Walmart case study

- Dataset: Walmart dataset
- Name: S.Tejeswara Rao
- Email: tejavishnu2000@gmail.com

Problem Statement:

- About 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).

Importing all required python libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Probability & statastics libraries
from scipy.stats import norm
from scipy.stats import binom
from scipy.stats import poisson
from scipy.stats import expon
from scipy.stats import geom
from scipy.stats import uniform
# Hypothesis test requred libraries
from statsmodels.stats.weightstats import ztest as ztest
from scipy.stats import ttest_1samp
from scipy.stats import ttest_ind
from scipy.stats import ttest_rel
from scipy.stats import f_oneway
from scipy.stats import levene
from scipy.stats import kruskal
```

Read walmart dataset

```
walmart_data = pd.read_csv('/content/sample_data/walmart_data.csv')
```

```
walmart_data.head(5)
```

	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	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	

```
rows,columns=walmart_data.shape
print(f"Number of rows: {rows} \nNumber of columns: {columns}")
```

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

Datatypes of walmart dataset

```
walmart_data.dtypes
```

	0
User_ID	int64
Product_ID	object
Gender	object
Age	object
Occupation	int64
City_Category	object
Stay_In_Current_City_Years	object
Marital_Status	int64
Product_Category	int64
Purchase	int64

dtype: object

Along with datatypes we could see walmart dataset how much memory consumes in disk, columns contains null values or not.

```
walmart_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
```

```
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64
 8   Product_Category            550068 non-null  int64
 9   Purchase                    550068 non-null  int64
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

Observatins:

we can clearly see that the all columns alligned with proper data types. But one column "Stay_In_Current_City_Years" is object datatype , We need to convert this into numeric column for better analysis.

```
walmart_data['Stay_In_Current_City_Years'].unique()
```

```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

Column "Stay_In_Current_City_Years" has 5 different years 0,1,2,3 and 4+. So converting 4+ into 4.

Keep in mind when we see 4 years -- that is more than 4 years.

```
walmart_data['Stay_In_Current_City_Years'] = walmart_data['Stay_In_Current_City_Years'].str.replace('+','')
```

```
walmart_data.dtypes
```

	0
User_ID	int64
Product_ID	object
Gender	object
Age	object
Occupation	int64
City_Category	object
Stay_In_Current_City_Years	int64
Marital_Status	int64
Product_Category	int64
Purchase	int64

dtype: object

Check any duplicate records are present.

```
print(f"Number of duplicate rows: {walmart_data.duplicated().sum()}")
```

```
Number of duplicate rows: 0
```

Handling missing/null values

```
walmart_data.isnull().sum()
```

	0
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

We can clearly observe that there is no null/missing values. So no need of any missing value handling.

Numerical columns and categorical culumns.

```
columns=walmart_data.columns
numerical_columns=['Stay_In_Current_City_Years', 'Purchase']
categorical_columns=['User_ID','Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category', 'Marital_Statu
print(f"There are {len(columns)} columns = {list(columns)}")
print(f"There are {len(numerical_columns)} numerical columns = {list(numerical_columns)}")
print(f"There are {len(categorical_columns)} categorical columns = {list(categorical_columns)}")
```

```
There are 10 columns = ['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category', 'Stay_
There are 2 numerical columns = ['Stay_In_Current_City_Years', 'Purchase']
There are 8 categorical columns = ['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Catego
```

Value counts VS nunique for every column in our dataset.

```
for i in columns:
  print(f"{i}: {walmart_data[i].nunique()}")
```

```
User_ID: 5891
Product_ID: 3631
Gender: 2
Age: 7
Occupation: 21
City_Category: 3
Stay_In_Current_City_Years: 5
Marital_Status: 2
Product_Category: 20
Purchase: 18105
```

```
for i in columns:
  print(f"{i}: {walmart_data[i].value_counts()}")
  print('-'*50)
```

```
A    147720
Name: count, dtype: int64
-------------------------------------------------
Stay_In_Current_City_Years: Stay_In_Current_City_Years
1    193821
2    101838
3     95285
4     84726
0     74398
Name: count, dtype: int64
-------------------------------------------------
Marital_Status: Marital_Status
0    324731
1    225337
Name: count, dtype: int64
-------------------------------------------------
Product_Category: Product_Category
5     150933
1     140378
8     113925
11     24287
2      23864
6      20466
3      20213
4      11753
16      9828
15      6290
13      5549
10      5125
12      3947
7       3721
18      3125
20      2550
19      1603
14      1523
17       578
9        410
Name: count, dtype: int64
-------------------------------------------------
Purchase: Purchase
7011     191
7193     188
6855     187
6891     184
7012     183
        ...
23491      1
18345      1
3372       1
855        1
21489      1
Name: count, Length: 18105, dtype: int64
-------------------------------------------------
```

Unique values in categorical columns

```
for i in categorical_columns:
  print(f"{i}: {walmart_data[i].unique()}")
```

```
User_ID: [1000001 1000002 1000003 ... 1004113 1005391 1001529]
Product_ID: ['P00069042' 'P00248942' 'P00087842' ... 'P00370293' 'P00371644'
 'P00370853']
Gender: ['F' 'M']
Age: ['0-17' '55+' '26-35' '46-50' '51-55' '36-45' '18-25']
Occupation: [10 16 15  7 20  9  1 12 17  0  3  4 11  8 19  2 18  5 14 13  6]
City_Category: ['A' 'C' 'B']
Marital_Status: [0 1]
Product_Category: [ 3  1 12  8  5  4  2  6 14 11 13 15  7 16 18 10 17  9 20 19]
```

Descriptive statastics for all numerical columns in walmart dataset.

```
for i in numerical_columns:
  print(f"{i}: {walmart_data[i].describe()}\n")
  print('-'*50)
```

```
Stay_In_Current_City_Years: count    550068.000000
mean          1.858418
std           1.289443
min           0.000000
25%           1.000000
50%           2.000000
75%           3.000000
max           4.000000
Name: Stay_In_Current_City_Years, dtype: float64

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

--------------------------------------------------
```

```
walmart_data.describe()
```

	User_ID	Occupation	Stay_In_Current_City_Years	Marital_Status	Produ
count	5.500680e+05	550068.000000	550068.000000	550068.000000	55
mean	1.003029e+06	8.076707	1.858418	0.409653	
std	1.727592e+03	6.522660	1.289443	0.491770	
min	1.000001e+06	0.000000	0.000000	0.000000	
25%	1.001516e+06	2.000000	1.000000	0.000000	
50%	1.003077e+06	7.000000	2.000000	0.000000	
75%	1.004478e+06	14.000000	3.000000	1.000000	
max	1.006040e+06	20.000000	4.000000	1.000000	

Detecting outliers

```
for i in numerical_columns:
  q1=walmart_data[i].quantile(0.25)
  q3=walmart_data[i].quantile(0.75)
  iqr=q3-q1
  lower_bound=q1-1.5*iqr
  upper_bound=q3+1.5*iqr
  outliers=walmart_data[(walmart_data[i]<lower_bound) | (walmart_data[i]>upper_bound)]
  print(f"Number of outliers in {i}: {len(outliers)}")
```

```
Number of outliers in Stay_In_Current_City_Years: 0
Number of outliers in Purchase: 2677
```

We can observe that the only 2 columns ['Product_Category', 'Purchase'] having outliers.

No need to clip these two columns because product category and purchages are important factors to improve bussiness.

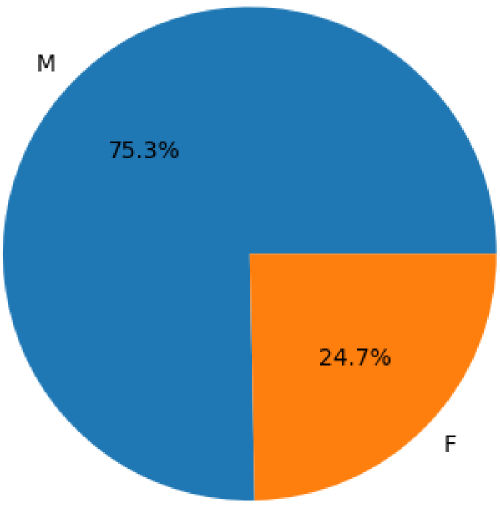
```
walmart_data['Gender'].value_counts()
```

Gender	count
M	414259
F	135809

dtype: int64

Male users are more compared to femele users in given sample dataset. It doesn't mean males users are more than female users.

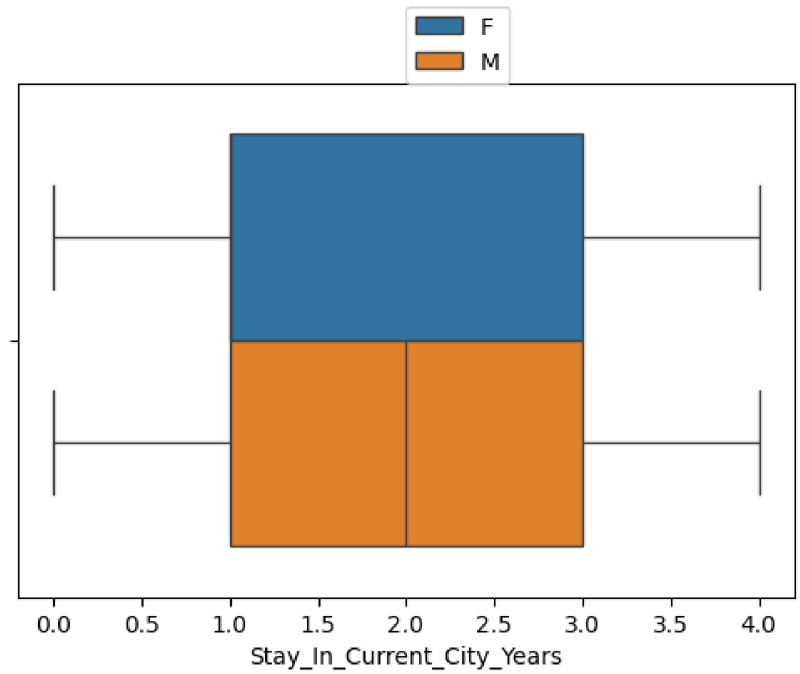
```
plt.pie(walmart_data['Gender'].value_counts(),labels=walmart_data['Gender'].value_counts().index,autopct='%
plt.show()
```

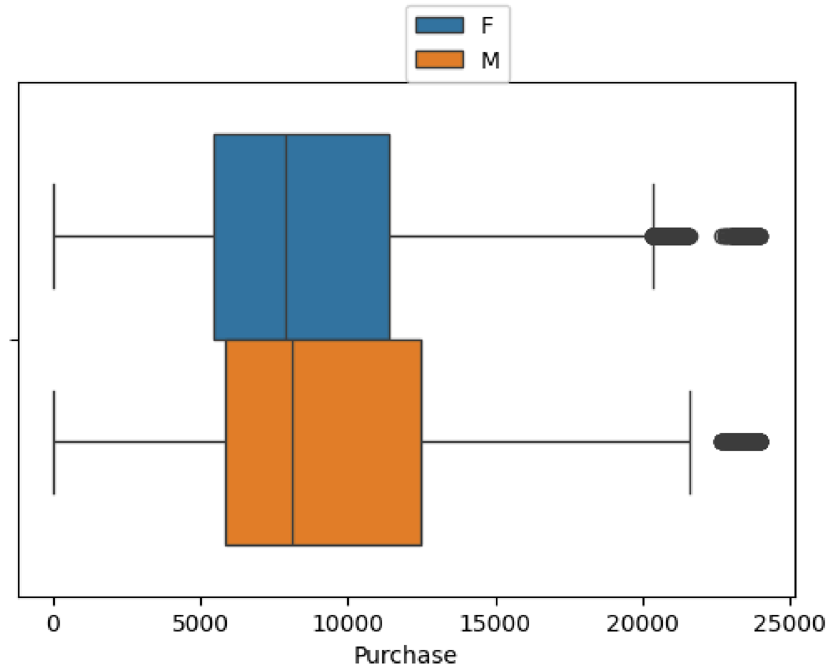


Our sample walmart dataset contains 75.3 % of male data and 24.7 % of female data.

Double-click (or enter) to edit

```
for i in numerical_columns:
  plt.figure(figsize=(6,4))
  sns.boxplot(x=i,data=walmart_data,hue='Gender')
  plt.legend(loc=(0.5,1))
  plt.show()
```

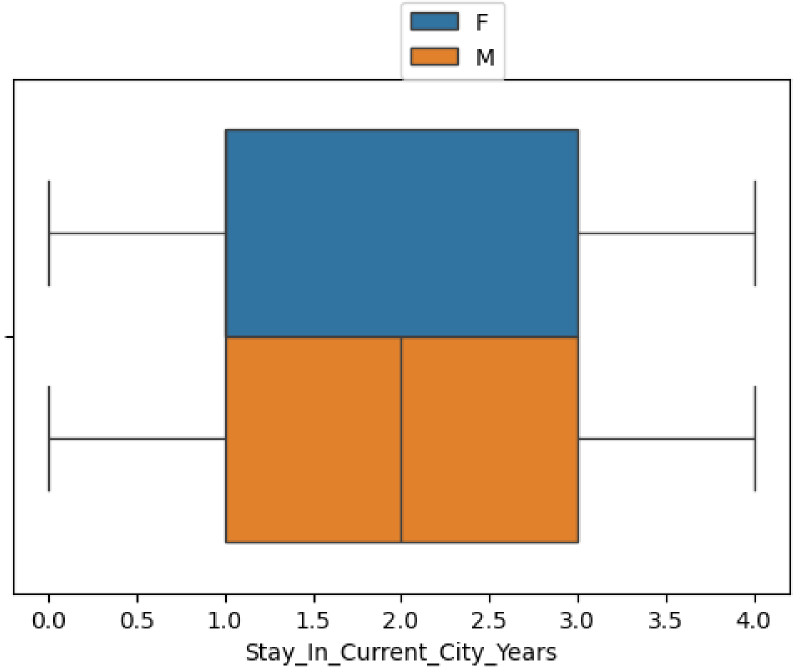


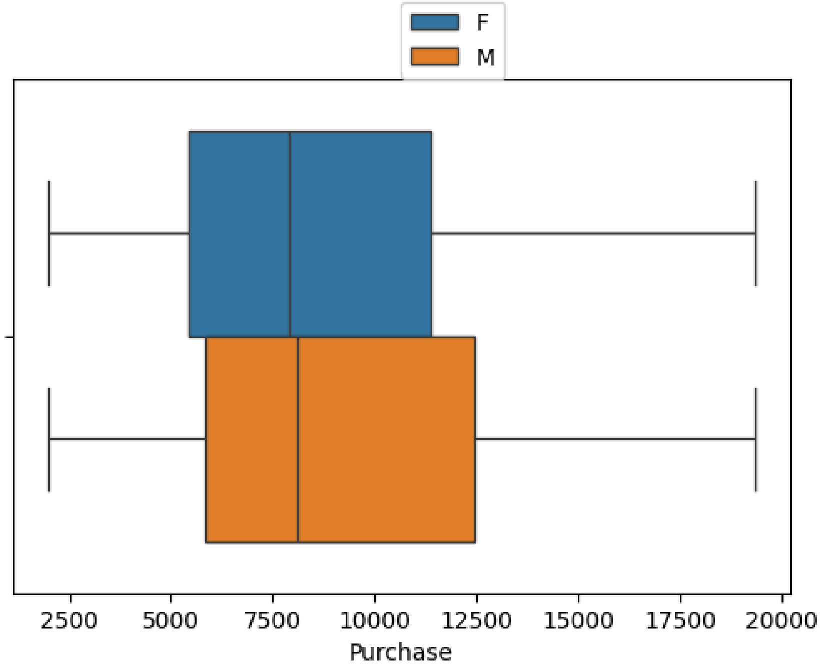


Clipping the outlier column where data lessthan 5% and greater than 95%.

```
walmart_data['Purchase']=np.where(walmart_data['Purchase']<walmart_data['Purchase'].quantile(0.05),walmart_
walmart_data['Purchase']=np.where(walmart_data['Purchase']>walmart_data['Purchase'].quantile(0.95),walmart_
```

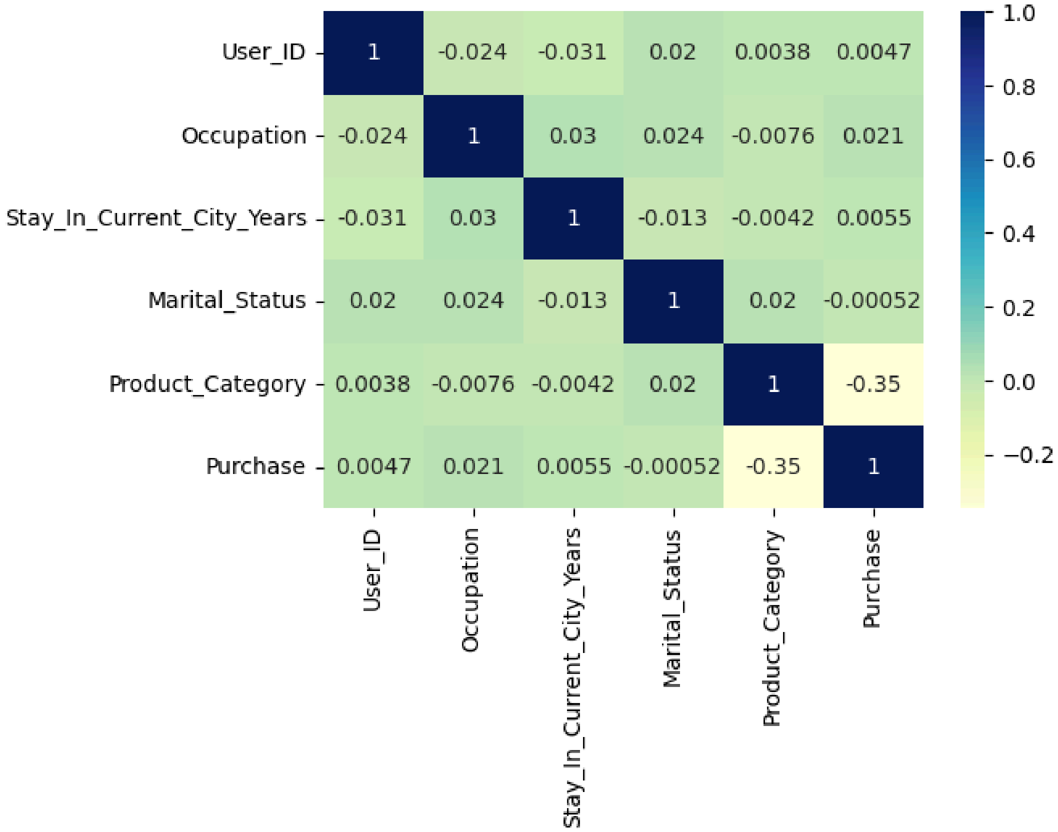
```
for i in numerical_columns:
  plt.figure(figsize=(6,4))
  sns.boxplot(x=i,data=walmart_data,hue='Gender')
  plt.legend(loc=(0.5,1))
  plt.show()
```



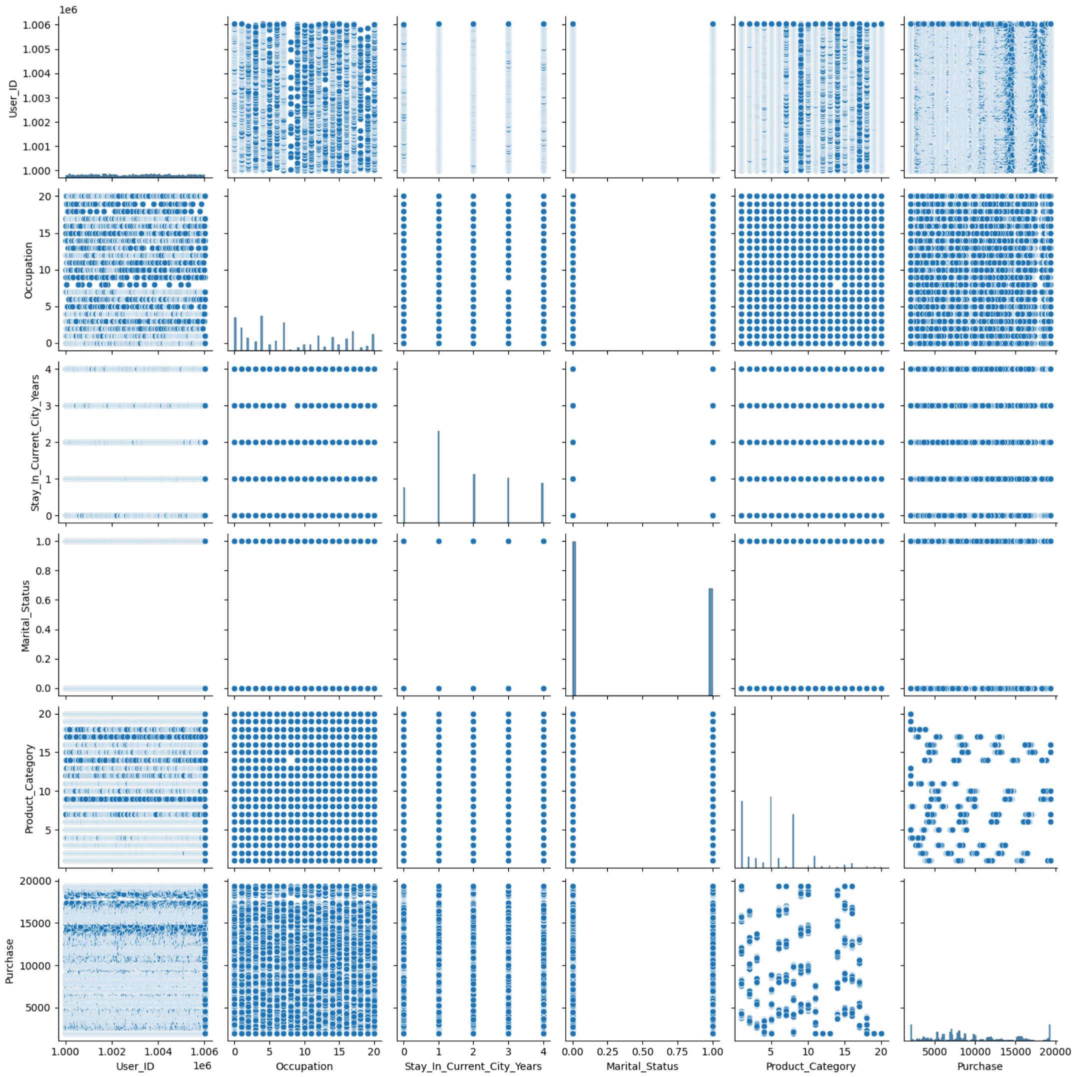


So outlier clipped where <5% and >95%.

