In [3]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import math
from scipy.stats import norm
from scipy.stats import binom
```

In [4]:
```
walmart=pd.read_csv("walmart_data.csv")
```

In [3]:
```
walmart.shape
```

Out[3]:
```
(550068, 10)
```

In [4]:
```
walmart.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
```

No null values

In [6]:
```
walmart.head()
```

Out[6]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_
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	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	

Unique Values

In [41]:
```
walmart["User_ID"].nunique()
```

Out[41]: 5891

In [42]:
```
walmart["Product_ID"].nunique()
```

Out[42]: 3631

In [43]:
```
walmart["Gender"].nunique()
```

Out[43]: 2

In [44]:
```
walmart["Age"].nunique()
```

Out[44]: 7

In [45]:
```
walmart["Occupation"].nunique()
```

Out[45]: 21

In [47]:
```
walmart["City_Category"].nunique()
```

Out[47]: 3

In [48]:
```
walmart["Stay_In_Current_City_Years"].nunique()
```

Out[48]: 5

In [50]:
```
walmart["Marital_Status"].nunique()
```

Out[50]: 2

In [51]:
```
walmart["Product_Category"].nunique()
```

Out[51]: 20

In [52]:
```
walmart["Purchase"].nunique()
```

Out[52]: 18105

Value Counts

In [55]:
```
walmart["User_ID"].value_counts(normalize=True)*100
```

Out[55]:
```
1001680    0.186522
1004277    0.177978
1001941    0.163253
1001181    0.156708
1000889    0.149618
             ...
1002690    0.001273
1002111    0.001273
1005810    0.001273
1004991    0.001273
1000708    0.001091
Name: User_ID, Length: 5891, dtype: float64
```

In [56]:
```
walmart["Product_ID"].value_counts(normalize=True)*100
```

Out[56]:
```
P00265242    0.341776
P00025442    0.293600
P00110742    0.293055
P00112142    0.283965
P00057642    0.267240
               ...
P00314842    0.000182
P00298842    0.000182
P00231642    0.000182
P00204442    0.000182
P00066342    0.000182
Name: Product_ID, Length: 3631, dtype: float64
```

In [57]:
```
walmart["Gender"].value_counts(normalize=True)
```

Out[57]:
```
M    0.753105
F    0.246895
Name: Gender, dtype: float64
```

In [58]:
```
walmart["Age"].value_counts(normalize=True)
```

Out[58]:
```
26-35    0.399200
36-45    0.199999
18-25    0.181178
46-50    0.083082
51-55    0.069993
55+      0.039093
0-17     0.027455
Name: Age, dtype: float64
```

In [59]:
```
walmart["Occupation"].value_counts(normalize=True)
```

Out[59]:
```
4     0.131453
0     0.126599
7     0.107501
1     0.086218
17    0.072796
20    0.061014
12    0.056682
14    0.049647
2     0.048336
16    0.046123
6     0.037005
3     0.032087
10    0.023506
5     0.022137
15    0.022115
11    0.021063
19    0.015382
13    0.014049
18    0.012039
9     0.011437
8     0.002811
Name: Occupation, dtype: float64
```

In [60]:
```
walmart["City_Category"].value_counts(normalize=True)
```

Out[60]:
```
B    0.420263
C    0.311189
A    0.268549
Name: City_Category, dtype: float64
```

In [61]:
```
walmart["Stay_In_Current_City_Years"].value_counts(normalize=True)
```

Out[61]:
```
1     0.352358
2     0.185137
3     0.173224
4+    0.154028
0     0.135252
Name: Stay_In_Current_City_Years, dtype: float64
```

In [62]:
```
walmart["Marital_Status"].value_counts(normalize=True)
```

Out[62]:
```
0    0.590347
1    0.409653
Name: Marital_Status, dtype: float64
```

In [63]:
```
walmart["Product_Category"].value_counts(normalize=True)
```

Out[63]:
```
5     0.274390
1     0.255201
8     0.207111
11    0.044153
2     0.043384
6     0.037206
3     0.036746
4     0.021366
16    0.017867
15    0.011435
13    0.010088
10    0.009317
12    0.007175
7     0.006765
18    0.005681
20    0.004636
19    0.002914
14    0.002769
17    0.001051
9     0.000745
Name: Product_Category, dtype: float64
```

In [64]:
```
walmart["Purchase"].value_counts(normalize=True)
```

Out[64]:
```
7011     0.000347
7193     0.000342
6855     0.000340
6891     0.000335
7012     0.000333
           ...
23491    0.000002
18345    0.000002
3372     0.000002
855      0.000002
21489    0.000002
Name: Purchase, Length: 18105, dtype: float64
```

Unique_Values

In [65]:
```
walmart["User_ID"].unique()
```

Out[65]:
```
array([1000001, 1000002, 1000003, ..., 1004113, 1005391, 1001529],
      dtype=int64)
```

In [66]:
```
walmart["Product_ID"].unique()
```

Out[66]:
```
array(['P00069042', 'P00248942', 'P00087842', ..., 'P00370293',
       'P00371644', 'P00370853'], dtype=object)
```

In [67]:
```
walmart["Gender"].unique()
```

Out[67]:
```
array(['F', 'M'], dtype=object)
```

In [74]:
```
walmart["Age"].unique()
```

Out[74]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

In [73]:
```
walmart["Occupation"].unique()
```

Out[73]:
```
array([10, 16, 15,  7, 20,  9,  1, 12, 17,  0,  3,  4, 11,  8, 19,  2, 18,
        5, 14, 13,  6], dtype=int64)
```

In [72]:
```
walmart["City_Category"].unique()
```

Out[72]:
```
array(['A', 'C', 'B'], dtype=object)
```

In [68]:
```
walmart["Stay_In_Current_City_Years"].unique()
```

Out[68]:
```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

In [69]:
```
walmart["Marital_Status"].unique()
```

Out[69]:
```
array([0, 1], dtype=int64)
```

In [70]:
```
walmart["Product_Category"].unique()
```

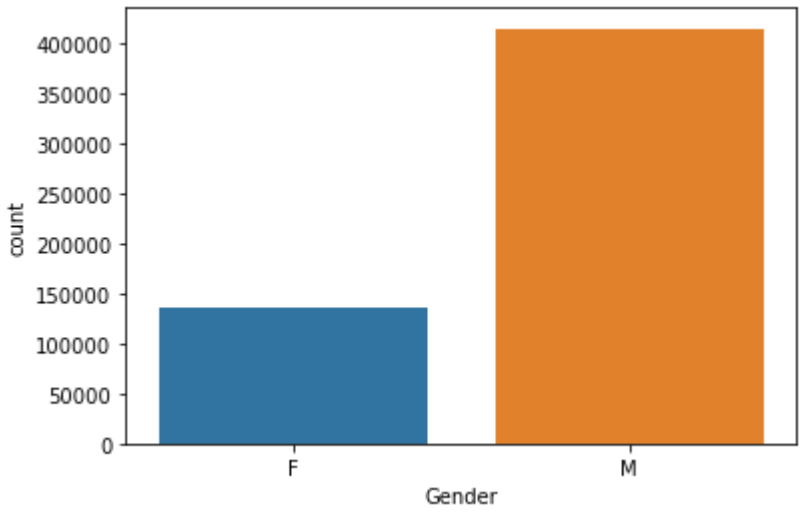
Out[70]:
```
array([ 3,  1, 12,  8,  5,  4,  2,  6, 14, 11, 13, 15,  7, 16, 18, 10, 17,
        9, 20, 19], dtype=int64)
```

In [71]:
```
walmart["Purchase"].unique()
```

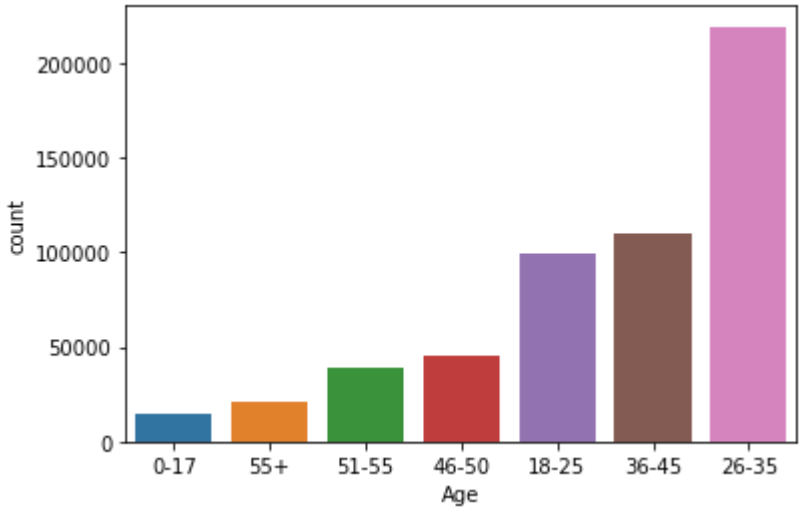
Out[71]:
```
array([ 8370, 15200,  1422, ...,   135,   123,   613], dtype=int64)
```

