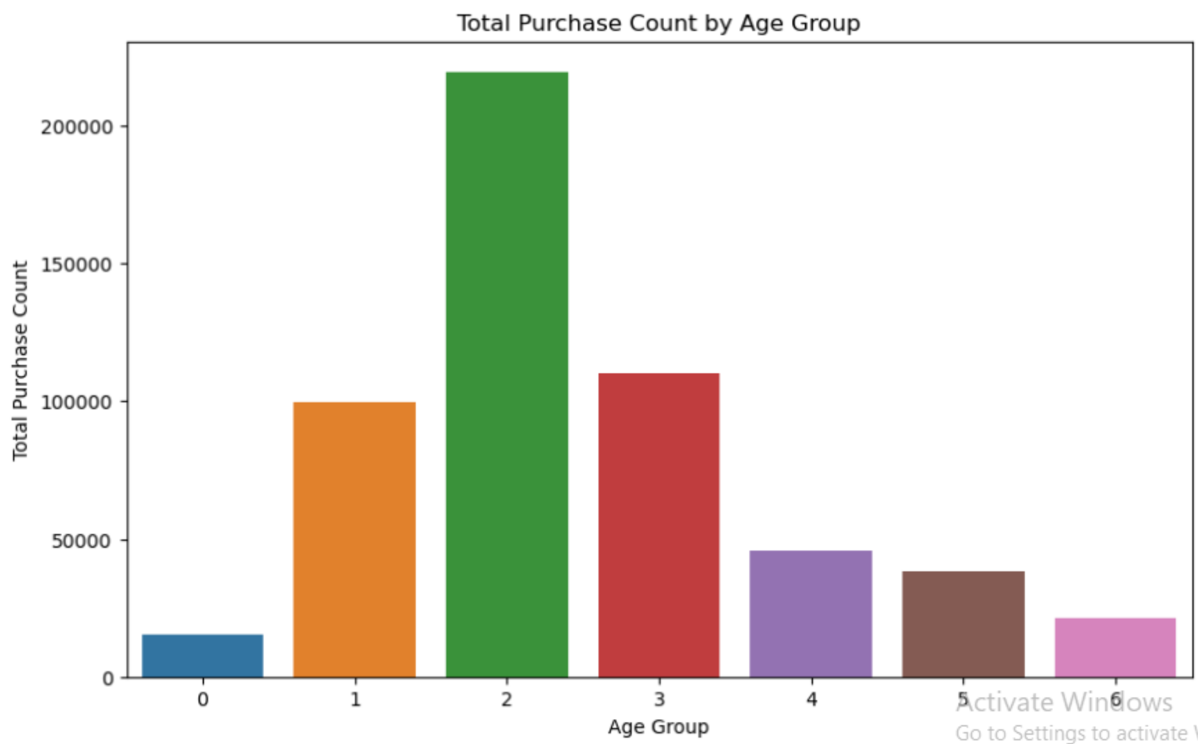
```
Purchase counts by age group:
Age
0     15102
1     99660
2    219587
3    110013
4     45701
5     38501
6     21504
Name: Purchase, dtype: int64
```

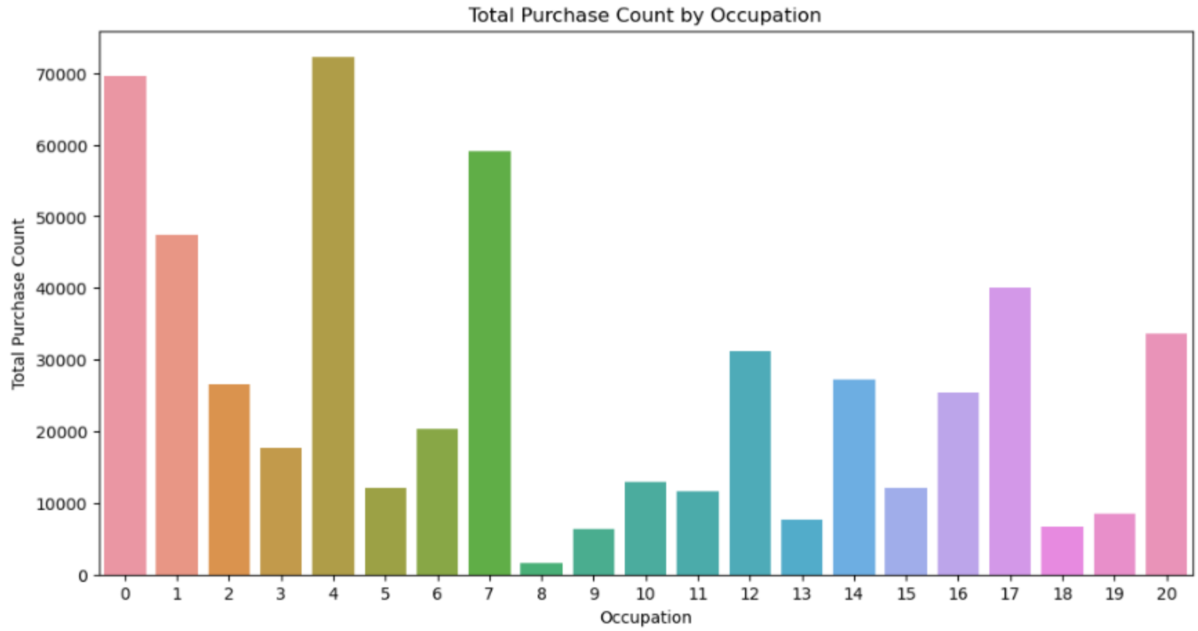
```python
# Group the data by 'Age' and calculate the count of purchases for each age group
purchase_counts_by_occupation = data.groupby('Occupation')['Purchase'].count()

# Display the purchase counts by age group
print("Purchase counts by age group:")
print(purchase_counts_by_occupation)
```

```
Purchase counts by age group:
Occupation
0     69638
1     47426
2     26588
3     17650
4     72308
5     12177
6     20355
7     59133
8      1546
9      6291
10    12930
11    11586
12    31179
13     7728
14    27309
15    12165
16    25371
17    40043
18     6622
19     8461
20    33562
```