```
num_columns= walmart_data.select_dtypes(include='number').columns
plt.figure(figsize=(6,4))
ax=sns.heatmap(walmart_data[num_columns].corr(),annot=True,cmap='YlGnBu')
ax.set(xlabel="", ylabel="")
#ax.xaxis.tick_top()
plt.show()
```




```
sns.pairplot(walmart_data[num_columns])
plt.show()
```



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.

Given that the Walmart american retail corporation has more than 100 million customers worldwide.

But given the dataset of 5.5 lacks records (i.e rows=550068).

Out of this 5.5 lack recors, 4.1 lack recors of data belongs to males and 1.35 lack recors of data belogs to female.

Means 75.3% are male and 24.7% are female.

```
walmart_female=walmart_data[walmart_data['Gender']=='F']
walmart_male=walmart_data[walmart_data['Gender']=='M']
```

```
walmart_female['Purchase'].describe()
```

	Purchase
count	135809.000000
mean	8736.540266
std	4596.984614
min	1984.000000
25%	5433.000000
50%	7914.000000
75%	11400.000000
max	19336.000000

dtype: float64

```
walmart_male['Purchase'].describe()
```

	Purchase
count	414259.000000
mean	9427.240997
std	4925.953492
min	1984.000000
25%	5863.000000
50%	8098.000000
75%	12454.000000
max	19336.000000

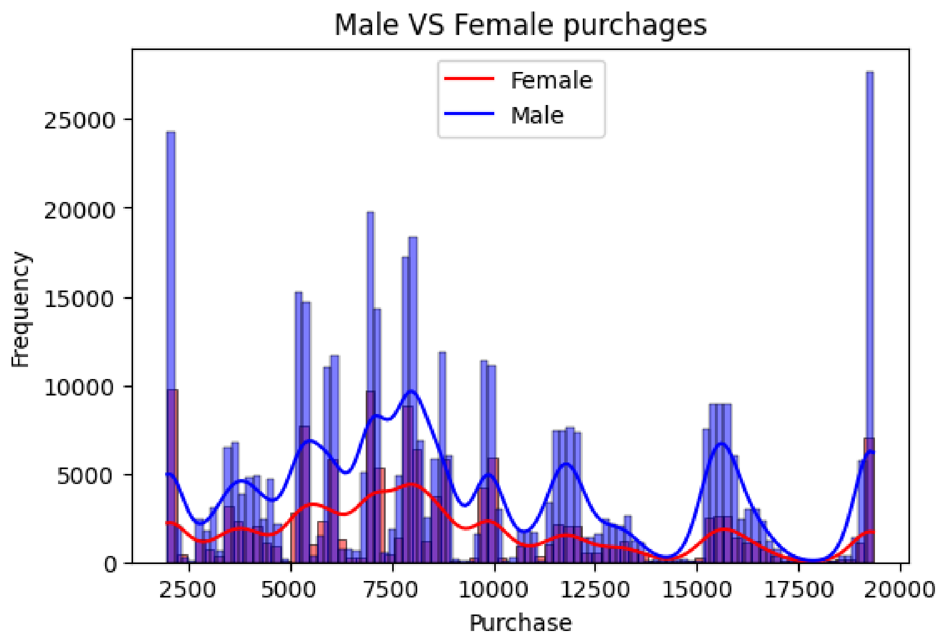
dtype: float64

Male average purchase per transaction = 9437.52604

Female average purchage per transaction = 8734.565765

We are make sure this is not population data, so we can't conclude with this results.

```
plt.figure(figsize=(6,4))
sns.histplot(walmart_female['Purchase'],color='red',kde=True)
sns.histplot(walmart_male['Purchase'],color='blue',kde=True)
plt.legend(['Female','Male'])
plt.xlabel('Purchase')
plt.ylabel('Frequency')
plt.title('Male VS Female purchages')
plt.show()
```



Given data not exactly following gaussian distribution.

According to the central limit theorem -- sample means of multiple sample follows gaussian distribution when **sample size greater than 30**.

Now collecting 10000 samples with size 200 for both males and females. Then both males and females means of 10000 samples follows gaussian distribution.

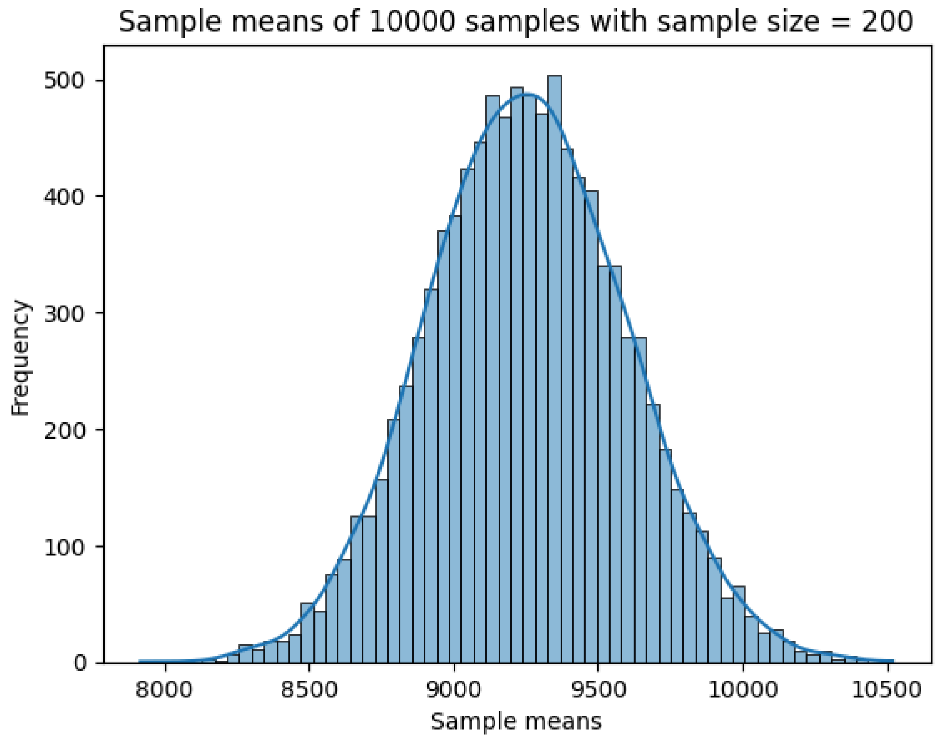
Applying CTL to given walmart sample of dataset.

Apply CTL to given walmart dataset.(i.e given dataset is one sample from 100 million userrs of walmart american corpuration).

```
import random
random.seed(40)
#sample size
sample_size=200

# collecting means of 10000 samples with sample size = 200
sample_means_data = []

for i in range(10000):
  sample=random.sample(range(1, len(walmart_data)), sample_size)
  sample_data=walmart_data['Purchase'].iloc[sample]

  #sample mean
  sample_mean=round(np.mean(sample_data),2)
  # all the sample means appending into sample_means.
  sample_means_data.append(sample_mean)

print(f"Sample means of 10000 samples: {sample_means_data}")
```

```
Sample means of 10000 samples: [9239.55, 9086.08, 9688.62, 8674.85, 8697.54, 9247.66, 9545.42, 8868.91,
```

```
sns.histplot(sample_means_data,kde=True)
plt.title('Sample means of 10000 samples with sample size = 200')
plt.xlabel('Sample means')
plt.ylabel('Frequency')
plt.show()
```

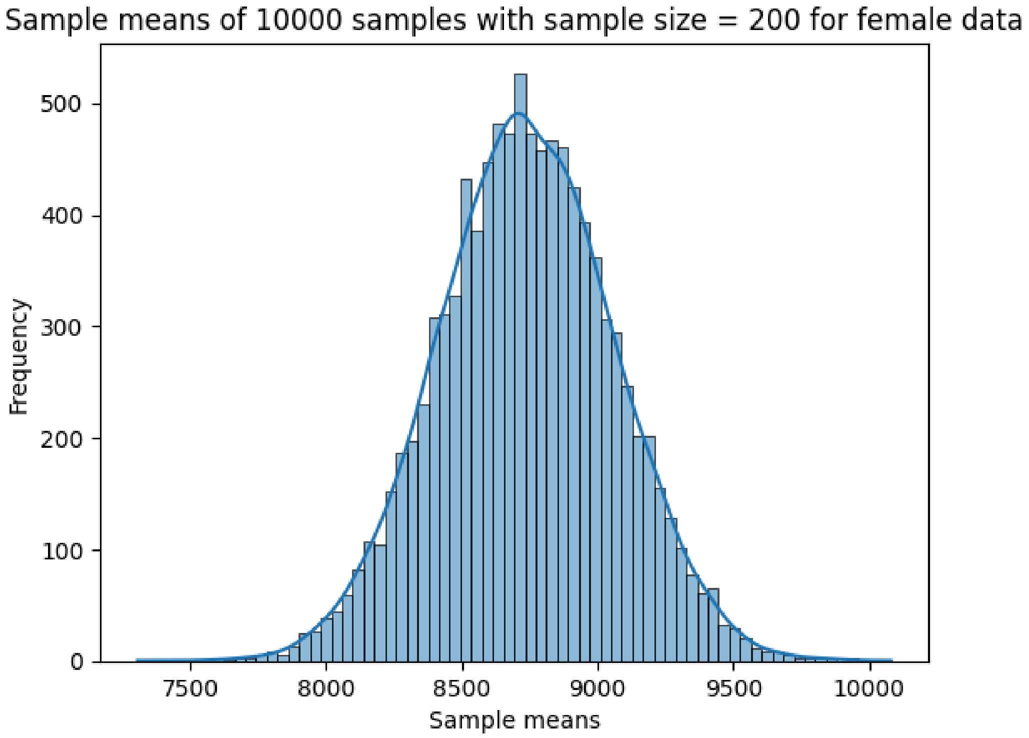


Applying CLT to female data first

```
random.seed(40)

# sample size
sample_size=200

# collecting means of 10000 samples with sample size = 200
females_sample_means = []

for i in range(10000):
  sample=random.sample(range(1, len(walmart_female)), sample_size)
  sample_data=walmart_female['Purchase'].iloc[sample]

  #sample mean
  sample_mean=round(np.mean(sample_data),2)
  # all the sample means appending into sample_means.
  females_sample_means.append(sample_mean)

print(f"Sample means of 10000 samples: {females_sample_means}")
```

```
Sample means of 10000 samples: [8470.58, 9267.62, 8761.8, 8490.9, 8600.46, 8985.05, 8985.75, 8710.23, 8
```

```
sns.histplot(females_sample_means,kde=True)
plt.title('Sample means of 10000 samples with sample size = 200 for female data')
plt.xlabel('Sample means')
plt.ylabel('Frequency')
plt.show()
```

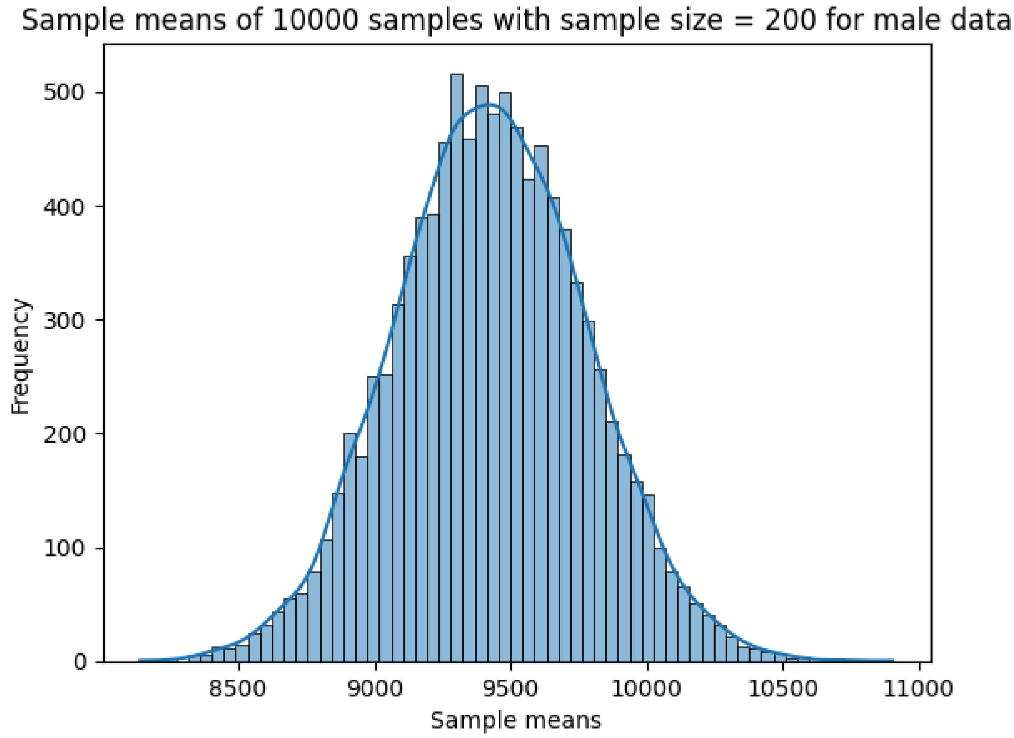


Applying CLT to male data

```
# collecting means of 10000 samples with sample size = 200
random.seed(40)
male_sample_means = []

for i in range(10000):
  sample=random.sample(range(1, len(walmart_male)), sample_size)
  sample_data=walmart_male['Purchase'].iloc[sample]

  #sample mean
  sample_mean=round(np.mean(sample_data),2)
  # all the sample means appending into sample_means.
  male_sample_means.append(sample_mean)

print(f"Sample means of 10000 samples: {male_sample_means}")
```

```
Sample means of 10000 samples: [9060.18, 9406.42, 9215.96, 9535.63, 9816.94, 9001.7, 10032.76, 9709.77,
```

```
sns.histplot(male_sample_means,kde=True)
plt.title('Sample means of 10000 samples with sample size = 200 for male data')
plt.xlabel('Sample means')
plt.ylabel('Frequency')
plt.show()
```



Double-click (or enter) to edit

```
mean_sample_means_data=round(np.mean(sample_means_data),2)
mean_sample_means_female=round(np.mean(females_sample_means),2)
mean_sample_means_male=round(np.mean(male_sample_means),2)
print(f"Mean of sample means of data: {mean_sample_means_data}")
print(f"Mean of sample means of female data: {mean_sample_means_female}")
print(f"Mean of sample means of male data: {mean_sample_means_male}")
```

```
Mean of sample means of data: 9253.39
Mean of sample means of female data: 8736.37
Mean of sample means of male data: 9427.06
```

Double-click (or enter) to edit

```
std_sample_means_data=round(np.std(sample_means_data),2)
print(f"Standard deviation of sample means of data: {std_sample_means_data}")
std_sample_means_female=round(np.std(females_sample_means),2)
print(f"Standard deviation of sample means of female data: {std_sample_means_female}")
std_sample_means_male=round(np.std(male_sample_means),2)
print(f"Standard deviation of sample means of male data: {std_sample_means_male}")
```

```
Standard deviation of sample means of data: 343.69
Standard deviation of sample means of female data: 322.71
Standard deviation of sample means of male data: 351.85
```

```
#Total data distribution

#As we already know sample size = 200
data_sample_size=sample_size

# Total sample mean is means of sample means.
data_sample_mean= mean_sample_means_data

#Standard error = total sample standard deviation / sqrt(200)
standard_error= std_sample_means_data
population_std=std_sample_means_data * np.sqrt(data_sample_size)

#Population mean is same as total sample mean
population_mean=mean_sample_means_data

# we dont know population standard deviation.

#Z_score at 95% confidence level
z_score=norm.ppf(0.95)

#Interval at 5% significant (or) 95% confidence level
lower_bound=round(data_sample_mean-z_score*standard_error,2)
upper_bound=round(data_sample_mean+z_score*standard_error,2)
interval=(lower_bound,upper_bound)

#print all data
print(f"Total sample mean: {data_sample_mean}")
print(f"Total standard error: {standard_error}")
print(f"Total population standard deviation: {population_std}")
print(f"Total population mean: {population_mean}")
print('-'*100)
print(f"Total z_score at 95% confidence level: {z_score}")
print(f"Total interval at 5% significant (or) 95% confidence level: {interval}")
print('-'*100)

# -----------------------------------------------------------------------------------
#Z_score at 90% confidence level
z_score=norm.ppf(0.90)

#Interval at 10% significant (or) 90% confidence level
lower_bound=round(data_sample_mean-z_score*standard_error,2)
upper_bound=round(data_sample_mean+z_score*standard_error,2)
interval=(lower_bound,upper_bound)
print(f"Total z_score at 90% confidence level: {z_score}")
print(f"Total interval at 10% significant (or) 90% confidence level: {interval}")
print('-'*100)

# -------------------------------------------------------------------------------------
#Z_score at 99% confidence level
z_score=norm.ppf(0.99)

#Interval at 1% significant (or) 99% confidence level
lower_bound=round(data_sample_mean-z_score*standard_error,2)
upper_bound=round(data_sample_mean+z_score*standard_error,2)
interval=(lower_bound,upper_bound)
print(f"Total z_score at 99% confidence level: {z_score}")
print(f"Total interval at 1% significant (or) 99% confidence level: {interval}")
```

```
Total sample mean: 9253.39
Total standard error: 343.69
Total population standard deviation: 4860.5105925200905
Total population mean: 9253.39
----------------------------------------------------------------------------------------------------
Total z_score at 95% confidence level: 1.6448536269514722
Total interval at 5% significant (or) 95% confidence level: (8688.07, 9818.71)
```

```
------------------------------------------------------------------------------------------------------
Total z_score at 90% confidence level: 1.2815515655446004
Total interval at 10% significant (or) 90% confidence level: (8812.93, 9693.85)
------------------------------------------------------------------------------------------------------
Total z_score at 99% confidence level: 2.3263478740408408
Total interval at 1% significant (or) 99% confidence level: (8453.85, 10052.93)
```

At 5% significant level,walmart data population confidence interval = (8688.07, 9818.71)

```
#Female data distribution

#As we already know sample size = 200
female_sample_size=sample_size

# Female sample mean is means of sample means.
female_sample_mean= mean_sample_means_female

#Standard error = female sample standard deviation / sqrt(200)
standard_error= std_sample_means_female
female_population_std=std_sample_means_female * np.sqrt(female_sample_size)

#Population mean is same as female sample mean
population_mean=mean_sample_means_female

# we dont know population standard deviation.

#Z_score at 95% confidence level
z_score=norm.ppf(0.95)

#Interval at 5% significant (or) 95% confidence level
lower_bound=round(female_sample_mean-z_score*standard_error,2)
upper_bound=round(female_sample_mean+z_score*standard_error,2)
interval=(lower_bound,upper_bound)

#print all data
print(f"Female sample mean: {female_sample_mean}")
print(f"Female standard error: {standard_error}")
print(f"Female population standard deviation: {female_population_std}")
print(f"Female population mean: {population_mean}")
print('-'*100)
print(f"Female z_score at 95% confidence level: {z_score}")
print(f"Female interval at 5% significant (or) 95% confidence level: {interval}")
print('-'*100)

# ------------------------------------------------------------------------------------
#Z_score at 90% confidence level
z_score=norm.ppf(0.90)