Univariate Analysis

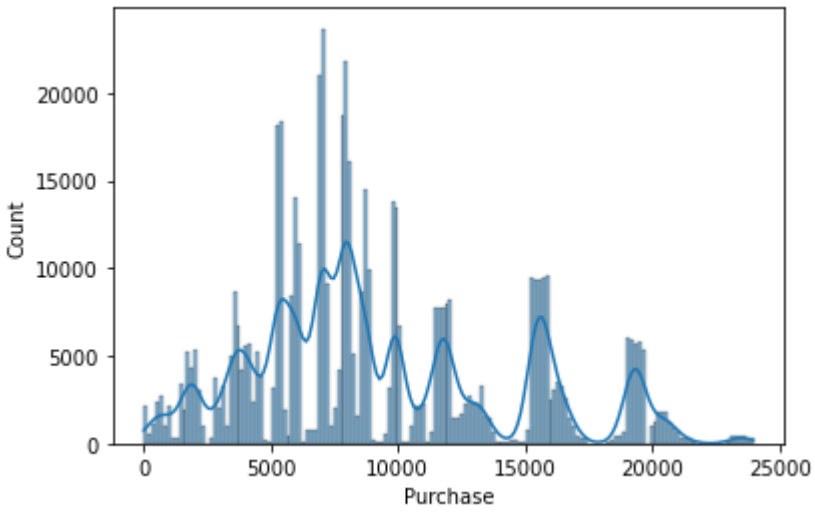
In [76]:
```
sns.countplot(data=walmart,x="Gender")
plt.show()
```



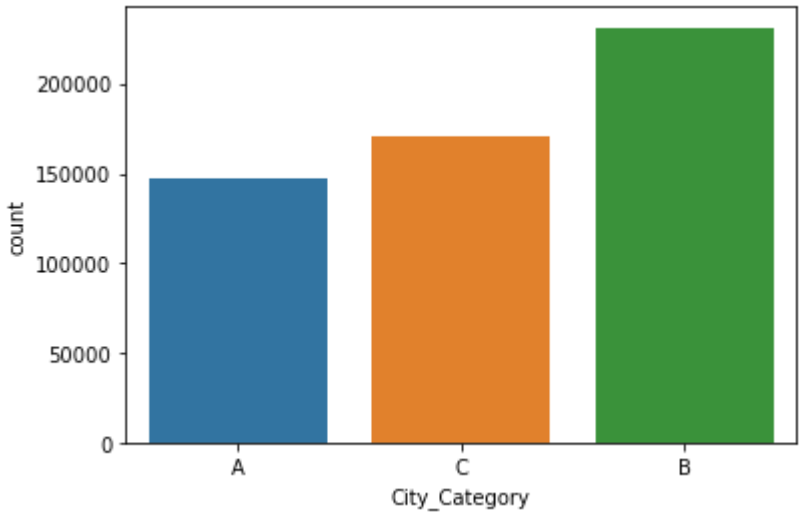
In [82]:
```
sns.countplot(data=walmart,x="Age",order=walmart.groupby("Age")["User_ID"].count().sor
plt.show()
```



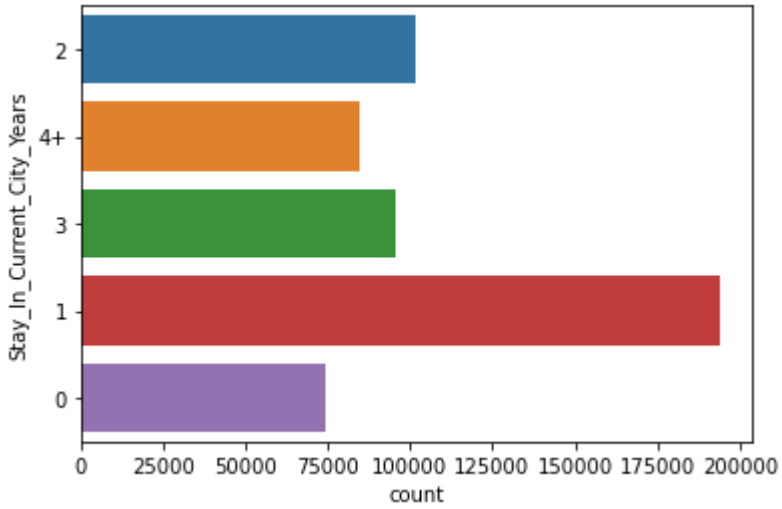
In [84]:
```
sns.histplot(data=walmart,x="Purchase",kde=True)
plt.show()
```



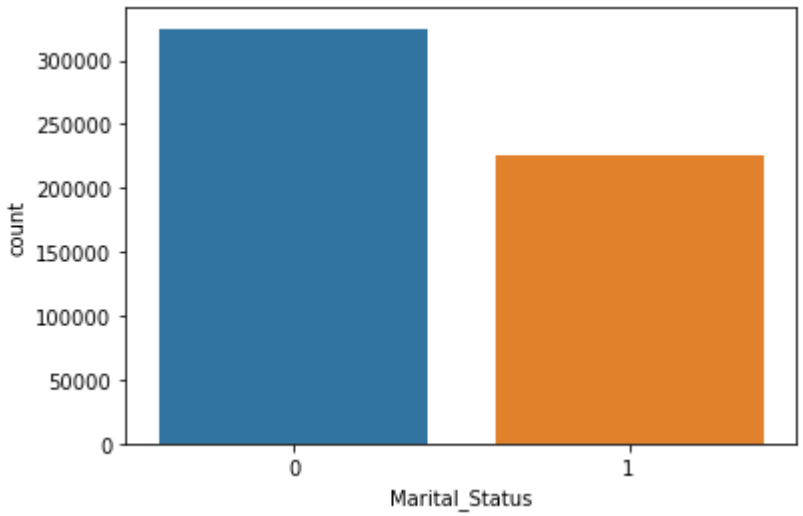
In [85]:
```
sns.countplot(data=walmart,x="City_Category")
plt.show()
```



In [89]:
```
sns.countplot(data=walmart,y="Stay_In_Current_City_Years")
plt.show()
```



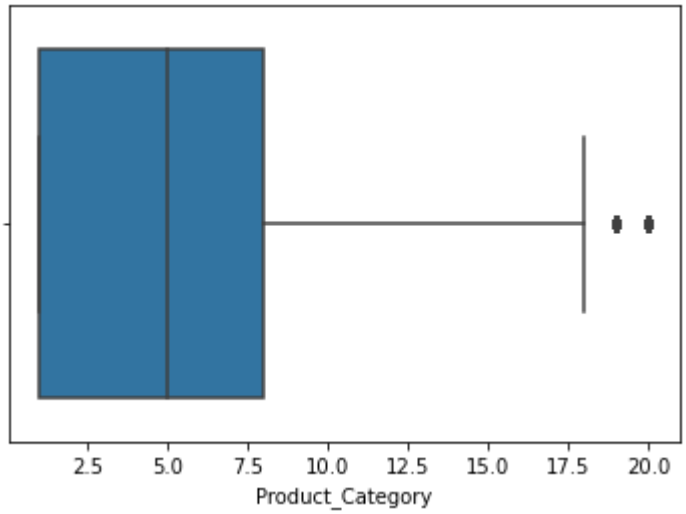
In [90]:
```
sns.countplot(data=walmart,x="Marital_Status")
plt.show()
```



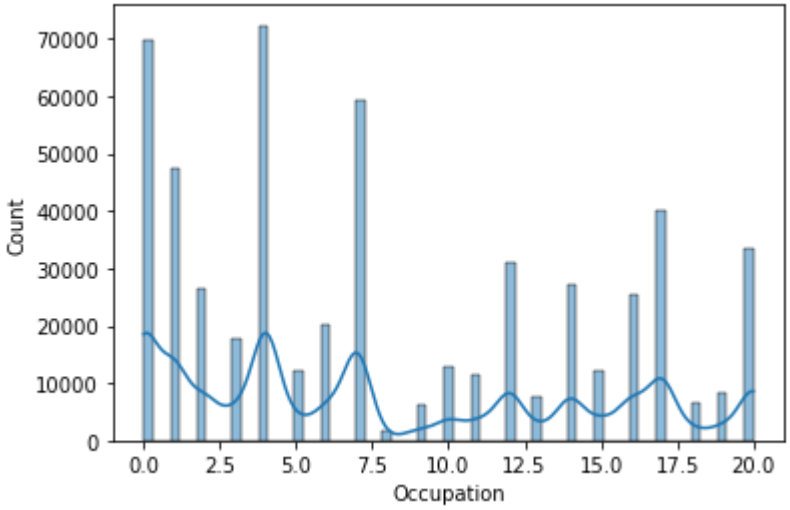
In [92]:
```
walmart["Product_Category"].dtype
```

