Business Problem

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

Dataset Fields Decription

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

In [1]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [123…
```
import scipy.stats as st
from scipy.stats import norm
```

1. Checking the structure & characteristics of the dataset.

In [62]:
```
df = pd.read_csv('walmart.csv')
df
```

Out[62]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	M
0	1000001	P00069042	F	0-17	10	A	2	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-	10	A	2	

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	M
				17				
4	1000002	P00285442	M	55+	16	C	4+	
...	...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	1	
550064	1006035	P00375436	F	26-35	1	C	3	
550065	1006036	P00375436	F	26-35	15	B	4+	
550066	1006038	P00375436	F	55+	1	C	2	
550067	1006039	P00371644	F	46-50	0	B	4+	

550068 rows × 10 columns

In [5]:
```
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
```

In [7]:
```
df.shape
```

Out[7]:
```
(550068, 10)
```

- There are total 550068 rows and 10 columns in data.

Value Count for each Column : Showing unique values along with frequency -

In [12]:
```
df['Product_ID'].value_counts()
```

Out[12]:
```
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
```

```
             ...
P00314842      1
P00298842      1
P00231642      1
P00204442      1
P00066342      1
Name: Product_ID, Length: 3631, dtype: int64
```

In [13]:
```
df['Gender'].value_counts()
```

Out[13]:
```
M    414259
F    135809
Name: Gender, dtype: int64
```

In [15]:
```
df['City_Category'].value_counts()
```

Out[15]:
```
B    231173
C    171175
A    147720
Name: City_Category, dtype: int64
```

In [18]:
```
df['Age'].value_counts()
```

Out[18]:
```
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
```

In [16]:
```
df['Stay_In_Current_City_Years'].value_counts()
```

Out[16]:
```
1     193821
2     101838
3      95285
4+     84726
0      74398
Name: Stay_In_Current_City_Years, dtype: int64
```

In [17]:
```
df['Marital_Status'].value_counts()
```

Out[17]:
```
0    324731
1    225337
Name: Marital_Status, dtype: int64
```

In [32]:
```
df['Product_Category'].value_counts()
```

Out[32]:
```
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: Product_Category, dtype: int64
```

In [37]:

```
df['Occupation'].value_counts()
```

Out[37]:

```
4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: Occupation, dtype: int64
```

In [39]:

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

Out[39]:

```
1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
           ... 
1002690       7
1002111       7
1005810       7
1004991       7
1000708       6
Name: User_ID, Length: 5891, dtype: int64
```

In [40]:

```
df['Purchase'].value_counts()
```

Out[40]:

```
7011     191
7193     188
6855     187
6891     184
7012     183
        ... 
23491      1
18345      1
3372       1
```

```
855        1
21489      1
Name: Purchase, Length: 18105, dtype: int64
```

Observation :-

- There are total of 550068 rows and 10 columns
- There are no null values all columns
- User_ID , Occupation, Marital_Status , Product_Category , Purchase are numeric (int) Fields
- Product_ID , Gender , Age ,Stay_In_Current_City_Years , Marital_Status are Object Fields.
- Also shown above the unique values in each column along with their frequecy .

2. Null values & Outliers Detection

In [63]:

```
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
```

- Converting few int column to object based on seeing the unique values in it and logic
- Marital Status has only 2 unique values 0 and 1 so it should be an object data type.
- User_ID should be Object data type as if we treat it as int and then taking out its mean,standard deviation,variance will not convey correct and right info.
- Occupation and Product_Category is also something which must be Object Column and not treated as int

In [64]:

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

In [65]:

```
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  object
 1   Product_ID                  550068 non-null  object
```

```
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  object
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  object
 8   Product_Category            550068 non-null  object
 9   Purchase                    550068 non-null  int64
dtypes: int64(1), object(9)
memory usage: 42.0+ MB
```

In [66]:

```
df.isna().sum()
```

Out[66]:

```
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
```

- There are no NULL values in all columns in given dataset.

In [67]:

```
df.describe(include=object)  # For object type column
```

Out[67]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
count	550068	550068	550068	550068	550068	550068	550068
unique	5891	3631	2	7	21	3	5
top	1001680	P00265242	M	26-35	4	B	1
freq	1026	1880	414259	219587	72308	231173	193821

In [68]:

```
df_describe = df.describe() # For all int type columns
df_describe
```

Out[68]:

	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

In [69]:

```
df_describe.loc['IQR',:]=df_describe.loc['75%',:] - df_describe.loc['25%',:]
df_describe
```

Out[69]:

	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
IQR	6231.000000

In [70]:

```
df_describe.loc['Upper Wisker',:] = df_describe.loc['75%',:] + (1.5* df_describe.loc
df_describe.loc['Lower Wisker',:] = df_describe.loc['25%',:] - (1.5* df_describe.loc
df_describe
```

Out[70]:

	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
IQR	6231.000000
Upper Wisker	21400.500000
Lower Wisker	-3523.500000

- Any value in a column which is greater than (>) then Upper Wisker(UW) or any value in a column which is less than (<) then Lower Wisker(LW) is called 'OUTLIER'
- Mean is sensitive to outliers and median is not sensitive to outliers , so more the outliers in a column the mean is changed more.

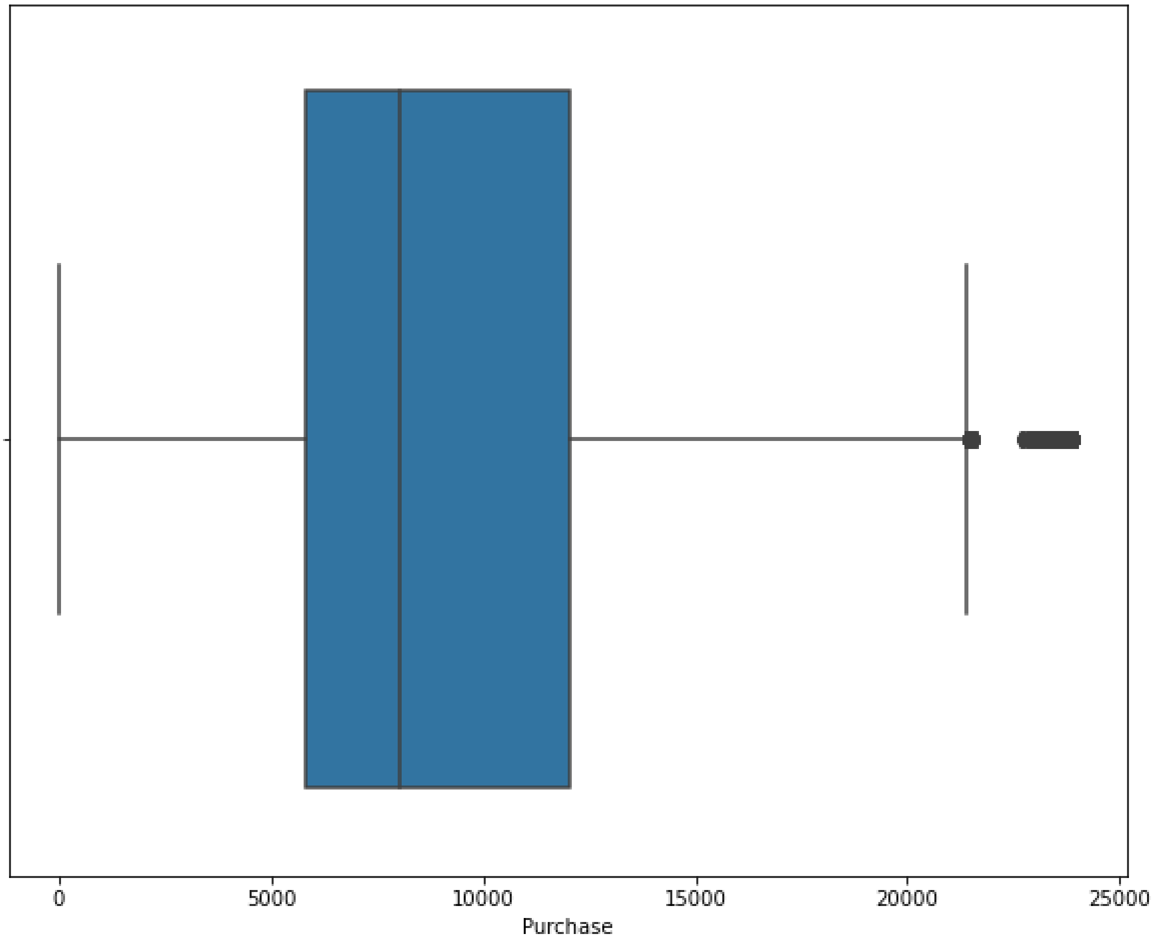
Box Plot

In [75]:

```
num_Purchase_Outliers= df[(df['Purchase'] > df_describe.loc['Upper Wisker','Purchase
print('Total Outliers in Purchase Column = ',num_Purchase_Outliers.shape[0])
```

```
plt.figure(figsize=(10,8))
sns.boxplot(x=df['Purchase'])
plt.show()
```

```
Total Outliers in Purchase Column =  2677
```



3. Data exploration - Male and Female Data

In [78]:

```
df_male = df.loc[df['Gender'] =='M',: ]
df_female = df.loc[df['Gender']=='F',:]
```

In [173…

```
print(df_male.shape)
print(df_female.shape)
```

```
(414259, 10)
(135809, 10)
```

In [84]:

```
# Average Male Expenses
df_male['Purchase'].mean()
```

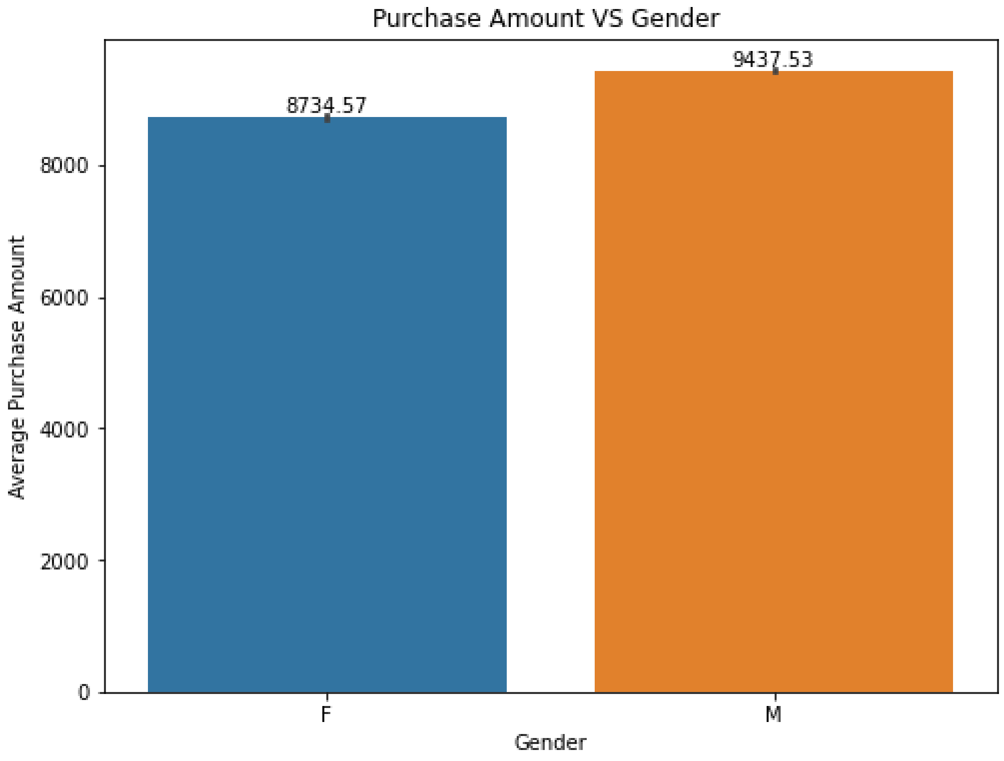
Out[84]: 9437.526040472265

In [85]:

```
# Average Female Expenses
df_female['Purchase'].mean()
```

Out[85]: 8734.565765155476

In [86]:
```
plt.figure(figsize=(8,6))

ax= sns.barplot(data = df , x='Gender' , y="Purchase")

for i in ax.containers:
    ax.bar_label(i)

plt.ylabel('Average Purchase Amount')
plt.xlabel('Gender')

plt.title('Purchase Amount VS Gender ')
plt.show()
```