#Interval at 10% significant (or) 90% confidence level
lower_bound=round(female_sample_mean-z_score*standard_error,2)
upper_bound=round(female_sample_mean+z_score*standard_error,2)
interval=(lower_bound,upper_bound)
print(f"Female z_score at 90% confidence level: {z_score}")
print(f"Female interval at 10% significant (or) 90% confidence level: {interval}")
print('-'*100)

# --------------------------------------------------------------------------------------
#Z_score at 99% confidence level
z_score=norm.ppf(0.99)

#Interval at 1% significant (or) 99% confidence level
lower_bound=round(female_sample_mean-z_score*standard_error,2)
upper_bound=round(female_sample_mean+z_score*standard_error,2)
interval=(lower_bound,upper_bound)
print(f"Female z_score at 99% confidence level: {z_score}")
print(f"Female interval at 1% significant (or) 99% confidence level: {interval}")
```

```
Female sample mean: 8736.37
Female standard error: 322.71
Female population standard deviation: 4563.808587134215
Female population mean: 8736.37
----------------------------------------------------------------------------------------------------
Female z_score at 95% confidence level: 1.6448536269514722
Female interval at 5% significant (or) 95% confidence level: (8205.56, 9267.18)
```

```
----------------------------------------------------------------------------------------------------
Female z_score at 90% confidence level: 1.2815515655446004
Female interval at 10% significant (or) 90% confidence level: (8322.8, 9149.94)
----------------------------------------------------------------------------------------------------
Female z_score at 99% confidence level: 2.3263478740408408
Female interval at 1% significant (or) 99% confidence level: (7985.63, 9487.11)
```

At 5% significant level,Female data confodence interval = (8205.56, 9267.18)

```
#Male data Distribution

#As we already know sample size = 200
male_sample_size=sample_size

# Male sample mean is means of sample means.
male_sample_mean= mean_sample_means_male

#Standard error = male sample standard deviation / sqrt(200)
standard_error= std_sample_means_male
male_population_std=std_sample_means_male * np.sqrt(male_sample_size)

#Male population mean is same as male samle mean
population_mean=mean_sample_means_male

# we dont know population standard deviation.

#Z_score at 95% confidence level
z_score=norm.ppf(0.95)

#Interval at 5% significant (or) 95% confidence
lower_bound=round(male_sample_mean-z_score*standard_error,2)
upper_bound=round(male_sample_mean+z_score*standard_error,2)
interval= (lower_bound,upper_bound)

#print all data
print(f"Male sample mean: {male_sample_mean}")
print(f"Male standard error: {standard_error}")
print(f"Male population standard deviation: {male_population_std}")
print(f"Male population mean: {population_mean}")
print('-'*100)
print(f"Male z_score at 95% confidence level: {z_score}")
print(f"Male interval at 5% significant (or) 95% confidence level: {interval}")
print('-'*100)
# ---------------------------------------------------------------------------------------
#Z_score at 90% confidence level
z_score=norm.ppf(0.90)

#Interval at 10% significant (or) 90% confidence level
lower_bound=round(male_sample_mean-z_score*standard_error,2)
upper_bound=round(male_sample_mean+z_score*standard_error,2)
interval=(lower_bound,upper_bound)
print(f"Male z_score at 90% confidence level: {z_score}")
print(f"Male interval at 10% significant (or) 90% confidence level: {interval}")
print('-'*100)

# -----------------------------------------------------------------------------------------
#Z_score at 99% confidence level
z_score=norm.ppf(0.99)

#Interval at 1% significant (or) 99% confidence level
lower_bound=round(male_sample_mean-z_score*standard_error,2)
upper_bound=round(male_sample_mean+z_score*standard_error,2)
interval=(lower_bound,upper_bound)
print(f"Male z_score at 99% confidence level: {z_score}")
print(f"Male interval at 1% significant (or) 99% confidence level: {interval}")
```

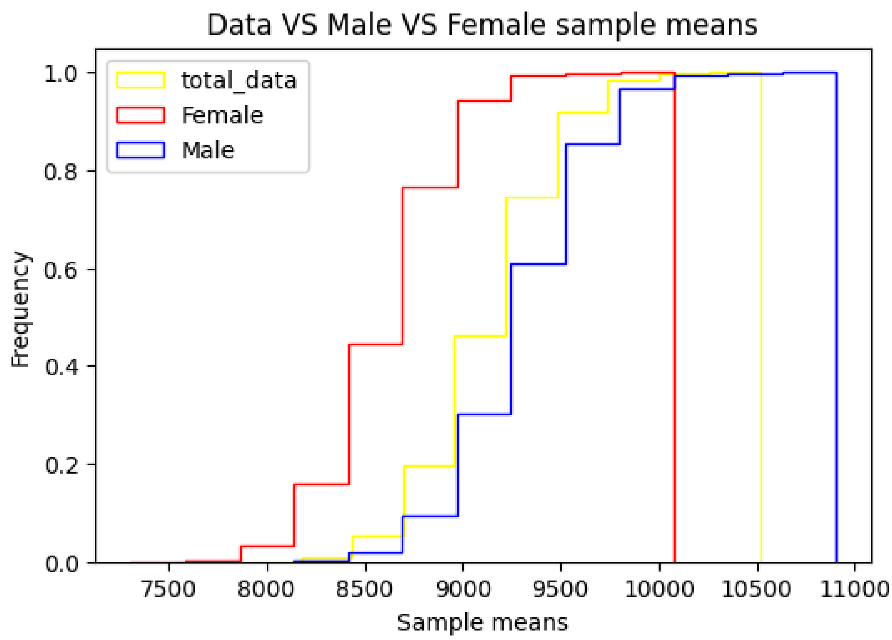
```
Male sample mean: 9427.06
Male standard error: 351.85
Male population standard deviation: 4975.910419209736
Male population mean: 9427.06
----------------------------------------------------------------------------------------------------
Male z_score at 95% confidence level: 1.6448536269514722
Male interval at 5% significant (or) 95% confidence level: (8848.32, 10005.8)
```

```
--------------------------------------------------------------------------------------------
Male z_score at 90% confidence level: 1.2815515655446004
Male interval at 10% significant (or) 90% confidence level: (8976.15, 9877.97)
--------------------------------------------------------------------------------------------
Male z_score at 99% confidence level: 2.3263478740408408
Male interval at 1% significant (or) 99% confidence level: (8608.53, 10245.59)
```

At 5% significant level, Male data confidence interval = (8848.32, 10005.8)

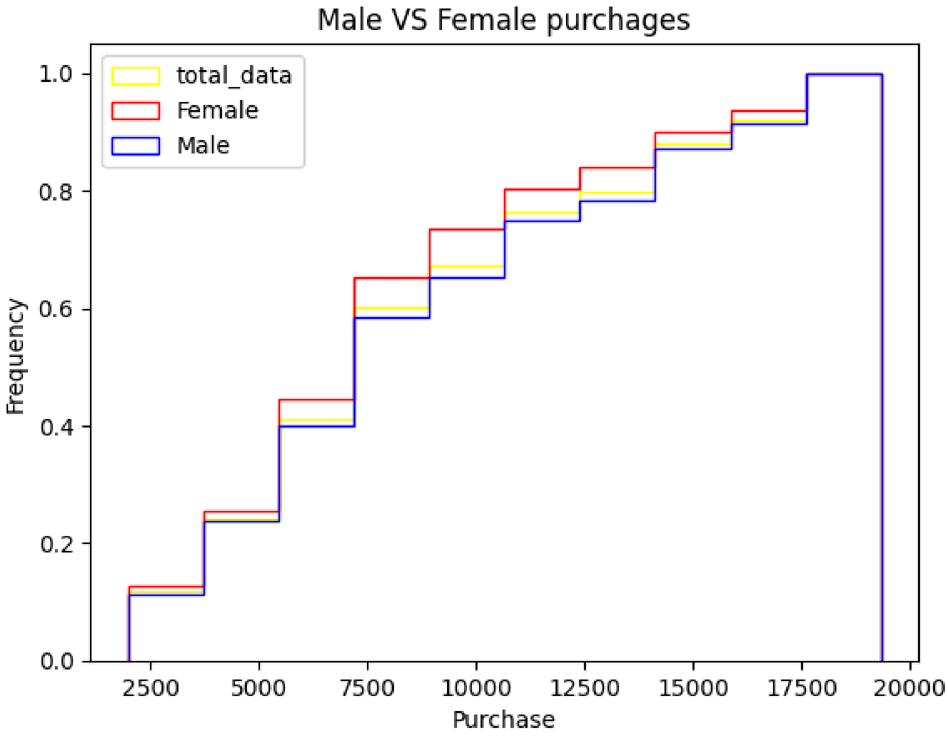
we can see there is no overlapping between male and female 95% confidence interval

```
#Histplot of sample means of DATA VS MALES VS FEMALES
plt.figure(figsize=(6,4))
plt.hist(sample_means_data, color= 'yellow',label='total_data',histtype='step',density=True,cumulative=True
plt.hist(females_sample_means,color='red',label='Female',histtype='step',density=True,cumulative=True)
plt.hist(male_sample_means,color='blue',label='Male',histtype='step',density=True,cumulative=True)
plt.xlabel('Sample means')
plt.ylabel('Frequency')
plt.title('Data VS Male VS Female sample means')
plt.legend()
plt.show()
```



Start coding or generate with AI.

```
#Histplot of DATA VS MALES VS FEMALES
plt.hist(walmart_data['Purchase'], color= 'yellow',label='total_data',histtype='step',density=True,cumulati
plt.hist(walmart_female['Purchase'],color='red',label='Female',histtype='step',density=True,cumulative=True
plt.hist(walmart_male['Purchase'],color='blue',label='Male',histtype='step',density=True,cumulative=True)
plt.xlabel('Purchase')
plt.ylabel('Frequency')
plt.title('Male VS Female purchages')
plt.legend()
plt.show()
```



Same thing we have to verify with hypothesis.

Using t-test we can find that the average spent for transaction of males same as average spent for transaction of males

Null Hypothesis: male average spent = female average spent

Alternative Hypothesis: male average spent != female average spent

```
#t-tets
t_statastics,p_value=ttest_ind(walmart_male['Purchase'],walmart_female['Purchase'])

# At 95% confidence level
alpha=0.05
print(f"t_statastics: {t_statastics}")
print(f"p_value: {p_value}")
if p_value < alpha:
  print('Reject Null Hypothesis at 95% confidence \nAverage male spent amount is different than Average fem
else:
  print('Accept Null Hypothesis at 95% confidence \nAverage male spent amount is same as Average female spe
print("-"*100)

# At 90% confidence interval
alpha=0.1
print(f"t_statastics: {t_statastics}")
print(f"p_value: {p_value}")
if p_value < alpha:
  print('Reject Null Hypothesis at 90% confidence \nAverage male spent amount is different than Average fem
else:
  print('Accept Null Hypothesis at 90% confidence \nAverage male spent amount is same as Average female spe
print("-"*100)

# At 99% confidence interval
alpha=0.01
if p_value < alpha:
  print('Reject Null Hypothesis at 99% confidence \nAverage male spent amount is different than Average fem
else:
  print('Accept Null Hypothesis at 99% confidence \nAverage male spent amount is same as Average female spe
```

```
t_statastics: 45.574933432542736
p_value: 0.0
Reject Null Hypothesis at 95% confidence
Average male spent amount is different than Average female spent amount

----------------------------------------------------------------------------------------------------
t_statastics: 45.574933432542736
p_value: 0.0
Reject Null Hypothesis at 90% confidence
Average male spent amount is different than Average female spent amount

----------------------------------------------------------------------------------------------------
Reject Null Hypothesis at 99% confidence
Average male spent amount is different than Average female spent amount
```

Null Hypothesis: male average spent = female average spent

Alternative Hypothesis: male average spent > female average spent

```
t_statastics,p_value=ttest_ind(walmart_male['Purchase'],walmart_female['Purchase'],alternative='greater')

# At 95% confidence level
alpha=0.05
print(f"t_statastics: {t_statastics}")
print(f"p_value: {p_value}")
if p_value < alpha:
  print('Reject Null Hypothesis at 95% confidence \nAverage male spent amount is larger than Average female
else:
  print('Fail to reject Null Hypothesis at 95% confidence \nAverage male spent amount is same as Average fe
print("-"*100)

# At 90% confidence interval
alpha=0.1
print(f"t_statastics: {t_statastics}")
print(f"p_value: {p_value}")
if p_value < alpha:
  print('Reject Null Hypothesis at 90% confidence \nAverage male spent amount is larger than Average female
else:
  print('Fail to reject Null Hypothesis at 90% confidence \nAverage male spent amount is same as Average fe
print("-"*100)

# At 99% confidence interval
alpha=0.01
if p_value < alpha:
  print('Reject Null Hypothesis at 99% confidence \nAverage male spent amount is larger than Average female
else:
  print('Fail to reject Null Hypothesis at 99% confidence \nAverage male spent amount is same as Average fe
```

```
t_statastics: 45.574933432542736
p_value: 0.0
Reject Null Hypothesis at 95% confidence
Average male spent amount is larger than Average female spent amount

----------------------------------------------------------------------------------------------------
t_statastics: 45.574933432542736
p_value: 0.0
Reject Null Hypothesis at 90% confidence
Average male spent amount is larger than Average female spent amount

----------------------------------------------------------------------------------------------------
Reject Null Hypothesis at 99% confidence
Average male spent amount is larger than Average female spent amount
```

Null Hypothesis: male average spent = female average spent

Alternative Hypothesis: male average spent < female average spent

```
t_statastics,p_value=ttest_ind(walmart_male['Purchase'],walmart_female['Purchase'],alternative='less')

# At 95% confidence level
alpha=0.05
print(f"t_statastics: {t_statastics}")
print(f"p_value: {p_value}")
if p_value < alpha:
  print('Reject Null Hypothesis at 95% confidence \nAverage male spent amount is smaller than Average femal
else:
  print('Fail to reject Null Hypothesis at 95% confidence \nAverage male spent amount is same as Average fe
print("-"*100)

# At 90% confidence interval
alpha=0.1
print(f"t_statastics: {t_statastics}")
print(f"p_value: {p_value}")
if p_value < alpha:
  print('Reject Null Hypothesis at 90% confidence \nAverage male spent amount is smaller than Average femal
else:
  print('Fail to reject Null Hypothesis at 90% confidence \nAverage male spent amount is same as Average fe
print("-"*100)

# At 99% confidence interval
alpha=0.01
print(f"t_statastics: {t_statastics}")
print(f"p_value: {p_value}")
if p_value < alpha:
  print('Reject Null Hypothesis at 99% confidence \nAverage male spent amount is smaller than Average femal
else:
  print('Fail to reject Null Hypothesis at 99% confidence\nAverage male spent amount is same as Average fem
```

```
t_statastics: 45.574933432542736
p_value: 1.0
Fail to reject Null Hypothesis at 95% confidence
Average male spent amount is same as Average female spent amount

----------------------------------------------------------------------------------------------------
t_statastics: 45.574933432542736
p_value: 1.0
Fail to reject Null Hypothesis at 90% confidence
Average male spent amount is same as Average female spent amount

----------------------------------------------------------------------------------------------------
t_statastics: 45.574933432542736
p_value: 1.0
Fail to reject Null Hypothesis at 99% confidence
Average male spent amount is same as Average female spent amount
```

Insights:

**As per the CLT, we could see population, female customers and male customers mean spent amount lies at below ranges.* *

Population mean:

Population mean at 95% confidence interval = (8688.07, 9818.71). Population meen lies between 8688.07 and 9818.71 range at 5% singnificant level.

Population mean at 90% confidence interval = (8812.93, 9693.85). Population meen lies between 8812.93 and 9693.85 range at 10% singnificant level.

Population mean at 99% confidence interval = (8453.85, 10052.93). Population meen lies between 8453.85 and 10052.93 range at 1% singnificant level.

Female population mean:

- Female sample mean: 8736.37
- Female standard error: 322.71
- Female population standard deviation: 4563.808587134215
- Female population mean: 8736.37

Female population mean at 95% confidence interval = (8205.56, 9267.18). Population meen lies between 8205.56 and 9267.18 range at 5% singnificant level.

Female population mean at 90% confidence interval = (8322.8, 9149.94). Population meen lies between 8322.8 and 9149.94 range at 10% singnificant level.

Female population mean at 99% confidence interval = (7985.63, 9487.11). Population meen lies between 7985.63 and 9487.11 range at 99% singnificant level.

Male population mean:

- Male sample mean: 9427.06
- Male standard error: 351.85
- Male population standard deviation: 4975.910419209736
- Male population mean: 9427.06

Male population mean at 95% confidence interval = (8848.32, 10005.8). Population meen lies between 8848.32 and 10005.8 range at 5% singnificant level.

Male population mean at 90% confidence interval = (8976.15, 9877.97). Population meen lies between 8976.15 and 9877.97 range at 10% singnificant level.

Male population mean at 99% confidence interval = (8608.53, 10245.59). Population meen lies between 8608.53 and 10245.59 range at 1% singnificant level.

Q: 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?

Ans:

As per the above confidence intervals at 90%,95% and 99%. There is a overlapping ranges between male female. There is a huge difference between means of male spending and female spending.

At 90% confidence level.

Female_interval = (8322.8, 9149.94)

Male_interval = (8976.15, 9877.97)

At 95% confidence level.

Female_interval = (8205.56, 9267.18)

Male_interval = (8848.32, 10005.8)

At 99% confidence level.

Female_interval = (7985.63, 9487.11)

Male_interval = (8608.53, 10245.59)

Conclusion:

Female mean spent amount and male mean spent amount overlaping ranges at 90%,95% and 99% confidence intervals. And we can see female is less purchges than male.

Verified same by ttest hypothesis testing, there is a significant evidence to reject null hypothesis means female average spent amount is less than male spent amount.

So focus on males with current strategies & products. For females provide more female attractive products.

Q: Perform the same activity for Married vs Unmarried and Age

ANS:

Peforming the same activity to married and unmarried people first.

```
# Married VS Unmarried

married_data=walmart_data[walmart_data['Marital_Status']==1]
unmarried_data=walmart_data[walmart_data['Marital_Status']==0]


uniqueue_male_female=[married_data['User_ID'].nunique(),unmarried_data['User_ID'].nunique()]
print(f"married people = {uniqueue_male_female[0]}")
print(f"unmarried people = {uniqueue_male_female[1]}")
```

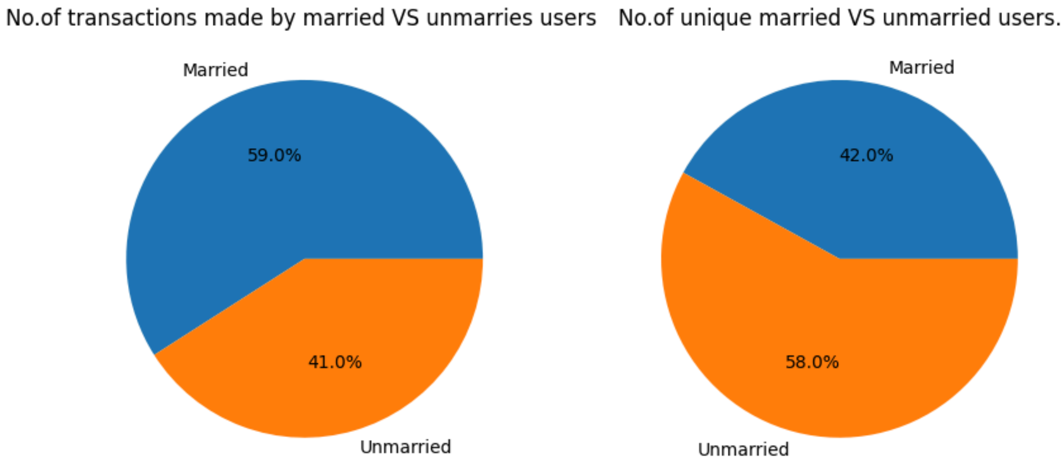
```
married people = 2474
unmarried people = 3417
```

```
walmart_data['Marital_Status'].value_counts()
```

Marital_Status	count
0	324731
1	225337

dtype: int64

