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

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

In [430…
```
#Importing necessary python libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
```

In [104…
```
df = pd.read_csv('/Users/bose/Downloads/walmart.csv')
```

In [105…
```
df.head()
```

Out[105]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Yea
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	

In [106…
```
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 [107…

```
df.shape
```

Out[107]:

```
(550068, 10)
```

There are **550068 rows** and **10 columns** in the dataset

In [108…

```
# Checking for null values
df.isna().sum().sum()
```

Out[108]:

```
0
```

There are **No Null values** in the dataset

In [109…

```
#Checking for duplicate values
df[df.duplicated()]
```

Out[109]:

User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Year

There are **No Duplicate values** in the dataset

In [110…

```
# Datatype of attributes
df.dtypes
```

Out[110]:

```
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 [111…

```
#Columns of dataset
df.columns
```

Out[111]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Catego
ry',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

Statistical Summary -

In [112…
```
df.describe()
```

Out[112]:

	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 [113…
```
df.describe(include='object')
```

Out[113]:

	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

In [114…
```
# Finding number of users
df['User_ID'].nunique()
```

Out[114]:
```
5891
```

There are **5891** users in the given dataset

In [115…
```
# Finding number of products
df['Product_ID'].nunique()
```

Out[115]:
```
3631
```

There are **3631** unique products in the given dataset

Non Graphical Analysis -

Value counts and Unique elements of each column -

User_Id Column

In [116…
```
df['User_ID'].value_counts()
```

Out[116]:
```
1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
           ...
1002690       7
1002111       7
1005810       7
1004991       7
1000708       6
Name: User_ID, Length: 5891, dtype: int64
```

In [117…
```
print('Unique values of User_ID column are :',df['User_ID'].unique())
```

```
Unique values of User_ID column are : [1000001 1000002 1000003 ... 1004113 1
005391 1001529]
```

In [118…
```
df['User_ID'].nunique()
```

Out[118]:
```
5891
```

Observation -

- There are **5891 users** in the given dataset
- Top customers is customer with User_ID : 1001680

Product_ID column

In [119…
```
df['Product_ID'].value_counts()
```

Out[119]:
```
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
             ...
P00314842       1
P00298842       1
P00231642       1
P00204442       1
P00066342       1
Name: Product_ID, Length: 3631, dtype: int64
```

In [120…
```
print('Unique values of Product_ID column are :',df['Product_ID'].unique())
```

```
Unique values of Product_ID column are : ['P00069042' 'P00248942' 'P0008784
2' ... 'P00370293' 'P00371644'
 'P00370853']
```

In [121…
```
df['Product_ID'].nunique()
```

Out[121]:
```
3631
```

Observation -

- There are **3631 products** in the given dataset
- Top product is product with Product_ID : P00265242

Gender Column

In [122…
```
df['Gender'].value_counts()
```

Out[122]:
```
M    414259
F    135809
Name: Gender, dtype: int64
```

In [123…
```
print('Unique values of Gender column are :',df['Gender'].unique())
```

```
Unique values of Gender column are : ['F' 'M']
```

In [124…
```
df['Gender'].nunique()
```

Out[124]:
```
2
```

In [299…
```
df['Gender'].value_counts(normalize=True).round(2)*100
```

Out[299]:
```
M    75.0
F    25.0
Name: Gender, dtype: float64
```

Observations -

- **Gender** consist of **2** unique values - **Male and Female**
- No of **Male customers** - **4,14,259**
- No of **Female customers** - **1,25,809**
- **Male customers** account for **75%** of total customers
- Wheres **Female customers** account for **25%** of total customers

Age Column

In [126…
```
df['Age'].value_counts()
```

Out[126]:
```
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
```

In [127…
```
print('Unique values of Age column are :',df['Age'].unique())
```

```
Unique values of Age column are : ['0-17' '55+' '26-35' '46-50' '51-55' '36-
45' '18-25']
```

In [128…
```
df['Age'].nunique()
```

Out[128]:
```
7
```

In [129…
```
df['Age'].value_counts(normalize=True).round(2)*100
```

Out[129]:
```
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
```

Observations -

- **Age** column consist of **7** unique categories
- Most number of customers **(40%)** come under the age group **(26-35)**
- Least no of customers **(3%)** come under the age group **(0-17)**

Occupation Column

In [130…
```
df['Occupation'].value_counts()
```

Out[130]:
```
4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: Occupation, dtype: int64
```

In [131…
```
print('Unique values of Occupation column are :',df['Occupation'].unique())
```

```
Unique values of Occupation column are : [10 16 15  7 20  9  1 12 17  0  3
 4 11  8 19  2 18  5 14 13  6]
```

In [132…
```
df['Occupation'].nunique()
```

Out[132]:
```
21
```

In [133…
```
df['Occupation'].value_counts(normalize=True).round(2)*100
```

Out[133]:
```
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
```

Observation -

- There are **21** unique categories for Occupation column
- Highest number of people come in the category **4 and 0** (13% each)
- Least number of people come in the category **8**

City_Category Column

In [134…
```
df['City_Category'].value_counts()
```

Out[134]:
```
B    231173
C    171175
A    147720
Name: City_Category, dtype: int64
```

In [135…
```
print('Unique values of City_Category column are :',df['City_Category'].uniq
```

```
Unique values of City_Category column are : ['A' 'C' 'B']
```

In [136…
```
df['City_Category'].nunique()
```

Out[136]:
```
3
```

In [137…
```
df['City_Category'].value_counts(normalize=True).round(2)*100
```

Out[137]:
```
B    42.0
C    31.0
A    27.0
Name: City_Category, dtype: float64
```

Observation -

- There are **3** unique values in **City_Category** column
- Unique Values are **A, B and C**
- Most no of customers come in the category **B** (42%)
- Least no of customers come in the category **A** (27%)

Stay_In_Current_City_Years Column

In [138…
```
df['Stay_In_Current_City_Years'].value_counts()
```
Out[138]:
```
1      193821
2      101838
3       95285
4+      84726
0       74398
Name: Stay_In_Current_City_Years, dtype: int64
```

In [139…
```
df['Stay_In_Current_City_Years'].nunique()
```
Out[139]:
```
5
```

In [140…
```
df['Stay_In_Current_City_Years'].unique()
```
Out[140]:
```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

In [141…
```
df['Stay_In_Current_City_Years'].value_counts(normalize=True).round(2)*100
```
Out[141]:
```
1      35.0
2      19.0
3      17.0
4+     15.0
0      14.0
Name: Stay_In_Current_City_Years, dtype: float64
```

Observation -

- There are **5** unique values in **Stay_In_Current_City_Years** column
- Unique Values are **'2', '4+', '3', '1', '0'**
- Majority of customers have stayed in their current city for **1 year** (35%)
- Least no of customers have stayed **less than a year** (19%) in their current city

Marital_Status Column

In [142…
```
df['Marital_Status'].value_counts()
```
Out[142]:
```
0    324731
1    225337
Name: Marital_Status, dtype: int64
```

In [143…
```
df['Marital_Status'].nunique()
```
Out[143]:
```
2
```

In [144…
```
df['Marital_Status'].unique()
```
Out[144]:
```
array([0, 1])
```

In [145…
```
df['Marital_Status'].value_counts(normalize=True).round(2)*100
```
Out[145]:
```
0    59.0
1    41.0
Name: Marital_Status, dtype: float64
```

Observation -

- There are **2** unique values in **Marital_Status** column
- Unique Values are **0 and 1**

- **Single** customers account for **59%** of total
- **Married** customers account for **41%** of total

Product_Category Column

In [146…
```
df['Product_Category'].value_counts()
```

Out[146]:
```
5     150933
1     140378
8     113925
11     24287
2      23864
6      20466
3      20213
4      11753
16      9828
15      6290
13      5549
10      5125
12      3947
7       3721
18      3125
20      2550
19      1603
14      1523
17       578
9        410
Name: Product_Category, dtype: int64
```

In [147…
```
df['Product_Category'].nunique()
```

Out[147]:
```
20
```

In [148…
```
print('Unique values of Product_Category column are :',df['Product_Category'
```

```
Unique values of Product_Category column are : [ 3  1 12  8  5  4  2  6 14 1
1 13 15  7 16 18 10 17  9 20 19]
```

In [149…
```
df['Product_Category'].value_counts(normalize=True).round(2)*100
```

Out[149]:
```
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
```

Observation -

- There are **20** unique **Product_Category** columns
- Most products come under **Product_Category 5** (27%)
- Least no of products come in category **9, 14, 17, 19, 20** (0%)

Purchase Column

In [150…
```
df['Purchase'].value_counts()
```

Out[150]:
```
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
```

In [151…
```
df['Purchase'].nunique()
```

Out[151]:
```
18105
```

In [152…
```
print('Unique values of Purchase column are :',df['Purchase'].unique())
```

