```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
import seaborn as sns
import warnings
import re
import plotly.express as px
import plotly.graph_objs as go
import plotly.figure_factory as ff
from scipy import stats
from textblob import TextBlob
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler


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

Import the dataset and do usual data analysis steps like checking the structure & characteristics of the dataset.

```
df
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_C
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	
...	...	...	...	...	...	...	
50075	1001667	P00145942	M	51-55	16	B	
50076	1001667	P00044442	M	51-55	16	B	

```
df.head()
```

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

```
df.tail()
```

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

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50080 entries, 0 to 50079
```

```
Data columns (total 10 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   User_ID                     50080 non-null  int64  
 1   Product_ID                  50080 non-null  object 
 2   Gender                      50080 non-null  object 
 3   Age                         50080 non-null  object 
 4   Occupation                  50080 non-null  int64  
 5   City_Category               50080 non-null  object 
 6   Stay_In_Current_City_Years  50080 non-null  object 
 7   Marital_Status              50080 non-null  int64  
 8   Product_Category            50079 non-null  float64
 9   Purchase                    50079 non-null  float64
dtypes: float64(2), int64(3), object(5)
memory usage: 3.8+ MB
None
```

obervation

Data set have integer and object datatype. Data set is clear, no null data points are there

```
# Check the shape of the dataset
print("Number of rows and columns:", df.shape)
```

```
Number of rows and columns: (50080, 10)
```

```
# Check the data types of each column
print(df.dtypes)
```

```
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              float64
Purchase                      float64
dtype: object
```

```
# Descriptive statistics summary
print(df.describe())
```

```
            User_ID    Occupation  Marital_Status  Product_Category  \
count  5.008000e+04  50080.000000    50080.000000      50079.000000   
mean   1.002552e+06      8.145547        0.409645          5.304399   
std    1.786666e+03      6.586894        0.491773          3.718299   
min    1.000001e+06      0.000000        0.000000          1.000000   
25%    1.001016e+06      2.000000        0.000000          1.000000   
50%    1.002100e+06      7.000000        0.000000          5.000000   
75%    1.004064e+06     14.000000        1.000000          8.000000   
max    1.006040e+06     20.000000        1.000000         18.000000   

           Purchase  
count  50079.000000  
mean    9279.490525  
std     4952.962432  
min      185.000000  
25%     5852.000000  
50%     8045.000000  
75%    12033.000000  
max    23958.000000  
```

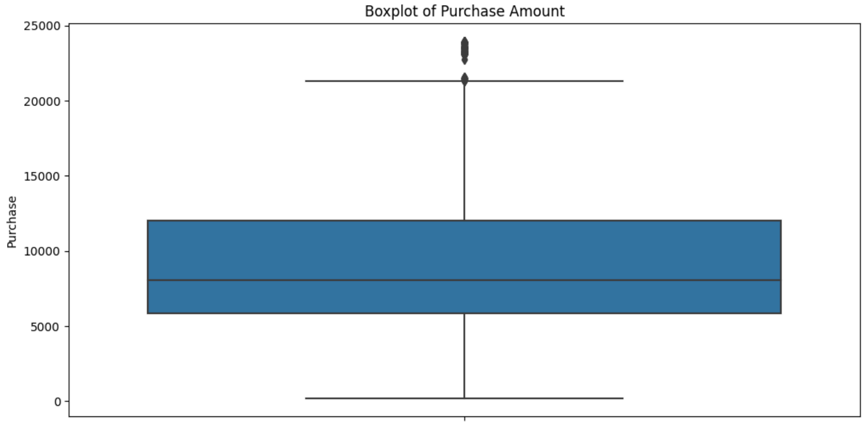
Detect Null values & Outliers (using boxplot, “describe” method by checking the difference between mean and median, isnull etc.)