```
plt.figure(figsize=(10,6))

plt.subplot(1,2,1)
plt.pie(walmart_data['Marital_Status'].value_counts(),labels=['Married','Unmarried'],autopct='%1.1f%%')
plt.title('No.of transactions made by married VS unmarries users ')

plt.subplot(1,2,2)
plt.pie(uniqueue_male_female,labels=['Married','Unmarried'],autopct='%1.1f%%')
plt.title('No.of unique married VS unmarried users.')
plt.show()
```

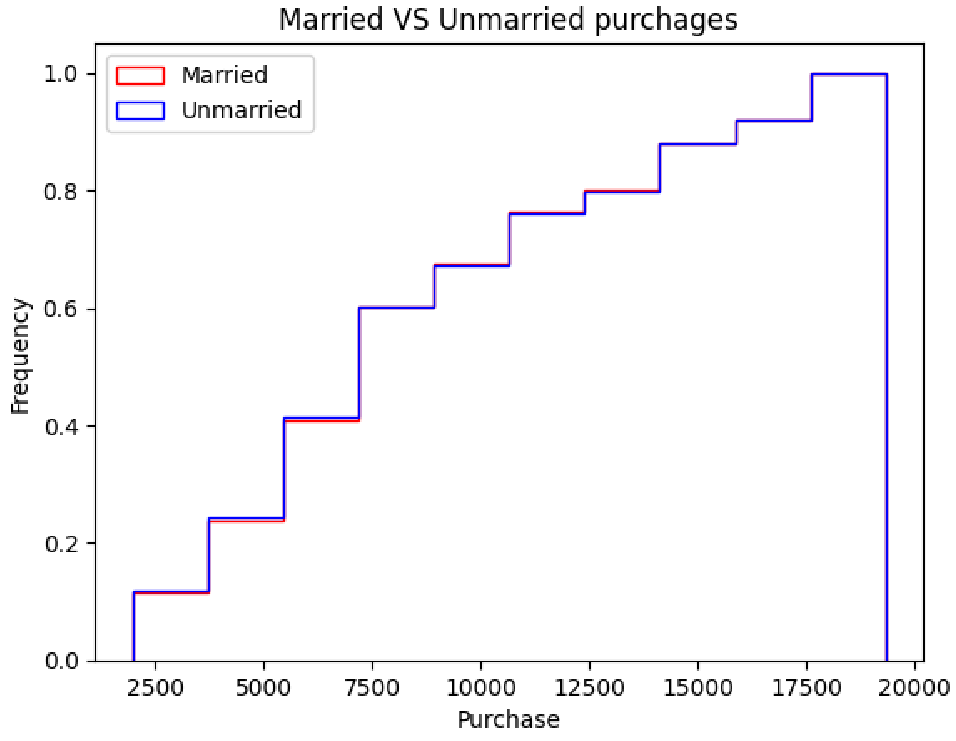


Double-click (or enter) to edit

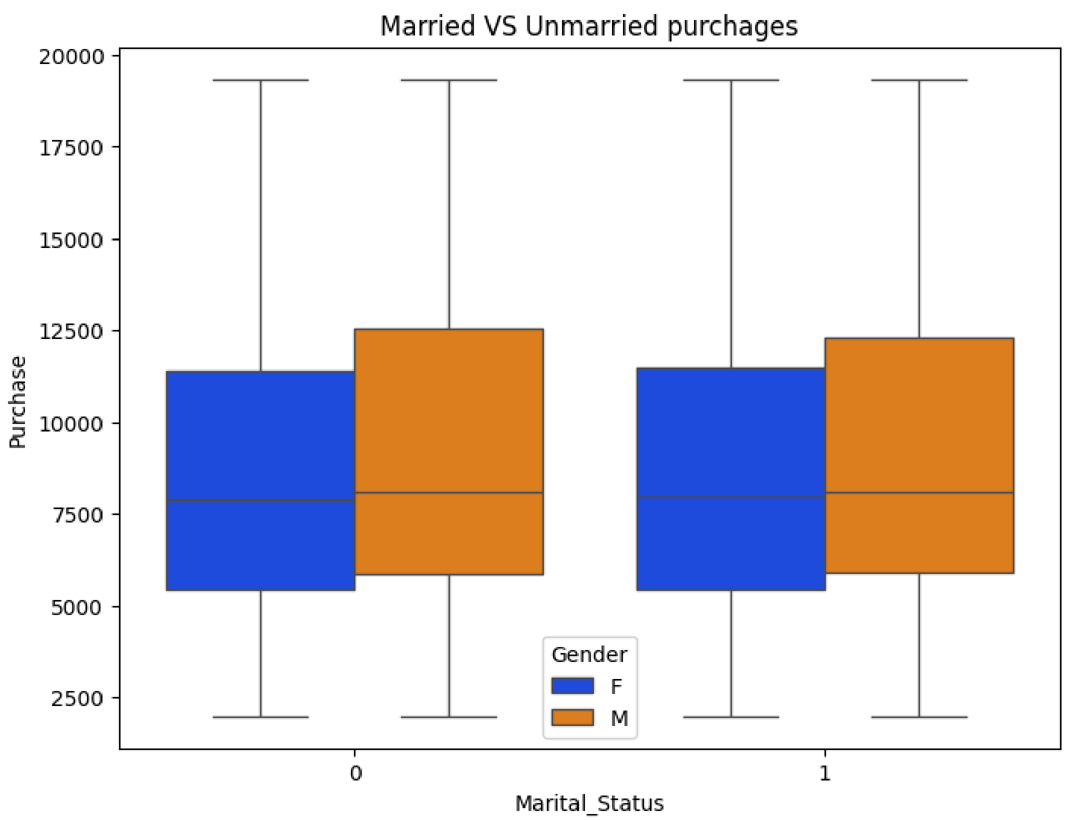
```
married_purchages_avg=married_data['Purchase'].mean()
unmarried_purchages_avg=unmarried_data['Purchase'].mean()
print(f"married_purchages: {married_purchages_avg}")
print(f"unmarried_purchages: {unmarried_purchages_avg}")
print(f"difference between married and unmarried: {married_purchages_avg-unmarried_purchages_avg}")
```

```
married_purchages: 9253.669823420034
unmarried_purchages: 9258.820463706883
difference between married and unmarried: -5.150640286849011
```

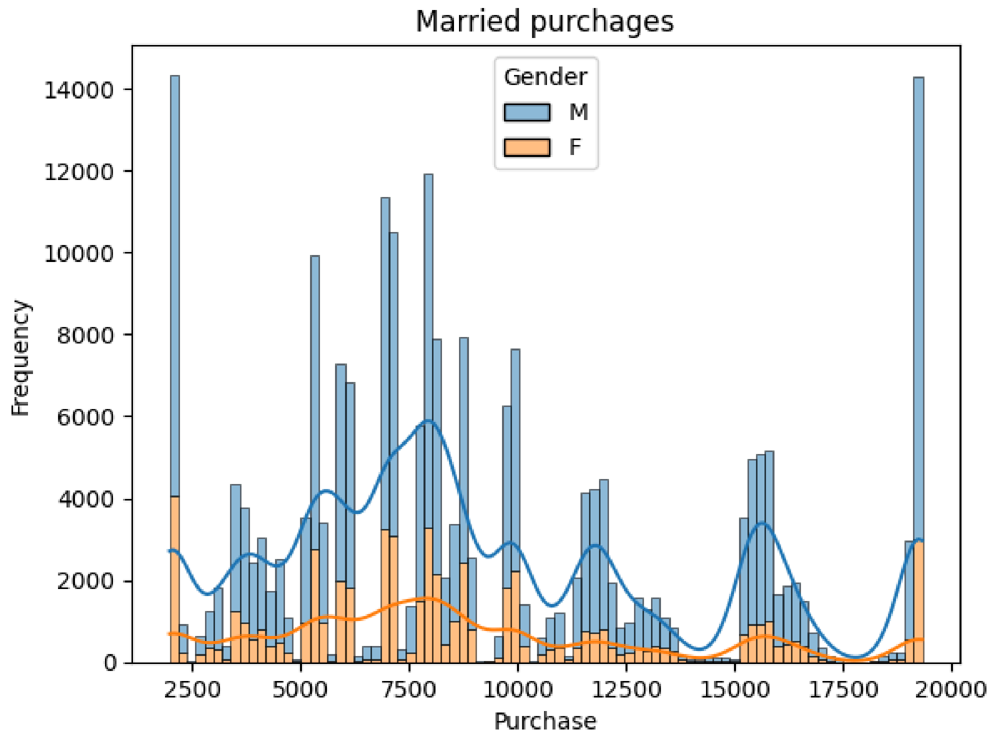
```
plt.hist(married_data['Purchase'],color='red',label='Married',histtype='step',density=True,cumulative=True)
plt.hist(unmarried_data['Purchase'],color='blue',label='Unmarried',histtype='step',density=True,cumulative=
plt.xlabel('Purchase')
plt.ylabel('Frequency')
plt.title('Married VS Unmarried purchages')
plt.legend(labels=['Married','Unmarried'])
plt.show()
```




```
plt.figure(figsize=(8,6))
sns.boxplot(x='Marital_Status',y='Purchase',hue='Gender',data=walmart_data,palette='bright')
plt.xlabel('Marital_Status')
plt.ylabel('Purchase')
plt.title('Married VS Unmarried purchages')
plt.show()
```



```
sns.histplot(data=married_data,x='Purchase',hue='Gender',multiple='stack',kde=True)
plt.xlabel('Purchase')
plt.ylabel('Frequency')
plt.title('Married purchages')
plt.show()
```



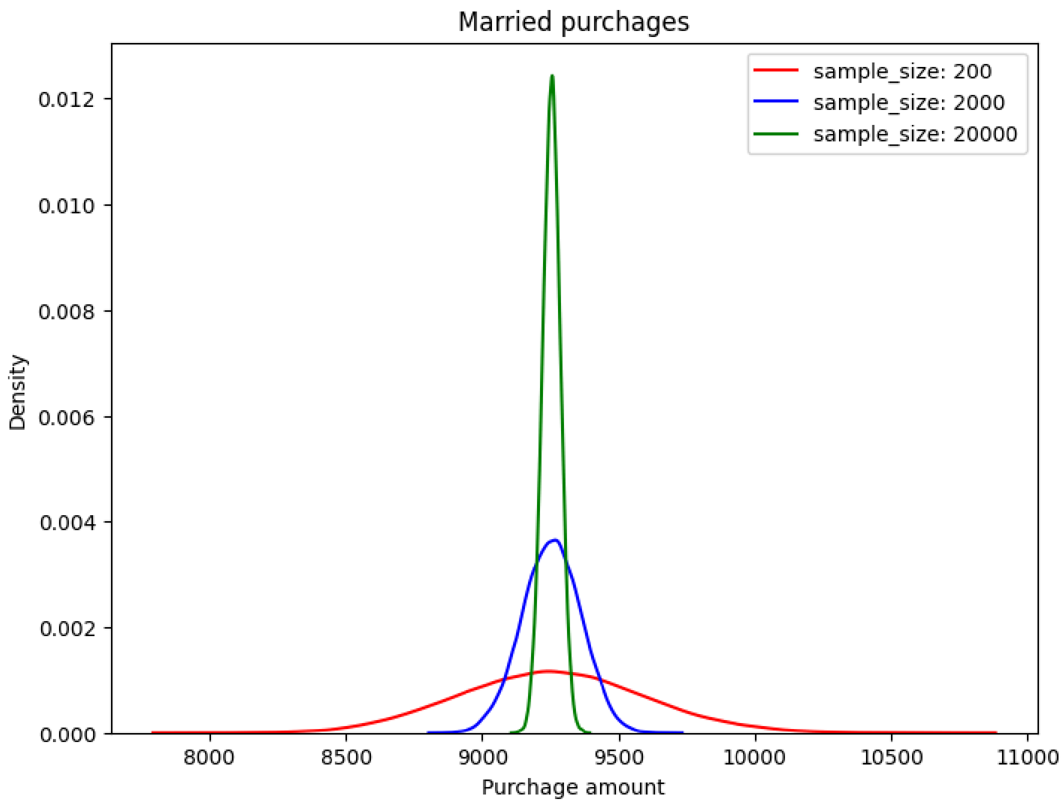
Start coding or generate with AI.

We can clerly see it doesn't follow gaussian distribution. So now applying CTL.

```
# Married users data

random.seed(42)
# sample size = 200
sample_size_200 = 200

#mean of all samples stored in sample_mean
sample_mean_200 = []

for i in range(30000):
    random_sample = random.sample(range(0, len(married_data)), sample_size_200)
    sample = married_data['Purchase'].iloc[random_sample]
    sample_mean_200.append(sample.mean())


# sample size = 2000
sample_size_2000 = 2000

#mean of all samples stored in sample_mean
sample_mean_2000 = []


for i in range(30000):
    random_sample = random.sample(range(0, len(married_data)), sample_size_2000)
    sample = married_data['Purchase'].iloc[random_sample]
    sample_mean_2000.append(sample.mean())


# sample size = 20000
sample_size_20000 = 20000

#mean of all samples stored in sample_mean
sample_mean_20000 = []


for i in range(30000):
    random_sample = random.sample(range(0, len(married_data)), sample_size_20000)
    sample = married_data['Purchase'].iloc[random_sample]
    sample_mean_20000.append(sample.mean())

#Means of all 30000 samples means
sample_mean_mean_200 = round(np.mean(sample_mean_200),2)
sample_mean_mean_2000 = round(np.mean(sample_mean_2000),2)
sample_mean_mean_20000 = round(np.mean(sample_mean_20000),2)

#std of all 30000 sample means
sample_std_200 = round(np.std(sample_mean_200),2)
population_std_200 = round(sample_std_200 * np.sqrt(sample_size_200),2)
sample_std_2000 = round(np.std(sample_mean_2000),2)
population_std_2000 = round(sample_std_2000 * np.sqrt(sample_size_2000),2)
sample_std_20000 = round(np.std(sample_mean_20000),2)
population_std_20000 = round(sample_std_20000 * np.sqrt(sample_size_20000),2)

print(f"sample_means_200: {sample_mean_mean_200}")
print(f"sample_means_2000: {sample_mean_mean_2000}")
print(f"sample_means_20000: {sample_mean_mean_20000}")
print("-"*100)
print(f"sample_std_200(std error): {sample_std_200}")
print(f"sample_std_2000(std error): {sample_std_2000}")
print(f"sample_std_20000(std error): {sample_std_20000}")
print("-"*100)
print(f"population_std_200: {population_std_200}")
print(f"population_std_2000: {population_std_2000}")
print(f"population_std_20000: {population_std_20000}")
```

```
sample_means_200: 9254.52
sample_means_2000: 9254.09
sample_means_20000: 9253.41
-----------------------------------------------------------------------------------------------------
sample_std_200(std error): 344.19
sample_std_2000(std error): 107.43
sample_std_20000(std error): 32.68
-----------------------------------------------------------------------------------------------------
population_std_200: 4867.58
population_std_2000: 4804.42
population_std_20000: 4621.65
```

```python
plt.figure(figsize=(8,6))
sns.kdeplot(sample_mean_200,color='red',label='sample_size_200')
sns.kdeplot(sample_mean_2000,color='blue',label='sample_size_2000')
sns.kdeplot(sample_mean_20000,color='green',label='sample_size_20000')
plt.xlabel('Purchage amount')
plt.ylabel('Density')
plt.title('Married purchages')
plt.legend(["sample_size: 200","sample_size: 2000","sample_size: 20000"])
plt.show()
```



```
# Marrid users average purchases at 90%, 95% and 99% confidence level

#Z scores at 90%,95% and 99%
z_score_90=norm.ppf(0.90)
z_score_95=norm.ppf(0.95)
z_score_99=norm.ppf(0.99)

# caluculating confidence intervals ranges with sample size 200.
# Caluculating confidence interval ranges at 90%
lower_bound_90=round(sample_mean_mean_200-z_score_90*sample_std_200,2)
upper_bound_90=round(sample_mean_mean_200+z_score_90*sample_std_200,2)
interval_married_200_90=(lower_bound_90,upper_bound_90)

# Caluculating confidence interval ranges at 95%
lower_bound_95=round(sample_mean_mean_200-z_score_95*sample_std_200,2)
upper_bound_95=round(sample_mean_mean_200+z_score_95*sample_std_200,2)
interval_married_200_95=(lower_bound_95,upper_bound_95)

# Caluculating confidence interval ranges at 99%
lower_bound_99=round(sample_mean_mean_200-z_score_99*sample_std_200,2)
upper_bound_99=round(sample_mean_mean_200+z_score_99*sample_std_200,2)
interval_married_200_99=(lower_bound_99,upper_bound_99)

#calculating confidence intervals ranges with sample size 2000
# Caluculating confidence interval ranges at 90%
lower_bound_90=round(sample_mean_mean_2000-z_score_90*sample_std_2000,2)
upper_bound_90=round(sample_mean_mean_2000+z_score_90*sample_std_2000,2)
interval_married_2000_90=(lower_bound_90,upper_bound_90)

#caluculating confidence interval ranges at 95%
lower_bound_95=round(sample_mean_mean_2000-z_score_95*sample_std_2000,2)
upper_bound_95=round(sample_mean_mean_2000+z_score_95*sample_std_2000,2)
interval_married_2000_95=(lower_bound_95,upper_bound_95)

# Calculating confidence interval ranges at 99%
lower_bound_99=round(sample_mean_mean_2000-z_score_99*sample_std_2000,2)
upper_bound_99=round(sample_mean_mean_2000+z_score_99*sample_std_2000,2)
interval_married_2000_99=(lower_bound_99,upper_bound_99)

#calculating confidence intervals ranges with sample size 20000
# Caluculating confidence interval ranges at 90%
lower_bound_90=round(sample_mean_mean_20000-z_score_90*sample_std_20000,2)
upper_bound_90=round(sample_mean_mean_20000+z_score_90*sample_std_20000,2)
interval_married_20000_90=(lower_bound_90,upper_bound_90)

#caluculating confidence interval ranges at 95%
lower_bound_95=round(sample_mean_mean_20000-z_score_95*sample_std_20000,2)
upper_bound_95=round(sample_mean_mean_20000+z_score_95*sample_std_20000,2)
interval_married_20000_95=(lower_bound_95,upper_bound_95)

# Calculating confidence interval ranges at 99%
lower_bound_99=round(sample_mean_mean_20000-z_score_99*sample_std_20000,2)
upper_bound_99=round(sample_mean_mean_20000+z_score_99*sample_std_20000,2)
interval_married_20000_99=(lower_bound_99,upper_bound_99)

print(f"Married users average purchases at 90% confidence interval range(sample_size=200): {interval_marrie
print(f"Married users average purchases at 95% confidence interval range(sample_size=200): {interval_marrie
print(f"Married users average purchases at 99% confidence interval range(sample_size=200): {interval_marrie
print("-"*100)
print(f"Married users average purchases at 90% confidence interval range(sample_size=2000): {interval_marri
print(f"Married users average purchases at 95% confidence interval range(sample_size=2000): {interval_marri
print(f"Married users average purchases at 99% confidence interval range(sample_size=2000): {interval_marri
print("-"*100)
print(f"Married users average purchases at 90% confidence interval range(sample_size=20000): {interval_marr
print(f"Married users average purchases at 95% confidence interval range(sample_size=20000): {interval_marr
```

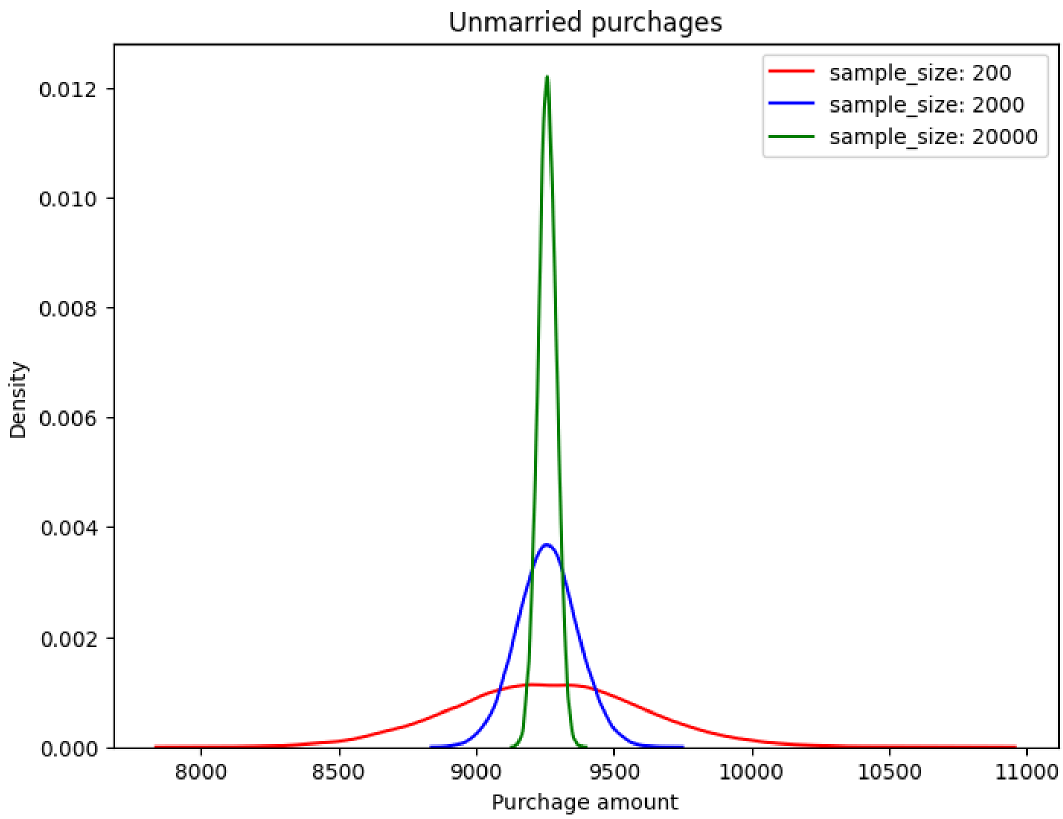
```
print(f"Married users average purchases at 99% confidence interval range(sample_size=20000): {interval_marr
```