- Seeing above Bar Plot we can see Average Purchase amount / expenses of Male is higher than Female

In [87]:
```
df_male.describe()
```

Out[87]:

	Purchase
count	414259.00000
mean	9437.52604
std	5092.18621
min	12.00000
25%	5863.00000
50%	8098.00000
75%	12454.00000
max	23961.00000

In [88]:
```
df_female.describe()
```

Out[88]:

	Purchase
count	135809.000000
mean	8734.565765
std	4767.233289
min	12.000000
25%	5433.000000
50%	7914.000000
75%	11400.000000
max	23959.000000

95% Confidence Interval for Male Average Spends

In [134…
```
# Method 1 : Using Formula Directly
norm.interval(alpha = 0.95, loc= df_male['Purchase'].mean() , scale = st.sem(df_male
```

Out[134…
```
(9422.01944736257, 9453.032633581959)
```

In [143…
```
# Method 2 : Bootstrapping Method

def boot_strap_method(data, sample_size,confidence_intterval):
    ans=[]
    for reps in range(sample_size):
        bootstrapped_samples = np.random.choice(data, size=data.shape[0])
        bootstrapped_mean =  np.mean(bootstrapped_samples)
        ans.append(bootstrapped_mean)
    # % CI : [x1,x2]
    x1 = np.percentile(ans,(100-confidence_intterval)/2)
    x2 = np.percentile(ans,confidence_intterval + (100-confidence_intterval)/2)
    return [np.round(x1,2),np.round(x2,2)]
```

In [144…
```
# Boot Strap method with sample size = 100
boot_strap_method(data = df_male['Purchase'], sample_size = 100,confidence_intterval
```

Out[144…
```
[9424.34, 9452.3]
```

In [145…
```
# Boot Strap method with sample size = 500
boot_strap_method(data = df_male['Purchase'], sample_size = 500,confidence_intterval
```

Out[145…
```
[9421.62, 9454.69]
```

In [146…
```
# Boot Strap method with sample size = 1000
boot_strap_method(data = df_male['Purchase'], sample_size = 1000,confidence_intterva
```

Out[146…
```
[9421.83, 9452.67]
```

In [147…

```
# Boot Strap method with sample size = 5000
boot_strap_method(data = df_male['Purchase'], sample_size = 5000,confidence_intterva
```

Out[147…

```
[9421.39, 9452.92]
```

In [148…

```
# Boot Strap method with sample size = 7000
boot_strap_method(data = df_male['Purchase'], sample_size = 7000,confidence_intterva
```

Out[148…

```
[9422.22, 9453.2]
```

In [150…

```
# Boot Strap method with sample size = 10000
boot_strap_method(data = df_male['Purchase'], sample_size = 10000,confidence_intterv
```

Out[150…

```
[9421.66, 9452.96]
```

- In Bootstrap method we observe that as sample size we increase the Confidence Interval becomes more accurate -> Confidence Interval becomes closer to the CI value got directly from norm.interval formula
- 95% CI (Confidence Interval) we means that " there is a 95% chance that the confidence interval [9422,9453] (approx) contains true population mean spend of male.

95% Confidence Interval for Female Average Spends

In [156…

```
# Method 1 : Using Formula Directly
norm.interval(alpha = 0.95, loc= df_female['Purchase'].mean() , scale = st.sem(df_fe
```

Out[156…

```
(8709.21154714068, 8759.919983170272)
```

In [157…

```
# Method 2 : Bootstrapping Method

print("CI using Boot Strap method with sample size = 100  :",
      boot_strap_method(data = df_female['Purchase'], sample_size = 100,confidence_i

print("CI using Boot Strap method with sample size = 500  :",
      boot_strap_method(data = df_female['Purchase'], sample_size = 500,confidence_i

print("CI using Boot Strap method with sample size = 1000  :",
      boot_strap_method(data = df_female['Purchase'], sample_size = 1000,confidence_

print("CI using Boot Strap method with sample size = 5000  :",
      boot_strap_method(data = df_female['Purchase'], sample_size = 5000,confidence_

print("CI using Boot Strap method with sample size = 10000  :",
      boot_strap_method(data = df_female['Purchase'], sample_size = 10000,confidence
```

```
CI using Boot Strap method with sample size = 100  : [8708.63, 8756.94]
CI using Boot Strap method with sample size = 500  : [8712.65, 8759.15]
CI using Boot Strap method with sample size = 1000  : [8708.75, 8761.21]
CI using Boot Strap method with sample size = 5000  : [8709.12, 8760.35]
CI using Boot Strap method with sample size = 10000  : [8709.97, 8759.55]
```

Observation 95%CI :

- 95% CI for Male Average Spends = [9422 , 9453] (Note : Have rounded the 2 decimal places in CI)
- 95% CI for Female Average Spends= [8709 , 8760] (Note : Have rounded the 2 decimal places in CI)
- There is a 95% chance that the confidence interval [9422,9453] contains true population mean spend of male.
- There is a 95% chance that the confidence interval [8709 , 8760] contains true population mean spend of female.
- So from this we can also conclude that male population average spends is more than spends of female as the CI interval of male has higher lower and upper paramters in 95% CI ap compared to femal 95% CI.
- 95% Confidence intervals of average male and female spends are NOT OVERLAPPING

90% Confidence Interval for Male and Female Average Spends

In [159…

```
# Method 1 : Using Formula Directly
print('90% CI for Male Average Spends :',
      norm.interval(alpha = 0.90, loc= df_male['Purchase'].mean() , scale = st.sem(d

print('90% CI for Female Average Spends :',
      norm.interval(alpha = 0.90, loc= df_female['Purchase'].mean() , scale = st.sem
```

```
90% CI for Male Average Spends : (9424.512497305488, 9450.539583639042)
90% CI for Female Average Spends : (8713.287834648021, 8755.84369566293)
```

In [160…

```
# Method 2 : Bootstrapping Method
print("90% CI for Male Average Spends with different sample size :- ")
print("CI using Boot Strap method with sample size = 100  :",
      boot_strap_method(data = df_male['Purchase'], sample_size = 100,confidence_int

print("CI using Boot Strap method with sample size = 500  :",
      boot_strap_method(data = df_male['Purchase'], sample_size = 500,confidence_int

print("CI using Boot Strap method with sample size = 1000  :",
      boot_strap_method(data = df_male['Purchase'], sample_size = 1000,confidence_in

print("CI using Boot Strap method with sample size = 5000  :",
      boot_strap_method(data = df_male['Purchase'], sample_size = 5000,confidence_in

print("CI using Boot Strap method with sample size = 10000  :",
      boot_strap_method(data = df_male['Purchase'], sample_size = 10000,confidence_i


print("\n\n90% CI for Female Average Spends with different sample size :- ")
print("CI using Boot Strap method with sample size = 100  :",
      boot_strap_method(data = df_female['Purchase'], sample_size = 100,confidence_i

print("CI using Boot Strap method with sample size = 500  :",
      boot_strap_method(data = df_female['Purchase'], sample_size = 500,confidence_i

print("CI using Boot Strap method with sample size = 1000  :",
      boot_strap_method(data = df_female['Purchase'], sample_size = 1000,confidence_

print("CI using Boot Strap method with sample size = 5000  :",
      boot_strap_method(data = df_female['Purchase'], sample_size = 5000,confidence_

print("CI using Boot Strap method with sample size = 10000  :",
      boot_strap_method(data = df_female['Purchase'], sample_size = 10000,confidence
```

```
90% CI for Male Average Spends with different sample size :-
CI using Boot Strap method with sample size = 100  : [9423.92, 9453.59]
CI using Boot Strap method with sample size = 500  : [9426.4, 9451.44]
CI using Boot Strap method with sample size = 1000  : [9424.9, 9450.92]
CI using Boot Strap method with sample size = 5000  : [9424.44, 9450.44]
CI using Boot Strap method with sample size = 10000  : [9424.37, 9450.36]


90% CI for Female Average Spends with different sample size :-
CI using Boot Strap method with sample size = 100  : [8706.26, 8755.78]
CI using Boot Strap method with sample size = 500  : [8715.96, 8755.27]
CI using Boot Strap method with sample size = 1000  : [8712.74, 8755.01]
CI using Boot Strap method with sample size = 5000  : [8713.35, 8755.72]
CI using Boot Strap method with sample size = 10000  : [8712.98, 8756.01]
```

Observation 90%CI :

- 90% CI for Male Average Spends = [9424 , 9450] (Note : Have rounded the 2 decimal places in CI)
- 90% CI for Female Average Spends= [8713 , 8756] (Note : Have rounded the 2 decimal places in CI)
- There is a 90% chance that the confidence interval [9424 , 9450] contains true population mean spend of male.
- There is a 90% chance that the confidence interval [8713 , 8756] contains true population mean spend of female.
- So from this we can also conclude that male population average spends is more than spends of female as the CI interval of male has higher lower and upper paramters in 90% CI ap compared to femal 90% CI.
- 90% Confidence intervals of average male and female spends are NOT OVERLAPPING

99% Confidence Interval for Male and Female Average Spends

In [161…

```
# Method 1 : Using Formula Directly
print('90% CI for Male Average Spends :',
      norm.interval(alpha = 0.99, loc= df_male['Purchase'].mean() , scale = st.sem(d

print('90% CI for Female Average Spends :',
      norm.interval(alpha = 0.99, loc= df_female['Purchase'].mean() , scale = st.sem
```

```
90% CI for Male Average Spends : (9417.146922669479, 9457.90515827505)
90% CI for Female Average Spends : (8701.244674438389, 8767.886855872563)
```

In [162…

```
# Method 2 : Bootstrapping Method
print("99% CI for Male Average Spends with different sample size :- ")
print("CI using Boot Strap method with sample size = 100  :",
      boot_strap_method(data = df_male['Purchase'], sample_size = 100,confidence_int

print("CI using Boot Strap method with sample size = 500  :",
      boot_strap_method(data = df_male['Purchase'], sample_size = 500,confidence_int

print("CI using Boot Strap method with sample size = 1000  :",
      boot_strap_method(data = df_male['Purchase'], sample_size = 1000,confidence_in

print("CI using Boot Strap method with sample size = 5000  :",
      boot_strap_method(data = df_male['Purchase'], sample_size = 5000,confidence_in

print("CI using Boot Strap method with sample size = 10000  :",
```

```
    boot_strap_method(data = df_male['Purchase'], sample_size = 10000,confidence_i


print("\n\n99% CI for Female Average Spends with different sample size :- ")
print("CI using Boot Strap method with sample size = 100  :",
      boot_strap_method(data = df_female['Purchase'], sample_size = 100,confidence_i

print("CI using Boot Strap method with sample size = 500  :",
      boot_strap_method(data = df_female['Purchase'], sample_size = 500,confidence_i

print("CI using Boot Strap method with sample size = 1000  :",
      boot_strap_method(data = df_female['Purchase'], sample_size = 1000,confidence_

print("CI using Boot Strap method with sample size = 5000  :",
      boot_strap_method(data = df_female['Purchase'], sample_size = 5000,confidence_

print("CI using Boot Strap method with sample size = 10000  :",
      boot_strap_method(data = df_female['Purchase'], sample_size = 10000,confidence
```

```
99% CI for Male Average Spends with different sample size :-
CI using Boot Strap method with sample size = 100  : [9419.24, 9454.78]
CI using Boot Strap method with sample size = 500  : [9418.06, 9457.31]
CI using Boot Strap method with sample size = 1000  : [9416.72, 9456.15]
CI using Boot Strap method with sample size = 5000  : [9417.29, 9458.19]
CI using Boot Strap method with sample size = 10000  : [9417.39, 9457.23]


99% CI for Female Average Spends with different sample size :-
CI using Boot Strap method with sample size = 100  : [8705.89, 8761.63]
CI using Boot Strap method with sample size = 500  : [8704.29, 8770.16]
CI using Boot Strap method with sample size = 1000  : [8703.23, 8768.87]
CI using Boot Strap method with sample size = 5000  : [8702.54, 8767.57]
CI using Boot Strap method with sample size = 10000  : [8701.79, 8767.31]
```

Observation 99%CI :

- 99% CI for Male Average Spends = [9417 , 9458] (Note : Have rounded the 2 decimal places in CI)
- 99% CI for Female Average Spends= [8701 , 8768] (Note : Have rounded the 2 decimal places in CI)
- There is a 99% chance that the confidence interval [9417 , 9458] contains true population mean spend of male.
- There is a 99% chance that the confidence interval [8701 , 8768] contains true population mean spend of female.
- So from this we can also conclude that male population average spends is more than spends of female as the CI interval of male has higher lower and upper paramters in 99% CI ap compared to femal 99% CI.
- 99% Confidence intervals of average male and female spends are NOT OVERLAPPING

4. Data exploration - Marital Status Data

In [163…

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

Out[163…

```
0    324731
1    225337
Name: Marital_Status, dtype: int64
```

In [170…

```
# Marital Status Column has value 0 and 1 . Considering 0 as 'Single' / 'Unmarried'
# 1 as 'Partnered'/'Married' Marital Status
df_single    = df.loc[df['Marital_Status'] ==  0,: ]
df_partnered = df.loc[df['Marital_Status'] ==  1,: ]
```

In [171…

```
df_single.shape
```

Out[171…

```
(324731, 10)
```

In [172…

```
df_partnered.shape
```

Out[172…

```
(225337, 10)
```

In [174…

```
# Average Single Marital Status Expense
df_single['Purchase'].mean()
```

Out[174…

```
9265.907618921507
```

In [175…

```
# Average Partnered Marital Status Expense
df_partnered['Purchase'].mean()
```

Out[175…

```
9261.174574082374
```

90% Confidence Interval for Single and Partnered Marital Status Spends

In [176…

```
# Method 1 : Using Formula Directly
print('90% CI for Single Marital Status Average Spends :',
      norm.interval(alpha = 0.90, loc= df_single['Purchase'].mean() , scale = st.sem

print('90% CI for Partnered Marital Status Average Spends :',
      norm.interval(alpha = 0.90, loc= df_partnered['Purchase'].mean() , scale = st.
```

```
90% CI for Single Marital Status Average Spends : (9251.396385823671, 9280.418852019
342)
90% CI for Partnered Marital Status Average Spends : (9243.790713903045, 9278.558434
261702)
```

In [177…

```
# Method 2 : Bootstrapping Method
print("90% CI for Single Marital Status Average Spends with different sample size :-
print("CI using Boot Strap method with sample size = 100  :",
      boot_strap_method(data = df_single['Purchase'], sample_size = 100,confidence_i

print("CI using Boot Strap method with sample size = 500  :",
      boot_strap_method(data = df_single['Purchase'], sample_size = 500,confidence_i

print("CI using Boot Strap method with sample size = 1000  :",
      boot_strap_method(data = df_single['Purchase'], sample_size = 1000,confidence_

print("CI using Boot Strap method with sample size = 5000  :",
      boot_strap_method(data = df_single['Purchase'], sample_size = 5000,confidence_

print("CI using Boot Strap method with sample size = 10000  :",
      boot_strap_method(data = df_single['Purchase'], sample_size = 10000,confidence
```

```
print("\n\n90% CI for Partnered Marital Status Average Spends with different sample
print("CI using Boot Strap method with sample size = 100  :",
      boot_strap_method(data = df_partnered['Purchase'], sample_size = 100,confidenc

print("CI using Boot Strap method with sample size = 500  :",
      boot_strap_method(data = df_partnered['Purchase'], sample_size = 500,confidenc

print("CI using Boot Strap method with sample size = 1000  :",
      boot_strap_method(data = df_partnered['Purchase'], sample_size = 1000,confiden

print("CI using Boot Strap method with sample size = 5000  :",
      boot_strap_method(data = df_partnered['Purchase'], sample_size = 5000,confiden

print("CI using Boot Strap method with sample size = 10000  :",
      boot_strap_method(data = df_partnered['Purchase'], sample_size = 10000,confide
```

```
90% CI for Single Marital Status Average Spends with different sample size :-
CI using Boot Strap method with sample size = 100  : [9251.89, 9280.59]
CI using Boot Strap method with sample size = 500  : [9252.42, 9281.16]
CI using Boot Strap method with sample size = 1000  : [9250.21, 9280.46]
CI using Boot Strap method with sample size = 5000  : [9251.53, 9279.98]
CI using Boot Strap method with sample size = 10000  : [9251.64, 9280.75]


90% CI for Partnered Marital Status Average Spends with different sample size :-
CI using Boot Strap method with sample size = 100  : [9243.91, 9275.97]
CI using Boot Strap method with sample size = 500  : [9243.75, 9278.05]
CI using Boot Strap method with sample size = 1000  : [9244.08, 9278.96]
CI using Boot Strap method with sample size = 5000  : [9244.24, 9278.91]
CI using Boot Strap method with sample size = 10000  : [9244.2, 9278.32]
```

Observation 90%CI :

- 90% CI for Single Status Average Spends = [9252 , 9281] (Note : Have rounded the 2 decimal places in CI)
- 90% CI for Partnered Average Spends= [9244, 9278] (Note : Have rounded the 2 decimal places in CI)
- There is a 90% chance that the confidence interval [9252 , 9281] contains true population mean spend of Single Marital Status people.
- There is a 90% chance that the confidence interval [9244, 9278]contains true population mean spend of Partnered Marital Status people.
- So from this we can also conclude that Single Marital Status population average spends is slightly more than spends of Partnered Marital Status people as the CI interval of Single status people has higher lower and upper paramters in 90% CI ap compared to Partnered Marital Status 90% CI.
- 90% Confidence intervals of average spends of Single and Partnered marital status people ARE OVERLAPPING in average spends range of (9252,9278)

95% Confidence Interval for Single and Partnered Marital Status Spends

In [178…

```
# Method 1 : Using Formula Directly
print('95% CI for Single Marital Status Average Spends :',
      norm.interval(alpha = 0.95, loc= df_single['Purchase'].mean() , scale = st.sem
```

```
print('95% CI for Partnered Marital Status Average Spends :',
      norm.interval(alpha = 0.95, loc= df_partnered['Purchase'].mean() , scale = st.
```

```
95% CI for Single Marital Status Average Spends : (9248.61641818668, 9283.198819656332)
95% CI for Partnered Marital Status Average Spends : (9240.460427057078, 9281.888721107669)
```

In [179…

```
# Method 2 : Bootstrapping Method
print("95% CI for Single Marital Status Average Spends with different sample size :-
print("CI using Boot Strap method with sample size = 100  :",
      boot_strap_method(data = df_single['Purchase'], sample_size = 100,confidence_i

print("CI using Boot Strap method with sample size = 500  :",
      boot_strap_method(data = df_single['Purchase'], sample_size = 500,confidence_i

print("CI using Boot Strap method with sample size = 1000  :",
      boot_strap_method(data = df_single['Purchase'], sample_size = 1000,confidence_

print("CI using Boot Strap method with sample size = 5000  :",
      boot_strap_method(data = df_single['Purchase'], sample_size = 5000,confidence_

print("CI using Boot Strap method with sample size = 10000  :",
      boot_strap_method(data = df_single['Purchase'], sample_size = 10000,confidence


print("\n\n95% CI for Partnered Marital Status Average Spends with different sample
print("CI using Boot Strap method with sample size = 100  :",
      boot_strap_method(data = df_partnered['Purchase'], sample_size = 100,confidenc

print("CI using Boot Strap method with sample size = 500  :",
      boot_strap_method(data = df_partnered['Purchase'], sample_size = 500,confidenc

print("CI using Boot Strap method with sample size = 1000  :",
      boot_strap_method(data = df_partnered['Purchase'], sample_size = 1000,confiden

print("CI using Boot Strap method with sample size = 5000  :",
      boot_strap_method(data = df_partnered['Purchase'], sample_size = 5000,confiden

print("CI using Boot Strap method with sample size = 10000  :",
      boot_strap_method(data = df_partnered['Purchase'], sample_size = 10000,confide
```

```
95% CI for Single Marital Status Average Spends with different sample size :-
CI using Boot Strap method with sample size = 100  : [9247.18, 9281.76]
CI using Boot Strap method with sample size = 500  : [9249.14, 9283.17]
CI using Boot Strap method with sample size = 1000  : [9247.05, 9283.08]
CI using Boot Strap method with sample size = 5000  : [9248.85, 9283.2]
CI using Boot Strap method with sample size = 10000  : [9248.54, 9283.23]


95% CI for Partnered Marital Status Average Spends with different sample size :-
CI using Boot Strap method with sample size = 100  : [9240.46, 9280.06]
CI using Boot Strap method with sample size = 500  : [9240.75, 9280.7]
CI using Boot Strap method with sample size = 1000  : [9241.48, 9281.05]
CI using Boot Strap method with sample size = 5000  : [9240.63, 9282.22]
CI using Boot Strap method with sample size = 10000  : [9240.56, 9282.0]
```

Observation 95%CI :

- 95% CI for Single Status Average Spends = [9249 , 9283] (Note : Have rounded the 2 decimal places in CI)

- 95% CI for Partnered Average Spends= [9240, 9282] (Note : Have rounded the 2 decimal places in CI)
- There is a 95% chance that the confidence interval [9249 , 9283] contains true population mean spend of Single Marital Status customer.
- There is a 95% chance that the confidence interval [9240, 9282] contains true population mean spend of Partnered Marital Status customer.
- So from this we can also conclude that Single Marital Status population average spends is slightly more than spends of Partnered Marital Status people as the CI interval of Single status people has higher lower and upper paramters in 95% CI ap compared to Partnered Marital Status 95% CI.
- 95% Confidence intervals of average spends of Single and Partnered marital status customers ARE OVERLAPPING in average spends range of (9249,9282)

99% Confidence Interval for Single and Partnered Marital Status Spends

In [180…

```
# Method 1 : Using Formula Directly
print('99% CI for Single Marital Status Average Spends :',
      norm.interval(alpha = 0.99, loc= df_single['Purchase'].mean() , scale = st.sem

print('99% CI for Partnered Marital Status Average Spends :',
      norm.interval(alpha = 0.99, loc= df_partnered['Purchase'].mean() , scale = st.
```

```
99% CI for Single Marital Status Average Spends : (9243.183129136169, 9288.632108706
845)
99% CI for Partnered Marital Status Average Spends : (9233.951570329937, 9288.397577
83481)
```

In [181…

```
# Method 2 : Bootstrapping Method
print("99% CI for Single Marital Status Average Spends with different sample size :-
print("CI using Boot Strap method with sample size = 100  :",
      boot_strap_method(data = df_single['Purchase'], sample_size = 100,confidence_i

print("CI using Boot Strap method with sample size = 500  :",
      boot_strap_method(data = df_single['Purchase'], sample_size = 500,confidence_i

print("CI using Boot Strap method with sample size = 1000  :",
      boot_strap_method(data = df_single['Purchase'], sample_size = 1000,confidence_

print("CI using Boot Strap method with sample size = 5000  :",
      boot_strap_method(data = df_single['Purchase'], sample_size = 5000,confidence_

print("CI using Boot Strap method with sample size = 10000  :",
      boot_strap_method(data = df_single['Purchase'], sample_size = 10000,confidence


print("\n\n99% CI for Partnered Marital Status Average Spends with different sample
print("CI using Boot Strap method with sample size = 100  :",
      boot_strap_method(data = df_partnered['Purchase'], sample_size = 100,confidenc

print("CI using Boot Strap method with sample size = 500  :",
      boot_strap_method(data = df_partnered['Purchase'], sample_size = 500,confidenc

print("CI using Boot Strap method with sample size = 1000  :",
      boot_strap_method(data = df_partnered['Purchase'], sample_size = 1000,confiden

print("CI using Boot Strap method with sample size = 5000  :",
```

```
      boot_strap_method(data = df_partnered['Purchase'], sample_size = 5000,confiden

print("CI using Boot Strap method with sample size = 10000  :",
      boot_strap_method(data = df_partnered['Purchase'], sample_size = 10000,confide
```

```
99% CI for Single Marital Status Average Spends with different sample size :-
CI using Boot Strap method with sample size = 100  : [9250.14, 9282.26]
CI using Boot Strap method with sample size = 500  : [9246.4, 9288.11]
CI using Boot Strap method with sample size = 1000  : [9242.02, 9288.41]
CI using Boot Strap method with sample size = 5000  : [9242.8, 9288.71]
CI using Boot Strap method with sample size = 10000  : [9243.35, 9287.93]


99% CI for Partnered Marital Status Average Spends with different sample size :-
CI using Boot Strap method with sample size = 100  : [9240.53, 9288.19]
CI using Boot Strap method with sample size = 500  : [9233.18, 9287.64]
CI using Boot Strap method with sample size = 1000  : [9233.05, 9287.35]
CI using Boot Strap method with sample size = 5000  : [9234.43, 9288.47]
CI using Boot Strap method with sample size = 10000  : [9233.7, 9289.98]
```

Observation 99%CI :

- 99% CI for Single Status Average Spends = [9243 , 9288] (Note : Have rounded the 2 decimal places in CI)
- 99% CI for Partnered Average Spends= [9234, 9289] (Note : Have rounded the 2 decimal places in CI)
- There is a 99% chance that the confidence interval [9243 , 9288] contains true population mean spend of Single Marital Status customer.
- There is a 99% chance that the confidence interval [9234, 9289] contains true population mean spend of Partnered Marital Status customer.
- So from this we can also conclude that Single Marital Status population average spends is slightly more than spends of Partnered Marital Status people as the CI interval of Single status people has higher lower paramters in 99% CI ap compared to Partnered Marital Status 99% CI.
- 99% Confidence intervals of average spends of Single and Partnered marital status customers ARE OVERLAPPING in average spends range of (9243,9288)

5. Data exploration - Age

In [198…

```
# There are 7 age groups / categories in Age column
df['Age'].value_counts()
```

Out[198…

```
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
```

In [199…

```
df_26_to_35    = df.loc[df['Age'] ==  '26-35',: ]
df_36_to_45    = df.loc[df['Age'] ==  '36-45',: ]
df_18_to_25    = df.loc[df['Age'] ==  '18-25',: ]
df_46_to_50    = df.loc[df['Age'] ==  '46-50',: ]
df_51_to_55    = df.loc[df['Age'] ==  '51-55',: ]
```

```
df_55_plus      = df.loc[df['Age'] ==  '55+',: ]
df_0_to_17      = df.loc[df['Age'] ==  '0-17',: ]
```

In [202…

```
print('Mean =', df_26_to_35['Purchase'].mean())
df_26_to_35.shape
```

```
Mean = 9252.690632869888
```

Out[202…

```
(219587, 10)
```

In [203…

```
print('Mean =', df_36_to_45['Purchase'].mean())
df_36_to_45.shape
```

```
Mean = 9331.350694917874
```

Out[203…

```
(110013, 10)
```

In [204…

```
print('Mean =', df_18_to_25['Purchase'].mean())
df_18_to_25.shape
```

```
Mean = 9169.663606261289
```

Out[204…

```
(99660, 10)
```

In [205…

```
print('Mean =', df_46_to_50['Purchase'].mean())
df_46_to_50.shape
```

```
Mean = 9208.625697468327
```

Out[205…

```
(45701, 10)
```

In [206…

```
print('Mean =', df_51_to_55['Purchase'].mean())
df_51_to_55.shape
```

```
Mean = 9534.808030960236
```

Out[206…

```
(38501, 10)
```

In [207…

```
print('Mean =', df_55_plus['Purchase'].mean())
df_55_plus.shape
```

```
Mean = 9336.280459449405
```

Out[207…

```
(21504, 10)
```

In [208…

```
print('Mean =', df_0_to_17['Purchase'].mean())
df_0_to_17.shape
```

```
Mean = 8933.464640444974
```

Out[208…

```
(15102, 10)
```

90% Confidence Interval for each Age Group average Spends

In [220…

```
# Method 1 : Using Formula Directly
age_groups = df['Age'].value_counts().index
for i in age_groups:
    data = df.loc[df['Age'] == i,'Purchase']
    print('90% CI for Age Group '+i+' Average Spends:',
          norm.interval(alpha = 0.90, loc= data.mean() , scale = st.sem(data)) )
```

```
90% CI for Age Group 26-35 Average Spends: (9235.103000581124, 9270.278265158651)
90% CI for Age Group 36-45 Average Spends: (9306.441376202305, 9356.260013633442)
90% CI for Age Group 18-25 Average Spends: (9143.433031607847, 9195.89418091473)
90% CI for Age Group 46-50 Average Spends: (9170.406859081895, 9246.84453585476)
90% CI for Age Group 51-55 Average Spends: (9492.161430973249, 9577.454630947223)
90% CI for Age Group 55+ Average Spends: (9280.067707714425, 9392.493211184385)
90% CI for Age Group 0-17 Average Spends: (8865.053694527898, 9001.87558636205)
```

In [226…

```python
# Method 2 : Bootstrapping Method
age_groups = df['Age'].value_counts().index
for i in age_groups:
    data_ = df.loc[df['Age'] == i,'Purchase']
    print("For Age Group ",i)
    print("CI using Boot Strap method with sample size = 100  :",
      boot_strap_method(data = data_, sample_size = 100,confidence_intterval = 90) )
    print("CI using Boot Strap method with sample size = 500  :",
      boot_strap_method(data = data_, sample_size = 500,confidence_intterval = 90) )
    print("CI using Boot Strap method with sample size = 1000  :",
      boot_strap_method(data = data_, sample_size = 1000,confidence_intterval = 90)
    print("CI using Boot Strap method with sample size = 5000  :",
      boot_strap_method(data = data_, sample_size = 5000,confidence_intterval = 90)
    print("CI using Boot Strap method with sample size = 7000  :",
      boot_strap_method(data = data_, sample_size = 7000,confidence_intterval = 90)
    print("CI using Boot Strap method with sample size = 10000  :",
      boot_strap_method(data = data_, sample_size = 10000,confidence_intterval = 90)
    print("\n")
```

```
For Age Group  26-35
CI using Boot Strap method with sample size = 100  : [9231.39, 9268.22]
CI using Boot Strap method with sample size = 500  : [9235.02, 9268.47]
CI using Boot Strap method with sample size = 1000  : [9235.32, 9268.52]
CI using Boot Strap method with sample size = 5000  : [9235.25, 9270.7]
CI using Boot Strap method with sample size = 7000  : [9235.29, 9270.36]
CI using Boot Strap method with sample size = 10000  : [9234.97, 9270.49]


For Age Group  36-45
CI using Boot Strap method with sample size = 100  : [9307.14, 9355.11]
CI using Boot Strap method with sample size = 500  : [9309.79, 9356.77]
CI using Boot Strap method with sample size = 1000  : [9305.9, 9357.27]
CI using Boot Strap method with sample size = 5000  : [9306.06, 9356.93]
CI using Boot Strap method with sample size = 7000  : [9306.26, 9355.6]
CI using Boot Strap method with sample size = 10000  : [9306.51, 9356.36]


For Age Group  18-25
CI using Boot Strap method with sample size = 100  : [9141.64, 9194.06]
CI using Boot Strap method with sample size = 500  : [9143.31, 9196.72]
CI using Boot Strap method with sample size = 1000  : [9143.93, 9196.02]
CI using Boot Strap method with sample size = 5000  : [9144.27, 9195.68]
CI using Boot Strap method with sample size = 7000  : [9143.62, 9195.63]
CI using Boot Strap method with sample size = 10000  : [9143.37, 9195.11]


For Age Group  46-50
CI using Boot Strap method with sample size = 100  : [9175.01, 9241.67]
CI using Boot Strap method with sample size = 500  : [9172.36, 9246.86]
CI using Boot Strap method with sample size = 1000  : [9171.44, 9248.04]
CI using Boot Strap method with sample size = 5000  : [9170.19, 9246.77]
CI using Boot Strap method with sample size = 7000  : [9171.75, 9246.55]
CI using Boot Strap method with sample size = 10000  : [9170.87, 9246.68]
```

```
For Age Group  51-55
CI using Boot Strap method with sample size = 100  : [9496.91, 9567.92]
CI using Boot Strap method with sample size = 500  : [9495.48, 9578.64]
CI using Boot Strap method with sample size = 1000  : [9491.25, 9577.64]
CI using Boot Strap method with sample size = 5000  : [9492.11, 9577.47]
CI using Boot Strap method with sample size = 7000  : [9491.72, 9577.1]
CI using Boot Strap method with sample size = 10000  : [9492.5, 9577.33]


For Age Group  55+
CI using Boot Strap method with sample size = 100  : [9284.32, 9383.48]
CI using Boot Strap method with sample size = 500  : [9285.85, 9390.17]
CI using Boot Strap method with sample size = 1000  : [9282.18, 9387.38]
CI using Boot Strap method with sample size = 5000  : [9280.31, 9391.82]
CI using Boot Strap method with sample size = 7000  : [9279.18, 9392.74]
CI using Boot Strap method with sample size = 10000  : [9280.96, 9392.97]


For Age Group  0-17
CI using Boot Strap method with sample size = 100  : [8877.27, 9006.91]
CI using Boot Strap method with sample size = 500  : [8866.79, 8998.4]
CI using Boot Strap method with sample size = 1000  : [8865.82, 8999.13]
CI using Boot Strap method with sample size = 5000  : [8864.75, 9003.53]
CI using Boot Strap method with sample size = 7000  : [8864.63, 9001.57]
CI using Boot Strap method with sample size = 10000  : [8864.27, 9001.44]
```

Observation :-

- 90% CI for Age Group 26-35 Average Spends: (9235, 9270)
- 90% CI for Age Group 36-45 Average Spends: (9306, 9356)
- 90% CI for Age Group 18-25 Average Spends: (9143, 9195)
- 90% CI for Age Group 46-50 Average Spends: (9170, 9246)
- 90% CI for Age Group 51-55 Average Spends: (9492, 9577)
- 90% CI for Age Group 55+ Average Spends: (9280, 9392)
- 90% CI for Age Group 0-17 Average Spends: (8864, 9001)
- 90% Confidence Interval are overlapping for Age Group 18-25 and 46-50 with average spend in range (9170,9195). And there is overlapping for Age group 36-45 and 55+ with average spend in range (9306,9356) .

95% Confidence Interval for each Age Group average Spends

In [230…

```python
# Method 1 : Using Formula Directly
age_groups = df['Age'].value_counts().index
for i in age_groups:
    data = df.loc[df['Age'] == i,'Purchase']
    print('95% CI for Age Group '+i+' Average Spends:',
          norm.interval(alpha = 0.95, loc= data.mean() , scale = st.sem(data)) )
```

```
95% CI for Age Group 26-35 Average Spends: (9231.733676400028, 9273.647589339747)
95% CI for Age Group 36-45 Average Spends: (9301.669410965314, 9361.031978870433)
95% CI for Age Group 18-25 Average Spends: (9138.407948753442, 9200.919263769136)
95% CI for Age Group 46-50 Average Spends: (9163.085142648752, 9254.166252287903)
95% CI for Age Group 51-55 Average Spends: (9483.991472776577, 9585.624589143894)
95% CI for Age Group 55+ Average Spends: (9269.29883441773, 9403.262084481079)
95% CI for Age Group 0-17 Average Spends: (8851.947970542686, 9014.981310347262)
```

In [231…

```
# Method 2 : Bootstrapping Method
age_groups = df['Age'].value_counts().index
for i in age_groups:
    data_ = df.loc[df['Age'] == i,'Purchase']
    print("For Age Group ",i)
    print("95% CI using Boot Strap method with sample size = 100  :",
      boot_strap_method(data = data_, sample_size = 100,confidence_intterval = 95) )
    print("95% CI using Boot Strap method with sample size = 500  :",
      boot_strap_method(data = data_, sample_size = 500,confidence_intterval = 95) )
    print("95% CI using Boot Strap method with sample size = 1000  :",
      boot_strap_method(data = data_, sample_size = 1000,confidence_intterval = 95)
    print("95% CI using Boot Strap method with sample size = 5000  :",
      boot_strap_method(data = data_, sample_size = 5000,confidence_intterval = 95)
    print("95% CI using Boot Strap method with sample size = 7000  :",
      boot_strap_method(data = data_, sample_size = 7000,confidence_intterval = 95)
    print("95% CI using Boot Strap method with sample size = 10000  :",
      boot_strap_method(data = data_, sample_size = 10000,confidence_intterval = 95)
    print("\n")
```

```
For Age Group  26-35
95% CI using Boot Strap method with sample size = 100  : [9232.23, 9274.07]
95% CI using Boot Strap method with sample size = 500  : [9234.22, 9275.7]
95% CI using Boot Strap method with sample size = 1000  : [9231.67, 9274.42]
95% CI using Boot Strap method with sample size = 5000  : [9231.26, 9273.6]
95% CI using Boot Strap method with sample size = 7000  : [9232.08, 9273.69]
95% CI using Boot Strap method with sample size = 10000  : [9231.51, 9273.74]


For Age Group  36-45
95% CI using Boot Strap method with sample size = 100  : [9300.83, 9357.36]
95% CI using Boot Strap method with sample size = 500  : [9299.89, 9361.63]
95% CI using Boot Strap method with sample size = 1000  : [9302.33, 9359.72]
95% CI using Boot Strap method with sample size = 5000  : [9301.49, 9360.75]
95% CI using Boot Strap method with sample size = 7000  : [9301.65, 9361.03]
95% CI using Boot Strap method with sample size = 10000  : [9301.26, 9361.05]


For Age Group  18-25
95% CI using Boot Strap method with sample size = 100  : [9138.73, 9194.78]
95% CI using Boot Strap method with sample size = 500  : [9135.87, 9200.74]
95% CI using Boot Strap method with sample size = 1000  : [9140.05, 9199.04]
95% CI using Boot Strap method with sample size = 5000  : [9138.95, 9201.41]
95% CI using Boot Strap method with sample size = 7000  : [9137.51, 9201.63]
95% CI using Boot Strap method with sample size = 10000  : [9138.45, 9200.73]


For Age Group  46-50
95% CI using Boot Strap method with sample size = 100  : [9163.19, 9256.07]
95% CI using Boot Strap method with sample size = 500  : [9164.15, 9250.11]
95% CI using Boot Strap method with sample size = 1000  : [9168.51, 9254.71]
95% CI using Boot Strap method with sample size = 5000  : [9162.78, 9255.36]
95% CI using Boot Strap method with sample size = 7000  : [9162.9, 9254.56]
95% CI using Boot Strap method with sample size = 10000  : [9163.73, 9253.96]


For Age Group  51-55
95% CI using Boot Strap method with sample size = 100  : [9487.54, 9583.04]
95% CI using Boot Strap method with sample size = 500  : [9484.82, 9587.55]
95% CI using Boot Strap method with sample size = 1000  : [9486.49, 9587.67]
95% CI using Boot Strap method with sample size = 5000  : [9487.12, 9586.07]
95% CI using Boot Strap method with sample size = 7000  : [9484.34, 9586.44]
95% CI using Boot Strap method with sample size = 10000  : [9484.09, 9585.28]
```

```
For Age Group  55+
95% CI using Boot Strap method with sample size = 100  : [9268.23, 9403.35]
95% CI using Boot Strap method with sample size = 500  : [9270.31, 9406.97]
95% CI using Boot Strap method with sample size = 1000  : [9266.62, 9397.5]
95% CI using Boot Strap method with sample size = 5000  : [9269.71, 9402.36]
95% CI using Boot Strap method with sample size = 7000  : [9269.16, 9403.5]
95% CI using Boot Strap method with sample size = 10000  : [9270.91, 9404.45]


For Age Group  0-17
95% CI using Boot Strap method with sample size = 100  : [8859.4, 8995.39]
95% CI using Boot Strap method with sample size = 500  : [8850.33, 9012.7]
95% CI using Boot Strap method with sample size = 1000  : [8850.92, 9018.33]
95% CI using Boot Strap method with sample size = 5000  : [8852.38, 9012.63]
95% CI using Boot Strap method with sample size = 7000  : [8852.65, 9014.09]
95% CI using Boot Strap method with sample size = 10000  : [8851.58, 9014.19]
```

Observation :-

- 95% CI for Age Group 26-35 Average Spends: (9231, 9273)
- 95% CI for Age Group 36-45 Average Spends: (9301, 9361)
- 95% CI for Age Group 18-25 Average Spends: (9138, 9200)
- 95% CI for Age Group 46-50 Average Spends: (9163, 9254)
- 95% CI for Age Group 51-55 Average Spends: (9483, 9585)
- 95% CI for Age Group 55+ Average Spends: (9269, 9403)
- 95% CI for Age Group 0-17 Average Spends: (8851, 9014)
- 95% Confidence Interval are overlapping for Age Group 18-25 and 46-50 with average spend in range (9163,9200) . Also overlapping for Age group 26-35 and 55+ with average spend in range (9269,9273) . And there is overlapping for Age group 36-45 and 55+ with average spend in range (9301,9361) .

99% Confidence Interval for each Age Group average Spends

In [232…

```
# Method 1 : Using Formula Directly
age_groups = df['Age'].value_counts().index
for i in age_groups:
    data = df.loc[df['Age'] == i,'Purchase']
    print('99% CI for Age Group '+i+' Average Spends:',
          norm.interval(alpha = 0.99, loc= data.mean() , scale = st.sem(data)) )
```

```
99% CI for Age Group 26-35 Average Spends: (9225.148523415806, 9280.23274232397)
99% CI for Age Group 36-45 Average Spends: (9292.342875603326, 9370.358514232421)
99% CI for Age Group 18-25 Average Spends: (9128.586709366526, 9210.740503156052)
99% CI for Age Group 46-50 Average Spends: (9148.775263210646, 9268.476131726009)
99% CI for Age Group 51-55 Average Spends: (9468.02375292888, 9601.59230899159)
99% CI for Age Group 55+ Average Spends: (9248.251682432667, 9424.309236466142)
99% CI for Age Group 0-17 Average Spends: (8826.333576446717, 9040.59570444323)
```

In [233…

```
# Method 2 : Bootstrapping Method
age_groups = df['Age'].value_counts().index
for i in age_groups:
    data_ = df.loc[df['Age'] == i,'Purchase']
    print("For Age Group ",i)
    print("99% CI using Boot Strap method with sample size = 100  :",
      boot_strap_method(data = data_, sample_size = 100,confidence_intterval = 99) )
```

```
    print("99% CI using Boot Strap method with sample size = 500  :",
      boot_strap_method(data = data_, sample_size = 500,confidence_intterval = 99) )
    print("99% CI using Boot Strap method with sample size = 1000  :",
      boot_strap_method(data = data_, sample_size = 1000,confidence_intterval = 99)
    print("99% CI using Boot Strap method with sample size = 5000  :",
      boot_strap_method(data = data_, sample_size = 5000,confidence_intterval = 99)
    print("99% CI using Boot Strap method with sample size = 7000  :",
      boot_strap_method(data = data_, sample_size = 7000,confidence_intterval = 99)
    print("99% CI using Boot Strap method with sample size = 10000  :",
      boot_strap_method(data = data_, sample_size = 10000,confidence_intterval = 99)
    print("\n")
```

```
For Age Group  26-35
99% CI using Boot Strap method with sample size = 100  : [9230.59, 9279.6]
99% CI using Boot Strap method with sample size = 500  : [9223.47, 9279.13]
99% CI using Boot Strap method with sample size = 1000  : [9225.62, 9278.59]
99% CI using Boot Strap method with sample size = 5000  : [9224.75, 9280.26]
99% CI using Boot Strap method with sample size = 7000  : [9225.54, 9279.57]
99% CI using Boot Strap method with sample size = 10000  : [9225.8, 9280.99]


For Age Group  36-45
99% CI using Boot Strap method with sample size = 100  : [9300.56, 9375.67]
99% CI using Boot Strap method with sample size = 500  : [9292.52, 9367.6]
99% CI using Boot Strap method with sample size = 1000  : [9295.02, 9371.19]
99% CI using Boot Strap method with sample size = 5000  : [9294.01, 9370.56]
99% CI using Boot Strap method with sample size = 7000  : [9292.51, 9370.68]
99% CI using Boot Strap method with sample size = 10000  : [9293.0, 9368.61]


For Age Group  18-25
99% CI using Boot Strap method with sample size = 100  : [9134.19, 9201.62]
99% CI using Boot Strap method with sample size = 500  : [9130.37, 9202.64]
99% CI using Boot Strap method with sample size = 1000  : [9134.31, 9205.91]
99% CI using Boot Strap method with sample size = 5000  : [9127.02, 9210.74]
99% CI using Boot Strap method with sample size = 7000  : [9127.88, 9210.41]
99% CI using Boot Strap method with sample size = 10000  : [9129.26, 9209.56]


For Age Group  46-50
99% CI using Boot Strap method with sample size = 100  : [9160.49, 9269.49]
99% CI using Boot Strap method with sample size = 500  : [9149.28, 9264.35]
99% CI using Boot Strap method with sample size = 1000  : [9155.58, 9269.63]
99% CI using Boot Strap method with sample size = 5000  : [9147.87, 9265.7]
99% CI using Boot Strap method with sample size = 7000  : [9148.5, 9272.3]
99% CI using Boot Strap method with sample size = 10000  : [9148.71, 9268.49]


For Age Group  51-55
99% CI using Boot Strap method with sample size = 100  : [9467.74, 9607.73]
99% CI using Boot Strap method with sample size = 500  : [9452.47, 9608.67]
99% CI using Boot Strap method with sample size = 1000  : [9465.19, 9603.39]
99% CI using Boot Strap method with sample size = 5000  : [9467.24, 9602.99]
99% CI using Boot Strap method with sample size = 7000  : [9469.52, 9602.7]
99% CI using Boot Strap method with sample size = 10000  : [9465.67, 9600.25]


For Age Group  55+
99% CI using Boot Strap method with sample size = 100  : [9260.37, 9435.49]
99% CI using Boot Strap method with sample size = 500  : [9244.18, 9427.91]
99% CI using Boot Strap method with sample size = 1000  : [9252.08, 9429.97]
99% CI using Boot Strap method with sample size = 5000  : [9248.95, 9421.66]
99% CI using Boot Strap method with sample size = 7000  : [9250.35, 9420.44]
99% CI using Boot Strap method with sample size = 10000  : [9245.62, 9424.23]
```

```
For Age Group  0-17
99% CI using Boot Strap method with sample size = 100  : [8841.07, 9014.59]
99% CI using Boot Strap method with sample size = 500  : [8821.09, 9035.16]
99% CI using Boot Strap method with sample size = 1000  : [8820.6, 9029.68]
99% CI using Boot Strap method with sample size = 5000  : [8831.41, 9035.57]
99% CI using Boot Strap method with sample size = 7000  : [8825.87, 9040.92]
99% CI using Boot Strap method with sample size = 10000  : [8824.57, 9041.39]
```

Observation :-

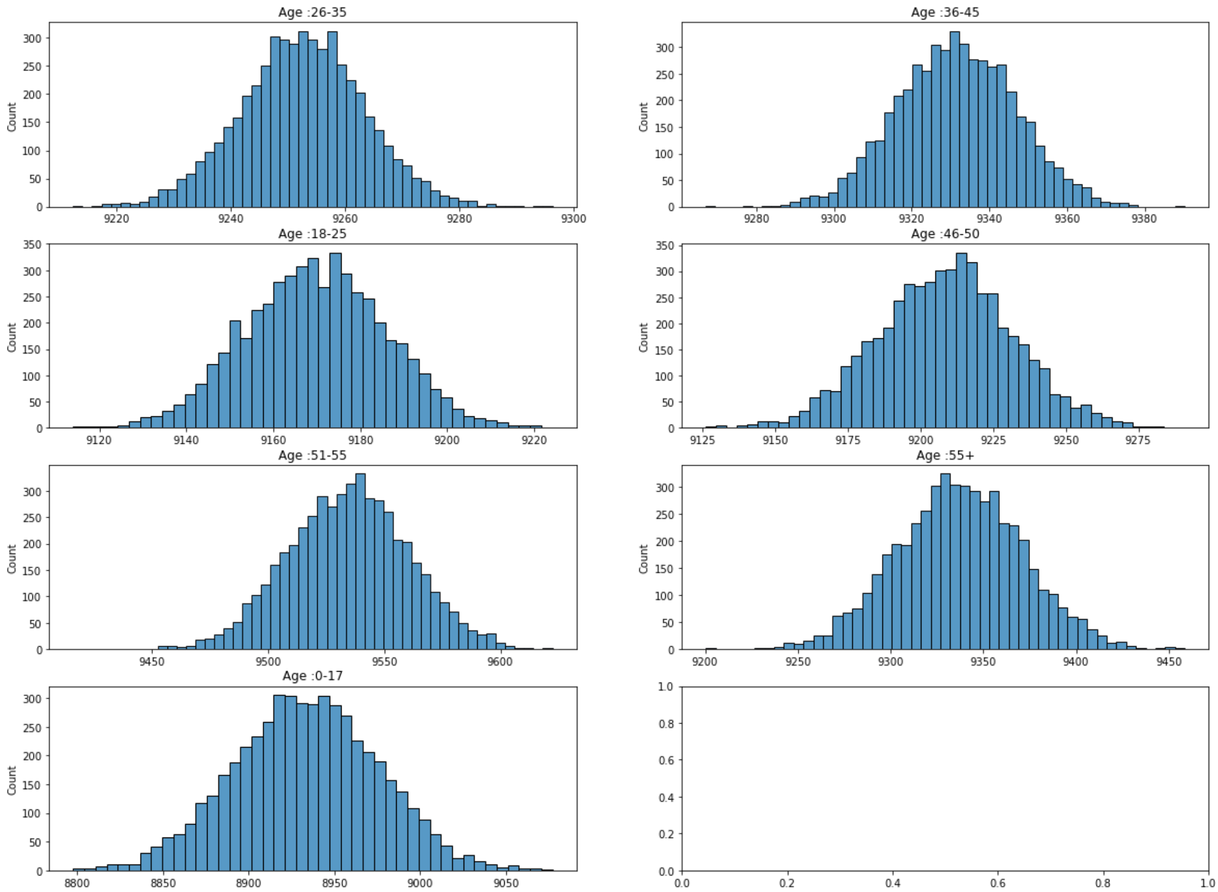
- 99% CI for Age Group 26-35 Average Spends: (9225, 9280)
- 99% CI for Age Group 36-45 Average Spends: (9292, 9370)
- 99% CI for Age Group 18-25 Average Spends: (9128, 9210)
- 99% CI for Age Group 46-50 Average Spends: (9148, 9268)
- 99% CI for Age Group 51-55 Average Spends: (9468, 9601)
- 99% CI for Age Group 55+ Average Spends: (9248, 9424)
- 99% CI for Age Group 0-17 Average Spends: (8826, 9040)
- 99% Confidence Interval are overlapping for Age Group 18-25 and 46-50 with average spend in range (9148,9210) . There is overlapping for Age group 26-35 and 55+ with average spend in range (9248,9280) . And there is overlapping for Age group 36-45 and 55+ with average spend in range (9292,9370) .

Histogram Plot : For Avg. Amount spend in Age group Category

In [254…

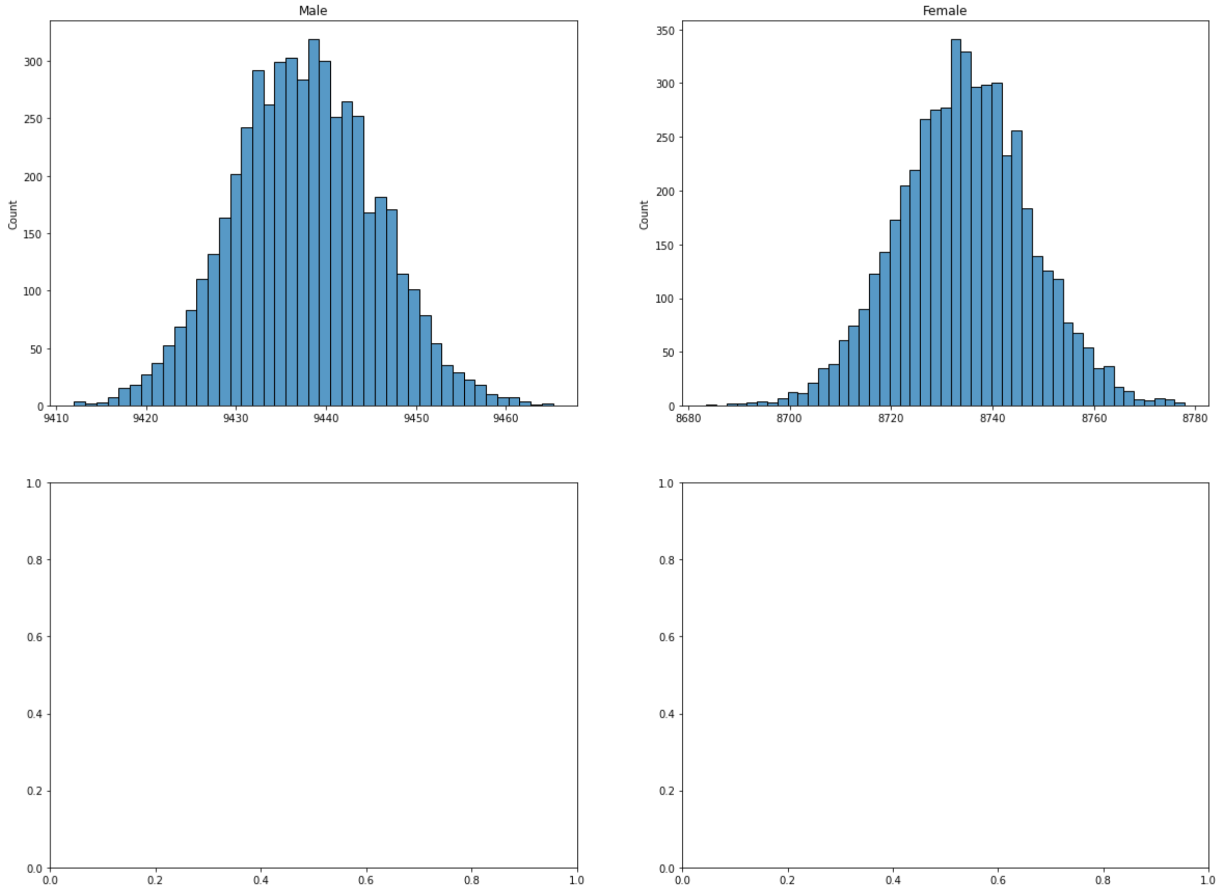
```python
def boot_strap_method2(data, sample_size,confidence_intterval):
    ans=[]
    for reps in range(sample_size):
        bootstrapped_samples = np.random.choice(data, size=data.shape[0])
        bootstrapped_mean =  np.mean(bootstrapped_samples)
        ans.append(bootstrapped_mean)
    return ans


fig, axis = plt.subplots(nrows=4, ncols=2, figsize=(20, 15))
# 99% CI

sns.histplot(boot_strap_method2(data = df.loc[df['Age'] == '26-35','Purchase'],
                                sample_size = 5000,confidence_intterval = 99),ax=axi
axis[0,0].set_title("Age :26-35")

sns.histplot(boot_strap_method2(data = df.loc[df['Age'] == '36-45','Purchase'],
                                sample_size = 5000,confidence_intterval = 99),ax=axi
axis[0,1].set_title("Age :36-45")

sns.histplot(boot_strap_method2(data = df.loc[df['Age'] == '18-25','Purchase'],
                                sample_size = 5000,confidence_intterval = 99),ax=axi
axis[1,0].set_title("Age :18-25")

sns.histplot(boot_strap_method2(data = df.loc[df['Age'] == '46-50','Purchase'],
                                sample_size = 5000,confidence_intterval = 99),ax=axi
axis[1,1].set_title("Age :46-50")

sns.histplot(boot_strap_method2(data = df.loc[df['Age'] == '51-55','Purchase'],
                                sample_size = 5000,confidence_intterval = 99),ax=axi
axis[2,0].set_title("Age :51-55")
```

```
sns.histplot(boot_strap_method2(data = df.loc[df['Age'] == '55+','Purchase'],
                                sample_size = 5000,confidence_intterval = 99),ax=axi
axis[2,1].set_title("Age :55+")

sns.histplot(boot_strap_method2(data = df.loc[df['Age'] == '0-17','Purchase'],
                                sample_size = 5000,confidence_intterval = 99),ax=axi
axis[3,0].set_title("Age :0-17")

plt.show()
```



In [256…

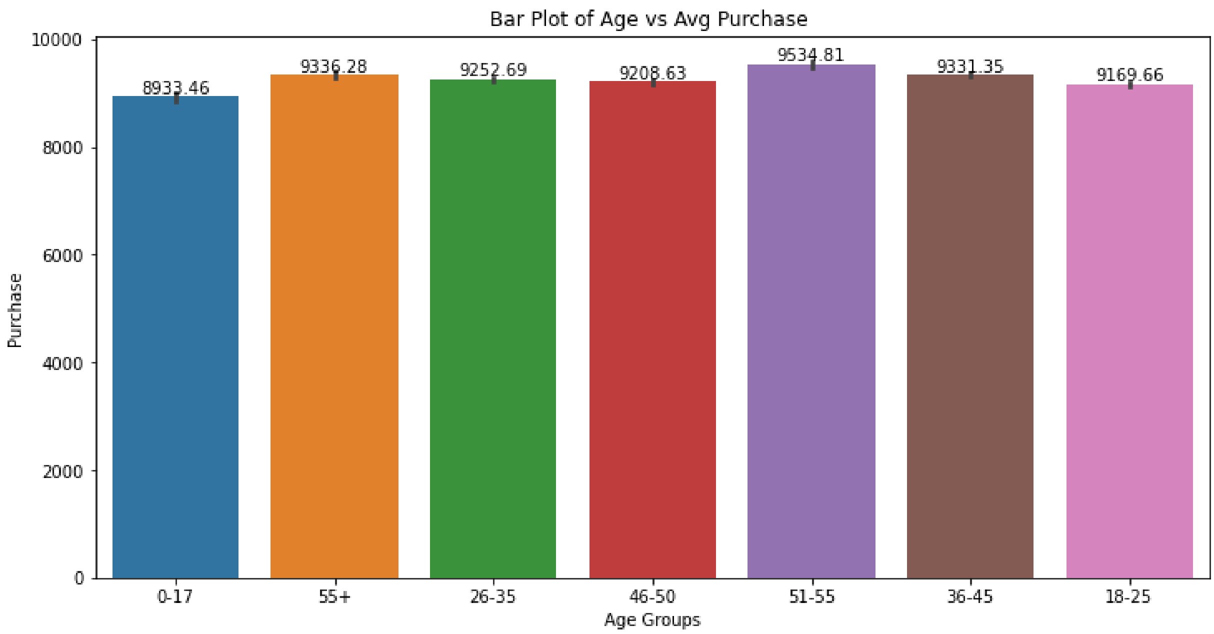
```
def boot_strap_method2(data, sample_size,confidence_intterval):
    ans=[]
    for reps in range(sample_size):
        bootstrapped_samples = np.random.choice(data, size=data.shape[0])
        bootstrapped_mean =  np.mean(bootstrapped_samples)
        ans.append(bootstrapped_mean)
    return ans


fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(20, 15))

# 99% CI
sns.histplot(boot_strap_method2(data = df.loc[df['Gender'] == 'M','Purchase'],
                                sample_size = 5000,confidence_intterval = 99),ax=axi
axis[0,0].set_title("Male")

sns.histplot(boot_strap_method2(data = df.loc[df['Gender'] == 'F','Purchase'],
                                sample_size = 5000,confidence_intterval = 99),ax=axi
axis[0,1].set_title("Female")


plt.show()
```



Plot Univariate and Bivariate plots

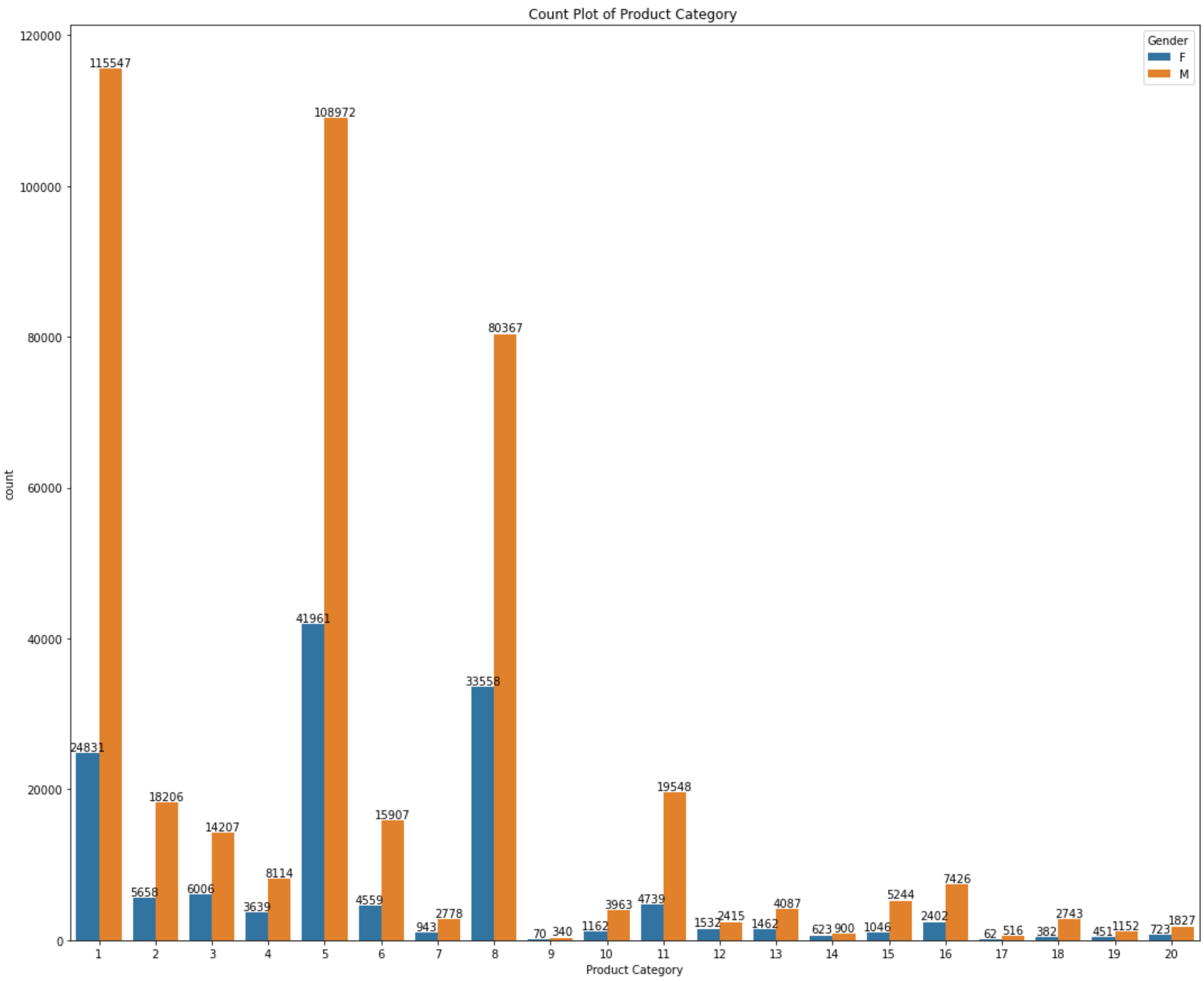
In [236…

```
plt.figure(figsize=(12,6))

ax= sns.barplot(data = df , x='Age' ,y='Purchase')

for i in ax.containers:
    ax.bar_label(i)

plt.xlabel('Age Groups')

plt.title('Bar Plot of Age vs Avg Purchase ')
plt.show()
```



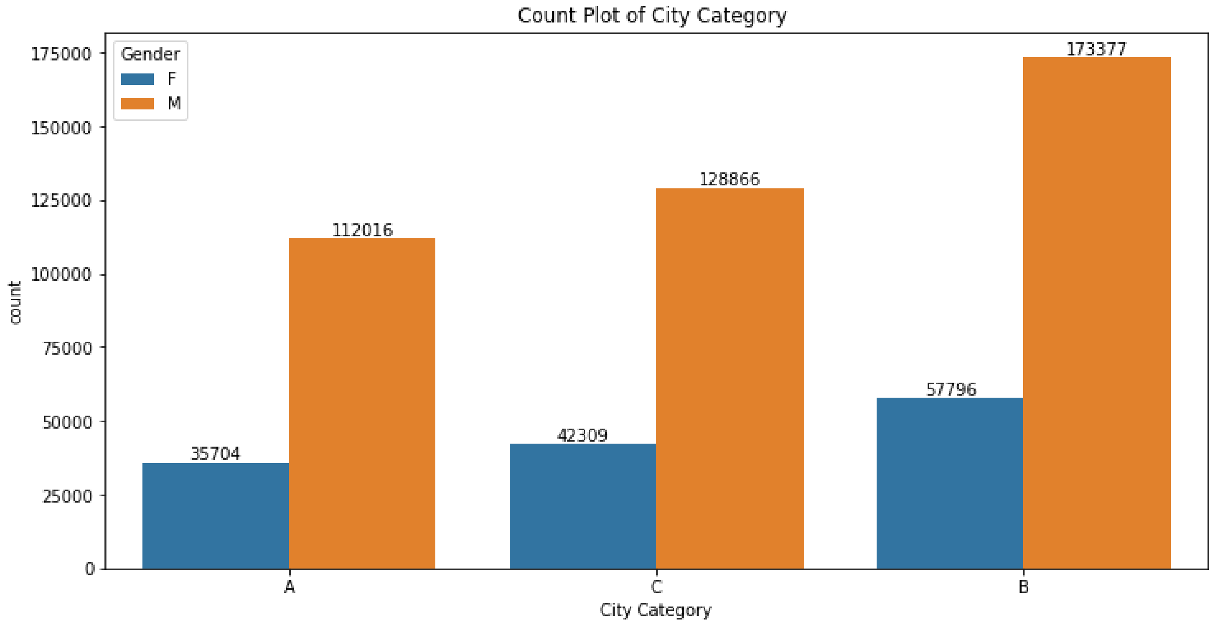
In [97]:

```
plt.figure(figsize=(18,15))

ax= sns.countplot(data = df , x='Product_Category', hue='Gender' )

for i in ax.containers:
    ax.bar_label(i)

plt.xlabel('Product Category')

plt.title('Count Plot of Product Category ')
plt.show()
```



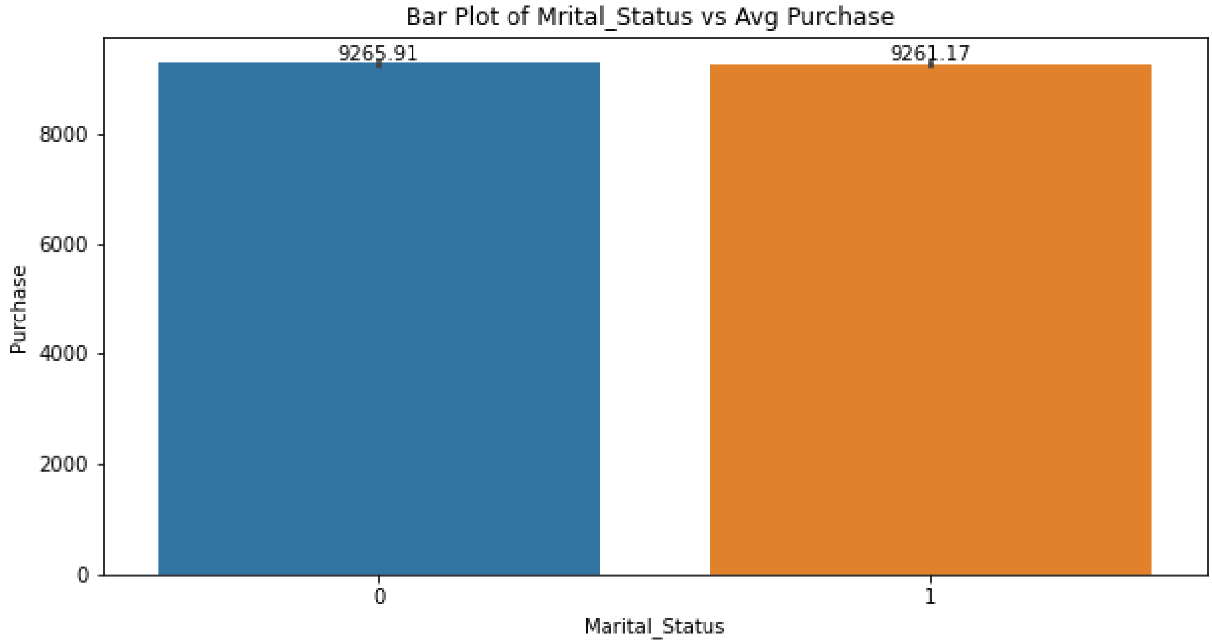
In [244…

```
plt.figure(figsize=(12,6))

ax= sns.countplot(data = df , x='City_Category', hue='Gender' )

for i in ax.containers:
    ax.bar_label(i)

plt.xlabel('City Category')

plt.title('Count Plot of City Category ')
plt.show()
```



In [235…

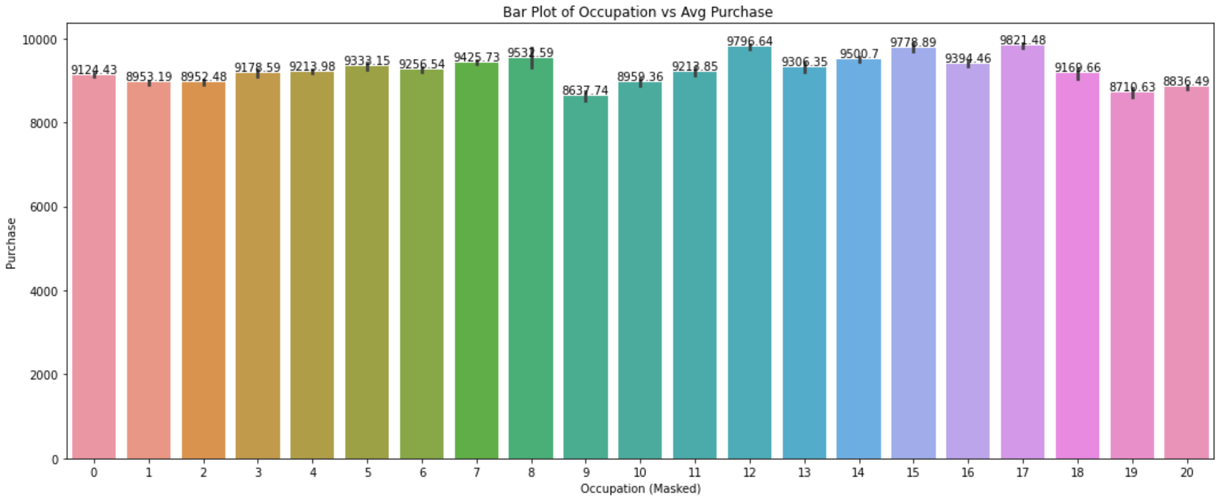
```
plt.figure(figsize=(10,5))

ax= sns.barplot(data = df , x='Marital_Status' ,y='Purchase')

for i in ax.containers:
    ax.bar_label(i)

plt.xlabel('Marital_Status')

plt.title('Bar Plot of Mrital_Status vs Avg Purchase ')
plt.show()
```



In [239…

```
plt.figure(figsize=(18,7))

ax= sns.barplot(data = df , x='Occupation' ,y='Purchase')

for i in ax.containers:
    ax.bar_label(i)

plt.xlabel('Occupation (Masked)')
```

```
plt.title('Bar Plot of Occupation vs Avg Purchase ')
plt.show()
```



In [ ]:

Insights :-

- There are no null values all columns.
- There are 2677 Total Outliers in Purchase Column.
- Confidence interval is the interval, we calculated using the sample data, within which the population parameter will lie .Below are the 90% CI , 95% CI and 99% CI values calulated for different Gender , Marital Status and Age Groups

Confidence Interval for Male / Female Average Amount Spend

- 90% CI for Male Average Spends = [9424 , 9450] , 90% CI for Female Average Spends= [8713 , 8756]
- 95% CI for Male Average Spends = [9422 , 9453] , 95% CI for Female Average Spends= [8709 , 8760]
- 99% CI for Male Average Spends = [9417 , 9458] , 99% CI for Female Average Spends= [8701 , 8768]
- As we go from 90% CI to 95% CI and to 99% CI we see that the range / width of values keep on increasing as we increace the Percentage of Confidence Interval
- Confidence Interval of Average Male and Female spends are NOT OVERLAPPING with each other in 90% , 95% and 99% CI.
- From 90% / 95% / 99% CI we can see the average amount spend by Male is large compared to females.
- 90% CI for Male Average Spend Amount means that there is 90% chance that the confidence interval [9424 , 9450] contains population mean amount spend By Male customer.

Confidence Interval for Single / Partnered Marital Status Average Amount Spend

- 90% CI for Single Status Average Spends = [9252 , 9281] , 90% CI for Partnered Average Spends= [9244, 9278]
- 95% CI for Single Status Average Spends = [9249 , 9283] , 95% CI for Partnered Average Spends= [9240, 9282]
- 99% CI for Single Status Average Spends = [9243 , 9288] , 99% CI for Partnered Average Spends= [9234, 9289]
- As we go from 90% CI to 95% CI and to 99% CI we see that the width of CI keep on increasing as we increace the Percentage of Confidence Interval .
- Confidence Interval of Average spends of Single and Partnered Marital Status Customer are overlapping with each other in 90% , 95% and 99% CI.
- 90% Confidence intervals of average spends of Single and Partnered marital status people ARE OVERLAPPING in average spends range of (9252,9278)
- 95% Confidence intervals of average spends of Single and Partnered marital status customers ARE OVERLAPPING in average spends range of (9249,9282)
- 99% Confidence intervals of average spends of Single and Partnered marital status customers ARE OVERLAPPING in average spends range of (9243,9288)
- From 90% / 95% / 99% CI we can see the average amount spend by Single Marital Status customer is almost same to Partnered Marital Status Customer.

Confidence Interval for each age group Average Amount Spend

- 90% CI for Age Group 26-35 Average Spends: (9235, 9270)
- 90% CI for Age Group 36-45 Average Spends: (9306, 9356)
- 90% CI for Age Group 18-25 Average Spends: (9143, 9195)
- 90% CI for Age Group 46-50 Average Spends: (9170, 9246)
- 90% CI for Age Group 51-55 Average Spends: (9492, 9577)
- 90% CI for Age Group 55+ Average Spends: (9280, 9392)
- 90% CI for Age Group 0-17 Average Spends: (8864, 9001)
- 90% Confidence Interval ARE OVERLAPPING for Age Group 18-25 and 46-50 with average spend in range (9170,9195). And there is overlapping for Age group 36-45 and 55+ with average spend in range (9306,9356) .

- 95% CI for Age Group 26-35 Average Spends: (9231, 9273)
- 95% CI for Age Group 36-45 Average Spends: (9301, 9361)
- 95% CI for Age Group 18-25 Average Spends: (9138, 9200)
- 95% CI for Age Group 46-50 Average Spends: (9163, 9254)
- 95% CI for Age Group 51-55 Average Spends: (9483, 9585)
- 95% CI for Age Group 55+ Average Spends: (9269, 9403)
- 95% CI for Age Group 0-17 Average Spends: (8851, 9014)
- 95% Confidence Interval ARE OVERLAPPING for Age Group 18-25 and 46-50 with average spend in range (9163,9200) . Also overlapping for Age group 26-35 and 55+ with average spend in range (9269,9273) . And there is overlapping for Age group 36-45 and 55+ with average spend in range (9301,9361) .

- 99% CI for Age Group 26-35 Average Spends: (9225, 9280)
- 99% CI for Age Group 36-45 Average Spends: (9292, 9370)
- 99% CI for Age Group 18-25 Average Spends: (9128, 9210)

- 99% CI for Age Group 46-50 Average Spends: (9148, 9268)
- 99% CI for Age Group 51-55 Average Spends: (9468, 9601)
- 99% CI for Age Group 55+ Average Spends: (9248, 9424)
- 99% CI for Age Group 0-17 Average Spends: (8826, 9040)
- 99% Confidence Interval ARE OVERLAPPING for Age Group 18-25 and 46-50 with average spend in range (9148,9210) . There is overlapping for Age group 26-35 and 55+ with average spend in range (9248,9280) . And there is overlapping for Age group 36-45 and 55+ with average spend in range (9292,9370) .

- As we go from 90% CI to 95% CI and to 99% CI we see that the width of CI keep on increasing as we increace the Percentage of Confidence Interval .

- Confidence Interval of Average spends of each Age Group Customer are overlapping with each other in 90% , 95% and 99% CI for few age groups.
- From 90% / 95% / 99% CI we can see the average amount spend by 51-55 Age group customer is highest and the average amount spend by 0-17 Age group customer is lowest .

Recommendations :-

- Confidence Interval of Average Male and Female spends are NOT OVERLAPPING with each other in 90% , 95% and 99% CI. So we can say that the male population average spends is more than female population average spends this can be inferred as the 90% / 95% /99% CI (Eg. Male has higher lower and upper paramters in 99% CI as compared to female 99% CI) . So Company needs to Target more of Female audience to reduce the gap between Male and Femal avg. amount spend. Note : Confidence interval is the interval, we calculated using the sample data, within which the population parameter will lie .

- Company should focus on retaining the male customers. More of female focused products should be introduced and special discounts - like clearance sale on existing female products can be done to increase the average amount spend by female.

- From Confidence Interval for Avg. amount spend by each age group we can see Age Group 51-55 has the highest avg. spend amount and the Age Group 0-17 has least avg. spend amount that is expected as cusotmer of age gorup 0-17 are considered children /minor and generally thier parents will buy products for them. The difference in avg. spend amount between Age group 51-55 and other age groups is not much , so this is a plus point showing company has wide variety of products catering to needs of all major age groups.

- 99% Confidence Interval ARE OVERLAPPING for Age Group 18-25 and 46-50 for average spend amount . There is overlapping for Age group 26-35 and 55+ for average spend amount . And there is overlapping for Age group 36-45 and 55+ for average spendamount. So we cannot compare these groups with each other to check if avg. amount spend is higher or lower compared to other.

- Company should target more of younger Age Groups Like 18-25 , 26-35 and 36-45 . New Products in field of technology can be introduced . Cashbacks and special discounts like student discount can be given to 18-25 Age Group People.

- 90% and 95% and 99% Confidence intervals of average spends of Single and Partnered marital status people ARE OVERLAPPING so we cannoty compare Single and Partneres Marital Status customer with each other for highest or lowest average spend amount .
- From Count Plot of City Category we can see that Count of orders purchased by Female in All cities A, B, C are almost 30% to that of Male. So city wise campaign and adviretisement can be done specially for Female Gender to encourage more females to buy products from Walmart .
- Seeing the Count Plot of Product Category we can see that Product Category 1 , 5 and 8 are most ordered product category among males and females . And product category 7, 9,10,12,13,14,15,16,17,18,19,20 are least ordered categories. These least order product categories must be either changed with new product categories or latest products in these categories must be introduced .
- Company can introduce programs such as loyalty program where in they give cashback to thier most frequent customers.

In [ ]:

```

```