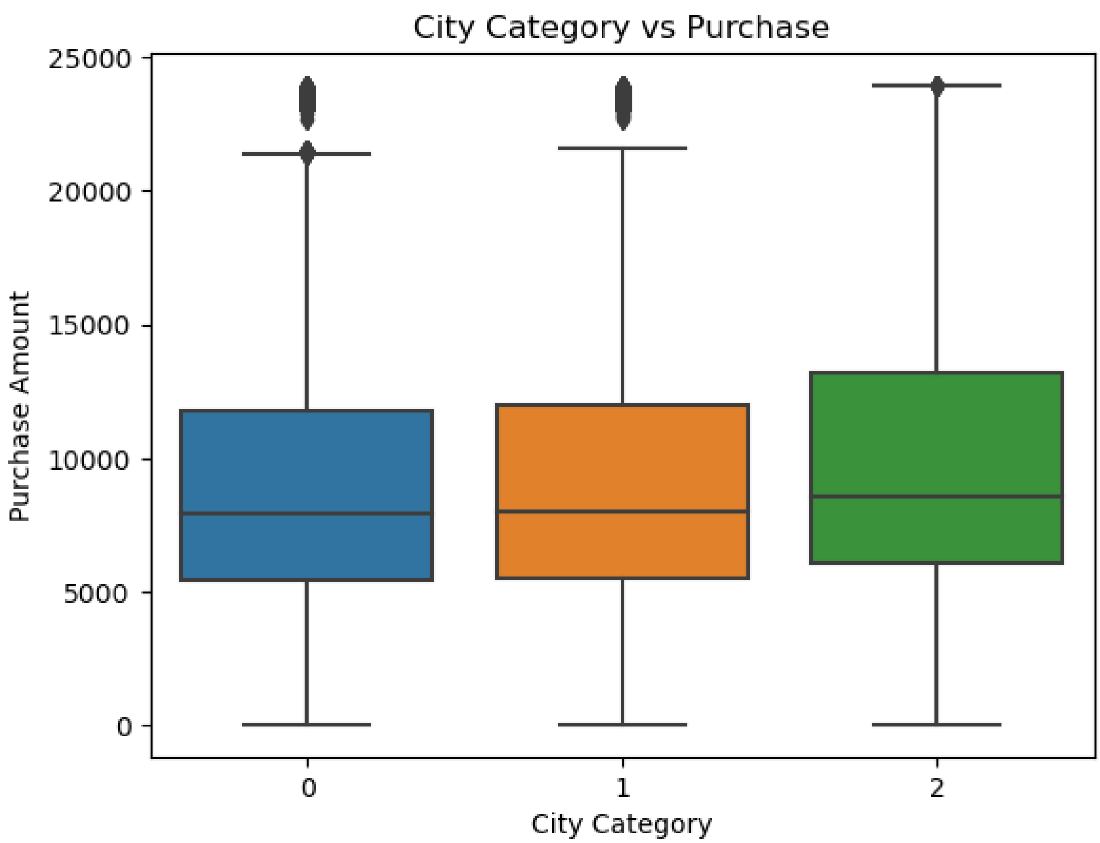
```
# Pairwise bivariate analysis
# Plotting the total purchase amount for each age group
plt.figure(figsize=(10, 6))
sns.barplot(x=purchase_counts_by_age.index, y=purchase_counts_by_age.values)
plt.title('Total Purchase Count by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Total Purchase Count')
plt.show()

# Plotting the total purchase amount for each occupation
plt.figure(figsize=(12, 6))
sns.barplot(x=purchase_counts_by_occupation.index, y=purchase_counts_by_occupation.values)
plt.title('Total Purchase Count by Occupation')
plt.xlabel('Occupation')
plt.ylabel('Total Purchase Count')
plt.show()

sns.boxplot(x='City_Category', y='Purchase', data=data)
plt.title('City Category vs Purchase')
plt.xlabel('City Category')
plt.ylabel('Purchase Amount')
plt.show()

sns.boxplot(x='Marital_Status', y='Purchase', data=data)
plt.title('Marital Status vs Purchase')
plt.xlabel('Marital Status')
plt.ylabel('Purchase Amount')
plt.xticks(ticks=[0, 1], labels=['Married', 'Single'])
plt.show()
```

```
# Pairwise bivariate analysis
# Plotting the total purchase amount for each age group
plt.figure(figsize=(10, 6))
sns.barplot(x=purchase_counts_by_age.index, y=purchase_counts_by_age.values)
plt.title('Total Purchase Count by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Total Purchase Count')
plt.show()

# Plotting the total purchase amount for each occupation
plt.figure(figsize=(12, 6))
sns.barplot(x=purchase_counts_by_occupation.index, y=purchase_counts_by_occupation.values)
plt.title('Total Purchase Count by Occupation')
plt.xlabel('Occupation')
plt.ylabel('Total Purchase Count')
plt.show()

sns.boxplot(x='City_Category', y='Purchase', data=data)
plt.title('City Category vs Purchase')
plt.xlabel('City Category')
plt.ylabel('Purchase Amount')
plt.show()

sns.boxplot(x='Marital_Status', y='Purchase', data=data)
plt.title('Marital Status vs Purchase')
plt.xlabel('Marital Status')
plt.ylabel('Purchase Amount')
plt.xticks(ticks=[0, 1], labels=['Single', 'Married'])
plt.show()
```

