The Problem Statement

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.

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).

```
# from google.colab import drive
# drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Importing Libraries

+ Code + Text

```
import pandas as pd
import seaborn as sns
import numpy as np
from scipy.stats import norm, sem


np.random.seed(10)
```

Importing the Data

```
df = pd.read_csv("/content/drive/MyDrive/Scaler_Assignments/walmart_data.csv")
df.head(100)
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Ca
0	1000001	P00069042	F	0-17	10	A	2	0	

Data Exploration

2	1000001	P00087842	F	0-	10	A	2	0	

```
df.info()
```

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

We can observe here that there are NO NULL Values here. So No Null Values treatement is necessary in this case.

```
df.describe()
```

	User_ID	Occupation	Marital_Status	Product_Category	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9263.968713
std	1.727592e+03	6.522660	0.491770	3.936211	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	12.000000
25%	1.001516e+06	2.000000	0.000000	1.000000	5823.000000
50%	1.003077e+06	7.000000	0.000000	5.000000	8047.000000
75%	1.004478e+06	14.000000	1.000000	8.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	23961.000000

```
df.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Cat
0	1000001	P00069042	F	0-17	10	A	2	0	

```
# Global Function Definitions.

sample_size = 500
iterations = 1000
conf_int = 0.90


def get_sampled_arr(iters,array, size):
  means = []
  for _ in range(iters):
    sample_mean = np.mean(array.sample(size, replace=True))
    means.append(sample_mean)
  return means

def Cal_CI(conf_int, data):
  mean = np.round(np.mean(data),4)
  std_err = np.round(np.std(data),4)
  z = np.round(norm.ppf(1-conf_int/2),4)
  margin_err = z*std_err
  lower_bound = mean - margin_err
  upper_bound = mean+margin_err
  return mean, lower_bound, upper_bound


def get_CI_Stats(fltr,i,n,conf_int):
  '''
  For Calculation of CI Metrices for DIfferent Age Groups.
  fltr = The filter value for the Age Range.
  i = No of Iterations
  n = Sample Size for BootStrapping.
  conf_int = Confidence Interval

  Returns - Population Mean, Random Sample Range,Random Sample Mean, X1 and X2

  '''
  df1 = df[df["Age"]==fltr].drop(columns="Age")
  smp_arr = get_sampled_arr(i,df1["Purchase"],n)
  smp_mean, lower_bnd, uppr_bnd = Cal_CI(conf_int,smp_arr)

  return smp_mean, smp_arr, lower_bnd,uppr_bnd


df_female = df[df["Gender"]=="F"].drop(columns="Gender")
df_female.head()
```

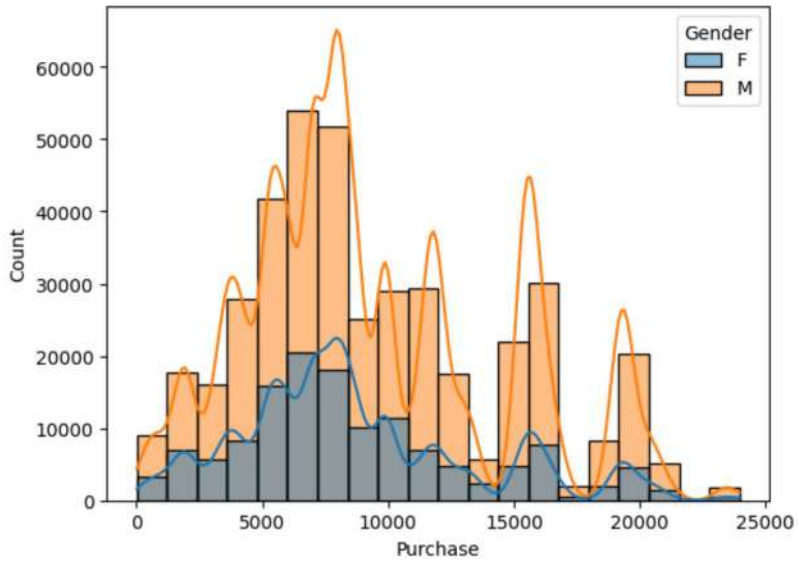
	User_ID	Product_ID	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	0-17	10	A	2	0	3	8370
1	1000001	P00248942	0-17	10	A	2	0	1	15200
2	1000001	P00087842	0-17	10	A	2	0	12	1422

```
df_male = df[df["Gender"]=="M"].drop(columns="Gender")
df_male.head()
```

	User_ID	Product_ID	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
4	1000002	P00285442	55+	16	C	4+	0	8	7969
5	1000003	P00193542	26-35	15	A	3	0	1	15227
6	1000004	P00184942	46-50	7	B	2	1	1	19215
7	1000004	P00346142	46-50	7	B	2	1	1	15854
8	1000004	P0097242	46-50	7	B	2	1	1	15686

```
sns.histplot(data=df,x="Purchase", bins = 20, kde = True, hue = "Gender")
```

```
<Axes: xlabel='Purchase', ylabel='Count'>
```



Looking at this graph itself, we may get a feel of the data, Gender-wise.

- The Data is not normally distributed.
- The Purchase data it skewed towards the right.
- Average Expenditure for Males seem to be higher than females.

```
print(f'The Population mean for Female expenses is {np.round(df_female["Purchase"].mean(),4)}')
```

```
The Population mean for Female expenses is 8734.5658
```

```
print(f'The Population mean for males is {np.round(df_male["Purchase"].mean(),4)}')
```

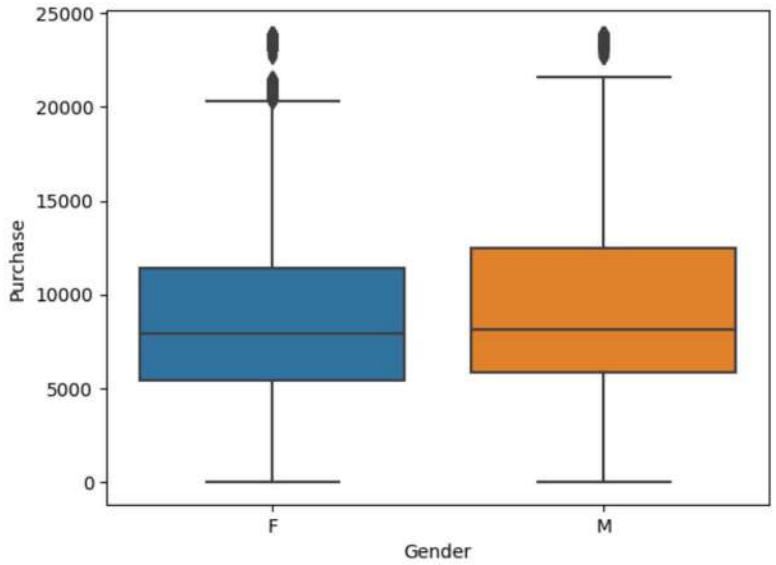
```
The Population mean for males is 9437.526
```

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

Gender	count	mean	std	min	25%	50%	75%	max
F	135809.0	8734.565765	4767.233289	12.0	5433.0	7914.0	11400.0	23959.0
M	414259.0	9437.526040	5092.186210	12.0	5863.0	8098.0	12454.0	23961.0