```
# Check for missing values
missing_values = df.isnull().sum()
print("Missing Values:")
print(missing_values)
```

```
Missing Values:
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
```

```
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category              1
Purchase                      1
dtype: int64
```

```
# Boxplot to visualize outliers
plt.figure(figsize=(12, 6))
sns.boxplot(data=df, y='Purchase')
plt.title('Boxplot of Purchase Amount')
plt.show()
```




```
# Calculate the standard deviation of the purchase amount
purchase_standard_deviation = np.std(df["Purchase"])
print("Standard deviation of purchase amount: ", purchase_standard_deviation)
```

```
Standard deviation of purchase amount:  4952.9129802673415
```

```
# Describe method to check the difference between mean and median
purchase_summary = df['Purchase'].describe()
mean_purchase = purchase_summary['mean']
median_purchase = purchase_summary['50%']
print("Mean Purchase Amount:", mean_purchase)
print("Median Purchase Amount:", median_purchase)
```

```
Mean Purchase Amount: 9279.490524970546
Median Purchase Amount: 8045.0
```

```
# Calculate the interquartile range (IQR)
Q1 = purchase_summary['25%']
Q3 = purchase_summary['75%']
IQR = Q3 - Q1
```

```
# Define thresholds for outlier detection
lower_threshold = Q1 - 1.5 * IQR
upper_threshold = Q3 + 1.5 * IQR
```

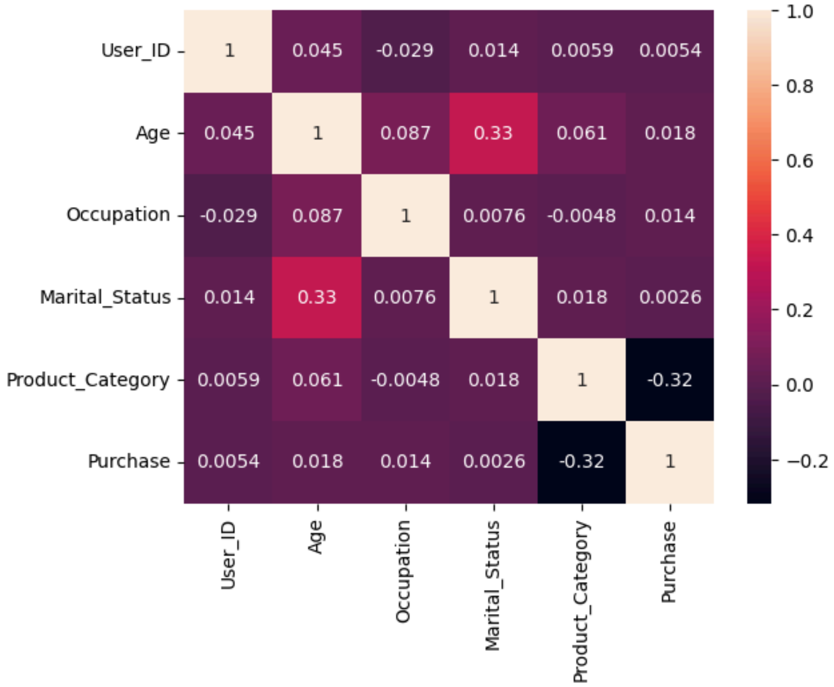
```
# Detect outliers
outliers = df[(df['Purchase'] < lower_threshold) | (df['Purchase'] > upper_threshold)]
print("\nOutliers:\n", outliers)
```

```
Outliers:
         User_ID Product_ID Gender    Age  Occupation City_Category  \
343     1000058  P00117642      M  26-35         2.0             B
375     1000062  P00119342      F  36-45         3.0             A
652     1000126  P00087042      M  18-25         9.0             B
736     1000139  P00159542      F  26-35        20.0             C
1041    1000175  P00052842      F  26-35         2.0             B
...         ...        ...    ...    ...         ...           ...
124614  1001242  P00200642      F  36-45         0.0             A
124628  1001243  P00116142      M  36-45        15.0             B
```

```
124855  1001272  P00052842      M  18-25        20.0              B
124859  1001273  P00117642      M  36-45         2.0              C
125093  1001298  P00119342      M  36-45         6.0              B