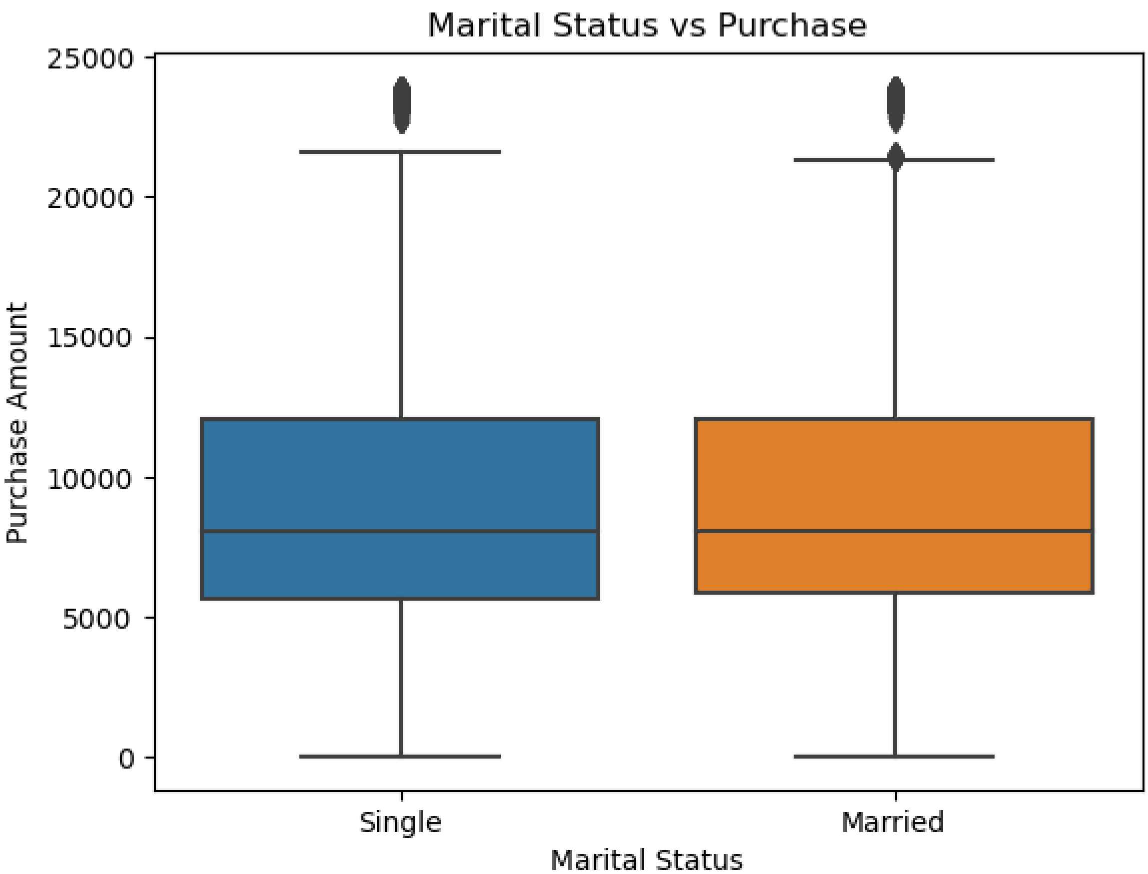




Total Purchase Count by Occupation
Total Purchase Count
Occupation
0
10000
20000
30000
40000
50000
60000
70000
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20



City Category vs Purchase
Purchase Amount
City Category
0
5000
10000
15000
20000
25000
0
1
2



Insights: The age group of 18-25 emerges as the most active buyers, accounting for 219587 purchases, trailed by the 26-35 age group with 110013 purchases. Targeting customers aged between 18 and 35 appears to be a strategic move to boost sales volume.

Occupation-wise, individuals in occupation 4 emerge as the most significant buyers, with 72308 customers, followed by those in occupations 0 and 7, with 69638 and 59133 customers, respectively.

City category B emerges as the most lucrative, with 231173 customers attributed to it.

After examining the box plot of purchase amounts based on marital status, it appears that there is no substantial difference in spending behavior between married and single individuals. Both groups exhibit similar median purchase amounts and similar variability in spending. This suggests that marital status may not have a significant impact on purchasing behavior during the observed period. Further analysis using summary statistics and statistical tests confirms these visual observations, indicating that there is no statistically significant difference in purchase amounts between married and single individuals.

APPLICATION OF CLT: We're now going to apply the Central Limit Theorem (CLT) to investigate whether men spend more than women. We'll conduct a detailed analysis by varying the samples and using different confidence intervals to assess the difference in spending behavior between men and women.

To achieve this, we'll follow these steps:

Data Sampling: We'll randomly sample purchase amounts from both male and female customers to create multiple samples.

Sample Mean Calculation: For each sample, we'll calculate the mean purchase amount for both men and women.