```
sns.boxplot(x='Gender', y='Purchase', data=df)
```

```
<Axes: xlabel='Gender', ylabel='Purchase'>
```

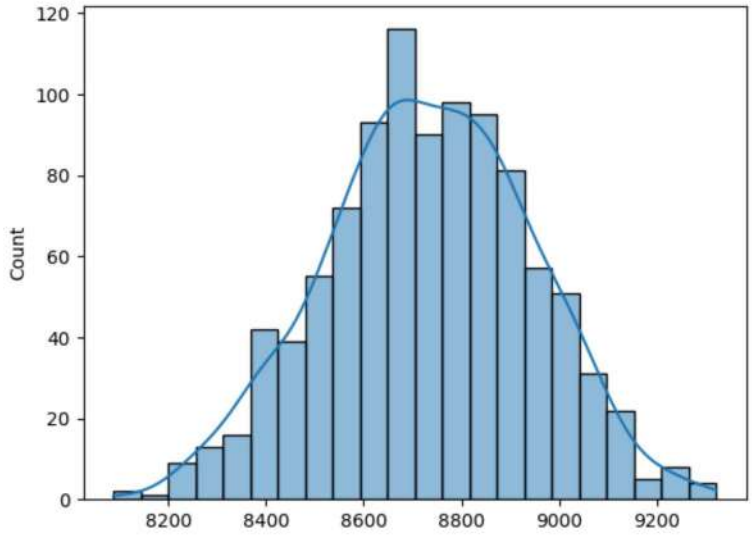


Here, we observe that the average expenses of Men are much higher than of the women populaation. Let us drive this home by taking bootstrapped samples of both the males and Female population and conclude on 95% confidence interval.

```
fem_smp_arr = get_sampled_arr(iterations,df_female["Purchase"],sample_size)
m_smp_arr = get_sampled_arr(iterations,df_male["Purchase"],sample_size)
```

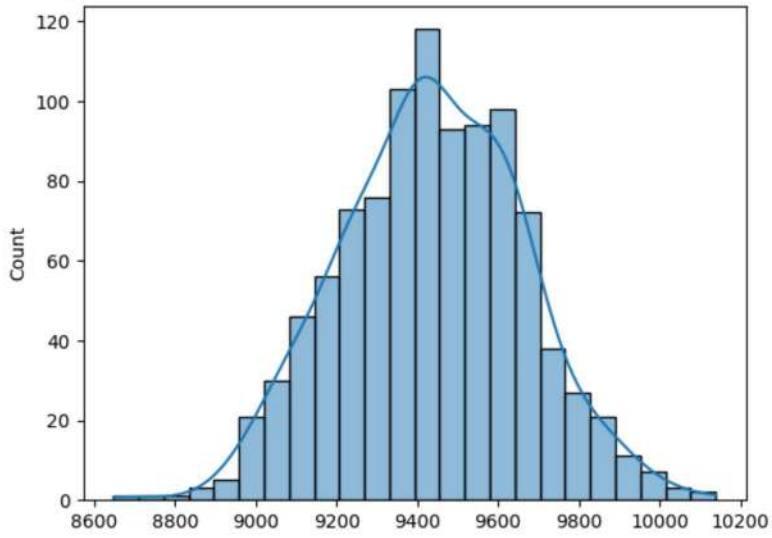
```
sns.histplot(fem_smp_arr, kde=True)
```

```
<Axes: ylabel='Count'>
```




```
sns.histplot(m_smp_arr, kde=True)
```

```
<Axes: ylabel='Count'>
```



We observe that both the smaple means of the Men and women's purchases follow a normal distribution. Hence, we may apply CLT on the smapled means.

Using 90% Confidence Interval

```
smp_mean, lower_bnd, uppr_bnd = Cal_CI(conf_int,fem_smp_arr)

print(f'The Sample Stats for Women with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
```

```
The Sample Stats for Women with 90.0% Confidence is:
  Sample Mean : 8731.6663
  Lower Bound : 8705.115996280001
  Upper Bound : 8758.21660372
```

```
smp_mean, lower_bnd, uppr_bnd = Cal_CI(conf_int,m_smp_arr)

print(f'The Sample Stats for Men with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
```

```
The Sample Stats for Men with 90.0% Confidence is:
  Sample Mean : 9436.5673
  Lower Bound : 9408.115595230001
  Upper Bound : 9465.01900477
```

Using 95% Confidence Interval

```
conf_int = 0.95

smp_mean, lower_bnd, uppr_bnd = Cal_CI(conf_int,fem_smp_arr)

print(f'The Sample Stats for Women with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
```

```
The Sample Stats for Women with 95.0% Confidence is:
  Sample Mean : 8731.6663
  Lower Bound : 8718.42283108
  Upper Bound : 8744.90976892
```

```
smp_mean, lower_bnd, uppr_bnd = Cal_CI(conf_int,m_smp_arr)

print(f'The Sample Stats for Men with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
```

```
The Sample Stats for Men with 95.0% Confidence is:
  Sample Mean : 9436.5673
  Lower Bound : 9422.375399530001
  Upper Bound : 9450.75920047
```

Using 99% Confidence Interval

```
conf_int = 0.99

smp_mean, lower_bnd, uppr_bnd = Cal_CI(conf_int,fem_smp_arr)

print(f'The Sample Stats for Women with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
```

```
The Sample Stats for Women with 99.0% Confidence is:
  Sample Mean : 8731.6663
  Lower Bound : 8729.026055
  Upper Bound : 8734.306545000001
```

```
smp_mean, lower_bnd, uppr_bnd = Cal_CI(conf_int,m_smp_arr)

print(f'The Sample Stats for Men with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
```

```
The Sample Stats for Men with 99.0% Confidence is:
  Sample Mean : 9436.5673
  Lower Bound : 9433.737973750001
  Upper Bound : 9439.39662625
```

As we can see that for the same range of values the average spend of women is lesser compared to men. Their bounds are overlapping which means that for the dataset, we can increase the sample and see how it affects the means. However, for the purposes of this illustration, we can say that Men Spend more than women.

Analysis For Marital Status