```
Unique values of Purchase column are : [ 8370 15200  1422 ...   135   123
613]
```

In [153…
```
df['Purchase'].aggregate([min,max])
```

Out[153]:
```
min       12
max    23961
Name: Purchase, dtype: int64
```

Observation -

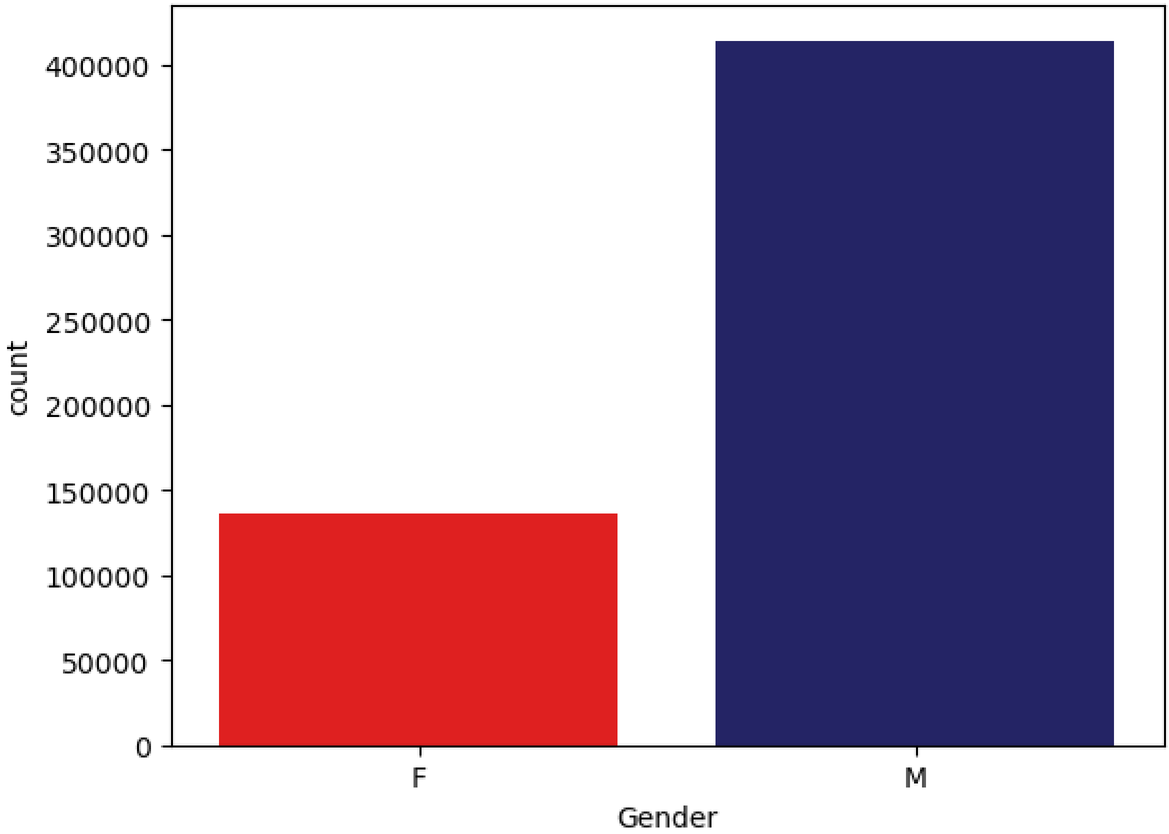
- There are **18,105** unique values in the **Purchase** column
- **Highest purchase** value is **23,961**
- **Lowest purchase** value is **12**

Visual Analysis

Univariate Analysis

For Categorical Variables -

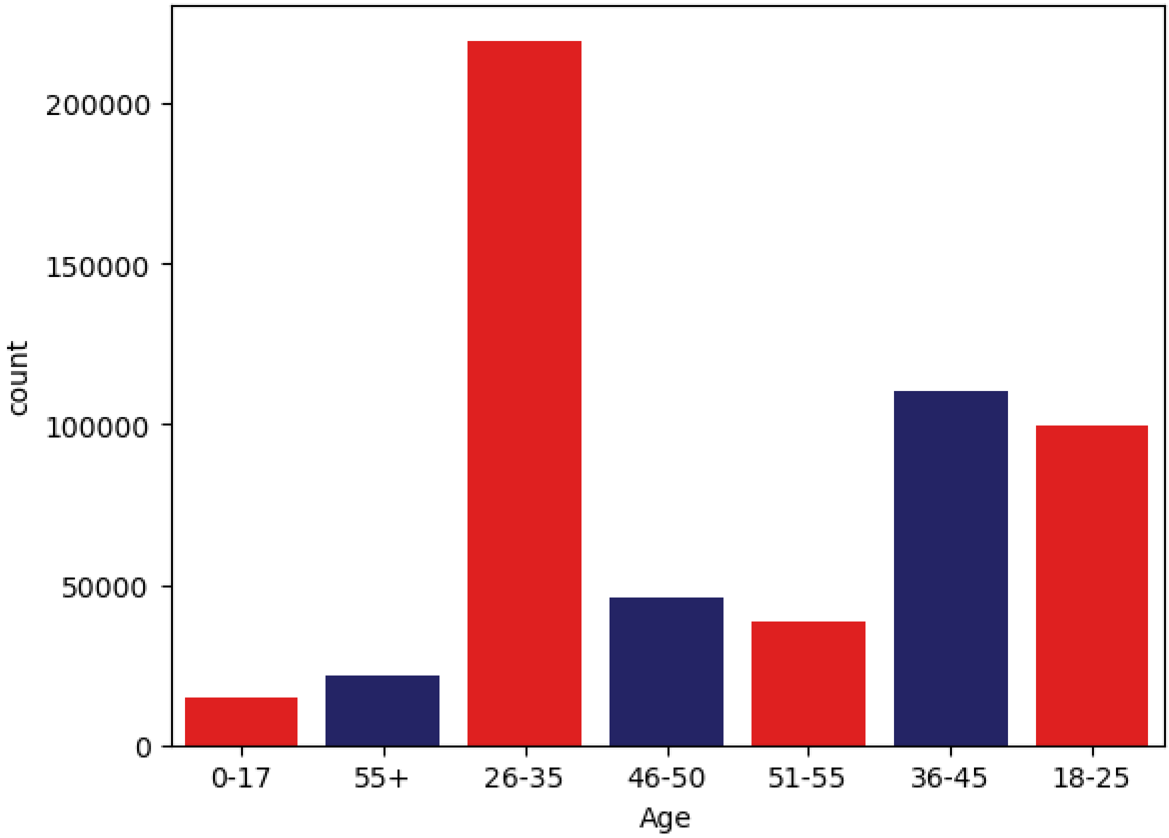
In [260…
```
sns.countplot(data=df,x='Gender',palette=['red','midnightblue'])
plt.show()
```



Observation -

- **Male customers are more** compared to **Female customers**
- Male customers - 75% of total customers
- Female customers -25% of total customers

In [262…
```
sns.countplot(data=df,x='Age',palette=['red','midnightblue'])
plt.show()
```

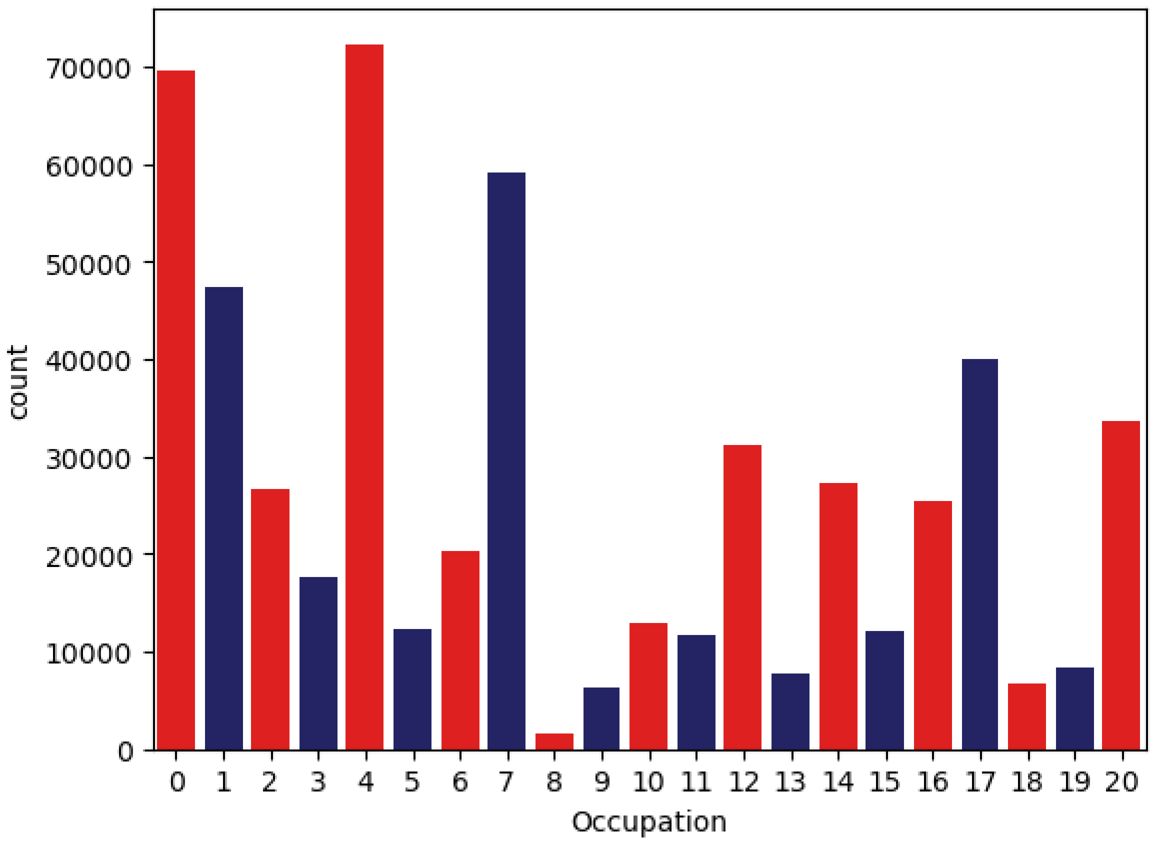


Observation -

- **Majority of customers** fall in the age group **(26-35)**
- Age group with **least no of customers** are **(0-17)**

In [264…

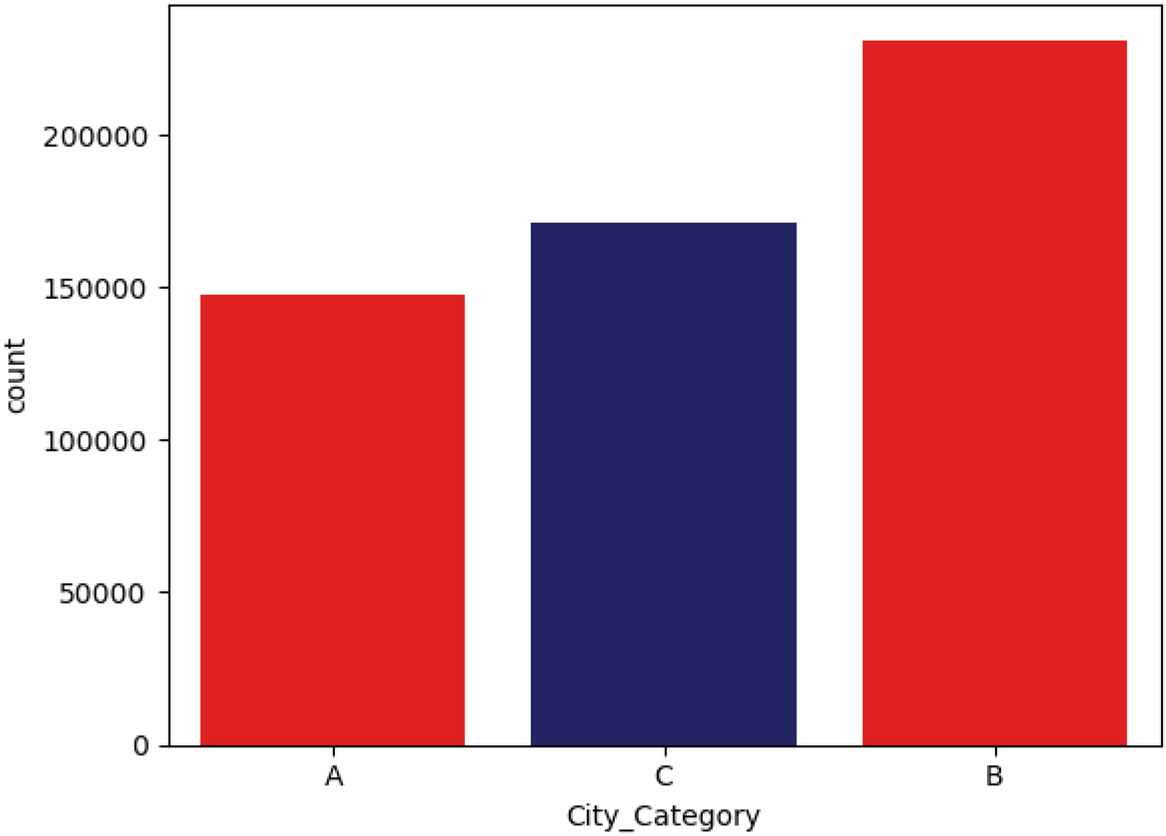
```
sns.countplot(data=df,x='Occupation',palette=['red','midnightblue'])
plt.show()
```



Observation -

- **Majority of customers** comes from occupation group **0 and 4**
- **Least no of customers** comes from occupation group **8**

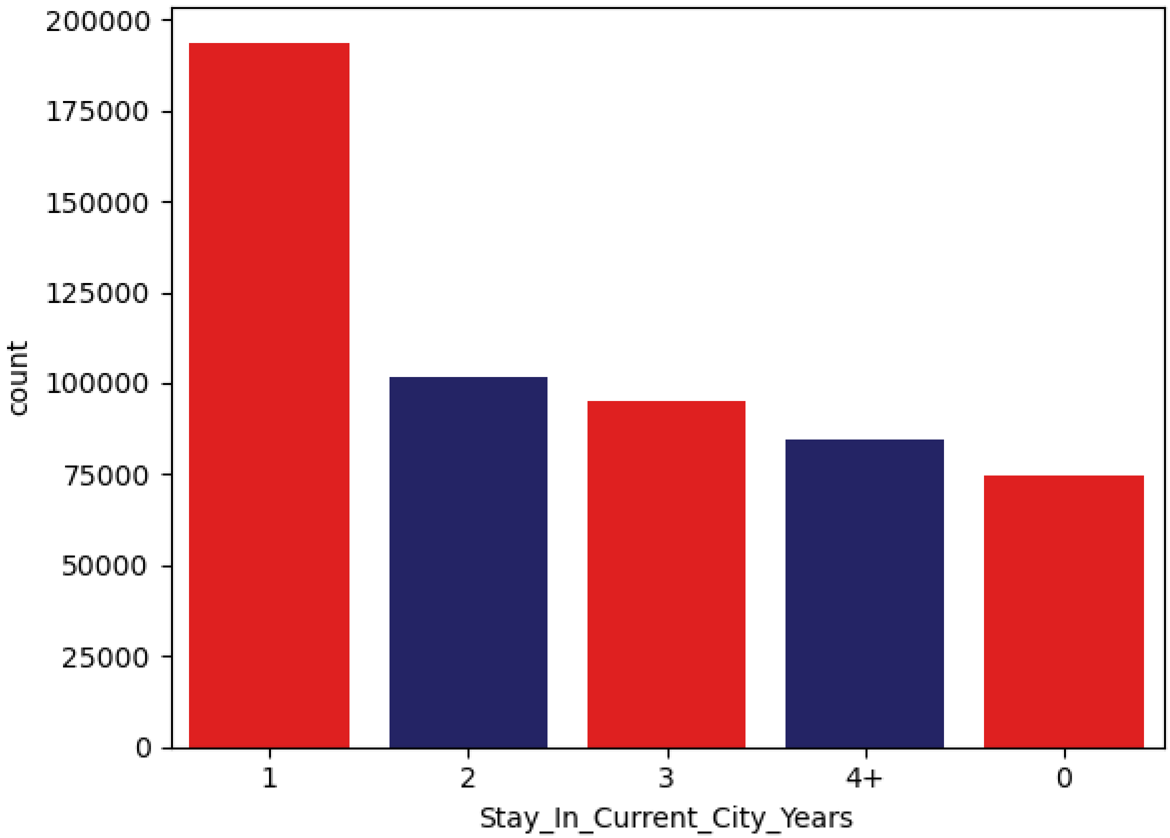
In [265…
```
sns.countplot(data=df,x='City_Category',palette=['red','midnightblue'])
plt.show()
```



Observation -

- **Majority of customers** come from **City B**
- **Least no of customers** come from **City A**

In [270…
```
sns.countplot(data=df,x='Stay_In_Current_City_Years',palette=['red','midnigh
plt.show()
```

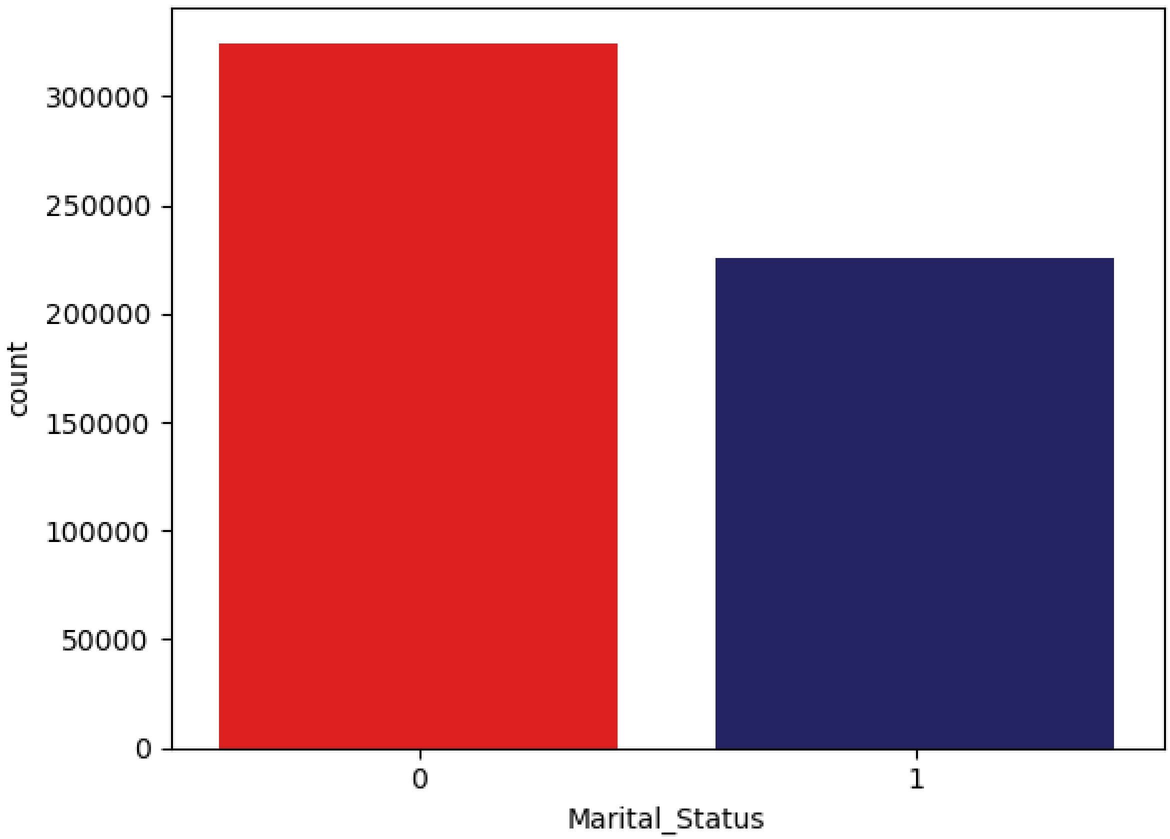


Observation -

- **Majority of customers** have been staying in their **Current City for 1 years**
- **Least no of customers** have stayed in the current city for **less than a year**

In [271…

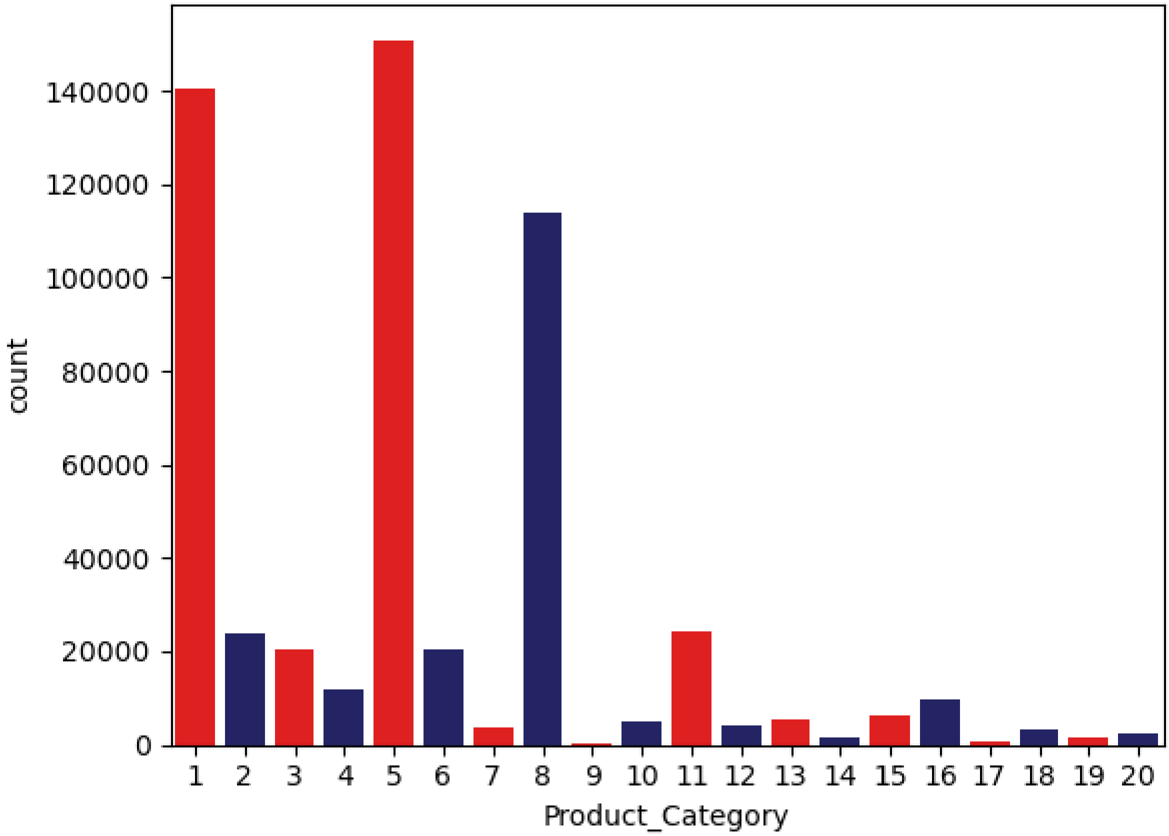
```
sns.countplot(data=df,x='Marital_Status',palette=['red','midnightblue'])
plt.show()
```



Observation -

- Number of **Married people** are **less** compared to number of **Unmarried people**

In [272…
```
sns.countplot(data=df,x='Product_Category',palette=['red','midnightblue'])
plt.show()
```

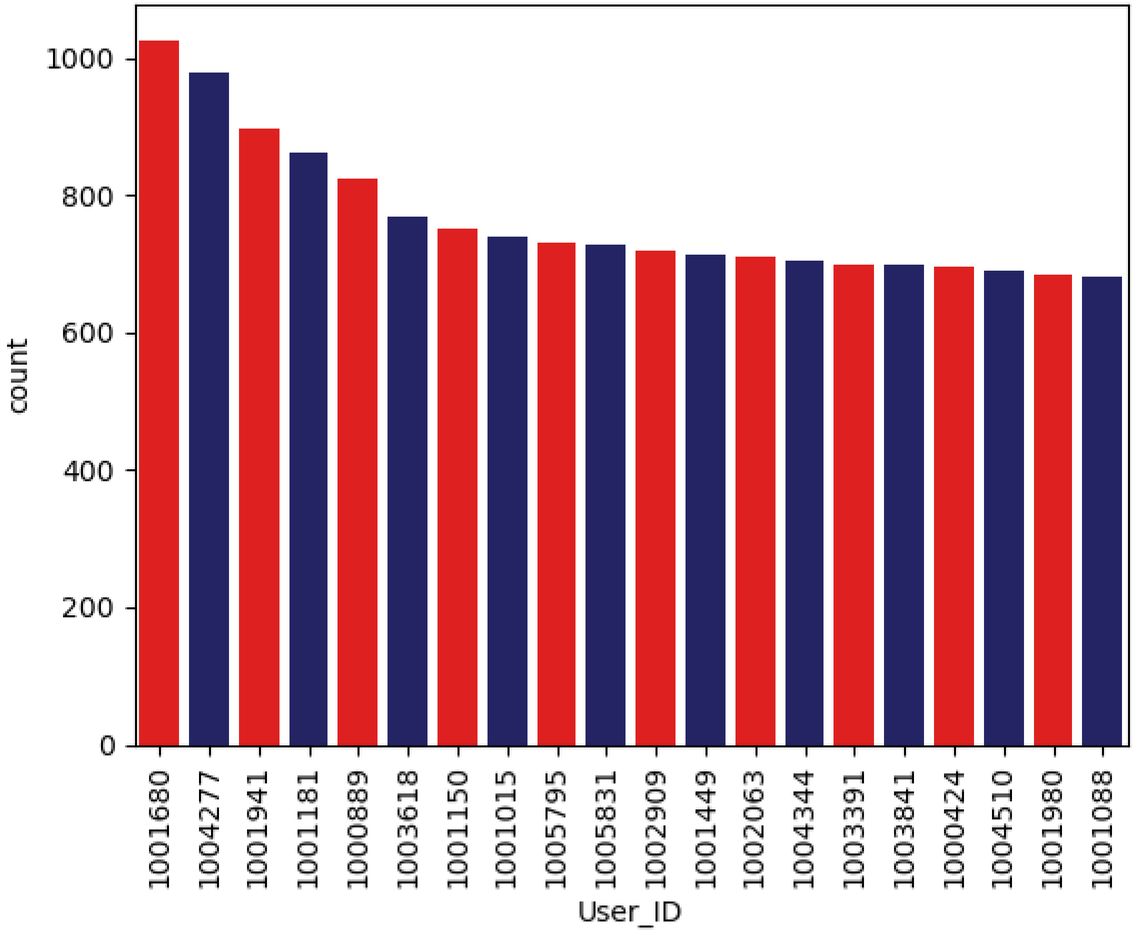


Observation -

- **Majority of products** come under the category **5**
- **Least no of products** come in the category **9, 14, 17, 19, 20**

For Continuous Variables-

In [281…
```
sns.countplot(data=df,x='User_ID',palette=['red','midnightblue'],order=df['U
plt.xticks(rotation=90)
plt.show()
```

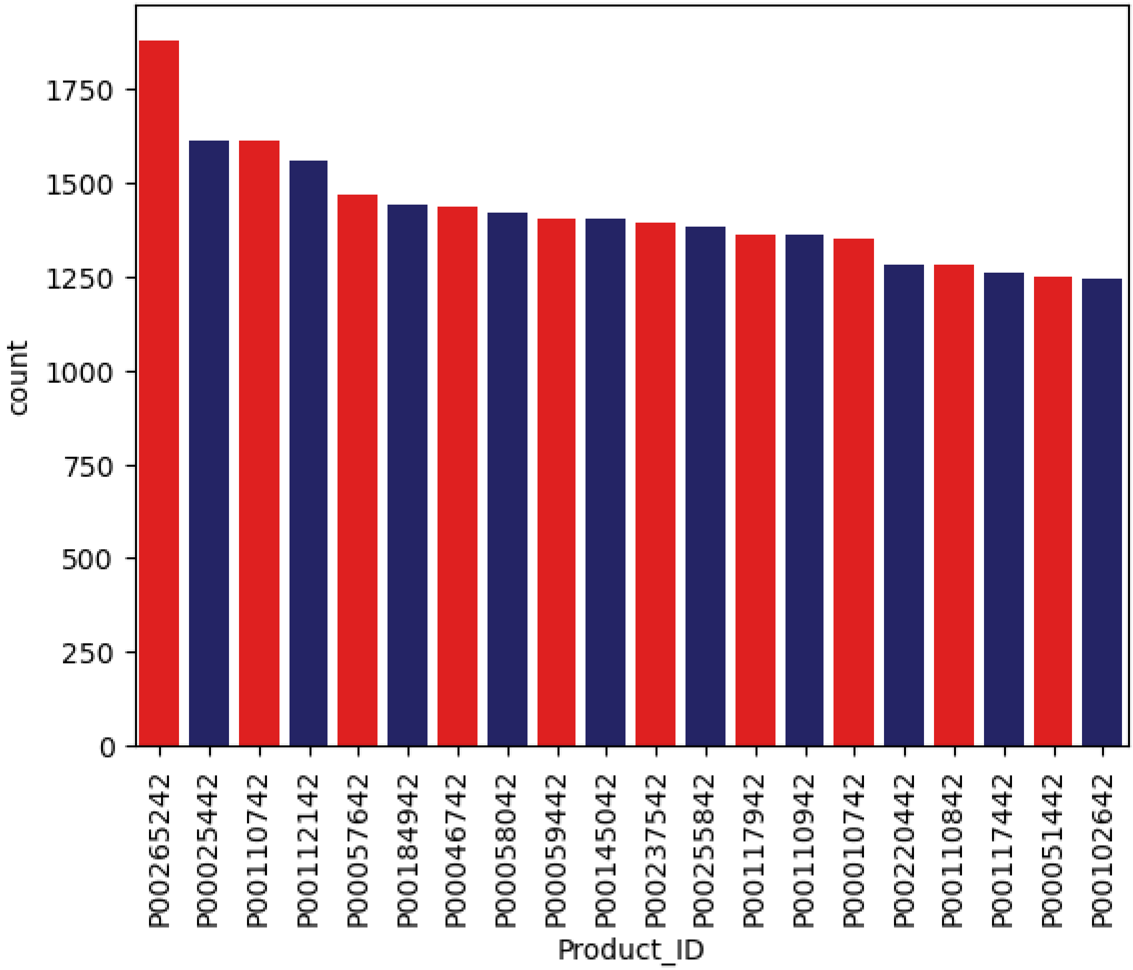


Observation -

- **User ID 1001680** is the **most frquent customer** from the given dataset

In [282…