```
Married users average purchases at 90% confidence interval range(sample_size=200): (8813.42, 9695.62)
Married users average purchases at 95% confidence interval range(sample_size=200): (8688.38, 9820.66)
Married users average purchases at 99% confidence interval range(sample_size=200): (8453.81, 10055.23)
----------------------------------------------------------------------------------------------------
Married users average purchases at 90% confidence interval range(sample_size=2000): (9116.41, 9391.77)
Married users average purchases at 95% confidence interval range(sample_size=2000): (9077.38, 9430.8)
Married users average purchases at 99% confidence interval range(sample_size=2000): (9004.17, 9504.01)
----------------------------------------------------------------------------------------------------
Married users average purchases at 90% confidence interval range(sample_size=20000): (9211.53, 9295.29)
Married users average purchases at 95% confidence interval range(sample_size=20000): (9199.66, 9307.16)
Married users average purchases at 99% confidence interval range(sample_size=20000): (9177.38, 9329.44)
```

```
# Unmarried users data

random.seed(42)
# sample size = 200
sample_size_200 = 200

#mean of all samples stored in sample_mean
sample_mean_200 = []

for i in range(30000):
    random_sample = random.sample(range(0, len(unmarried_data)), sample_size_200)
    sample = unmarried_data['Purchase'].iloc[random_sample]
    sample_mean_200.append(sample.mean())


# sample size = 2000
sample_size_2000 = 2000

#mean of all samples stored in sample_mean
sample_mean_2000 = []


for i in range(30000):
    random_sample = random.sample(range(0, len(unmarried_data)), sample_size_2000)
    sample = unmarried_data['Purchase'].iloc[random_sample]
    sample_mean_2000.append(sample.mean())


# sample size = 20000
sample_size_20000 = 20000

#mean of all samples stored in sample_mean
sample_mean_20000 = []


for i in range(30000):
    random_sample = random.sample(range(0, len(unmarried_data)), sample_size_20000)
    sample = unmarried_data['Purchase'].iloc[random_sample]
    sample_mean_20000.append(sample.mean())

#Means of all 30000 samples means
sample_mean_mean_200 = round(np.mean(sample_mean_200),2)
sample_mean_mean_2000 = round(np.mean(sample_mean_2000),2)
sample_mean_mean_20000 = round(np.mean(sample_mean_20000),2)

#std of all 30000 sample means
sample_std_200 = round(np.std(sample_mean_200),2)
population_std_200 = round(sample_std_200 * np.sqrt(sample_size_200),2)
sample_std_2000 = round(np.std(sample_mean_2000),2)
population_std_2000 = round(sample_std_2000 * np.sqrt(sample_size_2000),2)
sample_std_20000 = round(np.std(sample_mean_20000),2)
population_std_20000 = round(sample_std_20000 * np.sqrt(sample_size_20000),2)

print(f"sample_means_200: {sample_mean_mean_200}")
print(f"sample_means_2000: {sample_mean_mean_2000}")
print(f"sample_means_20000: {sample_mean_mean_20000}")
print("-"*100)
print(f"sample_std_200(std error): {sample_std_200}")
print(f"sample_std_2000(std error): {sample_std_2000}")
print(f"sample_std_20000(std error): {sample_std_20000}")
print("-"*100)
print(f"population_std_200: {population_std_200}")
print(f"population_std_2000: {population_std_2000}")
print(f"population_std_20000: {population_std_20000}")
```

```
sample_means_200: 9258.78
sample_means_2000: 9259.46
sample_means_20000: 9259.1
-----------------------------------------------------------------------------------------------------
sample_std_200(std error): 341.34
sample_std_2000(std error): 108.77
sample_std_20000(std error): 33.18
-----------------------------------------------------------------------------------------------------
population_std_200: 4827.28
population_std_2000: 4864.34
population_std_20000: 4692.36
```

```
plt.figure(figsize=(8,6))
sns.kdeplot(sample_mean_200,color='red',label='sample_size_200')
sns.kdeplot(sample_mean_2000,color='blue',label='sample_size_2000')
sns.kdeplot(sample_mean_20000,color='green',label='sample_size_20000')
plt.xlabel('Purchage amount')
plt.ylabel('Density')
plt.title('Unmarried purchages')
plt.legend(["sample_size: 200","sample_size: 2000","sample_size: 20000"])
plt.show()
```



```
# Unmarrid users average purchases at 90%, 95% and 99% confidence level

#Z scores at 90%,95% and 99%
z_score_90=norm.ppf(0.90)
z_score_95=norm.ppf(0.95)
z_score_99=norm.ppf(0.99)

# caluculating confidence intervals ranges with sample size 200.
# Caluculating confidence interval ranges at 90%
lower_bound_90=round(sample_mean_mean_200-z_score_90*sample_std_200,2)
upper_bound_90=round(sample_mean_mean_200+z_score_90*sample_std_200,2)
interval_unmarried_200_90=(lower_bound_90,upper_bound_90)

# Caluculating confidence interval ranges at 95%
lower_bound_95=round(sample_mean_mean_200-z_score_95*sample_std_200,2)
upper_bound_95=round(sample_mean_mean_200+z_score_95*sample_std_200,2)
interval_unmarried_200_95=(lower_bound_95,upper_bound_95)

# Caluculating confidence interval ranges at 99%
lower_bound_99=round(sample_mean_mean_200-z_score_99*sample_std_200,2)
upper_bound_99=round(sample_mean_mean_200+z_score_99*sample_std_200,2)
interval_unmarried_200_99=(lower_bound_99,upper_bound_99)

#calculating confidence intervals ranges with sample size 2000
# Caluculating confidence interval ranges at 90%
lower_bound_90=round(sample_mean_mean_2000-z_score_90*sample_std_2000,2)
upper_bound_90=round(sample_mean_mean_2000+z_score_90*sample_std_2000,2)
interval_unmarried_2000_90=(lower_bound_90,upper_bound_90)

#caluculating confidence interval ranges at 95%
lower_bound_95=round(sample_mean_mean_2000-z_score_95*sample_std_2000,2)
upper_bound_95=round(sample_mean_mean_2000+z_score_95*sample_std_2000,2)
interval_unmarried_2000_95=(lower_bound_95,upper_bound_95)

# Calculating confidence interval ranges at 99%
lower_bound_99=round(sample_mean_mean_2000-z_score_99*sample_std_2000,2)
upper_bound_99=round(sample_mean_mean_2000+z_score_99*sample_std_2000,2)
interval_unmarried_2000_99=(lower_bound_99,upper_bound_99)

#calculating confidence intervals ranges with sample size 20000
# Caluculating confidence interval ranges at 90%
lower_bound_90=round(sample_mean_mean_20000-z_score_90*sample_std_20000,2)
upper_bound_90=round(sample_mean_mean_20000+z_score_90*sample_std_20000,2)
interval_unmarried_20000_90=(lower_bound_90,upper_bound_90)

#caluculating confidence interval ranges at 95%
lower_bound_95=round(sample_mean_mean_20000-z_score_95*sample_std_20000,2)
upper_bound_95=round(sample_mean_mean_20000+z_score_95*sample_std_20000,2)
interval_unmarried_20000_95=(lower_bound_95,upper_bound_95)

# Calculating confidence interval ranges at 99%
lower_bound_99=round(sample_mean_mean_20000-z_score_99*sample_std_20000,2)
upper_bound_99=round(sample_mean_mean_20000+z_score_99*sample_std_20000,2)
interval_unmarried_20000_99=(lower_bound_99,upper_bound_99)

print(f"Unmarried users average purchases at 90% confidence interval range(sample_size=200): {interval_unma
print(f"Unmarried users average purchases at 95% confidence interval range(sample_size=200): {interval_unma
print(f"Unmarried users average purchases at 99% confidence interval range(sample_size=200): {interval_unma
print("-"*100)
print(f"Unmarried users average purchases at 90% confidence interval range(sample_size=2000): {interval_unm
print(f"Unmarried users average purchases at 95% confidence interval range(sample_size=2000): {interval_unm
print(f"Unmarried users average purchases at 99% confidence interval range(sample_size=2000): {interval_unm
print("-"*100)
print(f"Unmarried users average purchases at 90% confidence interval range(sample_size=20000): {interval_ur
print(f"Unmarried users average purchases at 95% confidence interval range(sample_size=20000): {interval_ur
```

```
print(f"Unmarried users average purchases at 99% confidence interval range(sample_size=20000): {interval_ur
```

```
Unmarried users average purchases at 90% confidence interval range(sample_size=200): (8821.34, 9696.22)
Unmarried users average purchases at 95% confidence interval range(sample_size=200): (8697.33, 9820.23)
Unmarried users average purchases at 99% confidence interval range(sample_size=200): (8464.7, 10052.86)
----------------------------------------------------------------------------------------------------
Unmarried users average purchases at 90% confidence interval range(sample_size=2000): (9120.07, 9398.85
Unmarried users average purchases at 95% confidence interval range(sample_size=2000): (9080.55, 9438.37
Unmarried users average purchases at 99% confidence interval range(sample_size=2000): (9006.42, 9512.5)
----------------------------------------------------------------------------------------------------
Unmarried users average purchases at 90% confidence interval range(sample_size=20000): (9216.58, 9301.6
Unmarried users average purchases at 95% confidence interval range(sample_size=20000): (9204.52, 9313.6
Unmarried users average purchases at 99% confidence interval range(sample_size=20000): (9181.91, 9336.2
```

We can verify the married amd unmarried mean purchage amount **using 2 sample z test.**

```
# 2 sample z-test

# Case-1
#Null Hypothesis H0: married & unmarried people average purchage amount is equal
#Alternative Hypothesis Ha: married & unmarried people having different average purchage amount

z_statastics,p_value=ztest(x1=married_data['Purchase'],x2=unmarried_data['Purchase'],value=0)

def Hypothesis_test(z_statastics,p_value,alpha):
  print(f"statastics: {z_statastics}")
  print(f"p_value: {p_value}")
  if p_value < alpha:
    print('Reject Null Hypothesis at 95% confidence \nAverage married people spent amount is different than
  else:
    print('Fail to reject Null Hypothesis at 95% confidence \nAverage married people spent amount is same a
    print("-"*100)

# At 95% confidence level
alpha=0.05
Hypothesis_test(z_statastics,p_value,alpha)

# At 90% confidence interval
alpha=0.1
Hypothesis_test(z_statastics,p_value,alpha)

# At 99% confidence interval
alpha=0.01
Hypothesis_test(z_statastics,p_value,alpha)
```

```
statastics: -0.3868627106847199
p_value: 0.6988578483633914
Fail to reject Null Hypothesis at 95% confidence
Average married people spent amount is same as Average unmarried people spent amount

----------------------------------------------------------------------------------------------------
statastics: -0.3868627106847199
p_value: 0.6988578483633914
Fail to reject Null Hypothesis at 95% confidence
Average married people spent amount is same as Average unmarried people spent amount

----------------------------------------------------------------------------------------------------
statastics: -0.3868627106847199
p_value: 0.6988578483633914
Fail to reject Null Hypothesis at 95% confidence
Average married people spent amount is same as Average unmarried people spent amount

----------------------------------------------------------------------------------------------------
```

Start coding or generate with AI.

Insights:

From the CLT with sample sizes 200, 2000 and 20000 and 30000 samples. we could see the confidence interval ranges at 90%, 95% and 99%.

Married people data:

- sample mean purchase amount with sample size 200 = 9254.52
- sample mean purchase amount with sample size 2000 = 9254.09
- sample mean purchase amount with sample size 20000 = 9253.41

- sample standard deviation(std error) with sample size 200 = 344.19
- sample standard deviation(std error) with sample size 2000 = 107.43
- sample standard deviation(std error) with sample size 20000 = 32.68

- married population standard deviation with sample size 200 = 4867.58
- married population standard deviation with sample size 2000 = 4804.42
- married population standard deviation with sample size 20000 = 4621.65

- Married users average purchases at 90% confidence interval range(sample_size=200): (8813.42, 9695.62)
- Married users average purchases at 95% confidence interval range(sample_size=200): (8688.38, 9820.66)
- Married users average purchases at 99% confidence interval range(sample_size=200): (8453.81, 10055.23)

- Married users average purchases at 90% confidence interval range(sample_size=2000): (9116.41, 9391.77)
- Married users average purchases at 95% confidence interval range(sample_size=2000): (9077.38, 9430.8)
- Married users average purchases at 99% confidence interval range(sample_size=2000): (9004.17, 9504.01)

- Married users average purchases at 90% confidence interval range(sample_size=20000): (9211.53, 9295.29)
- Married users average purchases at 95% confidence interval range(sample_size=20000): (9199.66, 9307.16)
- Married users average purchases at 99% confidence interval range(sample_size=20000): (9177.38, 9329.44)

Unmarried people data:

- sample mean purchage amount with sample size 200 = 9258.78
- sample mean purchage amount with sample size 2000 = 9259.46
- sample mean purchage amount with sample size 20000 = 9259.1

- sample standard deviation(std error) with sample size 200 = 341.34
- sample standard deviation(std error) with sample size 2000 = 108.77
- sample standard deviation(std error) with sample size 20000 = 33.18

- married population standard deviation with sample size 200 = 4827.28
- married population standard deviation with sample size 2000 = 4864.34
- married population standard deviation with sample size 20000 = 4692.36

- Unmarried users average purchases at 90% confidence interval range(sample_size=200): (8821.34, 9696.22)
- Unmarried users average purchases at 95% confidence interval range(sample_size=200): (8697.33, 9820.23)
- Unmarried users average purchases at 99% confidence interval range(sample_size=200): (8464.7, 10052.86)

- Unmarried users average purchases at 90% confidence interval range(sample_size=2000): (9120.07, 9398.85)
- Unmarried users average purchases at 95% confidence interval range(sample_size=2000): (9080.55, 9438.37)
- Unmarried users average purchases at 99% confidence interval range(sample_size=2000): (9006.42, 9512.5)

- Unmarried users average purchases at 90% confidence interval range(sample_size=20000): (9216.58, 9301.62)
- Unmarried users average purchases at 95% confidence interval range(sample_size=20000): (9204.52, 9313.68)
- Unmarried users average purchases at 99% confidence interval range(sample_size=20000): (9181.91, 9336.29)

Conclusion:

As per the above confidence intervals at 90%,95% and 99% with diffrent sample sizes 200, 2000 and 2000, There is a overlapping average purchage ranges between married and unmarried.

Married and unmarried mean spent amount overlaping ranges at 90%,95% and 99% confidence intervals. And we can see married and unmarried most likely equally purchages.

Verified same by 2-sample z-test hypothesis testing, there is no significant evidence to reject null hypothesis at 90%, 95% and 99% confidence level means Average married people spent amount is same as Average unmarried people spent amount

Same analysis on different age group people

```
walmart_data['Age'].unique()
```

```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

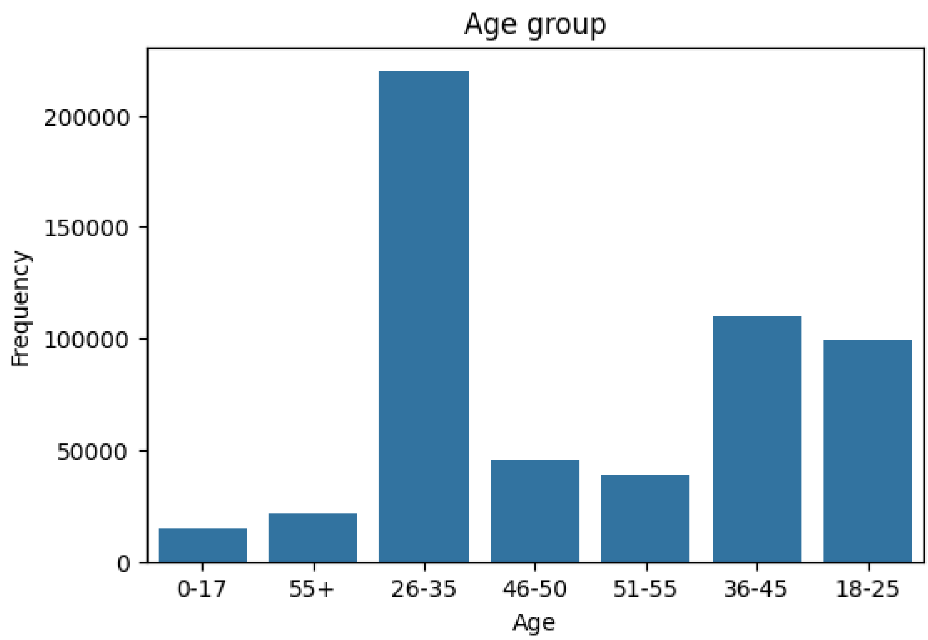
```
walmart_data['Age'].value_counts()
```

Age	count
26-35	219587
36-45	110013
18-25	99660
46-50	45701
51-55	38501
55+	21504
0-17	15102

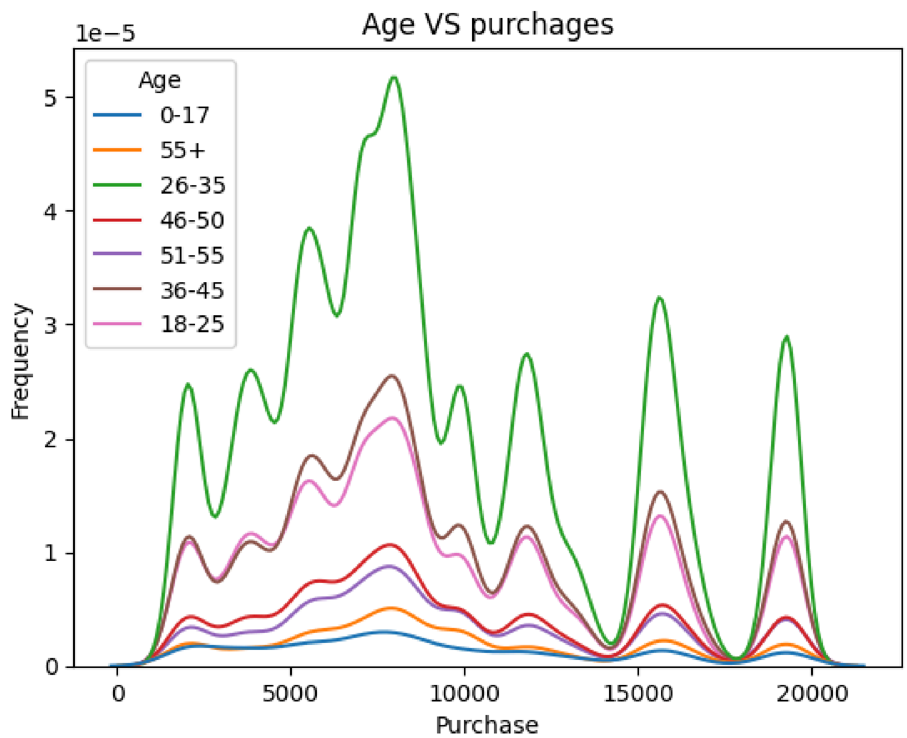
dtype: int64

```
# different age group data records saved separately.

age_0_17=walmart_data[walmart_data['Age']=='0-17']
age_18_25=walmart_data[walmart_data['Age']=='18-25']
age_26_35=walmart_data[walmart_data['Age']=='26-35']
age_36_45=walmart_data[walmart_data['Age']=='36-45']
age_46_50=walmart_data[walmart_data['Age']=='46-50']
age_51_55=walmart_data[walmart_data['Age']=='51-55']
age_55_above=walmart_data[walmart_data['Age']=='55+']
```

```
plt.figure(figsize=(6,4))
sns.countplot(x='Age',data=walmart_data)
plt.title('Age group')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```




```
sns.kdeplot(data=walmart_data,x='Purchase',hue='Age')
plt.xlabel('Purchase')
plt.ylabel('Frequency')
plt.title('Age VS purchages')
plt.show()
```



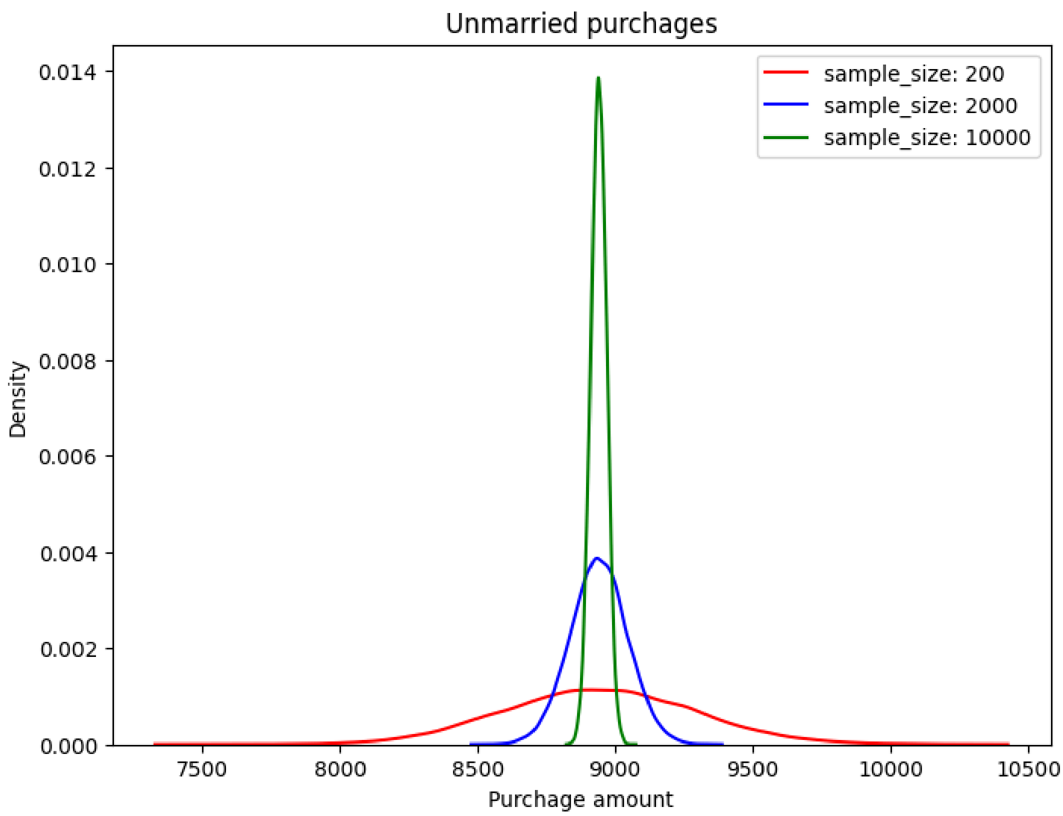
Use CLT to get normal distribution

same way take 30000 samples with different sample sizes 200, 2000 and 20000.