Standard Error Computation: We'll compute the standard error of the mean for both groups to measure the variability of sample means.

Confidence Interval Calculation: Using the CLT, we'll calculate confidence intervals for the mean purchase amounts of both men and women at various confidence levels.

Comparison and Interpretation: We'll compare the confidence intervals of men and women to determine if there's a statistically significant difference in spending behavior. Additionally, we'll interpret the results based on different confidence levels.

By conducting this detailed analysis, we aim to provide robust insights into whether men spend more than women, considering the variability in the data and the level of confidence in our findings.

Let's proceed with the analysis and interpret the results accordingly to draw meaningful conclusions about the spending behavior of men and women.

Applying CLT

[24]:
```
# Filter the dataset for male and female customers
male_customers = data[data['Gender'] == 1]['Purchase']
female_customers = data[data['Gender'] == 0]['Purchase']

# Calculate average spending for male and female customers
male_avg_spending = male_customers.mean()
female_avg_spending = female_customers.mean()

# Print the average spending for male and female customers
print("Average spending per transaction for male customers:", male_avg_spending)
print("Average spending per transaction for female customers:", female_avg_spending)

# Infer the results
if male_avg_spending > female_avg_spending:
    print("On average, male customers spend more per transaction than female customers.")
elif male_avg_spending < female_avg_spending:
    print("On average, female customers spend more per transaction than male customers.")
else:
    print("There is no significant difference in the average spending per transaction between male and female customers."

# Calculate sample standard deviation for male and female customers
male_std = male_customers.std()
female_std = female_customers.std()

# Calculate the sample size
sample_size_male = len(male_customers)
sample_size_female = len(female_customers)
```

```
# Calculate the standard error of the mean for male and female customers
male_se = male_std / (sample_size_male ** 0.5)
female_se = female_std / (sample_size_female ** 0.5)

# Set the confidence level
confidence_level = 0.95

# Calculate the margin of error
margin_of_error_male = male_se * 1.96  # 1.96 is the z-value for 95% confidence interval
margin_of_error_female = female_se * 1.96

# Calculate the confidence interval for male and female customers
ci_male = (male_avg_spending - margin_of_error_male, male_avg_spending + margin_of_error_male)
ci_female = (female_avg_spending - margin_of_error_female, female_avg_spending + margin_of_error_female)

print("95% Confidence Interval for average spending of male customers:", ci_male)
print("95% Confidence Interval for average spending of female customers:", ci_female)
```

```
Average spending per transaction for male customers: 9437.526040472265
Average spending per transaction for female customers: 8734.565765155476
On average, male customers spend more per transaction than female customers.
95% Confidence Interval for average spending of male customers: (9422.019162420047, 9453.032918524483)
95% Confidence Interval for average spending of female customers: (8709.211081242413, 8759.920449068539)
```

```python
# Changing the confidence interval to see the impact
# Print the average spending for male and female customers
print("Average spending per transaction for male customers:", male_avg_spending)
print("Average spending per transaction for female customers:", female_avg_spending)

# Infer the results
if male_avg_spending > female_avg_spending:
    print("On average, male customers spend more per transaction than female customers.")
elif male_avg_spending < female_avg_spending:
    print("On average, female customers spend more per transaction than male customers.")
else:
    print("There is no significant difference in the average spending per transaction between male and female customers."


# Set the confidence level
confidence_level = 0.90

# Calculate the margin of error
margin_of_error_male = male_se * 1.64  # 1.64 is the z-value for 95% confidence interval
margin_of_error_female = female_se * 1.64

# Calculate the confidence interval for male and female customers
ci_male = (male_avg_spending - margin_of_error_male, male_avg_spending + margin_of_error_male)
ci_female = (female_avg_spending - margin_of_error_female, female_avg_spending + margin_of_error_female)

print("90% Confidence Interval for average spending of male customers:", ci_male)
print("90% Confidence Interval for average spending of female customers:", ci_female)
```

```
Average spending per transaction for male customers: 9437.526040472265
