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


from scipy.stats import norm, t
from scipy.stats import binom, geom
```

Part 1

```
# There are 50 million male and female
# This is the given sample
df=pd.read_csv("/content/Walmart.csv")
df
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Year
0	1000001	P00069042	F	0-17	10.0	A	
1	1000001	P00248942	F	0-17	10.0	A	
2	1000001	P00087842	F	0-17	10.0	A	
3	1000001	P00085442	F	0-17	10.0	A	
4	1000002	P00285442	M	55+	16.0	C	4
...	...	...	...	...	...	...	.
275485	1000461	P00102942	M	51-55	7.0	B	
275486	1000461	P00231242	M	51-55	7.0	B	

```
# basic info of table
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 275490 entries, 0 to 275489
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     275490 non-null  int64  
 1   Product_ID                  275490 non-null  object 
 2   Gender                      275490 non-null  object 
 3   Age                         275490 non-null  object 
 4   Occupation                  275489 non-null  float64
 5   City_Category               275489 non-null  object 
 6   Stay_In_Current_City_Years  275489 non-null  object 
 7   Marital_Status              275489 non-null  float64
 8   Product_Category            275489 non-null  float64
 9   Purchase                    275489 non-null  float64
dtypes: float64(4), int64(1), object(5)
memory usage: 21.0+ MB
```

```
# what is the shape ?
df.shape
```

```
(275490, 10)
```

```
# knowing about all the columns
df.columns
```

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

```
# checking martial columns
df['Marital_Status'].value_counts()
```

```
0.0    162457
1.0    113032
Name: Marital_Status, dtype: int64
```

```
# checking Occupation columns
df['Occupation'].value_counts()
```

```
4.0    36318
0.0    34923
7.0    29741
1.0    23856
17.0   20036
20.0   16796
12.0   15387
14.0   13670
2.0    13375
16.0   12715
6.0    10226
3.0     8843
10.0    6350
15.0    6123
5.0     6013
11.0    5857
19.0    4195
13.0    3891
18.0    3264
9.0     3172
8.0      738
Name: Occupation, dtype: int64
```

```
# checking Product_Category columns
df['Product_Category'].value_counts()
```

```
5.0    76363
1.0    70774
8.0    57666
11.0   12172
2.0    12043
3.0    10199
6.0    10160
4.0     5955
16.0    4958
15.0    3162
13.0    2783
10.0    2585
12.0    1985
7.0     1882
18.0    1538
14.0     754
17.0     302
9.0      208
Name: Product_Category, dtype: int64
```

```
df.describe()
```

	User_ID	Occupation	Marital_Status	Product_Category	Purchase
count	2.754900e+05	275489.000000	275489.000000	275489.000000	275489.000000
mean	1.003001e+06	8.063026	0.410296	5.293493	9321.198745
std	1.742508e+03	6.521791	0.491888	3.744453	4973.658109
min	1.000001e+06	0.000000	0.000000	1.000000	185.000000
25%	1.001469e+06	2.000000	0.000000	1.000000	5864.000000
50%	1.003051e+06	7.000000	0.000000	5.000000	8060.000000
75%	1.004465e+06	14.000000	1.000000	8.000000	12062.000000
max	1.006040e+06	20.000000	1.000000	18.000000	23961.000000

```
Mean_Purchase = 9321.198745
Median_puchase = np.median(df['Purchase'])
print(Median_puchase)
difference = Mean_Purchase-Median_puchase
print(difference)
```

```
8060.0
1261.1987449999997
```

```
# no of unique user_id
df['User_ID'].nunique()
```

```
5891
```

```
df['City_Category'].unique()
```

```
array(['A', 'C', 'B'], dtype=object)
```

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

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

```
# cheking gender
df['Gender'].unique()
```

```
array(['F', 'M'], dtype=object)
```

```
# count no of male and female
df['Gender'].value_counts()
```

```
M    207545
F     67944
Name: Gender, dtype: int64
```

Part 2 null_value and outliers ang graph

```
# checking null value
df.isna().sum()
```

```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    1
City_Category                 1
Stay_In_Current_City_Years    1
Marital_Status                1
Product_Category              1
Purchase                      1
dtype: int64
```

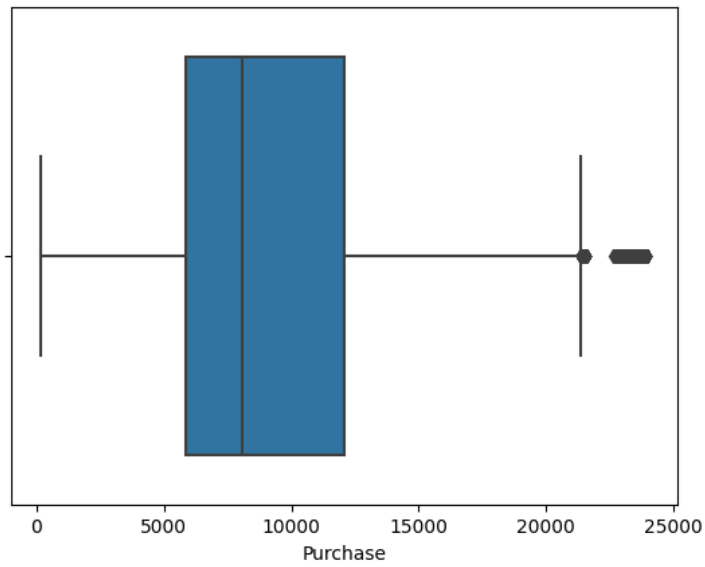
```
# dop last row as it contain most columns null
df=df.dropna(axis=0)
```

```
df.tail(5)
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status
275484	1000461	P00194042	M	51-55	7.0	B	0	0.0
275485	1000461	P00102942	M	51-55	7.0	B	0	0.0
275486	1000461	P00231242	M	51-55	7.0	B	0	0.0
				51-				

```
# outliers
sns.boxplot(x=df['Purchase'])
```

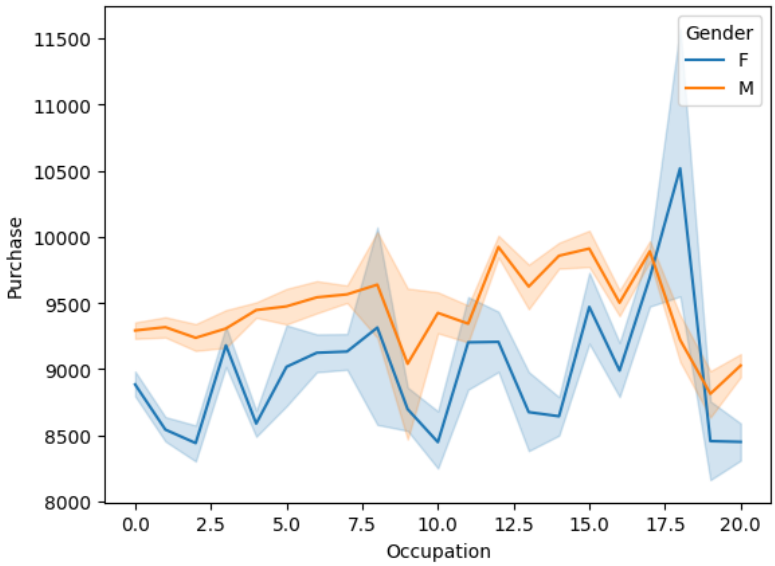
```
<Axes: xlabel='Purchase'>
```



```
outlier=df[df['Purchase']>22000]
outlier.head()
```

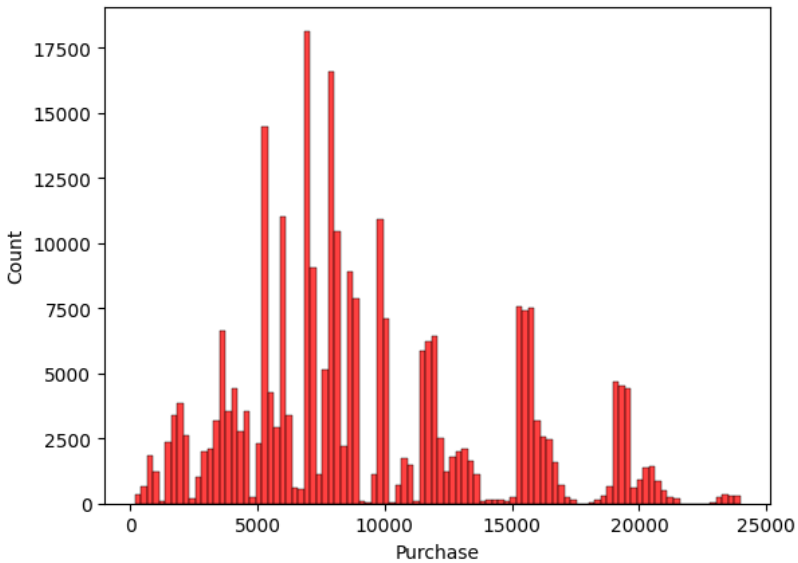
	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status
343	1000058	P00117642	M	26-35	2.0	B	3	0.0
375	1000062	P00119342	F	36-45	3.0	A	1	0.0
652	1000126	P00087042	M	18-25	9.0	B	1	0.0
				26-				

```
# how occupation related to purchase [NNC]
sns.lineplot(data=df,
 x="Occupation",
 y="Purchase",
 hue='Gender'
)
plt.show()
```



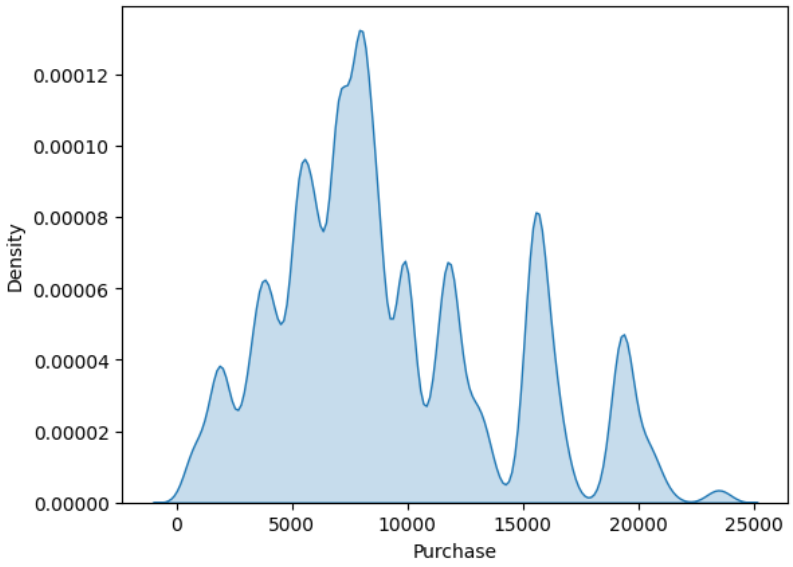

```
# purchase distribution
sns.histplot(df,x=df['Purchase'],bins=100,color='r')
```

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




```
sns.kdeplot(x=df["Purchase"],shade=True)
plt.show()
```

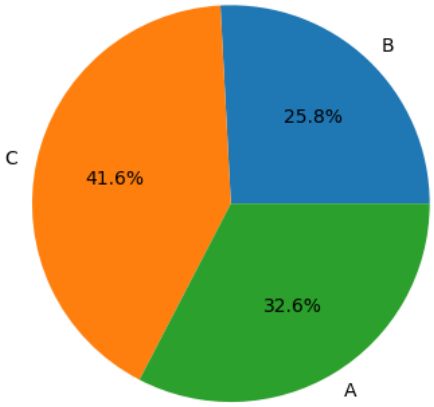
```
<ipython-input-167-3dfcfa70c9cb>:1: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(x=df["Purchase"],shade=True)
```




```
# which city has high purchase
l=df.groupby('City_Category')['Purchase'].sum()
l
```

```
City_Category
A    6.635956e+08
B    1.067039e+09
C    8.372536e+08
Name: Purchase, dtype: float64
```

```
plt.pie(x=l,labels=['B','C','A'],autopct='%1.1f%%')
plt.show()
```



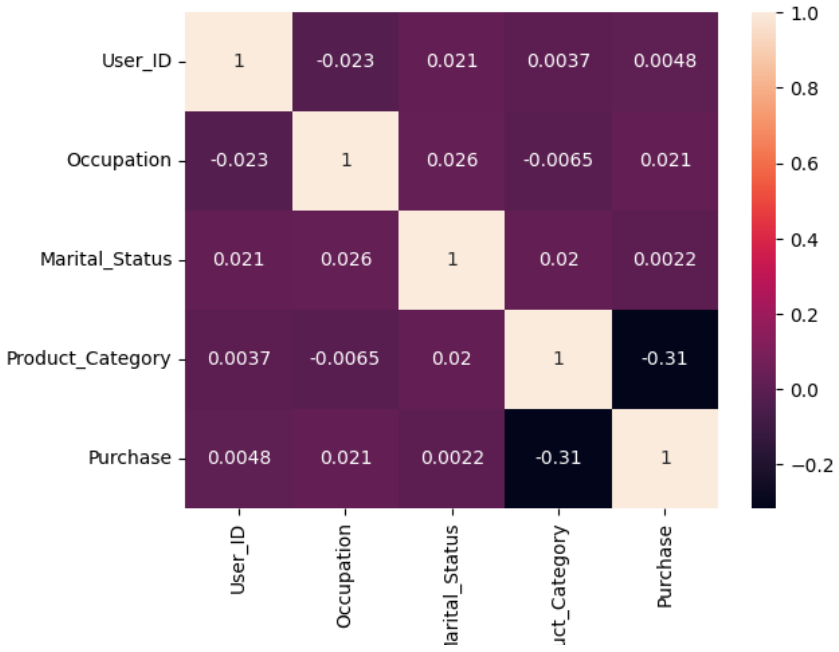

```
df.corr()
```

```
<ipython-input-187-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is depre
  df.corr()
```

	User_ID	Occupation	Marital_Status	Product_Category	Purchase
User_ID	1.000000	-0.023222	0.021444	0.003651	0.004835
Occupation	-0.023222	1.000000	0.026302	-0.006517	0.020561
Marital_Status	0.021444	0.026302	1.000000	0.020187	0.002154
Product_Category	0.003651	-0.006517	0.020187	1.000000	-0.314495
Purchase	0.004835	0.020561	0.002154	-0.314495	1.000000

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

```
<ipython-input-189-8df7bcac526d>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is depre
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```



part 3

#Estimating population mean 50 millian

```
# two saperate DataFrame for male and Female
male_data = df[df['Gender'] =='M']
female_data = df[df['Gender'] =='F']
```

```
male_data.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Pro
4	1000002	P00285442	M	55+	16.0	C	4+	0.0	
5	1000003	P00193542	M	26-35	15.0	A	3	0.0	
6	1000004	P00184942	M	46-50	7.0	B	2	1.0	
7	1000004	P00346142	M	46-50	7.0	B	2	1.0	