```
# Different age group people data

def populate_sample_mean(data,sample_size):
  random.seed(42)
  #mean of all samples stored in sample_mean
  sample_mean = []

  for i in range(30000):
      random_sample = random.sample(range(0, len(data)), sample_size)
      sample = data['Purchase'].iloc[random_sample]
      sample_mean.append(sample.mean())
  return sample_mean


def calculate_sample_mean_std(sample_mean):
  sample_mean_mean = round(np.mean(sample_mean),2)
  sample_std = round(np.std(sample_mean),2)
  population_std = round((sample_std * np.sqrt(sample_size)),2)
  return (sample_mean_mean,sample_std,population_std)

def print_sample_mean_std(sample_mean_mean,sample_std,population_std,sample_size):
  print(f"sample_means_{sample_size}: {sample_mean_mean}")
  print(f"sample_std_{sample_size}(std error): {sample_std}")
  print(f"population_std_{sample_size}: {population_std}")
  print("-"*100)

# Age group = 0-17

 #sample size = 200
sample_size_200 = 200
sample_mean_200 = populate_sample_mean(age_0_17,sample_size_200)

#sample size = 2000
sample_size_2000 = 2000
sample_mean_2000 = populate_sample_mean(age_0_17,sample_size_2000)

#sample size = 10000
sample_size_10000 = 10000
sample_mean_10000 = populate_sample_mean(age_0_17,sample_size_10000)

#Means & std of all 30000 samples means and population(group-(0-17)) std.
sample_mean_mean_200,sample_std_200,population_std_200 = calculate_sample_mean_std(sample_mean_200)
sample_mean_mean_2000,sample_std_2000,population_std_2000 = calculate_sample_mean_std(sample_mean_2000)
sample_mean_mean_10000,sample_std_10000,population_std_10000 = calculate_sample_mean_std(sample_mean_10000)

# print all the data
print_sample_mean_std(sample_mean_mean_200,sample_std_200,population_std_200,sample_size_200)
print_sample_mean_std(sample_mean_mean_2000,sample_std_2000,population_std_2000,sample_size_2000)
print_sample_mean_std(sample_mean_mean_10000,sample_std_10000,population_std_10000,sample_size_10000)
```

```
sample_means_200: 8941.41
sample_std_200(std error): 345.09
population_std_200: 4880.31
----------------------------------------------------------------------------------------------------
sample_means_2000: 8940.96
sample_std_2000(std error): 103.39
population_std_2000: 1462.16
----------------------------------------------------------------------------------------------------
sample_means_10000: 8940.63
sample_std_10000(std error): 28.62
population_std_10000: 404.75
----------------------------------------------------------------------------------------------------
```

```
plt.figure(figsize=(8,6))
sns.kdeplot(sample_mean_200,color='red',label='sample_size_200')
sns.kdeplot(sample_mean_2000,color='blue',label='sample_size_2000')
sns.kdeplot(sample_mean_10000,color='green',label='sample_size_10000')
plt.xlabel('Purchage amount')
plt.ylabel('Density')
plt.title('Unmarried purchages')
plt.legend(["sample_size: 200","sample_size: 2000","sample_size: 10000"])
plt.show()
```



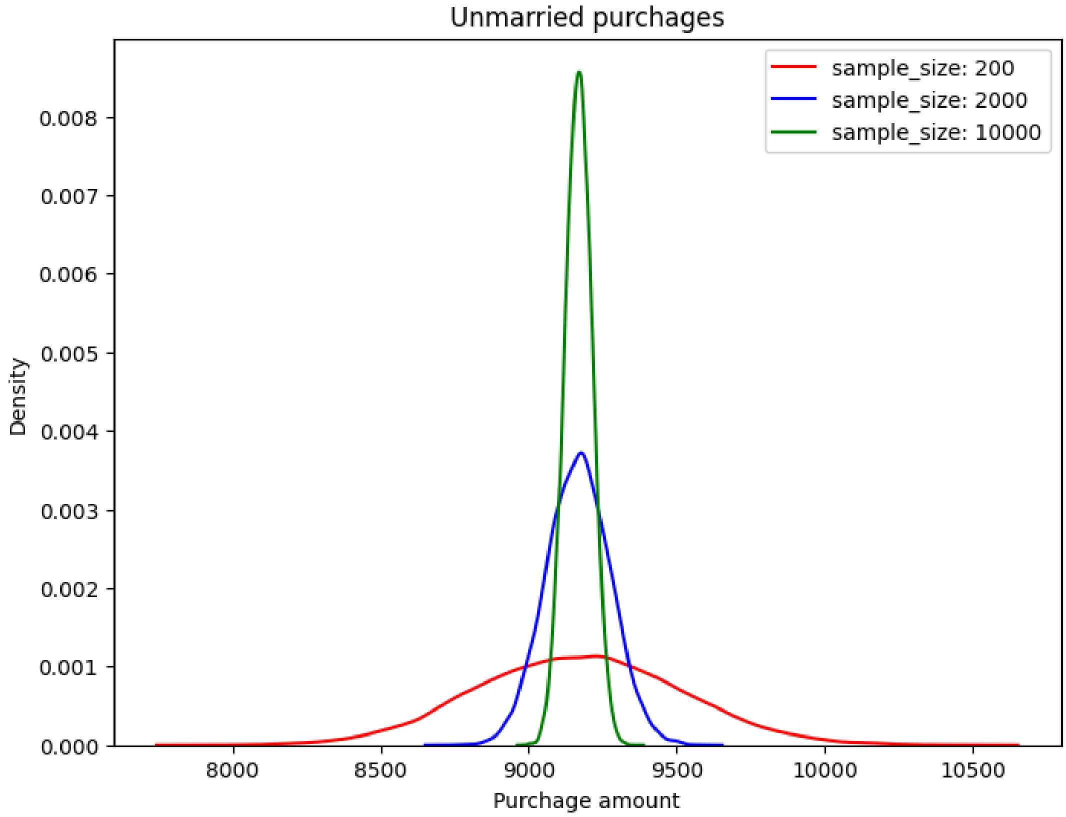
```
# Age group 0-17
age_group='0-17'
def confidence_interval(sample_mean_mean,sample_std,sample_size):
  #Z score at 90%, 95% and 99%
  z_score_90=norm.ppf(0.90)
  z_score_95=norm.ppf(0.95)
  z_score_99=norm.ppf(0.99)

  # Caluculating confidence interval ranges at 90%
  lower_bound_90=round(sample_mean_mean-z_score_90*sample_std,2)
  upper_bound_90=round(sample_mean_mean+z_score_90*sample_std,2)
  interval_90=(lower_bound_90,upper_bound_90)

  ## Caluculating confidence interval ranges at 95%
  lower_bound_95=round(sample_mean_mean-z_score_95*sample_std,2)
  upper_bound_95=round(sample_mean_mean+z_score_95*sample_std,2)
  interval_95=(lower_bound_95,upper_bound_95)

  ## Caluculating confidence interval ranges at 99%
  lower_bound_99=round(sample_mean_mean-z_score_99*sample_std,2)
  upper_bound_99=round(sample_mean_mean+z_score_99*sample_std,2)
  interval_99=(lower_bound_99,upper_bound_99)

  return (interval_90,interval_95,interval_99)

def print_confidence_interval(interval_90,interval_95,interval_99,sample_size,age_group):
  print(f"Age group {age_group} average purchases at 90% confidence interval range(sample_size={sample_size
  print(f"Age group {age_group} average purchases at 95% confidence interval range(sample_size={sample_size
  print(f"Age group {age_group} average purchases at 99% confidence interval range(sample_size={sample_size
  print("-"*100)


# caluculating confidence intervals ranges with sample size 200.
interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99 = confidence_interval(sample_mean_mean_200,s
# Printing the data
print_confidence_interval(interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99,sample_size_200,ag

#calculating confidence intervals ranges with sample size 2000
interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99 = confidence_interval(sample_mean_mean_2000,
# Printing the data
print_confidence_interval(interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99,sample_size_2000,a

#calculating confidence intervals ranges with sample size 10000
interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99 = confidence_interval(sample_mean_mean_1000
# Printing the data
print_confidence_interval(interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99,sample_size_10000,
```

```
Age group 0-17 average purchases at 90% confidence interval range(sample_size=200): (8499.16, 9383.66)
Age group 0-17 average purchases at 95% confidence interval range(sample_size=200): (8373.79, 9509.03)
Age group 0-17 average purchases at 99% confidence interval range(sample_size=200): (8138.61, 9744.21)
----------------------------------------------------------------------------------------------------
Age group 0-17 average purchases at 90% confidence interval range(sample_size=2000): (8808.46, 9073.46)
Age group 0-17 average purchases at 95% confidence interval range(sample_size=2000): (8770.9, 9111.02)
Age group 0-17 average purchases at 99% confidence interval range(sample_size=2000): (8700.44, 9181.48)
----------------------------------------------------------------------------------------------------
Age group 0-17 average purchases at 90% confidence interval range(sample_size=10000): (8903.95, 8977.31
Age group 0-17 average purchases at 95% confidence interval range(sample_size=10000): (8893.55, 8987.71
Age group 0-17 average purchases at 99% confidence interval range(sample_size=10000): (8874.05, 9007.21
----------------------------------------------------------------------------------------------------
```

```
# Age group = 18-25

#sample size = 200
sample_size_200 = 200
sample_mean_200 = populate_sample_mean(age_18_25,sample_size_200)

#sample size = 2000
sample_size_2000 = 2000
sample_mean_2000 = populate_sample_mean(age_18_25,sample_size_2000)

#sample size = 10000
sample_size_10000 = 10000
sample_mean_10000 = populate_sample_mean(age_18_25,sample_size_10000)

#Means & std of all 30000 samples means and population(group-(18-25)) std.
sample_mean_mean_200,sample_std_200,population_std_200 = calculate_sample_mean_std(sample_mean_200)
sample_mean_mean_2000,sample_std_2000,population_std_2000 = calculate_sample_mean_std(sample_mean_2000)
sample_mean_mean_10000,sample_std_10000,population_std_10000 = calculate_sample_mean_std(sample_mean_10000)

# print all the data
print_sample_mean_std(sample_mean_mean_200,sample_std_200,population_std_200,sample_size_200)
print_sample_mean_std(sample_mean_mean_2000,sample_std_2000,population_std_2000,sample_size_2000)
print_sample_mean_std(sample_mean_mean_10000,sample_std_10000,population_std_10000,sample_size_10000)
```

```
sample_means_200: 9169.82
sample_std_200(std error): 344.99
population_std_200: 4878.9
----------------------------------------------------------------------------------------------------
sample_means_2000: 9168.46
sample_std_2000(std error): 107.7
population_std_2000: 1523.11
----------------------------------------------------------------------------------------------------
sample_means_10000: 9169.08
sample_std_10000(std error): 45.95
population_std_10000: 649.83
----------------------------------------------------------------------------------------------------
```

```
plt.figure(figsize=(8,6))
sns.kdeplot(sample_mean_200,color='red',label='sample_size_200')
sns.kdeplot(sample_mean_2000,color='blue',label='sample_size_2000')
sns.kdeplot(sample_mean_10000,color='green',label='sample_size_10000')
plt.xlabel('Purchage amount')
plt.ylabel('Density')
plt.title('Unmarried purchages')
plt.legend(["sample_size: 200","sample_size: 2000","sample_size: 10000"])
plt.show()
```



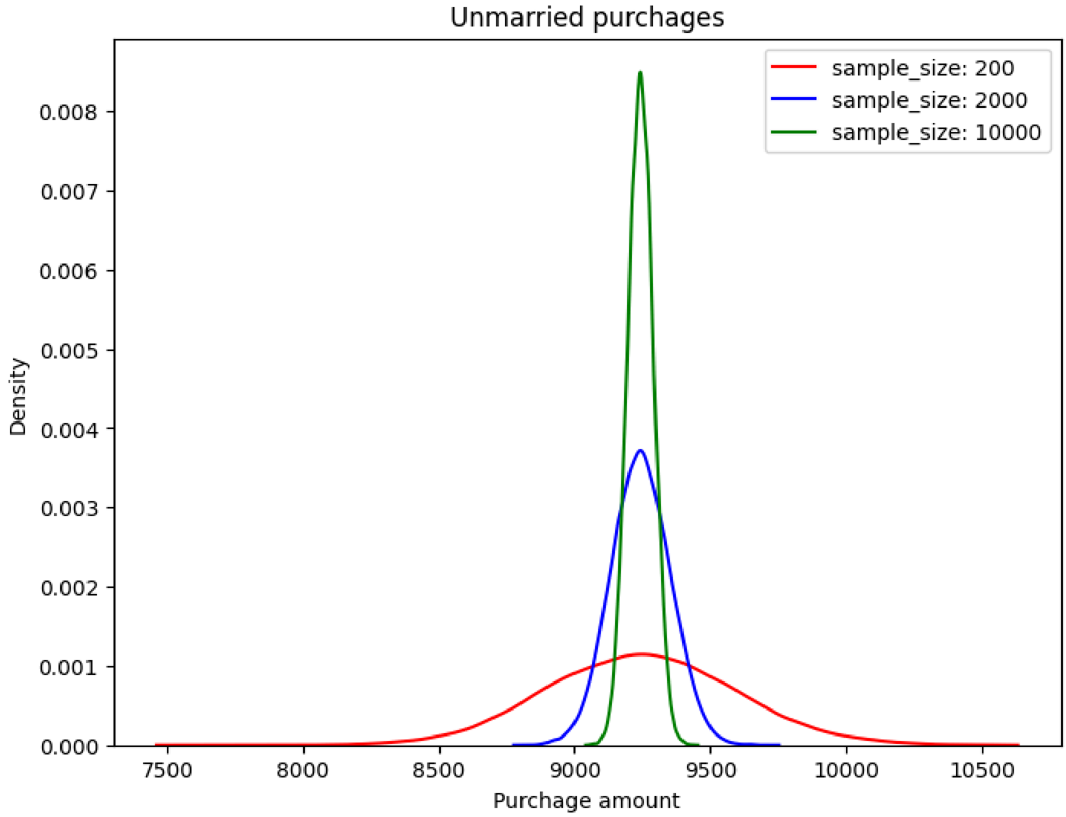

```
# Age group 18-25
age_group='18-25'

# caluculating confidence intervals ranges with sample size 200.
interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99 = confidence_interval(sample_mean_mean_200,s
# Printing the data
print_confidence_interval(interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99,sample_size_200,ag

#calculating confidence intervals ranges with sample size 2000
innterval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99 = confidence_interval(sample_mean_mean_2000
# Printing the data
print_confidence_interval(interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99,sample_size_2000,a

#calculating confidence intervals ranges with sample size 10000
interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99 = confidence_interval(sample_mean_mean_10000
# Printing the data
print_confidence_interval(interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99,sample_size_10000,
```

```
Age group 18-25 average purchases at 90% confidence interval range(sample_size=200): (8727.7, 9611.94)
Age group 18-25 average purchases at 95% confidence interval range(sample_size=200): (8602.36, 9737.28)
Age group 18-25 average purchases at 99% confidence interval range(sample_size=200): (8367.25, 9972.39)
----------------------------------------------------------------------------------------------------
Age group 18-25 average purchases at 90% confidence interval range(sample_size=2000): (8727.7, 9611.94)
Age group 18-25 average purchases at 95% confidence interval range(sample_size=2000): (8991.31, 9345.61
Age group 18-25 average purchases at 99% confidence interval range(sample_size=2000): (8917.91, 9419.01
----------------------------------------------------------------------------------------------------
Age group 18-25 average purchases at 90% confidence interval range(sample_size=10000): (9110.19, 9227.9
Age group 18-25 average purchases at 95% confidence interval range(sample_size=10000): (9093.5, 9244.66
Age group 18-25 average purchases at 99% confidence interval range(sample_size=10000): (9062.18, 9275.9
----------------------------------------------------------------------------------------------------
```

```
# Age group = 26-35

#sample size = 200
sample_size_200 = 200
sample_mean_200 = populate_sample_mean(age_26_35,sample_size_200)

#sample size = 2000
sample_size_2000 = 2000
sample_mean_2000 = populate_sample_mean(age_26_35,sample_size_2000)

#sample size = 10000
sample_size_10000 = 10000
sample_mean_10000 = populate_sample_mean(age_26_35,sample_size_10000)

#Means & std of all 30000 samples means and population(group-(26-35)) std.
sample_mean_mean_200,sample_std_200,population_std_200 = calculate_sample_mean_std(sample_mean_200)
sample_mean_mean_2000,sample_std_2000,population_std_2000 = calculate_sample_mean_std(sample_mean_2000)
sample_mean_mean_10000,sample_std_10000,population_std_10000 = calculate_sample_mean_std(sample_mean_10000)

# print all the data
print_sample_mean_std(sample_mean_mean_200,sample_std_200,population_std_200,sample_size_200)
print_sample_mean_std(sample_mean_mean_2000,sample_std_2000,population_std_2000,sample_size_2000)
print_sample_mean_std(sample_mean_mean_10000,sample_std_10000,population_std_10000,sample_size_10000)
```

```
sample_means_200: 9245.23
sample_std_200(std error): 345.63
population_std_200: 4887.95
----------------------------------------------------------------------------------------------------
sample_means_2000: 9243.69
sample_std_2000(std error): 107.96
population_std_2000: 1526.78
----------------------------------------------------------------------------------------------------
sample_means_10000: 9243.48
sample_std_10000(std error): 47.1
population_std_10000: 666.09
----------------------------------------------------------------------------------------------------
```

```
plt.figure(figsize=(8,6))
sns.kdeplot(sample_mean_200,color='red',label='sample_size_200')
sns.kdeplot(sample_mean_2000,color='blue',label='sample_size_2000')
sns.kdeplot(sample_mean_10000,color='green',label='sample_size_10000')
plt.xlabel('Purchage amount')
plt.ylabel('Density')
plt.title('Unmarried purchages')
plt.legend(["sample_size: 200","sample_size: 2000","sample_size: 10000"])
plt.show()
```



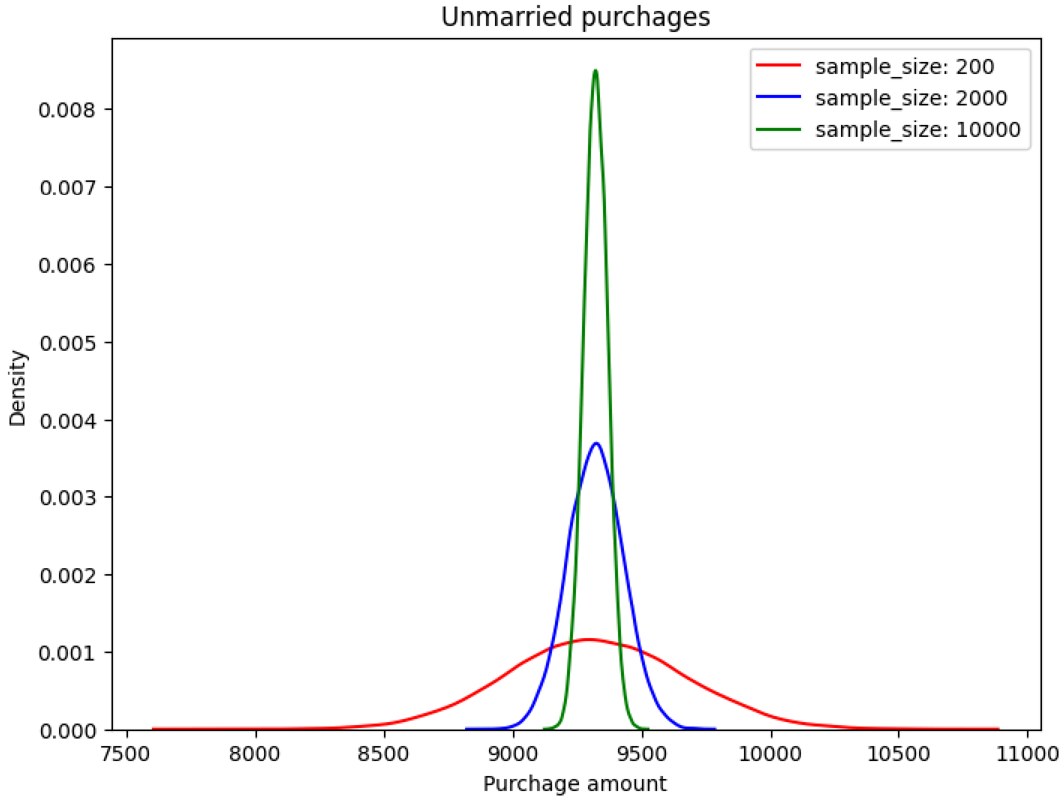

