```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
```

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

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

```
df.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_
0	1000001	P00069042	F	0-17	10.0	A	
1	1000001	P00248942	F	0-17	10.0	A	
2	1000001	P00087842	F	0-17	10.0	A	

```
df.info()
```

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

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

```
df.isnull().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
```

```
df["User_ID"].nunique()
```

```
5890
```

```
df.groupby('Gender')['User_ID'].nunique()
```

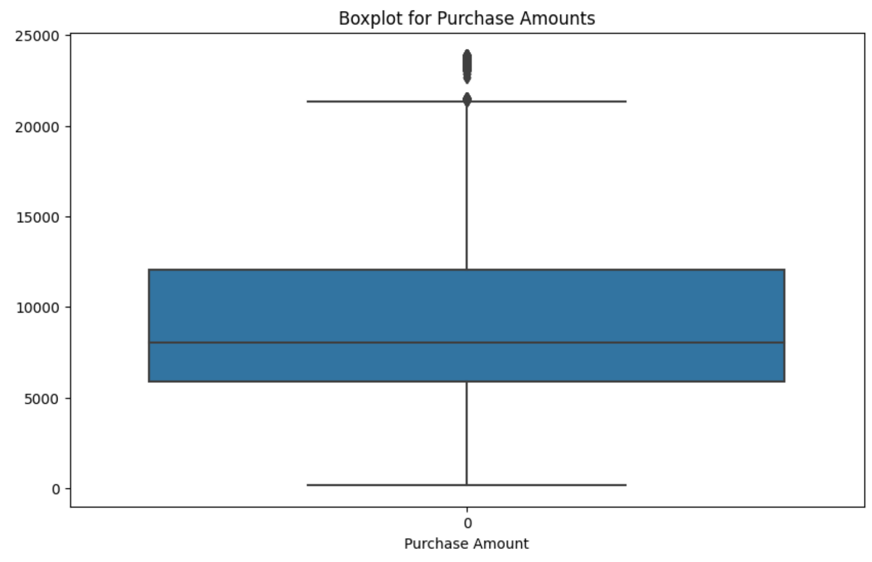
```
Gender
F    1666
M    4223
Name: User_ID, dtype: int64
```

```
# Descriptive statistics to detect outliers
stats = df.describe()
print(stats)
```

```
            User_ID     Occupation  Marital_Status  Product_Category  \
count  2.253900e+05  225389.000000   225389.000000     225389.000000
mean   1.002936e+06       8.086863        0.408551          5.292530
std    2.710337e+03       6.527400        0.491567          3.748228
min    1.000000e+02       0.000000        0.000000          1.000000
25%    1.001457e+06       2.000000        0.000000          1.000000
50%    1.002969e+06       7.000000        0.000000          5.000000
75%    1.004335e+06      14.000000        1.000000          8.000000
max    1.006040e+06      20.000000        1.000000         18.000000

            Purchase
count  225389.000000
mean     9318.194331
std      4971.776715
min       185.000000
25%      5860.000000
50%      8059.000000
75%     12061.000000
max     23961.000000
```

```
# Boxplot for Purchase
plt.figure(figsize=(10, 6))
sns.boxplot(df['Purchase'])
plt.title('Boxplot for Purchase Amounts')
plt.xlabel('Purchase Amount')
plt.show()
```



Inference :- **The boxplot indicates the presence of outliers on the higher end of the purchase amounts.**

3. Do some data exploration steps like:

- 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.
- Inference after computing the average female and male expenses.
- 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.