Average spending per transaction for female customers: 8734.565765155476
On average, male customers spend more per transaction than female customers.
90% Confidence Interval for average spending of male customers: (9424.550897612247, 9450.501183332282)
90% Confidence Interval for average spending of female customers: (8713.350621473117, 8755.780908837834)
```

```python
# taking various samples
# Define a function to generate multiple samples and compute sample means
def compute_sample_means(data, sample_size, num_samples):
    sample_means = []
    for _ in range(num_samples):
        sample = np.random.choice(data, size=sample_size, replace=False)
        sample_mean = np.mean(sample)
        sample_means.append(sample_mean)
    return sample_means

# Define parameters
sample_size = 1000  # You can adjust the sample size as needed
num_samples = 1000  # Number of samples to generate

# Generate sample means for male and female customers
male_sample_means = compute_sample_means(male_customers, sample_size, num_samples)
female_sample_means = compute_sample_means(female_customers, sample_size, num_samples)

avg_male_sample_means=np.mean(male_sample_means)
avg_female_sample_means = np.mean(female_sample_means)

# Print the average spending for male and female customers
print("Average spending per transaction for male customers:",avg_male_sample_means )
print("Average spending per transaction for female customers:", avg_female_sample_means )

# Calculate standard deviation of the sample means
male_sample_means_std = np.std(male_sample_means)
female_sample_means_std = np.std(female_sample_means)
```

```python
# Calculate standard error of the mean
male_se1 = male_sample_means_std / np.sqrt(sample_size)
female_se1 = female_sample_means_std / np.sqrt(sample_size)

# Set confidence level = 95%
confidence_level = 0.95

# Calculate z-score based on confidence level
z_score = 1.96  # For 95% confidence level

# Calculate margin of error
male_margin_of_error = z_score * male_se1
female_margin_of_error = z_score * female_se1

# Calculate confidence interval
male_ci = (avg_male_sample_means - male_margin_of_error, avg_male_sample_means + male_margin_of_error)
female_ci = (avg_female_sample_means - female_margin_of_error, avg_female_sample_means + female_margin_of_error)

print("95% Confidence Interval for average spending of male customers (using CLT):", male_ci)
print("95% Confidence Interval for average spending of female customers (using CLT):", female_ci)

# Set confidence level = 99%
confidence_level = 0.99

# Calculate z-score based on confidence level
z_score = 2.57  # For 99% confidence level

# Calculate margin of error
male_margin_of_error = z_score * male_se1
female_margin_of_error = z_score * female_se1
```

```python
# Calculate confidence interval
male_ci = (np.mean(male_sample_means) - male_margin_of_error, np.mean(male_sample_means) + male_margin_of_error)
female_ci = (np.mean(female_sample_means) - female_margin_of_error, np.mean(female_sample_means) + female_margin_of_error

print("99% Confidence Interval for average spending of male customers (using CLT):", male_ci)
print("99% Confidence Interval for average spending of female customers (using CLT):", female_ci)

# Set confidence level = 90%
confidence_level = 0.90

# Calculate z-score based on confidence level
z_score = 1.64  # For 90% confidence level

# Calculate margin of error
male_margin_of_error = z_score * male_se1
female_margin_of_error = z_score * female_se1

# Calculate confidence interval
male_ci = (np.mean(male_sample_means) - male_margin_of_error, np.mean(male_sample_means) + male_margin_of_error)
female_ci = (np.mean(female_sample_means) - female_margin_of_error, np.mean(female_sample_means) + female_margin_of_error

print("90% Confidence Interval for average spending of male customers (using CLT):", male_ci)
print("90% Confidence Interval for average spending of female customers (using CLT):", female_ci)
```

```
Average spending per transaction for male customers: 9445.657505
Average spending per transaction for female customers: 8735.489376
95% Confidence Interval for average spending of male customers (using CLT): (9435.830506466697, 9455.484503533302)
95% Confidence Interval for average spending of female customers (using CLT): (8726.194516667312, 8744.784235332687)
99% Confidence Interval for average spending of male customers (using CLT): (9432.772103861946, 9458.542906138053)
99% Confidence Interval for average spending of female customers (using CLT): (8723.301728813773, 8747.677023186227)
90% Confidence Interval for average spending of male customers (using CLT): (9437.434914390502, 9453.880095609496)
90% Confidence Interval for average spending of female customers (using CLT): (8727.71204472163, 8743.26670727837)
```