Out[92]:
```
dtype('int64')
```

In [93]:
```
sns.boxplot(data=walmart,x="Product_Category")
plt.show()
```

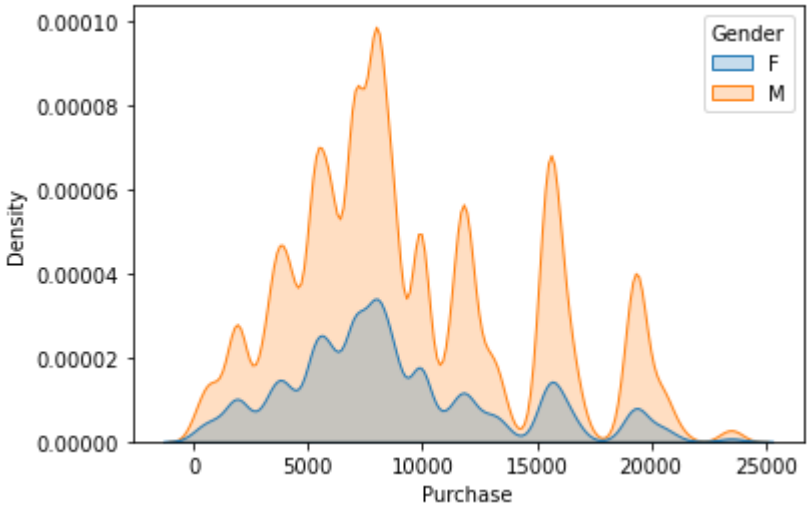


In [96]:
```
sns.histplot(data=walmart,x="Occupation",kde=True)
plt.show()
```

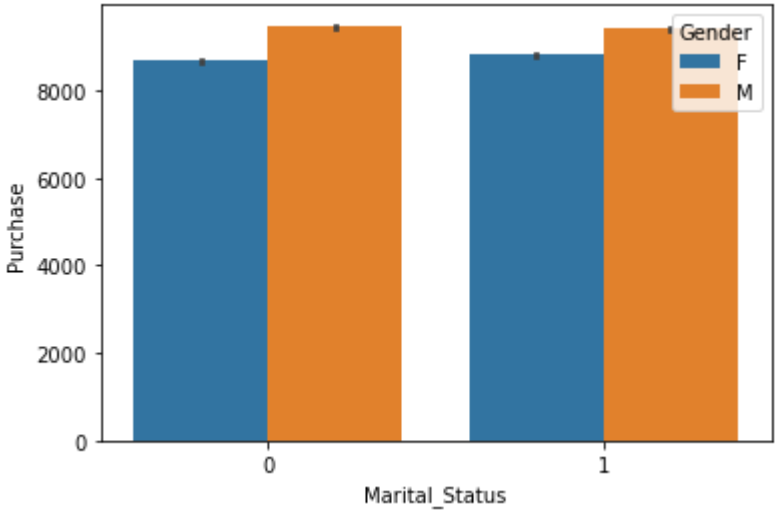


Bivariate Analysis

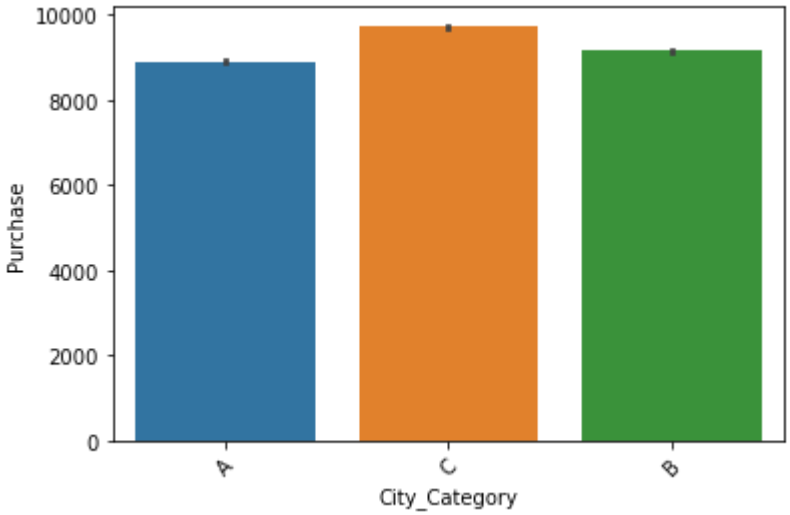
In [99]:
```
sns.kdeplot(data=walmart,x="Purchase",hue="Gender",shade=True)
plt.show()
```



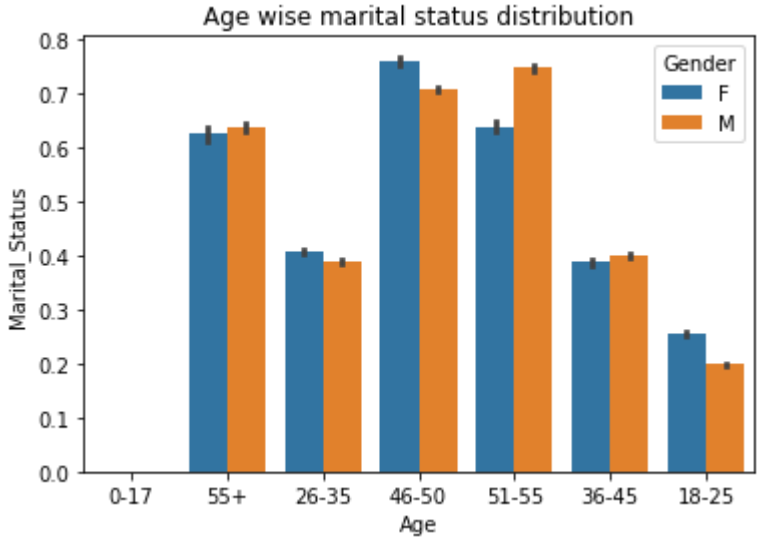
In [101…
```
sns.barplot(data=walmart,x="Marital_Status",y="Purchase",hue="Gender")
plt.show()
```



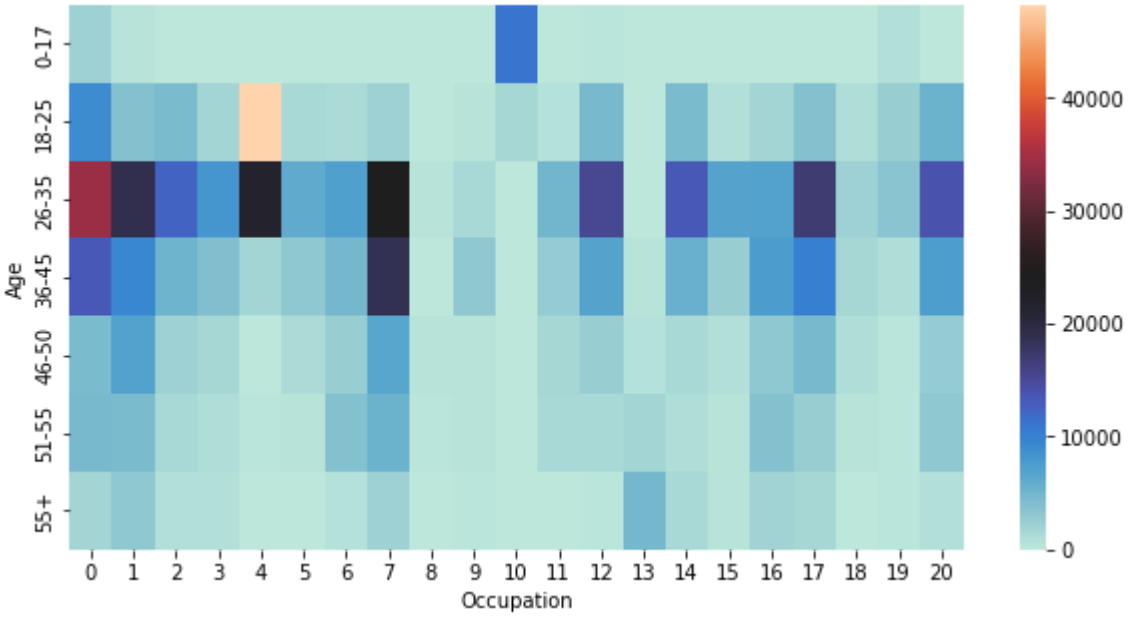
In [105…
```
sns.barplot(data=walmart,x="City_Category",y="Purchase")
plt.xticks(rotation=45)
plt.show()
```



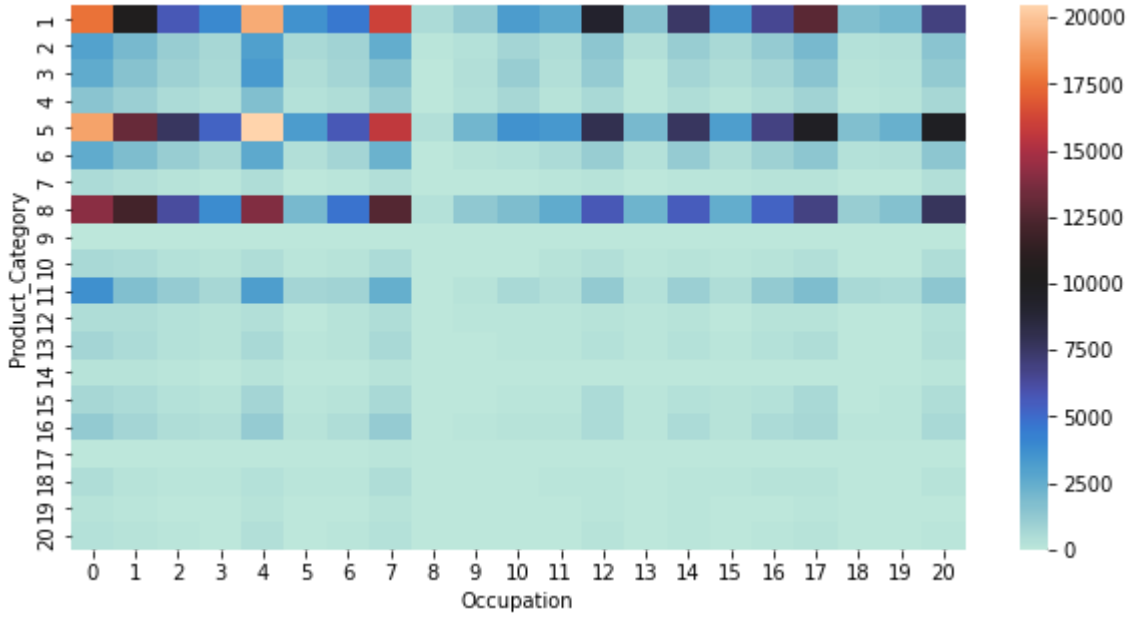
In [107…
```
sns.barplot(data=walmart,x="Age",y="Marital_Status",hue="Gender")
plt.title("Age wise marital status distribution")
plt.show()
```