```
df_married = df[df["Marital_Status"]==1].drop(columns="Marital_Status")
df_married.head()
```

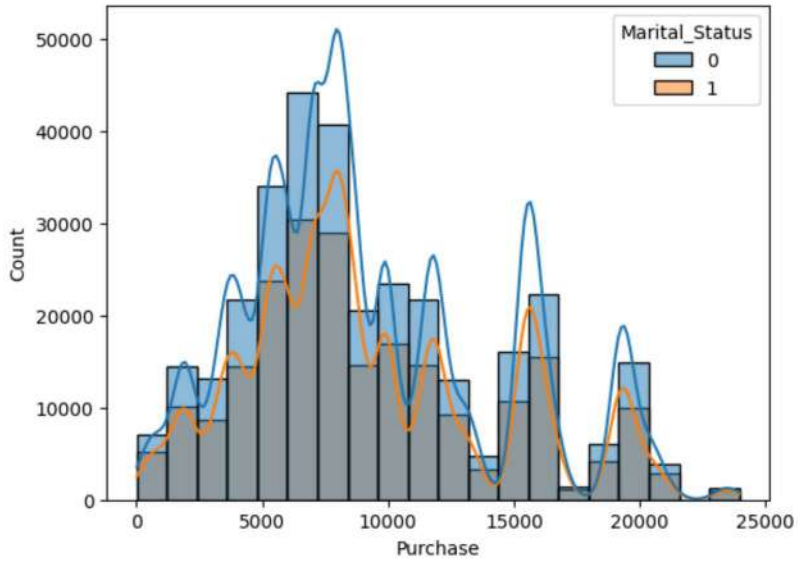
	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Product_Category	Purchase
6	1000004	P00184942	M	46-50	7	B	2	1	19215
7	1000004	P00346142	M	46-50	7	B	2	1	15854
8	1000004	P0097242	M	46-50	7	B	2	1	15686
9	1000005	P00274942	M	26-35	20	A	1	8	7871
10	1000005	P00251242	M	26-35	20	A	1	5	5254

```
df_unmarried = df[df["Marital_Status"]==0].drop(columns="Marital_Status")
df_unmarried.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	3	8370
1	1000001	P00248942	F	0-17	10	A	2	1	15200
2	1000001	P00087842	F	0-17	10	A	2	12	1422
3	1000001	P00085442	F	0-17	10	A	2	12	1057
4	1000002	P00285442	M	55+	16	C	4+	8	7969

```
sns.histplot(data=df,x="Purchase", bins = 20, kde = True, hue = "Marital_Status")
```

```
<Axes: xlabel='Purchase', ylabel='Count'>
```



Loooking at this graph as well, we can identify that the Unmarried people spend more than the married patrons. Though the graphs for both these categories are running close, the overall graph for Married people is lower than Unmarried ones.

```
df_married["Purchase"].mean()
```

```
9261.174574082374
```

```
df_unmarried["Purchase"].mean()
```

```
9265.907618921507
```

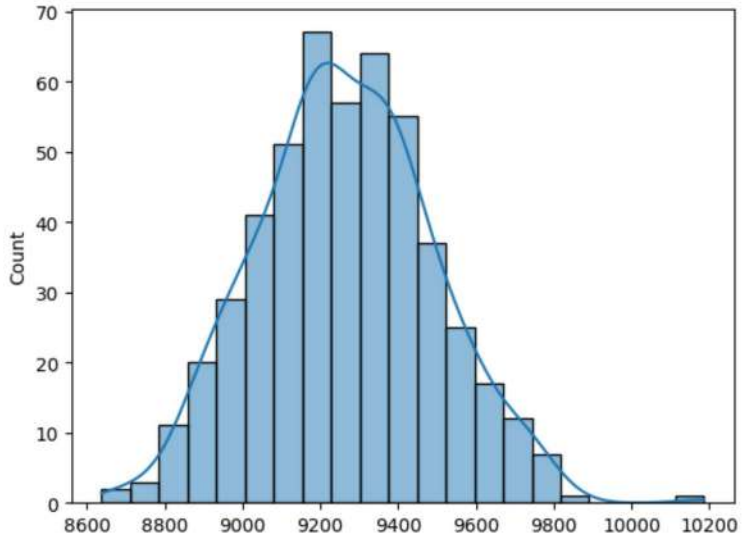
Here, as mentioned above, there seems to be only a slight difference and the Unmarried people have only ever so slightly higher spends than the married people. Let us again check this out by taking bootstrapped samples of both the Married and Unmarried population and conclude on

95% confidence interval.

```
marr_arr = get_sampled_arr(sample_size,df_married["Purchase"],sample_size)
unmarr_arr = get_sampled_arr(sample_size,df_unmarried["Purchase"],sample_size)
```

```
sns.histplot(marr_arr, kde=True)
```

```
<Axes: ylabel='Count'>
```




```
sns.histplot(unmarr_arr, kde=True)
```

We observe that again, both the smaple means of the Married and Unmaried people's purchases follow a normal distribution. So, we may apply CLT on the sampled means.

Using 90% Confidence Interval

```
conf_int = 0.90

smp_mean, lower_bnd, uppr_bnd = Cal_CI(conf_int,marr_arr)

print(f'The Sample Stats for Married People with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
print(f'  Value Range : {uppr_bnd-lower_bnd}')
```

```
The Sample Stats for Married People with 90.0% Confidence is:
  Sample Mean : 9265.6845
  Lower Bound : 9237.30351405
  Upper Bound : 9294.06548595
  Value Range : 56.76197189999948
```

```
smp_mean, lower_bnd, uppr_bnd = Cal_CI(conf_int,unmarr_arr)

print(f'The Sample Stats for Unmarried People with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
print(f'  Value Range : {uppr_bnd-lower_bnd}')
```

```
The Sample Stats for Unmarried People with 90.0% Confidence is:
  Sample Mean : 9269.345
  Lower Bound : 9242.576833039999
  Upper Bound : 9296.11316696
  Value Range : 53.5363339200012
```

Using 95% Confidence Interval

```
conf_int = 0.95

smp_mean, lower_bnd, uppr_bnd = Cal_CI(conf_int,marr_arr)

print(f'The Sample Stats for Married People with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
print(f'  Value Range : {uppr_bnd-lower_bnd}')
```

```
The Sample Stats for Married People with 95.0% Confidence is:
  Sample Mean : 9265.6845
```

```
  Lower Bound : 9251.52787455
  Upper Bound : 9279.841125449999
  Value Range : 28.31325089999882
```

```
smp_mean, lower_bnd, uppr_bnd = Cal_CI(conf_int,unmarr_arr)

print(f'The Sample Stats for Unmarried People with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
print(f'  Value Range : {uppr_bnd-lower_bnd}')
```

```
The Sample Stats for Unmarried People with 95.0% Confidence is:
  Sample Mean : 9269.345
  Lower Bound : 9255.992859439999
  Upper Bound : 9282.69714056
  Value Range : 26.704281120000815
```

Using 99% Confidence Interval

```
conf_int = 0.99

smp_mean, lower_bnd, uppr_bnd = Cal_CI(conf_int,marr_arr)

print(f'The Sample Stats for Married People with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
print(f'  Value Range : {uppr_bnd-lower_bnd}')
```

```
The Sample Stats for Married People with 99.0% Confidence is:
  Sample Mean : 9265.6845
  Lower Bound : 9262.86220625
  Upper Bound : 9268.506793749999
  Value Range : 5.644587499999034
```

```
smp_mean, lower_bnd, uppr_bnd = Cal_CI(conf_int,unmarr_arr)

print(f'The Sample Stats for Unmarried People with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
print(f'  Value Range : {uppr_bnd-lower_bnd}')
```

```
The Sample Stats for Unmarried People with 99.0% Confidence is:
  Sample Mean : 9269.345
  Lower Bound : 9266.683089999999
  Upper Bound : 9272.00691
  Value Range : 5.323820000001433
```

Observations:

- As the confidence interval is increasing, the range of values is decreasing.
- Unmarried people had higher means than married people. Meaning that Unmarried people, for the same range of values, may have higher spends than married people.

```
df_female.groupby(['Marital_Status','Age'])[['Purchase']].sum().div(df_female['Purchase'].sum())*100
```

		Purchase
Marital_Status	Age	
0	0-17	3.573159
	18-25	13.121103
	26-35	21.794095
	36-45	12.668281
	46-50	2.338525
	51-55	2.757374
	55+	1.421996
1	18-25	4.200613
	26-35	15.549022
	36-45	7.853745
	46-50	7.499921
	51-55	4.784654
	55+	2.437514

```
df_female.groupby(['Marital_Status'])[['Purchase']].sum().div(df_female['Purchase'].sum())*100
```

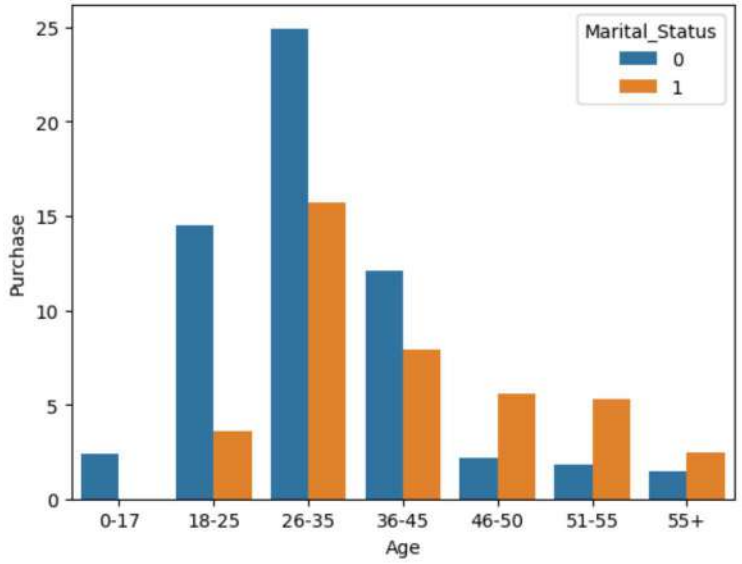
	Purchase
Marital_Status	
0	57.674532
1	42.325468

```
m_grp = df_male.groupby(['Marital_Status','Age'])[['Purchase']].sum().div(df_male['Purchase'].sum())*100
m_grp.reset_index(inplace=True)
m_grp
```

	Marital_Status	Age	Purchase
0	0	0-17	2.366679
1	0	18-25	14.535418
2	0	26-35	24.933661
3	0	36-45	12.119856
4	0	46-50	2.197628
5	0	51-55	1.818188
6	0	55+	1.492075
7	1	18-25	3.583480

```
sns.barplot(data = m_grp,x = 'Age',y='Purchase',hue='Marital_Status')
```

```
<Axes: xlabel='Age', ylabel='Purchase'>
```

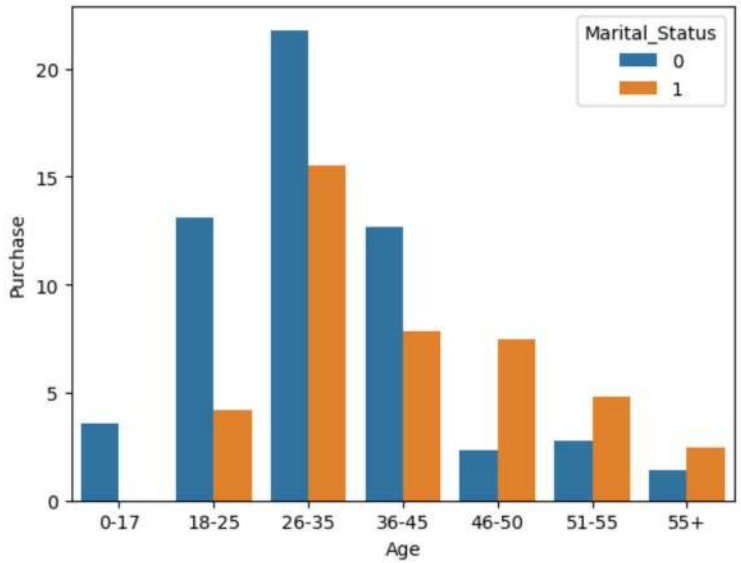



```
f_grp = df_female.groupby(['Marital_Status','Age'])[['Purchase']].sum().div(df_female['Purchase'].sum())*100
f_grp.reset_index(inplace=True)
f_grp
```

	Marital_Status	Age	Purchase
0	0	0-17	3.573159
1	0	18-25	13.121103
2	0	26-35	21.794095
3	0	36-45	12.668281
4	0	46-50	2.338525
5	0	51-55	2.757374

```
sns.barplot(data = f_grp,x = 'Age',y='Purchase',hue='Marital_Status')
```

```
<Axes: xlabel='Age', ylabel='Purchase'>
```

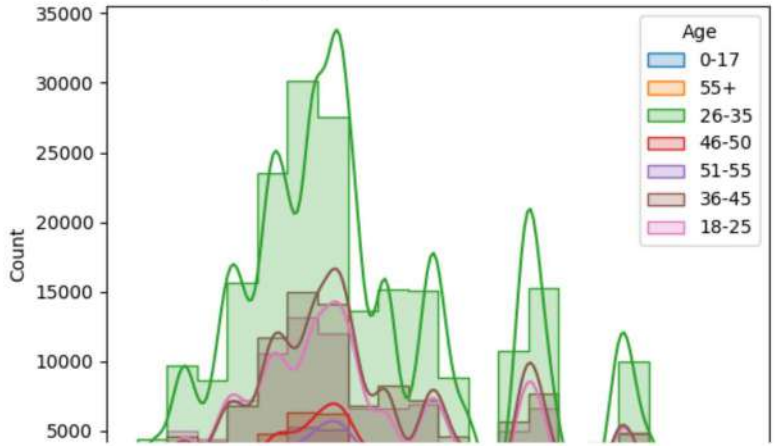


Analysis For Age Groups

```
sample_size = 700
iterations = 1000
conf_int = 0.90
```

```
sns.histplot(data=df,x="Purchase", bins = 20, kde = True, hue = "Age", element="step")
# sns.histplot(data=df,x="Purchase", bins = 20, kde = True, hue = "Age")
```

```
<Axes: xlabel='Purchase', ylabel='Count'>
```




```
df["Age"].unique()
```

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

```
df["Age"].unique()
```

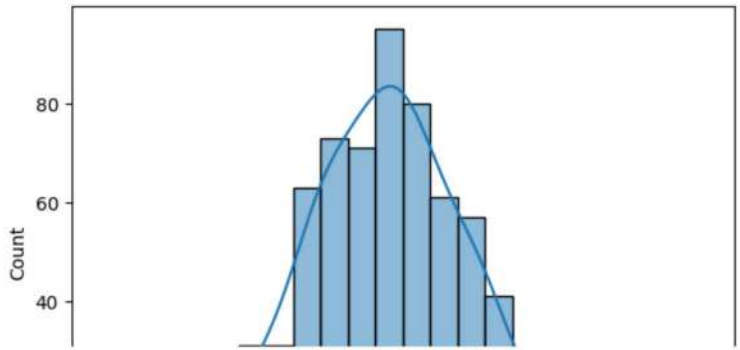
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

```
# For Age Group 0-17

smp_mean, sample_arr, x1, x2 =  get_CI_Stats("0-17",sample_size,iterations,conf_int)

print(f'The Sample Stats for 0-17 Age Group with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
print(f'  Value Range : {uppr_bnd-lower_bnd}')
```

```
The Sample Stats for 0-17 Age Group with 90.0% Confidence is:
  Sample Mean : 8936.9272
  Lower Bound : 9266.683089999999
  Upper Bound : 9272.00691
  Value Range : 5.323820000001433
```

```
sns.histplot(sample_arr, kde=True)
```

```
<Axes: ylabel='Count'>
```

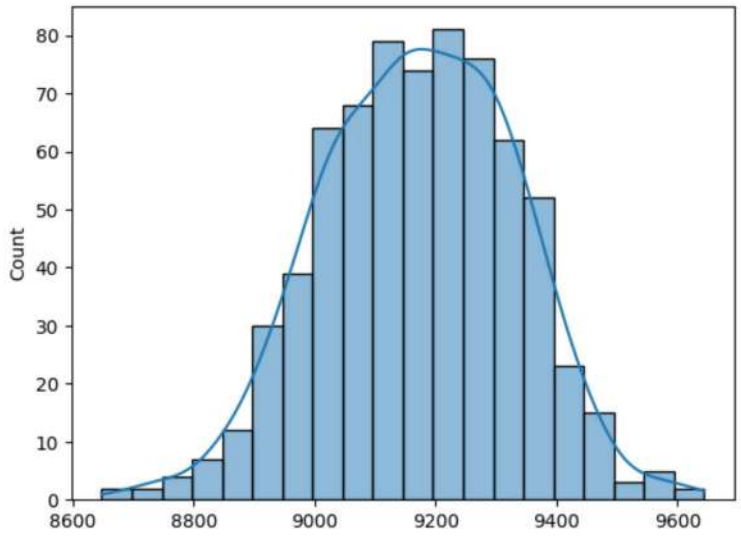



```
# For Age Group 18-25

smp_mean, sample_arr, x1, x2 =  get_CI_Stats("18-25",sample_size,iterations,conf_int)

print(f'The Sample Stats for 18-25 Age Group with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
print(f'  Value Range : {uppr_bnd-lower_bnd}')
```

```
The Sample Stats for 18-25 Age Group with 90.0% Confidence is:
  Sample Mean : 9171.6493
  Lower Bound : 9266.683089999999
  Upper Bound : 9272.00691
  Value Range : 5.323820000001433
```

```
sns.histplot(sample_arr, kde=True)
```

```
<Axes: ylabel='Count'>
```

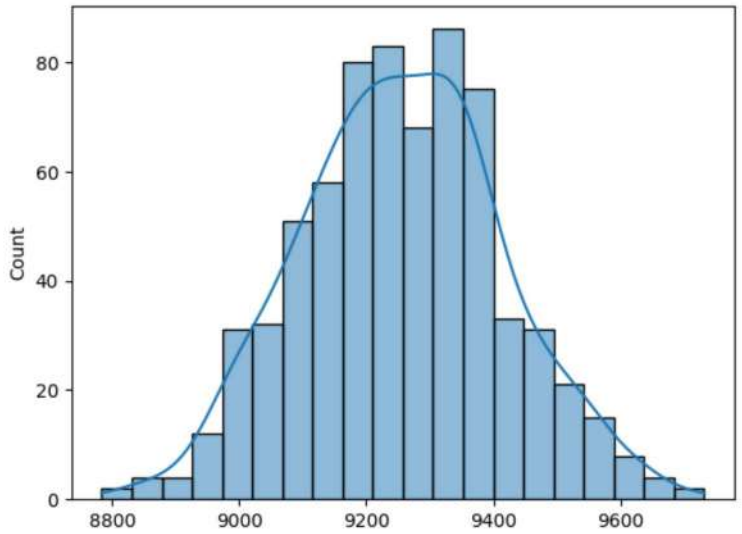


```
# For Age Group 26-35

smp_mean, sample_arr, x1, x2 =  get_CI_Stats("26-35",sample_size,iterations,conf_int)

print(f'The Sample Stats for 26-35 Age Group with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
print(f'  Value Range : {uppr_bnd-lower_bnd}')
```

```
The Sample Stats for 26-35 Age Group with 90.0% Confidence is:
  Sample Mean : 9255.8234
  Lower Bound : 9266.683089999999
  Upper Bound : 9272.00691
  Value Range : 5.323820000001433
```

```
sns.histplot(sample_arr, kde=True)
```

```
<Axes: ylabel='Count'>
```