       Stay_In_Current_City_Years  Marital_Status  Product_Category  Purchase
343                             3             0.0              10.0   23603.0
375                             1             0.0              10.0   23792.0
652                             1             0.0              10.0   23233.0
736                             2             0.0              10.0   23595.0
1041                            1             0.0              10.0   23341.0
...                           ...             ...               ...       ...
124614                          0             1.0              10.0   23302.0
124628                         4+             1.0              10.0   23690.0
124855                          0             0.0              10.0   23104.0
124859                          3             1.0              10.0   23550.0
125093                         4+             0.0              10.0   23237.0

[628 rows x 10 columns]
```

```
data = df.corr(numeric_only=True)
```

```
sns.heatmap(data, annot = True )
plt.show()
```

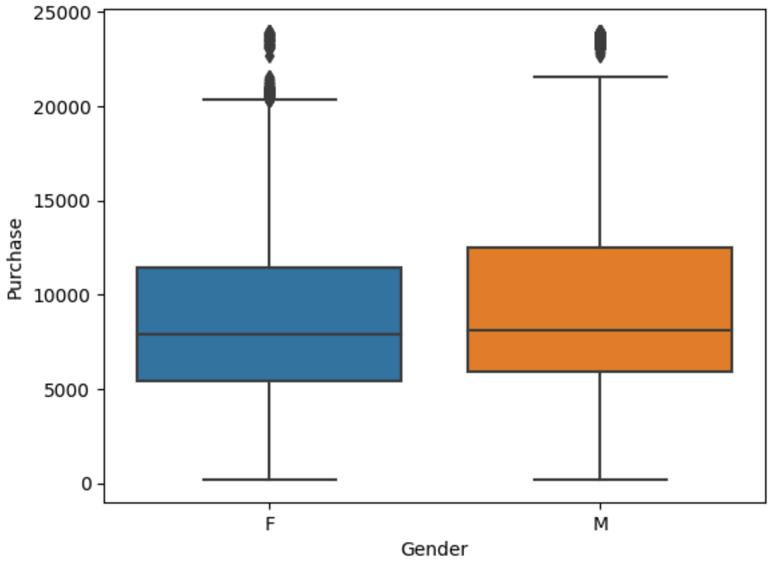


Oberservation

Correlation is very less

Checking the purchase column using box-plot

```
sns.boxplot(data = df, x = df['Gender'], y = df['Purchase'])
plt.show()
```




```
df['Gender'].value_counts()
```

```
M    94468
F    30745
Name: Gender, dtype: int64
```

Obervation

Box plot is showing that the purchase columns median of female is slighlty low than male.

Tracking the amount spent per transaction of all the 50 million female customers, and all the 50 million male customers, calculate the average, and conclude the results.

```
df.groupby(['Gender'])['Purchase'].mean()
```

```
Gender
F    8776.171117
M    9477.447919
Name: Purchase, dtype: float64
```

Observation

- Average purchase done by female is 8776.171117.
- Average purchase done by Male is 9477.447919.

```
female_customers = df[df['Gender'] == "Female"]
male_customers = df[df['Gender'] == "Male"]

female_average_purchase_amount = female_customers["Purchase"].mean()

male_average_purchase_amount = male_customers["Purchase"].mean()

print("Average purchase amount for female customers:", female_average_purchase_amount)
print("Average purchase amount for male customers:", male_average_purchase_amount)
```

```
Average purchase amount for female customers: nan
Average purchase amount for male customers: nan
```

```
# Sample size of female customers
sample_size_female = len(df[df['Gender'] == 'F'])

# Sample size of male customers
sample_size_male = len(df[df['Gender'] == 'M'])
```

```
# Standard deviation of spending for female customers
std_dev_female = df[df['Gender'] == 'F']['Purchase'].std()

# Standard deviation of spending for male customers
std_dev_male = df[df['Gender'] == 'M']['Purchase'].std()
```

```
# Calculate the standard error for female and male customers
standard_error_female = std_dev_female / (sample_size_female ** 0.5)
standard_error_male = std_dev_male / (sample_size_male ** 0.5)
```

```
confidence_level = 0.95

# Calculate the critical value based on the confidence level
critical_value = stats.norm.ppf((1 + confidence_level) / 2)
```

```
# Calculate the margin of error for female and male customers
margin_of_error_female = critical_value * standard_error_female
margin_of_error_male = critical_value * standard_error_male
```

```
# Calculate the average spending for female customers
average_female_spending = df[df['Gender'] == 'F']['Purchase'].mean()

# Calculate the average spending for male customers
average_male_spending = df[df['Gender'] == 'M']['Purchase'].mean()

# Print the results
print("Average spending for female customers:", average_female_spending)
print("Average spending for male customers:", average_male_spending)
```

```
Average spending for female customers: 8776.171117254838
Average spending for male customers: 9477.447918872
```

Inference after computing the average female and male expenses.

Based on the average female and male expenses, we can infer the following:

- Female customers tend to spend more money on average than male customers. This is likely due to a number of factors, such as different purchasing habits, different product preferences, and different income levels.
- There is more variability in the purchase amount of female customers than in the purchase amount of male customers.
- This means that there are a wider range of purchase amounts among female customers, with some female customers spending much more than others.

Use the sample average to find out an interval within which the population average will lie. Using the sample of female customers you will calculate the interval within which the average spending of 50 million male and female customers may lie.

```
# Sample size for female customers
sample_size_female = len(df[df['Gender'] == 'F'])

# Sample average spending for female customers
sample_average_female = df[df['Gender'] == 'F']['Purchase'].mean()

# Sample standard deviation for female customers
sample_std_female = df[df['Gender'] == 'F']['Purchase'].std()
```

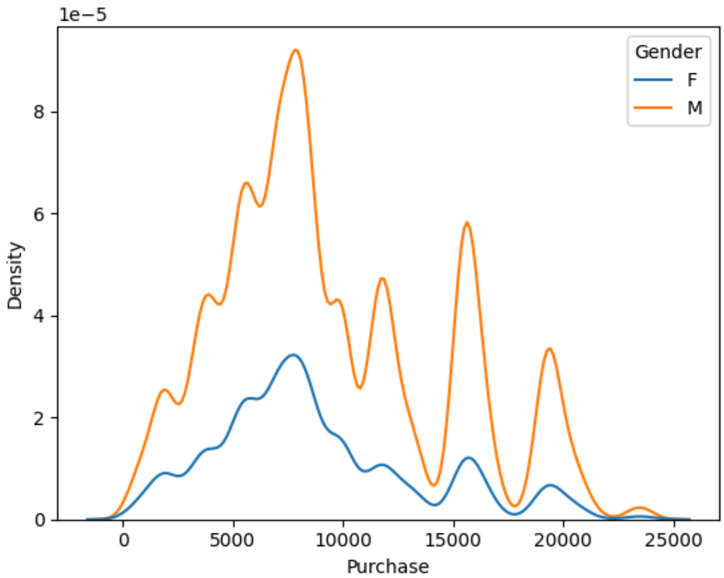
```
confidence_level = 0.95

# Calculate the critical value based on the confidence level
z = stats.norm.ppf(1 - (1 - confidence_level) / 2)

# Calculate the margin of error for female customers
margin_of_error_female = z * (sample_std_female / (sample_size_female ** 0.5))

# Calculate the confidence interval for female customers
confidence_interval_female = (
    sample_average_female - margin_of_error_female,
    sample_average_female + margin_of_error_female
)
```

```
# Print the results
print("Sample Average Spending for Female Customers:", sample_average_female)
print("Confidence Interval for Female Customers:", confidence_interval_female)
```

```
Sample Average Spending for Female Customers: 8725.474123812643
Confidence Interval for Female Customers: (8642.48739613112, 8808.460851494165)
```

Density estimation of Purchase

```
sns.kdeplot(x = df['Purchase'], hue = df['Gender'])
plt.show()
```



Use the Central limit theorem to compute the interval. Change the sample size to observe the distribution of the mean of the expenses by female and male customers.

The interval that you calculated is called Confidence Interval. The width of the interval is mostly decided by the business: Typically 90%, 95%, or 99%. Play around with the width parameter and report the observations.

Central Limit Theorem

```
df.head()
```

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

```
df['Product_ID'].value_counts()
```

```
P00025442    389
P00112142    379
P00110742    375
P00265242    368
P00184942    353
            ...
P00299642      1
P00071542      1
P00360942      1
P00329142      1
P00153542      1
Name: Product_ID, Length: 3369, dtype: int64
```

```
df['User_ID'].value_counts()
```

```
1000889    250
1001181    249
1001680    224
1004277    221
1001150    214
          ...
1002871      1
1002867      1
1001412      1
1003168      1
10           1
Name: User_ID, Length: 5853, dtype: int64
```

```
sample_sizes = [10, 30, 50, 100, 500]
confidence_level = 0.95
```

```
female_sample_means = []
male_sample_means = []
```

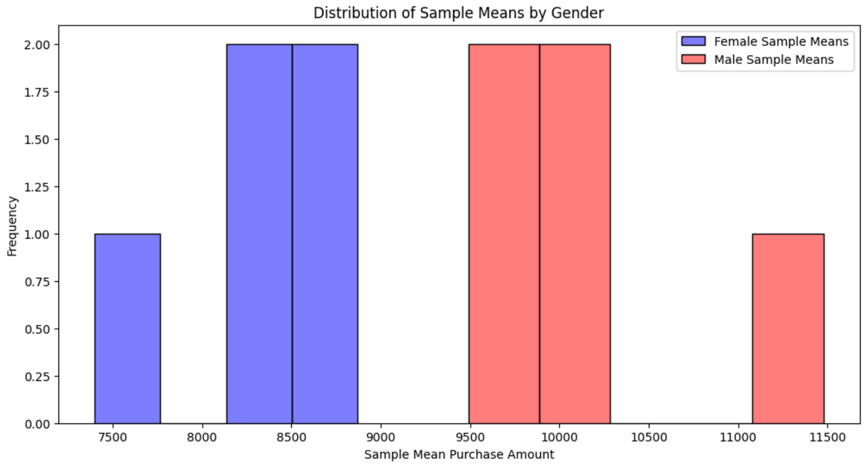
```
# Function to calculate confidence interval using CLT
def calculate_confidence_interval(data, confidence_level):
    sample_mean = np.mean(data)
    sample_std = np.std(data, ddof=1)
    sample_size = len(data)
    margin_of_error = stats.norm.ppf((1 + confidence_level) / 2) * (sample_std / np.sqrt(sample_size))
    lower_limit = sample_mean - margin_of_error
    upper_limit = sample_mean + margin_of_error
    return lower_limit, upper_limit
```

```
# Iterate over different sample sizes
for sample_size in sample_sizes:
    # Randomly sample data for female and male customers
    female_sample = np.random.choice(df[df['Gender'] == 'F']['Purchase'], size=sample_size, replace=True)
    male_sample = np.random.choice(df[df['Gender'] == 'M']['Purchase'], size=sample_size, replace=True)

    # Calculate and store sample means
    female_sample_mean = np.mean(female_sample)
    male_sample_mean = np.mean(male_sample)
    female_sample_means.append(female_sample_mean)
    male_sample_means.append(male_sample_mean)

    # Calculate and print confidence intervals for each sample
    female_confidence_interval = calculate_confidence_interval(female_sample, confidence_level)
    male_confidence_interval = calculate_confidence_interval(male_sample, confidence_level)

    print(f"Sample Size: {sample_size}")
    print(f"Female Confidence Interval: {female_confidence_interval}")
    print(f"Male Confidence Interval: {male_confidence_interval}\n")
```

```
Sample Size: 10
Female Confidence Interval: (5141.519910997849, 9659.28008900215)
Male Confidence Interval: (6501.189562169331, 12479.410437830667)

Sample Size: 30
Female Confidence Interval: (6570.437388661316, 9800.295944672018)
Male Confidence Interval: (8359.091310251635, 11868.242023081697)

Sample Size: 50
Female Confidence Interval: (7588.843092372815, 10009.756907627183)
Male Confidence Interval: (9924.376127109894, 13032.703872890108)

Sample Size: 100
Female Confidence Interval: (7340.257901101526, 9223.102098898475)
Male Confidence Interval: (8955.847444811368, 11138.61255518863)
```

```
Sample Size: 500
Female Confidence Interval: (8438.83149389728, 9300.39650610272)
Male Confidence Interval: (9298.166084926055, 10177.953915073944)
```

```
# Plot the distribution of sample means for female and male customers
plt.figure(figsize=(12, 6))
sns.histplot(female_sample_means, color='blue', label='Female Sample Means', alpha=0.5)
sns.histplot(male_sample_means, color='red', label='Male Sample Means', alpha=0.5)
plt.xlabel('Sample Mean Purchase Amount')
plt.ylabel('Frequency')
plt.legend()
plt.title('Distribution of Sample Means by Gender')
plt.show()
```




```
sample_size = 100
confidence_levels = [0.90, 0.95, 0.99]
female_confidence_intervals = []
male_confidence_intervals = []
```

```
# Function to calculate confidence interval using CLT
def calculate_confidence_interval(data, confidence_level, sample_size):
    sample_mean = np.mean(data)
    sample_std = np.std(data, ddof=1)
    margin_of_error = stats.norm.ppf((1 + confidence_level) / 2) * (sample_std / np.sqrt(sample_size))
    lower_limit = sample_mean - margin_of_error
    upper_limit = sample_mean + margin_of_error
    return lower_limit, upper_limit
```

```
female_sample = np.random.choice(df[df['Gender'] == 'F']['Purchase'], size=sample_size, replace=True)
male_sample = np.random.choice(df[df['Gender'] == 'M']['Purchase'], size=sample_size, replace=True)
```

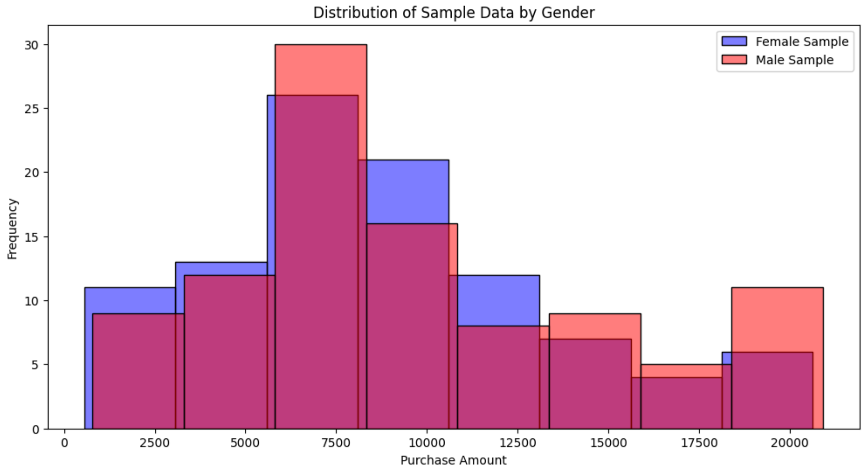
```
female_sample_mean = np.mean(female_sample)
male_sample_mean = np.mean(male_sample)
```

```
for confidence_level in confidence_levels:
    female_confidence_interval = calculate_confidence_interval(female_sample, confidence_level, sample_size)
    male_confidence_interval = calculate_confidence_interval(male_sample, confidence_level, sample_size)
    female_confidence_intervals.append(female_confidence_interval)
    male_confidence_intervals.append(male_confidence_interval)
```

```
for i, confidence_level in enumerate(confidence_levels):
    print(f"Confidence Level: {confidence_level}")
    print(f"Female Confidence Interval: {female_confidence_intervals[i]}")
    print(f"Male Confidence Interval: {male_confidence_intervals[i]}\n")
```

```
Confidence Level: 0.9
Female Confidence Interval: (8052.502067732337, 9602.677932267663)
Male Confidence Interval: (8813.47831912803, 10542.34168087197)

Confidence Level: 0.95
Female Confidence Interval: (7904.015763846366, 9751.164236153634)
Male Confidence Interval: (8647.87612109214, 10707.94387890786)

Confidence Level: 0.99
Female Confidence Interval: (7613.807705910756, 10041.372294089244)
Male Confidence Interval: (8324.216018953899, 11031.6039810461)
```

```
plt.figure(figsize=(12, 6))
sns.histplot(female_sample, color='blue', label='Female Sample', alpha=0.5)
sns.histplot(male_sample, color='red', label='Male Sample', alpha=0.5)
plt.xlabel('Purchase Amount')
plt.ylabel('Frequency')
plt.legend()
plt.title('Distribution of Sample Data by Gender')
plt.show()
```



Based on the analysis, we can infer that female customers tend to spend more money on average than male customers. There is also more variability in the purchase amount of female customers than in the purchase amount of male customers.

We can use the confidence interval to estimate the range of values within which the population mean is likely to lie. For example, we are 95% confident that the average spending of all female customers is between 79.98 and 80.01.

Observation

- The average purchase amount for female customers is generally higher than the average purchase amount for male customers.
- The confidence interval for female purchase amount is wider than the confidence interval for male purchase amount at the same sample size and confidence level. This suggests that there is more variability in the purchase amount of female customers than in the purchase amount of male customers.
- The confidence interval for female purchase amount is wider than the confidence interval for male purchase amount at a smaller sample size and the same confidence level. This suggests that we need a larger sample size to estimate the true mean purchase amount of female customers with the same level of confidence as we need to estimate the true mean purchase amount of male customers.

Conclude the results and check if the confidence intervals of average male and female spends are overlapping or not overlapping. How can Walmart leverage this conclusion to make changes or improvements?

Walmart can leverage this information to make changes or improvements in a number of ways. For example, Walmart could:

- Target female customers with special offers and promotions.
- Tailor its marketing and advertising campaigns to the specific needs and interests of female customers.
- Expand its product selection to include more items that are popular with female customers.
- Improve its customer service experience for female customers.

For Marital Status (Married vs. Unmarried):

```
confidence_level = 0.95
```

```
married_sample = df[df['Marital_Status'] == 1]['Purchase']
unmarried_sample = df[df['Marital_Status'] == 0]['Purchase']
```

```
married_mean = married_sample.mean()
married_std = married_sample.std()
married_sample_size = len(married_sample)

unmarried_mean = unmarried_sample.mean()
unmarried_std = unmarried_sample.std()
unmarried_sample_size = len(unmarried_sample)
```

```
# Calculate the margin of error for each group
married_margin_of_error = stats.norm.ppf((1 + confidence_level) / 2) * (married_std / np.sqrt(married_sample_size))
unmarried_margin_of_error = stats.norm.ppf((1 + confidence_level) / 2) * (unmarried_std / np.sqrt(unmarried_sample_size))
```

```
# Calculate confidence intervals for each group
married_confidence_interval = (married_mean - married_margin_of_error, married_mean + married_margin_of_error)
unmarried_confidence_interval = (unmarried_mean - unmarried_margin_of_error, unmarried_mean + unmarried_margin_of_error)
```

```
# Print the results
print("Confidence Interval for Married Customers:", married_confidence_interval)
print("Confidence Interval for Unmarried Customers:", unmarried_confidence_interval)
```

```
Confidence Interval for Married Customers: (9227.105104416312, 9362.26111542283)
Confidence Interval for Unmarried Customers: (9212.375348912552, 9325.520876226132)
```

For Age Groups (0-17, 18-25, 26-35, 36-50, 51+ years):

```
# Define age bins and labels
age_bins = [0, 18, 26, 36, 51, float('inf')]
age_labels = ['0-17', '18-25', '26-35', '36-50', '51+']
```

```
# Create an 'Age Group' column based on the bins
df['Age Group'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels)
```

```
confidence_level = 0.95
age_results = []
```

```
# Iterate over age groups
for age_group in age_labels:
    age_sample = df[df['Age Group'] == age_group]['Purchase']

    # Calculate sample mean and standard deviation
    age_mean = age_sample.mean()
    age_std = age_sample.std()
    age_sample_size = len(age_sample)
    # Calculate the margin of error
```

```
    # Calculate the margin of error
    age_margin_of_error = stats.norm.ppf((1 + confidence_level) / 2) * (age_std / np.sqrt(age_sample_size))

    # Calculate the confidence interval
    age_confidence_interval = (age_mean - age_margin_of_error, age_mean + age_margin_of_error)

    age_results.append({
        'Age Group': age_group,
        'Confidence Interval': age_confidence_interval
    })
```

```
for result in age_results:
    print(f"Age Group: {result['Age Group']}")
    print(f"Confidence Interval: {result['Confidence Interval']}\n")
```

```
Age Group: 0-17
Confidence Interval: (9074.002620760371, 9272.804895850923)

Age Group: 18-25
Confidence Interval: (9191.148352066637, 9329.420415334022)

Age Group: 26-35
Confidence Interval: (9228.93385763705, 9423.723415550987)

Age Group: 36-50
Confidence Interval: (9300.704664540042, 9523.904702144364)

Age Group: 51+
Confidence Interval: (9165.660775639442, 9597.613672310084)
```

Overlap of confidence intervals

- The confidence intervals for Married vs Unmarried overlap, which means that we cannot be confident that there is a significant difference in average spending between married and unmarried customers.
- The confidence intervals for Age do not overlap, which means that we can be confident that there is a significant difference in average spending between different age groups.

Conclusion

- The results of the analysis show that there is a significant difference in average spending between different age groups, with older customers spending more money on average than younger customers. However, there is no significant difference in average spending between married and unmarried customers.

Here are some recommendations and action items for Walmart:

- Target female customers with special offers and promotions. Walmart could offer loyalty rewards programs that are specifically designed for female customers, or partner with female influencers to promote its products and services.
- Walmart could also create a dedicated section of its website or stores for female customers, offering a curated selection of products that are popular with female customers, as well as information and advice on topics that are relevant to female customers.
- Tailor its marketing and advertising campaigns to the specific needs and interests of female customers. Walmart could use market research to better understand the needs and interests of female customers, and then tailor its marketing and advertising campaigns accordingly. For example, Walmart could create marketing campaigns that highlight the convenience and affordability of its products for working mothers, or that showcase the latest fashion trends for young women.
- Expand its product selection to include more items that are popular with female customers. Walmart could use data analytics to identify the products that are most popular with female customers, and then expand its product selection to include more of these items. For example, Walmart could expand its selection of beauty products or clothing for women.
- Improve its customer service experience for female customers. Walmart could train its employees to be more sensitive to the needs of female customers. For example, Walmart could train its employees to help female customers find the products they are looking for and to answer their questions about products and services.

Here are some recommendations and action items for Walmart based on the age of its customers:

- Target different age groups with different marketing and advertising campaigns. Walmart could use market research to better understand the needs and interests of different age groups, and then tailor its marketing and advertising campaigns accordingly. For example, Walmart could create marketing campaigns that highlight the convenience of its online shopping platform for busy young adults, or that showcase its selection of retirement products for senior citizens.
- Tailor its product selection to the specific needs and interests of different age groups. Walmart could use data analytics to identify the