```
sns.countplot(data=df,x='Product_ID',palette=['red','midnightblue'],order=df
plt.xticks(rotation=90)
plt.show()
```

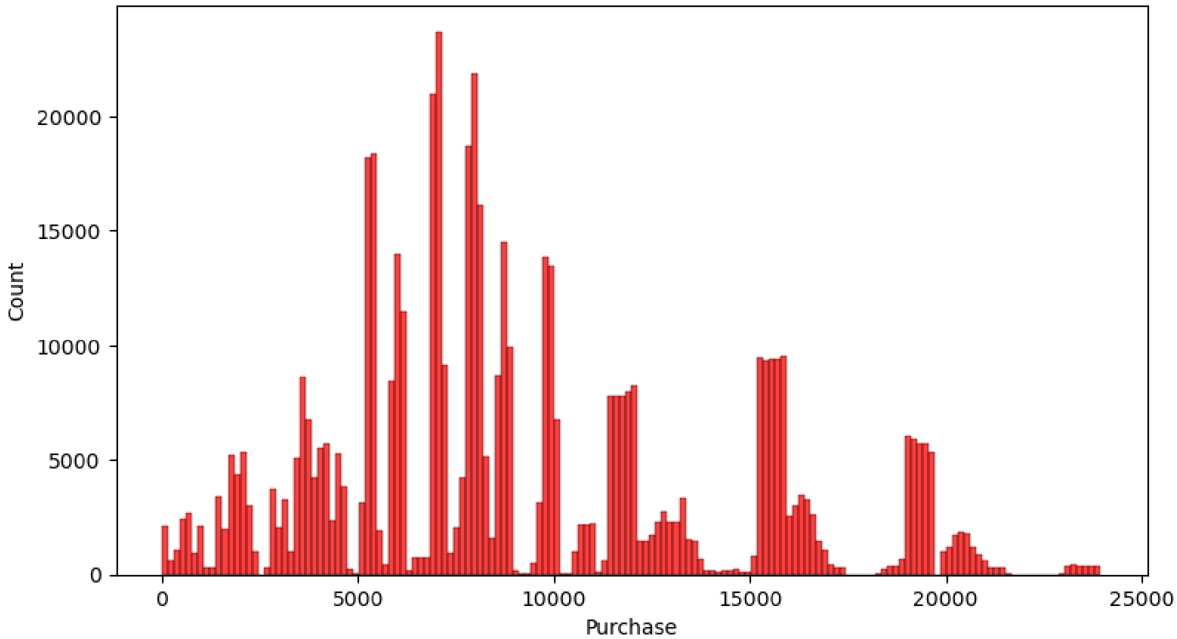


Observation -

- **Product ID P00265242** is the **most sold product** from the given dataset

In [298…

```
plt.figure(figsize = (9, 5))
sns.histplot(df["Purchase"], color = 'red')
plt.show()
```

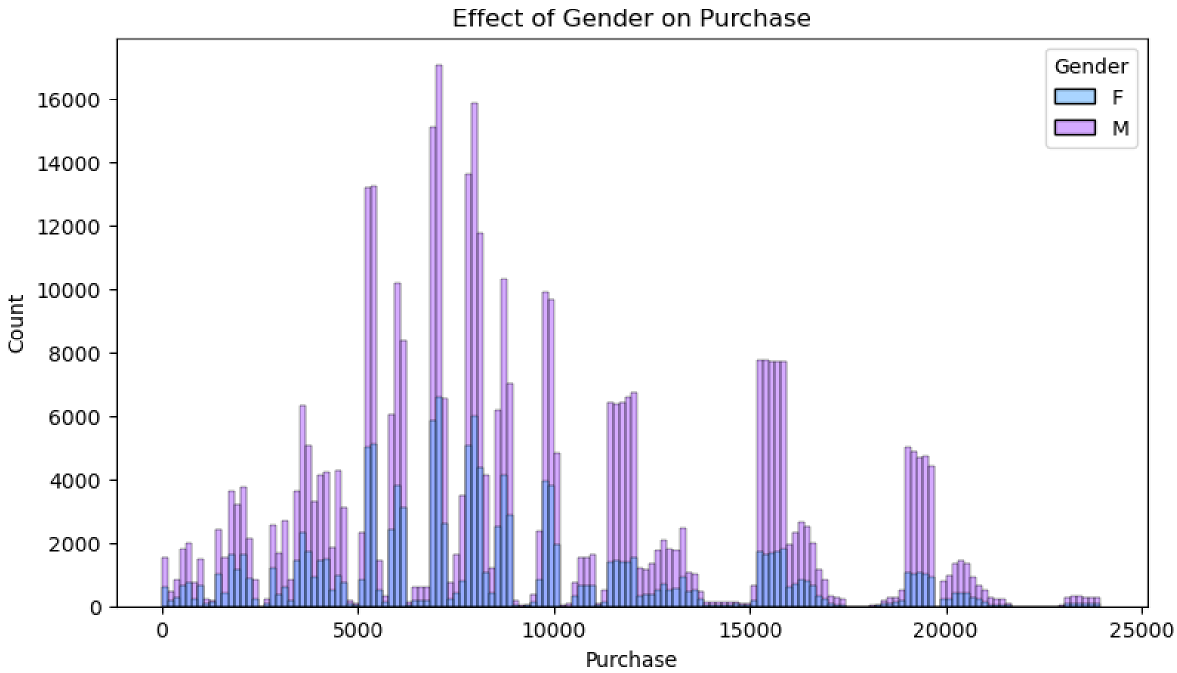


Observation -

- Purchase amount ranges from **12 to 23,961**
- **Meadian purchase amount is 8047**

Bivariate Analysis -

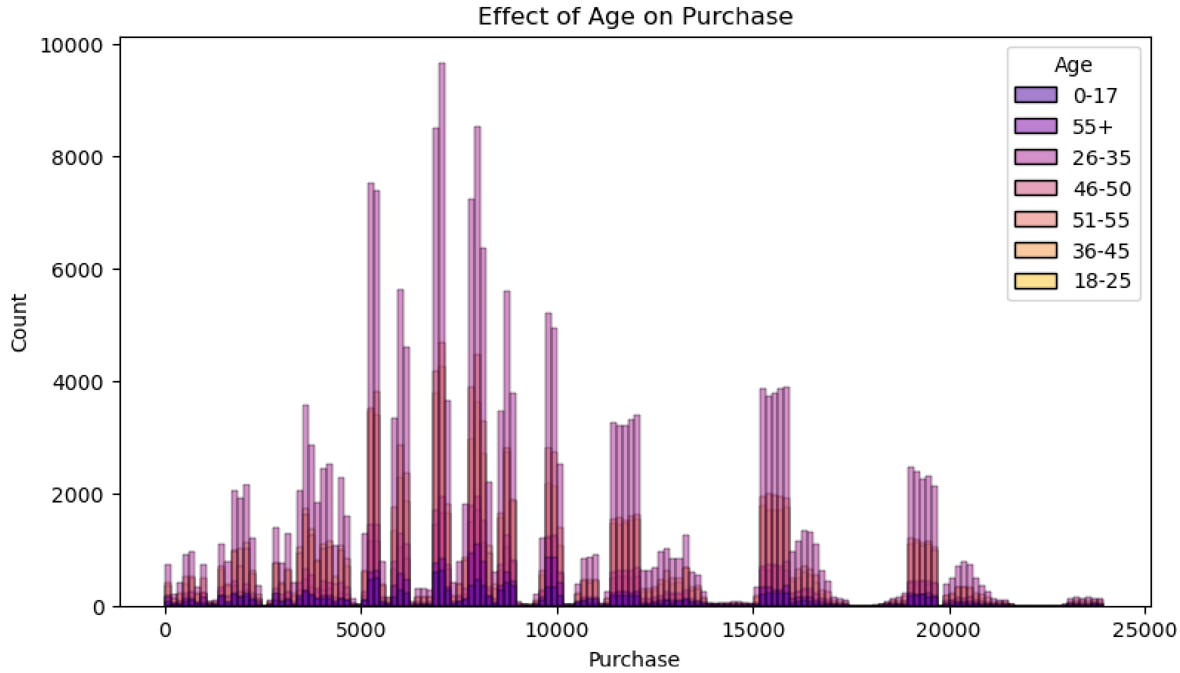
In [339…
```
plt.figure(figsize=(9,5))
plt.title('Effect of Gender on Purchase')
sns.histplot(data=df, x='Purchase', hue='Gender', palette='cool')
plt.show()
```



Observation -

- **Male customers tend to purchase more than Female**

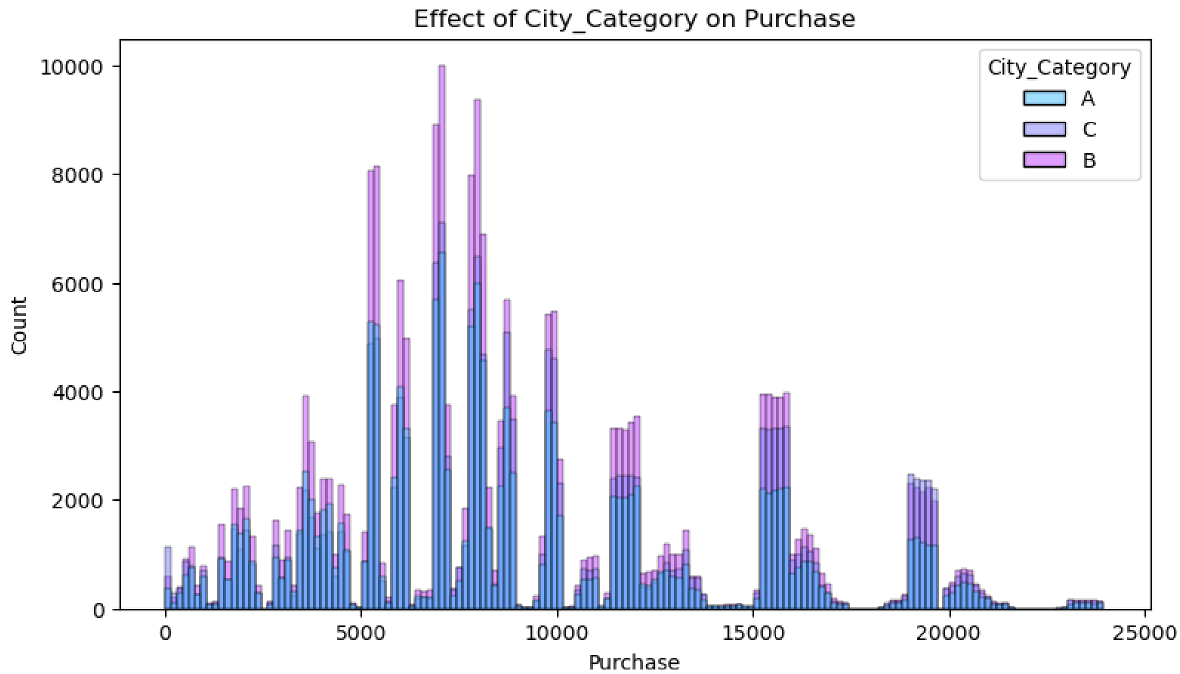
In [340…
```
plt.figure(figsize=(9,5))
plt.title('Effect of Age on Purchase')
sns.histplot(data=df, x='Purchase', hue='Age', palette='plasma')
plt.show()
```



Observation -

- **Highest number** of customers fall in the **Age group (26-35)**
- **Least number** of customers are from **Age group (0-17)**

In [401…
```
plt.figure(figsize=(9,5))
plt.title('Effect of City_Category on Purchase')
sns.histplot(data=df, x='Purchase', hue='City_Category', palette='cool')
plt.show()
```

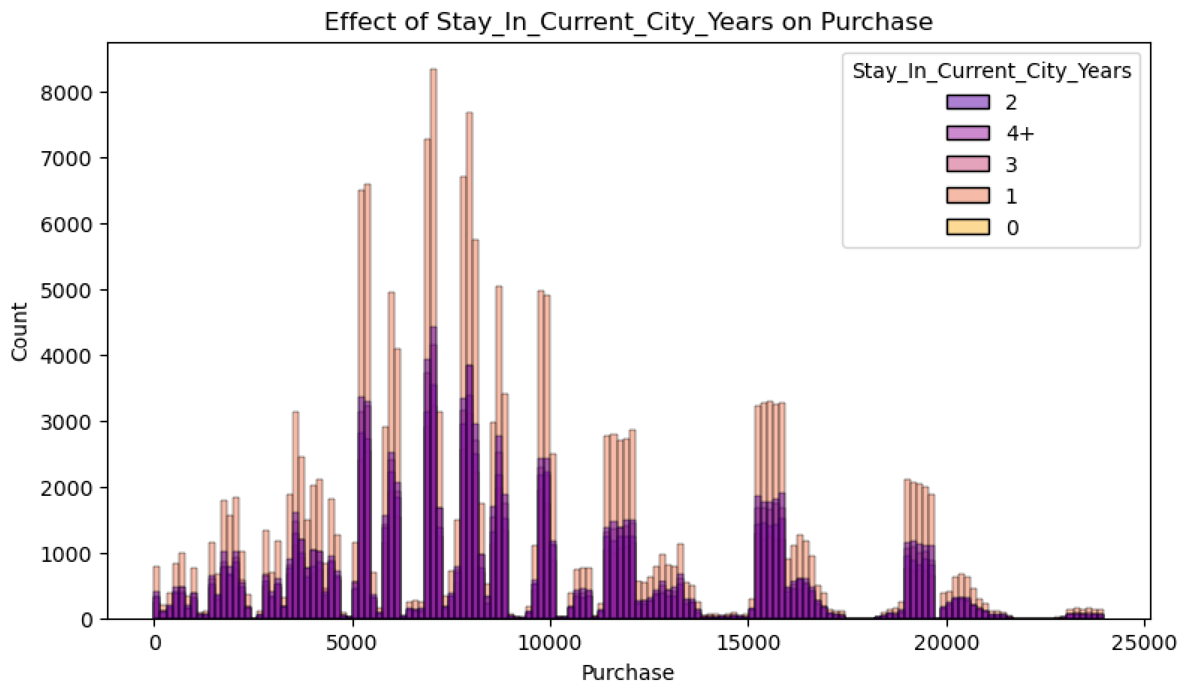


Observation -

- **Most number** of customers come from **City B**
- **Least number** of customers come from **City A**

In [348…
```
plt.figure(figsize=(9,5))
plt.title('Effect of Stay_In_Current_City_Years on Purchase')
```

```
sns.histplot(data=df, x='Purchase', hue='Stay_In_Current_City_Years', palett
plt.show()
```

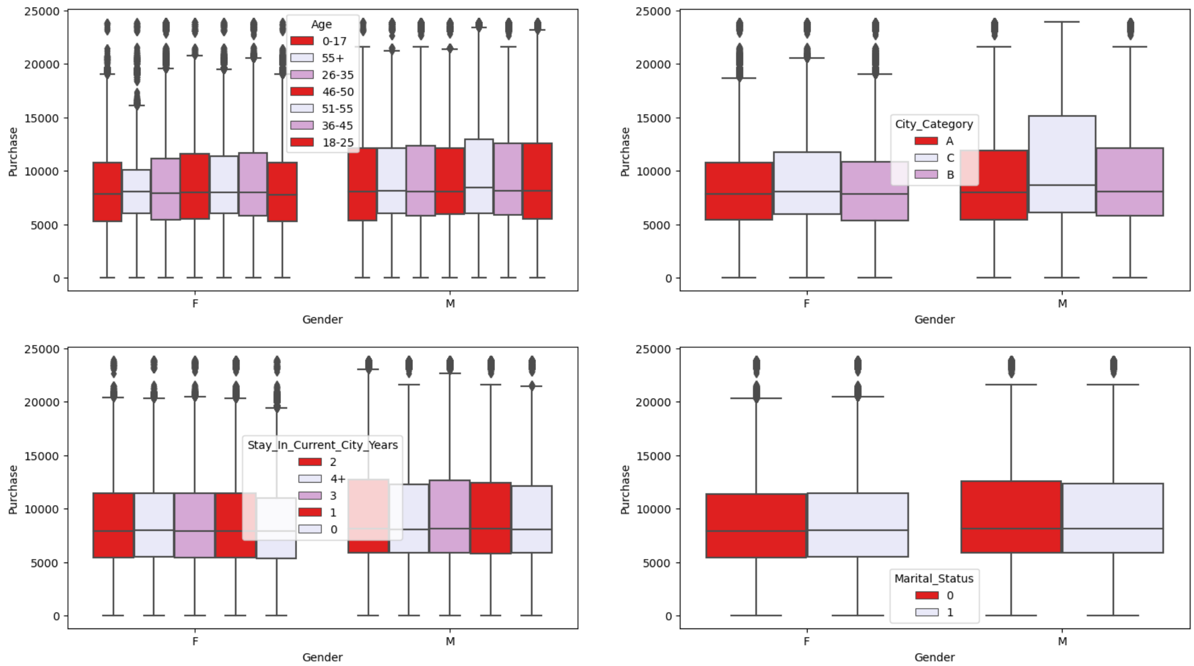


Observation -

- **Majority of customers** have been living in their current city for **1 year (35%)**
- It is followed by customer who have been living for **2 years (18%)** and **3 years (17%)**

In [366…

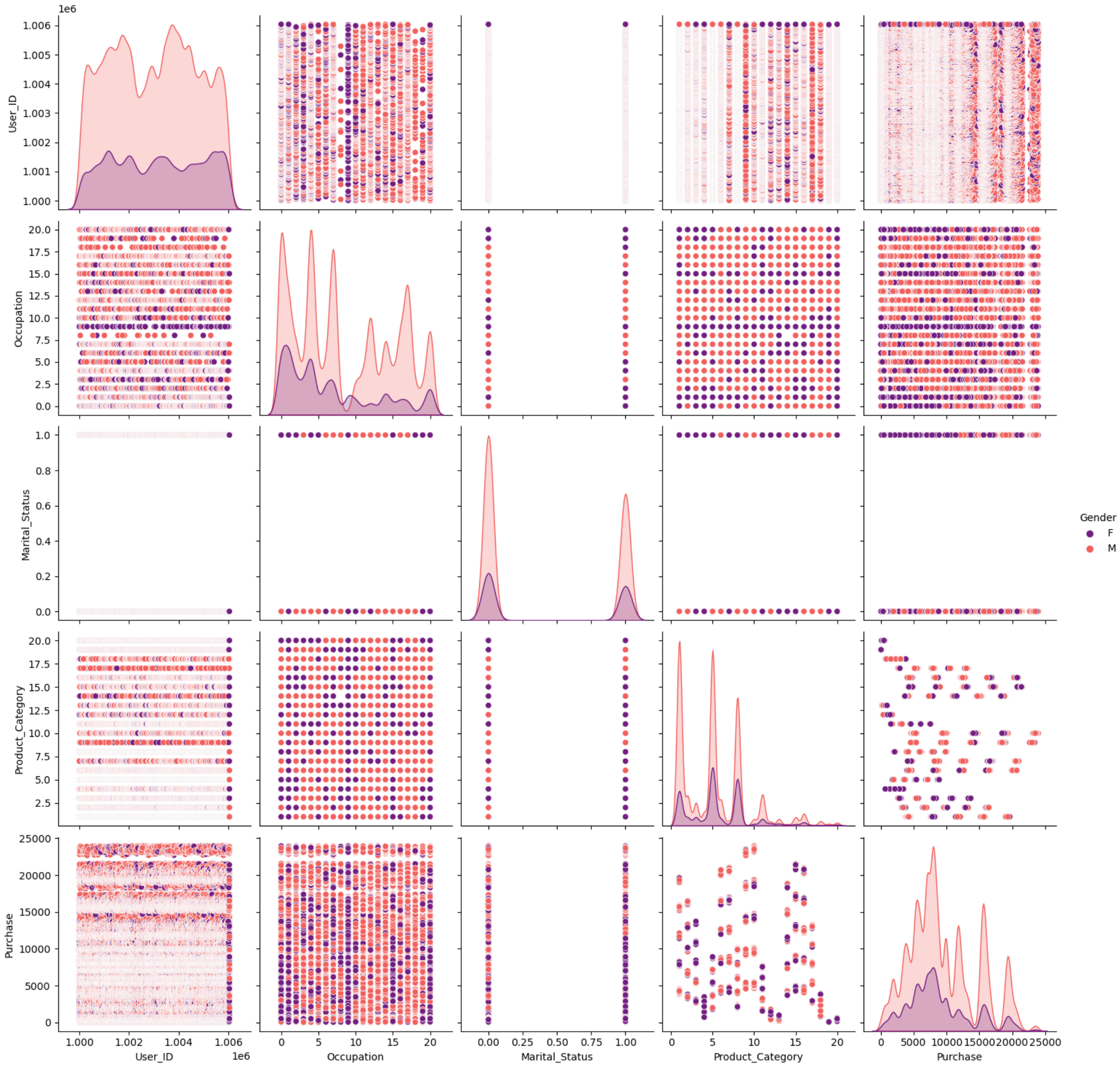
```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(18, 10))

sns.boxplot(x="Gender", y="Purchase", data=df, hue='Age', ax=axis[0,0], pale
sns.boxplot(data=df, x="Gender" ,y="Purchase", ax=axis[0,1], hue='City_Categ
sns.boxplot(data=df, x="Gender", y="Purchase", ax=axis[1,0], hue='Stay_In_Cu
sns.boxplot(data=df, x="Gender", y="Purchase", ax=axis[1,1],hue='Marital_Sta
plt.show()
```



Observation -

- Median purchase amount for Male is more than Female
- Median purchase value for differnt age groups is very close
- Median purchase value is more for City C compared to others
- Median purchase value for stay in current city is nearly same for all categories
- Median purchase value is same for Single and Married people

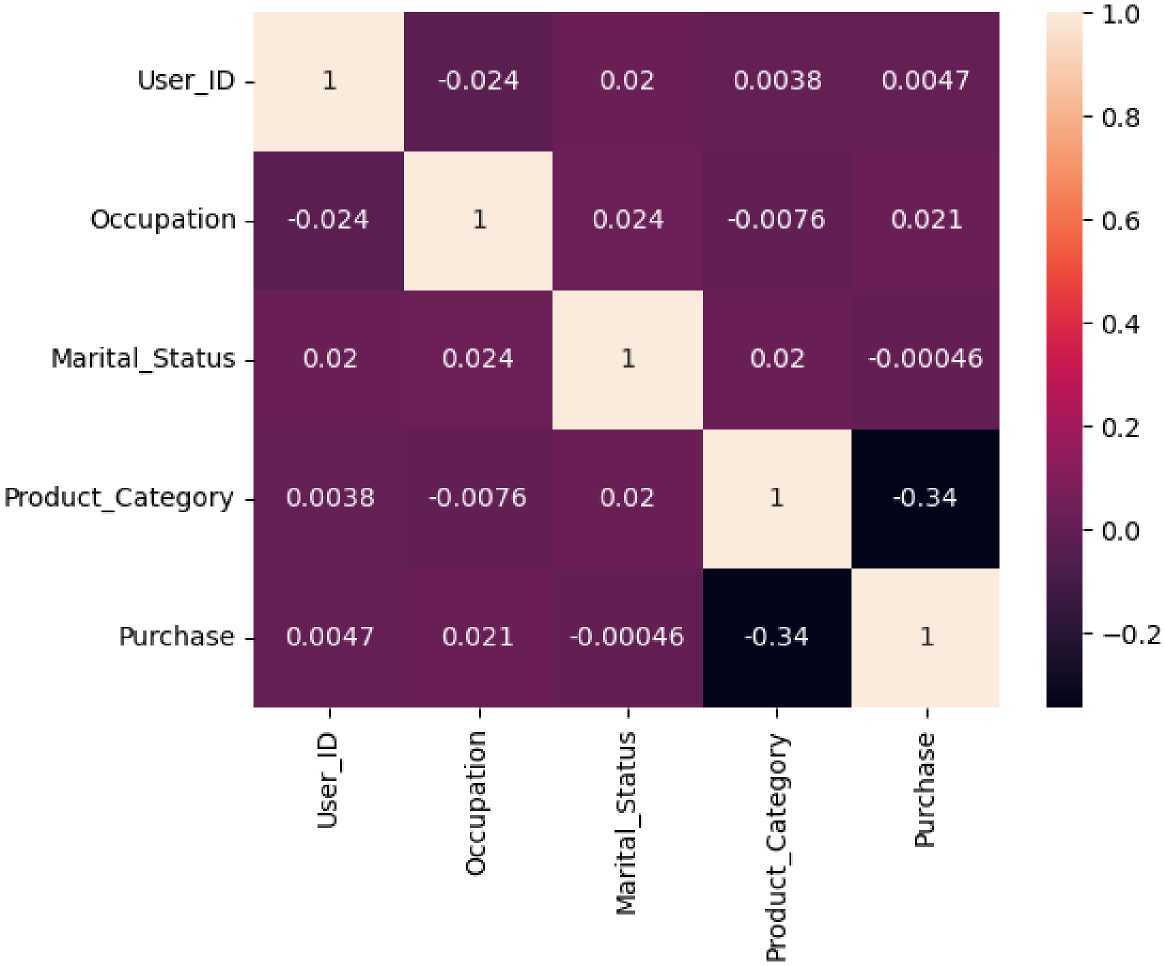
In [353…
```
sns.pairplot(data=df, hue='Gender', palette= 'magma', height=3)
plt.show()
```



Observation -

- Purchase Value seems to be higher for Male than Female

In [350…
```
sns.heatmap(df.corr(), annot=True)
plt.show()
```



Observations -

- There is not much correlation between any data
- Purchase has a negative correlation with Product_Category

Missing Value and Outlier Detection -

In [351…

```
# Check for missing or null values
df.isna().sum()
```

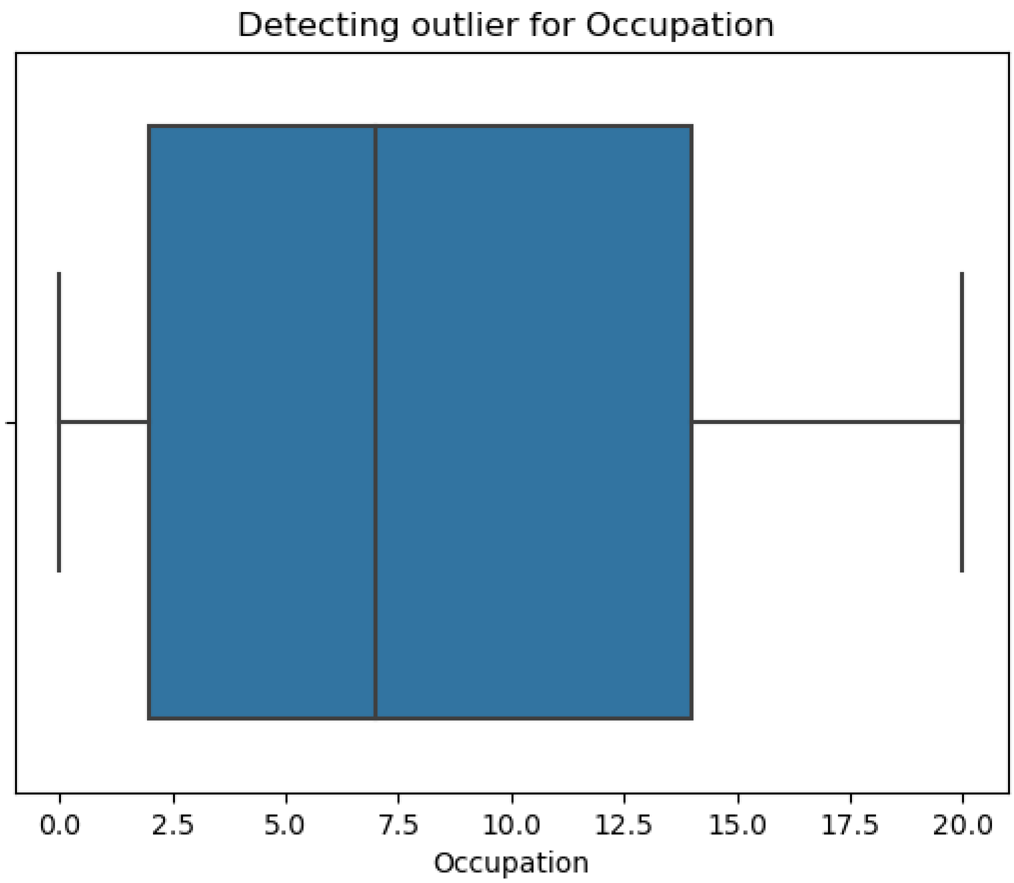
Out[351]:

```
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
```

As you can see there is **No Missing or Null values** in the dataset

In [381…

```
sns.boxplot(data=df,x ='Occupation')
plt.title('Detecting outlier for Occupation ')
plt.show()
print('Median is ',df['Occupation'].median())
print('Mean is ',df['Occupation'].mean())
#print('Difference between mean and median is 2.78')
```

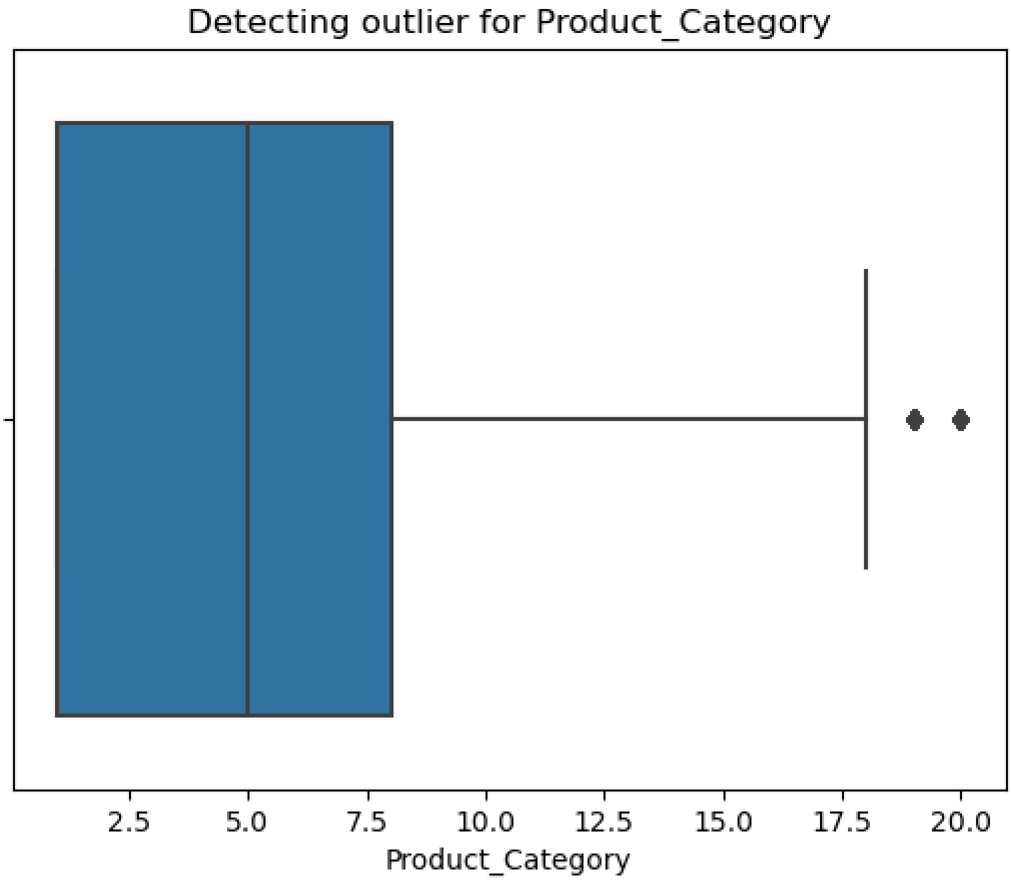



```
Median is  7.0
Mean is  8.076706879876669
```

There are **No Outliers** for **Occupation** column

In [380…

```
sns.boxplot(data=df,x ='Product_Category')
plt.title('Detecting outlier for Product_Category')
plt.show()
print('Median is ',df['Product_Category'].median())
print('Mean is ',df['Product_Category'].mean())
#print('Difference between mean and median is 2.78')
```

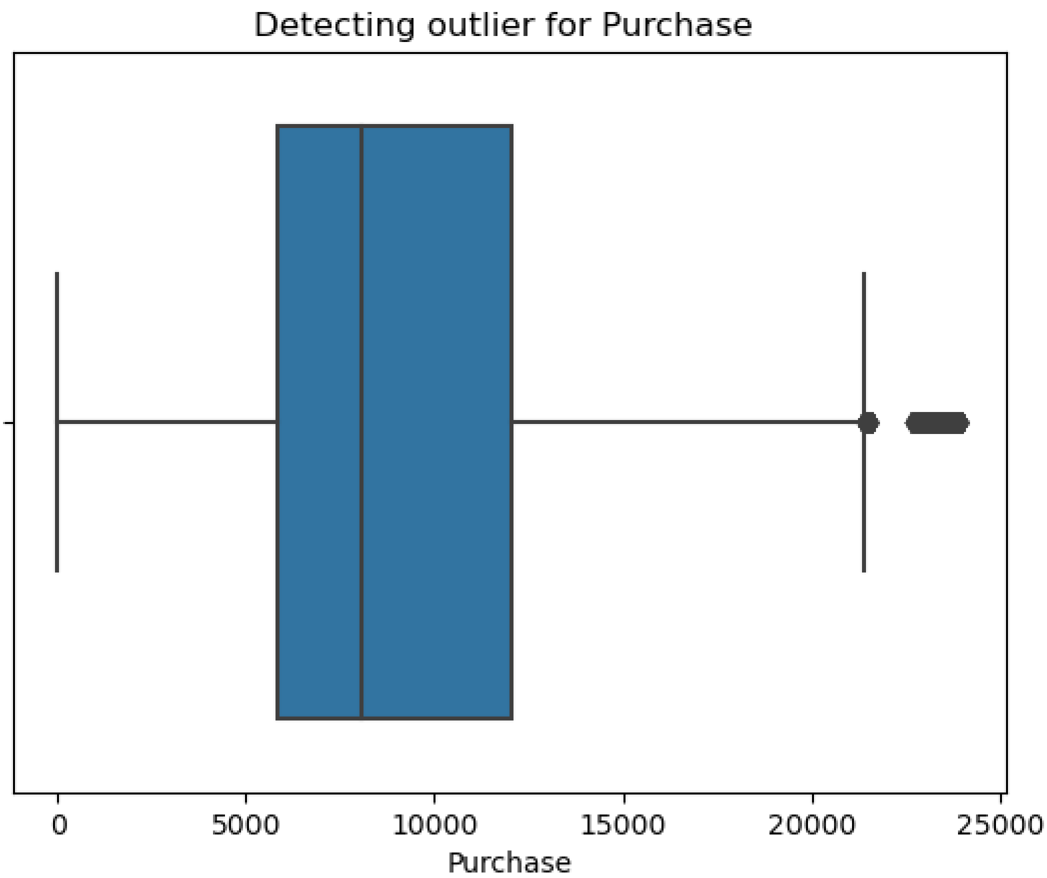



```
Median is  5.0
Mean is  5.404270017525106
```

Outliers present for **Product_Category** column **above 18**

In [388…

```
sns.boxplot(data=df,x ='Purchase')
plt.title('Detecting outlier for Purchase ')
plt.show()
print('Median Purchase value is ',df['Purchase'].median())
print('Mean Purchase Value is ',round(df['Purchase'].mean(),2))
#print('Difference between mean and median is 2.78')
```




```
Median Purchase value is  8047.0
Mean Purchase Value is  9263.97
```

Outliers present for **Purchase** column above **21,000**

Are Women spending more than Men ?

In [399…

```
#Total purchase amount for Male and female
total_purchase_amount = df.groupby('Gender')['Purchase'].sum()
total_purchase_amount
```

Out[399]:

```
Gender
F    1186232642
M    3909580100
Name: Purchase, dtype: int64
```

In [400…

```
#Total number of Male and Female
no_of_users = df.groupby('Gender')['Purchase'].count()
no_of_users
```

Out[400]:

```
Gender
F    135809
M    414259
Name: Purchase, dtype: int64
```

In [398…

```
#Average purchase amount of Male and Female
avg_purchase_amount = total_purchase_amount/no_of_users
avg_purchase_amount
```

Out[398]:

```
Gender
F    8734.565765
M    9437.526040
Name: Purchase, dtype: float64
```

Observation -

- On **Average Women are not spending more than Men**
- **Mean** purchase amount of **women** is **8734.56**
- **Mean** purchase amount of **men** is **9437.52**

In [404…
```
# Creating Sample
sample = df.sample(1000)
```

In [405…
```
sample.head()
```

Out[405]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_C
130581	1002047	P00155642	M	0-17	10	B	
176105	1003280	P00137242	M	26-35	7	B	
306113	1005148	P00022542	M	26-35	20	B	
67349	1004312	P00325242	M	26-35	18	A	
487820	1003217	P00194342	F	36-45	0	A	

In [406…
```
male_sample = sample[sample['Gender'] == 'M']
```

In [407…
```
male_sample.head()
```

Out[407]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_C
130581	1002047	P00155642	M	0-17	10	B	
176105	1003280	P00137242	M	26-35	7	B	
306113	1005148	P00022542	M	26-35	20	B	
67349	1004312	P00325242	M	26-35	18	A	
366605	1002421	P00025442	M	36-45	7	C	

In [440…
```
print("Sample Mean for Male is", male_sample['Purchase'].mean())
```

```
Sample Mean for Male is 9584.459170013386
```

In [408…
```
female_sample = sample[sample['Gender'] == 'F']
```

In [409…
```
female_sample.head()
```

Out[409]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_C
487820	1003217	P00194342	F	36-45	0	A	
419658	1004517	P00106042	F	18-25	1	A	
449706	1003308	P00157942	F	18-25	20	B	
19918	1003136	P00245642	F	26-35	4	C	
73536	1005329	P00193242	F	26-35	5	B	

In [453…

```
print("Sample Mean for Female is", female_sample['Purchase'].mean())
```

```
Sample Mean for Female is 8665.656126482214
```

In [452…

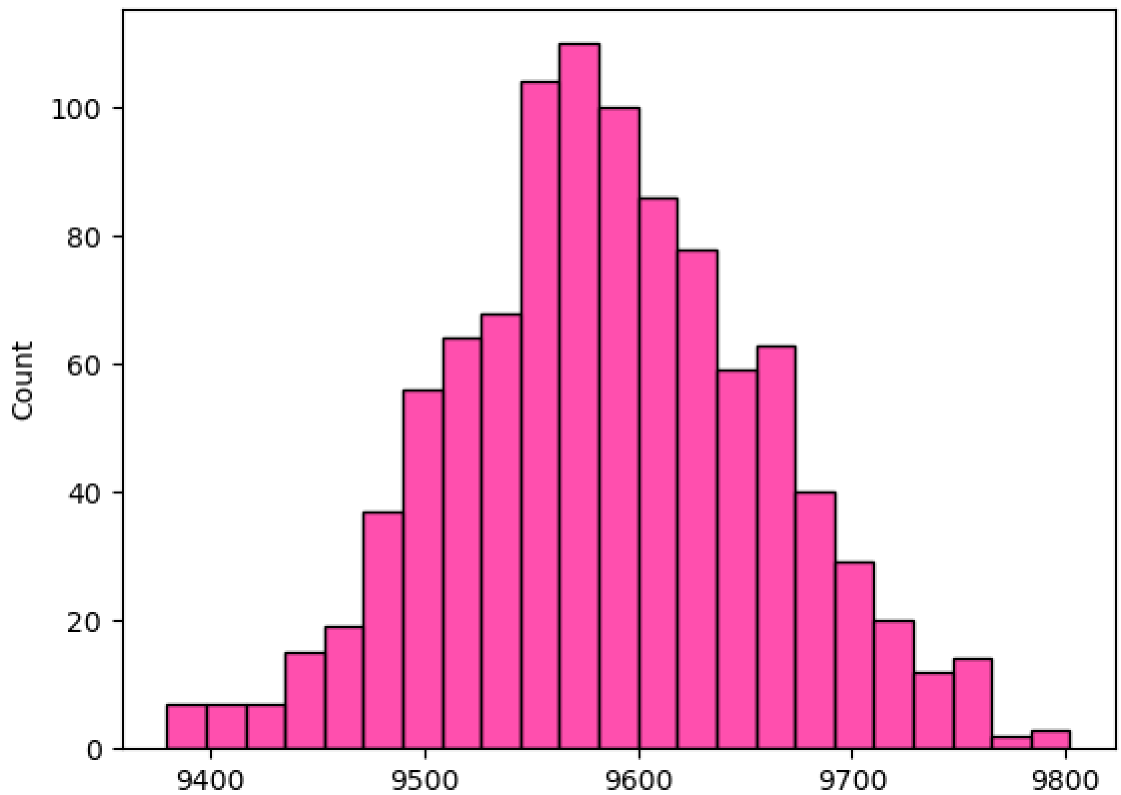
```
male_sample_mean = [male_sample.sample(5000, replace=True)['Purchase'].mean(
male_sample_mean[:10]
```

Out[452]:

```
[9591.8646,
 9559.4326,
 9548.5168,
 9527.42,
 9540.335,
 9642.3344,
 9727.0266,
 9507.8134,
 9649.9464,
 9566.188]
```

In [475…

```
sns.histplot(male_sample_mean,color='deeppink')
plt.show()
```

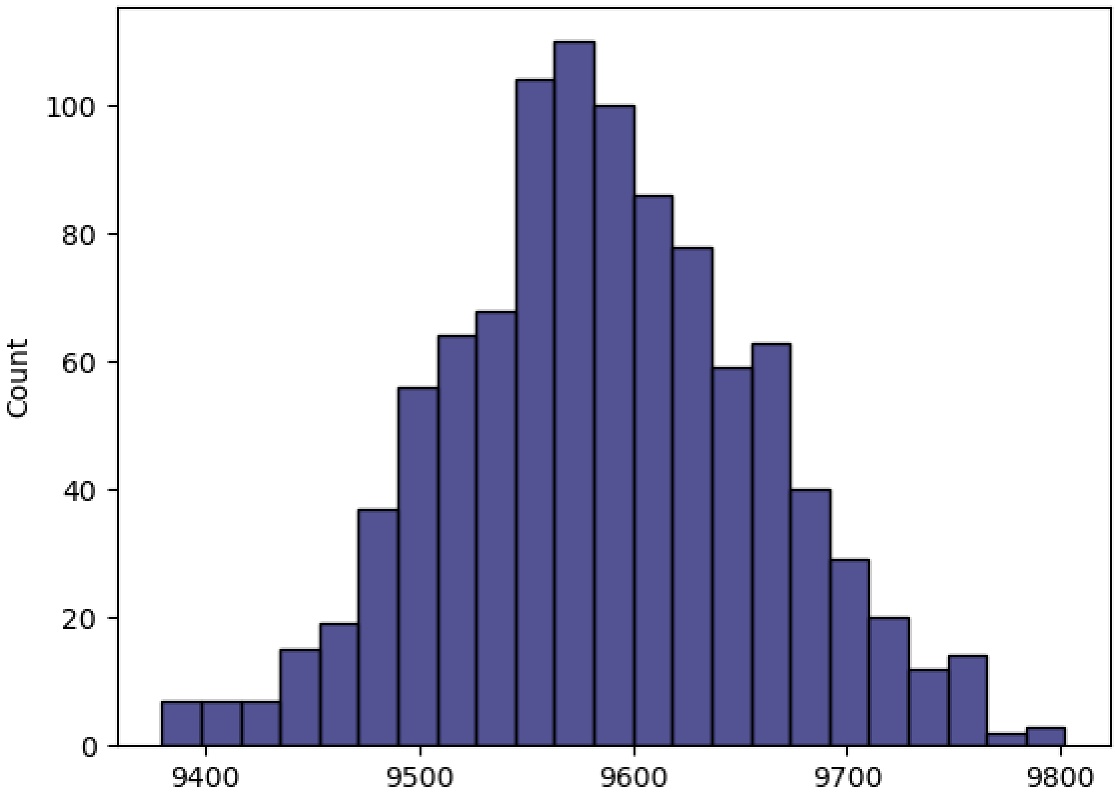


Graph appears to be Gaussian for mean of **male** samples

In [418…
```
female_sample_mean = [female_sample.sample(5000, replace=True)['Purchase'].m
female_sample_mean[:10]
```

Out[418]:
```
[8638.5568,
 8698.5562,
 8688.6294,
 8667.5456,
 8564.9238,
 8689.2942,
 8686.7792,
 8633.5114,
 8729.796,
 8727.3176]
```

In [474…
```
sns.histplot(male_sample_mean,color='midnightblue')
plt.show()
```



Graph follows Gaussian distribution for mean of **female** sample

In [445…
```
np.std(male_sample_mean).round(3)
```

Out[445]:
```
71.172
```

In [446…
```
np.std(female_sample_mean).round(3)
```

Out[446]:
```
69.018
```

CI - 90%

In [659…
```
# Confidence Interval of male = 90%
male_low = np.mean(male_sample_mean) + norm.ppf(0.05) * (np.std(male_sample_
male_high = np.mean(male_sample_mean) + norm.ppf(0.95) * (np.std(male_sample
male_low.round(2), male_high.round(2)
```

Out[659]: (9464.29, 9706.56)

In [660…
```
# Confidence Interval of female = 90%
female_low = np.mean(female_sample_mean) + norm.ppf(0.05) * (np.std(female_s
female_high = np.mean(female_sample_mean) + norm.ppf(0.95) * (np.std(female_
female_low.round(2), female_high.round(2)
```

Out[660]: (8551.08, 8778.13)

In [652…
```
# To check overlapping of Confidence Intervals
male_CI = np.percentile(male_sample_mean, [5, 95])
female_CI = np.percentile(female_sample_mean, [5, 95])
print("90% Confidence Interval for Male sample is : ",male_CI.round(2))
print("90% Confidence Interval for Female sample is : ",female_CI.round(2))
```

```
90% Confidence Interval for Male sample is :  [9466.4  9711.71]
90% Confidence Interval for Female sample is :  [8550.04 8778.51]
```

Observation -

- The confidence interval is **not overlapping** for Male and Female customers

CI - 95%

In [657…
```
# Confidence Interval of male = 95%
male_low = np.mean(male_sample_mean) + norm.ppf(0.025) * (np.std(male_sample
male_high = np.mean(male_sample_mean) + norm.ppf(0.975) * (np.std(male_sampl
male_low.round(2), male_high.round(2)
```

Out[657]: (9441.08, 9729.77)

In [658…
```
# Confidence Interval of female = 95%
female_low = np.mean(female_sample_mean) + norm.ppf(0.025) * (np.std(female_
female_high = np.mean(female_sample_mean) + norm.ppf(0.975) * (np.std(female
female_low.round(2), female_high.round(2)
```

Out[658]: (8529.33, 8799.88)

In [653…
```
# To check overlapping of Confidence Intervals
male_CI = np.percentile(male_sample_mean, [2.5, 97.5])
female_CI = np.percentile(female_sample_mean, [2.5, 97.5])
print("95% Confidence Interval for Male sample is : ",male_CI.round(2))
print("95% Confidence Interval for Female sample is : ",female_CI.round(2))
```

```
95% Confidence Interval for Male sample is :  [9441.16 9731.27]
95% Confidence Interval for Female sample is :  [8528.01 8804.7 ]
```

Observation -

For **95% Confidence Interval** we can conclude that purchase values for Male and Female are **not Overlapping**

CI - 99%

In [655…
```
# Confidence Interval of male = 99%
male_low = np.mean(male_sample_mean) + norm.ppf(0.005) * (np.std(male_sample
male_high = np.mean(male_sample_mean) + norm.ppf(0.995) * (np.std(male_sampl
male_low.round(2), male_high.round(2)
```

Out[655]: (9395.72, 9775.12)

In [656…
```
# Confidence Interval of female = 99%
female_low = np.mean(female_sample_mean) + norm.ppf(0.005) * (np.std(female_
female_high = np.mean(female_sample_mean) + norm.ppf(0.995) * (np.std(female
female_low.round(2), female_high.round(2)
```

Out[656]: (8486.83, 8842.38)

In [661…
```
# To check overlapping of Confidence Intervals
male_CI = np.percentile(male_sample_mean, [0.5, 99.5])
female_CI = np.percentile(female_sample_mean, [0.5, 99.5])
print("99% Confidence Interval for Male sample is : ",male_CI.round(2))
print("99% Confidence Interval for Female sample is : ",female_CI.round(2))
```

```
99% Confidence Interval for Male sample is :  [9386.78 9760.65]
99% Confidence Interval for Female sample is :  [8503.97 8847.24]
```

For **99% Confidence Interval**, the purchases of Male and Female are **not overlapping**

For Males -

- Population Mean for Male : 9437.52
- Mean of Sample mean for Male : 9584.45
- 90% CI for mean expense for Male users is (9464.287, 9706.56)
- 95% CI for mean expense for Male users is (9443.842, 9722.832)
- 99% CI for mean expense for Male users is (9400.009, 9766.664)

For Females -

- Population Mean for Female : 8734.56
- Mean of Sample mean for Female : 8665.65
- 90% CI for mean expense for Female users is (8551.082, 8778.129)
- 95% CI for mean expense for Female users is (8529.334, 8799.877)
- 99% CI for mean expense for Female users is (8486.828, 8842.383)

Purchase Range for Male and Female are not overlapping in any case

Married vs Unmarried -

In [455…
```
# Creating Sample
sample = df.sample(1000)
```

In [456…
```
sample.head()
```

Out[456]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_C
274789	1000333	P00219242	M	36-45	2	A	
141019	1003768	P00101942	M	26-35	4	B	
470188	1000436	P00199442	M	18-25	4	C	
243762	1001579	P00282042	M	26-35	0	A	
272383	1005978	P00134042	M	36-45	1	B	

In [457…

```
unmarried_sample = sample[sample['Marital_Status'] == 0]
```

In [458…

```
unmarried_sample.head()
```

Out[458]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_C
141019	1003768	P00101942	M	26-35	4	B	
470188	1000436	P00199442	M	18-25	4	C	
243762	1001579	P00282042	M	26-35	0	A	
272383	1005978	P00134042	M	36-45	1	B	
347030	1005448	P00086342	M	46-50	19	A	

In [459…

```
print("Sample Mean for Unmarried people is", unmarried_sample['Purchase'].me
```

```
Sample Mean for Unmarried people is 9021.34219269103
```

In [491…

```
df[df['Marital_Status']==0]['Purchase'].mean()
```

Out[491]:

```
9265.907618921507
```

In [460…

```
married_sample = sample[sample['Marital_Status'] == 1]
```

In [461…

```
married_sample.head()
```

Out[461]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_C
274789	1000333	P00219242	M	36-45	2	A	
194100	1005969	P00033542	M	46-50	13	C	
48297	1001407	P00109242	F	36-45	15	A	
281018	1001301	P00046142	F	26-35	2	C	
443239	1002158	P00057642	M	26-35	12	A	

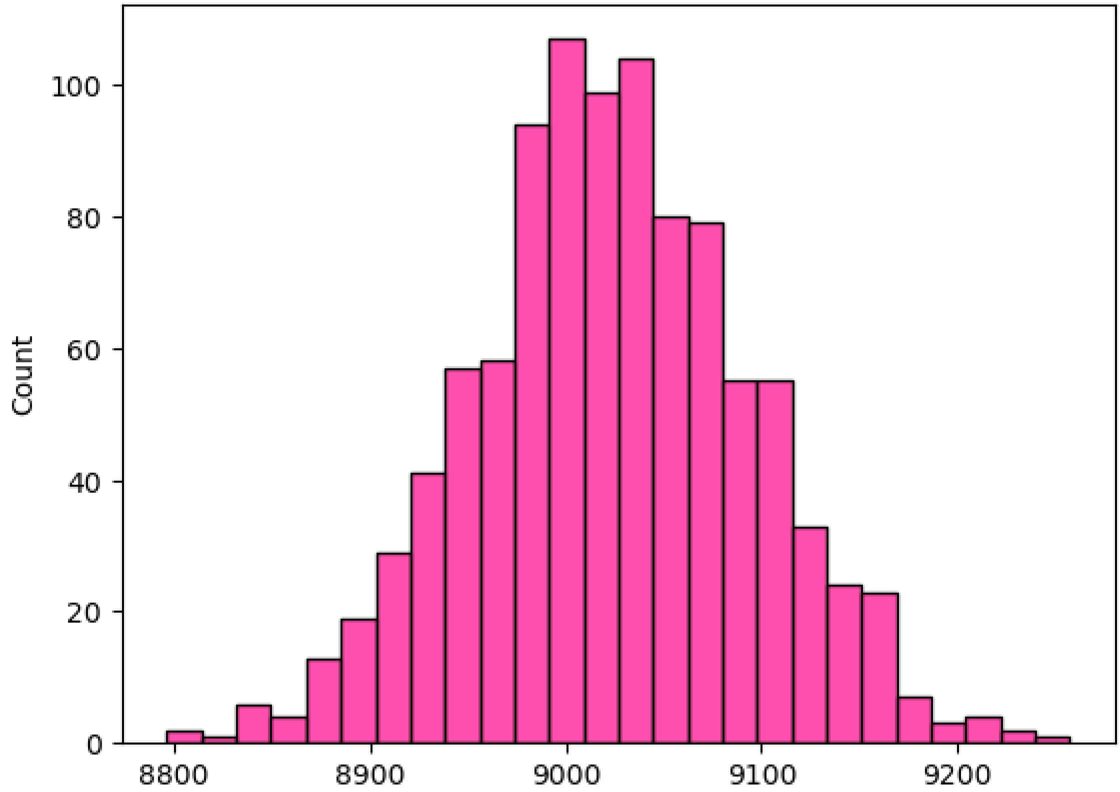
In [463…
```
print("Sample Mean for Married people is", married_sample['Purchase'].mean()
```
```
Sample Mean for Married people is 9183.27135678392
```

In [505…
```
df[df['Marital_Status']== 1]['Purchase'].mean()
```
Out[505]:
```
9261.174574082374
```

In [503…
```
married_sample.shape[0]
```
Out[503]:
```
398
```

In [464…
```
unmarried_sample_mean = [unmarried_sample.sample(5000, replace=True)['Purcha
unmarried_sample_mean[:10]
```
Out[464]:
```
[9167.076,
 9067.4868,
 9001.4112,
 9063.0636,
 9004.5784,
 9171.4416,
 9016.984,
 9107.7478,
 9110.8352,
 9070.124]
```

In [473…
```
sns.histplot(unmarried_sample_mean,color='deeppink')
plt.show()
```

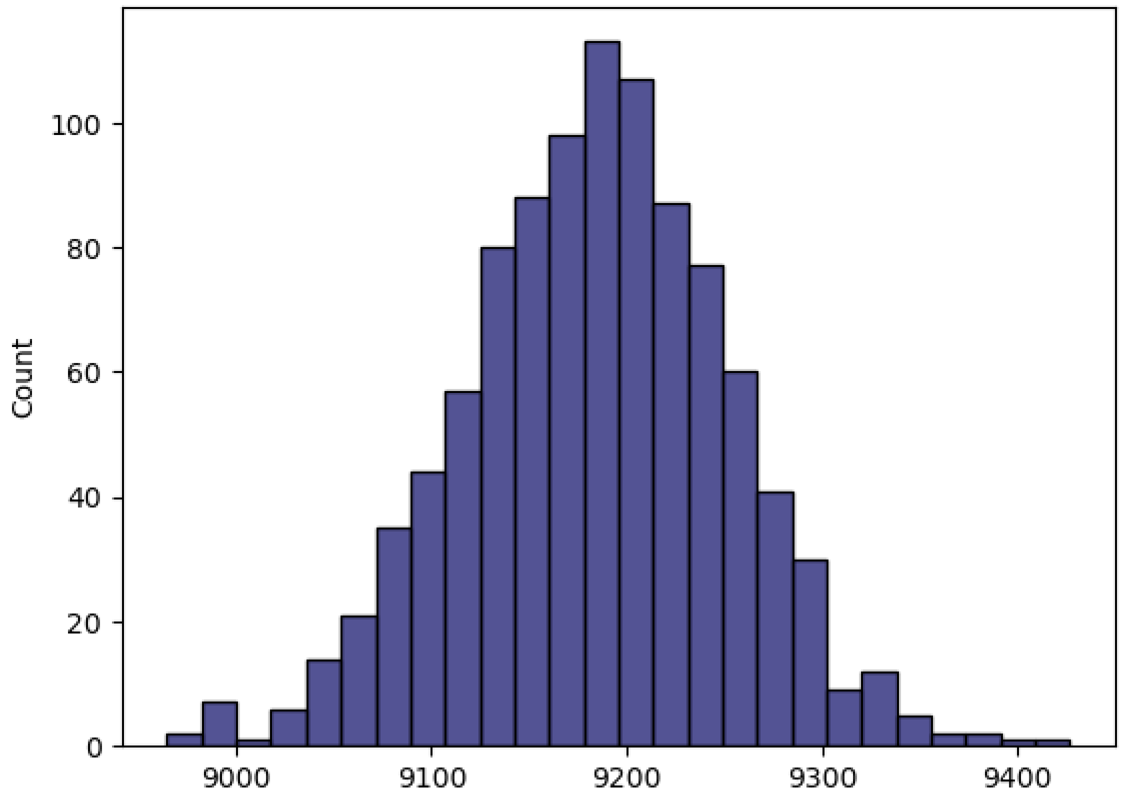


Graph appears to be Gaussian for mean of **Unmarried** samples

In [466…
```
married_sample_mean = [married_sample.sample(5000, replace=True)['Purchase']
married_sample_mean[:10]
```

Out[466]:
```
[9203.1052,
 9199.2244,
 9126.8482,
 9154.4452,
 9162.2126,
 9128.4812,
 9250.0428,
 9075.2306,
 9149.6738,
 9079.3828]
```

In [472…
```
sns.histplot(married_sample_mean,color='midnightblue')
plt.show()
```



Graph follows Gaussian distribution for mean of **female** sample

In [476…
```
np.std(unmarried_sample_mean).round(3)
```
Out[476]:
```
70.56
```

In [478…
```
np.std(married_sample_mean).round(3)
```
Out[478]:
```
67.852
```

CI - 90%

In [662…
```
# Confidence Interval of male = 90%
unmarried_low = np.mean(unmarried_sample_mean) + norm.ppf(0.05) * (np.std(un
unmarried_high = np.mean(unmarried_sample_mean) + norm.ppf(0.95) * (np.std(u
unmarried_low.round(2), unmarried_high.round(2)
```
Out[662]:
```
(8906.69, 9138.81)
```

In [663…
```
# Confidence Interval of female = 90%
married_low = np.mean(married_sample_mean) + norm.ppf(0.05) * (np.std(marrie
married_high = np.mean(married_sample_mean) + norm.ppf(0.95) * (np.std(marri
married_low.round(2), married_high.round(2)
```
Out[663]:
```
(9072.0, 9295.21)
```

In [665…
```
# To check overlapping of Confidence Intervals
unmarried_CI = np.percentile(unmarried_sample_mean, [5, 95])
married_CI = np.percentile(married_sample_mean, [5, 95])
print("90% Confidence Interval for Unmarried sample is :",unmarried_CI.round
print("90% Confidence Interval for Married sample is :",married_CI.round(2))
```
```
90% Confidence Interval for Unmarried sample is : [8906.43 9141.77]
90% Confidence Interval for Married sample is : [9071.35 9292.77]
```

Observation -

- For **90% confidence interval**, the mean purchase value seems to be **Overlapping** for **Married and Unmarried customers**

CI - 95%

In [666…

```
# Confidence Interval of male = 95%
unmarried_low = np.mean(unmarried_sample_mean) + norm.ppf(0.025) * (np.std(u
unmarried_high = np.mean(unmarried_sample_mean) + norm.ppf(0.975) * (np.std(
unmarried_low.round(2), unmarried_high.round(2)
```

Out[666]:

```
(8884.45, 9161.04)
```

In [667…

```
# Confidence Interval of female = 95%
married_low = np.mean(married_sample_mean) + norm.ppf(0.025) * (np.std(marri
married_high = np.mean(married_sample_mean) + norm.ppf(0.975) * (np.std(marr
married_low.round(2), married_high.round(2)
```

Out[667]:

```
(9050.62, 9316.59)
```

In [668…

```
# To check overlapping of Confidence Intervals
unmarried_CI = np.percentile(unmarried_sample_mean, [2.5, 97.5])
married_CI = np.percentile(married_sample_mean, [2.5, 97.5])
print("95% Confidence Interval for Unmarried sample is :",unmarried_CI.round
print("95% Confidence Interval for Married sample is :",married_CI.round(2))
```

```
90% Confidence Interval for Unmarried sample is : [8884.67 9159.88]
90% Confidence Interval for Married sample is : [9045.56 9314.87]
```

For **95% Confidence Interval**, we can conclude that purchase values for **Married and Unmarried** customers are **Overlapping**

CI - 99%

In [669…

```
# Confidence Interval of male = 99%
unmarried_low = np.mean(unmarried_sample_mean) + norm.ppf(0.005) * (np.std(u
unmarried_high = np.mean(unmarried_sample_mean) + norm.ppf(0.995) * (np.std(
unmarried_low.round(2), unmarried_high.round(2)
```

Out[669]:

```
(8841.0, 9204.5)
```

In [670…

```
# Confidence Interval of female = 99%
married_low = np.mean(married_sample_mean) + norm.ppf(0.005) * (np.std(marri
married_high = np.mean(married_sample_mean) + norm.ppf(0.995) * (np.std(marr
married_low.round(2), married_high.round(2)
```

Out[670]:

```
(9008.83, 9358.38)
```

In [671…

```
# To check overlapping of Confidence Intervals
unmarried_CI = np.percentile(unmarried_sample_mean, [0.5, 99.5])
married_CI = np.percentile(married_sample_mean, [0.5, 99.5])
print("99% Confidence Interval for Unmarried sample is :",unmarried_CI.round
print("99% Confidence Interval for Married sample is :",married_CI.round(2))
```

```
99% Confidence Interval for Unmarried sample is : [8846.81 9212.7 ]
99% Confidence Interval for Married sample is : [8992.68 9361.96]
```

For **99% Confidence Interval**, the purchase values of Married and Unmarried customers are **Overlapping**

For Unmarried Customers -

- Population Mean for Unmarried : 9265.90
- Mean of Sample mean for Unmarried : 9021.34
- 90% CI for mean expense for Unmarried users is (8906.433, 9141.775)
- 95% CI for mean expense for Unmarried users is (8884.666, 9159.88)
- 99% CI for mean expense for Unmarried users is (8846.811, 9212.698)

For Married Customers -

- Population Mean for Married : 9261.17
- Mean of Sample mean for Married : 9183.27
- 90% CI for mean expense for Married users is (9071.35 , 9292.775)
- 95% CI for mean expense for Married users is (9045.564, 9314.873)
- 99% CI for mean expense for Married users is (8992.68 , 9361.957)

In every case the Purchase Range seems to be Overlapping for Married and Unmarried customers

Age -

In [506…
```
# Creating Sample
sample = df.sample(1000)
```

In [507…
```
sample.head()
```

Out[507]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_(
274512	1000299	P00175042	M	26-35	12	C	
284917	1001868	P00362842	M	51-55	11	C	
449440	1003272	P00116742	M	36-45	0	B	
421294	1004809	P00230042	F	51-55	4	C	
427361	1005795	P00049442	M	26-35	1	A	

In [508…
```
age_0to17 = sample[sample['Age'] == '0-17']
```

In [509…
```
age_0to17.head()
```

Out[509]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_C
148840	1004965	P00112642	M	0-17	10	C	
384991	1005255	P00157342	M	0-17	10	A	
545702	1006006	P00187942	F	0-17	0	C	
476821	1001434	P00072842	F	0-17	10	A	
164133	1001353	P00248442	M	0-17	10	C	

In [510…
```
age_18to25 = sample[sample['Age'] == '18-25']
```

In [514…
```
age_26to35 = sample[sample['Age'] == '26-35']
```

In [537…
```
age_36to45 = sample[sample['Age'] == '36-45']
```

In [523…
```
age_46to50 = sample[sample['Age'] == '46-50']
```

In [522…
```
age_51to55 = sample[sample['Age'] == '51-55']
```

In [521…
```
age_55plus = sample[sample['Age'] == '55+']
```

In [593…
```
# Creating sample of means
age_0to17_mean = [age_0to17.sample(5000, replace=True)['Purchase'].mean() fo
age_0to17_mean[:10]
```

Out[593]:
```
[8733.6342,
 8626.6818,
 8707.2914,
 8725.9306,
 8831.3736,
 8763.504,
 8661.3868,
 8611.4028,
 8691.4272,
 8743.0144]
```

In [594…
```
age_18to25_mean = [age_18to25.sample(5000, replace=True)['Purchase'].mean()
```

In [529…
```
age_26to35_mean = [age_26to35.sample(5000, replace=True)['Purchase'].mean()
```

In [538…
```
age_36to45_mean = [age_36to45.sample(5000, replace=True)['Purchase'].mean()
```

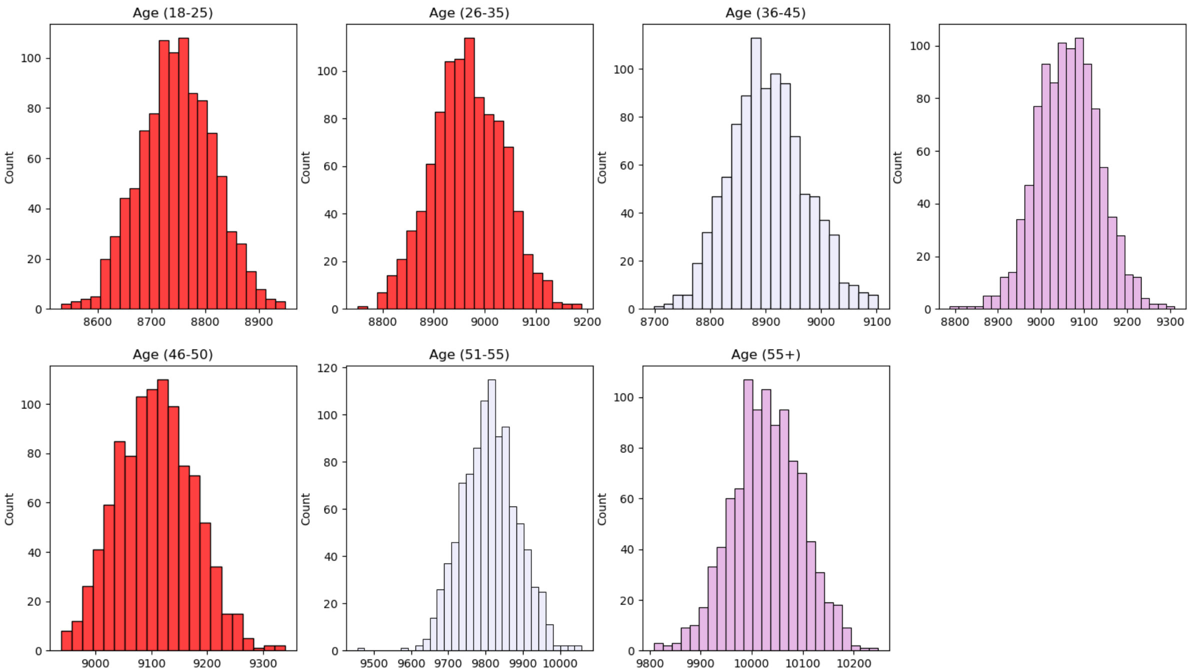
In [531…
```
age_46to50_mean = [age_46to50.sample(5000, replace=True)['Purchase'].mean()
```

In [532…
```
age_51to55_mean = [age_51to55.sample(5000, replace=True)['Purchase'].mean()
```

In [533…
```
age_55plus_mean = [age_55plus.sample(5000, replace=True)['Purchase'].mean()
```

In [597…
```
fig, axis = plt.subplots(nrows=2, ncols=4, figsize=(18, 10))

sns.histplot(age_0to17_mean,color='red',ax=axis[0,0])
```

```
axis[0,0].set_title(label='Age (0-17)')
sns.histplot(age_18to25_mean,color='red',ax=axis[0,1])
axis[0,0].set_title(label='Age (18-25)')
sns.histplot(age_26to35_mean,color='lavender',ax=axis[0,2])
axis[0,1].set_title(label='Age (26-35)')
sns.histplot(age_36to45_mean,color='plum',ax=axis[0,3])
axis[0,2].set_title(label='Age (36-45)')
sns.histplot(age_46to50_mean,color='red',ax=axis[1,0])
axis[1,0].set_title(label='Age (46-50)')
sns.histplot(age_51to55_mean,color='lavender',ax=axis[1,1])
axis[1,1].set_title(label='Age (51-55)')
sns.histplot(age_55plus_mean,color='plum',ax=axis[1,2])
axis[1,2].set_title(label='Age (55+)')
axis[1,3].set_axis_off()
plt.show()
```



Graph appears to be following Gaussian distribution for each of the sample mean of **Age Groups**

CI - 90%

In [672…
```
# Confidence Interval of Age (18-25) = 90%
age_0to17_low = np.mean(age_0to17_mean) + norm.ppf(0.05) * (np.std(age_0to17
age_0to17_high = np.mean(age_0to17_mean) + norm.ppf(0.95) * (np.std(age_0to1
age_0to17_low.round(2), age_0to17_high.round(2)
```

Out[672]: (8633.57, 8860.22)

In [673…
```
# Confidence Interval of Age (18-25) = 90%
age_18to25_low = np.mean(age_18to25_mean) + norm.ppf(0.05) * (np.std(age_18t
age_18to25_high = np.mean(age_18to25_mean) + norm.ppf(0.95) * (np.std(age_18
age_18to25_low.round(2), age_18to25_high.round(2)
```

Out[673]: (8852.51, 9081.78)

In [674…
```
# Confidence Interval of Age (26-35) = 90%
age_26to35_low = np.mean(age_26to35_mean) + norm.ppf(0.05) * (np.std(age_26t
age_26to35_high = np.mean(age_26to35_mean) + norm.ppf(0.95) * (np.std(age_26
age_26to35_low.round(2), age_26to35_high.round(2)
```

Out[674]: (8791.14, 9017.36)

In [675…
```
# Confidence Interval of Age (36-45) = 90%
age_36to45_low = np.mean(age_36to45_mean) + norm.ppf(0.05) * (np.std(age_36t
age_36to45_high = np.mean(age_36to45_mean) + norm.ppf(0.95) * (np.std(age_36
age_36to45_low.round(2), age_36to45_high.round(2)
```

Out[675]: (8944.26, 9182.45)

In [605…
```
# Confidence Interval of Age (46-50) = 90%
age_46to50_low = np.mean(age_46to50_mean) + norm.ppf(0.05) * (np.std(age_46t
age_46to50_high = np.mean(age_46to50_mean) + norm.ppf(0.95) * (np.std(age_46
age_46to50_low.round(3), age_46to50_high.round(3)
```

Out[605]: (8996.209, 9218.77)

In [626…
```
# Confidence Interval of Age (51-55) = 90%
age_51to55_low = np.mean(age_51to55_mean) + norm.ppf(0.05) * (np.std(age_51t
age_51to55_high = np.mean(age_51to55_mean) + norm.ppf(0.95) * (np.std(age_51
age_51to55_low.round(3), age_51to55_high.round(3)
```

Out[626]: (9688.696, 9933.23)

In [676…
```
# Confidence Interval of Age (55+) = 90%
age_55plus_low = np.mean(age_55plus_mean) + norm.ppf(0.05) * (np.std(age_55p
age_55plus_high = np.mean(age_55plus_mean) + norm.ppf(0.95) * (np.std(age_55
age_55plus_low.round(2), age_55plus_high.round(2)
```

Out[676]: (9916.72, 10141.47)

In [629…
```
# To check overlapping of Confidence Intervals
age_0to17_CI = np.percentile(age_0to17_mean, [5, 95])
age_18to25_CI = np.percentile(age_18to25_mean, [5, 95])
age_26to35_CI = np.percentile(age_26to35_mean, [5, 95])
age_36to45_CI = np.percentile(age_36to45_mean, [5, 95])
age_46to50_CI = np.percentile(age_46to50_mean, [5, 95])
age_51to55_CI = np.percentile(age_51to55_mean, [5, 95])
age_55plus_CI = np.percentile(age_55plus_mean, [5, 95])

print("90% Confidence Interval for Age (0-17) :", age_0to17_CI.round(2))
print("90% Confidence Interval for Age (18-25) :", age_18to25_CI.round(2))
print("90% Confidence Interval for Age (26-35) :", age_26to35_CI.round(2))
print("90% Confidence Interval for Age (36-45) :", age_36to45_CI.round(2))
print("90% Confidence Interval for Age (46-50) :", age_46to50_CI.round(2))
print("90% Confidence Interval for Age (51-55) :", age_51to55_CI.round(2))
print("90% Confidence Interval for Age (55+) :", age_55plus_CI.round(2))
```

```
90% Confidence Interval for Age (0-17) : [8630.96 8863.76]
90% Confidence Interval for Age (18-25) : [8850.28 9082.01]
90% Confidence Interval for Age (26-35) : [8797.15 9017.99]
90% Confidence Interval for Age (36-45) : [8949.33 9180.95]
90% Confidence Interval for Age (46-50) : [8999.55 9218.3 ]
90% Confidence Interval for Age (51-55) : [9690.27 9933.24]
90% Confidence Interval for Age (55+) : [ 9918.06 10141.43]
```

Observation -

- For **90% confidence interval**, the mean purchase value seems to be **Overlapping** for **different Age groups**

Age groups (0-17), (18-25), (26-35), (36-45), (46-50) are not overlapping with age group (51-55) and (55+)

CI - 95%

In [677…
```
# Confidence Interval of Age (18-25) = 95%
age_0to17_low = np.mean(age_0to17_mean) + norm.ppf(0.025) * (np.std(age_0to1
age_0to17_high = np.mean(age_0to17_mean) + norm.ppf(0.975) * (np.std(age_0to
age_0to17_low.round(2), age_0to17_high.round(2)
```
Out[677]:
```
(8611.86, 8881.93)
```

In [678…
```
# Confidence Interval of Age (18-25) = 95%
age_18to25_low = np.mean(age_18to25_mean) + norm.ppf(0.025) * (np.std(age_18
age_18to25_high = np.mean(age_18to25_mean) + norm.ppf(0.975) * (np.std(age_1
age_18to25_low.round(2), age_18to25_high.round(2)
```
Out[678]:
```
(8830.55, 9103.74)
```

In [679…
```
# Confidence Interval of Age (26-35) = 95%
age_26to35_low = np.mean(age_26to35_mean) + norm.ppf(0.025) * (np.std(age_26
age_26to35_high = np.mean(age_26to35_mean) + norm.ppf(0.975) * (np.std(age_2
age_26to35_low.round(2), age_26to35_high.round(2)
```
Out[679]:
```
(8769.47, 9039.03)
```

In [680…
```
# Confidence Interval of Age (36-45) = 95%
age_36to45_low = np.mean(age_36to45_mean) + norm.ppf(0.025) * (np.std(age_36
age_36to45_high = np.mean(age_36to45_mean) + norm.ppf(0.975) * (np.std(age_3
age_36to45_low.round(2), age_36to45_high.round(2)
```
Out[680]:
```
(8921.45, 9205.27)
```

In [681…
```
# Confidence Interval of Age (46-50) = 95%
age_46to50_low = np.mean(age_46to50_mean) + norm.ppf(0.025) * (np.std(age_46
age_46to50_high = np.mean(age_46to50_mean) + norm.ppf(0.975) * (np.std(age_4
age_46to50_low.round(2), age_46to50_high.round(2)
```
Out[681]:
```
(8974.89, 9240.09)
```

In [685…
```
# Confidence Interval of Age (51-55) = 95%
age_51to55_low = np.mean(age_51to55_mean) + norm.ppf(0.025) * (np.std(age_51
age_51to55_high = np.mean(age_51to55_mean) + norm.ppf(0.975) * (np.std(age_5
age_51to55_low.round(2), age_51to55_high.round(2)
```
Out[685]:
```
(9665.27, 9956.65)
```

In [686…
```
# Confidence Interval of Age (55+) = 95%
age_55plus_low = np.mean(age_55plus_mean) + norm.ppf(0.025) * (np.std(age_55
age_55plus_high = np.mean(age_55plus_mean) + norm.ppf(0.975) * (np.std(age_5
age_55plus_low.round(2), age_55plus_high.round(2)
```
Out[686]:
```
(9895.2, 10162.99)
```

In [684…
```
# To check overlapping of Confidence Intervals
age_0to17_CI = np.percentile(age_0to17_mean, [2.5, 97.5])
age_18to25_CI = np.percentile(age_18to25_mean, [2.5, 97.5])
age_26to35_CI = np.percentile(age_26to35_mean, [2.5, 97.5])
age_36to45_CI = np.percentile(age_36to45_mean, [2.5, 97.5])
```

```
age_46to50_CI = np.percentile(age_46to50_mean, [2.5, 97.5])
age_51to55_CI = np.percentile(age_51to55_mean, [2.5, 97.5])
age_55plus_CI = np.percentile(age_55plus_mean, [2.5, 97.5])

print("95% Confidence Interval for Age (0-17) :", age_0to17_CI.round(2))
print("95% Confidence Interval for Age (18-25) :", age_18to25_CI.round(2))
print("95% Confidence Interval for Age (26-35) :", age_26to35_CI.round(2))
print("95% Confidence Interval for Age (36-45) :", age_36to45_CI.round(2))
print("95% Confidence Interval for Age (46-50) :", age_46to50_CI.round(2))
print("95% Confidence Interval for Age (51-55) :", age_51to55_CI.round(2))
print("95% Confidence Interval for Age (55+) :", age_55plus_CI.round(2))
```

```
95% Confidence Interval for Age (0-17) : [8613.87 8880.14]
95% Confidence Interval for Age (18-25) : [8828.14 9102.82]
95% Confidence Interval for Age (26-35) : [8773.52 9047.12]
95% Confidence Interval for Age (36-45) : [8922.54 9209.91]
95% Confidence Interval for Age (46-50) : [8980.12 9238.56]
95% Confidence Interval for Age (51-55) : [9672.01 9955.99]
95% Confidence Interval for Age (55+) : [ 9889.55 10162.4 ]
```

For **95% Confidence Interval**, we can conclude that purchase values for **different Age Groups** are **Overlapping**

Age groups (0-17), (18-25), (26-35), (36-45), (46-50) are not overlapping with age group (51-55) and (55+)

CI - 99%

In [687…

```
# Confidence Interval of Age (18-25) = 99%
age_0to17_low = np.mean(age_0to17_mean) + norm.ppf(0.005) * (np.std(age_0to1
age_0to17_high = np.mean(age_0to17_mean) + norm.ppf(0.995) * (np.std(age_0tc
age_0to17_low.round(2), age_0to17_high.round(2)
```

Out[687]: `(8569.43, 8924.36)`

In [688…

```
# Confidence Interval of Age (18-25) = 99%
age_18to25_low = np.mean(age_18to25_mean) + norm.ppf(0.005) * (np.std(age_18
age_18to25_high = np.mean(age_18to25_mean) + norm.ppf(0.995) * (np.std(age_1
age_18to25_low.round(2), age_18to25_high.round(2)
```

Out[688]: `(8787.63, 9146.66)`

In [689…

```
# Confidence Interval of Age (26-35) = 99%
age_26to35_low = np.mean(age_26to35_mean) + norm.ppf(0.005) * (np.std(age_26
age_26to35_high = np.mean(age_26to35_mean) + norm.ppf(0.995) * (np.std(age_2
age_26to35_low.round(2), age_26to35_high.round(2)
```

Out[689]: `(8727.12, 9081.38)`

In [690…

```
# Confidence Interval of Age (36-45) = 99%
age_36to45_low = np.mean(age_36to45_mean) + norm.ppf(0.005) * (np.std(age_36
age_36to45_high = np.mean(age_36to45_mean) + norm.ppf(0.995) * (np.std(age_3
age_36to45_low.round(2), age_36to45_high.round(2)
```

Out[690]: `(8876.85, 9249.86)`

In [691…

```
# Confidence Interval of Age (46-50) = 99%
age_46to50_low = np.mean(age_46to50_mean) + norm.ppf(0.005) * (np.std(age_46
age_46to50_high = np.mean(age_46to50_mean) + norm.ppf(0.995) * (np.std(age_4
age_46to50_low.round(2), age_46to50_high.round(2)
```

Out[691]: (8933.22, 9281.75)

In [692…
```
# Confidence Interval of Age (51-55) = 99%
age_51to55_low = np.mean(age_51to55_mean) + norm.ppf(0.005) * (np.std(age_51
age_51to55_high = np.mean(age_51to55_mean) + norm.ppf(0.995) * (np.std(age_5
age_51to55_low.round(2), age_51to55_high.round(2)
```

Out[692]: (9619.49, 10002.43)

In [693…
```
# Confidence Interval of Age (55+) = 99%
age_55plus_low = np.mean(age_55plus_mean) + norm.ppf(0.005) * (np.std(age_55
age_55plus_high = np.mean(age_55plus_mean) + norm.ppf(0.995) * (np.std(age_5
age_55plus_low.round(2), age_55plus_high.round(2)
```

Out[693]: (9853.12, 10205.07)

In [650…
```
# To check overlapping of Confidence Intervals
age_0to17_CI = np.percentile(age_0to17_mean, [0.5, 99.5])
age_18to25_CI = np.percentile(age_18to25_mean, [0.5, 99.5])
age_26to35_CI = np.percentile(age_26to35_mean, [0.5, 99.5])
age_36to45_CI = np.percentile(age_36to45_mean, [0.5, 99.5])
age_46to50_CI = np.percentile(age_46to50_mean, [0.5, 99.5])
age_51to55_CI = np.percentile(age_51to55_mean, [0.5, 99.5])
age_55plus_CI = np.percentile(age_55plus_mean, [0.5, 99.5])

print("99% Confidence Interval for Age (0-17) :", age_0to17_CI.round(2))
print("99% Confidence Interval for Age (18-25) :", age_18to25_CI.round(2))
print("99% Confidence Interval for Age (26-35) :", age_26to35_CI.round(2))
print("99% Confidence Interval for Age (36-45) :", age_36to45_CI.round(2))
print("99% Confidence Interval for Age (46-50) :", age_46to50_CI.round(2))
print("99% Confidence Interval for Age (51-55) :", age_51to55_CI.round(2))
print("99% Confidence Interval for Age (55+) :", age_55plus_CI.round(2))
```

```
99% Confidence Interval for Age (0-17) : [8579.6  8920.18]
99% Confidence Interval for Age (18-25) : [8794.54 9139.18]
99% Confidence Interval for Age (26-35) : [8743.42 9085.37]
99% Confidence Interval for Age (36-45) : [8878.67 9248.9 ]
99% Confidence Interval for Age (46-50) : [8952.5  9278.36]
99% Confidence Interval for Age (51-55) : [ 9635.42 10000.37]
99% Confidence Interval for Age (55+) : [ 9849.41 10185.52]
```

For **99% Confidence Interval**, the purchase values of **different Age Groups** seems to be **Overlapping**

Observations -

- Age groups (0-17), (18-25), (26-35), (36-45), (46-50) are not overlapping with age group (51-55) and (55+)
- Age group (0-17) has the least purchasing range of (8579.6, 8920.18)

Inferences -

- Majority of customers are Males.**Male : 75%, Female : 25%**.
- **Women do not spend more than Men on Black Friday**

- Almost **70% customers have Age in the range of (18-45)** [(26-35) - 40%, (36-45) - 20%, (18-25) - 18%]
- Most of the customers come from **Occupation category 0 and 4 (13% each)**
- **City B** is home to majority of customers **(42%)**, followed by **City C (31%)** and **City A (27%)**
- Most of the customers have been **living in their current city for 1 year (35%)**, followed by **2 years(19%) and 3 years(17%)**
- Out of total customers, **59% of the customers are not married**. And **41% are married**
- **Products from Categories (1,5,8) account for around 74% of the Sales** (Category 5 - 27%, Category 1 - 26%, Category 8 - 21%)
- Using Central Limit Theorem and Confidence Interval on the above data, we can see 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, **most of the confidence interval range was Overlapping** while a **few were Not Overlapping**

Recommendations -

- **Female customers** only account for **25% of the Customer base**. **Walmart should promote more Female-centric products and conduct campaigns to increase the participation of Female customers**.
- **Sales to people above Age of 46 is very less**. Company should give some compliments/benifits for them, like add **5-10 extra billing counters exclusively for Senior Citizens**.
- To **improve participation from City A**, Walmart can **give more Offers and Discounts** for its customers in that City.
- Company should **introduce some Loyalty Program** for its customers who have been **living in their current city for more than 2 years** to increase sales from that category.
- **Married people purchase less than Unmarried people**. Company should **conduct some games and contests for married customers to increase their engagement** and thereby leading to increase in Sales
- Products from category 1,5 and 8 can be stocked more in the inventory as they account for around 74% of the sales.

- Male customers account for 75% of the sales. Company should try to retain this customer base by giving them points through loyalty programs