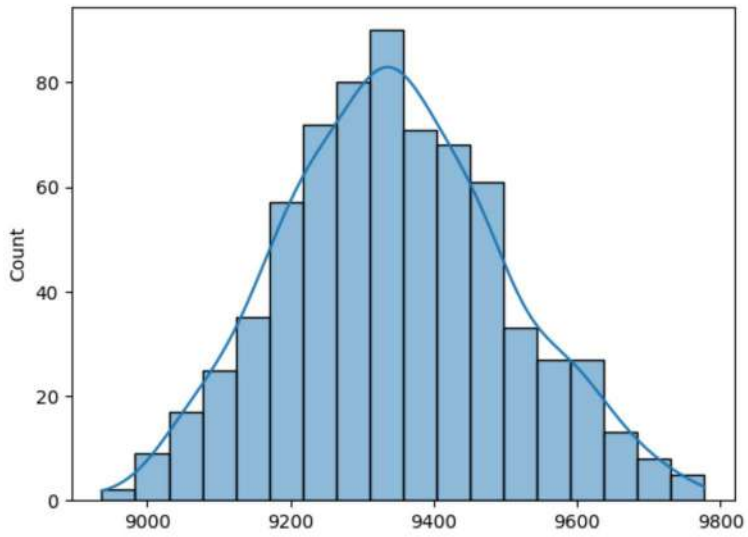

```
# For Age Group 36-45

smp_mean, sample_arr, x1, x2 =  get_CI_Stats("36-45",sample_size,iterations,conf_int)

print(f'The Sample Stats for 36-45 Age Group with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
print(f'  Value Range : {uppr_bnd-lower_bnd}')
```

```
The Sample Stats for 36-45 Age Group with 90.0% Confidence is:
  Sample Mean : 9344.9446
  Lower Bound : 9266.683089999999
```

```
  Upper Bound : 9272.00691
  Value Range : 5.323820000001433
```

```
sns.histplot(sample_arr, kde=True)
```

```
<Axes: ylabel='Count'>
```

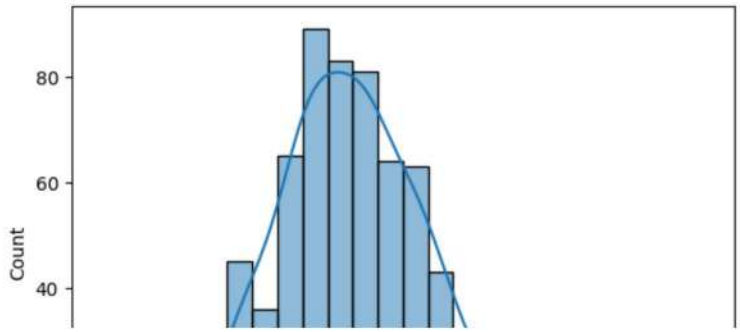



```
# For Age Group 46-50

smp_mean, sample_arr, x1, x2 =  get_CI_Stats("46-50",sample_size,iterations,conf_int)

print(f'The Sample Stats for 46-50 Age Group with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
print(f'  Value Range : {uppr_bnd-lower_bnd}')
```

```
The Sample Stats for 46-50 Age Group with 90.0% Confidence is:
  Sample Mean : 9211.0424
  Lower Bound : 9266.683089999999
  Upper Bound : 9272.00691
  Value Range : 5.323820000001433
```

```
sns.histplot(sample_arr, kde=True)
```

```
<Axes: ylabel='Count'>
```

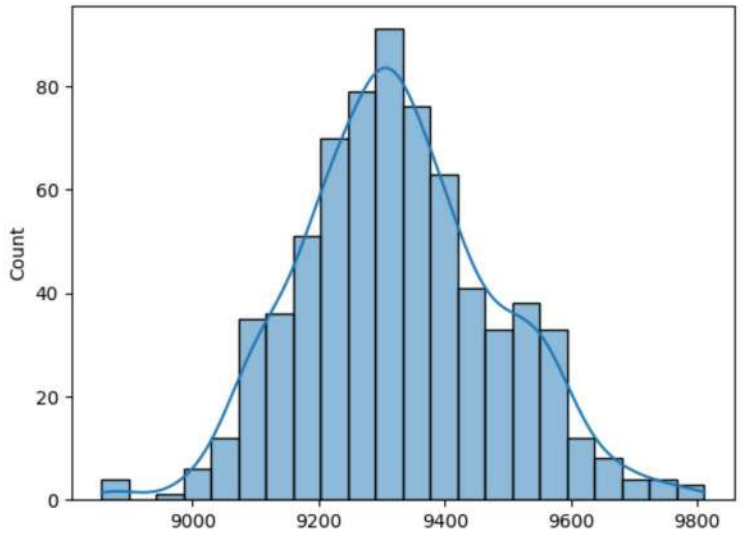



```
# For Age Group 55+

smp_mean, sample_arr, x1, x2 =  get_CI_Stats("55+",sample_size,iterations,conf_int)

print(f'The Sample Stats for 55+ Age Group with {conf_int*100}% Confidence is:')
print(f'  Sample Mean : {smp_mean}')
print(f'  Lower Bound : {lower_bnd}')
print(f'  Upper Bound : {uppr_bnd}')
print(f'  Value Range : {uppr_bnd-lower_bnd}')
```

```
The Sample Stats for 55+ Age Group with 90.0% Confidence is:
  Sample Mean : 9327.2989
  Lower Bound : 9266.683089999999
  Upper Bound : 9272.00691
  Value Range : 5.323820000001433
```

```
sns.histplot(sample_arr, kde=True)
```

```
<Axes: ylabel='Count'>
```



Raw Data Analysis on Means

Due to the huge amount of the dataset and the minimal differences to the naked eye, it would make sense , first , to perform a correlation analysis so that we can avoid any user biases, creeping into our comparitive analysis.

```
df.columns
```

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

```
df_cat = df[["Gender","Age","Occupation","City_Category",
             "Marital_Status",
             "Purchase"]]
```

```
df_cat.head()
```

	Gender	Age	Occupation	City_Category	Marital_Status	Purchase
0	F	0-17	10	A	0	8370
1	F	0-17	10	A	0	15200
2	F	0-17	10	A	0	1422
3	F	0-17	10	A	0	1057
4	M	55+	16	C	0	7969

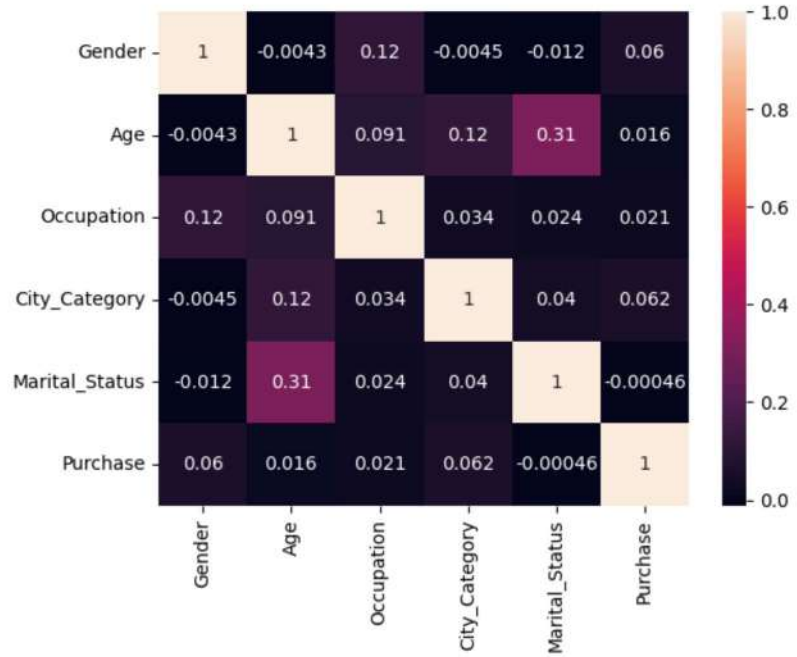
```
for x in df_cat.columns:
    if x != "Purchase":
        df_cat.loc[:,x] = df_cat[x].astype('category').cat.codes
```

```
<ipython-input-65-fbfc692859a9>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cat.loc[:,x] = df_cat[x].astype('category').cat.codes
<ipython-input-65-fbfc692859a9>:3: DeprecationWarning: In a future version, `df.iloc[:, i] = newvals` will attempt to set the values inplace instead of always setting a new array. To
  df_cat.loc[:,x] = df_cat[x].astype('category').cat.codes
<ipython-input-65-fbfc692859a9>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cat.loc[:,x] = df_cat[x].astype('category').cat.codes
<ipython-input-65-fbfc692859a9>:3: DeprecationWarning: In a future version, `df.iloc[:, i] = newvals` will attempt to set the values inplace instead of always setting a new array. To
  df_cat.loc[:,x] = df_cat[x].astype('category').cat.codes
<ipython-input-65-fbfc692859a9>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  df_cat.loc[:,x] = df_cat[x].astype('category').cat.codes
<ipython-input-65-fbfc692859a9>:3: DeprecationWarning: In a future version, `df.iloc[:, i] = newvals` will attempt to set the values inplace instead of always setting a new array. To
  df_cat.loc[:,x] = df_cat[x].astype('category').cat.codes
<ipython-input-65-fbfc692859a9>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cat.loc[:,x] = df_cat[x].astype('category').cat.codes
<ipython-input-65-fbfc692859a9>:3: DeprecationWarning: In a future version, `df.iloc[:, i] = newvals` will attempt to set the values inplace instead of always setting a new array. To
  df_cat.loc[:,x] = df_cat[x].astype('category').cat.codes
<ipython-input-65-fbfc692859a9>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cat.loc[:,x] = df_cat[x].astype('category').cat.codes
<ipython-input-65-fbfc692859a9>:3: DeprecationWarning: In a future version, `df.iloc[:, i] = newvals` will attempt to set the values inplace instead of always setting a new array. To
  df_cat.loc[:,x] = df_cat[x].astype('category').cat.codes
```

```
sns.heatmap(df_cat.corr(), annot=True)
```

```
<Axes: >
```



Observations:

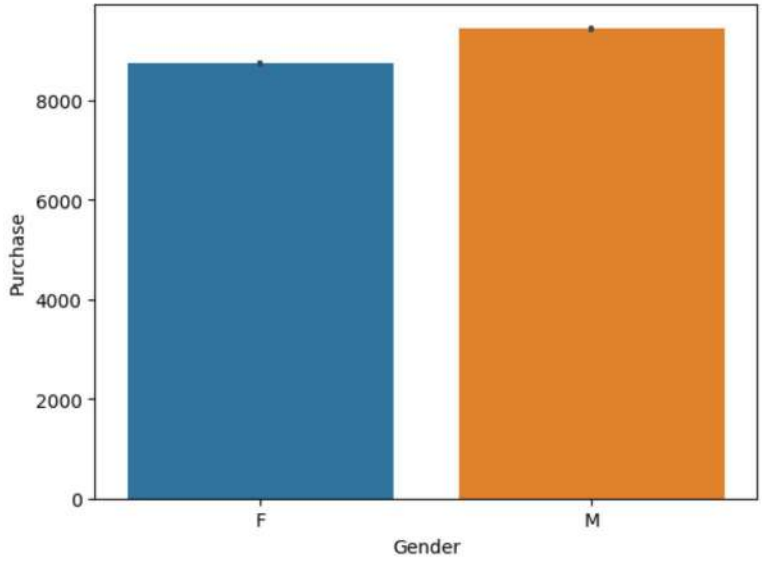
- Purchase is positively affected by
  - City_Category and Gender
  - Occupation

- Age

Graphs

```
sns.barplot(df,y="Purchase",x="Gender")
```

```
<Axes: xlabel='Gender', ylabel='Purchase'>
```

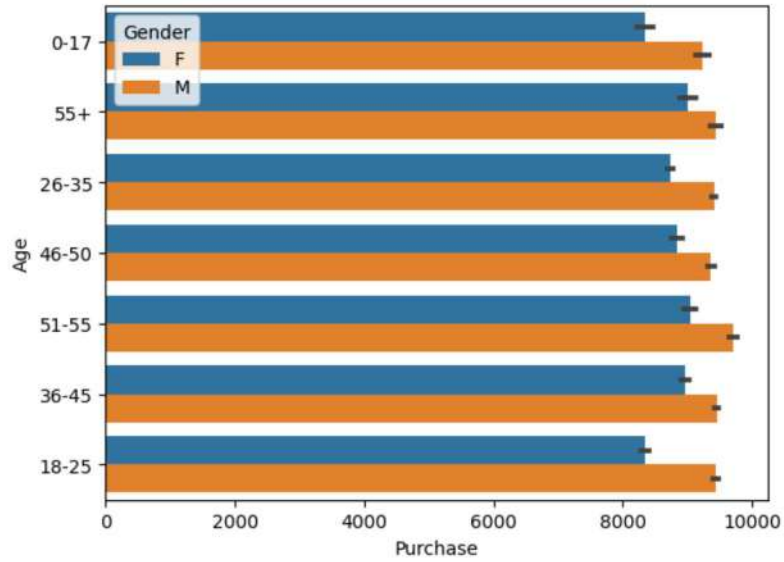



```
sns.barplot(df,y="Purchase",x="Marital_Status", hue="Gender")
```

```
<Axes: xlabel='Marital_Status', ylabel='Purchase'>
```

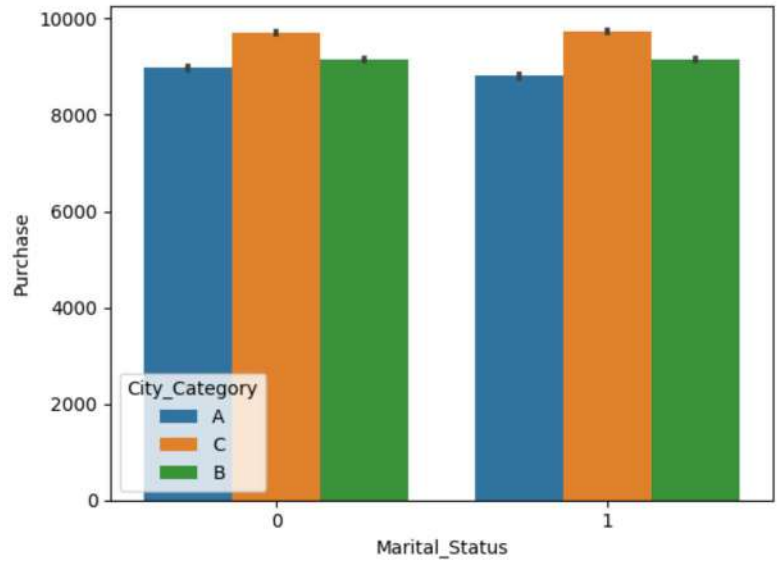
```
sns.barplot(df,hue="Gender",x="Purchase",y="Age")
```

```
<Axes: xlabel='Purchase', ylabel='Age'>
```



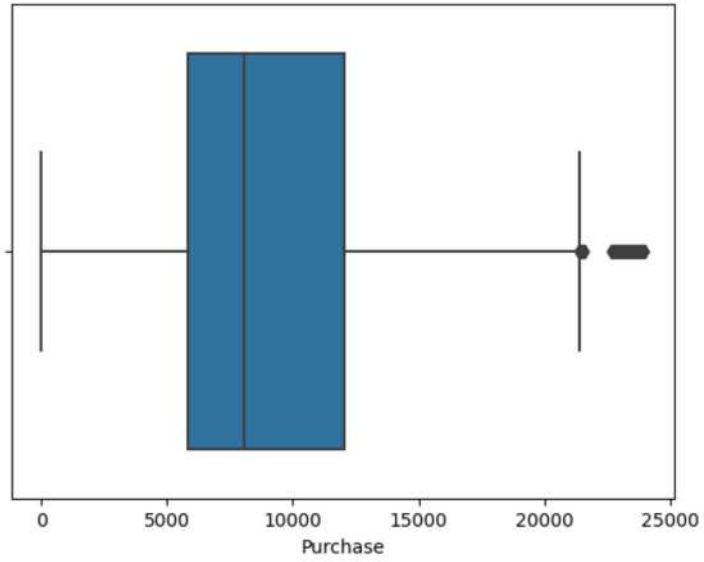

```
sns.barplot(df,y="Purchase",x="Marital_Status", hue="City_Category")
```

```
<Axes: xlabel='Marital_Status', ylabel='Purchase'>
```



```
sns.boxplot(x=df["Purchase"])
```

```
<Axes: xlabel='Purchase'>
```




```
df.groupby(["Gender","Age"])[["Purchase"]].mean().div(df["Purchase"].mean())*100
```

		Purchase
Gender	Age	
F	0-17	90.012955
	18-25	90.060539
	26-35	94.217198
	36-45	96.717123
	46-50	95.446123
	51-55	97.608811
	55+	97.226539
M	0-17	99.689172
	18-25	101.910350
	26-35	101.579980
	36-45	102.042590
	46-50	101.009317
	51-55	104.761740
	55+	101.880694

Observations:

- Purchasing is positively affected by the following factors:
  - Gender
  - City Category
  - Occupation
  - Age, though the relationship is weak.
- Men, on an average, purchase more than Women.
- City category which affects the purchase decision is C < B < A, regardless of the Gender. People in City Category C will purchase more than B, which will purchase more than A.
- Marital Status is a Statitically significant factor which affects the purchase decision. Unmarried people, on an average, tend to purchase more than Married people as shown above with the 95% Confidence.
- Gender is also a Statitically significant factor which affects the purchase decision. Men, on an average, tend to purchase more than Women as shown above with the 95% Confidence.
- People in the age of 26-35, make the most total value in purchases on an Average, regardless of the other factors. This is closely followed by 36-45 and 18-25 ranges.
- Items above 20k are outliers.
- The crowd, however, had most purchases in number by the age group of 26-35.

Confidence Interval Details (Gender):

- For a sample bootstrapped data of size 1000, even if the population distribution was not normal, the sample means was following a normal distribution, which means that CLT can be very well applied to these distributions.
- 95% confidence interval for this run resulted in men with average spend higher than women.
- Having said that, the sample mean was still closer to the population mean.

Insights

- Women Seem to be spending more than men if they are married compared to if they are unmarried. Combined expenditure power may be attributed to this, however, a further analysis can be done with more data to confirm the same.
- Walmart may target Married women to endorse products and there will be higher success rate.
- People from City C, regardless of other factors, seem to have higher purchasing power, which makes them an apt target for marketting high-end porducts.
- Prices over 22000 normally is an outlier, is an can be categorized as outliers and the target audience for these products may be people from city C or people with higher combined purchasing power.

✓ 0s completed at 12:42 AM