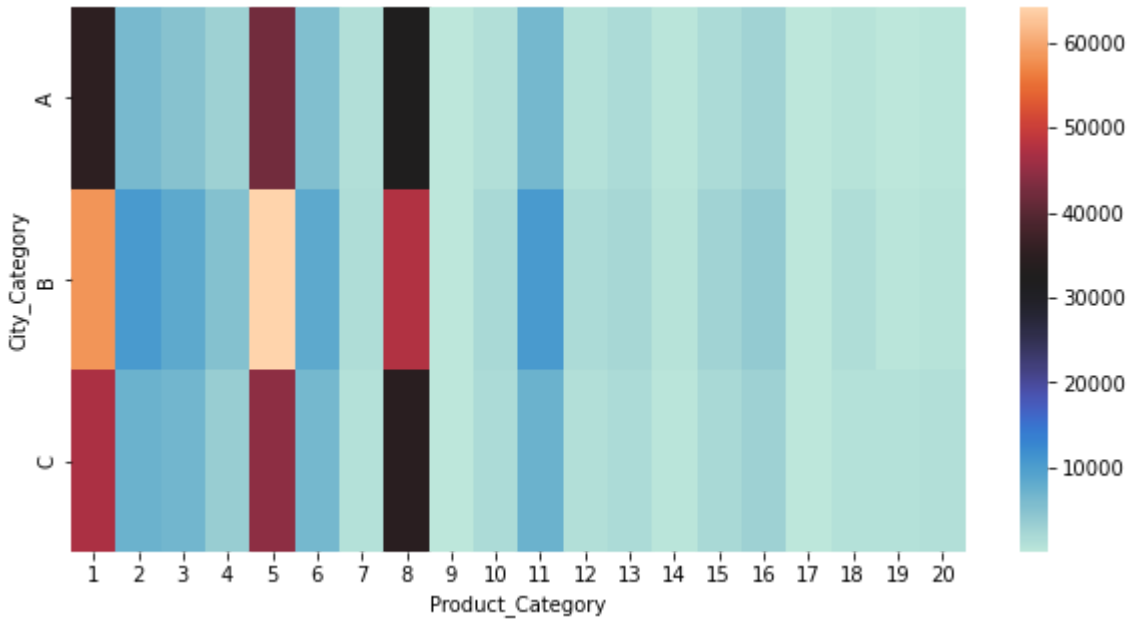
In [117…
```
plt.figure(figsize=(10,5))
sns.heatmap(pd.crosstab(index=walmart["Age"],columns=walmart["Occupation"]),cmap="ice
plt.show()
```



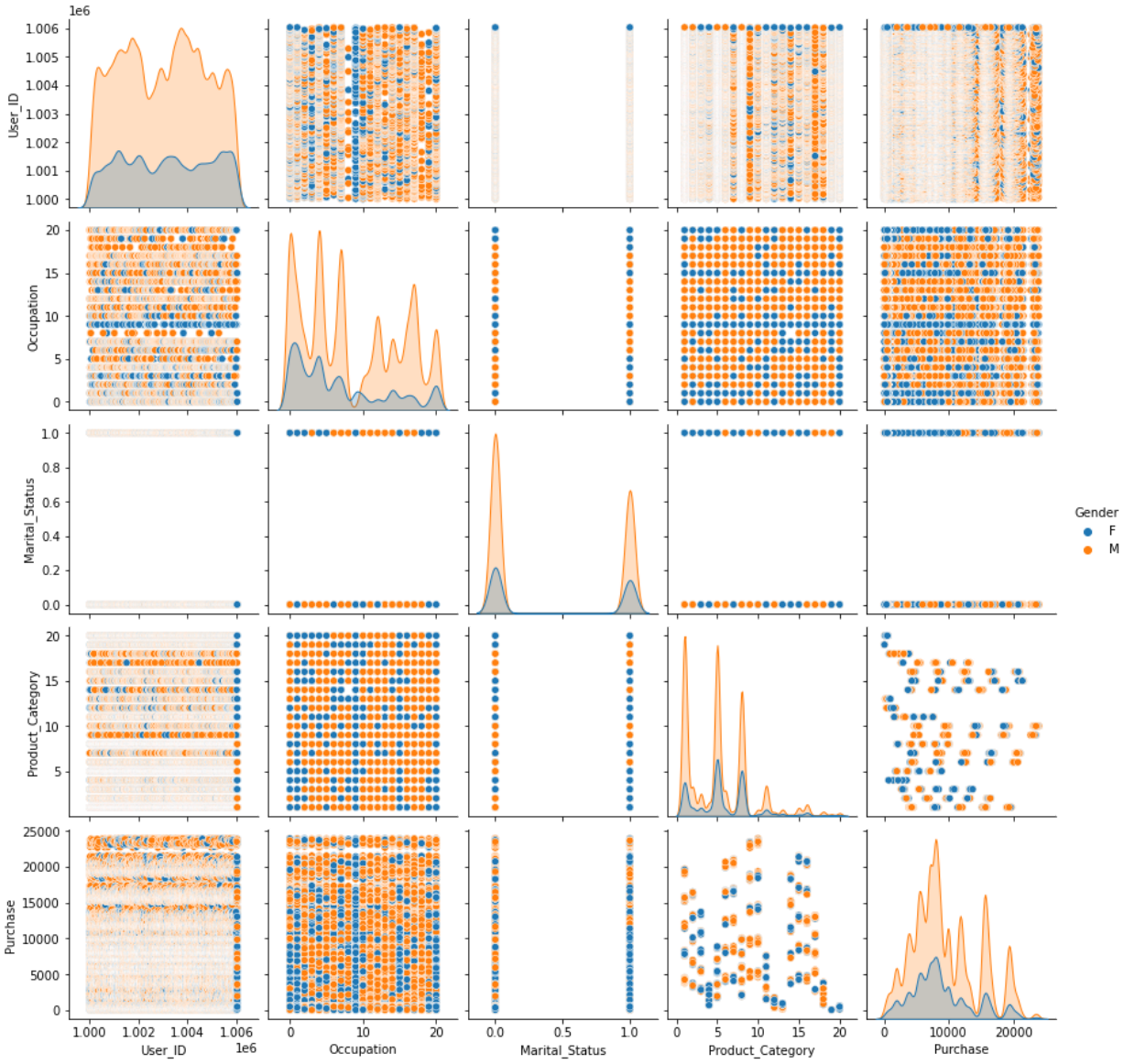
In [9]:
```
plt.figure(figsize=(10,5))
sns.heatmap(pd.crosstab(index=walmart["Product_Category"],columns=walmart["Occupation"
plt.show()
```



In [10]:
```
plt.figure(figsize=(10,5))
sns.heatmap(pd.crosstab(index=walmart["City_Category"],columns=walmart["Product_Categ
plt.show()
```



In [106…
```
sns.pairplot(data=walmart,hue="Gender")
plt.show()
```



Statistical Summary

In [11]:
```
walmart.describe().T
```

Out[11]:

	count	mean	std	min	25%	50%	75%	
User_ID	550068.0	1.003029e+06	1727.591586	1000001.0	1001516.0	1003077.0	1004478.0	1
Occupation	550068.0	8.076707e+00	6.522660	0.0	2.0	7.0	14.0	
Marital_Status	550068.0	4.096530e-01	0.491770	0.0	0.0	0.0	1.0	
Product_Category	550068.0	5.404270e+00	3.936211	1.0	1.0	5.0	8.0	
Purchase	550068.0	9.263969e+03	5023.065394	12.0	5823.0	8047.0	12054.0	

Null Value and outlier detection

In [12]:
```
walmart.isna().sum()
```

Out[12]:
```
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
```

In [ ]:
```
# No null values means no missing values in the datatset
```

In [16]:
```
Purchase_25=np.percentile(walmart["Purchase"],25)
Purchase_25
```

Out[16]:
```
5823.0
```

In [17]:
```
Purchase_75=np.percentile(walmart["Purchase"],75)
Purchase_75
```

Out[17]:
```
12054.0
```

In [18]:
```
IQR=Purchase_75-Purchase_25
IQR
```

Out[18]:
```
6231.0
```

In [19]:
```
Upper_whisker=Purchase_75+1.5*IQR
Upper_whisker
```

Out[19]:
```
21400.5
```

In [20]:
```
Lower_whisker=Purchase_75-1.5*IQR
Lower_whisker
```

Out[20]:
```
2707.5
```

In [26]:
```
Outlier_pct=walmart[(walmart["Purchase"]>Upper_whisker) | (walmart["Purchase"]<Lower_w
Outlier_pct
```

Out[26]:
```
7.207836122079452
```

There are 7.2% data that are outliers in the walmart Purchase column.

Business Insights based on Non- Graphical and Visual Analysis

Insights: There are around 5 lakhs data for male and female spending on Black Friday in Walmart. A sample of 50 million of female data and 50 million of male data is to be extrapolated using the population data of 5 lakh. Here

nearly 70% data is of Male customers and 30% data is of female customers. By reconnaisance it looks that the average male spending is more than the female however to emerge a complete picture we will have to apply central limit theorem and statistics and confidence interval to comment on each gender spending in Black Friday Sales. Numerous people of differnt age bands are shopping during the time and it is found that the age band of 18-46 are purchasing in bulk amount as they are the people emerging maximum to shop. Married and Unmarried people are equally likely to sspend in Black Friday with Unmarried people shopping higher than the married couples. City Category A, B, C are more or less equally likely to shop in Black Friday just that CIty Category c has little higher sale than that of A and B. Product Category from 0-8 account for 50% sales. All occupation types are more or less equally likely to shop according to given data.