```
# Age group 26-35
age_group='26-35'

# caluculating confidence intervals ranges with sample size 200.
interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99 = confidence_interval(sample_mean_mean_200,s
# Printing the data
print_confidence_interval(interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99,sample_size_200,ag

#calculating confidence intervals ranges with sample size 2000
innterval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99 = confidence_interval(sample_mean_mean_2000
# Printing the data
print_confidence_interval(interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99,sample_size_2000,a

#calculating confidence intervals ranges with sample size 10000
interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99 = confidence_interval(sample_mean_mean_10000
# Printing the data
print_confidence_interval(interval_age_0_17_90,interval_age_0_17_95,interval_age_0_17_99,sample_size_10000,
```

```
Age group 26-35 average purchases at 90% confidence interval range(sample_size=200): (8802.29, 9688.17)
Age group 26-35 average purchases at 95% confidence interval range(sample_size=200): (8676.72, 9813.74)
Age group 26-35 average purchases at 99% confidence interval range(sample_size=200): (8441.17, 10049.29
----------------------------------------------------------------------------------------------------
Age group 26-35 average purchases at 90% confidence interval range(sample_size=2000): (8802.29, 9688.17
Age group 26-35 average purchases at 95% confidence interval range(sample_size=2000): (9066.11, 9421.27
Age group 26-35 average purchases at 99% confidence interval range(sample_size=2000): (8992.54, 9494.84
----------------------------------------------------------------------------------------------------
Age group 26-35 average purchases at 90% confidence interval range(sample_size=10000): (9183.12, 9303.8
Age group 26-35 average purchases at 95% confidence interval range(sample_size=10000): (9166.01, 9320.9
Age group 26-35 average purchases at 99% confidence interval range(sample_size=10000): (9133.91, 9353.6
----------------------------------------------------------------------------------------------------
```

```
# Age group = 36-45

#sample size = 200
sample_size_200 = 200
sample_mean_200 = populate_sample_mean(age_36_45,sample_size_200)

#sample size = 2000
sample_size_2000 = 2000
sample_mean_2000 = populate_sample_mean(age_36_45,sample_size_2000)

#sample size = 10000
sample_size_10000 = 10000
sample_mean_10000 = populate_sample_mean(age_36_45,sample_size_10000)

#Means & std of all 30000 samples means and population(group-(36-45)) std.
sample_mean_mean_200,sample_std_200,population_std_200 = calculate_sample_mean_std(sample_mean_200)
sample_mean_mean_2000,sample_std_2000,population_std_2000 = calculate_sample_mean_std(sample_mean_2000)
sample_mean_mean_10000,sample_std_10000,population_std_10000 = calculate_sample_mean_std(sample_mean_10000)

# print all the data
print_sample_mean_std(sample_mean_mean_200,sample_std_200,population_std_200,sample_size_200)
print_sample_mean_std(sample_mean_mean_2000,sample_std_2000,population_std_2000,sample_size_2000)
print_sample_mean_std(sample_mean_mean_10000,sample_std_10000,population_std_10000,sample_size_10000)
```

```
sample_means_200: 9322.15
sample_std_200(std error): 341.98
population_std_200: 4836.33
----------------------------------------------------------------------------------------------------
sample_means_2000: 9322.84
sample_std_2000(std error): 107.53
population_std_2000: 1520.7
----------------------------------------------------------------------------------------------------
sample_means_10000: 9323.31
sample_std_10000(std error): 46.49
population_std_10000: 657.47
----------------------------------------------------------------------------------------------------
```

```
plt.figure(figsize=(8,6))
sns.kdeplot(sample_mean_200,color='red',label='sample_size_200')
sns.kdeplot(sample_mean_2000,color='blue',label='sample_size_2000')
sns.kdeplot(sample_mean_10000,color='green',label='sample_size_10000')
plt.xlabel('Purchage amount')
plt.ylabel('Density')
plt.title('Unmarried purchages')
plt.legend(["sample_size: 200","sample_size: 2000","sample_size: 10000"])
plt.show()
```



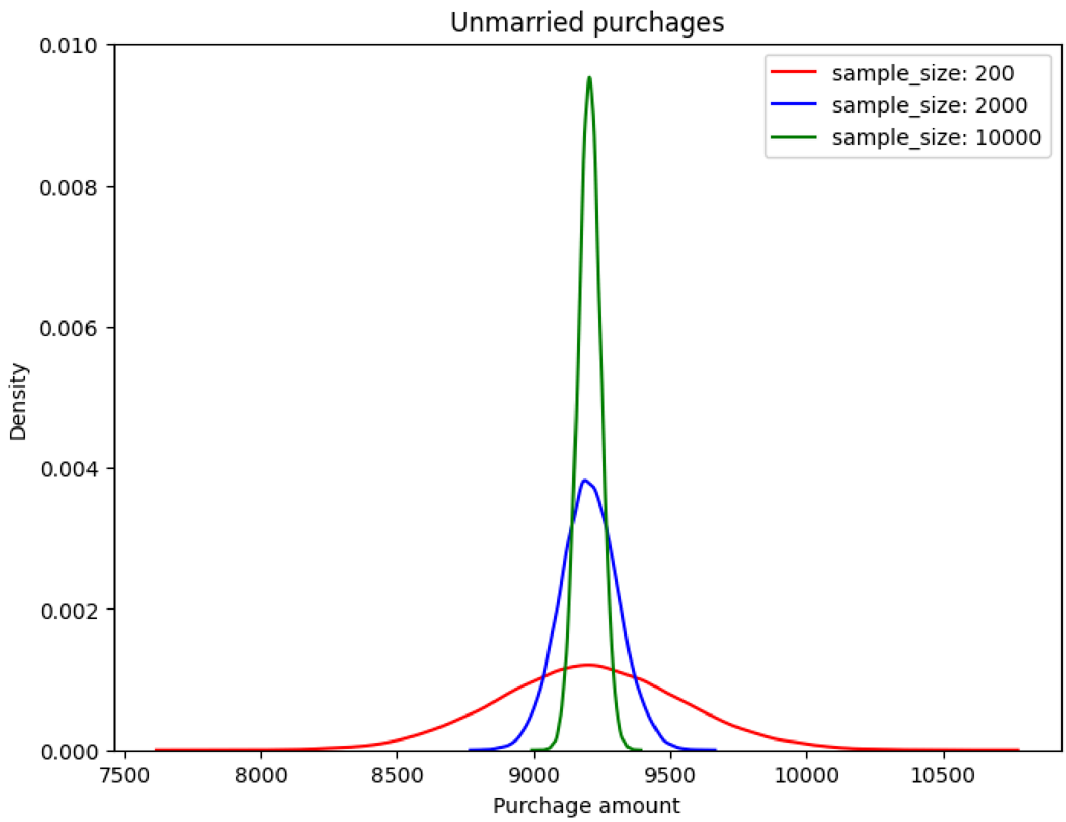

```
# Age group 36-45
age_group='36-45'

# caluculating confidence intervals ranges with sample size 200.
interval_age_36_45_90,interval_age_36_45_95,interval_age_36_45_99 = confidence_interval(sample_mean_mean_20
# Printing the data
print_confidence_interval(interval_age_36_45_90,interval_age_36_45_95,interval_age_0_17_99,sample_size_200,

#calculating confidence intervals ranges with sample size 2000
innterval_age_36_45_90,interval_age_36_45_95,interval_age_36_45_99 = confidence_interval(sample_mean_mean_2
# Printing the data
print_confidence_interval(interval_age_36_45_90,interval_age_36_45_95,interval_age_36_45_99,sample_size_200

#calculating confidence intervals ranges with sample size 10000
interval_age_36_45_90,interval_age_36_45_95,interval_age_36_45_99 = confidence_interval(sample_mean_mean_10
# Printing the data
print_confidence_interval(interval_age_36_45_90,interval_age_36_45_95,interval_age_36_45_99,sample_size_100
```

```
Age group 36-45 average purchases at 90% confidence interval range(sample_size=200): (8883.88, 9760.42)
Age group 36-45 average purchases at 95% confidence interval range(sample_size=200): (8759.64, 9884.66)
Age group 36-45 average purchases at 99% confidence interval range(sample_size=200): (9133.91, 9353.05)
---------------------------------------------------------------------------------------------------
Age group 36-45 average purchases at 90% confidence interval range(sample_size=2000): (8883.88, 9760.42
Age group 36-45 average purchases at 95% confidence interval range(sample_size=2000): (9145.97, 9499.71
Age group 36-45 average purchases at 99% confidence interval range(sample_size=2000): (9072.69, 9572.9
---------------------------------------------------------------------------------------------------
Age group 36-45 average purchases at 90% confidence interval range(sample_size=10000): (9263.73, 9382.8
Age group 36-45 average purchases at 95% confidence interval range(sample_size=10000): (9246.84, 9399.7
Age group 36-45 average purchases at 99% confidence interval range(sample_size=10000): (9215.16, 9431.4
---------------------------------------------------------------------------------------------------
```

```
# Age group = 46-50

#sample size = 200
sample_size_200 = 200
sample_mean_200 = populate_sample_mean(age_46_50,sample_size_200)

#sample size = 2000
sample_size_2000 = 2000
sample_mean_2000 = populate_sample_mean(age_46_50,sample_size_2000)

#sample size = 10000
sample_size_10000 = 10000
sample_mean_10000 = populate_sample_mean(age_46_50,sample_size_10000)

#Means & std of all 30000 samples means and population(group-(46-50)) std.
sample_mean_mean_200,sample_std_200,population_std_200 = calculate_sample_mean_std(sample_mean_200)
sample_mean_mean_2000,sample_std_2000,population_std_2000 = calculate_sample_mean_std(sample_mean_2000)
sample_mean_mean_10000,sample_std_10000,population_std_10000 = calculate_sample_mean_std(sample_mean_10000)

# print all the data
print_sample_mean_std(sample_mean_mean_200,sample_std_200,population_std_200,sample_size_200)
print_sample_mean_std(sample_mean_mean_2000,sample_std_2000,population_std_2000,sample_size_2000)
print_sample_mean_std(sample_mean_mean_10000,sample_std_10000,population_std_10000,sample_size_10000)
```

```
sample_means_200: 9204.94
sample_std_200(std error): 336.84
population_std_200: 4763.64
----------------------------------------------------------------------------------------------------
sample_means_2000: 9204.01
sample_std_2000(std error): 104.7
population_std_2000: 1480.68
----------------------------------------------------------------------------------------------------
sample_means_10000: 9204.15
sample_std_10000(std error): 42.47
population_std_10000: 600.62
----------------------------------------------------------------------------------------------------
```

```
plt.figure(figsize=(8,6))
sns.kdeplot(sample_mean_200,color='red',label='sample_size_200')
sns.kdeplot(sample_mean_2000,color='blue',label='sample_size_2000')
sns.kdeplot(sample_mean_10000,color='green',label='sample_size_10000')
plt.xlabel('Purchage amount')
plt.ylabel('Density')
plt.title('Unmarried purchages')
plt.legend(["sample_size: 200","sample_size: 2000","sample_size: 10000"])
plt.show()
```



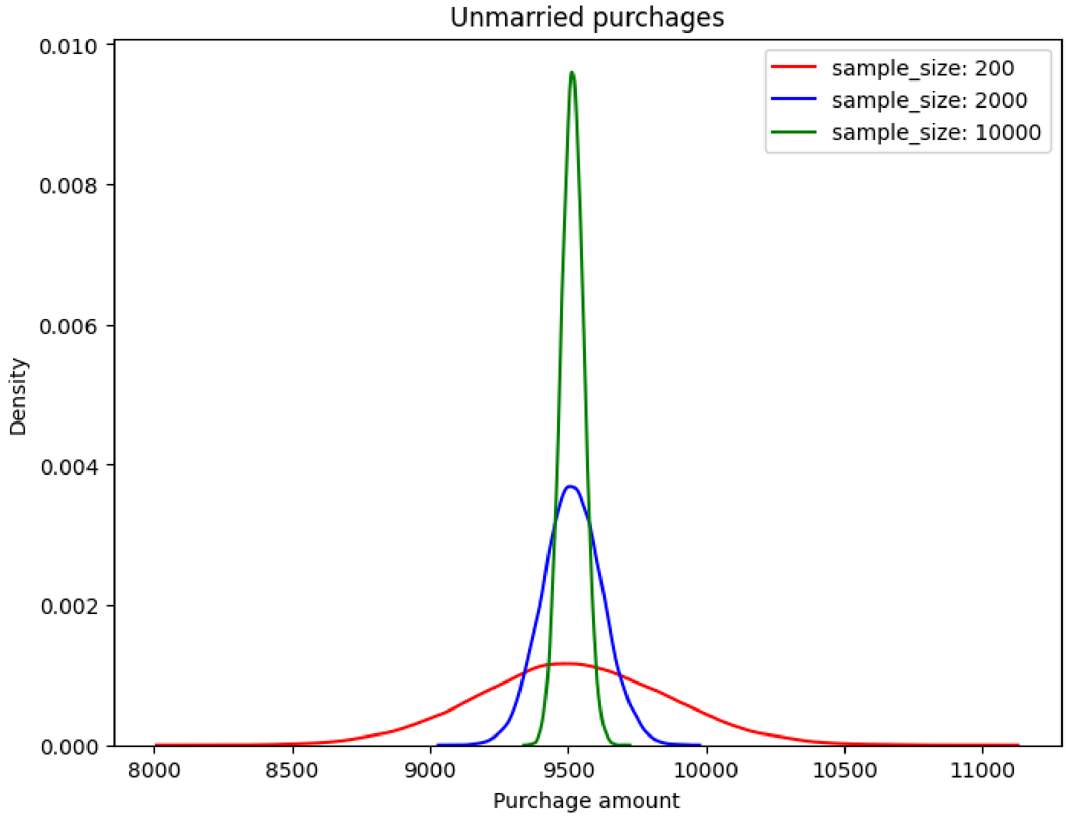

```
# Age group 46-50
age_group='46-50'

# caluculating confidence intervals ranges with sample size 200.
interval_age_46_50_90,interval_age_46_50_95,interval_age_46_50_99 = confidence_interval(sample_mean_mean_20
# Printing the data
print_confidence_interval(interval_age_46_50_90,interval_age_46_50_95,interval_age_46_50_99,sample_size_200

#calculating confidence intervals ranges with sample size 2000
innterval_age_46_50_90,interval_age_46_50_95,interval_age_46_50_99 = confidence_interval(sample_mean_mean_2
# Printing the data
print_confidence_interval(interval_age_46_50_90,interval_age_46_50_95,interval_age_46_50_99,sample_size_200

#calculating confidence intervals ranges with sample size 10000
interval_age_46_50_90,interval_age_46_50_95,interval_age_46_50_99 = confidence_interval(sample_mean_mean_10
# Printing the data
print_confidence_interval(interval_age_46_50_90,interval_age_46_50_95,interval_age_46_50_99,sample_size_100
```

```
Age group 46-50 average purchases at 90% confidence interval range(sample_size=200): (8773.26, 9636.62)
Age group 46-50 average purchases at 95% confidence interval range(sample_size=200): (8650.89, 9758.99)
Age group 46-50 average purchases at 99% confidence interval range(sample_size=200): (8421.33, 9988.55)
----------------------------------------------------------------------------------------------------
Age group 46-50 average purchases at 90% confidence interval range(sample_size=2000): (8773.26, 9636.62
Age group 46-50 average purchases at 95% confidence interval range(sample_size=2000): (9031.79, 9376.23
Age group 46-50 average purchases at 99% confidence interval range(sample_size=2000): (8960.44, 9447.58
----------------------------------------------------------------------------------------------------
Age group 46-50 average purchases at 90% confidence interval range(sample_size=10000): (9149.72, 9258.5
Age group 46-50 average purchases at 95% confidence interval range(sample_size=10000): (9134.29, 9274.6
Age group 46-50 average purchases at 99% confidence interval range(sample_size=10000): (9105.35, 9302.9
----------------------------------------------------------------------------------------------------
```

```
# Age group = 51-55

#sample size = 200
sample_size_200 = 200
sample_mean_200 = populate_sample_mean(age_51_55,sample_size_200)

#sample size = 2000
sample_size_2000 = 2000
sample_mean_2000 = populate_sample_mean(age_51_55,sample_size_2000)

#sample size = 10000
sample_size_10000 = 10000
sample_mean_10000 = populate_sample_mean(age_51_55,sample_size_10000)

#Means & std of all 30000 samples means and population(group-(51_55)) std.
sample_mean_mean_200,sample_std_200,population_std_200 = calculate_sample_mean_std(sample_mean_200)
sample_mean_mean_2000,sample_std_2000,population_std_2000 = calculate_sample_mean_std(sample_mean_2000)
sample_mean_mean_10000,sample_std_10000,population_std_10000 = calculate_sample_mean_std(sample_mean_10000)

# print all the data
print_sample_mean_std(sample_mean_mean_200,sample_std_200,population_std_200,sample_size_200)
print_sample_mean_std(sample_mean_mean_2000,sample_std_2000,population_std_2000,sample_size_2000)
print_sample_mean_std(sample_mean_mean_10000,sample_std_10000,population_std_10000,sample_size_10000)
```

```
sample_means_200: 9515.96
sample_std_200(std error): 343.65
population_std_200: 4859.94
----------------------------------------------------------------------------------------------------
sample_means_2000: 9515.76
sample_std_2000(std error): 106.63
population_std_2000: 1507.98
----------------------------------------------------------------------------------------------------
sample_means_10000: 9515.22
sample_std_10000(std error): 41.66
population_std_10000: 589.16
----------------------------------------------------------------------------------------------------
```

```
plt.figure(figsize=(8,6))
sns.kdeplot(sample_mean_200,color='red',label='sample_size_200')
sns.kdeplot(sample_mean_2000,color='blue',label='sample_size_2000')
sns.kdeplot(sample_mean_10000,color='green',label='sample_size_10000')
plt.xlabel('Purchage amount')
plt.ylabel('Density')
plt.title('Unmarried purchages')
plt.legend(["sample_size: 200","sample_size: 2000","sample_size: 10000"])
plt.show()
```



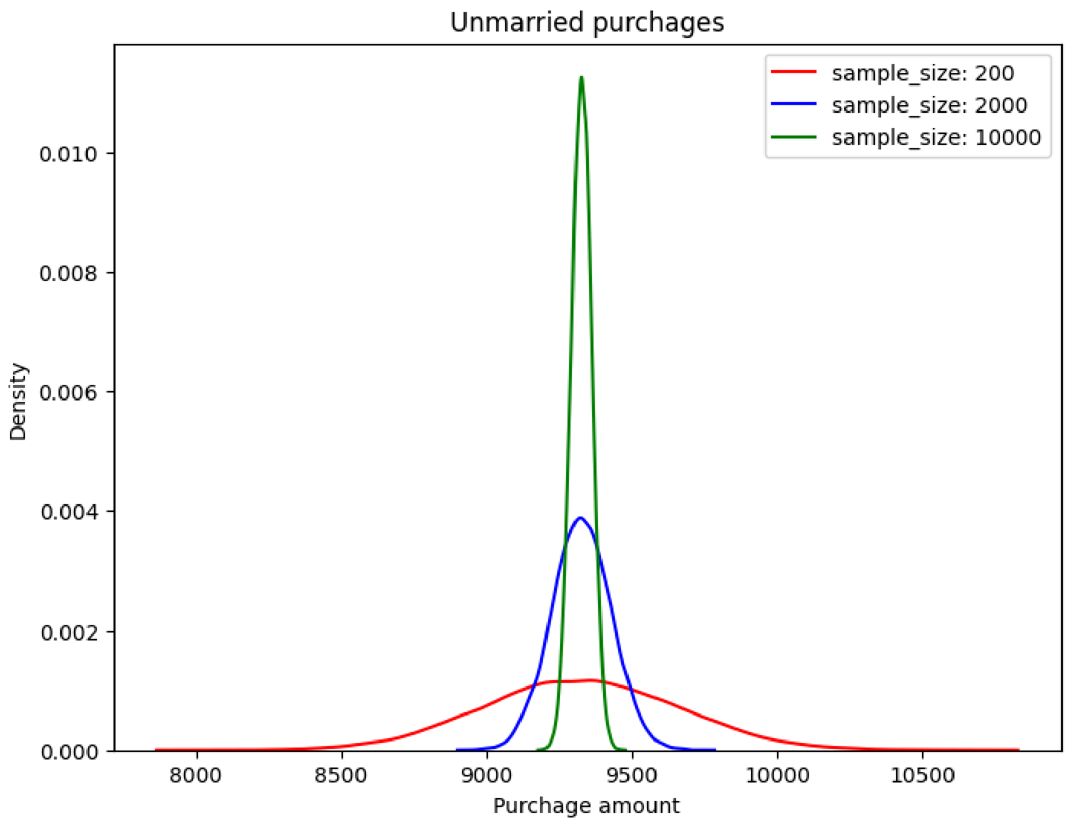