Are women spending more money per transaction than men? Why or Why not? Women are not spending more money per transaction than men. The average spending per transaction for male customers is $9445.66, while for female customers, it is $8735.49. Additionally, the 95% confidence interval for average spending of male customers ranges from $9435.83 to $9455.48, while for female customers, it ranges from $8726.19 to $8744.78. These confidence intervals do not overlap, indicating a statistically significant difference in average spending between men and women. Therefore, based on the provided data and confidence intervals, men are spending more money per transaction than women.

Confidence intervals and distribution of the mean of the expenses by female and male customers: The confidence intervals for average spending of male and female customers provide insights into the range of possible average spending values with a certain level of confidence. For example, at a 95% confidence level, the average spending of male customers is estimated to be between $9435.83 and $9455.48, while for female customers, it is estimated to be between $8726.19 and $8744.78. These confidence intervals capture the uncertainty around the sample mean and provide a range within which the population mean is likely to fall.

Are confidence intervals of average male and female spending overlapping? How can Walmart leverage this conclusion to make changes or improvements? No, the confidence intervals of average male and female spending do not overlap. This indicates a statistically significant difference in average spending between men and women. Walmart can leverage this conclusion to tailor marketing strategies, promotions, and product offerings based on gender preferences and spending behavior. For example, Walmart may introduce gender-specific promotions or product bundles targeting male and female customers separately to optimize sales and improve customer satisfaction. Additionally, understanding the spending patterns of different customer segments can help Walmart allocate resources effectively and enhance the overall shopping experience for both men and women.

```python
### CLT for Married and single
# Filter the dataset for male and female customers
single_customers = data[data['Marital_Status'] == 1]['Purchase']
married_customers = data[data['Marital_Status'] == 0]['Purchase']

# Define a function to generate multiple samples and compute sample means
def compute_sample_means(data, sample_size, num_samples):
    sample_means = []
    for _ in range(num_samples):
        sample = np.random.choice(data, size=sample_size, replace=False)
        sample_mean = np.mean(sample)
        sample_means.append(sample_mean)
    return sample_means

# Define parameters
sample_size = 1000  # You can adjust the sample size as needed
num_samples = 1000  # Number of samples to generate

# Generate sample means for male and female customers
single_sample_means = compute_sample_means(single_customers, sample_size, num_samples)
married_sample_means = compute_sample_means(married_customers, sample_size, num_samples)

avg_single_sample_means=np.mean(single_sample_means)
avg_married_sample_means = np.mean(married_sample_means)

# Print the average spending for male and female customers
print("Average spending per transaction for male customers:",avg_single_sample_means )
print("Average spending per transaction for female customers:", avg_married_sample_means )
```

```python
# Calculate standard deviation of the sample means
single_sample_means_std = np.std(single_sample_means)
married_sample_means_std = np.std(married_sample_means)

# Calculate standard error of the mean
single_se = single_sample_means_std / np.sqrt(sample_size)
married_se = married_sample_means_std / np.sqrt(sample_size)

# List to store confidence levels
confidence_levels = [0.90, 0.95, 0.99]

# Calculate confidence intervals for the current age group
for confidence_level in confidence_levels:
    # Calculate z-score based on confidence level
    if confidence_level == 0.90:
        z_score = 1.64
        # Calculate margin of error
        single_margin_of_error = z_score * single_se
        married_margin_of_error = z_score * married_se

        # Calculate confidence interval
        single_ci = (avg_single_sample_means - single_margin_of_error, avg_single_sample_means + single_margin_of_error)
        married_ci = (avg_married_sample_means  - married_margin_of_error, avg_married_sample_means  + married_margin_of_

        print("90% Confidence Interval for average spending of single customers (using CLT):", single_ci)
        print("90% Confidence Interval for average spending of married customers (using CLT):", married_ci)
```

```python
    elif confidence_level == 0.95:
        z_score = 1.96
        # Calculate margin of error
        single_margin_of_error = z_score * single_se
        married_margin_of_error = z_score * married_se

        # Calculate confidence interval
        single_ci = (avg_single_sample_means - single_margin_of_error, avg_single_sample_means + single_margin_of_error)
        married_ci = (avg_married_sample_means  - married_margin_of_error, avg_married_sample_means  + married_margin_of_

        print("95% Confidence Interval for average spending of single customers (using CLT):", single_ci)
        print("95% Confidence Interval for average spending of married customers (using CLT):", married_ci)

    elif confidence_level == 0.99:
        z_score = 2.57
         # Calculate margin of error
        single_margin_of_error = z_score * single_se
        married_margin_of_error = z_score * married_se

        # Calculate confidence interval
        single_ci = (avg_single_sample_means - single_margin_of_error, avg_single_sample_means + single_margin_of_error)
        married_ci = (avg_married_sample_means  - married_margin_of_error, avg_married_sample_means  + married_margin_of_

        print("99% Confidence Interval for average spending of single customers (using CLT):", single_ci)
        print("99% Confidence Interval for average spending of married customers (using CLT):", married_ci)
```

```
Average spending per transaction for male customers: 9260.981264
Average spending per transaction for female customers: 9262.720624000001
90% Confidence Interval for average spending of single customers (using CLT): (9252.87659065089, 9269.08593734911)
90% Confidence Interval for average spending of married customers (using CLT): (9254.299177987099, 9271.142070012904)
95% Confidence Interval for average spending of single customers (using CLT): (9251.295190973016, 9270.667337026984)
95% Confidence Interval for average spending of married customers (using CLT): (9252.655969008973, 9272.78527899103)
99% Confidence Interval for average spending of single customers (using CLT): (9248.280647837068, 9273.681880162932)
99% Confidence Interval for average spending of married customers (using CLT): (9249.523601894418, 9275.917646105585)
```

Are MARRIED spending more money per transaction than SINGLE? Why or Why not?

Based on the provided data, there is no significant difference in the average spending per transaction between married and single customers. The average spending per transaction for married customers is $9262.72, while for single customers, it is $9260.98. Additionally, the confidence intervals for both married and single customers overlap at different confidence levels (90%, 95%, and 99%). This overlap suggests that there is no statistically significant difference in average spending between married and single customers. Therefore, we cannot conclude that married customers are spending more money per transaction than single customers based on the provided data and confidence intervals.

Confidence intervals and distribution of the mean of the expenses by MARRIED and SINGLE customers: The confidence intervals for average spending of married and single customers provide insights into the range of possible average spending values with a certain level of confidence. These confidence intervals capture the uncertainty around the sample mean and provide a range within which the population mean is likely to fall. The overlapping confidence intervals for both married and single customers indicate that there is no significant difference in average spending between the two groups.

Are confidence intervals of average MARRIED and SINGLE spending overlapping? How can Walmart leverage this conclusion to make changes or improvements? Yes, the confidence intervals of average spending for married and single customers are overlapping. This indicates that there is no statistically significant difference in average spending between married and single customers. Walmart can leverage this conclusion to adopt a customer-centric approach that caters to the diverse needs and preferences of both married and single customers. By focusing on personalized marketing strategies and promotions that resonate with different customer segments, Walmart can enhance customer satisfaction and loyalty. Additionally, Walmart can analyze other demographic factors or customer behaviors to identify potential areas for improvement and tailor strategies accordingly to drive sales and profitability.

```python
# Define age bins based on provided categories
age_bins = [0, 17, 25, 35, 45, 50, 55, np.inf]
age_labels = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

# Calculate confidence intervals using Central Limit Theorem
def compute_confidence_interval(sample_mean, population_std, sample_size, confidence_level=0.95):
    z_score = stats.norm.ppf((1 + confidence_level) / 2)
    margin_of_error = z_score * (population_std / np.sqrt(sample_size))
    lower_bound = sample_mean - margin_of_error
    upper_bound = sample_mean + margin_of_error
    return lower_bound, upper_bound

# Bin the 'Age' column
data['Age_group'] = pd.cut(data['Age'], bins=age_bins, labels=age_labels)

# Compute the average spending for each age group
average_spending_by_age = data.groupby('Age_group', observed=True)['Purchase'].mean()

# Compute sample size for each age group
sample_sizes = data['Age_group'].value_counts()

# Calculate confidence intervals using Central Limit Theorem
confidence_intervals_age = {}
for age_group in age_labels:
    sample_mean = average_spending_by_age.loc[age_group]
    sample_size = sample_sizes.loc[age_group]
    population_std = data[data['Age_group'] == age_group]['Purchase'].std()
     # Check if standard deviation is zero
```

```python
# Filter the dataset for different age groups
age_group_0_17 = data[data['Age'] == 0]['Purchase']
age_group_18_25 = data[data['Age'] == 1]['Purchase']
age_group_26_35 = data[data['Age'] == 2]['Purchase']
age_group_36_45 = data[data['Age'] == 3]['Purchase']
age_group_46_50 = data[data['Age'] == 4]['Purchase']
age_group_51_55 = data[data['Age'] == 5]['Purchase']
age_group_55plus = data[data['Age'] == 6]['Purchase']
```

```python
# Define age group datasets
age_group_datasets = {
    '0-17': age_group_0_17,
    '18-25': age_group_18_25,
    '26-35': age_group_26_35,
    '36-45': age_group_36_45,
    '46-50': age_group_46_50,
    '51-55': age_group_51_55,
    '55+': age_group_55plus
}

# Define confidence levels
confidence_levels = [0.90, 0.95, 0.99]

# Function to calculate confidence interval
def calculate_confidence_interval(sample_mean, sample_se, confidence_level):
    z_score = stats.norm.ppf((1 + confidence_level) / 2)
    margin_of_error = z_score * sample_se
    confidence_interval = (sample_mean - margin_of_error, sample_mean + margin_of_error)
    return confidence_interval

# Iterate over each age group dataset
for age_group, dataset in age_group_datasets.items():
    print(f"Age Group: {age_group}")

    # Calculate sample statistics
    sample_mean = dataset.mean()
    sample_std = dataset.std()
    sample_size = len(dataset)
```

```python
    # Calculate standard error of the mean
    sample_se = sample_std / np.sqrt(sample_size)

    # Generate multiple random samples and compute sample means
    num_samples = 1000
    sample_means = [np.mean(np.random.choice(dataset, size=sample_size, replace=True)) for _ in range(num_samples)]

    # Calculate confidence intervals for each confidence level
    for confidence_level in confidence_levels:
        confidence_interval = calculate_confidence_interval(np.mean(sample_means), sample_se, confidence_level)
        print(f"{confidence_level*100}% Confidence Interval for average spending (using CLT):", confidence_interval)
```

```
Age Group: 0-17
90.0% Confidence Interval for average spending (using CLT): (8865.053163406192, 9001.875055240343)
95.0% Confidence Interval for average spending (using CLT): (8851.94743942098, 9014.980779225556)
99.0% Confidence Interval for average spending (using CLT): (8826.333045325011, 9040.595173321524)
Age Group: 18-25
90.0% Confidence Interval for average spending (using CLT): (9143.136932601223, 9195.598081908107)
95.0% Confidence Interval for average spending (using CLT): (9138.111849746818, 9200.623164762512)
99.0% Confidence Interval for average spending (using CLT): (9128.290610359902, 9210.444404149428)
Age Group: 26-35
90.0% Confidence Interval for average spending (using CLT): (9235.06682796617, 9270.242092543696)
95.0% Confidence Interval for average spending (using CLT): (9231.697503785075, 9273.61141672479)
99.0% Confidence Interval for average spending (using CLT): (9225.112350800851, 9280.196569709015)
Age Group: 36-45
90.0% Confidence Interval for average spending (using CLT): (9306.24855918068, 9356.067196611817)
95.0% Confidence Interval for average spending (using CLT): (9301.47659394369, 9360.839161848808)
99.0% Confidence Interval for average spending (using CLT): (9292.150058581701, 9370.165697210796)
Age Group: 46-50
90.0% Confidence Interval for average spending (using CLT): (9170.911455764683, 9247.349132537543)
95.0% Confidence Interval for average spending (using CLT): (9163.589739331537, 9254.670848970689)
99.0% Confidence Interval for average spending (using CLT): (9149.279859893431, 9268.980728408795)
Age Group: 51-55
90.0% Confidence Interval for average spending (using CLT): (9493.200889922367, 9578.494089896341)
95.0% Confidence Interval for average spending (using CLT): (9485.030931725696, 9586.664048093013)
99.0% Confidence Interval for average spending (using CLT): (9469.063211878, 9602.63176794071)
Age Group: 55+
90.0% Confidence Interval for average spending (using CLT): (9280.139039838683, 9392.564543308643)
95.0% Confidence Interval for average spending (using CLT): (9269.370166541989, 9403.333416605337)
99.0% Confidence Interval for average spending (using CLT): (9248.323014556927, 9424.380568590399)
```

Which age group is spending maximum money per transaction? Based on the provided data, the age group 51-55 appears to be spending the maximum money per transaction. The 95.0% confidence interval for average spending in this age group ranges from $9485.03 to $9586.66, which is higher compared to other age groups.

Confidence intervals and distribution of the mean of the expenses by various age group customers: The confidence intervals for average spending in each age group provide insights into the range of possible average spending values with a certain level of confidence. These intervals capture the uncertainty around the sample mean and provide a range within which the population mean is likely to fall. The intervals vary across different age groups, reflecting differences in spending behavior among various demographic segments.

Are confidence intervals of any age group overlapping? How can Walmart leverage this conclusion to make changes or improvements? Yes, the confidence intervals of average spending for different age groups are overlapping to some extent. This indicates that there may not be a significant difference in average spending between certain age groups. However, it's essential to analyze the extent of overlap and consider the confidence levels associated with each interval to draw meaningful conclusions.

Walmart can leverage this conclusion to tailor marketing strategies and promotions based on specific age groups that exhibit distinct spending patterns. For example, Walmart may develop targeted advertising campaigns or product offerings tailored to the preferences and behaviors of different age segments. Additionally, Walmart can use customer data analytics to identify opportunities for optimizing pricing strategies, product placement, and inventory management to better serve the diverse needs of customers across various age groups.

<u>FINAL INSIGHTS:</u> Based on the exploration and application of the Central Limit Theorem (CLT) to the dataset provided, here are the final insights:

- **Average Spending by Gender and Marital Status:**
    - Male customers tend to spend slightly more per transaction compared to female customers. The average spending per transaction for male customers is $9445.66, while for female customers, it is $8735.49
    - There is no significant difference in average spending between married and single customers. The average spending per transaction for married customers is $9262.72, while for single customers, it is $9260.98. Additionally, the confidence intervals for both married and single customers overlap at different confidence levels (90%, 95%, and 99%). This overlap suggests that there is no statistically significant difference in average spending between married and single customers.
- **Average Spending by Age Group:**
    - The age group 51-55 appears to be spending the maximum money per transaction, followed by the age group 55+.
    - There is variation in average spending across different age groups, with some overlapping confidence intervals indicating similar spending behavior.
- **Comments on Distribution and Relationship Between Variables:**
    - The distribution of purchase amounts exhibits variability across gender, marital status, and age groups, with some groups showing higher average spending than others.
    - There may be underlying factors influencing spending behavior, such as income, lifestyle, and purchasing preferences, which could further explain the observed differences.
    - Bivariate analysis reveals potential relationships between demographic variables and spending behavior, highlighting the importance of considering multiple factors when analyzing customer data.
- **Comments on Univariate and Bivariate Plots:**
    - Univariate plots illustrate the distribution of purchase amounts within each demographic group, providing insights into the variability and central tendency of spending behavior.

- Bivariate plots explore relationships between demographic variables and spending, helping identify patterns and correlations that may inform targeted marketing strategies and promotions.

- **Generalizing Insights for the Population:**
    - While the analysis provides valuable insights into spending behavior within the dataset, generalizing findings to the population requires careful consideration of sample representativeness and potential biases.
    - Walmart can leverage the observed patterns and trends to inform business strategies and decision-making processes, but it's essential to validate findings through further research and analysis, including larger sample sizes and external data sources.

Overall, the exploration and application of CLT provide valuable insights into customer spending behavior, helping Walmart refine its understanding of customer demographics and buying patterns to optimize business strategies and enhance sales tactics.

RECOMMENDATIONS: There are few recommendations I would like to suggest based on the detailed and comprehensive analysis I have done. Can generate more insights and provide more recommendations if some more data is available.

Tailored Marketing Campaigns: Direct marketing efforts towards the male demographic, leveraging insights that they tend to spend slightly more per transaction. Create targeted advertising campaigns and promotions tailored to appeal to male customers, with a specific focus on those aged 18-35, who represent the most active buyers.

Product Bundles and Offers: Develop bundled product offers and discounts designed to resonate with the preferences of customers aged 18-35. Highlight products and promotions that are popular among this demographic to drive higher spending per transaction and increase sales volume.

Enhanced Customer Experience: Prioritize customer service initiatives and online shopping experiences tailored to the preferences of customers aged 18-35. Implement personalized recommendations and streamlined checkout processes to cater to this demographic's needs and enhance overall satisfaction.

Customer Engagement Programs: Launch loyalty programs and rewards schemes targeted towards customers aged 18-35 to incentivize repeat purchases and foster long-term loyalty. Utilize community engagement initiatives to build brand trust and loyalty within this demographic.

Data-Driven Decision Making: Utilize customer data analytics to gain insights into the preferences and purchasing behavior of customers aged 18-35. Leverage this data to inform decisions on product assortment, pricing strategies, and inventory management, with a focus on maximizing sales volume.

Cross-Selling Opportunities: Identify cross-selling opportunities based on the preferences of customers aged 18-35. Recommend complementary products and bundle offers that appeal to their interests, driving additional sales and increasing average transaction value.

Continuous Improvement: Continuously gather feedback from customers aged 18-35 to identify areas for improvement in product offerings, customer service, and overall shopping experience. Remain agile in responding to their evolving preferences and market trends to maintain competitiveness.

Employee Training and Development: Train frontline staff to understand the preferences and behaviors of customers aged 18-35 and provide personalized assistance and recommendations. Empower employees to deliver exceptional customer service to enhance satisfaction within this demographic.

Sustainability Initiatives: Incorporate sustainability initiatives into business practices that resonate with the values of customers aged 18-35. Highlight eco-friendly products and support environmental causes to align with their preferences and foster brand loyalty.

By aligning business strategies with the preferences of the target demographic (customers aged 18-35) and focusing on key segments such as male customers, occupations 4, 0, and 7, and City Category B, Walmart can capitalize on opportunities to drive sales volume, increase customer satisfaction, and maintain a competitive edge in the market.