CLT and Confidence Interval for Male and female Customers

In [27]:
```
walmart_female_purchase=walmart.loc[walmart["Gender"]=="F","Purchase"]
```

In [28]:
```
walmart_female_purchase
```

Out[28]:
```
0          8370
1         15200
2          1422
3          1057
14         5378
          ...
550061      599
550064      371
550065      137
550066      365
550067      490
Name: Purchase, Length: 135809, dtype: int64
```

In [29]:
```
walmart_male_purchase=walmart.loc[walmart["Gender"]=="M","Purchase"]
```

In [30]:
```
walmart_male_purchase
```

Out[30]:
```
4          7969
5         15227
6         19215
7         15854
8         15686
          ...
550057       61
550058      121
550060      494
550062      473
550063      368
Name: Purchase, Length: 414259, dtype: int64
```

As creating a sample of 50 million female and 50 million male customers using bootstrap would take higher time, we will assume a sample distribution of means of 50 samples to create a population of 50 million.

As Central limit theorem states that the sample distribution of means will be a Gaussian distribution given that we consider a bigger sample.If Sample_size>30, we can assume Gaussian Distribution.

In [37]:
```
female_sample_mean=walmart_female_purchase.mean()
female_sample_mean
```

Out[37]:
```
8734.565765155476
```

In [32]:
```
female_population_std=walmart_female_purchase.std()
```

In [33]:
```
female_sample_std=female_population_std/np.sqrt(50)
```

Calculating z score of 2.5 percentile and 97.5 percentile

In [34]:
```
z_95_lower=norm.ppf(0.025)
```

In [35]:
```
z_95_upper=norm.ppf(0.975)
```

In [36]:
```
female_95pct_confidence=(female_sample_mean+z_95_lower*female_sample_std,female_sample
female_95pct_confidence
```

Out[36]:
```
(7413.180395716255, 10055.951134594696)
```

90 Percentile female purchases

In [38]:
```
z_90_lower=norm.ppf(0.05)
```

In [39]:
```
z_90_upper=norm.ppf(0.95)
```

In [40]:
```
female_90pct_confidence=(female_sample_mean+z_90_lower*female_sample_std,female_sample
female_90pct_confidence
```

Out[40]:
```
(7625.62420568454, 9843.507324626411)
```

99 percentile female purchases

In [41]:
```
z_99_lower=norm.ppf(0.005)
```

In [42]:
```
z_99_upper=norm.ppf(0.995)
```

In [43]:
```
female_99pct_confidence=(female_sample_mean+z_99_lower*female_sample_std,female_sample
female_99pct_confidence
```

Out[43]:
```
(6997.971020185672, 10471.160510125279)
```

Male Calculations

In [44]:
```
male_sample_mean=walmart_male_purchase.mean()
male_sample_mean
```

Out[44]:
```
9437.526040472265
```

In [45]:
```
male_population_std=walmart_male_purchase.std()
```

In [46]:
```
male_sample_std=male_population_std/np.sqrt(50)
```

90 percentile Male purchases

In [47]:
```
z_90_lower_male=norm.ppf(0.05)
z_90_upper_male=norm.ppf(0.95)
male_90pct_confidence=(male_sample_mean+z_90_lower_male*male_sample_std,male_sample_me
male_90pct_confidence
```

Out[47]:
```
(8252.994767527816, 10622.057313416713)
```

95 percentile Male Purchases

In [48]:
```
z_95_lower_male=norm.ppf(0.025)
z_95_upper_male=norm.ppf(0.975)
male_95pct_confidence=(male_sample_mean+z_95_lower_male*male_sample_std,male_sample_me
male_95pct_confidence
```

Out[48]:
```
(8026.069971985875, 10848.982108958655)
```

99 percentile Male Purchases

In [49]:
```
z_99_lower_male=norm.ppf(0.005)
z_99_upper_male=norm.ppf(0.995)
male_99pct_confidence=(male_sample_mean+z_99_lower_male*male_sample_std,male_sample_me
male_99pct_confidence
```

Out[49]:
```
(7582.558331597577, 11292.493749346952)
```

Female Purchases have a mean of 8734.56 with 95% Confidence Interval in range (7413.180395716255, 10055.951134594696)

Male Purchases have a mean of 9437.52 with 95% Confidence Interval in range (8026.069971985875, 10848.982108958655)

As the 95% confidence interval of male and female purchases **Coincides** with each other, the purchases of male and female are **statistically insignificant**

Creating a sample for cross checking the above data

In [78]:
```
sample_walmart_female=[walmart_female["Purchase"].sample(50).mean() for i in range(100
```