```
female_data.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Pr
0	1000001	P00069042	F	0-17	10.0	A	2	0.0	
1	1000001	P00248942	F	0-17	10.0	A	2	0.0	
2	1000001	P00087842	F	0-17	10.0	A	2	0.0	
				0-					

```
print(np.mean(male_data['Purchase']))
print(np.mean(female_data['Purchase']))
```

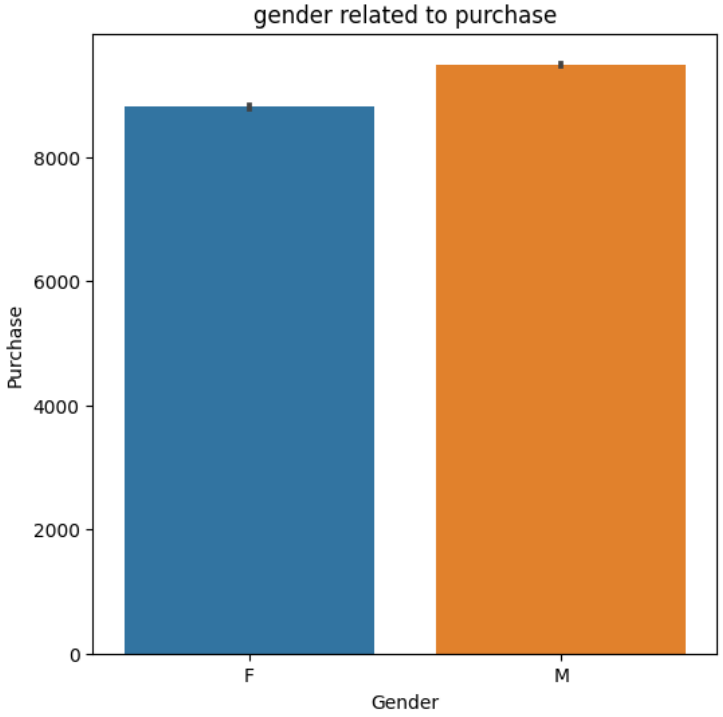
```
9489.457004505048
8807.228997409631
```

```
print(male_data.shape)
print(female_data.shape)
```

```
(207545, 10)
(67944, 10)
```

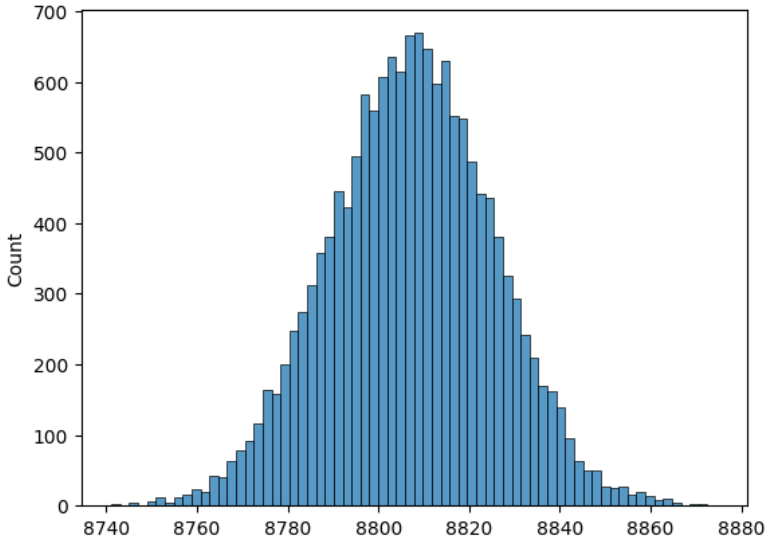
-->how gender related to purchase

```
plt.figure(figsize=(6,6))
plt.title('gender related to purchase',fontsize=12)
sns.barplot(data=df,
            x="Gender",
            y="Purchase",
            estimator=np.mean)

plt.show()
```




```
# female...,
# method 1 <boothstraping>///
sample = pd.Series(female_data['Purchase']) #sample
#sample len
sample_size=female_data.shape[0]
print(sample_size)

store_here=[]
for i in range (15000):
  random_sample=np.random.choice(sample,size=sample_size)
  random_sample_mean=random_sample.mean()
  store_here.append(random_sample_mean)

# hisplot to get visual clarity
sns.histplot(store_here)
```

```
67944
<Axes: ylabel='Count'>
```



```
# now 95% confidence interval thal female purchase population mean lie ...,
print(np.mean(store_here))
np.round(np.percentile(store_here,[2.5,97.5]))
```

```
8807.227284698183
array([8772., 8842.])
```

```
# method 2
# using CLT to find 95% confidence interval

sample_f=pd.Series(female_data['Purchase'])
# sample size
n=sample_f.size
print(n)
sample_f_mean=sample_f.mean()
print('sample_f_mean = ',sample_f_mean)
sample_f_std=sample_f.std()
print('sample_f_std',sample_f_std)
sample_f_se=sample_f_std/np.sqrt(n)
print('sample_f_se',sample_f_se)
```

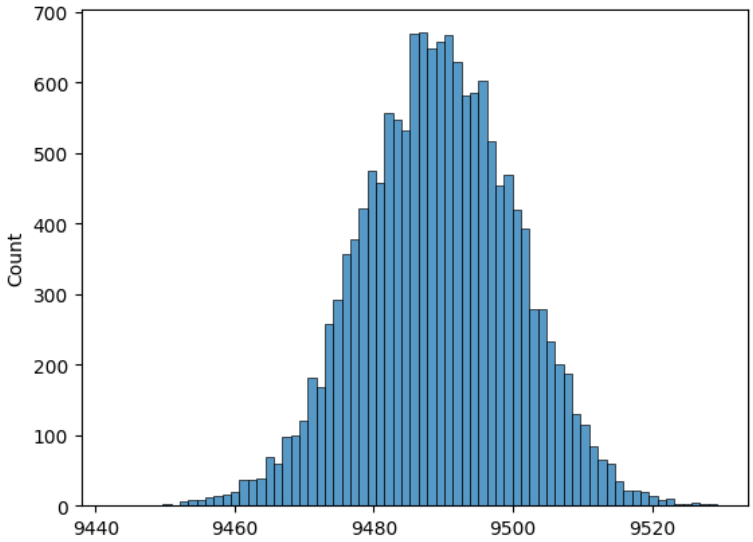
```
67944
sample_f_mean =  8807.228997409631
sample_f_std 4707.520731524716
sample_f_se 18.059955910862055
```

```
# z score of 95% is
print(norm.ppf(.975))
# left and right interval
print("left interval",round(sample_f_mean-(1.96*sample_f_se)),"right interval",round(sample_f_mean+(1.96*sample_f_se)))
```

```
1.959963984540054
left interval 8772 right interval 8843
```

```
#male
# method 1 <boothstraping>///
sample = pd.Series(male_data['Purchase']) #sample
#sample len
sample_size=male_data.shape[0]
print(sample_size)

store_here=[]
for i in range (15000):
  random_sample=np.random.choice(sample,size=sample_size)
  random_sample_mean=random_sample.mean()
  store_here.append(random_sample_mean)

# hisplot to get visual clarity
sns.histplot(store_here)
```

```
207545
<Axes: ylabel='Count'>
```




```
# now 95% confidence interval thal male purchase population mean lie ...,
print(np.mean(store_here))
np.round(np.percentile(store_here,[2.5,97.5]))
```

```
9489.340873932882
array([9467., 9511.])
```

```
# method 2
# using CLT to find 95% confidence interval

sample_f=pd.Series(male_data['Purchase'])
# sample size
n=sample_f.size
print(n)
sample_f_mean=sample_f.mean()
print('sample_f_mean = ',sample_f_mean)
sample_f_std=sample_f.std()
print('sample_f_std',sample_f_std)
sample_f_se=sample_f_std/np.sqrt(n)
print('sample_f_se',sample_f_se)
```

```
207545
sample_f_mean =  9489.457004505048
sample_f_std 5046.39189966458
sample_f_se 11.077068386909962
```

```
# z score of 95% is
print(norm.ppf(.975))
# left and right interval
print("left interval",round(sample_f_mean-(1.96*sample_f_se)),"right interval",round(sample_f_mean+(1.96*sample_f_se)))
```

```
1.959963984540054
left interval 9468 right interval 9511
```

```
# //insight//
#1.male 95% confidence interval[9468,9511]
#2.female 95% confidence interval[8772,8843]
#thus with 95% confidence that average male purchase is more then female


#//Q1 does female spend more money then men
# ans : no
```

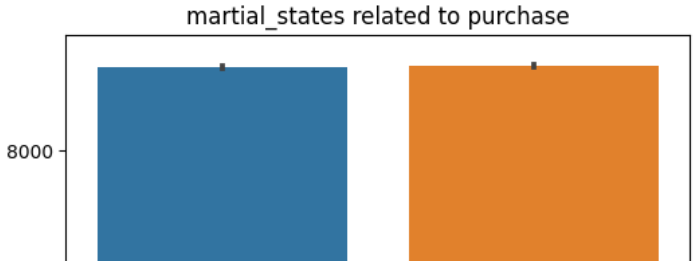
recamondation : There can we more item related to male then female,,,

```
df.groupby(['City_Category','Gender'])['Gender'].count()
```

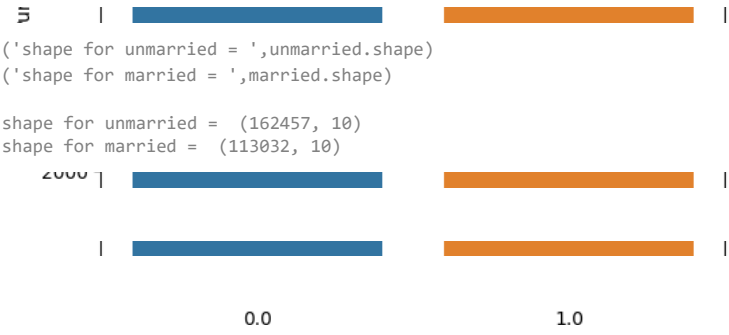
```
City_Category  Gender
A              F         17877
               M         56381
B              F         28953
               M         87227
C              F         21114
               M         63937
Name: Gender, dtype: int64
```

-->how martial_states related to purchase

```
plt.figure(figsize=(6,6))
plt.title('martial_states related to purchase',fontsize=12)
sns.barplot(data=df,
            x="Marital_Status",
            y="Purchase",
            estimator=np.mean)

plt.show()
```




```
# two saperate DataFrame for male and Female
unmarried = df[df['Marital_Status'] ==0.0]
married = df[df['Marital_Status'] ==1.0]
```



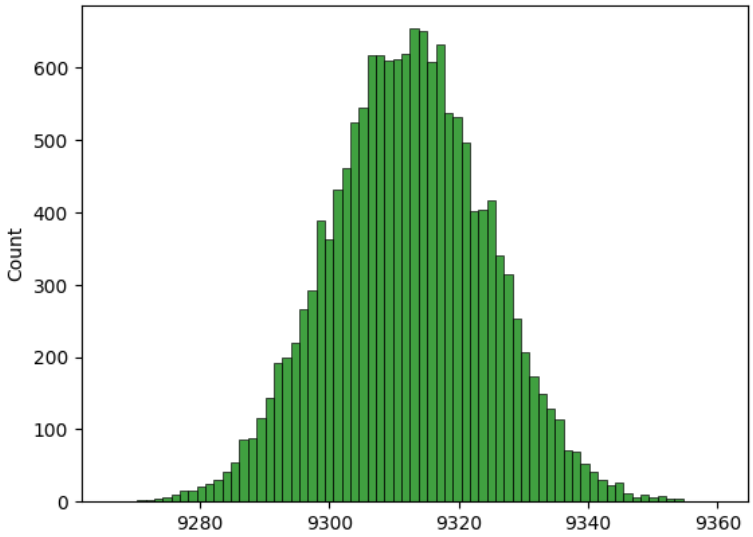
```
print('shape for unmarried = ',unmarried.shape)
print('shape for married = ',married.shape)
```

```
shape for unmarried =  (162457, 10)
shape for married =  (113032, 10)
```



```
# unmarried...,
# method 1 <boothstraping>///
sample = pd.Series(unmarried['Purchase']) #sample
#sample len
sample_size=unmarried.shape[0]
print(sample_size)

store_here=[]
for i in range (15000):
  random_sample=np.random.choice(sample,size=sample_size)
  random_sample_mean=random_sample.mean()
  store_here.append(random_sample_mean)

# hisplot to get visual clarity
sns.histplot(store_here,color='g')
```

```
162457
<Axes: ylabel='Count'>
```




```
# now 95% confidence interval thal unmarried purchase population mean lie ...,
print(np.mean(store_here))
np.round(np.percentile(store_here,[2.5,97.5]))
```

```
9312.200752265111
array([9288., 9336.])
```

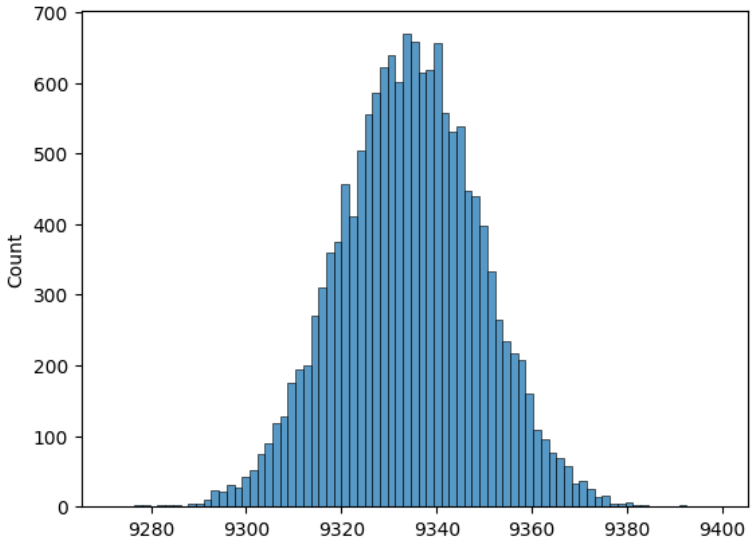
```
# now 95% confidence interval thal unmarried purchase population mean lie ...,
print(np.mean(store_here))
np.round(np.percentile(store_here,[5,95]))
```

```
9312.200752265111
array([9292., 9332.])
```

Double-click (or enter) to edit

```
# married...,
sample = pd.Series(married['Purchase']) #sample
#sample len
sample_size=married.shape[0]
print(sample_size)

store_here=[]
for i in range (15000):
  random_sample=np.random.choice(sample,size=sample_size)
  random_sample_mean=random_sample.mean()
  store_here.append(random_sample_mean)

# hisplot to get visual clarity
sns.histplot(store_here)
```

```
113032
<Axes: ylabel='Count'>
```




```
# now 95% confidence interval thal married purchase population mean lie ...,
print(np.mean(store_here))
np.round(np.percentile(store_here,[2.5,97.5]))
```

```
9334.255519658149
array([9305., 9363.])
```

```
# now 90% confidence interval thal married purchase population mean lie ...,
print(np.mean(store_here))
np.round(np.percentile(store_here,[5,95]))
```

```
9334.255519658149
array([9310., 9358.])
```

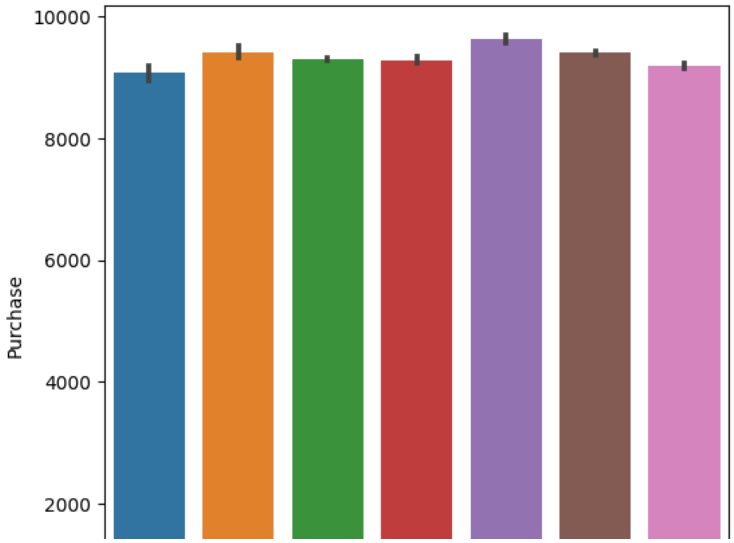
```
# //insight//
# unmarried purchase 95% lie in [9288,9336]
# married purchase 95% lie in [9305,9363]
```

so we can conclude that both married and unmarried purchase are overlaped

recomendation : it is advicable to focus on both reqirement of married and unmarried

--> how age related to purchase

```
plt.figure(figsize=(6,6))
sns.barplot(data=df,
            x="Age",
            y="Purchase",
            estimator=np.mean)
plt.show()
```




```
less_then_or_17 = df[df['Age']=='0-17']
From_18_25 = df[df['Age'] =='18-25']
From_26_35 = df[df['Age'] =='26-35']
From_36_45 = df[df['Age'] =='36-45']
From_46_50 = df[df['Age'] =='46-50']
From_51_55 = df[df['Age'] =='51-55']
more_then_or_55= df[df['Age'] =='55+']


# less_then_or_17...,
sample = pd.Series(less_then_or_17['Purchase']) #sample
#sample len
sample_size=less_then_or_17.shape[0]
print('sample_size =',sample_size)

store_here=[]
for i in range (15000):
  random_sample=np.random.choice(sample,size=sample_size)
  random_sample_mean=random_sample.mean()
  store_here.append(random_sample_mean)
# now 95% confidence interval thal less_then_or_17 purchase population mean lie ...,
print('mean = ',np.mean(store_here))
print("90% = ",np.round(np.percentile(store_here,[2.5,97.5])))
print("80% = ",np.round(np.percentile(store_here,[10,90])))
print("68% = ",np.round(np.percentile(store_here,[16,84])))

     sample_size = 7405
     mean =  9079.709808984919
     90% =  [8965. 9198.]
     80% =  [9005. 9155.]


# From_18_25...,
sample = pd.Series(From_18_25['Purchase']) #sample
#sample len
sample_size=From_18_25.shape[0]
print('sample_size =',sample_size)

store_here=[]
for i in range (15000):
  random_sample=np.random.choice(sample,size=sample_size)
  random_sample_mean=random_sample.mean()
  store_here.append(random_sample_mean)
# now 95% confidence interval thal less_then_or_17 purchase population mean lie ...,
print('mean = ',np.mean(store_here))
print("90% = ",np.round(np.percentile(store_here,[2.5,97.5])))
print("80% = ",np.round(np.percentile(store_here,[10,90])))
print("68% = ",np.round(np.percentile(store_here,[16,84])))

     sample_size = 50493
     mean =  9191.33547099334
     90% =  [9148. 9235.]
     80% =  [9162. 9220.]
     68% =  [9169. 9213.]


# From_26_35...,
sample = pd.Series(From_26_35['Purchase']) #sample
#sample len
sample_size=From_26_35.shape[0]
print(sample_size)

store_here=[]
for i in range (15000):
  random_sample=np.random.choice(sample,size=sample_size)
  random_sample_mean=random_sample.mean()
  store_here.append(random_sample_mean)
```

```
# now 95% confidence interval thal married purchase population mean lie ...,
print(np.mean(store_here))
print("90% = ",np.round(np.percentile(store_here,[2.5,97.5])))
print("80% = ",np.round(np.percentile(store_here,[10,90])))
print("68% = ",np.round(np.percentile(store_here,[16,84])))
```

```
109866
9300.051410230039
90% =  [9270. 9329.]
80% =  [9281. 9319.]
68% =  [9285. 9315.]
```

```
# From_36_45...,
sample = pd.Series(From_36_45['Purchase']) #sample
#sample len
sample_size=From_36_45.shape[0]
print(sample_size)

store_here=[]
for i in range (15000):
  random_sample=np.random.choice(sample,size=sample_size)
  random_sample_mean=random_sample.mean()
  store_here.append(random_sample_mean)
# now 95% confidence interval thal From_36_45 purchase population mean lie ...,
print(np.mean(store_here))
print("90% = ",np.round(np.percentile(store_here,[2.5,97.5])))
print("80% = ",np.round(np.percentile(store_here,[10,90])))
print("68% = ",np.round(np.percentile(store_here,[16,84])))
```

```
54855
9402.58393827606
90% =  [9361. 9444.]
80% =  [9375. 9430.]
68% =  [9381. 9424.]
```

```
# From_46_50...,
sample = pd.Series(From_46_50['Purchase']) #sample
#sample len
sample_size=From_46_50.shape[0]
print(sample_size)

store_here=[]
for i in range (15000):
  random_sample=np.random.choice(sample,size=sample_size)
  random_sample_mean=random_sample.mean()
  store_here.append(random_sample_mean)
# now 95% confidence interval thal From_46_50 purchase population mean lie ...,
print(np.mean(store_here))
print("90% = ",np.round(np.percentile(store_here,[2.5,97.5])))
print("80% = ",np.round(np.percentile(store_here,[10,90])))
print("68% = ",np.round(np.percentile(store_here,[16,84])))
```

```
22812
9285.985222944066
90% =  [9222. 9349.]
80% =  [9244. 9327.]
68% =  [9254. 9318.]
```

```
# From_51_55...,
sample = pd.Series(From_51_55['Purchase']) #sample
#sample len
sample_size=From_51_55.shape[0]
print(sample_size)

store_here=[]
for i in range (15000):
  random_sample=np.random.choice(sample,size=sample_size)
  random_sample_mean=random_sample.mean()
  store_here.append(random_sample_mean)
# now 95% confidence interval thal married From_51_55 population mean lie ...,
print(np.mean(store_here))
print("90% = ",np.round(np.percentile(store_here,[2.5,97.5])))
print("80% = ",np.round(np.percentile(store_here,[10,90])))
print("68% = ",np.round(np.percentile(store_here,[16,84])))
```

```
19311
9633.259708162877
90% =  [9563. 9704.]
80% =  [9587. 9680.]
68% =  [9598. 9670.]
```

```
# more_then_or_55...,
sample = pd.Series(more_then_or_55['Purchase']) #sample
#sample len
sample_size=more_then_or_55.shape[0]
print(sample_size)
```

```
store_here=[]
for i in range (15000):
  random_sample=np.random.choice(sample,size=sample_size)
  random_sample_mean=random_sample.mean()
  store_here.append(random_sample_mean)
#now 95% confidence interval thal more_then_or_55 purchase population mean lie ...,
print(np.mean(store_here))
print("95% = ",np.round(np.percentile(store_here,[2.5,97.5])))
#now 80% confidence interval thal more_then_or_55 purchase population mean lie ...,

print("80% = ",np.round(np.percentile(store_here,[10,90])))
print("68% = ",np.round(np.percentile(store_here,[16,84])))
```

```
10747
9414.57704935951
95% =  [9323. 9509.]
80% =  [9355. 9474.]
68% =  [9368. 9461.]
```

```
# 95% confidence interval
# less_then_or_17 = [8962., 9196.]
# From_18_25 = [9149., 9235.]
# From_26_35 = [9271., 9330.]
# From_36_45 = [9360., 9444.]
# From_46_50 = [9223., 9350.]
# From_51_55 = [9561., 9704.]
# more_then_or_55= [9321., 9509.]

# 80% confidence interval
# less_then_or_17 = [9004. 9153.]
# From_18_25 = [9162. 9220.]
# From_26_35 = [9281. 9319.]
# From_36_45 = [9376. 9430.]
# From_46_50 = [9245. 9327.]
# From_51_55 = [9586. 9678.]
# more_then_or_55= [9354. 9475.]
```

Age can be one of factors to increase costumers

as the age from 51-55 or more has high purchasing then any other age and age less then 17 has lowest purchasing rate

recomendation : 1. we can increase more product type for age <= 17

2. increase the quantity of product,that are purchase by age group 51 or above

Double-click (or enter) to edit