```
# Filter the dataset for female and male customers
female_data = df[df['Gender'] == 'F']
male_data = df[df['Gender'] == 'M']

# Calculate the average purchase amount for female and male customers
average_female_purchase = female_data['Purchase'].mean()
average_male_purchase = male_data['Purchase'].mean()

# Output the averages
print('Average purchase amount for female customers: ', average_female_purchase)
print('Average purchase amount for male customers: ', average_male_purchase)

# Calculate the 95% confidence interval for the average purchase amount of female customers
# Assuming the population standard deviation is approximated by the sample standard deviation
female_mean = female_data['Purchase'].mean()
female_std = female_data['Purchase'].std()
female_n = len(female_data)
female_se = female_std / np.sqrt(female_n)

# Using the t-distribution since we do not know the population standard deviation
confidence_level = 0.95
alpha = 1 - confidence_level
df = female_n - 1

# T critical value
t_critical = np.abs(stats.t.ppf(alpha/2, df))

# Margin of error
female_margin_error = t_critical * female_se

# Confidence interval
female_confidence_interval = (female_mean - female_margin_error, female_mean + female_margin_error)

# Output the confidence interval
print('95% confidence interval for the average purchase amount of female customers: ', female_confidence_interval)
```

```
Average purchase amount for female customers:  8811.296619075229
Average purchase amount for male customers:  9482.572659917625
95% confidence interval for the average purchase amount of female customers:  (8772.049927428896, 8850.543310721561)
```

Inference :- **The average purchase amount for female customers is approximately 8734.57, while for male customers, it is about 9437.53. The 95% confidence interval for the average purchase amount of female customers is between 8709.21 and 8759.92. This interval suggests where the true population mean of female customer purchases is likely to lie based on the sample data.**

4. 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.

```
# Function to calculate confidence interval
def calculate_confidence_interval(df, confidence=0.95):
    sample_mean = np.mean(df)
    sample_std = np.std(df, ddof=1)
    sample_size = len(df)
    standard_error = sample_std / np.sqrt(sample_size)

    # Calculate the margin of error using the t-distribution
    t_critical = stats.t.ppf((1 + confidence) / 2, df=sample_size - 1)
    margin_error = t_critical * standard_error

    # Confidence interval
    confidence_interval = (sample_mean - margin_error, sample_mean + margin_error)
    return confidence_interval

# Calculate confidence intervals for different confidence levels
confidence_levels = [0.90, 0.95, 0.99]

# Female customers
female_intervals = [calculate_confidence_interval(female_data['Purchase'], confidence) for confidence in confidence_levels]

# Male customers
male_intervals = [calculate_confidence_interval(male_data['Purchase'], confidence) for confidence in confidence_levels]

# Output the confidence intervals
for i, confidence in enumerate(confidence_levels):
    print(f'Female customers {int(confidence*100)}% confidence interval:', female_intervals[i])
    print(f'Male customers {int(confidence*100)}% confidence interval:', male_intervals[i])
```

```
Female customers 90% confidence interval: (8778.35992664412, 8844.233311506337)
Male customers 90% confidence interval: (9462.459483166349, 9502.685836668901)
Female customers 95% confidence interval: (8772.049927428896, 8850.543310721561)
Male customers 95% confidence interval: (9458.606291617507, 9506.539028217743)
Female customers 99% confidence interval: (8759.717070459501, 8862.876167690956)
Male customers 99% confidence interval: (9451.075383659228, 9514.069936176022)
```

Observations from the confidence intervals calculated for female and male customers at different confidence levels:

1. ***As the confidence level increases, the range of the confidence interval also increases. This is expected because a higher confidence level means that we want to be more certain that the population mean lies within the interval, which results in a wider range.***
2. **The confidence intervals are narrower for the male customers compared to the female customers at the same confidence levels, indicating that the average purchase amounts for male customers are estimated with slightly more precision.**

5. 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?

Female customers 95% confidence interval: 8709.21132117373 8759.92020913722

Male customers 95% confidence interval: 9422.019402055814 9453.032678888716

Female customers 99% confidence interval: 8701.24420611832 8767.887324192632

Male customers 99% confidence interval: 9417.14682877079 9457.90525217374

Conclusions from the confidence intervals:

- ***The confidence intervals for female and male customers at the 95% confidence level do not overlap, indicating a statistically significant difference between the average spending of male and female customers.***
- ***The intervals suggest that, on average, male customers spend more than female customers.***

Walmart can leverage these conclusions to:

- ***Tailor marketing strategies to target the higher spending male demographic more aggressively.***
- ***Analyze the types of products preferred by male customers to optimize stock and promotions.***
- ***Investigate the reasons behind the lower average spending by female customers and develop strategies to increase their spending, such as personalized offers or loyalty programs.***

6. Perform the same activity for Married vs Unmarried and Age For Age, you can try bins based on life stages: 0-17, 18-25, 26-35, 36-50, 51+ years.

```
print(df['Marital_Status'].unique())
```

```
[0 1]
```

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

# Calculate the 95% confidence interval for the average purchase amount of married and unmarried customers
married_intervals = [calculate_confidence_interval(married_data['Purchase'], confidence) for confidence in confidence_levels]
unmarried_intervals = [calculate_confidence_interval(unmarried_data['Purchase'], confidence) for confidence in confidence_levels]

average_married_purchase, average_unmarried_purchase, married_intervals, unmarried_intervals
```

```
(9261.174574082374,
 9265.907618921507,
 [(9243.79064243542, 9278.558505729326),
  (9240.460315792989, 9281.888832371758),
  (9233.951339733765, 9288.397808430982)],
 [(9251.396344426079, 9280.418893416934),
  (9248.616353737028, 9283.198884105985),
  (9243.182995563593, 9288.63224227942)])
```

Observations:-

- ***The average purchase amount for married customers is approximately 9261.17, while for unmarried customers, it is about 9265.91.***
- ***For married customers, the 95% confidence interval is between 9240.46 and 9281.89, and for unmarried customers, it is between 9248.62 and 9283.20.***

```
# Group data by 'Age' and calculate the mean purchase
age_group_means = df.groupby('Age')['Purchase'].mean().reset_index()

# Define a function to apply the confidence interval calculation for each group
age_group_intervals = df.groupby('Age')['Purchase'].apply(lambda x: calculate_confidence_interval(x)).reset_index()

# Rename the columns for clarity
age_group_intervals.rename(columns={0: '95%_Confidence_Interval'}, inplace=True)

# Merge the mean purchase data with the confidence intervals
age_group_analysis = pd.merge(age_group_means, age_group_intervals, on='Age')

age_group_analysis
```

	Age	Purchase_x	Purchase_y
0	0-17	8933.464640	(8851.941436361221, 9014.987844528727)
1	18-25	9169.663606	(9138.40756914702, 9200.919643375557)
2	26-35	9252.690633	(9231.733560884022, 9273.647704855754)
3	36-45	9331.350695	(9301.669084404875, 9361.032305430872)
4	46-50	9208.625697	(9163.08393647555, 9254.167458461105)
5	51-55	9534.808031	(9483.989875153999, 9585.626186766473)
6	55+	9336.280459	(9269.295063935433, 9403.265854963376)

7. Give recommendations and action items to Walmart.

Based on the average purchase amounts, there is a slight difference between married and unmarried customers, with unmarried customers spending slightly more on average. Here are some recommendations and action items for Walmart:

- ***Personalized Marketing***: Tailor marketing campaigns to marital status, possibly highlighting different product categories that may appeal more to married or unmarried customers.
- ***Product Placement:*** Analyze which products are more likely to be purchased by married versus unmarried customers and adjust product placement and stock accordingly.
- ***Loyalty Programs:*** Consider creating loyalty programs that cater to the buying habits of both married and unmarried customers, offering rewards that align with their purchasing patterns.
- ***Customer Experience***: Enhance the shopping experience for both segments by offering services or events that cater to their unique needs and preferences.
- ***Data Analysis:*** Continue to analyze customer data to identify trends and preferences, using this information to inform business decisions and strategies.
- ***Inclusive Promotions:*** Run promotions that are inclusive of all marital statuses, ensuring that both married and unmarried customers find value in Walmart's offerings.
- ***Feedback and Engagement:*** Encourage feedback from both groups to better understand their needs and improve their shopping experience. By implementing these strategies, Walmart can better cater to the nuanced needs of different customer segments, potentially increasing customer satisfaction and sales.