In [79]:
```
sample_walmart_female
```

Out[79]:
```
[7957.72,
 9694.34,
 7723.74,
 10037.14,
 8861.86,
 10010.86,
 8967.56,
 8128.64,
 9284.06,
 9453.92,
 9299.2,
 9658.32,
 8590.96,
 8529.92,
 9073.7,
 8111.94,
 8981.96,
 8438.52,
 9270.76,
 8332.38,
 8676.78,
 8982.82,
 8507.64,
 9219.46,
 9674.0,
 7968.98,
 8597.72,
 9117.34,
 8780.42,
 8620.62,
 8298.2,
 8899.5,
 7976.6,
 9548.7,
 8252.4,
 9018.82,
 9851.24,
 9031.9,
 9358.36,
 8624.4,
 8483.28,
 8594.5,
 8478.16,
 9114.52,
 9104.7,
 8694.7,
 9330.36,
 8274.38,
 9266.08,
 8484.6,
 8653.24,
 9042.68,
 8420.96,
 8433.66,
 8073.12,
 8466.72,
 7708.5,
 9029.4,
 8020.56,
 8958.14,
```

```
9308.0,
8454.44,
8381.36,
9538.94,
9377.92,
10243.9,
8043.86,
8588.3,
8673.32,
8232.66,
9862.72,
8962.22,
9569.8,
8458.8,
8011.88,
9874.96,
9047.36,
8479.08,
8376.58,
8039.0,
8768.62,
8289.0,
9105.44,
8036.18,
9340.66,
8575.58,
7566.36,
9076.66,
8163.04,
9086.98,
8922.24,
8189.98,
9592.2,
9331.52,
8112.12,
8972.52,
8884.92,
8087.72,
9425.54,
9472.96,
8300.28,
9765.9,
8890.56,
9004.62,
8006.62,
8623.34,
8575.22,
8403.82,
7757.1,
8108.98,
7565.5,
8676.58,
9277.54,
8920.52,
8358.38,
8510.32,
8961.32,
8819.42,
7777.16,
9393.06,
```

```
9123.72,
8360.92,
8512.7,
8334.82,
9136.86,
8966.08,
9414.54,
8619.98,
8822.14,
8384.58,
7835.02,
9009.26,
8326.46,
9367.06,
8597.12,
8443.08,
7939.82,
8816.94,
8067.14,
8118.66,
8534.44,
8521.66,
8991.16,
9674.74,
9006.86,
9419.34,
8563.12,
7235.08,
8532.64,
9193.78,
8202.38,
8798.68,
8777.18,
9772.92,
8519.3,
9129.12,
8309.76,
8921.48,
9061.34,
8190.1,
8310.64,
8500.62,
8995.96,
8497.6,
8194.48,
8452.54,
8631.94,
7949.76,
8320.04,
9274.94,
8529.02,
8800.06,
9367.9,
9582.12,
9683.0,
7988.0,
7068.04,
8933.02,
8675.12,
8699.72,
```

```
8104.24,
8348.18,
8024.38,
8522.14,
7430.34,
7887.64,
9016.58,
9597.66,
8686.92,
9698.46,
8377.98,
8555.92,
9600.86,
9661.16,
8398.48,
9131.78,
9046.06,
9421.14,
8105.34,
7841.78,
8178.18,
8727.92,
8614.8,
7522.78,
8890.32,
7854.04,
7877.76,
10154.94,
8650.12,
9259.68,
9695.54,
10846.6,
9753.2,
8535.1,
9721.16,
9206.54,
8580.58,
8687.08,
8629.52,
8417.98,
8276.9,
7701.3,
8520.28,
8787.14,
8011.76,
8391.52,
8698.4,
9237.9,
8518.94,
9058.18,
8188.32,
8231.08,
8095.2,
8355.22,
7358.4,
8393.8,
8600.98,
9098.84,
9008.24,
8554.64,
```

```
7751.14,
9181.26,
9621.7,
8171.96,
8839.72,
7903.2,
9697.7,
9339.68,
8437.48,
10449.36,
8391.06,
8910.92,
9187.68,
10329.88,
8228.16,
9558.62,
7871.24,
9214.22,
8919.28,
8537.44,
8193.12,
7989.2,
8034.52,
8524.74,
9736.48,
9540.8,
9397.44,
8977.32,
8915.0,
8264.54,
9886.76,
9480.44,
9306.38,
9718.3,
8141.34,
8720.44,
8003.28,
9446.02,
7228.42,
8744.9,
7777.5,
8923.08,
8480.44,
9147.76,
8362.1,
8636.9,
8937.88,
9057.8,
8467.88,
8469.06,
8583.46,
10063.2,
8413.82,
10133.62,
7468.16,
8596.1,
7621.18,
8381.26,
9249.68,
8601.18,
```

```
            9205.36,
            9630.32,
            9335.78,
            9705.6,
            7298.4,
            9540.02,
            9047.14,
            10077.46,
            7971.64,
            7873.52,
            7222.9,
            8874.02,
            8998.04,
            9387.5,
            10298.92,
            8474.76,
            7911.08,
            8768.72,
            8407.92,
            8744.3,
            8691.52,
            9249.58,
            8746.1,
            8197.12,
            9107.34,
            8516.4,
            9789.8,
            10144.54,
            9065.52,
            8063.46,
            8796.52,
            9450.04,
            7836.18,
            8548.06,
            8035.36,
            7967.64,
            8868.48,
            9132.3,
            8756.02,
            9758.42,
            10199.32,
            8599.66,
            8397.88,
            9676.82,
            9586.62,
            8911.84,
            8038.12,
            8330.66,
            7593.7,
            9394.48,
            9317.14,
            8988.02,
            9526.9,
            8371.26,
            8286.24,
            9335.74,
            8144.56,
            9213.6,
            8304.06,
            7757.96,
```

```
8611.08,
9492.46,
7997.94,
8744.12,
9077.38,
7270.38,
8483.86,
8137.54,
8206.8,
8749.72,
9097.06,
8998.34,
8180.74,
7619.28,
9756.16,
8904.58,
9192.1,
8398.44,
8094.7,
7067.0,
8271.18,
8371.9,
9532.58,
8852.34,
8158.38,
8852.98,
8920.52,
9417.22,
9650.94,
8421.94,
8683.18,
8051.76,
9282.12,
8580.22,
9536.52,
8257.76,
8640.54,
8214.1,
9886.0,
8961.78,
8563.24,
8720.44,
9036.86,
8885.1,
9200.44,
7802.38,
8636.56,
9698.82,
8803.76,
8764.82,
8294.58,
7668.48,
9045.48,
8087.38,
8659.94,
8618.84,
9094.68,
7807.28,
9333.62,
8971.72,
```

```
8284.06,
8869.88,
8145.14,
9785.86,
8099.7,
7642.62,
9149.06,
8391.04,
9847.96,
10006.8,
9613.88,
8500.44,
8591.62,
9502.28,
8600.5,
8673.82,
8701.92,
8969.28,
8573.96,
8806.58,
8355.52,
8335.18,
9136.84,
8839.22,
9419.92,
8905.36,
9192.56,
9318.56,
8497.92,
8769.82,
8404.24,
7863.24,
7242.14,
8600.22,
8719.94,
9253.1,
9751.3,
7804.5,
8248.88,
9395.86,
8211.02,
9515.74,
10263.34,
9079.76,
8971.36,
8217.56,
9419.76,
10091.04,
8953.52,
8271.22,
9292.44,
7654.92,
9210.6,
8081.2,
9331.9,
8644.16,
8296.52,
8715.42,
7957.62,
8969.18,
```

```
8850.92,
8869.4,
8343.36,
8931.1,
8714.48,
7901.18,
8302.14,
8903.3,
8980.54,
9556.98,
7571.54,
8636.38,
9604.3,
9129.36,
8579.7,
8560.86,
8630.46,
8070.88,
9163.8,
8501.5,
9077.92,
8906.76,
9114.08,
8013.94,
9397.34,
7525.3,
7380.26,
10025.5,
7606.56,
8330.38,
11004.9,
8353.56,
8295.52,
8329.68,
8589.66,
9102.46,
9176.02,
9675.26,
9224.86,
9437.68,
8801.66,
9487.56,
8336.46,
8574.32,
7474.34,
8897.04,
9567.74,
9212.14,
8820.72,
8465.98,
8994.08,
8704.68,
9507.42,
8577.92,
9111.5,
8331.46,
9398.74,
7707.94,
9170.48,
9073.16,
```

```
8589.12,
9454.3,
8808.24,
8339.12,
8138.5,
8101.3,
8699.2,
9376.62,
8696.16,
8323.16,
8320.14,
8995.58,
8616.66,
9590.16,
8605.74,
7880.92,
8351.04,
8404.86,
8335.26,
9546.24,
7477.98,
8354.92,
8055.04,
8583.54,
9020.04,
7666.68,
8966.5,
9274.14,
8048.12,
8484.1,
8153.0,
7161.66,
9191.16,
9108.44,
8388.4,
8554.86,
8773.58,
9637.96,
8135.1,
9471.54,
8954.34,
8434.84,
8297.08,
7892.24,
7937.1,
8330.58,
9184.42,
8418.9,
8658.08,
10485.44,
8782.18,
9175.74,
8452.96,
9981.74,
8217.64,
8859.3,
8120.28,
8454.9,
9116.98,
8164.7,
```

```
10002.96,
8266.46,
10645.64,
8384.4,
8079.0,
8588.5,
8982.32,
6710.86,
9625.48,
8523.26,
8974.44,
8441.28,
10472.32,
8517.16,
8472.28,
9334.74,
8209.92,
9134.04,
8911.12,
9121.56,
7702.36,
9742.16,
9161.88,
9402.38,
10076.5,
9060.08,
9514.26,
8396.88,
8583.66,
8433.92,
8651.26,
8446.68,
8188.12,
8902.18,
8791.66,
10163.58,
7971.92,
10379.88,
8894.4,
10009.38,
8294.5,
9220.3,
8109.68,
9139.74,
8318.94,
8241.9,
8242.16,
8400.4,
8891.46,
10314.86,
8070.78,
9845.36,
8268.1,
9202.88,
7836.12,
7141.02,
8951.08,
9611.8,
7996.12,
8678.1,
```

```
            9054.44,
            9783.88,
            8594.98,
            8947.16,
            8880.72,
            8426.42,
            8814.32,
            8062.18,
            9182.12,
            8713.34,
            8315.28,
            10035.76,
            8823.86,
            8173.32,
            9664.98,
            8744.76,
            8968.06,
            7282.96,
            9496.36,
            7788.14,
            9351.5,
            9161.28,
            7662.14,
            7991.32,
            9573.3,
            9032.26,
            9610.68,
            9556.26,
            7718.88,
            8661.72,
            9622.9,
            8365.82,
            8364.1,
            9564.58,
            9050.04,
            9837.68,
            7279.34,
            9188.04,
            8040.42,
            9909.58,
            8862.1,
            8728.86,
            9245.8,
            8232.44,
            8304.42,
            9165.9,
            8526.78,
            8446.62,
            7729.78,
            9631.26,
            8098.6,
            8112.22,
            8319.88,
            8509.52,
            7911.84,
            9034.68,
            9279.78,
            8153.76,
            7845.28,
            8079.08,
```

```
                9192.76,
                9035.52,
                8807.48,
                8948.7,
                8367.5,
                9251.5,
                10183.96,
                8297.94,
                8619.14,
                9528.58,
                9983.6,
                8244.76,
                7303.68,
                8580.38,
                8742.06,
                7316.54,
                9225.76,
                8523.32,
                7942.62,
                8929.18,
                8765.22,
                9362.48,
                8893.02,
                9203.72,
                8671.56,
                8794.94,
                7492.94,
                8073.78,
                8040.48,
                7910.04,
                8564.56,
                8041.42,
                9501.56,
                8410.84,
                8713.46,
                8618.44,
                7417.82,
                8412.18,
                8661.08,
                8601.5,
                9136.54,
                9112.42,
                8738.18,
                7616.7,
                8481.96,
                8747.82,
                9146.78,
                9192.46,
                7693.2,
                7928.92,
                8496.04,
                8859.62,
                9249.5,
                8700.94,
                8247.42,
                8273.78,
                8345.16,
                9150.66,
                9783.64,
                7721.26,
```

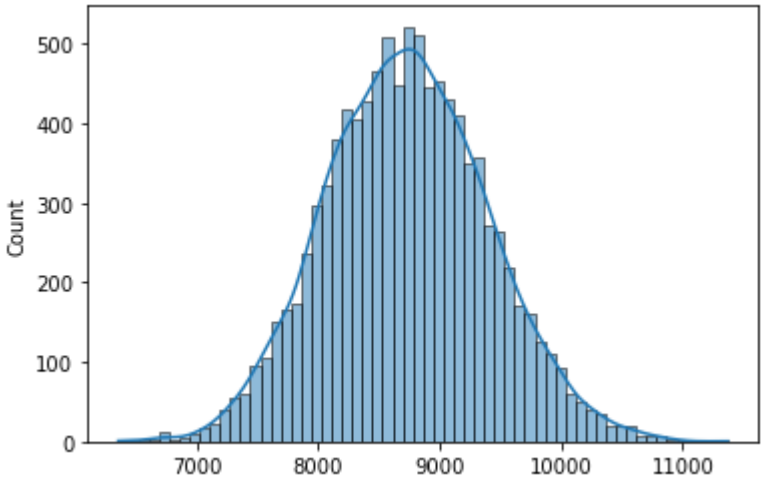
```
9209.44,
8752.74,
7366.88,
8184.4,
8979.02,
9503.94,
8848.32,
8863.12,
9964.9,
7747.24,
8889.72,
8290.6,
8645.46,
8604.48,
8413.02,
9266.92,
8668.62,
8751.3,
8972.24,
8493.62,
7500.06,
9505.24,
9231.34,
8421.06,
9828.68,
8552.78,
9333.0,
8436.22,
8682.56,
9618.46,
8753.34,
8473.36,
9461.06,
8529.14,
8445.98,
7252.02,
8866.86,
8974.38,
8482.72,
9615.4,
8538.78,
8312.72,
8648.22,
8480.86,
8854.16,
9130.74,
9209.88,
8776.64,
10539.86,
9343.56,
8878.68,
8158.92,
9742.5,
8309.84,
8466.52,
8766.74,
8801.3,
8520.7,
8994.36,
9513.6,
```

```
9377.78,
9287.4,
9352.88,
9067.6,
9701.78,
9267.66,
8993.84,
7250.54,
8796.58,
8134.92,
8759.5,
9334.06,
8769.82,
8354.56,
8808.7,
8990.44,
9791.08,
9086.98,
8857.4,
9449.58,
9382.82,
9225.58,
7522.96,
8608.96,
9747.78,
8779.56,
8875.68,
8997.8,
9284.38,
9217.28,
8691.58,
8751.1,
8963.5,
9156.44,
9262.44,
9700.22,
9089.24,
9299.7,
8575.8,
9352.84,
9008.38,
7587.62,
9988.54,
7432.28,
7707.4,
8803.86,
8315.5,
9296.46,
8848.24,
8950.96,
7835.42,
8080.34,
8455.22,
8502.6,
9496.56,
9182.14,
8788.64,
8061.1,
8375.94,
8333.86,
```

```
9167.34,
9173.34,
9005.84,
7950.9,
9654.8,
8083.34,
9254.16,
8525.08,
9149.98,
9212.08,
8411.14,
8246.14,
8816.52,
9588.98,
8460.12,
8872.24,
9531.64,
9156.08,
8750.44,
8362.96,
7780.16,
7478.88,
8982.22,
8061.08,
7364.22,
9354.28,
8820.16,
8609.72,
8556.84,
8887.56,
8720.84,
7990.48,
8736.88,
9361.02,
9945.18,
8573.22,
8562.34,
10362.2,
8390.68,
8044.34,
10361.82,
8576.38,
8621.38,
9553.0,
9275.2,
8136.86,
10018.74,
9098.1,
7719.36,
9153.0,
8209.64,
8382.76,
8501.88,
7615.8,
8829.14,
7606.44,
7897.14,
8991.16,
8036.34,
9231.74,
```

```
9288.54,
9147.86,
7884.14,
9905.18,
9473.74,
8169.44,
9540.94,
8748.88,
8210.26,
9205.02,
7355.92,
8743.42,
7802.92,
9459.16,
7963.16,
9376.6,
8734.8,
9020.84,
8517.22,
8263.66,
8290.24,
8003.94,
10072.9,
8188.34,
7530.66,
8283.0,
9192.58,
7541.16,
7803.62,
8518.04,
8877.46,
8010.18,
8837.34,
10093.94,
8006.6,
7985.78,
8659.68,
9877.34,
9361.88,
9226.14,
...]
```

In [81]:
```
sns.histplot(data=sample_walmart_female,kde=True)
plt.show()
```



It is a gaussian distribution

In [88]:
```
sample_walmart_female_mean=np.mean(sample_walmart_female)
sample_walmart_female_mean
```

Out[88]:
```
8730.132466000001
```

In [89]:
```
sample_walmart_female_std=np.std(sample_walmart_female)
sample_walmart_female_std
```

Out[89]:
```
665.1526256067842
```

In [90]:
```
np.percentile(sample_walmart_female,2.5)
```

Out[90]:
```
7464.67
```

In [91]:
```
np.percentile(sample_walmart_female,97.5)
```

Out[91]:
```
10063.2775
```

The values obtained from CLT for female purchases are: Mean:8734.56 , 95% CI(7413.180395716255, 10055.951134594696)

The values obtained from sample distribution of mean for female purchases are: Mean:8730.132466000001 , 95% CI (7464.67,10063.2775)

The values obtained from both appling CLT and by creating sample distribution of means gives the same result for all mean and confidence interval. It proves that just by applying CLT we can solve complex problems of probability distribution without applying bootstrapping method.

CLT and Confidence Interval for Married and Unmarried People

In [50]:
```
walmart.head()
```

Out[50]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_
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	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	

In [52]:
```
walmart_unmarried=walmart.loc[walmart["Marital_Status"]==0,"Purchase"]
walmart_unmarried
```

Out[52]:
```
0          8370
1         15200
2          1422
3          1057
4          7969
          ...
550056      254
550059       48
550062      473
550064      371
550066      365
Name: Purchase, Length: 324731, dtype: int64
```

In [53]:
```
walmart_married=walmart.loc[walmart["Marital_Status"]==1,"Purchase"]
walmart_married
```

Out[53]:
```
6         19215
7         15854
8         15686
9          7871
10         5254
          ...
550060      494
550061      599
550063      368
550065      137
550067      490
Name: Purchase, Length: 225337, dtype: int64
```

As creating a sample of 50 million married and 50 million unmarried customers using bootstrap would take higher time, we will assume a sample distribution of means of 50 samples to create a population of 50 million.

As Central limit theorem states that the sample distribution of means will be a Gaussian distribution given that we consider a bigger sample.If Sample_size>30, we can assume Gaussian Distribution.

Married Calculations

In [55]:
```
married_sample_mean=walmart_married.mean()
married_sample_mean
```

Out[55]:
```
9261.174574082374
```

In [56]:
```
married_population_std=walmart_married.std()
```

In [57]:
```
married_sample_std=married_population_std/np.sqrt(50)
```

90% Confidence Interval

In [58]:
```
z_90_lower_married=norm.ppf(0.05)
```

In [59]:
```
z_90_upper_married=norm.ppf(0.95)
```

In [60]:
```
married_90pct_confidence=(married_sample_mean+z_90_lower_married*married_sample_std,ma
married_90pct_confidence
```

Out[60]:
```
(8094.1567957972975, 10428.192352367449)
```

95% Confidence Interval

In [61]:
```
z_95_lower_married=norm.ppf(0.025)
```

In [62]:
```
z_95_upper_married=norm.ppf(0.975)
```

In [63]:
```
married_95pct_confidence=(married_sample_mean+z_95_lower_married*married_sample_std,ma
married_95pct_confidence
```

Out[63]:
```
(7870.587121631699, 10651.762026533048)
```

99% Confidence Interval

In [64]:
```
z_99_lower_married=norm.ppf(0.005)
```

In [65]:
```
z_99_upper_married=norm.ppf(0.995)
```

In [66]:
```
married_99pct_confidence=(married_sample_mean+z_99_lower_married*married_sample_std,ma
married_99pct_confidence
```

Out[66]:
```
(7433.632875651405, 11088.716272513342)
```

Unmarried Calculations

In [67]:
```
unmarried_sample_mean=walmart_unmarried.mean()
unmarried_sample_mean
```

Out[67]:
```
9265.907618921507
```

In [68]:
```
unmarried_population_std=walmart_unmarried.std()
```

In [69]:
```
unmarried_sample_std=unmarried_population_std/np.sqrt(50)
```

90% Confidence Interval

In [70]:
```
unmarried_90pct_confidence=(unmarried_sample_mean+z_90_lower*unmarried_sample_std,unma
unmarried_90pct_confidence
```

Out[70]:
```
(8096.458876623868, 10435.356361219145)
```

95% Confidence Interval

In [71]:
```
unmarried_95pct_confidence=(unmarried_sample_mean+z_95_lower*unmarried_sample_std,unma
unmarried_95pct_confidence
```

Out[71]:
```
(7872.423494186887, 10659.391743656124)
```

99% Confidence Interval

In [72]:
```
unmarried_99pct_confidence=(unmarried_sample_mean+z_99_lower*unmarried_sample_std,unma
unmarried_99pct_confidence
```

Out[72]:
```
(7434.5590478078675, 11097.256190035145)
```

Married Purchases have a mean of 9261.17 with 95% Confidence Interval in range (7870.587121631699, 10651.762026533048)

Unmarried Purchases have a mean of 9265.90 with 95% Confidence Interval in range (7872.423494186887, 10659.391743656124)

As the 95% confidence interval of married and unmarried purchases **Coincides** with each other, the purchases of male and female are **statistically insignificant**

CLT and Confidence Interval for different age band of people

In [92]:
```
walmart["Age"].unique()
```

Out[92]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

In [94]:
```
walmart_seventeen=walmart.loc[walmart["Age"]=='0-17',"Purchase"]
```

In [95]:
```
walmart_twentyfive=walmart.loc[walmart["Age"]=='18-25',"Purchase"]
```

In [96]:
```
walmart_thirtyfive=walmart.loc[walmart["Age"]=='26-35',"Purchase"]
```

In [97]:
```
walmart_fortyfive=walmart.loc[walmart["Age"]=='36-45',"Purchase"]
```

In [98]:
```
walmart_fifty=walmart.loc[walmart["Age"]=='46-50',"Purchase"]
```

In [99]:
```
walmart_fiftyfive=walmart.loc[walmart["Age"]=='51-55',"Purchase"]
```

In [100…
```
walmart_fiftyfiveplus=walmart.loc[walmart["Age"]=='55+',"Purchase"]
```

In [101…
```
walmart_seventeen
```

Out[101]:
```
0          8370
1         15200
2          1422
3          1057
85         7746
          ...
549904      256
550012       26
550024       12
550035       61
550046      236
Name: Purchase, Length: 15102, dtype: int64
```

In [102…
```
walmart_twentyfive
```

Out[102]:
```
70         1780
71        10754
72         2802
73        19473
74        19672
          ...
550000       14
550015      477
550017      363
550020       36
550032      491
Name: Purchase, Length: 99660, dtype: int64
```

In [103…
```
walmart_thirtyfive
```

Out[103]:
```
5         15227
9          7871
10         5254
11         3957
12         6073
          ...
550058      121
550059       48
550061      599
550064      371
550065      137
Name: Purchase, Length: 219587, dtype: int64
```

In [104…
```
walmart_fortyfive
```

Out[104]:
```
18        11788
29        16352
30         8886
31         5875
32         8854
          ...
550049      473
550050      368
550053      371
550054       60
550060      494
Name: Purchase, Length: 110013, dtype: int64
```

In [105…
```
walmart_fifty
```

Out[105]:
```
6         19215
7         15854
8         15686
52         5839
53        15912
          ...
550041      488
550043       48
550052      239
550062      473
550067      490
Name: Purchase, Length: 45701, dtype: int64
```

In [106…
```
walmart_fiftyfive
```

Out[106]:
```
14         5378
15         2079
16        13055
17         8851
67        15872
          ...
549985       24
550004       12
550037       62
550042      243
550063      368
Name: Purchase, Length: 38501, dtype: int64
```

In [107…
```
walmart_fiftyfiveplus
```

Out[107]:
```
4          7969
159        8596
160        5248
161       10592
162        3482
          ...
549925      121
549989       12
550008       50
550030      376
550066      365
Name: Purchase, Length: 21504, dtype: int64
```

Walmart 0-17 calculations using CLT and confidence interval

In [108…
```
seventeen_sample_mean=walmart_seventeen.mean()
```

In [109…
```
seventeen_sample_mean
```

Out[109]:
```
8933.464640444974
```

In [112…
```
seventeen_population_std=walmart_seventeen.std()
```

In [113…
```
seventeen_sample_std=seventeen_population_std/np.sqrt(50)
```

90% Confidence Interval

In [114…
```
seventeen_90pct_confidence=(seventeen_sample_mean+z_90_lower*seventeen_sample_std,seve
```

In [115…
```
seventeen_90pct_confidence
```

Out[115]:
```
(7744.530422868527, 10122.39885802142)
```

95% Confidence Interval

In [116…
```
seventeen_95pct_confidence=(seventeen_sample_mean+z_95_lower*seventeen_sample_std,seve
```

In [117…
```
seventeen_95pct_confidence
```

Out[117]:
```
(7516.762139836455, 10350.167141053493)
```

99% Confidence Interval

In [118…
```
seventeen_99pct_confidence=(seventeen_sample_mean+z_99_lower*seventeen_sample_std,seve
```

In [119…
```
seventeen_99pct_confidence
```

Out[119]:
```
(7071.601950996227, 10795.32732989372)
```

Walmart 18-26 calculations using CLT and confidence interval

In [121…
```
twentyfive_sample_mean=walmart_twentyfive.mean()
```

In [122…
```
twentyfive_sample_mean
```

Out[122]:
```
9169.663606261289
```

In [123…
```
twentyfive_population_std=walmart_twentyfive.std()
```

In [125…
```
twentyfive_sample_std=twentyfive_population_std/np.sqrt(50)
```

90% Confidence Interval

In [126…
```
twentyfive_90pct_confidence=(twentyfive_sample_mean+z_90_lower*twentyfive_sample_std,t
```

In [127…
```
twentyfive_90pct_confidence
```

Out[127]:
```
(7998.592557783694, 10340.734654738882)
```

95% Confidence Interval

In [128…
```
twentyfive_95pct_confidence=(twentyfive_sample_mean+z_95_lower*twentyfive_sample_std,t
```

In [129…
```
twentyfive_95pct_confidence
```

Out[129]:
```
(7774.246384482343, 10565.080828040234)
```

99% Confidence Interval

In [130…
```
twentyfive_99pct_confidence=(twentyfive_sample_mean+z_99_lower*twentyfive_sample_std,t
```

In [131…
```
twentyfive_99pct_confidence
```

Out[131]:
```
(7335.774514988781, 11003.552697533796)
```

Walmart 26-35 Calculations using CLT and Confidence Interval

In [132…
```
thirtyfive_sample_mean=walmart_thirtyfive.mean()
```

In [133…
```
thirtyfive_sample_std=walmart_thirtyfive.std()/np.sqrt(50)
```

90% Confidence Interval

In [134…
```
thirtyfive_90pct_confidence=(thirtyfive_sample_mean+z_90_lower*thirtyfive_sample_std,t
```

In [135…
```
thirtyfive_90pct_confidence
```

Out[135]:
```
(8087.1546450161, 10418.226620723675)
```

95% Confidence Interval

In [136…
```
thirtyfive_95pct_confidence=(thirtyfive_sample_mean+z_95_lower*thirtyfive_sample_std,t
```

In [137…
```
thirtyfive_95pct_confidence
```

Out[137]:
```
(7863.868842621841, 10641.512423117934)
```

99% Confidence Interval

In [138…
```
thirtyfive_99pct_confidence=(thirtyfive_sample_mean+z_99_lower*thirtyfive_sample_std,t
```

In [139…
```
thirtyfive_99pct_confidence
```

Out[139]: (7427.469407915763, 11077.911857824012)

walmart 36-45 Calculations using CLT and Confidence Interval

In [140…
```
fortyfive_sample_mean=walmart_fortyfive.mean()
```

In [141…
```
fortyfive_sample_std=walmart_fortyfive.std()/np.sqrt(50)
```

90% Confidence Interval

In [142…
```
fortyfive_90pct_confidence=(fortyfive_sample_mean+z_90_lower*fortyfive_sample_std,fort
```

In [143…
```
fortyfive_90pct_confidence
```

Out[143]: (8162.931047358126, 10499.770342477621)

95% confidence Interval

In [144…
```
fortyfive_95pct_confidence=(fortyfive_sample_mean+z_95_lower*fortyfive_sample_std,fort
```

In [145…
```
fortyfive_95pct_confidence
```

Out[145]: (7939.092812196464, 10723.608577639283)

99% Confidence Interval

In [146…
```
fortyfive_99pct_confidence=(fortyfive_sample_mean+z_99_lower*fortyfive_sample_std,fort
```

In [147…
```
fortyfive_99pct_confidence
```

Out[147]: (7501.613678983709, 11161.087710852038)

Walmart 45-50 calculations using CLT and Confidence Interval

In [148…
```
fifty_sample_mean=walmart_fifty.mean()
```

In [149…
```
fifty_sample_std=walmart_fifty.std()/np.sqrt(50)
```

90% Confidence Interval

In [150…
```
fifty_90pct_confidence=(fifty_sample_mean+z_90_lower*fifty_sample_std,fifty_sample_mea
```

In [151…
```
fifty_90pct_confidence
```

Out[151]: (8053.164588237038, 10364.086806699617)

95% Confidence Interval

In [152…
```
fifty_95pct_confidence=(fifty_sample_mean+z_95_lower*fifty_sample_std,fifty_sample_mea
```

In [153…
```
fifty_95pct_confidence
```
Out[153]:
```
(7831.80886554919, 10585.442529387465)
```

99% Confidence Interval

In [154…
```
fifty_99pct_confidence=(fifty_sample_mean+z_99_lower*fifty_sample_std,fifty_sample_mea
```

In [155…
```
fifty_99pct_confidence
```
Out[155]:
```
(7399.181662143912, 11018.069732792743)
```

Walmart 50-55 calculations using CLT and COnfidence Interval

In [156…
```
fiftyfive_sample_mean=walmart_fiftyfive.mean()
```

In [157…
```
fiftyfive_sample_std=walmart_fiftyfive.std()/np.sqrt(50)
```

90% Confidence Interval

In [158…
```
fiftyfive_90pct_confidence=(fiftyfive_sample_mean+z_90_lower*fiftyfive_sample_std,fift
```

In [159…
```
fiftyfive_90pct_confidence
```
Out[159]:
```
(8351.397539078236, 10718.218522842233)
```

95% Confidence Interval

In [160…
```
fiftyfive_95pct_confidence=(fiftyfive_sample_mean+z_95_lower*fiftyfive_sample_std,fift
```

In [161…
```
fiftyfive_95pct_confidence
```
Out[161]:
```
(8124.687455481226, 10944.928606439245)
```

99% Confidence Interval

In [162…
```
fiftyfive_99pct_confidence=(fiftyfive_sample_mean+z_99_lower*fiftyfive_sample_std,fift
```

In [163…
```
fiftyfive_99pct_confidence
```
Out[163]:
```
(7681.595457409579, 11388.020604510892)
```

Walmart 55+ Calculations using CLT and Confidence Interval

In [164…
```
fiftyfiveplus_sample_mean=walmart_fiftyfiveplus.mean()
```

In [165…
```
fiftyfiveplus_sample_std=walmart_fiftyfiveplus.std()/np.sqrt(50)
```

90 % Confidence Interval

In [168…
```
fiftyfiveplus_90pct_confidence=(fiftyfiveplus_sample_mean+z_90_lower*fiftyfiveplus_sam
```

In [169…
```
fiftyfiveplus_90pct_confidence
```

Out[169]:
```
(8170.51960204658, 10502.041316852228)
```

95% confidence Interval

In [170…
```
fiftyfiveplus_95pct_confidence=(fiftyfiveplus_sample_mean+z_95_lower*fiftyfiveplus_sam
```

In [171…
```
fiftyfiveplus_95pct_confidence
```

Out[171]:
```
(7947.190720606959, 10725.37019829185)
```

99% Confidence Interval

In [172…
```
fiftyfiveplus_99pct_confidence=(fiftyfiveplus_sample_mean+z_99_lower*fiftyfiveplus_sam
```

In [173…
```
fiftyfiveplus_99pct_confidence
```

Out[173]:
```
(7510.707090350419, 11161.85382854839)
```

As all the values for confidence interval 95% for different age bands coincide with each other, the difference is statistically insignificant

Insights:

- It is found that the data doesnt indicate any statistical significant difference between male and female spending in Black friday Sale in Walmart
- It is noted that there is not any statistical significant difference between married and unmarried spending .
- Also there is not any statistical difference for different age band spending . However the turnout of people in age bracket 18-45 is much higher than rest others.
- The distribution of population for spending is not a Gaussian distribution, however the sample distribution of means given that the sample is large enough follows gaussian distribution and hence all the calculations can be done using central limit theorem.

Recommendations:

- It is observed that the customer staying in particular city for around 1 year has the highest footfall in the Walmart store. So by providing some offers to them like early access, the sale further can be inreased.
- The people who have stayed in city for more than 1+ years whose footfall is little lesser than those around 1 year in city, their footfall can further be increased by providing some discounts.

- Targeted segmentation can also be incorporated by providing discounts to people of certain age band such as 18-25 years as their footfall is quite higher than rest.
- Equal spend on all genders for marketing as their spend is more or less same with male spending is little higher than female spending using their averages per transaction.
- Unmarried people tend to visit more often than married people. so by introducing some kind of couple products or couple games during entry, the married peoples' visits cam also be increased.

In [ ]: