Business-Case-Walmart--Confidence-Interval-and-CLT

Details:

Name : Tanmoy Basuli

Email-Id : basuli575tanmoy@gmail.com

Introduction

In this study-case, I'll give an Exploratory Data Analysis of the Walmart dataset given by scaler. I will explore the data and hopefully bring some insights.

- We will try to find some insights on below,
    - Univariate Analysis
    - Bivariate Analysis
- We will see Confidence Level of average male/female, married/unmarried, different age groups spent using Central Limit Theorm.
- Will try to create Customer Profiling - Categorization of users.

For visualizations I used, seaborn, pyplot, plotly. Some of the visuals interactive, and some of it static. But there's a lot improve. Feedback is always welcome.

Walmart

Walmart is an American multinational retail corporation that operates a chain of supercenters, discount departmental stores, and grocery stores from the United States. Walmart has more than 100 million customers worldwide.

Business Problem

The Management team at Walmart Inc. wants to analyze the customer purchase behavior (specifically, purchase amount) against the customer's gender and the various other factors to help the business make better decisions. They want to understand if the spending habits differ between male and female customers: Do women spend more on Black Friday than men? (Assume 50 million customers are male and 50 million are female).

Dataset: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094

The Outline of this notebook is as follows.

1. Basic Data Exploration
    - Feature Exploration
    - Summary Statistics
2. Data Cleaning
    - Null Value Analysis
    - Checking Duplicate Values

3. Exploratory data analysis (What is the Story Of Data)

Importing Libraries and Loading the Dataset

```
# Import Relevant Packages
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

First, lets load the dataset.

```
# Load the dataset

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

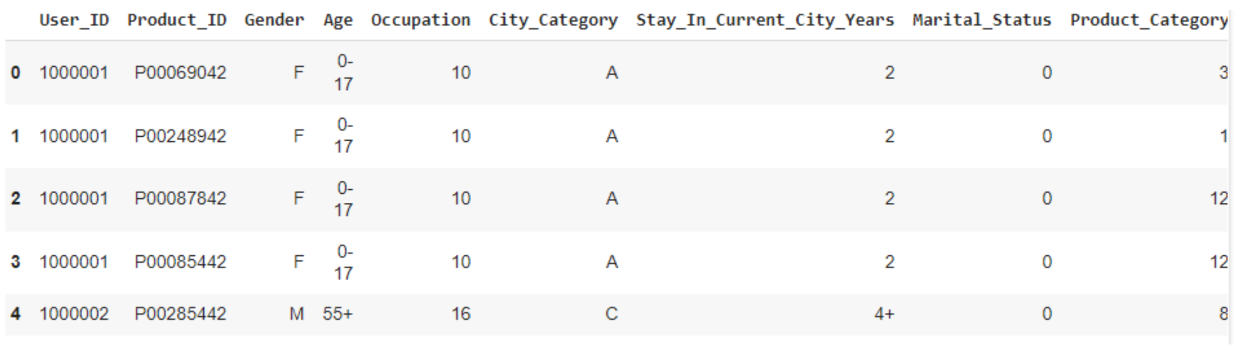
Basic Data Exploration

1. Feature Exploration
2. Summary Statistics

1. Feature Exploration

First, let us look at a quick peek of what the first five rows in the data has in store for us and what features we have.

```
#Let see first five row of our dataset

df.head()
```



	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category
0	1000001	P00069042	F	0-17	10	A	2	0	3
1	1000001	P00248942	F	0-17	10	A	2	0	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

Let's see columns,

```
# Let's check the columns we have

df.columns
```

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

Features & Descriptions:

In this dataset we have,

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

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

Now let see how much data we have in this dataset.

```
df.shape
```

(550068, 10)

We have 550068 Entity from 10 Features.

Now, Let's look What types of data we have:

```
# Type od the data we have

df.info()
```

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

Observation (Data Type)

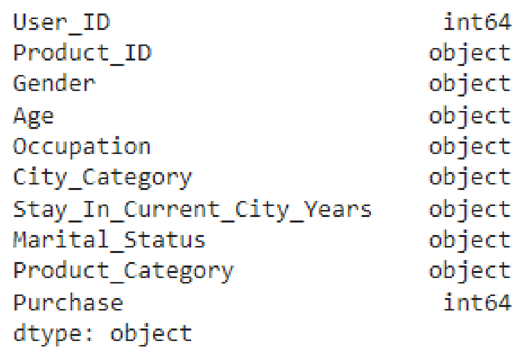
1. Data type of User_ID is int64.
2. Data type of Product_ID is object (string).
3. Data type of Gender is object (string).
4. Data type of Age is object (string).
5. Data type of Occupationis int64.
6. Data type of City_Category is object (string).
7. Data type of Stay_In_Current_City_Years is object (string).
8. Data type of Marital_Status is int64.
9. Data type of Product_Category is int64.
10. Data type of Purchase is int64.

We can easily find out from the above showcase, we have:

- 5 Categorical Feature
- 5 Numeric Feature

Change the data types of - Occupation, Marital_Status, Product_Category

```
# Changing the datatypes of 'Occupation', 'Marital_Status',
'Product_Category'.

cols = ['Occupation', 'Marital_Status', 'Product_Category']
df[cols] = df[cols].astype('object')
```

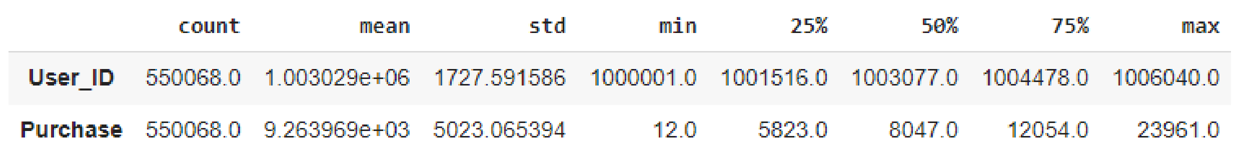


```
User_ID                        int64
Product_ID                    object
Gender                        object
Age                           object
Occupation                    object
City_Category                 object
Stay_In_Current_City_Years    object
Marital_Status                object
Product_Category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

3. **Summary Statistics(Non-Graphical Analysis: Value counts and unique attributes)**

Here we can see basic statistics in the data.

```
# Summary statistics for numerical features
numerical_features = df.select_dtypes(include='number')
# We have only 'release_year' as a numeric feature
numerical_features.describe().T
```



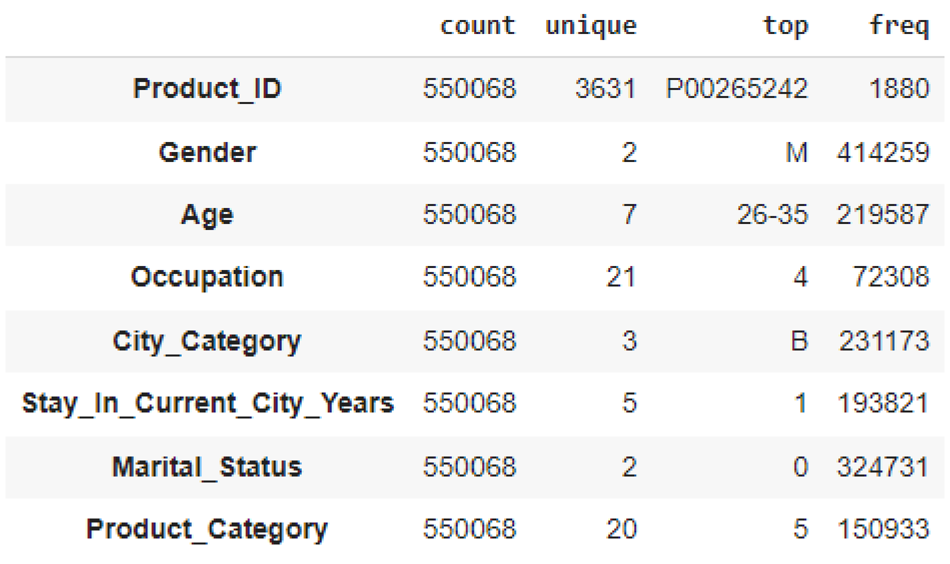
	count	mean	std	min	25%	50%	75%	max
User_ID	550068.0	1.003029e+06	1727.591586	1000001.0	1001516.0	1003077.0	1004478.0	1006040.0
Purchase	550068.0	9.263969e+03	5023.065394	12.0	5823.0	8047.0	12054.0	23961.0

Observations from Descriptive Statistics (Numerical)

1. We can't see there is missing value present over there.
2. Might have some outlier value in Purchase.

Now we will look into categorical data in our dataset.

```
# Summary statistics for categorical features
categorical_features = df.select_dtypes(include='object')
categorical_features.describe().T
```



	count	unique	top	freq
Product_ID	550068	3631	P00265242	1880
Gender	550068	2	M	414259
Age	550068	7	26-35	219587
Occupation	550068	21	4	72308
City_Category	550068	3	B	231173
Stay_In_Current_City_Years	550068	5	1	193821
Marital_Status	550068	2	0	324731
Product_Category	550068	20	5	150933

```
categorical_cols = ['Gender', 'Age', 'Occupation', 'City_Category',
'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category']
df[categorical_cols].melt().groupby(['variable',
'value'])[['value']].count()/len(df)
```

```
            value

variable    value

Age

0-17   0.027455

18-25  0.181178
```

```
26-35  0.399200

36-45  0.199999

46-50  0.083082

51-55  0.069993

55+    0.039093

City_Category

A      0.268549

B      0.420263

C      0.311189

Gender

F      0.246895

M      0.753105

Marital_Status

0      0.590347

1      0.409653

Occupation

0      0.126599

1      0.086218

2      0.048336

3      0.032087

4      0.131453

5      0.022137

6      0.037005
```

```
7       0.107501
8       0.002811
9       0.011437
10      0.023506
11      0.021063
12      0.056682
13      0.014049
14      0.049647
15      0.022115
16      0.046123
17      0.072796
18      0.012039
19      0.015382
20      0.061014
Product_Category
1       0.255201
2       0.043384
3       0.036746
4       0.021366
5       0.274390
6       0.037206
7       0.006765
8       0.207111
```

```
9       0.000745
10      0.009317
11      0.044153
12      0.007175
13      0.010088
14      0.002769
15      0.011435
16      0.017867
17      0.001051
18      0.005681
19      0.002914
20      0.004636
Stay_In_Current_City_Years
0       0.135252
1       0.352358
2       0.185137
3       0.173224
4+      0.154028
```

Observations

~ 80% of the users are between the age 18-50 (40%: 26-35, 18%: 18-25, 20%: 36-45)

75% of the users are Male and 25% are Female

60% Single, 40% Married

35% Staying in the city from 1 year, 18% from 2 years, 17% from 3 years

Total of 20 product categories are there

There are 20 different types of occupations in the city

How many users are there in the dataset?

```
# How many unique user we have in thi dataset

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

```
5891
```

How many products are there?

```
# How many unique product we have

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

```
3631
```

Now will check any None/Null values is there or not:

```
# Checking null value if we have
df.isnull().values.any()
```

```
False
```

```
# Controlling null values again
df.isnull().sum()
```

```
Product        0
Age          0
Gender        0
Education      0
MaritalStatus   0
Usage         0
Fitness        0
Income         0
Miles         0
dtype: int64
```

We can easily find there is no null values is present over the whole dataset.

Now will check for duplicated value present or not:

```
# Check Duplicate value
df.duplicated().sum()
```

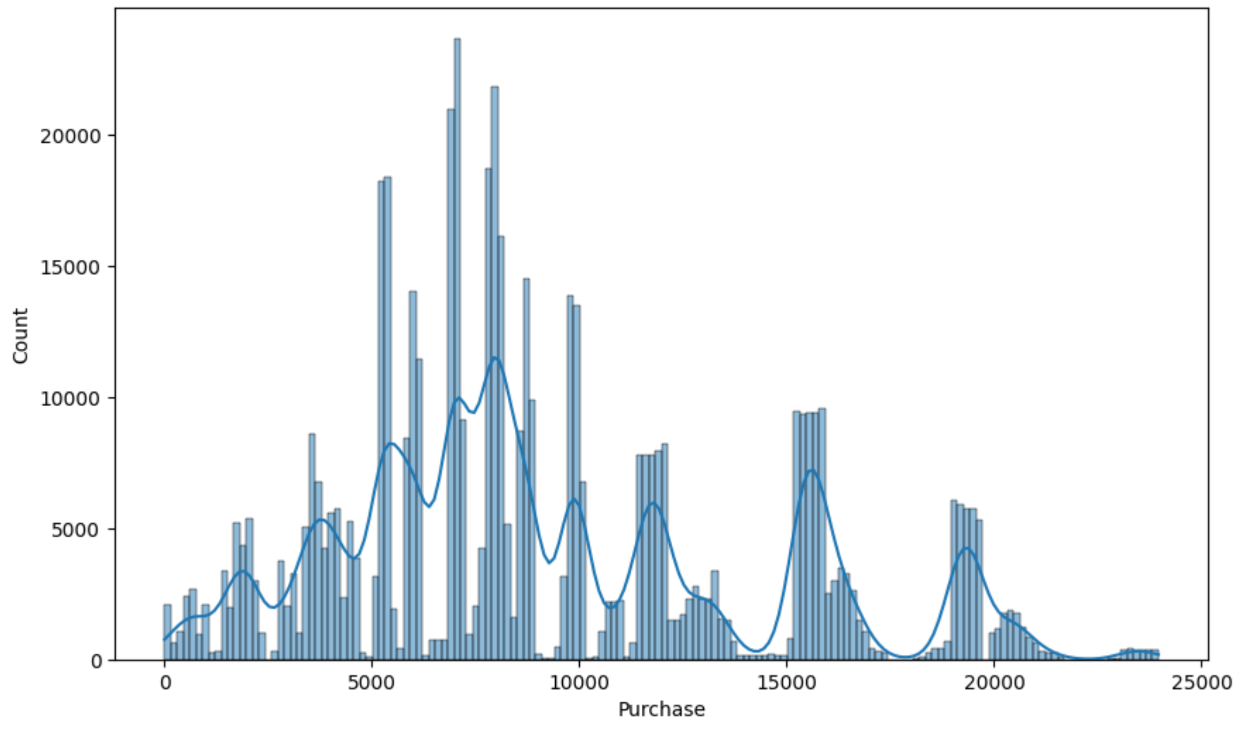
```
0
```

Visual Analysis

Univariate Analysis

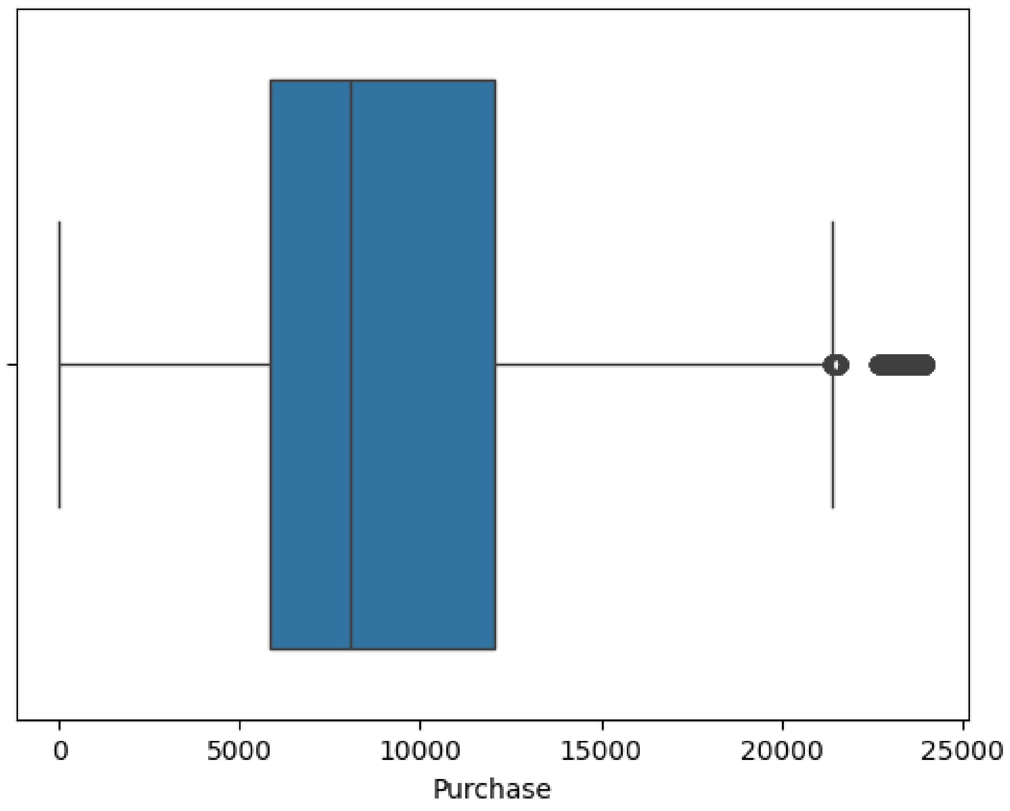
Understanding the distribution of data and try to find it outliers of the numerical feature.

```
# Try to see Purchase

plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Purchase', kde=True)
plt.show()
```

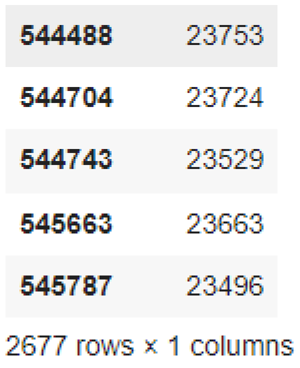


Can be able to detect outlier through boxplot.

```
# Checking with boxplot

sns.boxplot(data=df, x='Purchase', orient='h')
plt.show()
```



```
# Finding q1 and q3 points of purchase
Q3, Q1 = np.percentile(df['Purchase'], [75,25])

# Inter Quartile Range of Purchase
IQR = Q3 - Q1

# Trying to find out outliers extreeme points
A = Q3 + (1.5 * IQR)
B = Q1 - (1.5 * IQR)

# Now a task to find outliers
df[df['Purchase'] > A]['Purchase'].to_frame()
```



544488	23753
544704	23724
544743	23529
545663	23663
545787	23496

2677 rows × 1 columns

```
# Unique outlier of purchase
df[df['Purchase'] > A]['Purchase'].to_frame().nunique()
```

```
Purchase    1027
dtype: int64
```

Observations

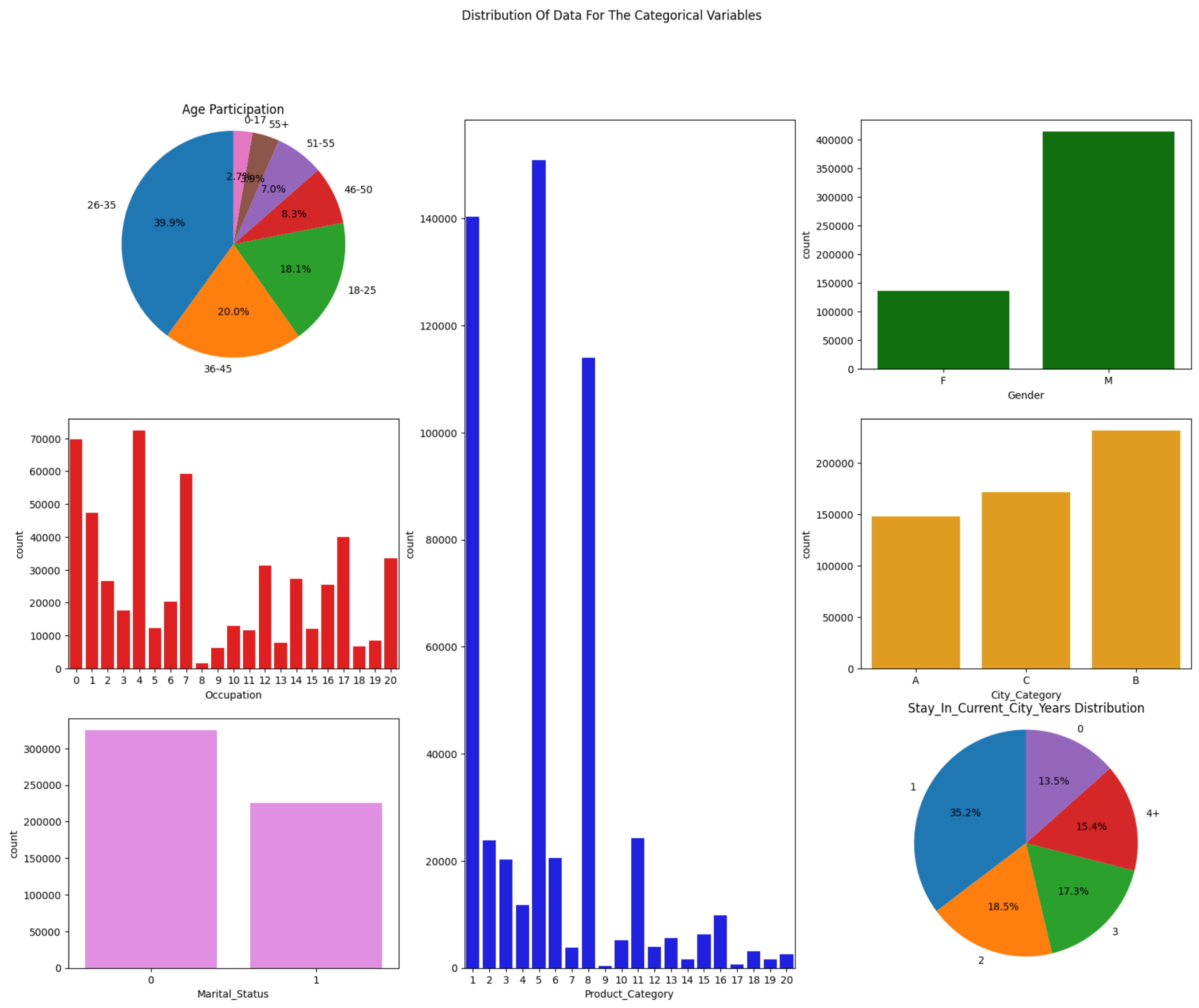
1. We can see around 6-8k, users have made purchases most of the products.
2. We can find out from around ~23k, outliers have started and the number of outliers is 2677. Where unique outliers are 1027.

Now will understand the distribution of data of the categorical variables.

- Gender
- Age
- Occupation
- City_Category
- Stay_In_Current_City_Years
- Marital_Status
- Product_Category

```
plt.figure(figsize=(20, 15))
plt.subplot(3, 3, 1)
data = df['Age'].value_counts(normalize=True)*100
plt.pie(data, labels=data.index, autopct='%1.1f%%', startangle=90)
plt.title('Age Participation')
plt.axis('equal')
```

```python
plt.subplot(3, 3, 3)
sns.countplot(data = df, x ='Gender', color = 'green')

plt.subplot(3, 3, 4)
sns.countplot(data = df, x ='Occupation', color = 'red')

plt.subplot(3, 3, 6)
sns.countplot(data = df, x ='City_Category', color = 'orange')

plt.subplot(3, 3, 9)
data1 = df['Stay_In_Current_City_Years'].value_counts(normalize=True)*100
plt.pie(data1, labels=data1.index, autopct='%1.1f%%', startangle=90)
plt.title('Stay_In_Current_City_Years Distribution')
plt.axis('equal')


plt.subplot(3, 3, 7)
sns.countplot(data = df, x ='Marital_Status', color = 'violet')

plt.subplot(1, 3, 2)
sns.countplot(data = df, x ='Product_Category', color = 'blue')

plt.suptitle("Distribution Of Data For The Categorical Variables")
plt.show()
```

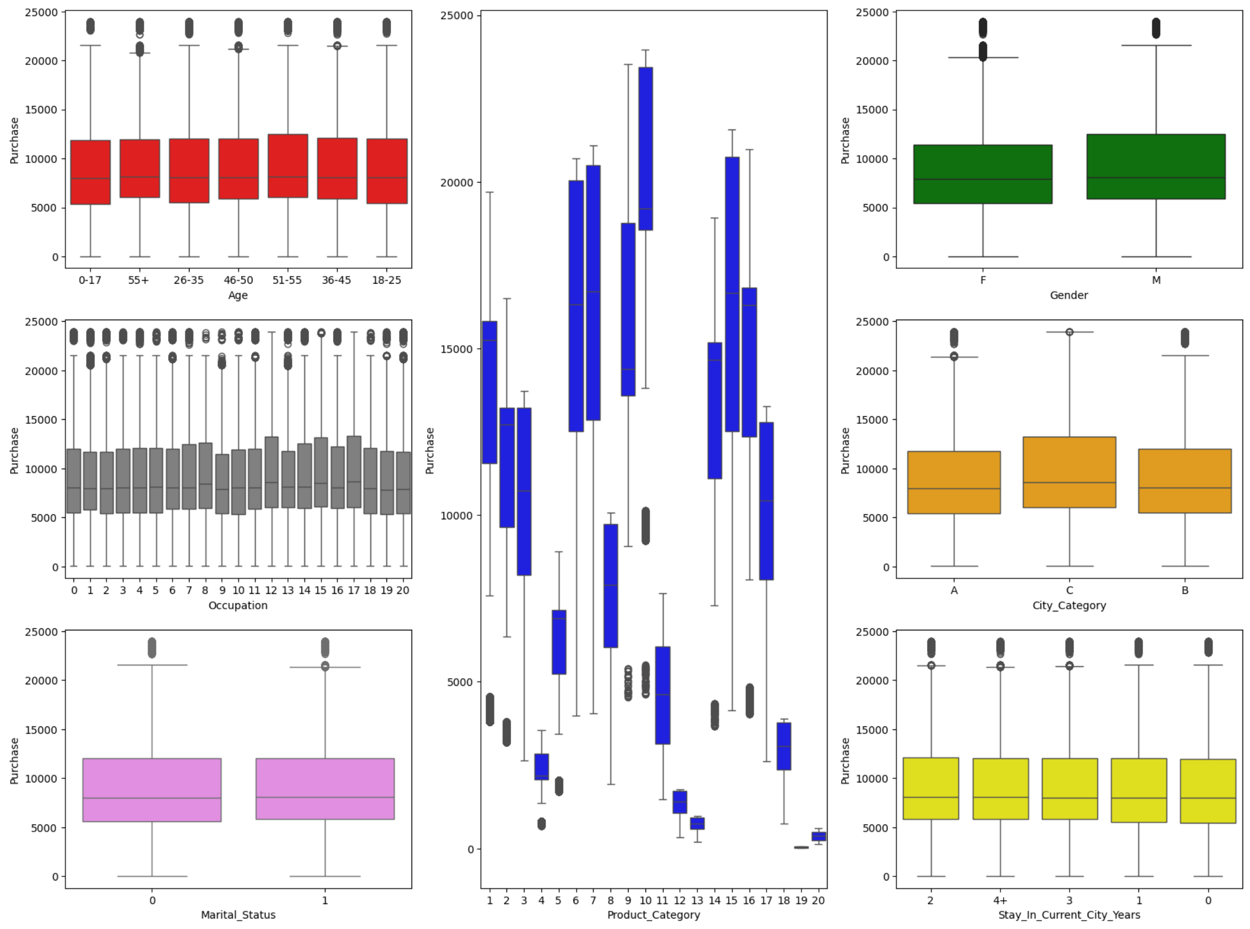


Observations

1. ~ 80% of the users are between the age 18-50 (40%: 26-35, 18%: 18-25, 20%: 36-45)
2. Most of the users in Walmart are male.
3. We can notice there are 20 different types of occupation and product present over there.
4. Most of users came from city B, follows C and A.
5. Bachelors love to spend time and buy product in Walmart. Count of single shows high as compared to couple or family.
6. Category of product 5,1 and 8 seems high.

Bivariate Analysis

```
plt.figure(figsize=(20, 15))
plt.subplot(3, 3, 1)
sns.boxplot(data = df, x ='Age', y = 'Purchase', color = 'red')

plt.subplot(3, 3, 3)
sns.boxplot(data = df, x ='Gender',y = 'Purchase', color = 'green')

plt.subplot(3, 3, 4)
sns.boxplot(data = df, x ='Occupation',y = 'Purchase', color = 'grey')

plt.subplot(3, 3, 6)
sns.boxplot(data = df, x ='City_Category',y = 'Purchase', color =
'orange')

plt.subplot(3, 3, 9)
sns.boxplot(data = df, x ='Stay_In_Current_City_Years',y = 'Purchase',
color = 'yellow')


plt.subplot(3, 3, 7)
sns.boxplot(data = df, x ='Marital_Status',y = 'Purchase', color =
'violet')

plt.subplot(1, 3, 2)
sns.boxplot(data = df, x ='Product_Category',y = 'Purchase', color =
'blue')

plt.suptitle("Distribution Of Data In Categorical Variables Against of
Continious Variable")
plt.show()
```

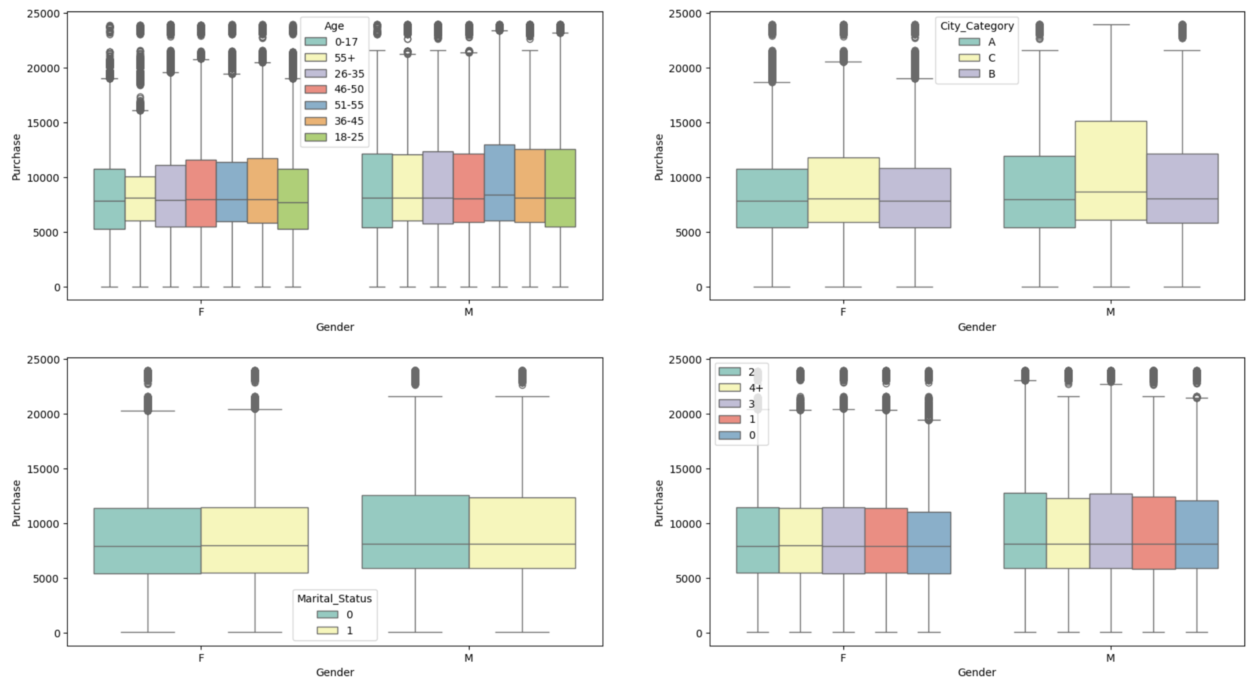


Distribution Of Data In Categorical Variables Against of Continious Variable

Multivariate Analysis

```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20, 6))
fig.subplots_adjust(top=1.5)
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Age', palette='Set3',
ax=axs[0,0])
sns.boxplot(data=df, y='Purchase', x='Gender', hue='City_Category',
palette='Set3', ax=axs[0,1])

sns.boxplot(data=df, y='Purchase', x='Gender', hue='Marital_Status',
palette='Set3', ax=axs[1,0])
```

```
sns.boxplot(data=df, y='Purchase', x='Gender',
hue='Stay_In_Current_City_Years', palette='Set3', ax=axs[1,1])
axs[1,1].legend(loc='upper left')

plt.show()
```



Now trying to check below requirements,

- 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.

```
# Trying to figure it out first transaction/purchase for each user id
based on their gender.
# creating a new df and storing the result of each customers purchase

new_df_amount = df.groupby(['User_ID',
'Gender'])['Purchase'].sum().reset_index()
```

```
new_df_amount
```

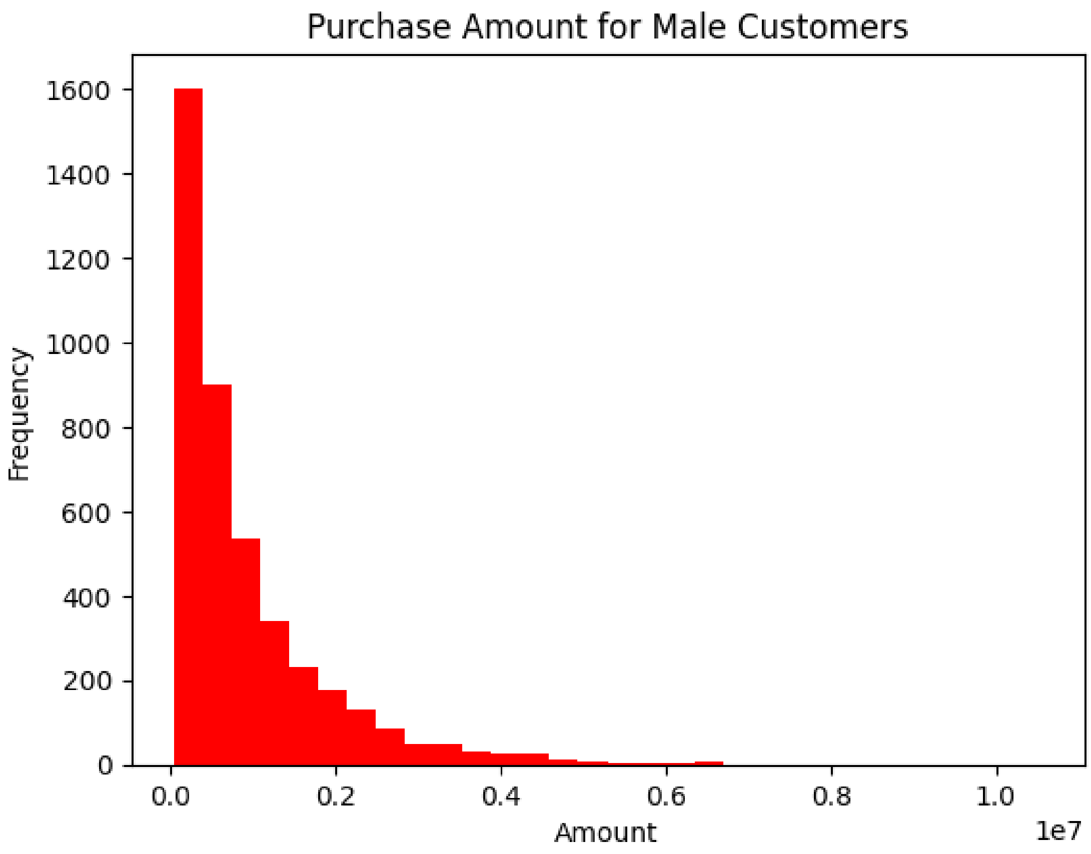
	User_ID	Gender	Purchase
0	1000001	F	334093
1	1000002	M	810472
2	1000003	M	341635
3	1000004	M	206468
4	1000005	M	821001
...	...	...	...
5886	1006036	F	4116058
5887	1006037	F	1119538
5888	1006038	F	90034
5889	1006039	F	590319
5890	1006040	M	1653299

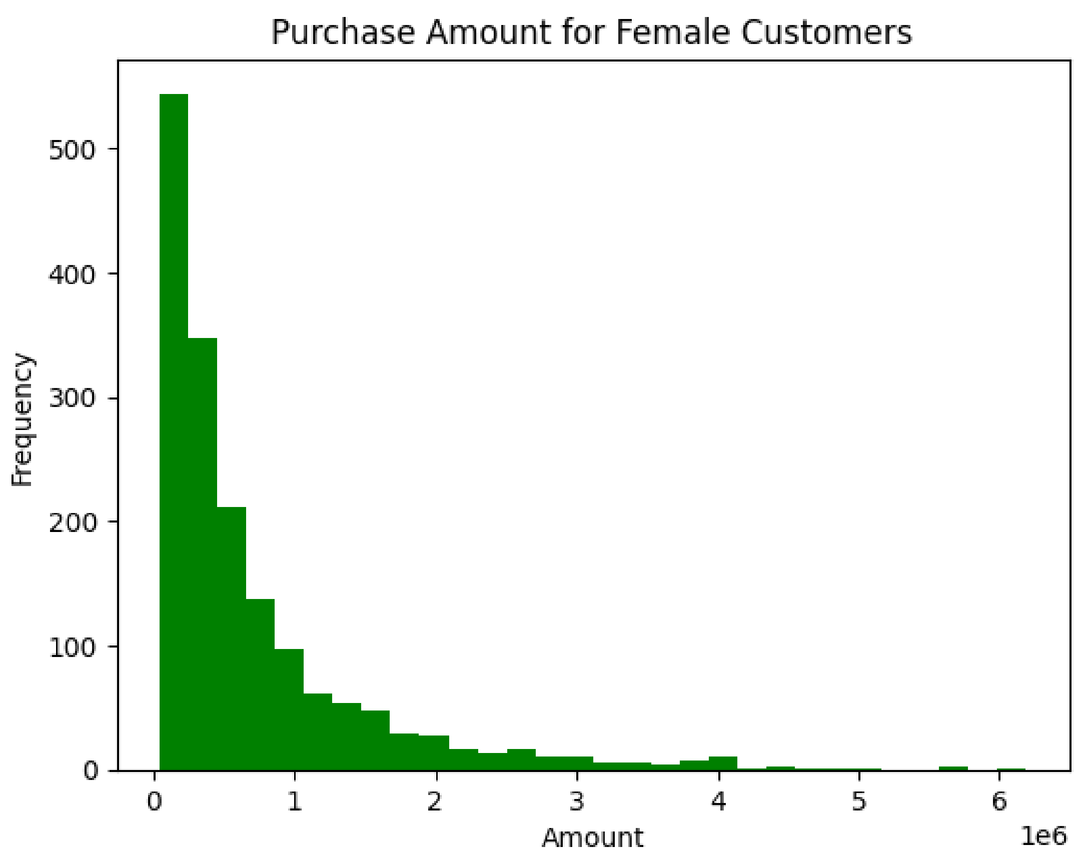
5891 rows × 3 columns

```
# trying to figure out the number of customers based on the male/female

customers = new_df_amount['Gender'].value_counts()
customers
```

```
M    4225
F    1666
Name: Gender, dtype: int64
```

```
# histogram of average amount spent for each customer - Male & Female
plt.figure()  # Create a new figure

plt.hist(new_df_amount[new_df_amount['Gender']=='M']['Purchase'],color =
'red', bins=30)
plt.title("Purchase Amount for Male Customers")
plt.xlabel("Amount")
plt.ylabel("Frequency")
plt.show()

plt.figure()  # Create a new figure
plt.hist(new_df_amount[new_df_amount['Gender']=='F']['Purchase'],color =
'green', bins=30)
plt.title("Purchase Amount for Female Customers")
plt.xlabel("Amount")
plt.ylabel("Frequency")
plt.show()
```





```
# trying to sort the purchase amount based on the gender
# storing the average value based on gender
#taking the round value

avg_amount =
new_df_amount.groupby(['Gender'])['Purchase'].mean().round().to_frame()
avg_amount
```

	Purchase
Gender	
F	712024.0
M	925344.0

Observations

1. Can see male customers made more shopping than the female customers.

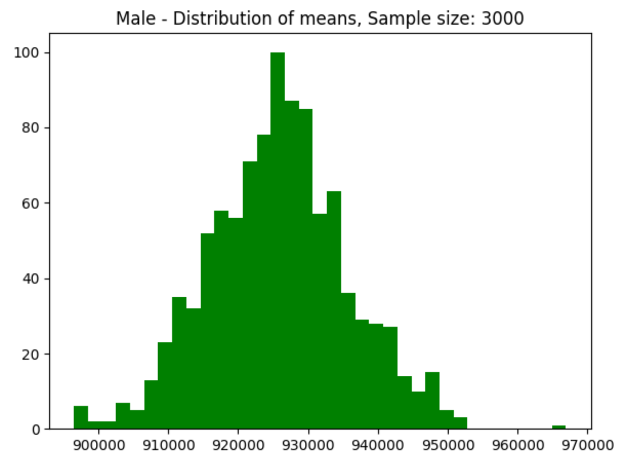
2. Number of Male customers are 4225 which is better than female, i.e 1666.
3. Avg purchased amount of male is 925344. And for the female it's bit low, 712024.

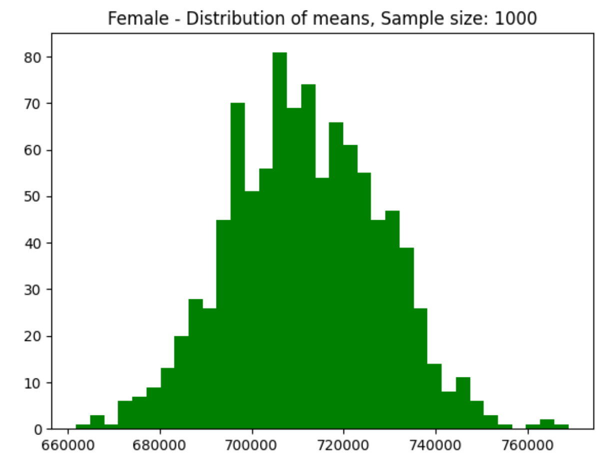
Now need to work on,

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.

```
# Differing male and female first

male_customer = new_df_amount[new_df_amount['Gender'] == 'M']
female_customer = new_df_amount[new_df_amount['Gender'] == 'F']

# taking some random samples
male_sample_size = 3000
female_sample_size = 1000
no_of_time_check_sample = 1000

# assigning empty bucket
male_sample_mean = []
female_sample_mean = []

# running a loop to collect 1000 different samples for both male and
female
for i in range(no_of_time_check_sample):

male_sample_mean.append(male_customer.sample(male_sample_size)['Purchase']
.mean())

female_sample_mean.append(female_customer.sample(female_sample_size)['Purc
hase'].mean())

# now try to show the mean result with help of histogram

plt.figure(figsize=(15, 5))

plt.subplot(1, 2, 1)
plt.hist(male_sample_mean, bins = 35, color = 'green')
plt.title("Male - Distribution of means, Sample size: 3000")

plt.subplot(1, 2, 2)
plt.hist(female_sample_mean, bins = 35, color = 'green')
plt.title("Female - Distribution of means, Sample size: 1000")

plt.show()
```





```
print(f"Mean of sample means of amount spend for Male:
{round((np.mean(male_sample_mean)))}")
print(f"Mean of sample means of amount spend for Female:
{round((np.mean(female_sample_mean)))}")

print(f"Actual mean of Male customers were :
{round(male_customer['Purchase'].mean())} && Standard Deviation :
{round(male_customer['Purchase'].std())}")
print(f"Actual mean of Female customers were :
{round(female_customer['Purchase'].mean())} && Standard Deviation :
{round(female_customer['Purchase'].std())}")
```

```
Mean of sample means of amount spend for Male: 925799
Mean of sample means of amount spend for Female: 711756
Actual mean of Male customers were : 925344 && Standard Deviation : 985830

Actual mean of Female customers were : 712024 && Standard Deviation :
807371
```

Observations

Now using the **Central Limit Theorem** for the **population** we can say that:

1. After taking 3000 male and 1000 female sample and took 1000 times of sample means we got to know, sample mean of male is 925799 and female is 711756.
2. And actuals mean were for male 925344 and for female 712024. Which is almost similar for both testing.

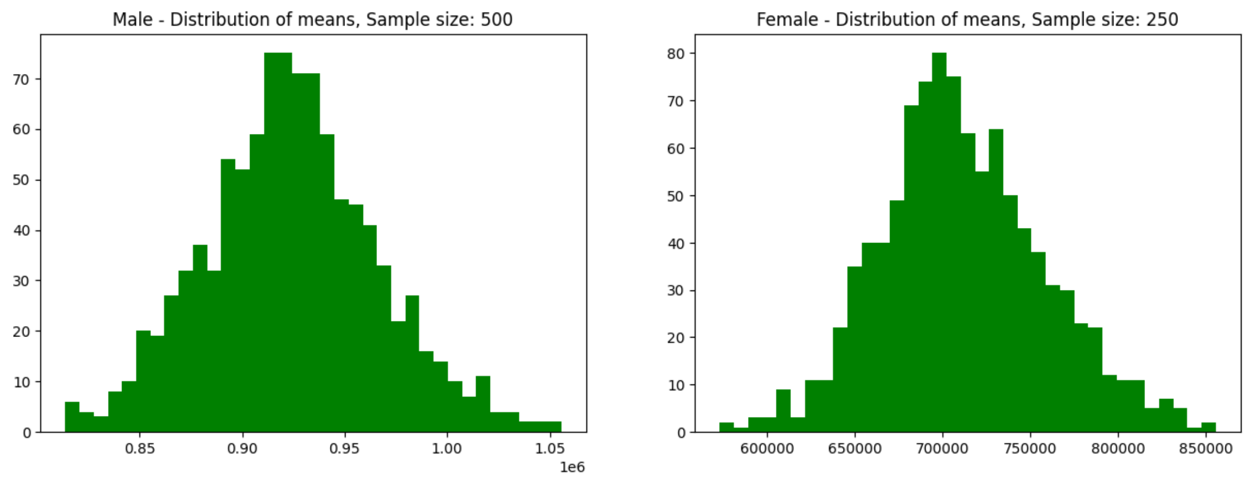
Now we will check with some small amount of sample.

Sample Size,

For Male : 500

For Female : 250

```
# Differing male and female first

male_customer = new_df_amount[new_df_amount['Gender'] == 'M']
female_customer = new_df_amount[new_df_amount['Gender'] == 'F']

# taking some random samples
male_sample_size = 500
female_sample_size = 250
no_of_time_check_sample = 1000

# assigning empty bucket
male_sample_mean = []
female_sample_mean = []

# running a loop to collect 1000 different samples for both male and
female
for i in range(no_of_time_check_sample):

male_sample_mean.append(male_customer.sample(male_sample_size)['Purchase']
.mean())

female_sample_mean.append(female_customer.sample(female_sample_size)['Purc
hase'].mean())

# now try to show the mean result with help of histogram

plt.figure(figsize=(15, 5))

plt.subplot(1, 2, 1)
plt.hist(male_sample_mean, bins = 35, color = 'green')
plt.title("Male - Distribution of means, Sample size: 500")

plt.subplot(1, 2, 2)
plt.hist(female_sample_mean, bins = 35, color = 'green')
plt.title("Female - Distribution of means, Sample size: 250")

plt.show()
```



```
print(f"Mean of sample means of amount spend for Male:
{round((np.mean(male_sample_mean)))}")
print(f"Mean of sample means of amount spend for Female:
{round((np.mean(female_sample_mean)))}")

print(f"Actual mean of Male customers were :
{round(male_customer['Purchase'].mean())} && Standard Deviation :
{round(male_customer['Purchase'].std())}")
print(f"Actual mean of Female customers were :
{round(female_customer['Purchase'].mean())} && Standard Deviation :
{round(female_customer['Purchase'].std())}")
```

```
Mean of sample means of amount spend for Male: 923697
Mean of sample means of amount spend for Female: 712069
Actual mean of Male customers were : 925344 && Standard Deviation : 985830

Actual mean of Female customers were : 712024 && Standard Deviation :
807371
```

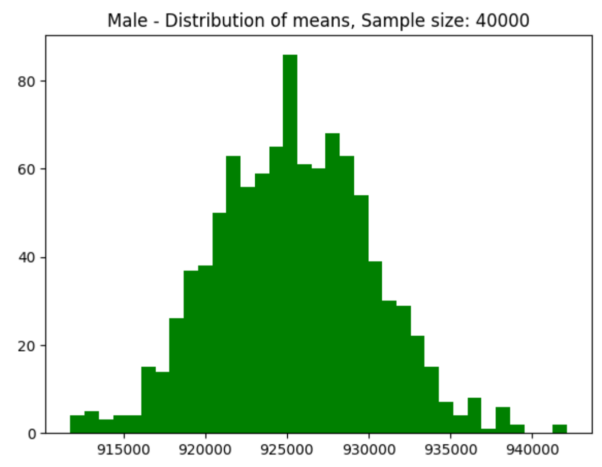
Now we will check with some bigger amount of sample.

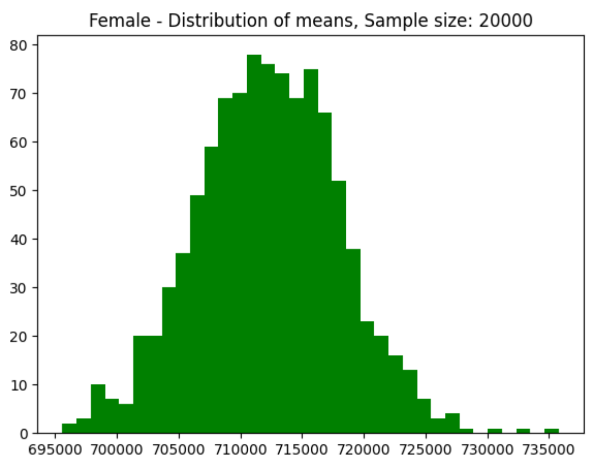
Sample Size,

For Male : 40000

For Female : 20000

```
# Differing male and female first

male_customer = new_df_amount[new_df_amount['Gender'] == 'M']
female_customer = new_df_amount[new_df_amount['Gender'] == 'F']

# taking some random samples
male_sample_size = 40000
female_sample_size = 20000
no_of_time_check_sample = 1000

# assigning empty bucket
male_sample_mean = []
female_sample_mean = []

# running a loop to collect 1000 different samples for both male and
female
for i in range(no_of_time_check_sample):
  male_sample_mean.append(male_customer.sample(male_sample_size,
replace=True)['Purchase'].mean())

female_sample_mean.append(female_customer.sample(female_sample_size,replac
e=True)['Purchase'].mean())

# now try to show the mean result with help of histogram

plt.figure(figsize=(15, 5))

plt.subplot(1, 2, 1)
plt.hist(male_sample_mean, bins = 35, color = 'green')
plt.title("Male - Distribution of means, Sample size: 40000")

plt.subplot(1, 2, 2)
plt.hist(female_sample_mean, bins = 35, color = 'green')
plt.title("Female - Distribution of means, Sample size: 20000")

plt.show()
```





```
print(f"Mean of sample means of amount spend for Male:
{round((np.mean(male_sample_mean)))}")
print(f"Mean of sample means of amount spend for Female:
{round((np.mean(female_sample_mean)))}")

print(f"Actual mean of Male customers were :
{round(male_customer['Purchase'].mean())} && Standard Deviation :
{round(male_customer['Purchase'].std())}")
print(f"Actual mean of Female customers were :
{round(female_customer['Purchase'].mean())} && Standard Deviation :
{round(female_customer['Purchase'].std())}")
```

```
Mean of sample means of amount spend for Male: 925408
Mean of sample means of amount spend for Female: 712243
Actual mean of Male customers were : 925344 && Standard Deviation : 985830

Actual mean of Female customers were : 712024 && Standard Deviation :
807371
```

Observations

Samples Mean for Male,

Sample Size : 500 → 923697

Sample Size : 3000 → 925799

Sample Size : 40000 → 925408

Samples Mean for Feale,

Sample Size : 250 → 712069

Sample Size : 1000 → 711756

Sample Size : 20000 → 712243

We can see sample mean are almost same for taking different samples.

Now,

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.

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?

Confidence interval → Z

99% Confidence Interval:

```
#99% Confidence Interval

from scipy.stats import norm

# find the Z value with help of norm
Z = norm.ppf(.99)

sample_mean_male = male_customer['Purchase'].mean()
sample_mean_female = female_customer['Purchase'].mean()

Standerd_Error_male =
male_customer['Purchase'].std()/np.sqrt(len(male_customer))
Standerd_Error_female =
female_customer['Purchase'].std()/np.sqrt(len(female_customer))

left_lim_male = (sample_mean_male - Z * Standerd_Error_male)
left_lim_female = (sample_mean_female - Z * Standerd_Error_female)

right_lim_male = (sample_mean_male + Z * Standerd_Error_male)
right_lim_female = (sample_mean_female + Z * Standerd_Error_female)

print("99% Confidence Interval:")
print(f"Male confidence interval of means:
{round(left_lim_male),round(right_lim_male)}")
```

```
print(f"Female confidence interval of means:
{round(left_lim_female),round(right_lim_female)}")
99% Confidence Interval:
Male confidence interval of means: (890062, 960627)

Female confidence interval of means: (666008, 758041)
```

Observations

For 99% Confidence Interval, the range for male & female is not overlapping.

Now we can infer about the population that, 99% of the times:

Average amounts spend by male customer will lie in between: (890062, 960627)

Average amounts spend by female customer will lie in between: (666008, 758041)

95% Confidence Interval:

```
#95% Confidence Interval

from scipy.stats import norm

# find the Z value with help of norm
Z = norm.ppf(.95)

sample_mean_male = male_customer['Purchase'].mean()
sample_mean_female = female_customer['Purchase'].mean()

Standerd_Error_male =
male_customer['Purchase'].std()/np.sqrt(len(male_customer))
Standerd_Error_female =
female_customer['Purchase'].std()/np.sqrt(len(female_customer))

left_lim_male = (sample_mean_male - Z * Standerd_Error_male)
left_lim_female = (sample_mean_female - Z * Standerd_Error_female)

right_lim_male = (sample_mean_male + Z * Standerd_Error_male)
right_lim_female = (sample_mean_female + Z * Standerd_Error_female)

print("95% Confidence Interval:")
print(f"Male confidence interval of means:
{round(left_lim_male),round(right_lim_male)}")
print(f"Female confidence interval of means:
{round(left_lim_female),round(right_lim_female)}")
95% Confidence Interval:
```

```
Male confidence interval of means: (900398, 950291)

Female confidence interval of means: (679489, 744560)
```

Observations

For 95% Confidence Interval, the range for male & female is not overlapping.

Now we can infer about the population that, 95% of the times:

Average amounts spend by male customer will lie in between: (900398, 950291)

Average amounts spend by female customer will lie in between: (679489, 744560)

90% Confidence Interval:

```
#90% Confidence Interval

from scipy.stats import norm

# find the Z value with help of norm
Z = norm.ppf(.90)

sample_mean_male = male_customer['Purchase'].mean()
sample_mean_female = female_customer['Purchase'].mean()

Standerd_Error_male =
male_customer['Purchase'].std()/np.sqrt(len(male_customer))
Standerd_Error_female =
female_customer['Purchase'].std()/np.sqrt(len(female_customer))

left_lim_male = (sample_mean_male - Z * Standerd_Error_male)
left_lim_female = (sample_mean_female - Z * Standerd_Error_female)

right_lim_male = (sample_mean_male + Z * Standerd_Error_male)
right_lim_female = (sample_mean_female + Z * Standerd_Error_female)

print("90% Confidence Interval:")
print(f"Male confidence interval of means:
{round(left_lim_male),round(right_lim_male)}")
print(f"Female confidence interval of means:
{round(left_lim_female),round(right_lim_female)}")
```

```
90% Confidence Interval:
Male confidence interval of means: (905908, 944781)
```

```
Female confidence interval of means: (686675, 737374)
```

Observations

For 90% Confidence Interval, the range for male & female is not overlapping.

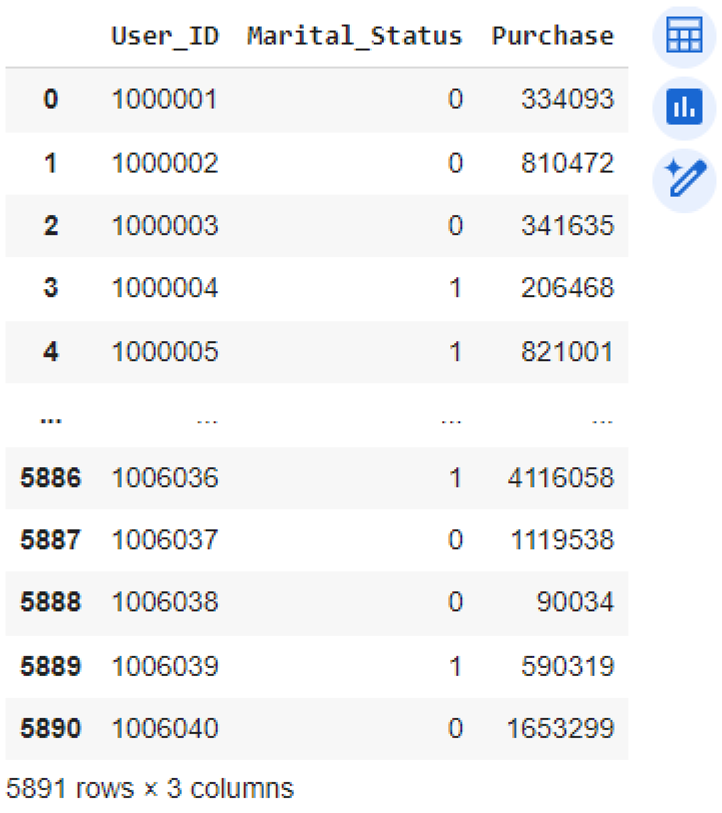
Now we can infer about the population that, 90% of the times:

Average amounts spend by male customer will lie in between: (905908, 944781)

Average amounts spend by female customer will lie in between: (686675, 737374)

Perform the same activity for Married vs Unmarried.

```
# Trying to figure it out first transaction/purchase for each user id
based on their Marital_Status.
# creating a new df and storing the result of each customers purchase

new_df_amount = df.groupby(['User_ID',
'Marital_Status'])['Purchase'].sum().reset_index()
new_df_amount
```



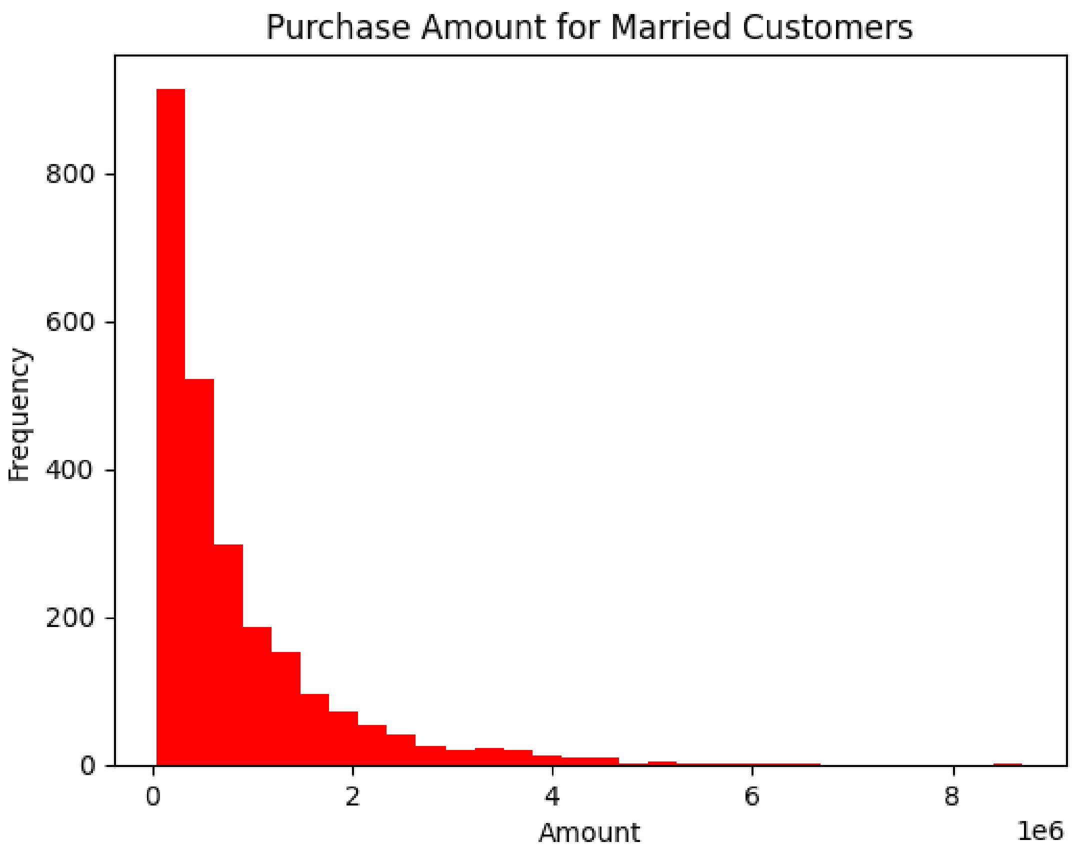
	User_ID	Marital_Status	Purchase
0	1000001	0	334093
1	1000002	0	810472
2	1000003	0	341635
3	1000004	1	206468
4	1000005	1	821001
...	...	...	...
5886	1006036	1	4116058
5887	1006037	0	1119538
5888	1006038	0	90034
5889	1006039	1	590319
5890	1006040	0	1653299

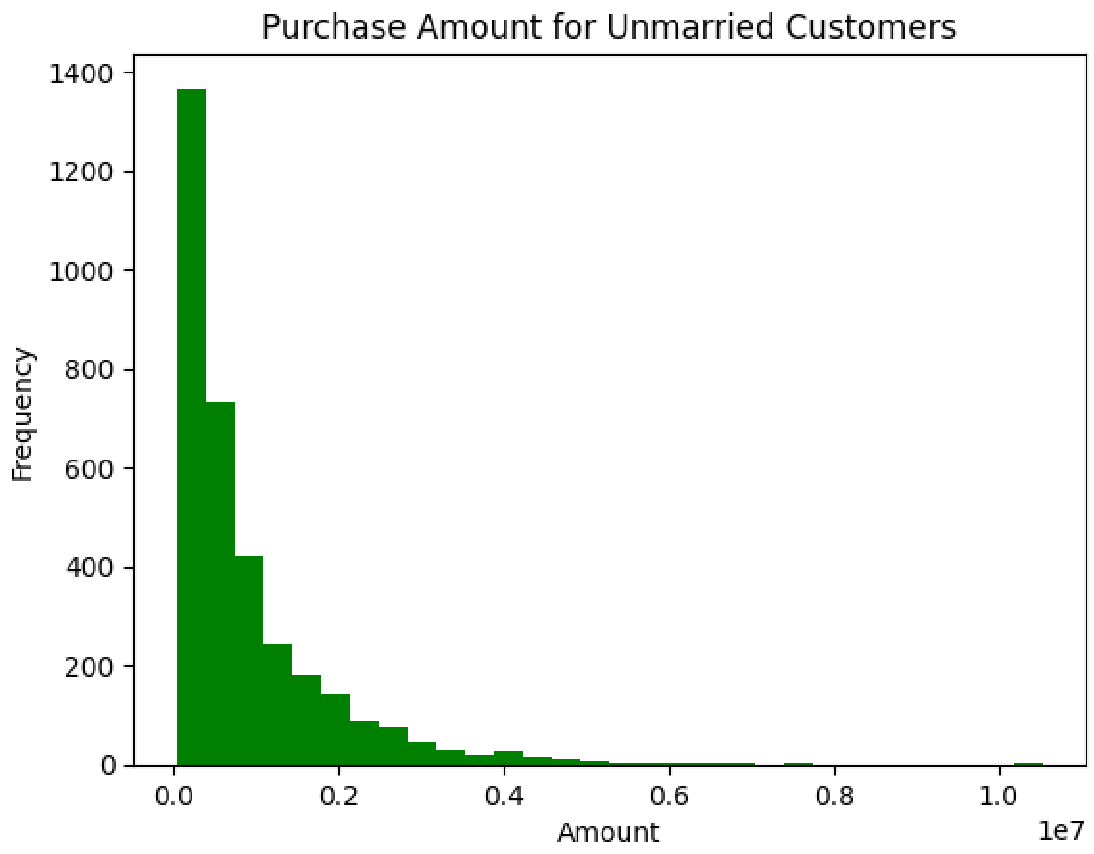
5891 rows × 3 columns

```
# trying to figure out the number of customers based on the Marital_Status

customers = new_df_amount['Marital_Status'].value_counts()
customers
```

```
0   3417
1   2474
Name: Marital_Status, dtype: int64
```

```
# histogram of average amount spent for each customer - Married &
Unmarried
plt.figure()  # Create a new figure

plt.hist(new_df_amount[new_df_amount['Marital_Status']==1]['Purchase'],col
or = 'red', bins=30)
plt.title("Purchase Amount for Married Customers")
plt.xlabel("Amount")
plt.ylabel("Frequency")
plt.show()

plt.figure()  # Create a new figure

plt.hist(new_df_amount[new_df_amount['Marital_Status']==0]['Purchase'],col
or = 'green', bins=30)
plt.title("Purchase Amount for Unmarried Customers")
plt.xlabel("Amount")
plt.ylabel("Frequency")
plt.show()
```


Purchase Amount for Married Customers
Frequency
0
200
400
600
800
Amount
0
2
4
6
8
1e6



```
# trying to sort the purchase amount based on the Marital_Status
# storing the average value based on Marital_Status
#taking the round value

avg_amount =
new_df_amount.groupby(['Marital_Status'])['Purchase'].mean().round().to_fr
ame()
avg_amount
```

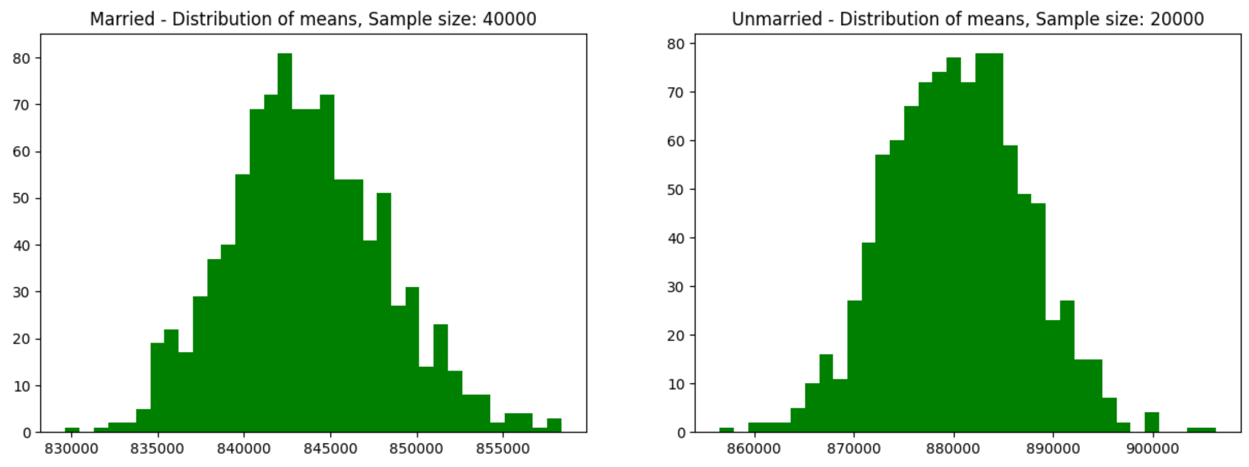
Sample Size,

Married → 40000

Unmarried → 20000

```
# Differing Married and Unmarried first

married_customer = new_df_amount[new_df_amount['Marital_Status'] == 1]
unmarried_customer = new_df_amount[new_df_amount['Marital_Status'] == 0]
```

```
# taking some random samples
married_sample_size = 40000
unmarried_sample_size = 20000
no_of_time_check_sample = 1000

# assigning empty bucket
married_sample_mean = []
unmarried_sample_mean = []

# running a loop to collect 1000 different samples for both married and
unmarried
for i in range(no_of_time_check_sample):
  married_sample_mean.append(married_customer.sample(married_sample_size,
replace=True)['Purchase'].mean())

unmarried_sample_mean.append(unmarried_customer.sample(unmarried_sample_si
ze,replace=True)['Purchase'].mean())

# now try to show the mean result with help of histogram

plt.figure(figsize=(15, 5))

plt.subplot(1, 2, 1)
plt.hist(married_sample_mean, bins = 35, color = 'green')
plt.title("Married - Distribution of means, Sample size: 40000")

plt.subplot(1, 2, 2)
plt.hist(unmarried_sample_mean, bins = 35, color = 'green')
plt.title("Unmarried - Distribution of means, Sample size: 20000")

plt.show()
```



```
Married: {round((np.mean(married_sample_mean)))}")
print(f"Mean of sample means of amount spend for Unmarried:
{round((np.mean(unmarried_sample_mean)))}")

print(f"Actual mean of Married customers were :
{round(married_customer['Purchase'].mean())} && Standard Deviation :
{round(married_customer['Purchase'].std())}")
print(f"Actual mean of Unmarried customers were :
{round(unmarried_customer['Purchase'].mean())} && Standard Deviation :
{round(unmarried_customer['Purchase'].std())}")
```

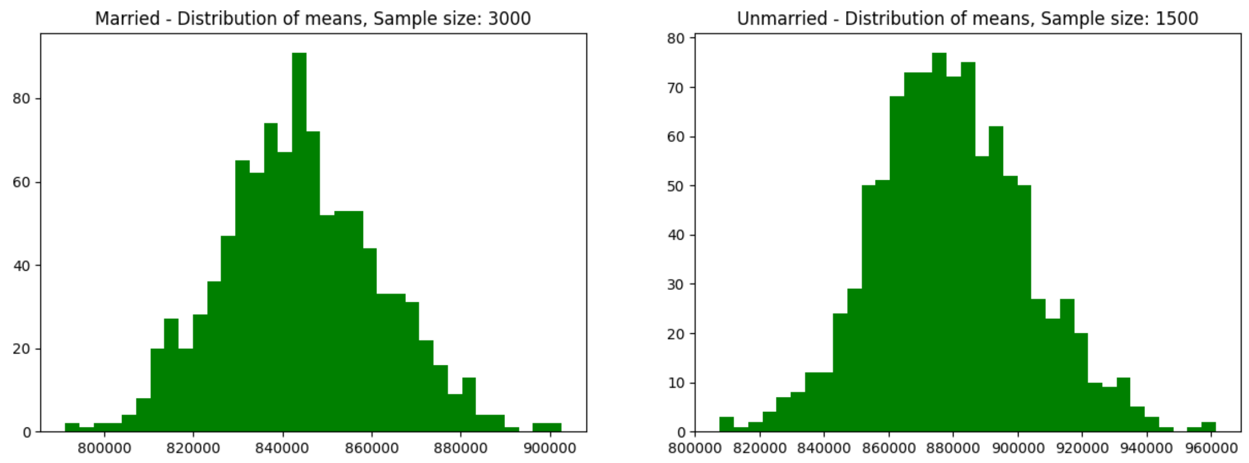
```
Mean of sample means of amount spend for Married: 843371
Mean of sample means of amount spend for Unmarried: 880658
Actual mean of Married customers were : 843527 && Standard Deviation :
935352

Actual mean of Unmarried customers were : 880576 && Standard Deviation :
949436
```

Married → 3000

Unmarried → 1500

```
# Differing Married and Unmarried first

married_customer = new_df_amount[new_df_amount['Marital_Status'] == 1]
unmarried_customer = new_df_amount[new_df_amount['Marital_Status'] == 0]

# taking some random samples
married_sample_size = 3000
unmarried_sample_size = 1500
no_of_time_check_sample = 1000

# assigning empty bucket
married_sample_mean = []
unmarried_sample_mean = []

# running a loop to collect 1000 different samples for both married and
unmarried
for i in range(no_of_time_check_sample):
  married_sample_mean.append(married_customer.sample(married_sample_size,
replace=True)['Purchase'].mean())

unmarried_sample_mean.append(unmarried_customer.sample(unmarried_sample_si
ze,replace=True)['Purchase'].mean())
```

```
# now try to show the mean result with help of histogram

plt.figure(figsize=(15, 5))

plt.subplot(1, 2, 1)
plt.hist(married_sample_mean, bins = 35, color = 'green')
plt.title("Married - Distribution of means, Sample size: 3000")

plt.subplot(1, 2, 2)
plt.hist(unmarried_sample_mean, bins = 35, color = 'green')
plt.title("Unmarried - Distribution of means, Sample size: 1500")

plt.show()
```



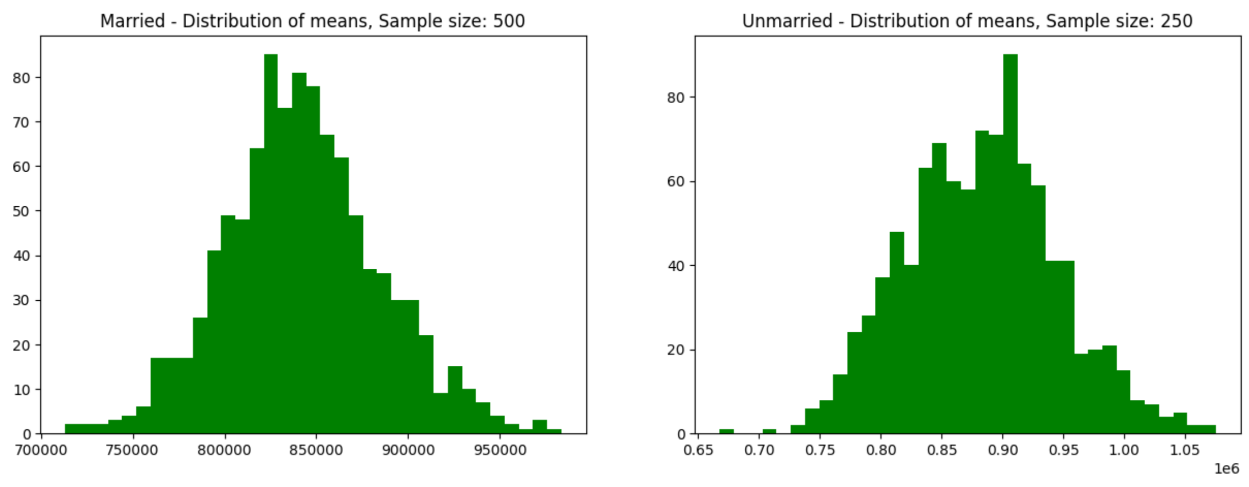
```
Married: {round((np.mean(married_sample_mean)))}")
print(f"Mean of sample means of amount spend for Unmarried:
{round((np.mean(unmarried_sample_mean)))}")

print(f"Actual mean of Married customers were :
{round(married_customer['Purchase'].mean())} && Standard Deviation :
{round(married_customer['Purchase'].std())}")
print(f"Actual mean of Unmarried customers were :
{round(unmarried_customer['Purchase'].mean())} && Standard Deviation :
{round(unmarried_customer['Purchase'].std())}")
```

```
Mean of sample means of amount spend for Married: 843682
Mean of sample means of amount spend for Unmarried: 880645
Actual mean of Married customers were : 843527 && Standard Deviation :
935352

Actual mean of Unmarried customers were : 880576 && Standard Deviation :
949436
```

Married → 500

Unmarried → 250

```
# Differing Married and Unmarried first

married_customer = new_df_amount[new_df_amount['Marital_Status'] == 1]
unmarried_customer = new_df_amount[new_df_amount['Marital_Status'] == 0]

# taking some random samples
married_sample_size = 500
unmarried_sample_size = 250
no_of_time_check_sample = 1000

# assigning empty bucket
married_sample_mean = []
unmarried_sample_mean = []

# running a loop to collect 1000 different samples for both married and
unmarried
for i in range(no_of_time_check_sample):
  married_sample_mean.append(married_customer.sample(married_sample_size,
replace=True)['Purchase'].mean())

unmarried_sample_mean.append(unmarried_customer.sample(unmarried_sample_si
ze,replace=True)['Purchase'].mean())

# now try to show the mean result with help of histogram

plt.figure(figsize=(15, 5))

plt.subplot(1, 2, 1)
plt.hist(married_sample_mean, bins = 35, color = 'green')
plt.title("Married - Distribution of means, Sample size: 500")

plt.subplot(1, 2, 2)
plt.hist(unmarried_sample_mean, bins = 35, color = 'green')
plt.title("Unmarried - Distribution of means, Sample size: 250")

plt.show()
```



```
print(f"Mean of sample means of amount spend for Married:
{round((np.mean(married_sample_mean)))}")
print(f"Mean of sample means of amount spend for Unmarried:
{round((np.mean(unmarried_sample_mean)))}")

print(f"Actual mean of Married customers were :
{round(married_customer['Purchase'].mean())} && Standard Deviation :
{round(married_customer['Purchase'].std())}")
print(f"Actual mean of Unmarried customers were :
{round(unmarried_customer['Purchase'].mean())} && Standard Deviation :
{round(unmarried_customer['Purchase'].std())}")
```

```
Mean of sample means of amount spend for Married: 842872
Mean of sample means of amount spend for Unmarried: 882319
Actual mean of Married customers were : 843527 && Standard Deviation :
935352

Actual mean of Unmarried customers were : 880576 && Standard Deviation :
949436
```

Observations

Samples Mean for married,

Sample Size : 500 → 842872

Sample Size : 3000 → 843682

Sample Size : 40000 → 843371

Samples Mean for Unmarried,

Sample Size : 250 → 882319

Sample Size : 1500 → 880645

Sample Size : 20000 → 880658

We can see sample mean are almost same for taking different samples.

Confidence Interval -> Z

99% Confidence Interval:

```
#99% Confidence Interval

from scipy.stats import norm

# find the Z value with help of norm
Z = norm.ppf(.99)

sample_mean_married = married_customer['Purchase'].mean()
sample_mean_unmarried = unmarried_customer['Purchase'].mean()

Standerd_Error_married =
married_customer['Purchase'].std()/np.sqrt(len(married_customer))
Standerd_Error_unmarried =
unmarried_customer['Purchase'].std()/np.sqrt(len(unmarried_customer))

left_lim_married = (sample_mean_married - Z * Standerd_Error_married)
left_lim_unmarried = (sample_mean_unmarried - Z *
Standerd_Error_unmarried)

right_lim_married = (sample_mean_married + Z * Standerd_Error_married)
right_lim_unmarried = (sample_mean_unmarried + Z *
Standerd_Error_unmarried)

print("99% Confidence Interval:")
print(f"Married confidence interval of means:
{round(left_lim_married),round(right_lim_married)}")
print(f"Unmarried confidence interval of means:
{round(left_lim_unmarried),round(right_lim_unmarried)}")
```

```
99% Confidence Interval:


Married confidence interval of means: (799780, 887274)

Unmarried confidence interval of means: (842791, 918361)
```

Observations

For 99% Confidence Interval, the range for married & unmarried is overlapping.

We need to reduce the confidence interval to 95% and lets check.

95% Confidence Interval:

```
#95% Confidence Interval

from scipy.stats import norm

# find the Z value with help of norm
Z = norm.ppf(.95)

sample_mean_married = married_customer['Purchase'].mean()
sample_mean_unmarried = unmarried_customer['Purchase'].mean()

Standerd_Error_married =
married_customer['Purchase'].std()/np.sqrt(len(married_customer))
Standerd_Error_unmarried =
unmarried_customer['Purchase'].std()/np.sqrt(len(unmarried_customer))

left_lim_married = (sample_mean_married - Z * Standerd_Error_married)
left_lim_unmarried = (sample_mean_unmarried - Z *
Standerd_Error_unmarried)

right_lim_married = (sample_mean_married + Z * Standerd_Error_married)
right_lim_unmarried = (sample_mean_unmarried + Z *
Standerd_Error_unmarried)

print("95% Confidence Interval:")
print(f"Married confidence interval of means:
{round(left_lim_married),round(right_lim_married)}")
print(f"Unmarried confidence interval of means:
{round(left_lim_unmarried),round(right_lim_unmarried)}")
```

```
95% Confidence Interval:
Married confidence interval of means: (812595, 874458)

Unmarried confidence interval of means: (853860, 907292)
```

Observations

For 95% Confidence Interval, the range for married & unmarried is overlapping.

We need to reduce the confidence interval to 90% and lets check.

90% Confidence Interval:

```
#90% Confidence Interval

from scipy.stats import norm

# find the Z value with help of norm
Z = norm.ppf(.90)

sample_mean_married = married_customer['Purchase'].mean()
sample_mean_unmarried = unmarried_customer['Purchase'].mean()

Standerd_Error_married =
married_customer['Purchase'].std()/np.sqrt(len(married_customer))
Standerd_Error_unmarried =
unmarried_customer['Purchase'].std()/np.sqrt(len(unmarried_customer))

left_lim_married = (sample_mean_married - Z * Standerd_Error_married)
left_lim_unmarried = (sample_mean_unmarried - Z *
Standerd_Error_unmarried)

right_lim_married = (sample_mean_married + Z * Standerd_Error_married)
right_lim_unmarried = (sample_mean_unmarried + Z *
Standerd_Error_unmarried)

print("90% Confidence Interval:")
print(f"Married confidence interval of means:
{round(left_lim_married),round(right_lim_married)}")
print(f"Unmarried confidence interval of means:
{round(left_lim_unmarried),round(right_lim_unmarried)}")
```

```
90% Confidence Interval:
Married confidence interval of means: (819427, 867626)
Unmarried confidence interval of means: (859761, 901391)
```

Observations

For 90% Confidence Interval, the range for married & unmarried is overlapping.

We need to reduce the confidence interval to 85% and let’s check.

85% Confidence Interval:

```
#85% Confidence Interval

from scipy.stats import norm

# find the Z value with help of norm
Z = norm.ppf(.85)

sample_mean_married = married_customer['Purchase'].mean()
sample_mean_unmarried = unmarried_customer['Purchase'].mean()

Standerd_Error_married =
married_customer['Purchase'].std()/np.sqrt(len(married_customer))
Standerd_Error_unmarried =
unmarried_customer['Purchase'].std()/np.sqrt(len(unmarried_customer))

left_lim_married = (sample_mean_married - Z * Standerd_Error_married)
left_lim_unmarried = (sample_mean_unmarried - Z *
Standerd_Error_unmarried)

right_lim_married = (sample_mean_married + Z * Standerd_Error_married)
right_lim_unmarried = (sample_mean_unmarried + Z *
Standerd_Error_unmarried)

print("85% Confidence Interval:")
print(f"Married confidence interval of means:
{round(left_lim_married),round(right_lim_married)}")
print(f"Unmarried confidence interval of means:
{round(left_lim_unmarried),round(right_lim_unmarried)}")
```

```
85% Confidence Interval:
Married confidence interval of means: (824037, 863017)
Unmarried confidence interval of means: (863742, 897410)
```

Observations

For 85% Confidence Interval, the range for married & unmarried is not overlapping.

Now we can infer about the population that, 85% of the times:

Average amounts spend by Married customer will lie in between: (824037, 863017)

Average amounts spend by Unmarried customer will lie in between: (863742, 897410)

Perform the same activity for Age

For Age, you can try bins based on life stages: 0-17, 18-25, 26-35, 36-50, 51+ years.

```
# Trying to figure it out first transaction/purchase for each user id
based on their Age group.
# creating a new df and storing the result of each customers purchase

new_df_amount = df.groupby(['User_ID',
'Age'])['Purchase'].sum().reset_index()
new_df_amount
```

	User_ID	Age	Purchase
0	1000001	0-17	334093
1	1000002	55+	810472
2	1000003	26-35	341635
3	1000004	46-50	206468
4	1000005	26-35	821001
...	...	...	...
5886	1006036	26-35	4116058
5887	1006037	46-50	1119538
5888	1006038	55+	90034
5889	1006039	46-50	590319
5890	1006040	26-35	1653299

5891 rows × 3 columns

```
# trying to figure out the number of customers based on the Age group.

customers = new_df_amount['Age'].value_counts()
customers
```

```
26-35   2053
36-45   1167
18-25   1069
46-50    531
51-55    481
55+     372
```

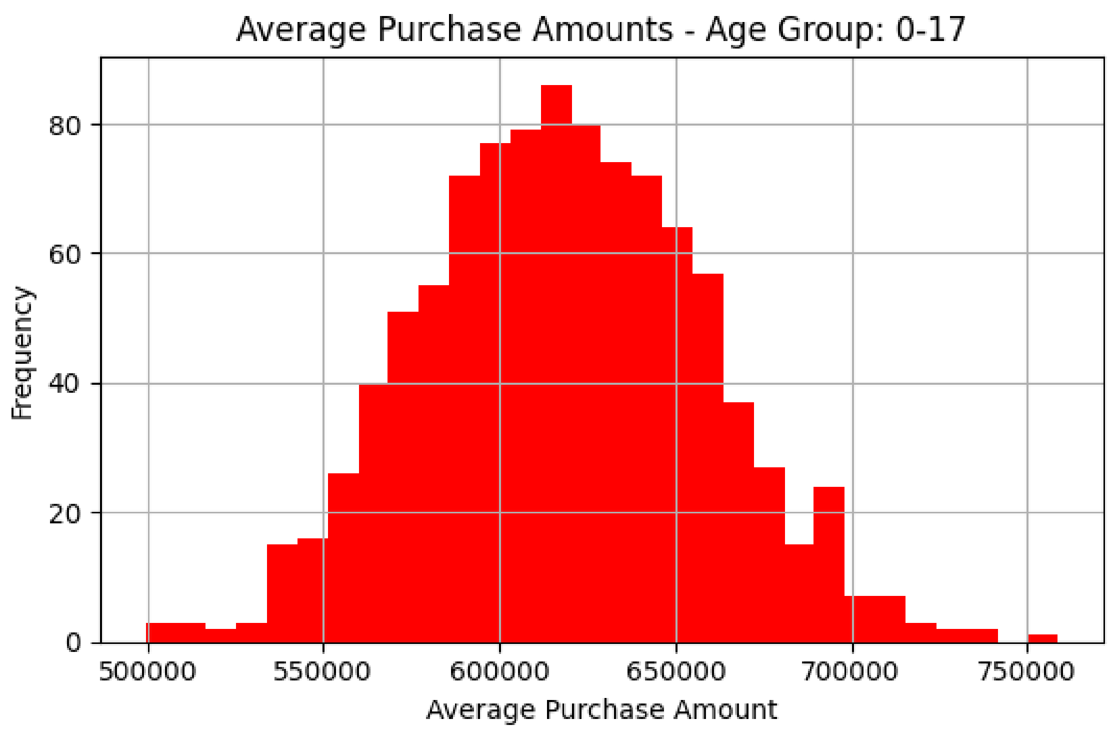
```
0-17     218
Name: Age, dtype: int64
```

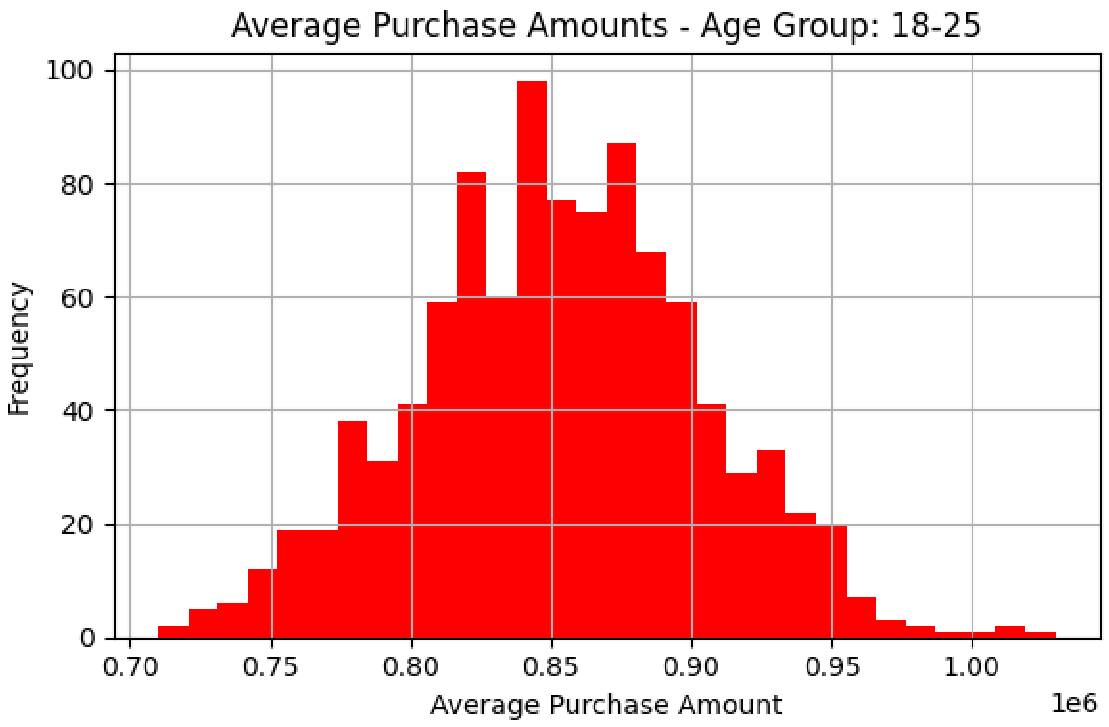
```
# trying to sort the purchase amount based on the Age group
# storing the average value based on Age groups
#taking the round value

avg_amount =
new_df_amount.groupby(['Age'])['Purchase'].mean().round().to_frame()
avg_amount
```

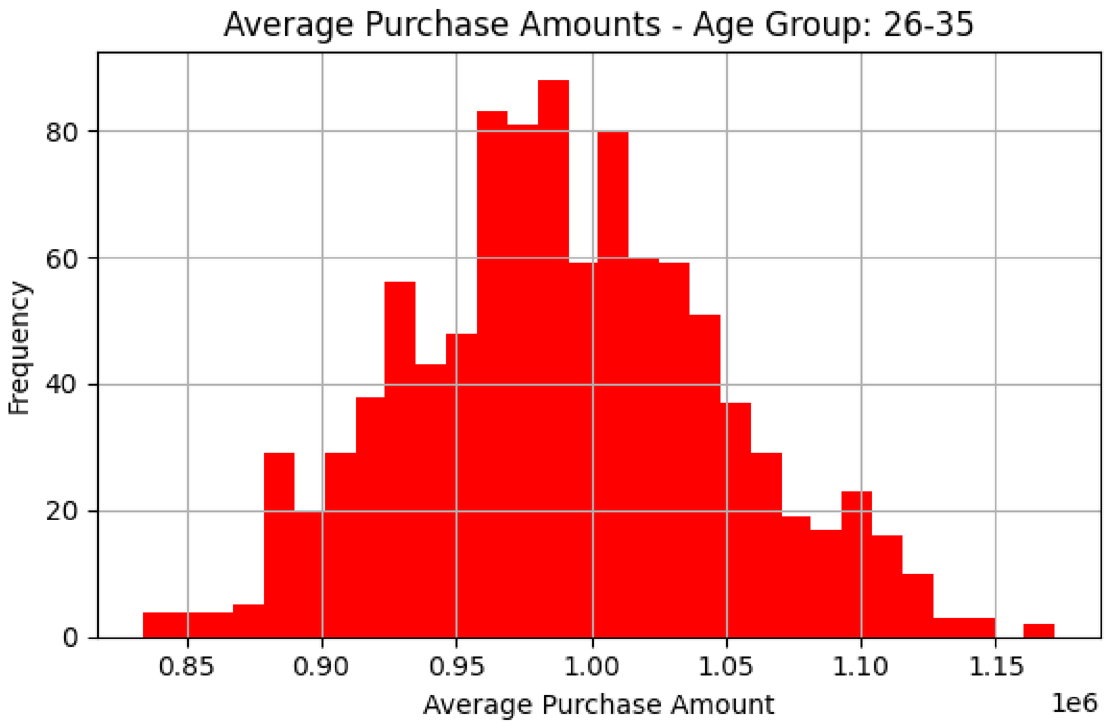
Age	Purchase
0-17	618868.0
18-25	854863.0
26-35	989659.0
36-45	879666.0
46-50	792549.0
51-55	763201.0
55+	539697.0

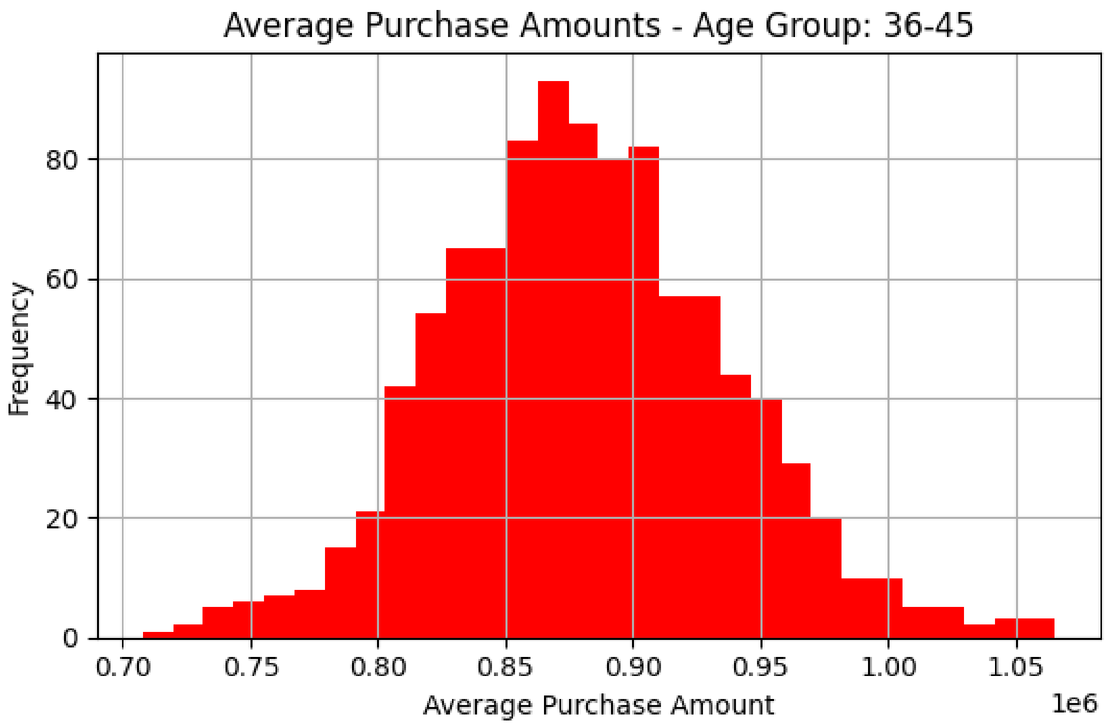
```
# first taking a list and put all age categories
age_bin = ['0-17','18-25','26-35','36-45','46-50','51-55','55+']

# assigning empty list
age_bin_mean = {}

# taking sample for each categories
sample_size = 300

no_of_time_check_sample = 1000
```

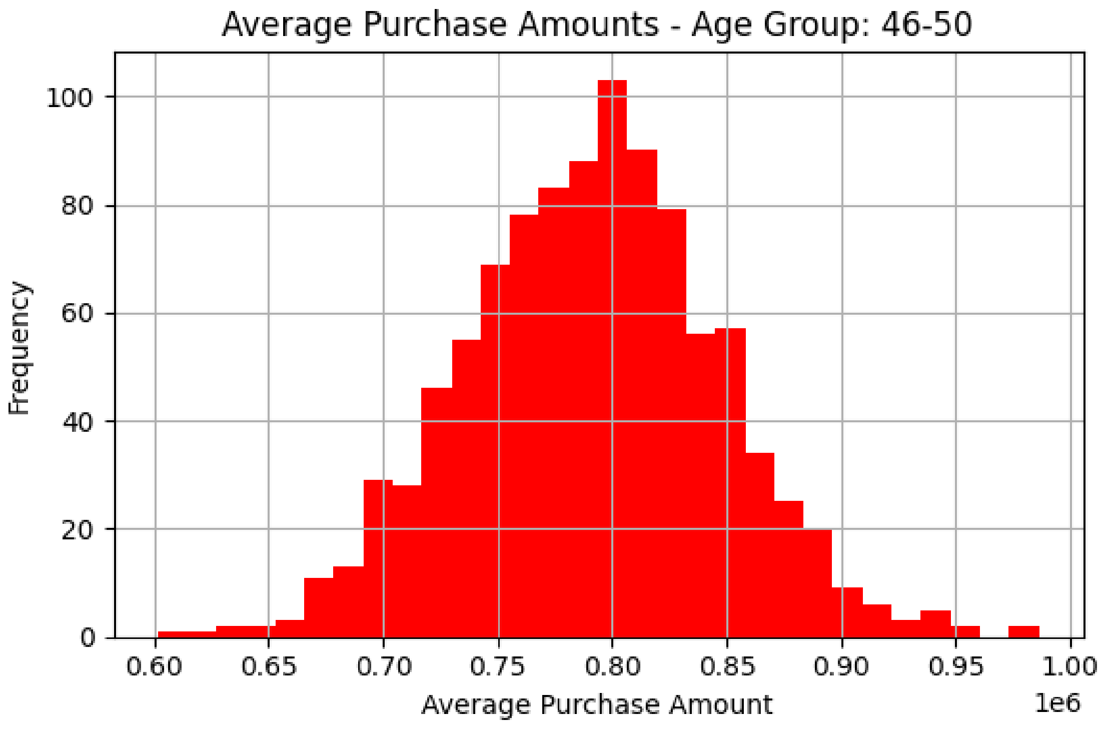
```
# assigning each age groups in to a list
for i in age_bin:
  age_bin_mean[i] = []

# for each age groups, now try to store mean value
for i in age_bin:
  for j in range(no_of_time_check_sample):
   mean = new_df_amount[new_df_amount['Age'] == i].sample(sample_size,
replace = True)['Purchase'].mean().round()
   age_bin_mean[i].append(mean)

# now try to plot hist plot for each age groups

for i in age_bin:
  plt.figure(figsize=(6, 4))
  plt.hist(age_bin_mean[i], bins = 30, color = 'red')
  plt.title(f"Average Purchase Amounts - Age Group: {i}")
  plt.xlabel("Average Purchase Amount")
  plt.ylabel("Frequency")
  plt.grid(True)
  plt.tight_layout()
```

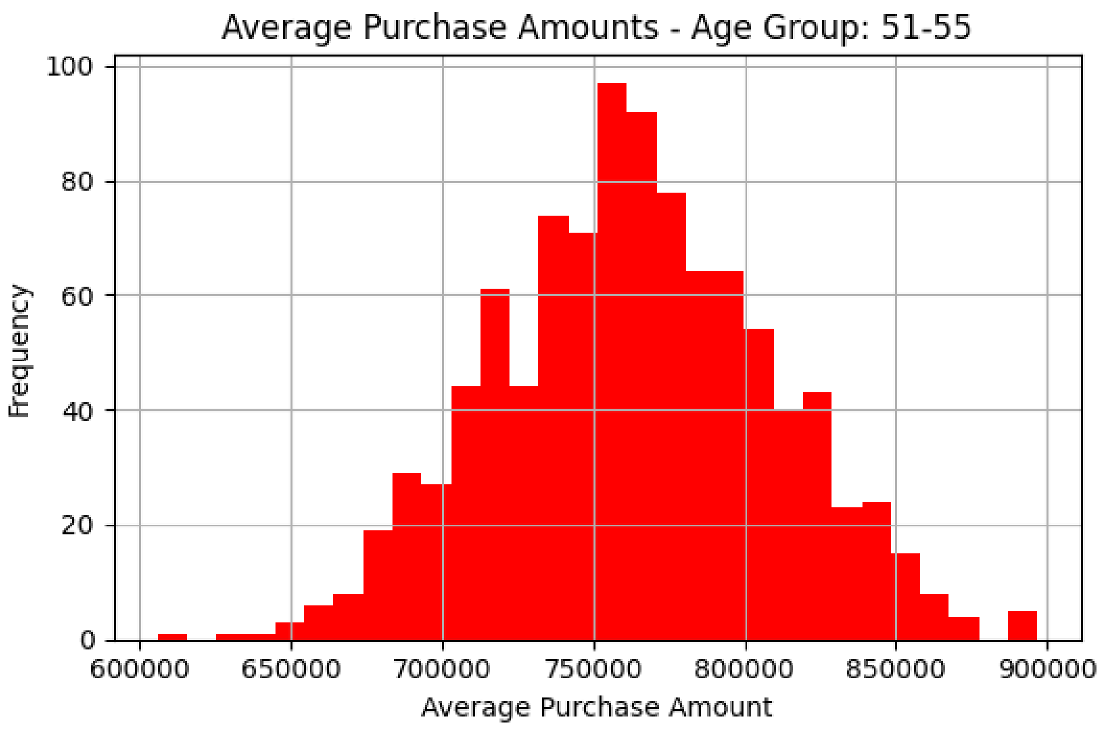


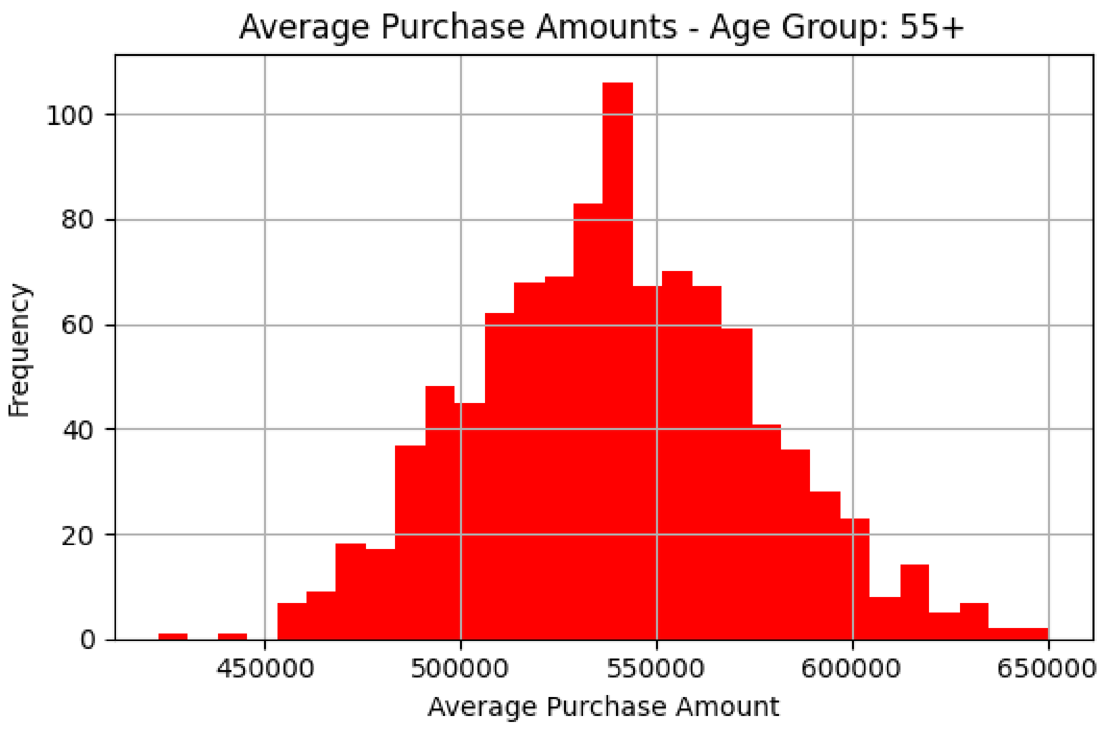

Average Purchase Amounts - Age Group: 18-25
Frequency
0
20
40
60
80
100
0.70
0.75
0.80
0.85
0.90
0.95
1.00
Average Purchase Amount
1e6


Average Purchase Amounts - Age Group: 26-35
Frequency
0
20
40
60
80
0.85
0.90
0.95
1.00
1.05
1.10
1.15
Average Purchase Amount
1e6


Average Purchase Amounts - Age Group: 36-45
Frequency
0
20
40
60
80
0.70
0.75
0.80
0.85
0.90
0.95
1.00
1.05
1e6
Average Purchase Amount


Average Purchase Amounts - Age Group: 46-50
Frequency
0
20
40
60
80
100
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95
1.00
1e6
Average Purchase Amount





Confidence Interval -> Z

99% Confidence Interval:

```
#99% Confidence Interval

from scipy.stats import norm
Z = norm.ppf(.99)
for i in ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']:

    new_df = new_df_amount[new_df_amount['Age']==i]

    margin_of_error_clt = Z *
new_df['Purchase'].std()/np.sqrt(len(new_df))
    sample_mean = new_df['Purchase'].mean()
    lower_lim = sample_mean - margin_of_error_clt
    upper_lim = sample_mean + margin_of_error_clt

    print("For age {} --> confidence interval of means: ({:.2f},
{:.2f})".format(i, lower_lim, upper_lim))
```

```
For age 26-35 --> confidence interval of means: (936693.49, 1042625.15)
For age 36-45 --> confidence interval of means: (812821.30, 946510.12)
For age 18-25 --> confidence interval of means: (791683.38, 918042.86)
For age 46-50 --> confidence interval of means: (698731.51, 886366.06)
For age 51-55 --> confidence interval of means: (679157.45, 847244.39)
For age 55+ --> confidence interval of means: (465219.71, 614174.78)

For age 0-17 --> confidence interval of means: (510615.06, 727120.56)
```

95% Confidence Interval:

```
#95% Confidence Interval

from scipy.stats import norm
Z = norm.ppf(.99)
for i in ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']:

    new_df = new_df_amount[new_df_amount['Age']==i]

    margin_of_error_clt = Z *
new_df['Purchase'].std()/np.sqrt(len(new_df))
    sample_mean = new_df['Purchase'].mean()
    lower_lim = sample_mean - margin_of_error_clt
    upper_lim = sample_mean + margin_of_error_clt

    print("For age {} --> confidence interval of means: ({:.2f},
{:.2f})".format(i, lower_lim, upper_lim))
```

```
For age 26-35 --> confidence interval of means: (936693.49, 1042625.15)
For age 36-45 --> confidence interval of means: (812821.30, 946510.12)
For age 18-25 --> confidence interval of means: (791683.38, 918042.86)
For age 46-50 --> confidence interval of means: (698731.51, 886366.06)
For age 51-55 --> confidence interval of means: (679157.45, 847244.39)
For age 55+ --> confidence interval of means: (465219.71, 614174.78)

For age 0-17 --> confidence interval of means: (510615.06, 727120.56)
```

90% Confidence Interval:

```
#90% Confidence Interval

from scipy.stats import norm
Z = norm.ppf(.99)
for i in ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']:

    new_df = new_df_amount[new_df_amount['Age']==i]

    margin_of_error_clt = Z *
new_df['Purchase'].std()/np.sqrt(len(new_df))
    sample_mean = new_df['Purchase'].mean()
    lower_lim = sample_mean - margin_of_error_clt
    upper_lim = sample_mean + margin_of_error_clt

    print("For age {} --> confidence interval of means: ({:.2f},
{:.2f})".format(i, lower_lim, upper_lim))
```

```
For age 0-17 --> confidence interval of means: (510615.06, 727120.56)

For age 18-25 --> confidence interval of means: (791683.38, 918042.86)

For age 26-35 --> confidence interval of means: (936693.49, 1042625.15)

For age 36-45 --> confidence interval of means: (812821.30, 946510.12)
For age 46-50 --> confidence interval of means: (698731.51, 886366.06)
For age 51-55 --> confidence interval of means: (679157.45, 847244.39)
For age 55+ --> confidence interval of means: (465219.71, 614174.78)
```

Observations

For 90% interval age

For age 0-17 --> confidence interval of means: (510615.06, 727120.56)

For age 18-25 --> confidence interval of means: (791683.38, 918042.86)

For age 26-35 --> confidence interval of means: (936693.49, 1042625.15)

For age 36-45 --> confidence interval of means: (812821.30, 946510.12)

For age 46-50 --> confidence interval of means: (698731.51, 886366.06)

For age 51-55 --> confidence interval of means: (679157.45, 847244.39)

For age 55+ --> confidence interval of means: (465219.71, 614174.78)

📊 Insights 📊

City Residency 🏙️

- 35% have been staying in the city for 1 year 🏠
- 18% have been staying for 2 years 🏠🏠
- 17% have been staying for 3 years 🏠🏠🏠

Product Categories 🛍️

- Total of 20 product categories available 📦

Occupations 💼

- 20 different types of occupations in the city 👔

User Demographics 📊

- Most users are Male ♂️
- 20 different types of Occupations and Product Categories 📋
- Majority of users belong to City Category B 🏙️
- More users are Single compared to Married 👤 vs 👫
- Product Categories 1, 5, 8, & 11 have the highest purchasing frequency 🛒 🔝

Average Spending 💰

- Average amount spent by Male customers: ₹925,344 💸
- Average amount spent by Female customers: ₹712,024 💸

Gender Ratio 👫

- 75% of the users are Male ♂️ and 25% are Female ♀️

Marital Status 👩❤️ 👨

- 60% are Single 👤 and 40% are Married 👫

Age Distribution 🎂

- ~ 80% of the users are aged between 18-50:
  - 40%: 26-35 📅
  - 18%: 18-25 🎈
  - 20%: 36-45 📆

Confidence Intervals 📈

Gender-wise

Now using the **Central Limit Theorem** for the **population**:

1. Average amount spend by **male** customers is **9,25,408.**
2. Average amount spend by **female** customers is **7,12,243.**

For **99% Confidence Interval**:

Now we can infer about the population that, **99% of the times**:

1. Average amount spend by male customer will lie in between: **(890062, 960627)**
2. Average amount spend by female customer will lie in between: **(666008, 758041)**

Marital Status-wise

Now using the **Central Limit Theorem** for the **population**:

1. Average amount spend by **Married** customers is **843527.**
2. Average amount spend by **Unmarried** customers is **880576.**

For **85% Confidence Interval**:
The confidence interval of means of Married and Unmarried is not overlapping.

Now we can infer about the population that, **85% of the times**:

1. Average amount spend by Married customer will lie in between: **(824037, 863017)**
2. Average amount spend by Unmarried customer will lie in between: **(863742, 897410)**

Age-wise

1For **90% Confidence Interval**:

1. For age 0-17 --> confidence interval of means: (510615.06, 727120.56)
2. For age 18-25 --> confidence interval of means: (791683.38, 918042.86)
3. For age 26-35 --> confidence interval of means: (936693.49, 1042625.15)
4. For age 36-45 --> confidence interval of means: (812821.30, 946510.12)
5. For age 46-50 --> confidence interval of means: (698731.51, 886366.06)
6. For age 51-55 --> confidence interval of means: (679157.45, 847244.39)
7. For age 55+ --> confidence interval of means: (465219.71, 614174.78)

Recommendations

1) Gender-focused Strategy

- Men tend to spend more than women. The company should prioritize retaining existing male customers and attracting new male customers.
- Should be focusing on female customer's choice, like and dislike part. Based on that attract customers by giving sale on many reasons.

2) Product Category Insight

- Products in categories 1, 5, 8, & 11 have the highest purchasing frequency and are favored by customers. The company can consider increasing the promotion and availability of these products, as well as boosting less-purchased items.

3) Marital Status Approach

- Unmarried customers exhibit higher spending compared to married customers. The company should concentrate on attracting and engaging unmarried customers.
- For married coupe needs to arrange valuable product based on their daily needs.
- Many of couples have children. Attract those children with some beautiful products and focus a sale on couple side.

4) Targeting Specific Age Group

- Customers aged 18-45 contribute more to the spending. To enhance revenue, the company should focus on acquiring customers within this age range.

5) City Category Strategy

- Male customers residing in City_Category C demonstrate higher spending compared to those in City_Category B or A. To increase revenue, the company should consider emphasizing product offerings in City_Category C.