```
# Age group 51-55
age_group='51-55'


# caluculating confidence intervals ranges with sample size 200.
interval_age_51_55_90,interval_age_51_55_95,interval_age_51_55_99 = confidence_interval(sample_mean_mean_20
# Printing the data
print_confidence_interval(interval_age_51_55_90,interval_age_51_55_95,interval_age_51_55_99,sample_size_200

#calculating confidence intervals ranges with sample size 2000
interval_age_51_55_90,interval_age_51_55_95,interval_age_51_55_99 = confidence_interval(sample_mean_mean_20
# Printing the data
print_confidence_interval(interval_age_51_55_90,interval_age_51_55_95,interval_age_51_55_99,sample_size_200

#calculating confidence intervals ranges with sample size 10000
interval_age_51_55_90,interval_age_51_55_95,interval_age_51_55_99 = confidence_interval(sample_mean_mean_10
# Printing the data
print_confidence_interval(interval_age_51_55_90,interval_age_51_55_95,interval_age_51_55_99,sample_size_100
```

```
Age group 51-55 average purchases at 90% confidence interval range(sample_size=200): (9075.55, 9956.37)
Age group 51-55 average purchases at 95% confidence interval range(sample_size=200): (8950.71, 10081.21
Age group 51-55 average purchases at 99% confidence interval range(sample_size=200): (8716.51, 10315.41
----------------------------------------------------------------------------------------------------
Age group 51-55 average purchases at 90% confidence interval range(sample_size=2000): (9379.11, 9652.41
Age group 51-55 average purchases at 95% confidence interval range(sample_size=2000): (9340.37, 9691.15
Age group 51-55 average purchases at 99% confidence interval range(sample_size=2000): (9267.7, 9763.82)
----------------------------------------------------------------------------------------------------
Age group 51-55 average purchases at 90% confidence interval range(sample_size=10000): (9461.83, 9568.6
Age group 51-55 average purchases at 95% confidence interval range(sample_size=10000): (9446.7, 9583.74
Age group 51-55 average purchases at 99% confidence interval range(sample_size=10000): (9418.3, 9612.14
----------------------------------------------------------------------------------------------------
```

```
# Age group = Above 55

#sample size = 200
sample_size_200 = 200
sample_mean_200 = populate_sample_mean(age_55_above,sample_size_200)

#sample size = 2000
sample_size_2000 = 2000
sample_mean_2000 = populate_sample_mean(age_55_above,sample_size_2000)

#sample size = 10000
sample_size_10000 = 10000
sample_mean_10000 = populate_sample_mean(age_55_above,sample_size_10000)

#Means & std of all 30000 samples means and population(group-(55_above)) std.
sample_mean_mean_200,sample_std_200,population_std_200 = calculate_sample_mean_std(sample_mean_200)
sample_mean_mean_2000,sample_std_2000,population_std_2000 = calculate_sample_mean_std(sample_mean_2000)
sample_mean_mean_10000,sample_std_10000,population_std_10000 = calculate_sample_mean_std(sample_mean_10000)

# print all the data
print_sample_mean_std(sample_mean_mean_200,sample_std_200,population_std_200,sample_size_200)
print_sample_mean_std(sample_mean_mean_2000,sample_std_2000,population_std_2000,sample_size_2000)
print_sample_mean_std(sample_mean_mean_10000,sample_std_10000,population_std_10000,sample_size_10000)
```

```
sample_means_200: 9325.53
sample_std_200(std error): 335.28
population_std_200: 4741.58
----------------------------------------------------------------------------------------------------
sample_means_2000: 9327.45
sample_std_2000(std error): 101.7
population_std_2000: 1438.26
----------------------------------------------------------------------------------------------------
sample_means_10000: 9327.49
sample_std_10000(std error): 35.0
population_std_10000: 494.97
----------------------------------------------------------------------------------------------------
```

```
plt.figure(figsize=(8,6))
sns.kdeplot(sample_mean_200,color='red',label='sample_size_200')
sns.kdeplot(sample_mean_2000,color='blue',label='sample_size_2000')
sns.kdeplot(sample_mean_10000,color='green',label='sample_size_10000')
plt.xlabel('Purchage amount')
plt.ylabel('Density')
plt.title('Unmarried purchages')
plt.legend(["sample_size: 200","sample_size: 2000","sample_size: 10000"])
plt.show()
```




```
# Age group above 55
age_group='55+'


# caluculating confidence intervals ranges with sample size 200.
interval_age_above_55_90,interval_age_above_55_95,interval_age_above_55_99 = confidence_interval(sample_mea
# Printing the data
print_confidence_interval(interval_age_above_55_90,interval_age_above_55_95,interval_age_51_55_99,sample_si

#calculating confidence intervals ranges with sample size 2000
interval_age_above_55_90,interval_age_above_55_95,interval_age_above_55_99 = confidence_interval(sample_mea
# Printing the data
print_confidence_interval(interval_age_above_55_90,interval_age_above_55_95,interval_age_above_55_99,sample

#calculating confidence intervals ranges with sample size 10000
interval_age_above_55_90,interval_age_above_55_95,interval_age_above_55_99 = confidence_interval(sample_mea
# Printing the data
print_confidence_interval(interval_age_above_55_90,interval_age_above_55_95,interval_age_above_55_99,sample
```

```
Age group 55+ average purchases at 90% confidence interval range(sample_size=200): (8895.85, 9755.21)
Age group 55+ average purchases at 95% confidence interval range(sample_size=200): (8774.04, 9877.02)
Age group 55+ average purchases at 99% confidence interval range(sample_size=200): (9418.3, 9612.14)
----------------------------------------------------------------------------------------------------
Age group 55+ average purchases at 90% confidence interval range(sample_size=2000): (9197.12, 9457.78)
Age group 55+ average purchases at 95% confidence interval range(sample_size=2000): (9160.17, 9494.73)
Age group 55+ average purchases at 99% confidence interval range(sample_size=2000): (9090.86, 9564.04)
----------------------------------------------------------------------------------------------------
Age group 55+ average purchases at 90% confidence interval range(sample_size=10000): (9282.64, 9372.34)
Age group 55+ average purchases at 95% confidence interval range(sample_size=10000): (9269.92, 9385.06)
Age group 55+ average purchases at 99% confidence interval range(sample_size=10000): (9246.07, 9408.91)
----------------------------------------------------------------------------------------------------
```

Here we are checking purchage for different age group people. So we can use ANOVA test to verify average purchage amount of all groups are same or different.

```
# One way anova test

#Null hypothesis H0: Average purchase among all the age groups are same
# Alternative Hypothesis Ha: Average purchase among all the age groups are different


f_statastics,p_value=f_oneway(age_0_17['Purchase'],age_18_25['Purchase'],age_26_35['Purchase'],age_36_45['P

# At 95% confidence level
alpha=0.05
Hypothesis_test(z_statastics,p_value,alpha)

# At 90% confidence interval
alpha=0.1
Hypothesis_test(z_statastics,p_value,alpha)

# At 99% confidence interval
alpha=0.01
Hypothesis_test(z_statastics,p_value,alpha)
```

```
statastics: -0.3868627106847199
p_value: 2.111079894476655e-48
Reject Null Hypothesis at 95% confidence
Average married people spent amount is different than Average unmarried people spent amount

statastics: -0.3868627106847199
p_value: 2.111079894476655e-48
Reject Null Hypothesis at 95% confidence
Average married people spent amount is different than Average unmarried people spent amount

statastics: -0.3868627106847199
p_value: 2.111079894476655e-48
Reject Null Hypothesis at 95% confidence
Average married people spent amount is different than Average unmarried people spent amount
```

Insights:

From the CLT, collected 30000 samples with sample sizes 200, 2000 and 10000. we could see the confidence interval ranges at 90%, 95% and 99%.

Age group 0-17:

- sample mean purchase amount with sample size 200 = 8941.41
- sample mean purchase amount with sample size 2000 = 8940.96
- sample mean purchase amount with sample size 10000 = 8940.63

- sample standard deviation(std error) with sample size 200 = 345.09
- sample standard deviation(std error) with sample size 2000 = 103.39
- sample standard deviation(std error) with sample size 20000 = 28.62

- Age group 0-17 population standard deviation with sample size 200 = 4880.31
- Age group 0-17 population standard deviation with sample size 2000 = 1462.16
- Age group 0-17 population standard deviation with sample size 20000 = 404.75

Age group 0-17 average purchases at 90% confidence interval range(sample_size=200): (9075.55, 9956.37) Age group 0-17 average purchases at 95% confidence interval range(sample_size=200): (8950.71, 10081.21) Age group 0-17 average purchases at 99% confidence interval range(sample_size=200): (8716.51, 10315.41)

Age group 0-17 average purchases at 90% confidence interval range(sample_size=2000): (9075.55, 9956.37) Age group 0-17 average purchases at 95% confidence interval range(sample_size=2000): (9340.37, 9691.15) Age group 0-17 average purchases at 99% confidence interval range(sample_size=2000): (9267.7, 9763.82)

Age group 0-17 average purchases at 90% confidence interval range(sample_size=10000): (9461.83, 9568.61) Age group 0-17 average purchases at 95% confidence interval range(sample_size=10000): (9446.7, 9583.74) Age group 0-17 average purchases at 99% confidence interval range(sample_size=10000): (9418.3, 9612.14)

Age group 18-25:

- sample mean purchase amount with sample size 200 = 9169.82
- sample mean purchase amount with sample size 2000 = 9168.46
- sample mean purchase amount with sample size 10000 = 9169.08

- sample standard deviation(std error) with sample size 200 = 345.09
- sample standard deviation(std error) with sample size 2000 = 103.39
- sample standard deviation(std error) with sample size 20000 = 28.62

- Age group 0-17 population standard deviation with sample size 200 = 4880.31
- Age group 0-17 population standard deviation with sample size 2000 = 1462.16
- Age group 0-17 population standard deviation with sample size 20000 = 404.75

- Age group 18-25 average purchases at 90% confidence interval range(sample_size=200): (8727.7, 9611.94)
- Age group 18-25 average purchases at 95% confidence interval range(sample_size=200): (8602.36, 9737.28)
- Age group 18-25 average purchases at 99% confidence interval range(sample_size=200): (8367.25, 9972.39)

- Age group 18-25 average purchases at 90% confidence interval range(sample_size=2000): (8727.7, 9611.94)
- Age group 18-25 average purchases at 95% confidence interval range(sample_size=2000): (8991.31, 9345.61)
- Age group 18-25 average purchases at 99% confidence interval range(sample_size=2000): (8917.91, 9419.01)

- Age group 18-25 average purchases at 90% confidence interval range(sample_size=10000): (9110.19, 9227.97)
- Age group 18-25 average purchases at 95% confidence interval range(sample_size=10000): (9093.5, 9244.66)
- Age group 18-25 average purchases at 99% confidence interval range(sample_size=10000): (9062.18, 9275.98)

Age group 26-35:

- sample_means_200: 9245.23
- sample_std_200(std error): 345.63
- population_std_200: 4887.95

- sample_means_2000: 9243.69
- sample_std_2000(std error): 107.96
- population_std_2000: 1526.78

- sample_means_10000: 9243.48
- sample_std_10000(std error): 47.1
- population_std_10000: 666.09

- Age group 26-35 average purchases at 90% confidence interval range(sample_size=200): (8802.29, 9688.17)
- Age group 26-35 average purchases at 95% confidence interval range(sample_size=200): (8676.72, 9813.74)
- Age group 26-35 average purchases at 99% confidence interval range(sample_size=200): (8441.17, 10049.29)

- Age group 26-35 average purchases at 90% confidence interval range(sample_size=2000): (8802.29, 9688.17)
- Age group 26-35 average purchases at 95% confidence interval range(sample_size=2000): (9066.11, 9421.27)
- Age group 26-35 average purchases at 99% confidence interval range(sample_size=2000): (8992.54, 9494.84)

- Age group 26-35 average purchases at 90% confidence interval range(sample_size=10000): (9183.12, 9303.84)
- Age group 26-35 average purchases at 95% confidence interval range(sample_size=10000): (9166.01, 9320.95)
- Age group 26-35 average purchases at 99% confidence interval range(sample_size=10000): (9133.91, 9353.05)

Age group 36-45:

- sample_means_200: 9322.15
- sample_std_200(std error): 341.98
- population_std_200: 4836.33

- sample_means_2000: 9322.84
- sample_std_2000(std error): 107.53
- population_std_2000: 1520.7

---

sample_means_10000: 9323.31 sample_std_10000(std error): 46.49 population_std_10000: 657.47

- Age group 36-45 average purchases at 90% confidence interval range(sample_size=200): (8883.88, 9760.42)
- Age group 36-45 average purchases at 95% confidence interval range(sample_size=200): (8759.64, 9884.66)
- Age group 36-45 average purchases at 99% confidence interval range(sample_size=200): (9133.91, 9353.05)

- Age group 36-45 average purchases at 90% confidence interval range(sample_size=2000): (8883.88, 9760.42)
- Age group 36-45 average purchases at 95% confidence interval range(sample_size=2000): (9145.97, 9499.71)
- Age group 36-45 average purchases at 99% confidence interval range(sample_size=2000): (9072.69, 9572.99)

- Age group 36-45 average purchases at 90% confidence interval range(sample_size=10000): (9263.73, 9382.89)
- Age group 36-45 average purchases at 95% confidence interval range(sample_size=10000): (9246.84, 9399.78)
- Age group 36-45 average purchases at 99% confidence interval range(sample_size=10000): (9215.16, 9431.46)

Age group 46-50:

- sample_means_200: 9204.94
- sample_std_200(std error): 336.84
- population_std_200: 4763.64

- sample_means_2000: 9204.01
- sample_std_2000(std error): 104.7
- population_std_2000: 1480.68

- sample_means_10000: 9204.15
- sample_std_10000(std error): 42.47
- population_std_10000: 600.62

- Age group 46-50 average purchases at 90% confidence interval range(sample_size=200): (8773.26, 9636.62)
- Age group 46-50 average purchases at 95% confidence interval range(sample_size=200): (8650.89, 9758.99)

- Age group 46-50 average purchases at 99% confidence interval range(sample_size=200): (8421.33, 9988.55)

- Age group 46-50 average purchases at 90% confidence interval range(sample_size=2000): (8773.26, 9636.62)
- Age group 46-50 average purchases at 95% confidence interval range(sample_size=2000): (9031.79, 9376.23)
- Age group 46-50 average purchases at 99% confidence interval range(sample_size=2000): (8960.44, 9447.58)

- Age group 46-50 average purchases at 90% confidence interval range(sample_size=10000): (9149.72, 9258.58)
- Age group 46-50 average purchases at 95% confidence interval range(sample_size=10000): (9134.29, 9274.01)
- Age group 46-50 average purchases at 99% confidence interval range(sample_size=10000): (9105.35, 9302.95)

Age group 51-55:

- sample_means_200: 9515.96
- sample_std_200(std error): 343.65
- population_std_200: 4859.94

- sample_means_2000: 9515.76
- sample_std_2000(std error): 106.63
- population_std_2000: 1507.98

- sample_means_10000: 9515.22
- sample_std_10000(std error): 41.66
- population_std_10000: 589.16

- Age group 51-55 average purchases at 90% confidence interval range(sample_size=200): (9075.55, 9956.37)
- Age group 51-55 average purchases at 95% confidence interval range(sample_size=200): (8950.71, 10081.21)
- Age group 51-55 average purchases at 99% confidence interval range(sample_size=200): (8716.51, 10315.41)

- Age group 51-55 average purchases at 90% confidence interval range(sample_size=2000): (9379.11, 9652.41)
- Age group 51-55 average purchases at 95% confidence interval range(sample_size=2000): (9340.37, 9691.15)
- Age group 51-55 average purchases at 99% confidence interval range(sample_size=2000): (9267.7, 9763.82)

- Age group 51-55 average purchases at 90% confidence interval range(sample_size=10000): (9461.83, 9568.61)
- Age group 51-55 average purchases at 95% confidence interval range(sample_size=10000): (9446.7, 9583.74)
- Age group 51-55 average purchases at 99% confidence interval range(sample_size=10000): (9418.3, 9612.14)

Age group above 55:

- sample_means_200: 9325.53
- sample_std_200(std error): 335.28
- population_std_200: 4741.58

- sample_means_2000: 9327.45
- sample_std_2000(std error): 101.7
- population_std_2000: 1438.26

- sample_means_10000: 9327.49
- sample_std_10000(std error): 35.0
- population_std_10000: 494.97

- Age group 55+ average purchases at 90% confidence interval range(sample_size=200): (8895.85, 9755.21)

- Age group 55+ average purchases at 95% confidence interval range(sample_size=200): (8774.04, 9877.02)
- Age group 55+ average purchases at 99% confidence interval range(sample_size=200): (9418.3, 9612.14)

- Age group 55+ average purchases at 90% confidence interval range(sample_size=2000): (9197.12, 9457.78)
- Age group 55+ average purchases at 95% confidence interval range(sample_size=2000): (9160.17, 9494.73)
- Age group 55+ average purchases at 99% confidence interval range(sample_size=2000): (9090.86, 9564.04)

- Age group 55+ average purchases at 90% confidence interval range(sample_size=10000): (9282.64, 9372.34)
- Age group 55+ average purchases at 95% confidence interval range(sample_size=10000): (9269.92, 9385.06)
- Age group 55+ average purchases at 99% confidence interval range(sample_size=10000): (9246.07, 9408.91)

Conclusion:

we can clearly see that the there is a significatnt diffrence between different age group people. Verified same with one way ANOVA test.

```
walmart_data['Age'].value_counts()
```

Age	count
26-35	219587
36-45	110013
18-25	99660
46-50	45701
51-55	38501
55+	21504
0-17	15102

dtype: int64

```
age_55_above
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_
4	1000002	P00285442	M	55+	16	C	
159	1000031	P00117442	M	55+	7	C	
160	1000031	P00322042	M	55+	7	C	
161	1000031	P00216342	M	55+	7	C	
162	1000031	P00329342	M	55+	7	C	
...	...	...	...	...	...	...	
549925	1005834	P00371644	M	55+	16	C	
549989	1005922	P00370853	M	55+	3	C	
550008	1005946	P00370853	F	55+	1	A	
550030	1005980	P00372445	M	55+	1	C	
550066	1006038	P00375436	F	55+	1	C	

21504 rows × 10 columns

Recomendations:

- Adjust Product Offerings Age-Specific Products: If the analysis shows that different age groups have significantly different spending patterns, Walmart should consider customizing its product assortment to cater to these preferences. For example, younger age groups may prefer trendy, budget-friendly items, while older customers might prioritize quality and premium products. Localized Inventory Management: Based on the demographic composition of different store locations, Walmart can optimize its inventory to meet the specific needs of the predominant age group, gender, or marital status in each area. This approach could reduce inventory costs and increase customer satisfaction by ensuring that stores stock the products most desired by their local customer base.
- Data-Driven Decision Making Continuous Monitoring and Analysis: Walmart should continue to monitor spending patterns across different demographics and adjust its strategies as needed. Regularly analyzing confidence intervals for spending data will help Walmart stay responsive to shifts in consumer behavior, allowing the company to adapt quickly to changing market conditions. Leverage Technology: Utilize advanced data analytics tools and machine learning algorithms to predict future spending trends based on demographic data. This proactive approach can help Walmart stay ahead of the competition by anticipating customer needs and adjusting its offerings in real-time.
- Walmart could develop gender-specific marketing campaigns or product lines to better appeal to the distinct spending patterns of each group. For example, targeted promotions on products that are more popular among one gender could enhance sales. Marital Status Targeting: If there is a significant difference in spending between married and unmarried individuals, Walmart can tailor its marketing and promotions accordingly.
- Customized Promotions: Develop and distribute promotional offers that are specifically designed to appeal to different demographic groups. This could involve offering discounts on family-oriented products for married customers or promoting new technology and fashion items to younger age groups. By implementing these recommendations, Walmart can better align its marketing, product offerings, and customer engagement strategies with the spending habits and preferences of its diverse customer base, ultimately driving increased sales and customer loyalty.

--END---

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit