Business Case:Walmart-Confidence Interval and CLT

About Walmart:-

Walmart is an American multinational retail corporation that operates a chain of supercenters, discount departmental stores, and grocery stores from the United States. Walmart has more than 100 million customers worldwide.

Business Problem :-

Walmart is an American multinational retail corporation that operates a chain of supercenters, discount departmental stores, and grocery stores from the United States. Walmart has more than 100 million customers worldwide.

Importing Python Libraries necessary while carrying out data exploration & visualisation

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import math
from scipy.stats import binom,norm ,geom
import warnings
warnings.filterwarnings("ignore")
```

Upload & read csv file in pandas dataframe -

In [ ]:

```
df=pd.read_csv("/content/walmart_data.txt")
```

Inspecting Dataset and Analyzing Different Metrics:

In [ ]:

```
df.head()
```

Out[4]:

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

In [ ]:

```
df.tail()
```

Out[5]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
550063	1006033	P00372445	M	51-55	13	B	1	1	20	368
550064	1006035	P00375436	F	26-35	1	C	3	0	20	371
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	137
550066	1006038	P00375436	F	55+	1	C	2	0	20	365
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	490

Observation On

Shape of Data

Data types

Statistical Summary

In [ ]:
```
df.shape
```
Out[6]:
```
(550068, 10)
```

In [ ]:
```
df.size
```
Out[7]:
```
5500680
```

In [ ]:
```
df.columns
```
Out[8]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

In [ ]:
```
df.nunique()
```
Out[9]:
```
User_ID                        5891
Product_ID                     3631
Gender                            2
Age                               7
Occupation                       21
City_Category                     3
Stay_In_Current_City_Years        5
Marital_Status                    2
Product_Category                 20
Purchase                      18105
dtype: int64
```

In [ ]:
```
df.dtypes
```
Out[10]:
```
User_ID                        int64
Product_ID                    object
Gender                        object
Age                           object
Occupation                     int64
City_Category                 object
Stay_In_Current_City_Years    object
Marital_Status                 int64
Product_Category               int64
Purchase                       int64
dtype: object
```

In [ ]:
```
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 [ ]:
```
df.describe()
```
Out[12]:

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

In [ ]:
```
df.describe(include=object)
```

Out[13]:

	Product_ID	Gender	Age	City_Category	Stay_In_Current_City_Years
count	550068	550068	550068	550068	550068
unique	3631	2	7	3	5
top	P00265242	M	26-35	B	1
freq	1880	414259	219587	231173	193821

Data Cleaning-

checking for missing values and duplicates

In [ ]:
```
df.isnull().sum().sort_values(ascending=True)
```

Out[14]:
```
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 [ ]:
```
df[df.duplicated()]
```

Out[15]:

User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase

Comment

No null value or duplicate value present in dataset

Non Graphical Analysis

In [ ]:
```
df.head()
```

Out[16]:

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

In [ ]:
```
# user_id wise unique values and count
df["User_ID"].unique()
```

Out[17]:
```
array([1000001, 1000002, 1000003, ..., 1004113, 1005391, 1001529])
```

In [ ]:
```
df["User_ID"].nunique()
```

Out[18]:
```
5891
```

In [ ]:
```
df["User_ID"].value_counts()
```

Out[19]:
```
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
```

Comment

We have 5891 enteries of user_id

Top three user_id are 1001680,1004277,1001941

In [ ]:
```
# Product wise unique values and count
df["Product_ID"].unique()
```

Out[20]:
```
array(['P00069042', 'P00248942', 'P00087842', ..., 'P00370293',
       'P00371644', 'P00370853'], dtype=object)
```

In [ ]:
```
df["Product_ID"].nunique()
```

Out[21]:
```
3631
```

In [ ]:
```
df["Product_ID"].value_counts()
```

Out[22]:
```
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
```

Comment

Walmart supermarket has 3631 different products.

Top three most demanding products are P00265242, P00025442, P00220742.

In [ ]:
```
# genderwise unique values and counts
df["Gender"].nunique()
```

Out[23]:
```
2
```

In [ ]:
```
df["Gender"].unique()
```

Out[24]:
```
array(['F', 'M'], dtype=object)
```

In [ ]:
```
df["Gender"].value_counts(normalize=True).round(2)*100
```

Out[25]:
```
M    75.0
F    25.0
Name: Gender, dtype: float64
```

Comment

we have 75% male customer and 25% female customer.

In [ ]:
```
# Agewise unique values and count
df['Age'].unique()
```

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

In [ ]:
```
df['Age'].nunique()
```

Out[27]:
```
7
```

In [ ]:
```
df["Age"].value_counts(normalize=True).round(2)*100
```

Out[28]:
```
26-35    40.0
36-45    20.0
18-25    18.0
46-50     8.0
51-55     7.0
55+       4.0
0-17      3.0
Name: Age, dtype: float64
```

Comment

We have customer from the age group 0 to 55+

Most of the customers are in the age group of 26-35(40%) followed by 36-45(20%)

In [ ]:
```
# occuptionwise unique values and counts
df["Occupation"].nunique()
```

Out[29]:
```
21
```

In [ ]:
```
df["Occupation"].unique()
```

Out[30]:
```
array([10, 16, 15,  7, 20,  9,  1, 12, 17,  0,  3,  4, 11,  8, 19,  2, 18,
        5, 14, 13,  6])
```

In [ ]:
```
df["Occupation"].value_counts(normalize=True).round(2)*100
```

Out[31]:
```
4     13.0
0     13.0
7     11.0
1      9.0
17     7.0
20     6.0
12     6.0
14     5.0
2      5.0
16     5.0
6      4.0
3      3.0
10     2.0
5      2.0
15     2.0
11     2.0
19     2.0
13     1.0
18     1.0
9      1.0
8      0.0
Name: Occupation, dtype: float64
```

Walmart have customers with occupation experience range from 0 to 20

Most customer of the walmart are having occupation with experience of 4,0,7.

In [ ]:
```
# citywise unique values and counts
df["City_Category"].unique()
```

Out[32]:
```
array(['A', 'C', 'B'], dtype=object)
```

In [ ]:
```
df["City_Category"].nunique()
```

Out[33]:
```
3
```

In [ ]:
```
df["City_Category"].value_counts(normalize=True).round(2)*100
```

Out[34]:
```
B    42.0
C    31.0
A    27.0
Name: City_Category, dtype: float64
```

Comment

All the cities were divided into three categories.

Most of the customer are from city_category-B followed by C

In [ ]:
```
# Current statewise Unique values and count.
df["Stay_In_Current_City_Years"].unique()
```

Out[35]:
```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

In [ ]:
```
df["Stay_In_Current_City_Years"].nunique()
```

Out[36]:
```
5
```

In [ ]:
```
df["Stay_In_Current_City_Years"].value_counts(normalize=True).round(2)*100
```

Out[37]:
```
1     35.0
2     19.0
3     17.0
4+    15.0
0     14.0
Name: Stay_In_Current_City_Years, dtype: float64
```

Comment

Most(35%) of the customer are staying in the particular city_category for 1 yr followed by 19% customer stay in a particular city _category for 2yrs.

In [ ]:
```
# Marital status wise count and unique value
df["Marital_Status"].unique()
```

Out[38]:
```
array([0, 1])
```

In [ ]:
```
df["Marital_Status"].nunique()
```

Out[39]:
```
2
```

In [ ]:
```
df["Marital_Status"].value_counts(normalize=True).round(2)*100
```

Out[40]:
```
0    59.0
1    41.0
Name: Marital_Status, dtype: float64
```

Comment

Marital status is divided into two category:"0" refers single and "1" refer married .

Most of the customer are single(59%) followed by married(41%)

In [ ]:
```

# Product_category wise count and unique values
df["Product_Category"].unique()
```

Out[41]:
```
array([ 3,  1, 12,  8,  5,  4,  2,  6, 14, 11, 13, 15,  7, 16, 18, 10, 17,
        9, 20, 19])
```

In [ ]:
```
df["Product_Category"].nunique()
```

Out[42]:
```
20
```

In [ ]:
```
df["Product_Category"].value_counts(normalize=True).round(2)*100
```

Out[43]:
```
5     27.0
1     26.0
8     21.0
11     4.0
2      4.0
6      4.0
3      4.0
4      2.0
16     2.0
15     1.0
13     1.0
10     1.0
12     1.0
7      1.0
18     1.0
20     0.0
19     0.0
14     0.0
17     0.0
9      0.0
Name: Product_Category, dtype: float64
```

Comment

Walmart have 20 different Product categories in their stores.

Product_category with 5,1,8 are top three among 20 in walmart inventory.

In [ ]:
```
# Purchasewise unique values and count.
df["Purchase"].unique()
```

Out[44]:
```
array([ 8370, 15200,  1422, ...,   135,   123,   613])
```

In [ ]:
```
df["Purchase"].nunique()
```

Out[45]:
```
18105
```

In [ ]:
```
df["Purchase"].value_counts()
```

Out[46]:
```
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
```

Comment

On an average most of the people who do shopping from walmart spend 7k

Visual Analysis:-

In [ ]:
```
df.head()
```

Out[47]:

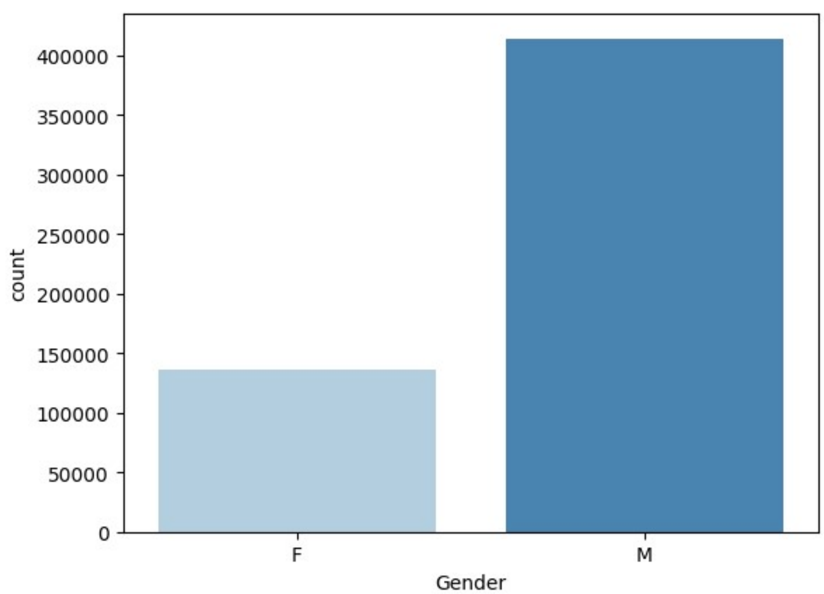
	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	1057
4	1000002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

A. Univariate

1.Count plots:-

In [ ]:
```
# Gender countplot
sns.countplot(data=df,x="Gender",palette="Blues")
```

Out[48]: <Axes: xlabel='Gender', ylabel='count'>

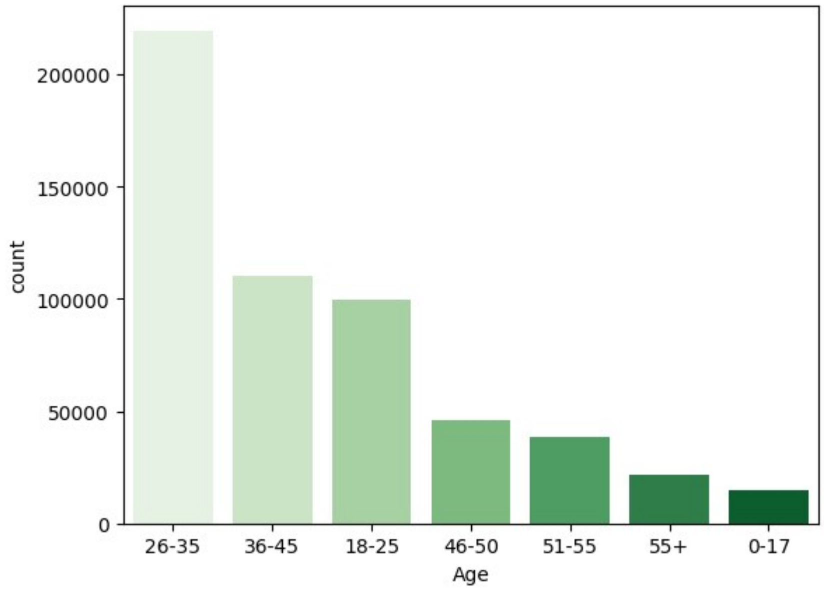


Comment

From the graph we can conclude that most of customer are male.

In [ ]:
```
# Age countplot
sns.countplot(data=df,x="Age",palette="Greens",order=df["Age"].value_counts().index)
```

Out[49]: <Axes: xlabel='Age', ylabel='count'>

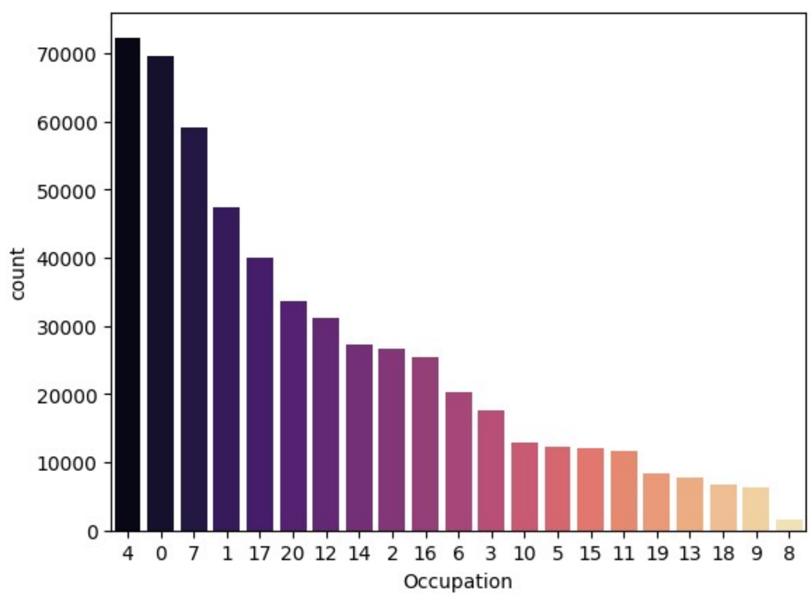


Comment

Customers of Age 26-35 are maximum among all.

In [ ]:
```
# Occupation Count plot
sns.countplot(data=df,x="Occupation",order=df["Occupation"].value_counts().index,palette="magma")
```

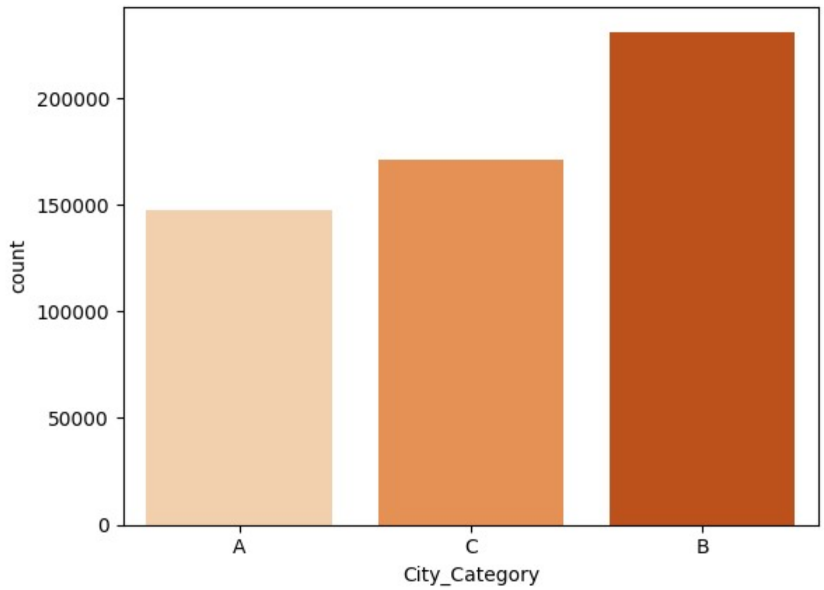
Out[50]: <Axes: xlabel='Occupation', ylabel='count'>



Comment

Most of the customer of walmart belongs to the occupation experience of 4yrs.

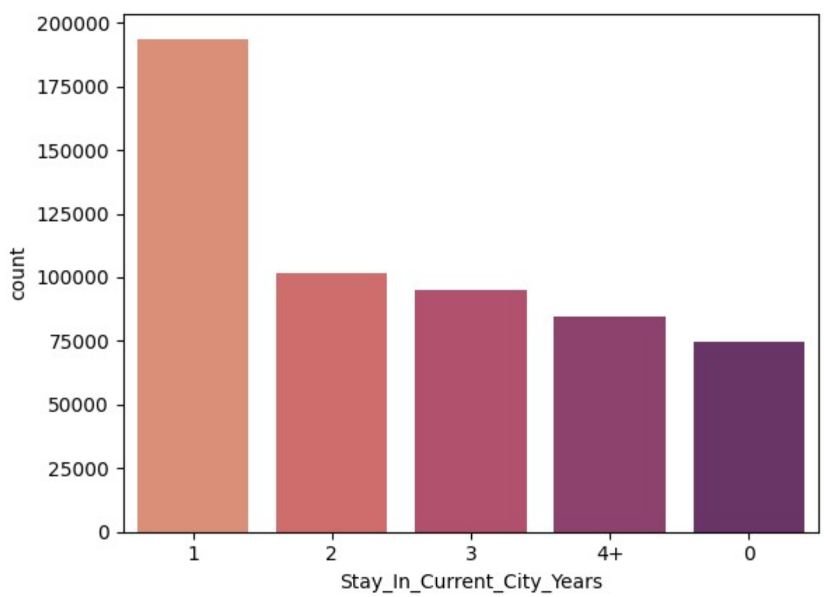
In [ ]:
```
# City_category countplot
sns.countplot(data=df,x="City_Category",palette="Oranges")
```
Out[51]: <Axes: xlabel='City_Category', ylabel='count'>



Comment

Most of the Customer are from the city_category -->B followed by A

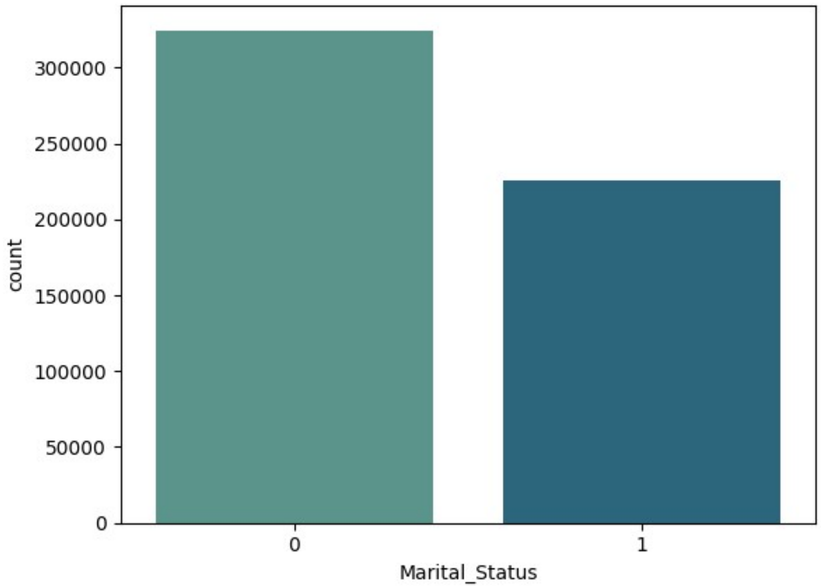
In [ ]:
```
# current city stay countplot
sns.countplot(data=df,x="Stay_In_Current_City_Years",order=df["Stay_In_Current_City_Years"].value_counts().index,palette="f
```
Out[52]: <Axes: xlabel='Stay_In_Current_City_Years', ylabel='count'>



Comment

Most of the customer who go to walmart for shopping are residing in their current city for 1yr .

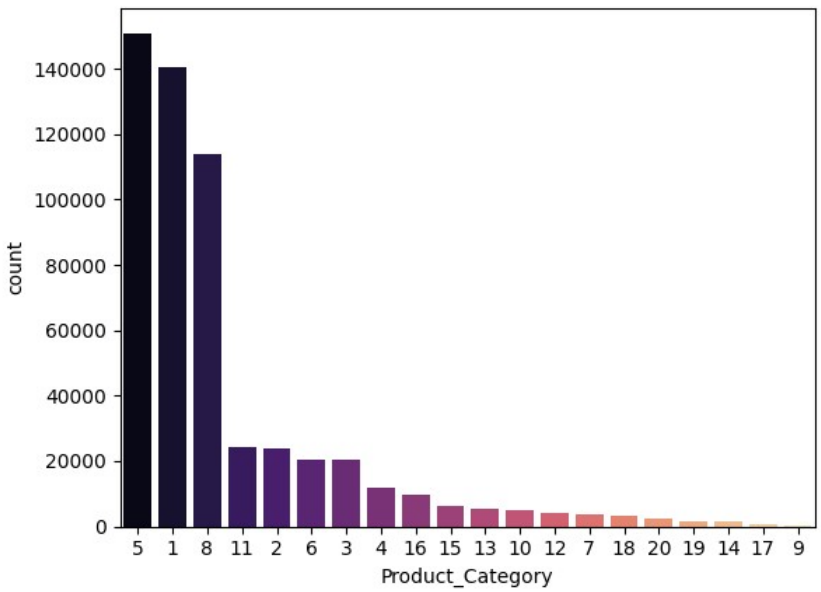
In [ ]:
```
# Marital status countplot
sns.countplot(data=df,x="Marital_Status",palette="crest")
```
Out[53]: <Axes: xlabel='Marital_Status', ylabel='count'>



Comment

From the graph it is concluded that most of the walmart customers are unmarried.

In [ ]:
```
# Product_category countplot
sns.countplot(data=df,x="Product_Category",order=df["Product_Category"].value_counts().index,palette="magma")
```
Out[54]: <Axes: xlabel='Product_Category', ylabel='count'>



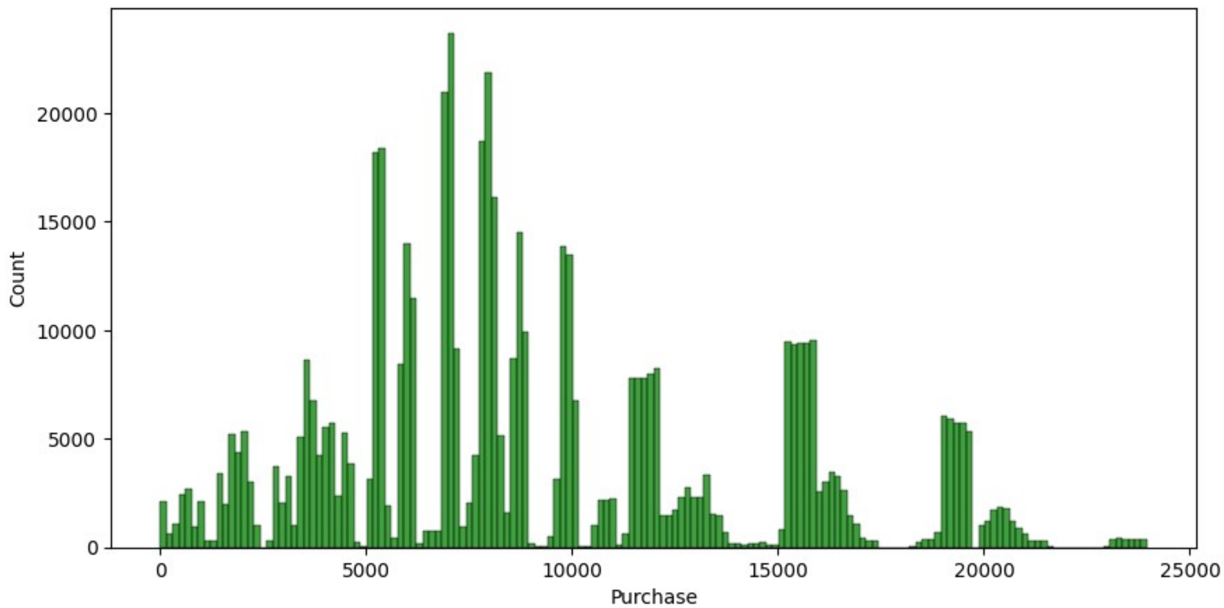
2.Histogram Plot

In [ ]:
```
df.head()
```

Out[55]:

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

In [ ]:
```
# Purchase plot
plt.figure(figsize=(10,5))
sns.histplot(df["Purchase"],color="g")
plt.show()
```



Comment
Customer who come to walmart for shopping most of them expend in the range of 6K-8K.

3.Box plot

To detect presence of outliers.

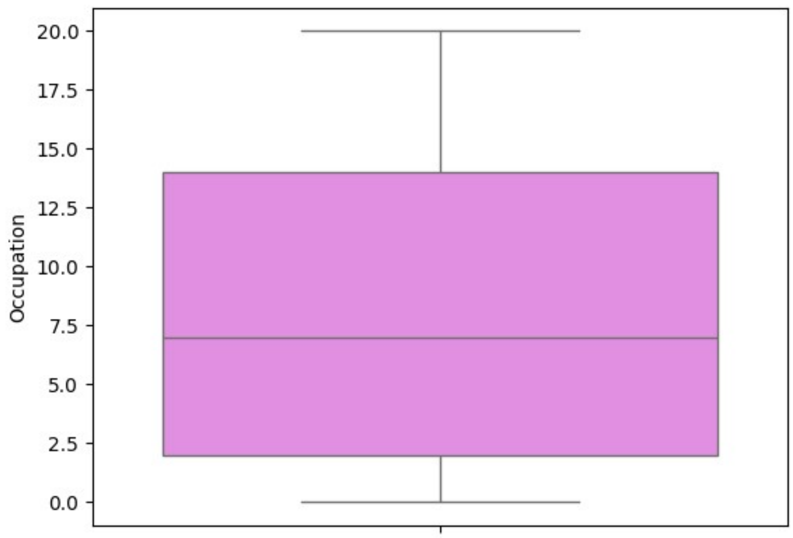
In [ ]:
```
df.head()
```

Out[57]:

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

In [ ]:

```
# Occupation Boxplot
sns.boxplot(df["Occupation"],orient="v",color="violet")
plt.show()
```

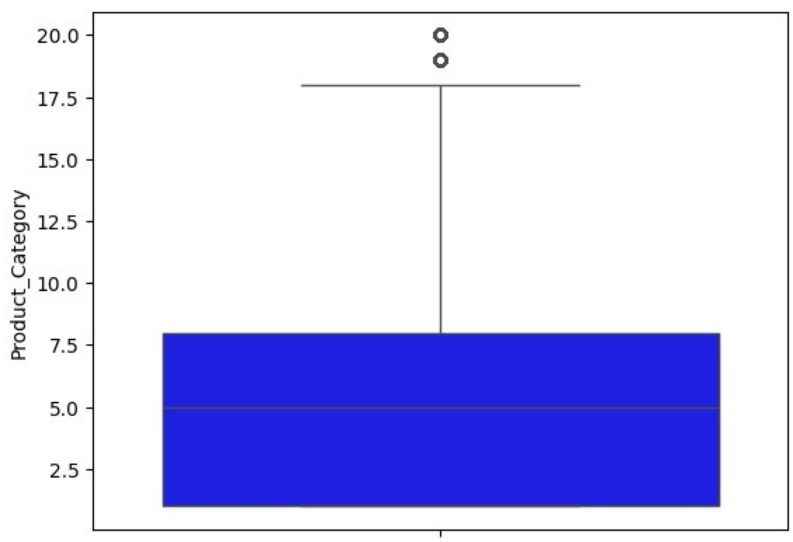


Comment

No outlier is present.

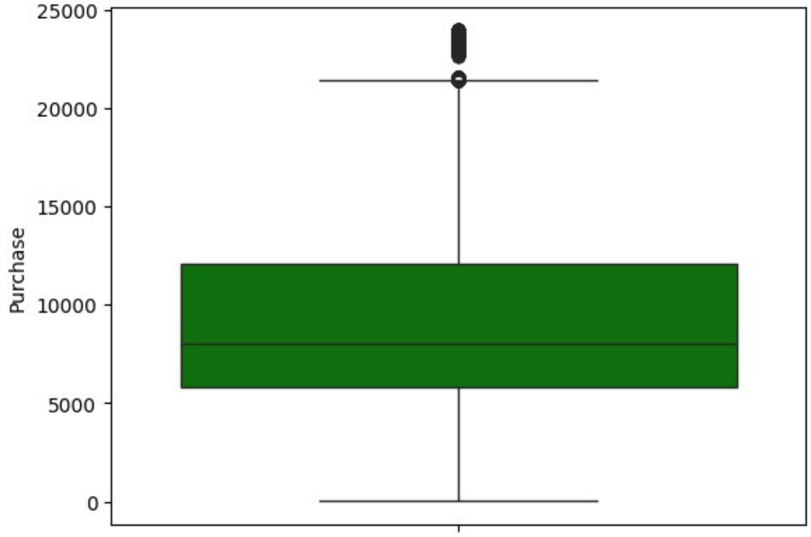
In [ ]:

```
# Product_category boxplot
sns.boxplot(df["Product_Category"],orient="v",color="blue")
plt.show()
```



comment
Outliers are above product_Category 17.

In [ ]:
```
# Purchase boxplot
sns.boxplot(df["Purchase"],orient="v",color="g")
plt.show()
```



comment
Outliers are above purchase amount of 20000.

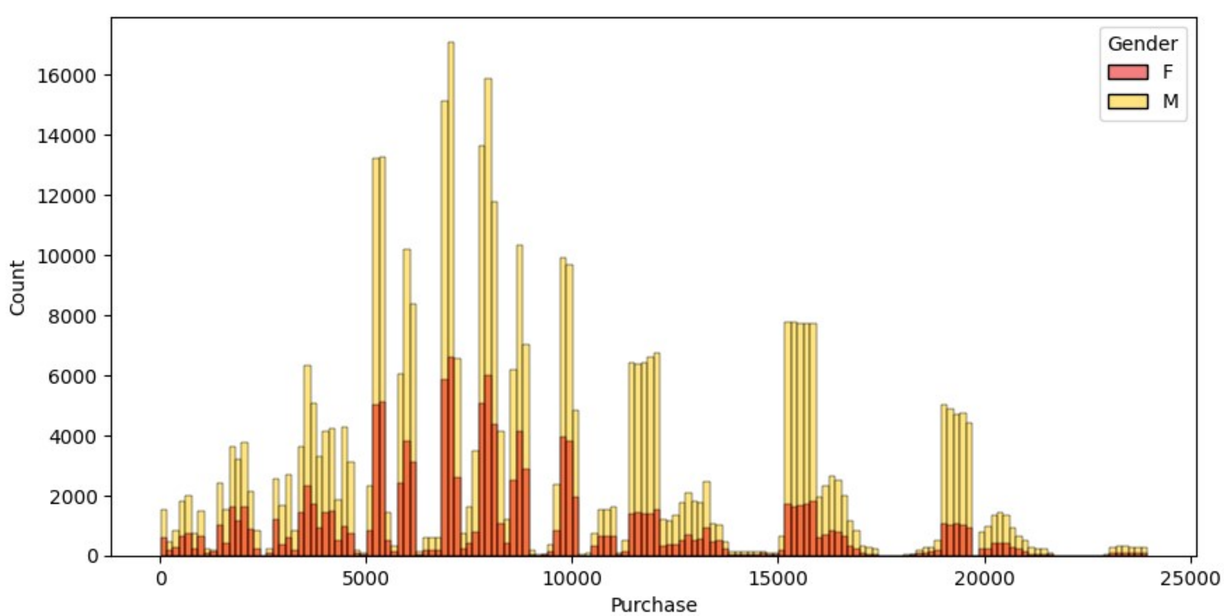
Bivariate Analysis
1.Histogram plot

In [ ]:
```
df.head()
```

Out[61]:

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

In [ ]:
```
# Purchase with respect to gender
plt.figure(figsize=(10,5))
sns.histplot(x=df["Purchase"],hue=df['Gender'],palette="hot")
plt.show()
```

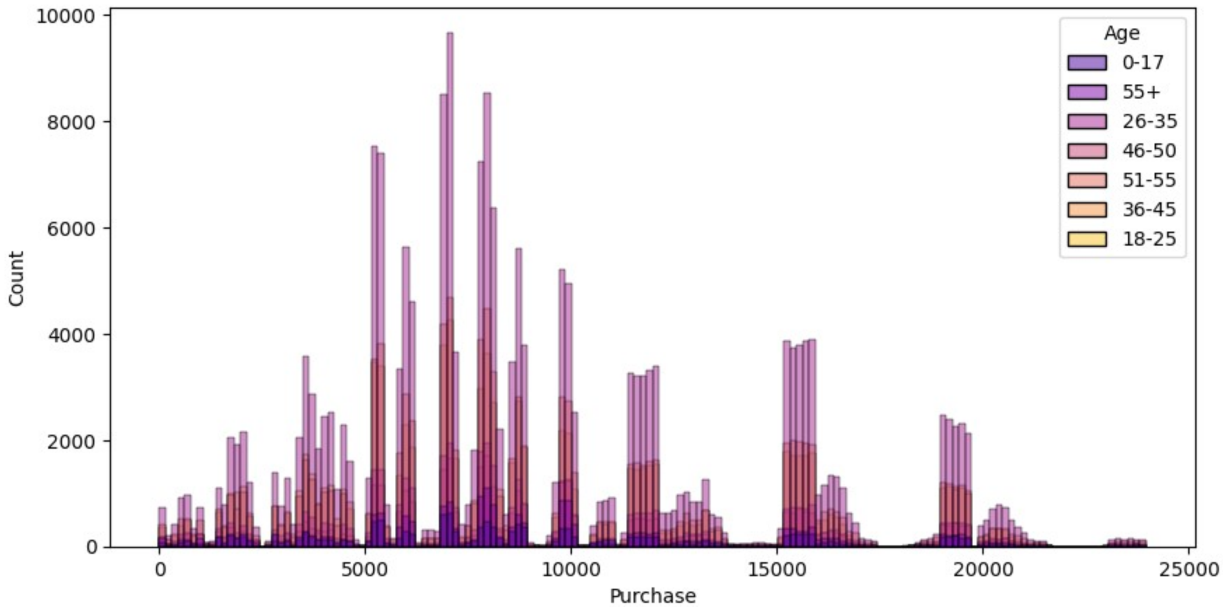


Comment

Male mostly prefer walmart for the shopping than women.

In [ ]:
```
# Purchase with respect to Age
plt.figure(figsize=(10,5))
sns.histplot(data=df,x="Purchase",hue="Age",palette="plasma")
```

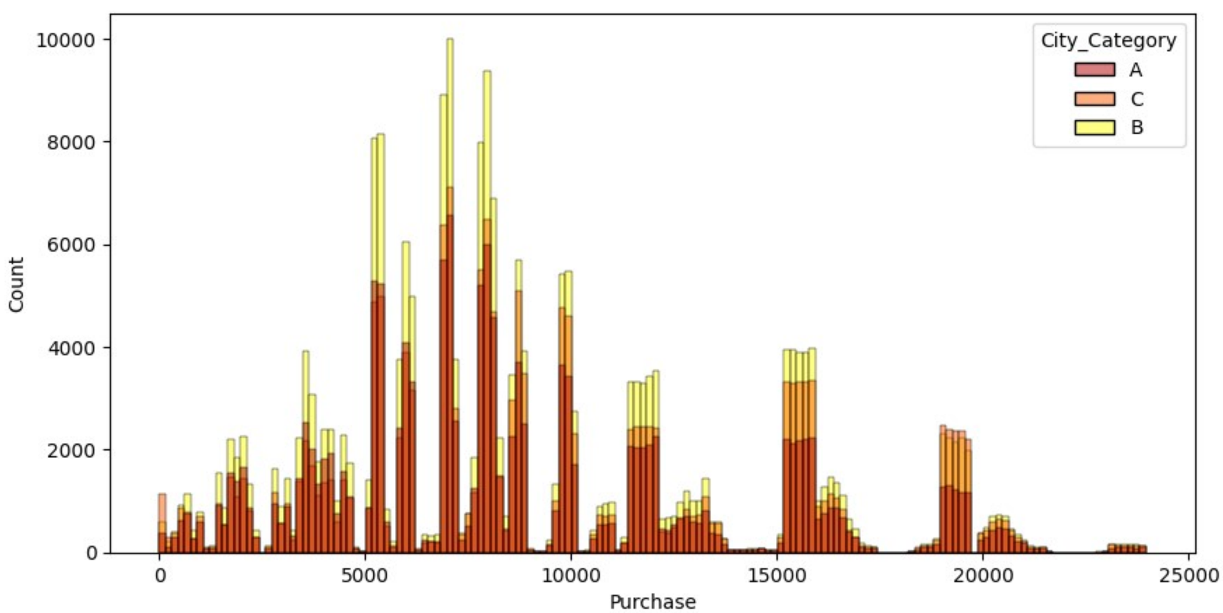
Out[63]:
```
<Axes: xlabel='Purchase', ylabel='Count'>
```



Comment

Maximum purchase are done by the customers of Age Group 26-35.

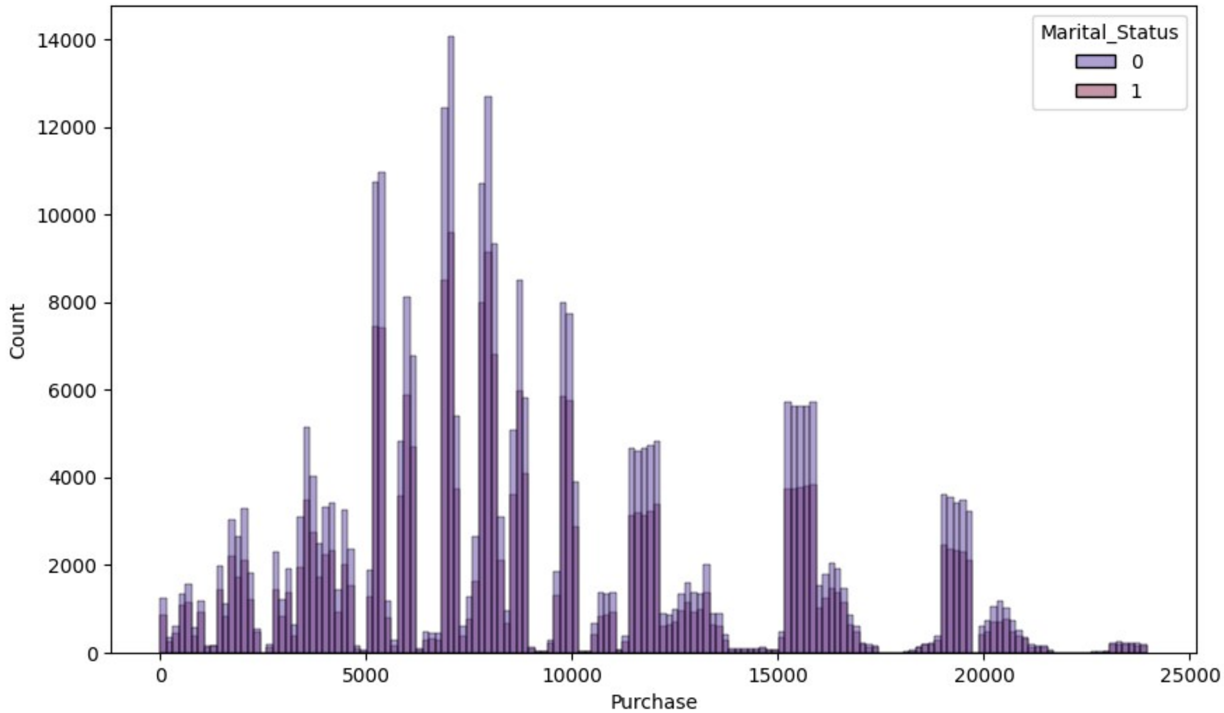
In [ ]:
```
# Purchase with respect to City_category
plt.figure(figsize=(10,5))
sns.histplot(data=df,x="Purchase",hue="City_Category",palette="hot")
plt.show()
```



Comment
Customers belonging to the City_category of B does maximum shopping at walmart followed by C and then A.

In [ ]:

```
# Purchase with respect to Marital Status
plt.figure(figsize=(10,6))
sns.histplot(data=df,x="Purchase",hue="Marital_Status",palette="twilight")
plt.show()
```

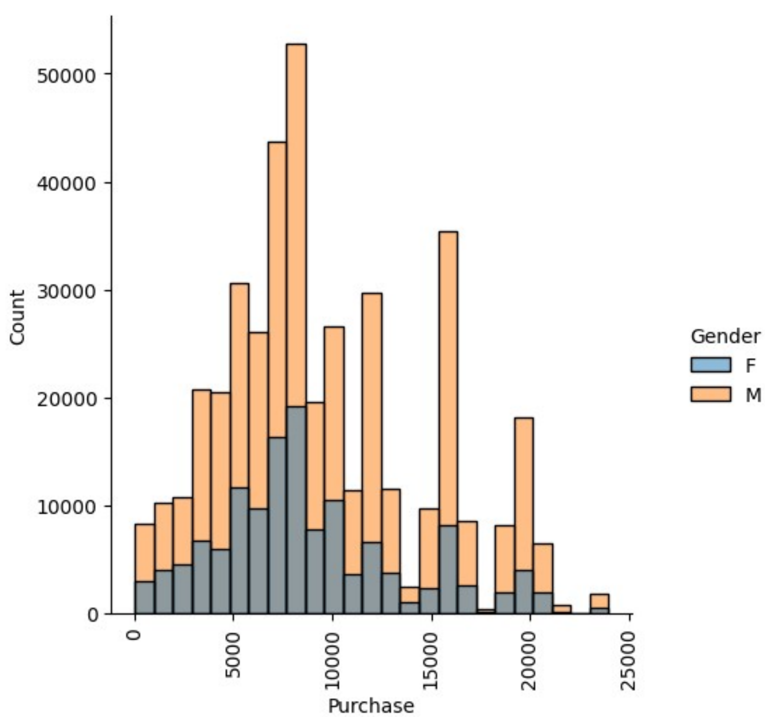


Comment
Mostly unmarried people do shopping from walmart in comparison to married.

2.Dis plot

In [ ]:

```
sns.displot(data=df,x="Purchase",hue="Gender",bins=25,color="magma_r")
plt.xticks(rotation=90)
plt.show()
```

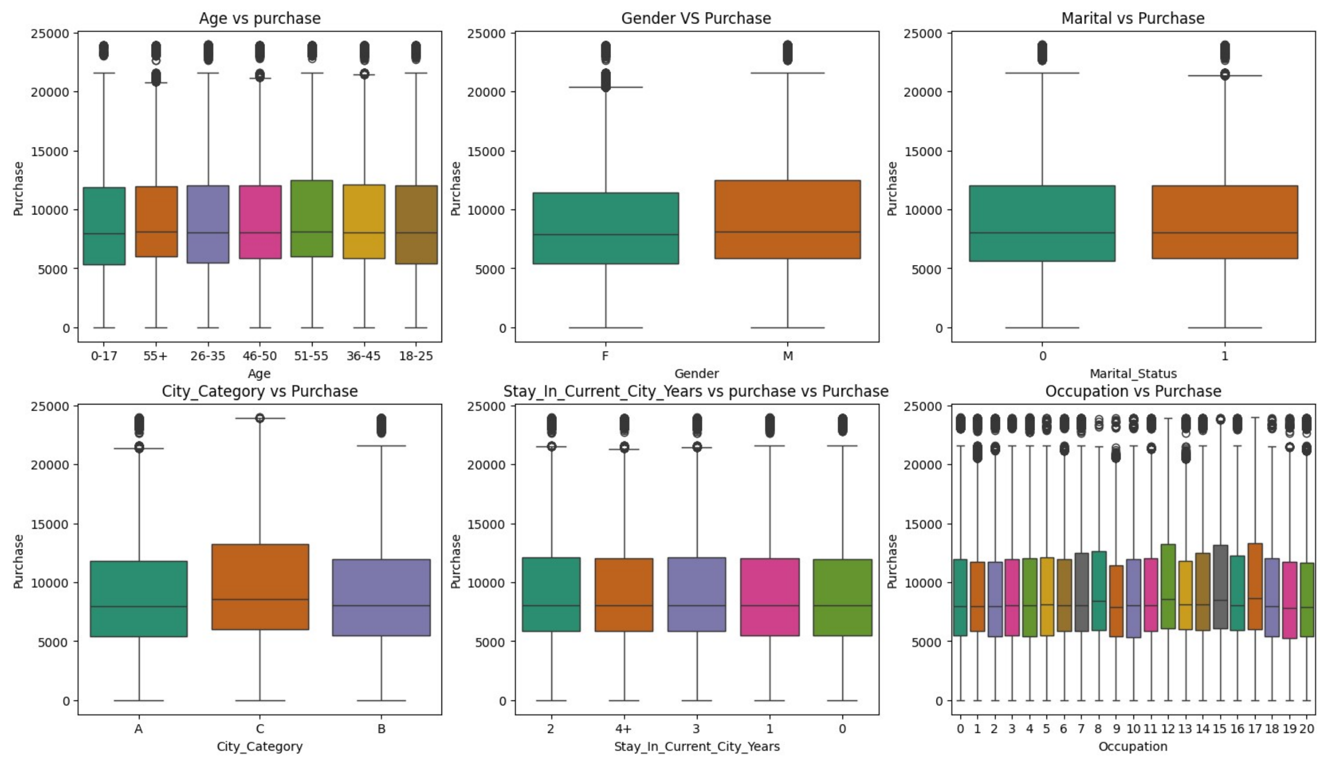


Comment

Males are purchasing more compare to female.

Box Plots-

In [ ]:
```python
# Purchase vs Various Parameter(gender,marital_status,Age,City_category,Current_city,Occuplation)
plt.figure(figsize=(18,10))
plt.subplot(2,3,1)
sns.boxplot(data=df,x="Age",y="Purchase",palette="Dark2")
plt.title("Age vs purchase",fontsize=12)
plt.subplot(2,3,2)
sns.boxplot(data=df,x="Gender",y="Purchase",palette="Dark2")
plt.title("Gender VS Purchase",fontsize=12)
plt.subplot(2,3,3)
sns.boxplot(data=df,x="Marital_Status",y="Purchase",palette="Dark2")
plt.title("Marital vs Purchase",fontsize=12)
plt.subplot(2,3,4)
sns.boxplot(data=df,x="City_Category",y="Purchase",palette="Dark2")
plt.title("City_Category vs Purchase",fontsize=12)
plt.subplot(2,3,5)
sns.boxplot(data=df,x="Stay_In_Current_City_Years",y="Purchase",palette="Dark2")
plt.title("Stay_In_Current_City_Years vs purchase vs Purchase",fontsize= 12)
plt.subplot(2,3,6)
sns.boxplot(data=df,x="Occupation",y="Purchase",palette="Dark2")
plt.title("Occupation vs Purchase",fontsize=12)
plt.show()
```



Comments

1) There is slight difference in the median purchase of male and female. (slightly higher for male)

2) Median purchase of every age group is nearly similar.

3) Median purchase of Occupational experience 12, 15 & 17 years are more amongst all.

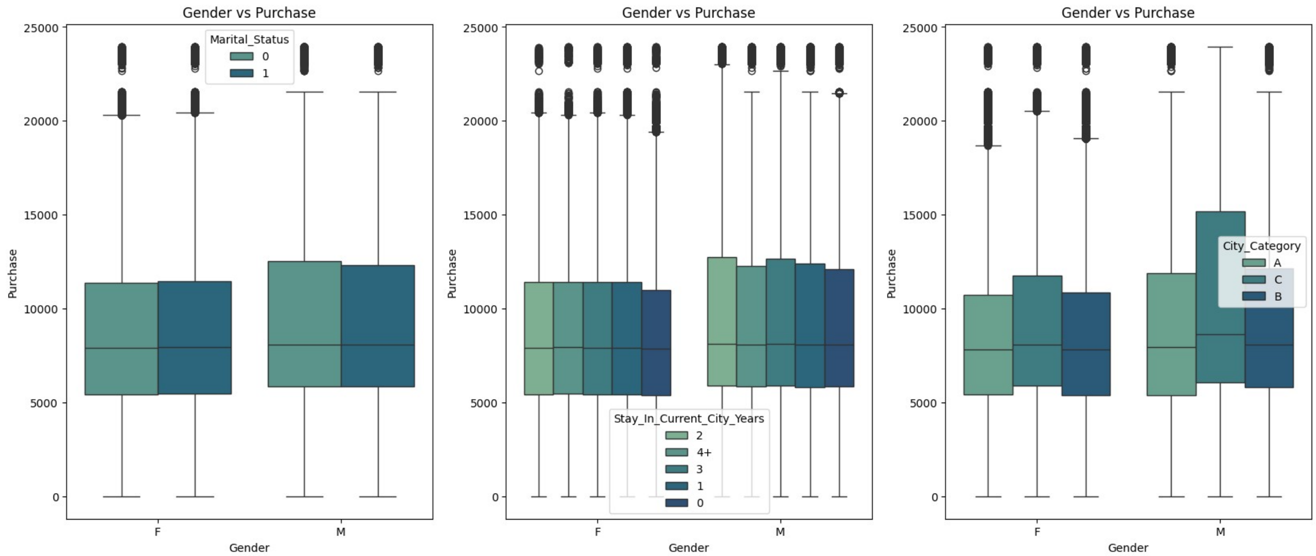
4) Median purchase for City Category 'C' is more than the rest City Category.

5) Median purchase for all current city stay is nearly equal.

6) Median purchase is almost equal for single and married people.

In [ ]:

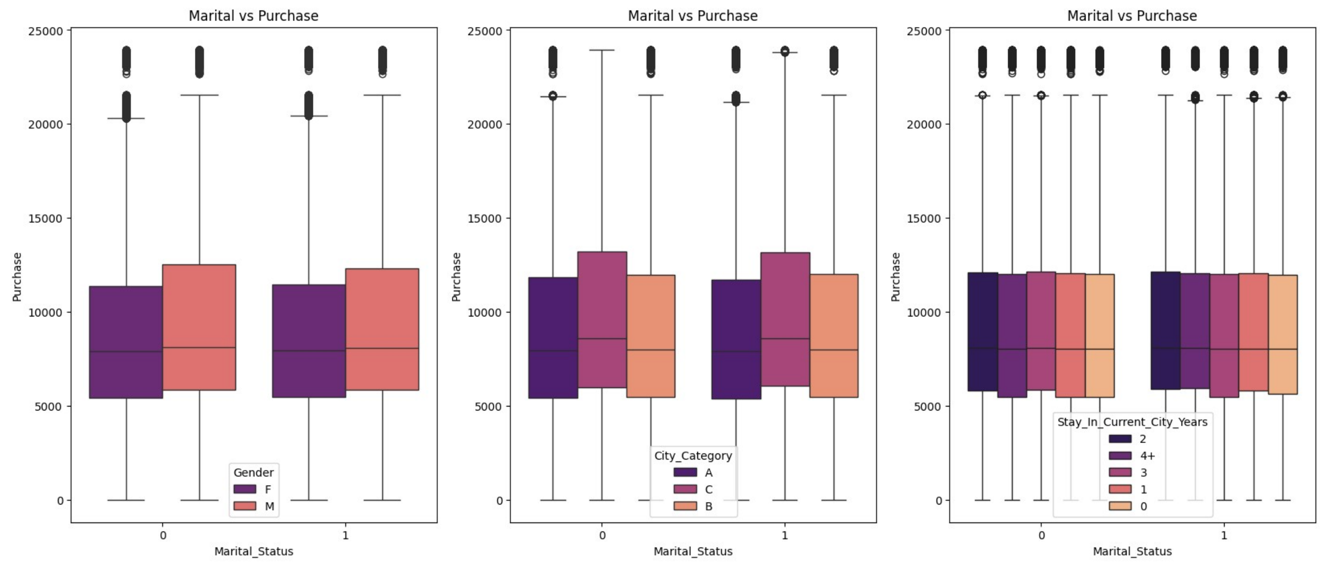
```
# Gender vs Purchase(hue as marital_status,city_category,current_city_stay,)
plt.figure(figsize=(20,8))
plt.subplot(1,3,1)
sns.boxplot(x="Gender",y="Purchase",hue="Marital_Status",data=df,palette="crest")
plt.title("Gender vs Purchase",fontsize=12)
plt.subplot(1,3,2)
sns.boxplot(x="Gender",y="Purchase",data=df,hue="Stay_In_Current_City_Years",palette="crest")
plt.title("Gender vs Purchase",fontsize=12)
plt.subplot(1,3,3)
sns.boxplot(x="Gender",y="Purchase",data=df,hue="City_Category",palette="crest")
plt.title("Gender vs Purchase",fontsize=12)
plt.show()
```



Comment

In every cases such as marital status, city category & current stay city, male customers are slightly more purchasing the product as compared to female customers.

In [ ]:
```
# Marital_status vs Purchase(with hue as Gender,Stay_in _current_city,city_category)
plt.figure(figsize=(20,8))
plt.subplot(1,3,1)
sns.boxplot(x="Marital_Status",y="Purchase",data=df,hue="Gender",palette="magma")
plt.title("Marital vs Purchase",fontsize=12)
plt.subplot(1,3,2)
sns.boxplot(x="Marital_Status",y="Purchase",data=df,hue="City_Category",palette="magma")
plt.title("Marital vs Purchase",fontsize=12)
plt.subplot(1,3,3)
sns.boxplot(x="Marital_Status",y="Purchase",data=df,hue="Stay_In_Current_City_Years",palette="magma")
plt.title("Marital vs Purchase",fontsize=12)
plt.show()
```



Comment
Purchase amount for both single & partnered customers are nearly same.

C. Multivariate Analysis -

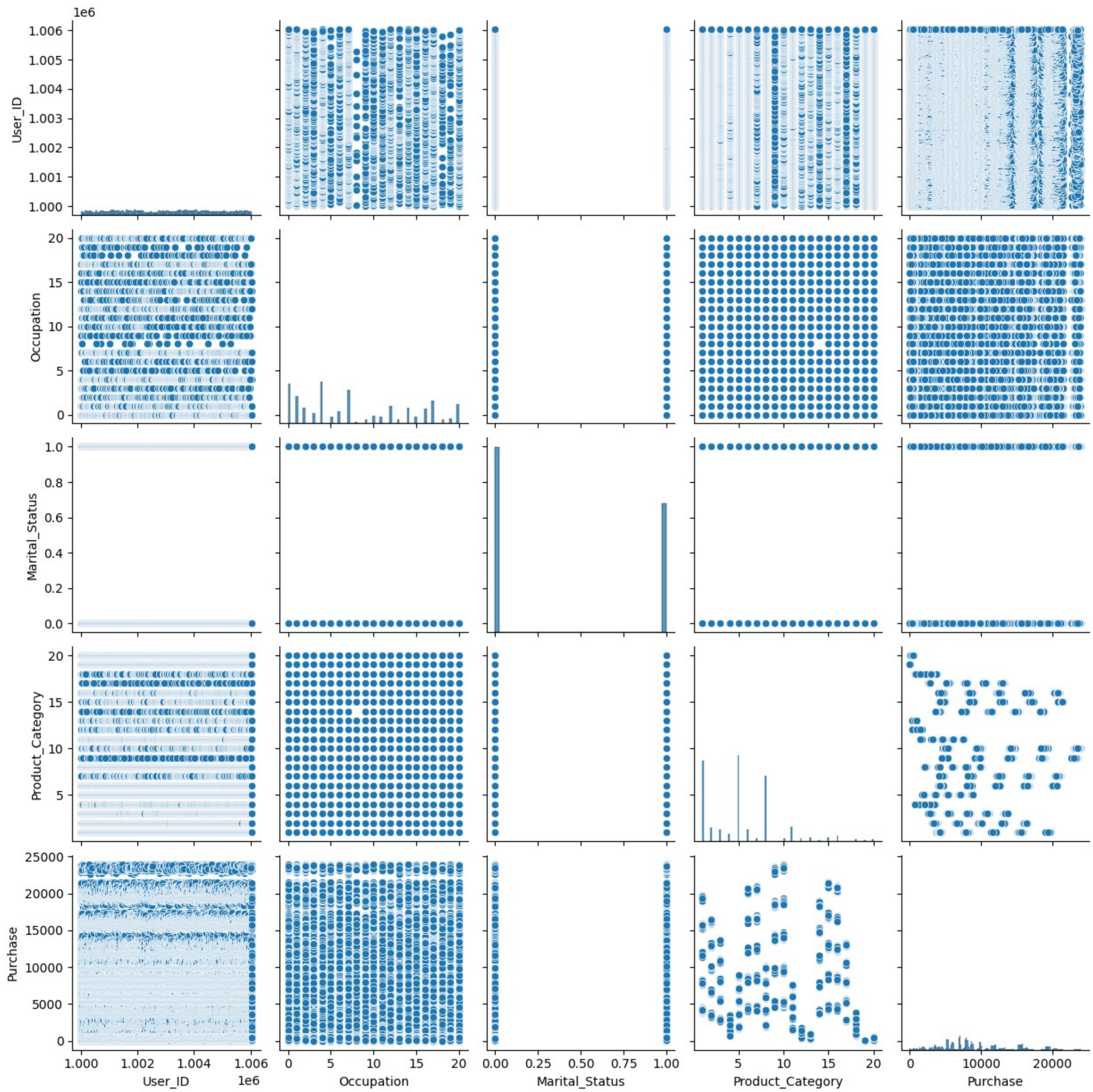
To check correlation

1.Pair Plots

In [ ]:

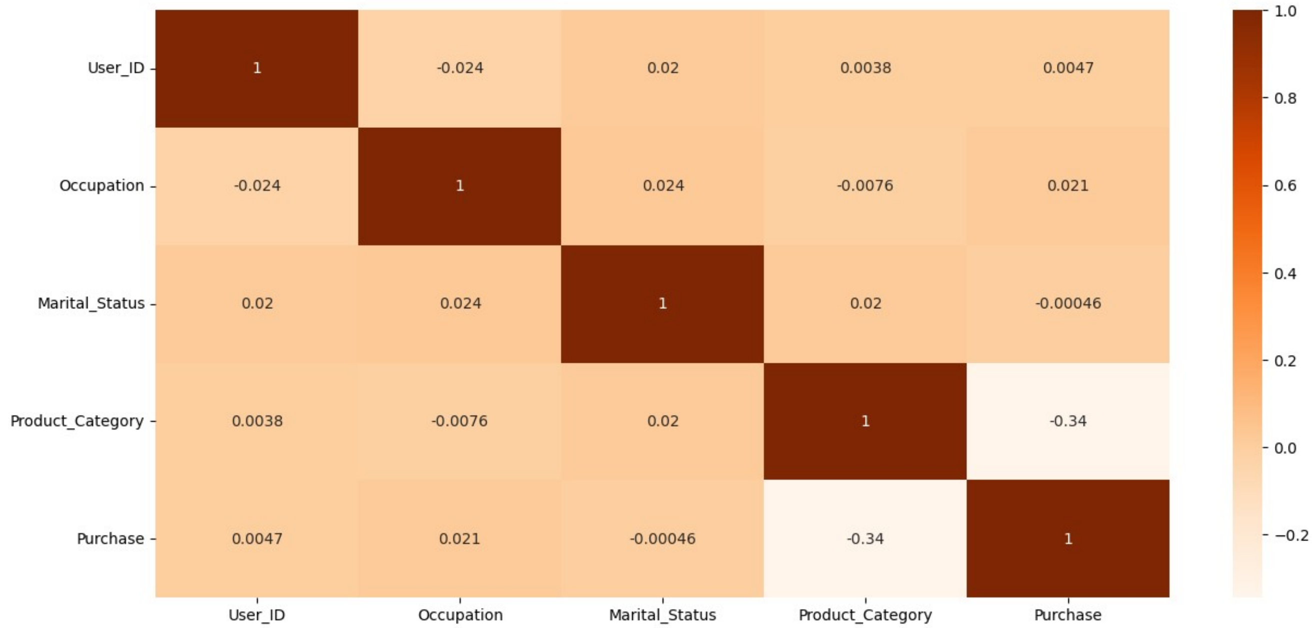
```
plt.figure(figsize=(12,10))
sns.pairplot(df)
plt.show()
```

<Figure size 1200x1000 with 0 Axes>



2.Correlation(Heatmaps)

In [ ]:
```python
# Heatmaps-
plt.figure(figsize=(15,7))
sns.heatmap(df.corr(), annot = True, cmap = "Oranges")
plt.show()
```



Comments

No positive or negative correlations can be seen from above pair plots & heatmaps.

D. CLT & Confidence Interval Analysis

In [ ]:
```python
samp=df.sample(500)
samp
```

Out[72]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
36197	1005576	P00127742	M	26-35	4	C	3	0	1	19221
201244	1001096	P00191442	M	0-17	10	C	1	0	1	11639
385158	1005283	P00025442	M	18-25	2	C	1	1	1	19677
485404	1002820	P00214642	F	36-45	0	A	2	0	11	3166
145639	1004447	P00102442	M	46-50	5	B	0	1	8	7887
...	...	...	...	...	...	...	...	...	...	...
215616	1003320	P00220442	M	26-35	1	B	1	1	5	8816
168634	1002002	P00116842	F	55+	13	C	2	1	2	16011
271560	1005841	P00153842	F	36-45	7	A	4+	1	8	9917
137918	1003332	P00022742	M	26-35	7	B	2	1	8	5953
387840	1005682	P00113242	M	18-25	0	B	2	0	1	19554

500 rows × 10 columns

Gender Analysis -

In [ ]:
```python
# overall mean for men
df[df["Gender"]=="M"]["Purchase"].mean()
```

Out[73]: 9437.526040472265

In [ ]:
```python
# overall mean for Women
df[df["Gender"]=="F"]["Purchase"].mean()
```

Out[74]: 8734.565765155476

In [ ]:
```python
# Sample Statistical Properties
samp.groupby("Gender")["Purchase"].describe()
```

Out[75]:

Gender	count	mean	std	min	25%	50%	75%	max
F	131.0	9256.305344	5145.845057	937.0	5669.0	8000.0	11941.0	21107.0
M	369.0	9609.384824	5006.973156	237.0	5968.0	8051.0	12853.0	23455.0

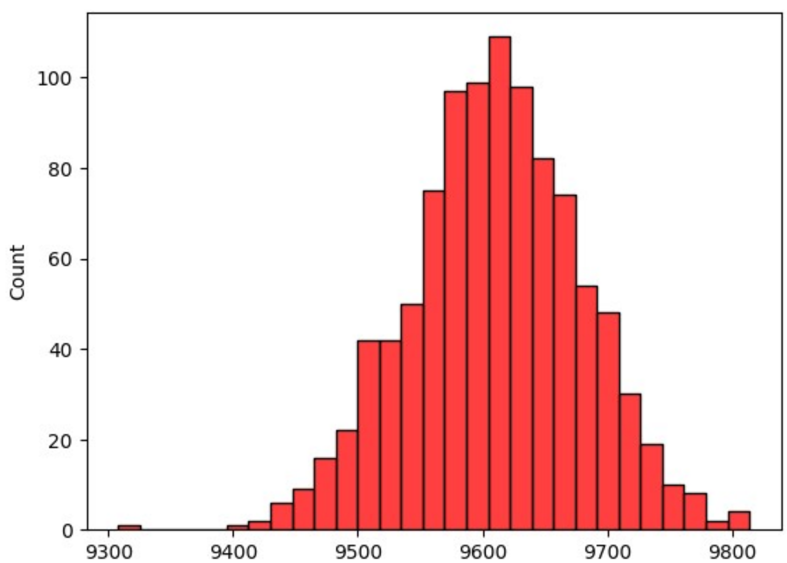
In [ ]:
```python
male_samp_mean = [samp[samp["Gender"] == "M"].sample(5000, replace = True)["Purchase"].mean() for i in range(1000)]
9480.9504,
9557.2912,
9486.1522,
9495.7548,
9605.7188,
9445.6576,
9499.529,
9530.2892,
9530.3016,
9524.3388,
9515.9612,
9565.0358,
9552.7132,
9559.0084,
9566.0034,
9592.6308,
9562.212,
9464.1886,
9695.7958,
```

Out[76]: (9695.7958,)

In [ ]:
```python
len(male_samp_mean)
```

Out[77]: 1000

In [ ]:
```python
sns.histplot(male_samp_mean,color="r")
plt.show()
```



In [ ]:

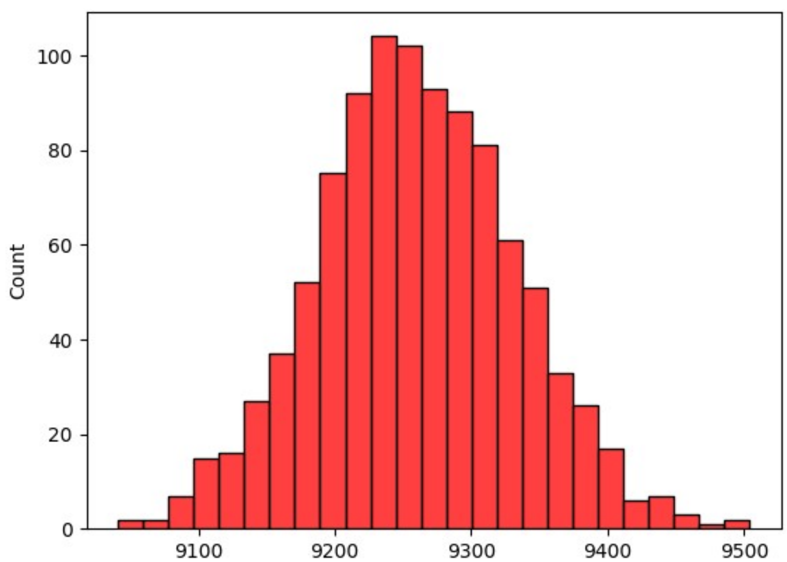
```
female_samp_mean = [samp[samp["Gender"] == "F"].sample(5000, replace = True)["Purchase"].mean() for i in range(1000)]
female_samp_mean
```

Out[79]:

```
[9136.8316,
 9194.682,
 9316.6526,
 9183.1522,
 9285.072,
 9202.881,
 9368.3458,
 9269.2232,
 9397.0328,
 9215.8106,
 9345.7824,
 9366.1464,
 9239.9064,
 9230.2686,
 9233.6244,
 9444.4384,
 9273.0736,
 9275.3458,
 9299.627,
```

In [ ]:

```
sns.histplot(female_samp_mean,color="r")
plt.show()
```



In [ ]:

```
# std deviation of male sample
male_std=np.std(male_samp_mean).round(3)
male_std
```

Out[81]: 69.001

In [ ]:

```
# std deviation of female sample
female_std=np.std(female_samp_mean).round(3)
female_std
```

Out[82]: 72.879

CI--90%

In [ ]:

```
# confidence Interval of male 90%
male_low=np.mean(male_samp_mean)+norm.ppf(0.05)*np.std(female_samp_mean)
male_high=np.mean(male_samp_mean)+norm.ppf(0.95)*np.std(female_samp_mean)
male_low.round(3),male_high.round(3)
```

Out[83]: (9489.932, 9729.682)

In [ ]:

```
# confidence Interval of female 90%
female_low=np.mean(female_samp_mean)+norm.ppf(0.05)*np.std(female_samp_mean)
female_high=np.mean(female_samp_mean)+norm.ppf(0.95)*np.std(female_samp_mean)
female_low.round(3),female_high.round(3)
```

Out[84]: (9139.843, 9379.592)

In [ ]:
```
# To check the overlapping of confidence interval
male_CI=np.percentile(male_samp_mean,[5,95])
female_CI=np.percentile(female_samp_mean,[5,95])
male_CI.round(3),female_CI.round(3)
```

Out[85]: (array([9495.478, 9721.252]), array([9139.509, 9383.44]))

Comment

From above result, for 90% CI - it is clear that confidence intervals of male & female average purchases are not overlapping.

CI-95%

In [ ]:
```
# Confidence Interval of male for 95% interval
male_low=np.mean(male_samp_mean)+norm.ppf(.025)*np.std(male_samp_mean)
male_high=np.mean(male_samp_mean)+norm.ppf(.975)*np.std(male_samp_mean)
male_low.round(3),male_high.round(3)
```

Out[86]: (9474.567, 9745.046)

In [ ]:
```
# Confidence Interval of female for 95% interval
female_low=np.mean(female_samp_mean)+norm.ppf(.025)*np.std(female_samp_mean)
female_high=np.mean(female_samp_mean)+norm.ppf(.975)*np.std(female_samp_mean)
female_low.round(3),female_high.round(3)
```

Out[87]: (9116.878, 9402.557)

In [ ]:
```
# To check the overlapping of confidence interval of male and female
male_ci=np.percentile(male_samp_mean,[2.5,97.5])
female_ci=np.percentile(female_samp_mean,[2.5,97.5])
male_ci.round(3),female_ci.round(3)
```

Out[88]: (array([9475.208, 9741.718]), array([9114.125, 9401.147]))

Comment

From above result, for 95% CI - it is clear that confidence intervals of male & female average purchases are not overlapping.

CI 99%

In [ ]:
```
# confidence Interval of male for CI---99
male_low=np.mean(male_samp_mean)+norm.ppf(.005)*np.std(male_samp_mean)
male_high=np.mean(male_samp_mean)+norm.ppf(.995)*np.std(male_samp_mean)
male_low.round(3),male_high.round(3)
```

Out[89]: (9432.072, 9787.541)

In [ ]:
```
# confidence Interval of female for CI---99
female_low=np.mean(female_samp_mean)+norm.ppf(.005)*np.std(female_samp_mean)
female_high=np.mean(female_samp_mean)+norm.ppf(.995)*np.std(female_samp_mean)
female_low.round(3),female_high.round(3)
```

Out[90]: (9071.994, 9447.441)

In [ ]:
```
# checking overlapping of confidence interval of male and female
male_ci=np.percentile(male_samp_mean,[.005,.995])
female_ci=np.percentile(female_samp_mean,[.005,.995])
male_ci.round(3),female_ci.round(3)
```

Out[91]: (array([9312.436, 9456.618]), array([9041.618, 9095.887]))

Comment

From above result, for 99% CI - it is clear that confidence intervals of male & female average purchases are not overlapping.

Inferences from Gender CI Analysis

1) For males 90% CI of means = [9235.911, 9466.956] & For females = [7939.954, 8135.586]

2) For males 95% CI of means = [9215.779, 9486.584] & For females = [7921.74 , 8158.717]

3) For males 99% CI of means = [9143.851, 9182.896] & For females = [7837.958, 7901.585]

4) From above analysis males are purchasing more with different condifence intervals as compared to females.

Marital Status Analysis

In [ ]:
```python
Samp2=df.sample(500)
Samp2
```

Out[92]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
274577	1000305	P00343442	F	18-25	0	B	3	1	12	1736
279789	1001141	P00222842	F	26-35	3	B	1	1	8	6168
144364	1004271	P00192942	M	36-45	7	B	2	0	5	5156
532269	1003957	P00157942	M	36-45	11	B	1	0	8	5962
511582	1000840	P00119342	F	26-35	3	C	1	1	10	14085
...	...	...	...	...	...	...	...	...	...	...
62654	1003648	P00018142	M	18-25	4	B	1	0	5	7099
476515	1001387	P00148642	F	51-55	13	B	1	1	6	20494
336884	1003842	P00171342	F	36-45	16	B	4+	0	13	560
62792	1003664	P00278642	F	36-45	1	A	0	0	5	7184
448025	1003026	P00182242	F	18-25	4	B	1	0	1	4298

500 rows × 10 columns

In [ ]:
```python
# overall mean for Single customer
df[df["Marital_Status"]==0]["Purchase"].mean().round(3)
```

Out[93]: 9265.908

In [ ]:
```python
# overall mean for married customer
df[df["Marital_Status"]==1]["Purchase"].mean().round(3)
```

Out[94]: 9261.175

In [ ]:
```python
# Sample statistical Properties
Samp2.groupby("Marital_Status").describe()
```

Out[95]:

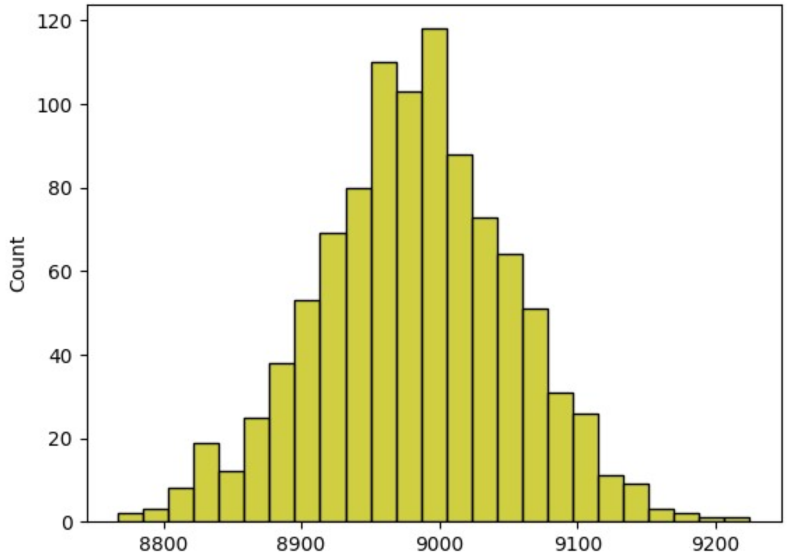
	User_ID								Occupation		...	Product_Category		
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count
Marital_Status														
0	296.0	1.003054e+06	1691.078585	1000092.0	1001517.00	1003194.5	1004412.00	1006010.0	296.0	7.300676	...	8.0	20.0	296.0
1	204.0	1.002916e+06	1703.424652	1000035.0	1001443.75	1003049.5	1004446.25	1006016.0	204.0	7.872549	...	8.0	18.0	204.0

2 rows × 32 columns

In [ ]:
```python
unmarried_samp2_mean=[Samp2[Samp2["Marital_Status"]==0].sample(5000,replace=True)["Purchase"].mean() for i in range(1000)]
unmarried_samp2_mean
```

Out[96]:
```
[8964.3152,
 8902.925,
 8902.629,
 8946.7808,
 9019.6918,
 9018.4678,
 8867.1376,
 9193.5724,
 8992.2472,
 9062.5668,
 8966.2016,
 8982.3044,
 9067.4258,
 8953.7754,
 9008.6898,
 9061.0676,
 8922.5524,
 8907.57,
 9008.0258,
```

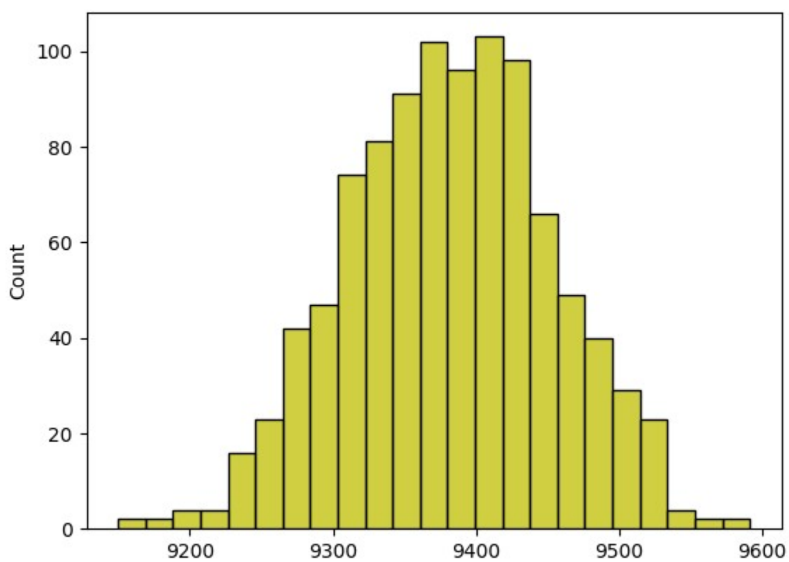
In [ ]:
```python
sns.histplot(unmarried_samp2_mean,color="y")
plt.show()
```



In [ ]:
```python
married_samp2_mean=[Samp2[Samp2["Marital_Status"]==1].sample(5000,replace=True)["Purchase"].mean() for i in range(1000)]
married_samp2_mean
```

Out[98]:
```
[9460.9092,
 9281.652,
 9416.3314,
 9324.0758,
 9388.1378,
 9429.0618,
 9412.6334,
 9382.847,
 9324.3558,
 9431.7326,
 9372.4562,
 9388.5462,
 9360.8188,
 9259.7246,
 9430.664,
 9365.8862,
 9297.2532,
 9238.8328,
 9322.5262,
```

In [ ]:
```python
sns.histplot(married_samp2_mean,color="y")
plt.show()
```



In [ ]:
```python
# standard deviation of unmarried customer
np.std(unmarried_samp2_mean).round(3)
```

Out[100]:
```
70.052
```

In [ ]:
```
# standard deviation of married customer
np.std(married_samp2_mean).round(3)
```
Out[101]: 71.591

CI --->90

In [ ]:
```
# Confidence Interval of Single(unmarried)-->90
unmarried_low=np.mean(unmarried_samp2_mean)+norm.ppf(.05)*np.std(unmarried_samp2_mean)
unmarried_high=np.mean(unmarried_samp2_mean)+norm.ppf(.95)*np.std(unmarried_samp2_mean)
unmarried_low.round(3),unmarried_high.round(3)
```
Out[102]: (8866.831, 9097.282)

In [ ]:
```
# confidence Interval of married customer--->90
married_low=np.mean(married_samp2_mean)+norm.ppf(.05)*np.std(married_samp2_mean)
married_high=np.mean(married_samp2_mean)+norm.ppf(.95)*np.std(married_samp2_mean)
married_low.round(3),married_high.round(3)
```
Out[103]: (9263.753, 9499.265)

In [ ]:
```
#  To check Overlapping of Confidence Interval
unmarried_CI=np.percentile(unmarried_samp2_mean,[5,95]).round(3)
married_CI=np.percentile(married_samp2_mean,[5,95]).round(3)
unmarried_CI,married_CI
```
Out[104]: (array([8865.922, 9097.569]), array([9263.346, 9500.324]))

Comment

From above result, for 90% CI - it is clear that confidence intervals of unmarried & married people average purchases are overlapping.

CI---95%

In [ ]:
```
# Confidence Interval of single(unmarried)----95%
unmarried_low=np.mean(unmarried_samp2_mean)+norm.ppf(.025)*np.std(unmarried_samp2_mean)
unmarried_high=np.mean(unmarried_samp2_mean)+norm.ppf(.975)*np.std(unmarried_samp2_mean)
unmarried_low.round(3),unmarried_high.round(3)
```
Out[105]: (8844.756, 9119.357)

In [ ]:
```
# confidence Interval of married---->95%
married_low=np.mean(married_samp2_mean)+norm.ppf(.025)*np.std(married_samp2_mean)
married_high=np.mean(married_samp2_mean)+norm.ppf(.975)*np.std(married_samp2_mean)
married_low.round(3),married_high.round(3)
```
Out[106]: (9241.194, 9521.824)

In [ ]:
```
# checking the overlapping of confidence Interval
unmarried_CI=np.percentile(unmarried_samp2_mean,[2.5,97.5]).round(3)
married_CI=np.percentile(married_samp2_mean,[2.5,97.5]).round(3)
unmarried_CI,married_CI
```
Out[107]: (array([8835.793, 9120.042]), array([9243.538, 9519.229]))

Comment

From above result, for 95% CI - it is clear that confidence intervals of unmarried & married people average purchases are overlapping.

CI----->99%

In [ ]:
```
# Confidence Interval of unmarried for 99%
unmarried_low=np.mean(unmarried_samp2_mean)+norm.ppf(.005)*np.std(unmarried_samp2_mean)
unmarried_high=np.mean(unmarried_samp2_mean)+norm.ppf(.995)*np.std(unmarried_samp2_mean)
unmarried_low.round(3),unmarried_high.round(3)
```
Out[108]: (8801.614, 9162.499)

In [ ]:
```
# confidence Interval of married for 99%
married_low=np.mean(married_samp2_mean)+norm.ppf(.005)*np.std(married_samp2_mean)
married_high=np.mean(married_samp2_mean)+norm.ppf(.995)*np.std(married_samp2_mean)
married_low.round(3),married_high.round(3)
```
Out[109]: (9197.104, 9565.914)

In [ ]:
```
# overlapping of married and unmarried --->99%
unmarried_CI=np.percentile(unmarried_samp2_mean,[.5,99.5]).round(3)
married_CI=np.percentile(married_samp2_mean,[.5,99.5]).round(3)
unmarried_CI,married_CI
```

Out[110]:
```
(array([8807.044, 9155.283]), array([9190.886, 9538.117]))
```

Comment

From above result, for 99% CI - it is clear that confidence intervals of unmarried & married people average purchases are overlapping.

Inferences from Marital Status CI Analysis -

1) For unmarried customers 90% CI of means = [9178.918, 9430.879] & For married customers = [8951.612, 9165.207

2) For unmarried customers 95% CI of means = [9160.84 , 9457.904] & For married customers = [8934.895, 9189.142]

3) For unmarried customers 99% CI of means = [9113.901, 9495.289] & For married customers = [8899.153, 9227.735]

Age Analysis:-

In [ ]:
```
age_samp=df.sample(500)
age_samp
```

Out[111]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
280278	1001203	P00317842	F	26-35	1	A	3	0	8	7917
549569	1005312	P00371644	M	26-35	1	A	1	0	20	489
176010	1003267	P00267542	M	26-35	4	A	2	0	1	7683
43488	1000713	P00184242	M	36-45	7	B	3	0	9	18783
478402	1001676	P00122542	M	18-25	4	B	4+	0	11	5903
...	...	...	...	...	...	...	...	...	...	...
88517	1001645	P00247542	F	18-25	9	B	1	1	8	8028
105776	1004286	P00102542	M	36-45	17	C	0	0	8	2092
305122	1004998	P00034742	F	18-25	4	C	1	1	5	5233
364812	1002097	P00017542	M	18-25	5	C	0	0	5	7135
419795	1004543	P00045342	M	26-35	2	A	0	0	1	11392

500 rows × 10 columns

Overall mean for different age group

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

Out[112]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

In [ ]:
```
df[df["Age"]=="0-17"]["Purchase"].mean()
```

Out[113]:
```
8933.464640444974
```

In [ ]:
```
df[df["Age"]=="18-25"]["Purchase"].mean()
```

Out[114]:
```
9169.663606261289
```

In [ ]:
```
df[df["Age"]=="26-35"]["Purchase"].mean()
```

Out[115]:
```
9252.690632869888
```

In [ ]:
```
df[df["Age"]=="36-45"]["Purchase"].mean()
```

Out[116]:
```
9331.350694917874
```

In [ ]:
```
df[df["Age"]=="46-50"]["Purchase"].mean()
```

Out[117]:
```
9208.625697468327
```

In [ ]:
```
df[df["Age"]=="51-55"]["Purchase"].mean()
```

Out[118]:
```
9534.808030960236
```

In [ ]:
```
df[df["Age"]=="55+"]["Purchase"].mean()
```

Out[119]: 9336.280459449405

In [ ]:
```
# Sample Statistical Properties:-
age_samp.groupby("Age")["Purchase"].describe()
```

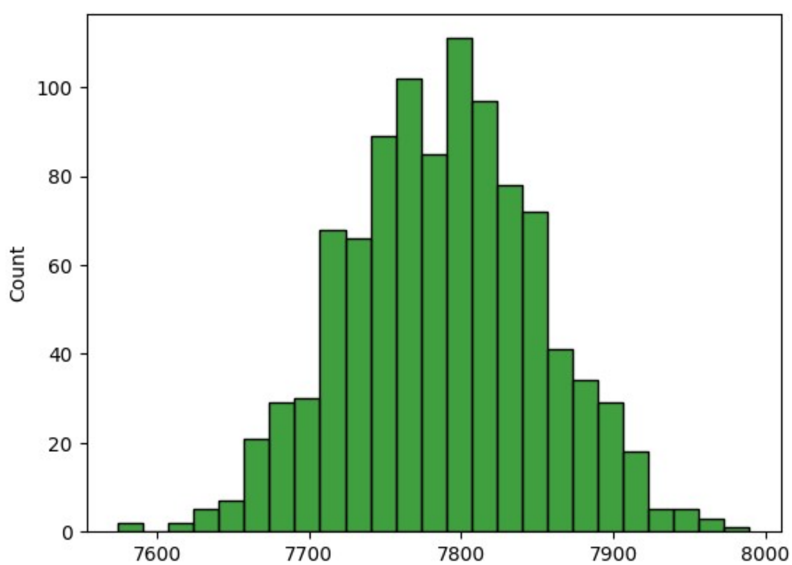
Out[120]:

Age	count	mean	std	min	25%	50%	75%	max
0-17	17.0	7788.176471	4714.927614	772.0	4243.00	6997.0	9988.00	18570.0
18-25	80.0	8002.875000	4199.868571	357.0	5328.25	7147.0	8868.25	20047.0
26-35	222.0	9144.617117	4822.256053	48.0	5968.75	8363.5	11685.25	23305.0
36-45	103.0	9344.893204	5397.951016	14.0	5283.00	8051.0	12314.00	23119.0
46-50	32.0	9902.875000	5606.610739	1759.0	6295.75	8210.0	13054.75	23699.0
51-55	35.0	8976.714286	5031.930471	2107.0	5443.50	8028.0	10057.50	20521.0
55+	11.0	8861.818182	4162.225794	1450.0	7041.50	9843.0	9926.50	17403.0

In [ ]:
```
mean_age_0_17=[age_samp[age_samp["Age"]=="0-17"].sample(5000,replace=True)["Purchase"].mean() for i in range(1000)]
mean_age_0_17
```

Out[121]:
```
[7803.6182,
 7794.5474,
 7762.5926,
 7824.034,
 7747.4768,
 7800.3846,
 7795.17,
 7808.5646,
 7789.0216,
 7822.6006,
 7752.6062,
 7729.6754,
 7871.5146,
 7719.2984,
 7798.8686,
 7856.7818,
 7825.597,
 7639.4348,
 7843.6814,
```

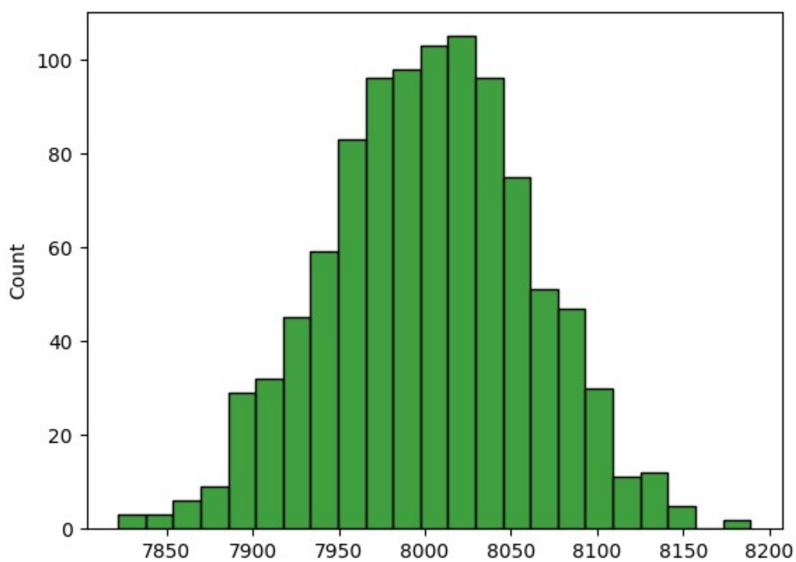
In [ ]:
```
sns.histplot(mean_age_0_17,color="g")
plt.show()
```



In [ ]:
```python
mean_age_18_25=[age_samp[age_samp["Age"]=="18-25"].sample(5000,replace=True)["Purchase"].mean() for i in range(1000)]
mean_age_18_25
```

Out[123]:
```
[8039.4694,
 7985.0492,
 7924.9892,
 7979.5588,
 7906.073,
 7914.0948,
 8023.8324,
 7936.7202,
 7988.6078,
 8017.46,
 8129.9744,
 8005.5894,
 7977.4638,
 8006.5288,
 8001.0094,
 8041.9286,
 7999.6122,
 7907.9422,
 8126.7104,
```

In [ ]:
```python
sns.histplot(mean_age_18_25,color="g")
plt.show()
```



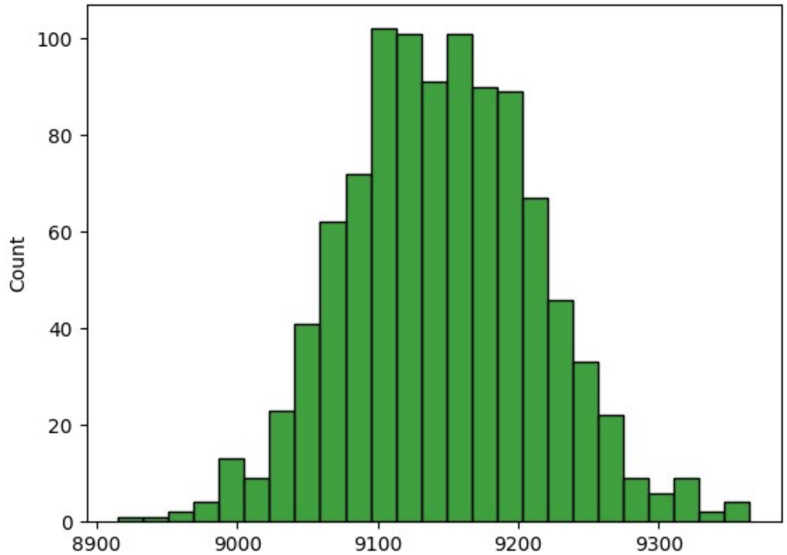
In [ ]:
```python
mean_age_26_35=[age_samp[age_samp["Age"]=="26-35"].sample(5000,replace=True)["Purchase"].mean() for i in range(1000)]
```

In [ ]:
```python
mean_age_26_35
```

Out[126]:
```
[9185.7222,
 9082.3524,
 9203.9032,
 9196.3088,
 9190.0052,
 9124.7888,
 9098.3694,
 9224.665,
 9115.1012,
 9181.624,
 9040.006,
 9231.8962,
 9166.7232,
 9257.8374,
 9166.0404,
 9134.4094,
 9197.5978,
 9060.1596,
 9179.8486,
```

In [ ]:

```
sns.histplot(mean_age_26_35,color="g")
plt.show()
```



In [ ]:

```
mean_age_36_45=[age_samp[age_samp["Age"]=="36-45"].sample(5000,replace=True)["Purchase"].mean() for i in range(1000)]
```

In [ ]:

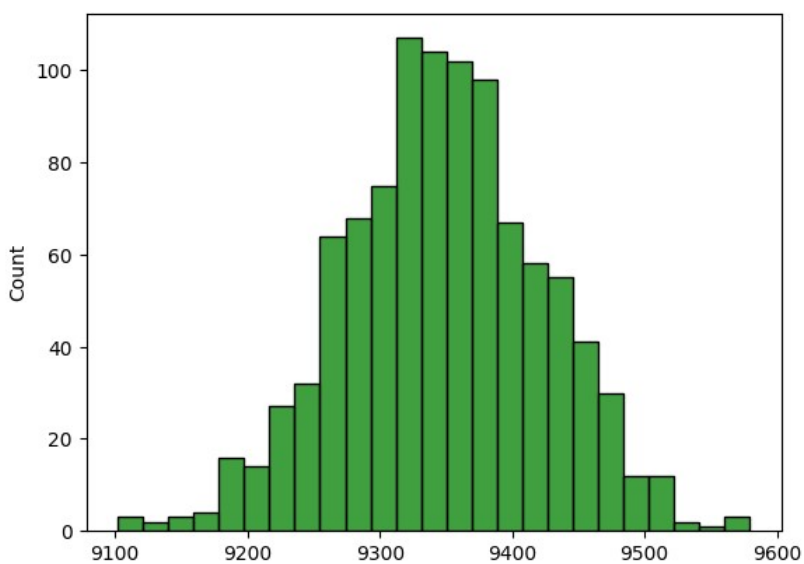
```
mean_age_36_45
```

Out[129]:

```
[9258.1046,
 9212.7248,
 9453.7844,
 9289.3382,
 9281.5978,
 9358.4958,
 9328.2398,
 9185.4274,
 9361.3434,
 9357.5696,
 9302.9732,
 9362.7176,
 9288.4222,
 9433.5556,
 9469.7048,
 9364.855,
 9327.8878,
 9335.2308,
 9340.9204,
```

In [ ]:

```
sns.histplot(mean_age_36_45,color="g")
plt.show()
```



In [ ]:

```
mean_age_46_50=[age_samp[age_samp["Age"]=="46-50"].sample(5000,replace=True)["Purchase"].mean() for i in range(1000)]
```

In [ ]:

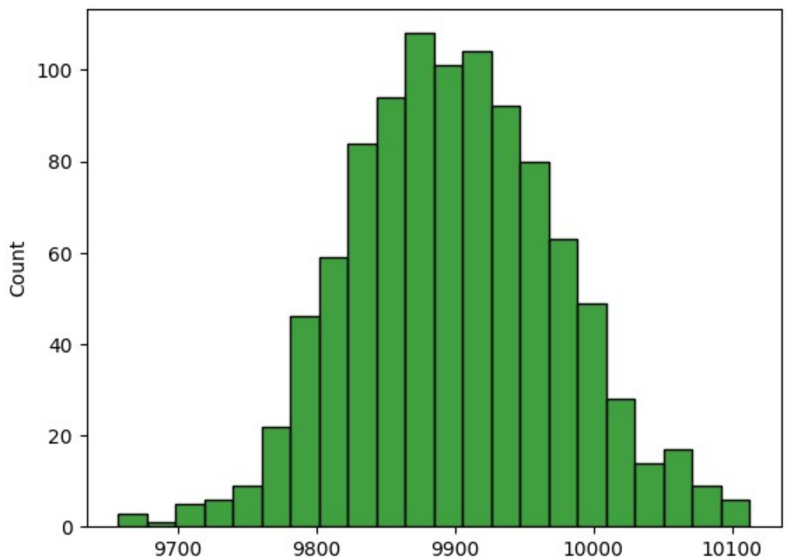
```
mean_age_46_50
```

Out[132]:

```
[9854.3494,
 9949.4214,
 9771.3532,
 9919.0922,
 9960.8132,
 9826.2538,
 9994.768,
 9800.1514,
 9979.6998,
 9874.6318,
 9942.7424,
 9897.1998,
 9733.6684,
 9939.0958,
 10112.588,
 9803.377,
 9834.3434,
 10052.6512,
 9938.6976,
```

In [ ]:

```
sns.histplot(mean_age_46_50,color="g")
plt.show()
```



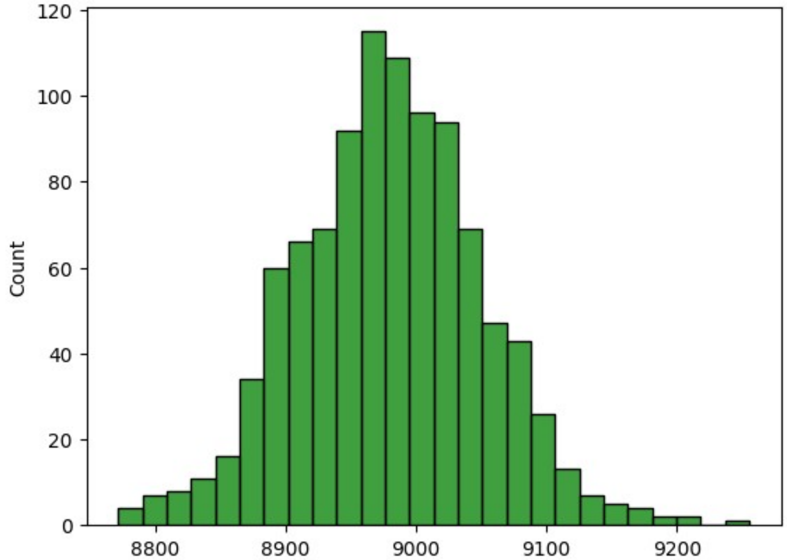
In [ ]:

```
mean_age_51_55=[age_samp[age_samp["Age"]=="51-55"].sample(5000,replace=True)["Purchase"].mean() for i in range(1000)]
mean_age_51_55
```

Out[134]:

```
[8957.1114,
 9030.222,
 8977.139,
 8961.4774,
 9053.9894,
 8976.8096,
 9013.0476,
 9029.4006,
 8900.0826,
 8954.1336,
 8956.266,
 8957.2332,
 9042.4236,
 8897.6702,
 8986.7538,
 9008.414,
 8857.3552,
 9018.7068,
 8986.956,
```

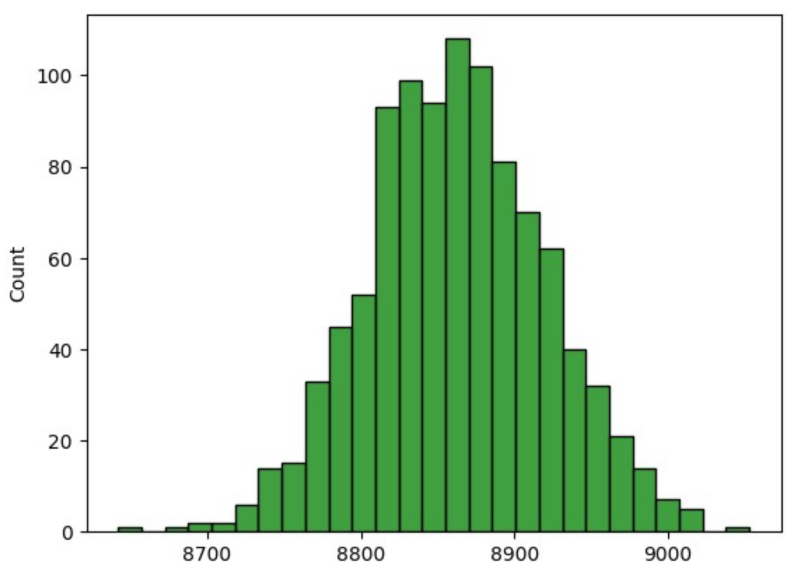
In [ ]:
```
sns.histplot(mean_age_51_55,color="g")
plt.show()
```



In [ ]:
```
mean_age_above_55=[age_samp[age_samp["Age"]=="55+"].sample(5000,replace=True)["Purchase"].mean() for i in range(1000)]
mean_age_above_55
```

Out[136]:
```
[9011.5828,
 8962.327,
 8867.7862,
 8857.9566,
 8720.6032,
 8771.3568,
 8916.6492,
 8946.5788,
 8835.8284,
 8961.013,
 8880.7448,
 8905.1346,
 8863.5658,
 8816.8728,
 8781.6356,
 8949.2558,
 8826.1178,
 8865.0438,
 8864.8244,
```

In [ ]:
```
sns.histplot(mean_age_above_55,color="g")
plt.show()
```



In [ ]:
```
# std deviation of age sample 0-17
np.std(mean_age_0_17).round(3)
```

Out[138]:
```
63.645
```

In [ ]:
```
# std deviation of age sample 18-25
np.std(mean_age_18_25).round(3)
```
Out[139]: 59.083

In [ ]:
```
# std deviation of age sample 26-35
np.std(mean_age_26_35).round(3)
```
Out[140]: 68.08

In [ ]:
```
# std deviation of age sample 36-45
np.std(mean_age_36_45).round(3)
```
Out[141]: 75.094

In [ ]:
```
# std deviation of age sample 46-50
np.std(mean_age_46_50).round(3)
```
Out[142]: 75.452

In [ ]:
```
# std deviation of age sample 51-55
np.std(mean_age_51_55).round(3)
```
Out[143]: 70.736

In [ ]:
```
# std deviation of age sample above 55
np.std(mean_age_above_55).round(3)
```
Out[144]: 58.157

CI--->95%

In [ ]:
```
# confidence Interval age (0-17):-
age_low_0_17=np.mean(mean_age_0_17)+norm.ppf(.025)*np.std(mean_age_0_17)
age_high_0_17=np.mean(mean_age_0_17)+norm.ppf(.975)*np.std(mean_age_0_17)
age_low_0_17.round(3),age_low_0_17.round(3)
```
Out[146]: (7662.86, 7662.86)

In [ ]:
```
# confidence Interval age (18-25):-
age_low_18_25=np.mean(mean_age_18_25)+norm.ppf(.025)*np.std(mean_age_18_25)
age_high_18_25=np.mean(mean_age_18_25)+norm.ppf(.975)*np.std(mean_age_18_25)
age_low_18_25.round(3),age_high_18_25.round(3)
```
Out[147]: (7886.164, 8117.766)

In [ ]:
```
# confidence Interval age(26-35):-
age_low_26_35=np.mean(mean_age_26_35)+norm.ppf(.025)*np.std(mean_age_26_35)
age_high_26_35=np.mean(mean_age_26_35)+norm.ppf(.975)*np.std(mean_age_26_35)
age_low_26_35.round(3),age_high_26_35.round(3)
```
Out[148]: (9012.337, 9279.207)

In [ ]:
```
# confidence Interval age(36-45):
age_low_36_45=np.mean(mean_age_36_45)+norm.ppf(.025)*np.std(mean_age_36_45)
age_high_36_45=np.mean(mean_age_36_45)+norm.ppf(.975)*np.std(mean_age_36_45)
age_low_36_45.round(3),age_high_36_45.round(3)
```
Out[150]: (9199.896, 9494.257)

In [ ]:
```
# confidence Interval age(46-50):-
age_low_46_50=np.mean(mean_age_46_50)+norm.ppf(.025)*np.std(mean_age_46_50)
age_high_46_50=np.mean(mean_age_46_50)+norm.ppf(.975)*np.std(mean_age_46_50)
age_low_46_50.round(3),age_high_46_50.round(3)
```
Out[151]: (9751.246, 10047.011)

In [ ]:
```
# Confidence Interval of age(51-55) = 95%
age_51_55_low = np.mean(mean_age_51_55) + norm.ppf(0.025) * (np.std(mean_age_51_55))
age_51_55_high = np.mean(mean_age_51_55) + norm.ppf(0.975) * (np.std(mean_age_51_55))
age_51_55_low.round(3), age_51_55_high.round(3)
```
Out[152]: (8840.982, 9118.26)

In [ ]:
```
# Confidence Interval of age(55+) = 95%
age_above_55_low = np.mean(mean_age_above_55) + norm.ppf(0.025) * (np.std(mean_age_above_55))
age_above_55_high = np.mean(mean_age_above_55) + norm.ppf(0.975) * (np.std(mean_age_above_55))
age_above_55_low.round(3), age_above_55_high.round(3)
```
Out[153]: (8748.004, 8975.974)

In [ ]:

```python
# To check overlapping of Confidence Intervals
age_0_17_CI = np.percentile(mean_age_0_17, [2.5, 97.5])
age_18_25_CI = np.percentile(mean_age_18_25, [2.5, 97.5])
age_26_35_CI = np.percentile(mean_age_26_35, [2.5, 97.5])
age_36_45_CI = np.percentile(mean_age_36_45, [2.5, 97.5])
age_46_50_CI = np.percentile(mean_age_46_50, [2.5, 97.5])
age_51_55_CI = np.percentile(mean_age_51_55, [2.5, 97.5])
age_above_55_CI = np.percentile(mean_age_above_55, [2.5, 97.5])
print("For age 00-17 --> confidence interval of means:" , age_0_17_CI.round(3))
print("For age 18-25 --> confidence interval of means:" , age_18_25_CI.round(3))
print("For age 26-35 --> confidence interval of means:" , age_26_35_CI.round(3))
print("For age 36-45 --> confidence interval of means:" , age_36_45_CI.round(3))
print("For age 46-50 --> confidence interval of means:" , age_46_50_CI.round(3))
print("For age 51-55 --> confidence interval of means:" , age_51_55_CI.round(3))
print("For age 56-++ --> confidence interval of means:" , age_above_55_CI.round(3))
```

```
For age 00-17 --> confidence interval of means: [7666.07  7911.188]
For age 18-25 --> confidence interval of means: [7889.916 8112.823]
For age 26-35 --> confidence interval of means: [9020.622 9282.387]
For age 36-45 --> confidence interval of means: [9195.166 9492.223]
For age 46-50 --> confidence interval of means: [ 9763.047 10054.9  ]
For age 51-55 --> confidence interval of means: [8836.541 9116.378]
For age 56-++ --> confidence interval of means: [8748.233 8977.387]
```

Comment

The confidence interval range is overlapping for some age bins while for some age bins it is not overlapping.

Insights from Data-

1) 59% Single, 41% Married.

2) 75% of the users are Male and 25% are Female.

3) nearly 80% of the users are between the age 18-50 (40%: 26-35, 18%: 18-25, 20%: 36-45).

4) Total of 20 product categories are there.

5) There are 20 differnent types of occupations in the city.

6) Customers mostly from city B(42%) followed by city C(31%) & then city A(27%).

7) 35% Staying in the city from 1 year, 18% from 2 years, 17% from 3 years.

8) From CLT graphs we have noticed that - a) for gender samples, the confidence interval range was not overlapping. b) for marital status samples, the confidence interval range was overlapping. c)for Age samples, the confidence interval range was overlapping for some age bins while for some age bins it was not overlapping.

Recommendations -

1) Unmarried customers spend more money than married customers, So in order to increase sales from married customers, walmart should give some discounts offers for married people .

2) As males are purchasing more as compared to females, walmart should retain the male customer. Also walmart should think to grow sales from female perspective like giving them some discounts or do advertise about the product to attract female customer base.

3) Customers in the age group of 18-25 are the favourable age range for the business, so walmart should retain these customers. Also for the age group which is less purchasing than the above mentioned age group, walmart should come up with some ideas to involve those age groups in order to increase the sales.

4) Walmart have strong customer base in 'City C', so walmart should retain these customers, Also walmart should think to change strategies in 'City B' & 'City A' in order to increase the sales in those cities as well. Walmart can do advertising using online flatforms such as social digital flatforms such as Youtube, Instagram.

5) There are some product categories such as 1, 5, 8 & 11 which are purchased by most of the customers. So, walmart can focus on the product categories other than this so that the sales from all other categories would be increase at some sufficient level.