walmart-case-study

May 10, 2024

1 Walmart Business Case

1.1 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).

1.2 Dataset

The company collected the transactional data of customers who purchased products from the Walmart Stores during Black Friday. The dataset has the following features: Dataset link: Walmart_data.csv

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

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

[ ]:
```
data = pd.read_csv('/content/drive/MyDrive/Business case datasets/walmart_data.
↪csv')
```

[ ]:
```
df1 = data.copy()
```

[ ]:
```
df1.head()
```

```
[ ]:    User_ID Product_ID Gender   Age  Occupation City_Category  \
     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

       Stay_In_Current_City_Years  Marital_Status  Product_Category  Purchase
     0                          2               0                 3      8370
     1                          2               0                 1     15200
     2                          2               0                12      1422
     3                          2               0                12      1057
     4                         4+               0                 8      7969
```

```
[ ]: df1.columns
```

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

```
[ ]: df1.dtypes
```

```
[ ]: 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
```

```
[ ]: df1.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
```

[ ]:
```
len(df1)
```

[ ]:
```
550068
```

2 Statistical summary

[ ]:
```
df1.describe()
```

[ ]:
```
            User_ID     Occupation  Marital_Status  Product_Category  \
count  5.500680e+05  550068.000000   550068.000000     550068.000000
mean   1.003029e+06       8.076707        0.409653          5.404270
std    1.727592e+03       6.522660        0.491770          3.936211
min    1.000001e+06       0.000000        0.000000          1.000000
25%    1.001516e+06       2.000000        0.000000          1.000000
50%    1.003077e+06       7.000000        0.000000          5.000000
75%    1.004478e+06      14.000000        1.000000          8.000000
max    1.006040e+06      20.000000        1.000000         20.000000

            Purchase
count  550068.000000
mean     9263.968713
std      5023.065394
min        12.000000
25%      5823.000000
50%      8047.000000
75%     12054.000000
max     23961.000000
```

[ ]:
```
df1.describe(include=object)
```

[ ]:
```
       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
```

[ ]:
```
df1.isna().sum()
```

```
[ ]: 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
```

```
[ ]: df1.duplicated().sum()
```

```
[ ]: 0
```

- There are no missing values.
- There are no duplicates.

3 Number of unique values in: each column

```
[ ]: # Number of unique values in each column
     for i in df1.columns:
       print(i, ':', df1[i].nunique())
```

```
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
```

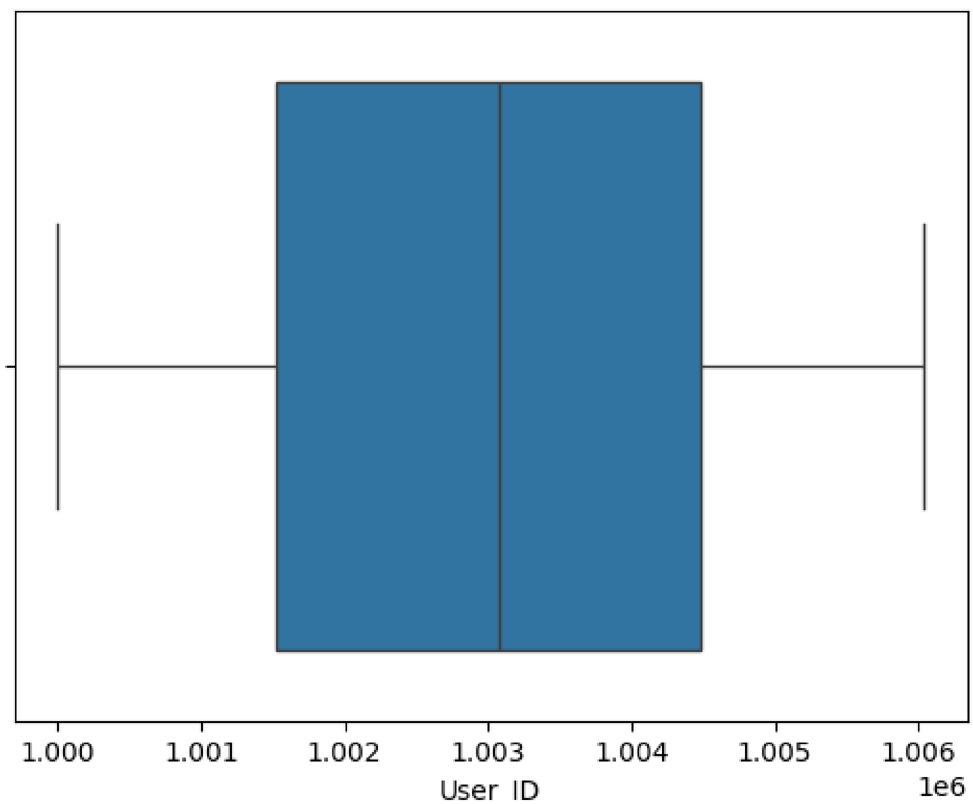
```
[ ]: for i in df1.columns:
         print('Number of Unique Values in',i,'column :', df1[i].nunique())
         print('-'*70)
```

```
Number of Unique Values in User_ID column : 5891
----------------------------------------------------------------------
Number of Unique Values in Product_ID column : 3631
----------------------------------------------------------------------
Number of Unique Values in Gender column : 2
----------------------------------------------------------------------
Number of Unique Values in Age column : 7
```

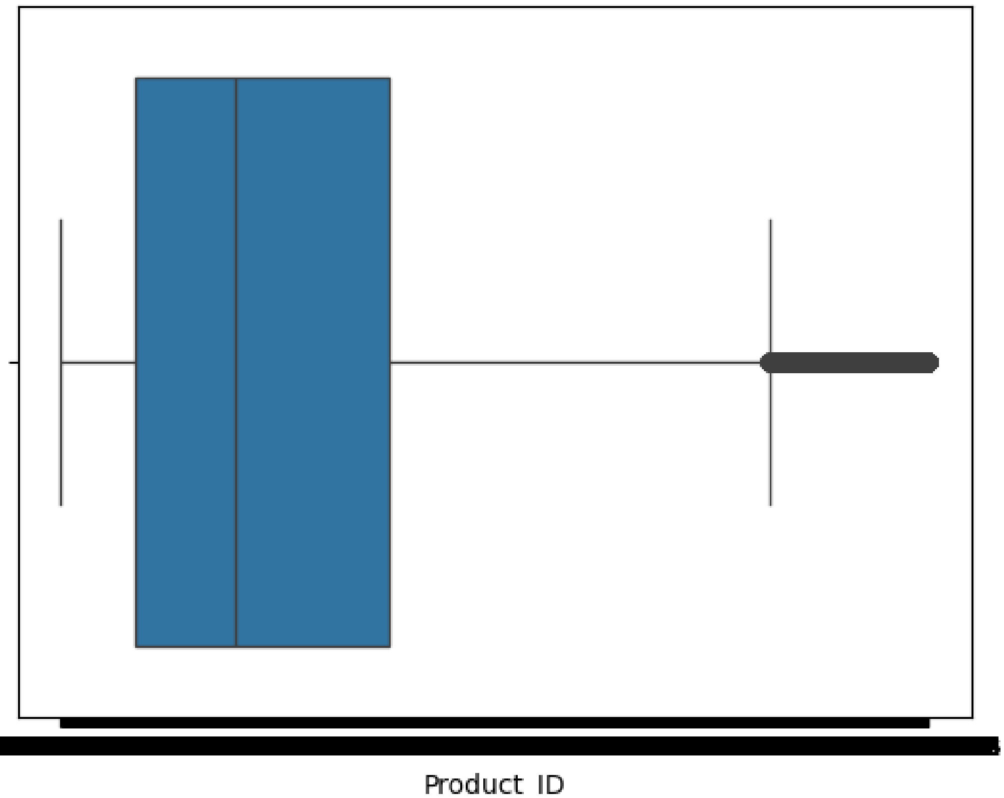
```
----------------------------------------------------------------------
Number of Unique Values in Occupation column : 21
----------------------------------------------------------------------
Number of Unique Values in City_Category column : 3
----------------------------------------------------------------------
Number of Unique Values in Stay_In_Current_City_Years column : 5
----------------------------------------------------------------------
Number of Unique Values in Marital_Status column : 2
----------------------------------------------------------------------
Number of Unique Values in Product_Category column : 20
----------------------------------------------------------------------
Number of Unique Values in Purchase column : 18105
----------------------------------------------------------------------
```

4 Missing Value & Outlier Detection (10 Points)

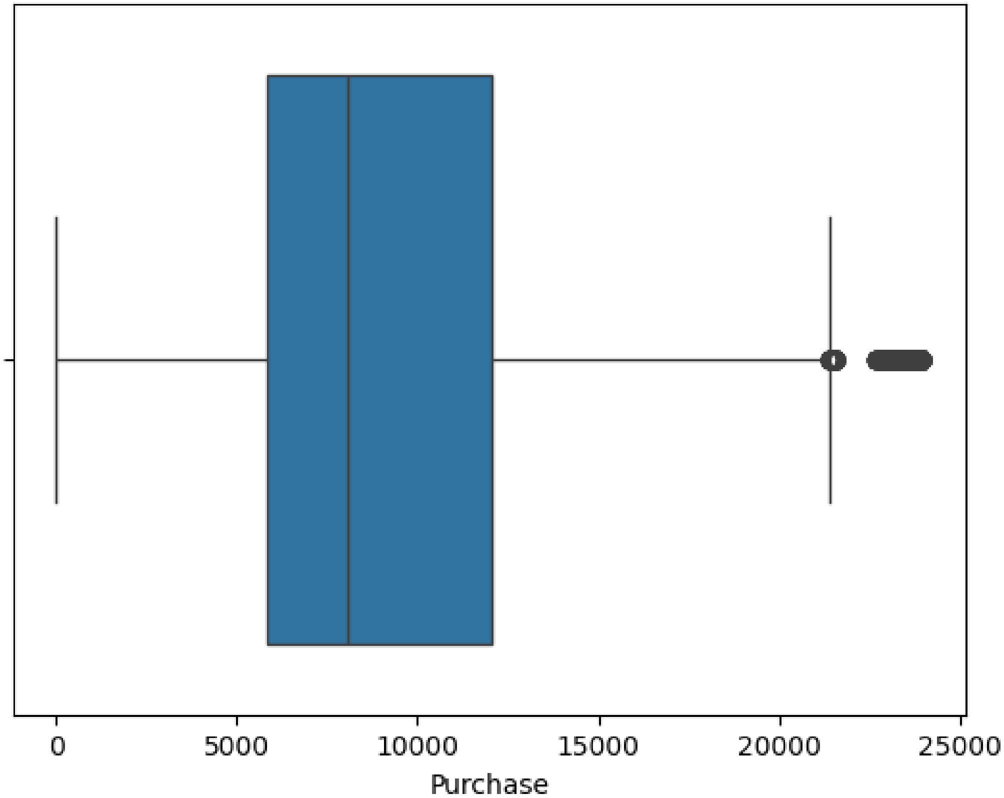
[ ]:
```
ax = sns.boxplot(x=df1["User_ID"])
plt.show()
```



[ ]:
```
ax = sns.boxplot(x=df1["Product_ID"])
plt.show()
```



[ ]:
```
ax = sns.boxplot(x=df1["Purchase"])
plt.show()
```



[ ]:
```
df1['Purchase'].max(),df1['Purchase'].min()
```

[ ]: (23961, 12)

[ ]:
```
# num_feat=['Purchase']
# for col in num_feat:
percentiles = df1['Purchase'].quantile([0.05,0.95]).values
print(percentiles)
df1['Purchase'] = np.clip(df1['Purchase'], percentiles[0], percentiles[1])
df1['Purchase'].max(),df1['Purchase'].min()
```

[1984. 19336.]

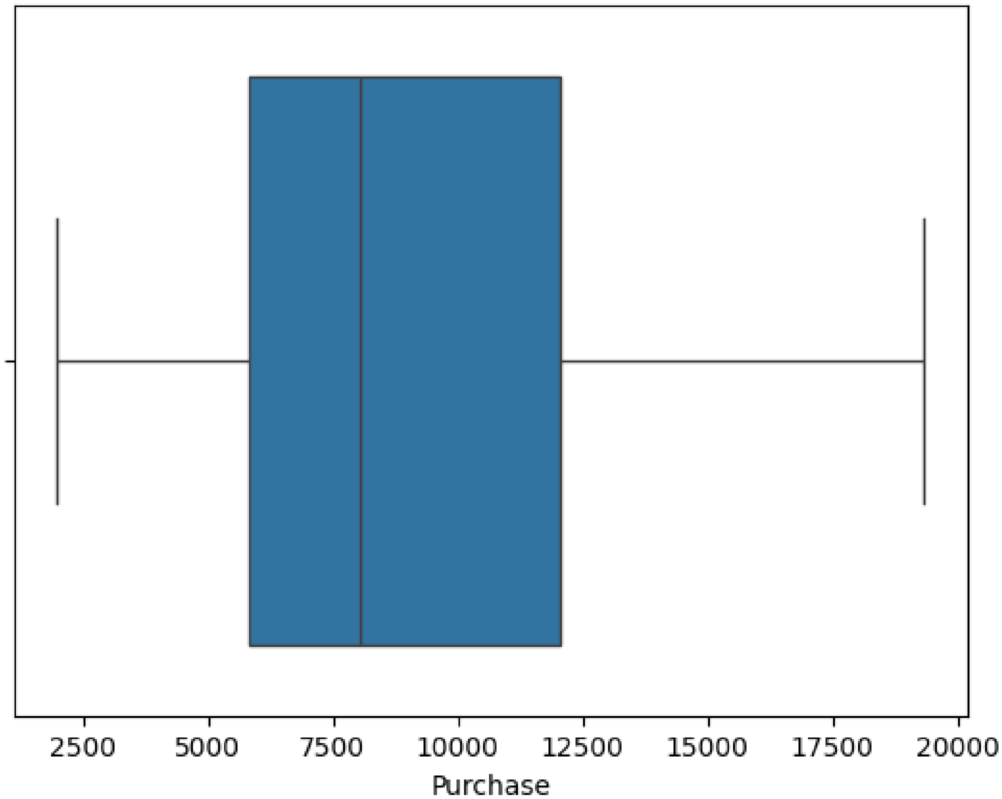
[ ]: (19336, 1984)

[ ]:
```
n=[1,2,3,45,46,47]
np.clip(n,3,45)
```

[ ]: array([3, 3, 3, 45, 45, 45])

[ ]:
```
sns.boxplot(x=df1['Purchase'])
```

[ ]: <Axes: xlabel='Purchase'>



#3. Data Exploration ## Non Graphical Analysis a. What products are different age groups buying?

[ ]:
```
df1.groupby(["Age"])["User_ID"].nunique()
```

[ ]:
```
Age
0-17      218
18-25    1069
26-35    2053
36-45    1167
46-50     531
51-55     481
55+       372
Name: User_ID, dtype: int64
```

[ ]:
```
df1['Age'].unique()
```

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

[ ]:
```
df1.groupby(['Age','Product_ID'])['Product_ID'].size()
```

[ ]:
```
Age    Product_ID
0-17   P00000142     55
       P00000242     19
       P00000342     11
       P00000442      2
       P00000542      9
                     ..
55+    P0099342       4
       P0099442       5
       P0099642       1
       P0099842       9
       P0099942       2
Name: Product_ID, Length: 20875, dtype: int64
```

[ ]:
```
new_df=df1.groupby('Age')['Product_ID'].size().sort_values(ascending=False)
new_df
```

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

[ ]:
```
df1[df1['Age']=='0-17'].groupby(['Product_Category']).value_counts()
```

[ ]:
```
Product_Category  User_ID  Product_ID  Gender  Age   Occupation  City_Category
Stay_In_Current_City_Years  Marital_Status  Purchase
1                 1000001  P00025442   F       0-17  10          A
2                           0               15416       1
                           P00110842   F       0-17  10          A
2                           0               11769       1
                  1004006  P00233442   M       0-17  7           C
3                           0               11431       1
                           P00173942   M       0-17  7           C
3                           0               11840       1
                           P00182242   M       0-17  7           C
3                           0               15629       1
                                                          ..
20                1003382  P00372445   F       0-17  10          A
2                           0               1984        1
                  1003460  P00375436   M       0-17  4           B
```

```
1                            0                1984        1
                 1003594  P00371644   M       0-17  0            B
0                            0                1984        1
                 1003604  P00375436   F       0-17  10           C
1                            0                1984        1
                 1006006  P00375436   F       0-17  0            C
1                            0                1984        1
Name: count, Length: 15102, dtype: int64
```

[ ]:
```
df1[df1['Age']=='26-35']['Product_Category'].value_counts().head(5)
```

[ ]:
```
Product_Category
5     61473
1     58249
8     44256
11     9874
2      8928
Name: count, dtype: int64
```

[ ]:
```
df1['Gender'].value_counts(normalize = True) * 100
```

[ ]:
```
Gender
M    75.310507
F    24.689493
Name: proportion, dtype: float64
```

[ ]:
```
df1[df1['Age']=='0-17']['Product_Category'].value_counts().head(5).index
```

[ ]:
```
Index([5, 1, 8, 3, 2], dtype='int64', name='Product_Category')
```

[ ]:
```
# df1[df1['Product_Category'].isin(index1)]['Product_Category']
```

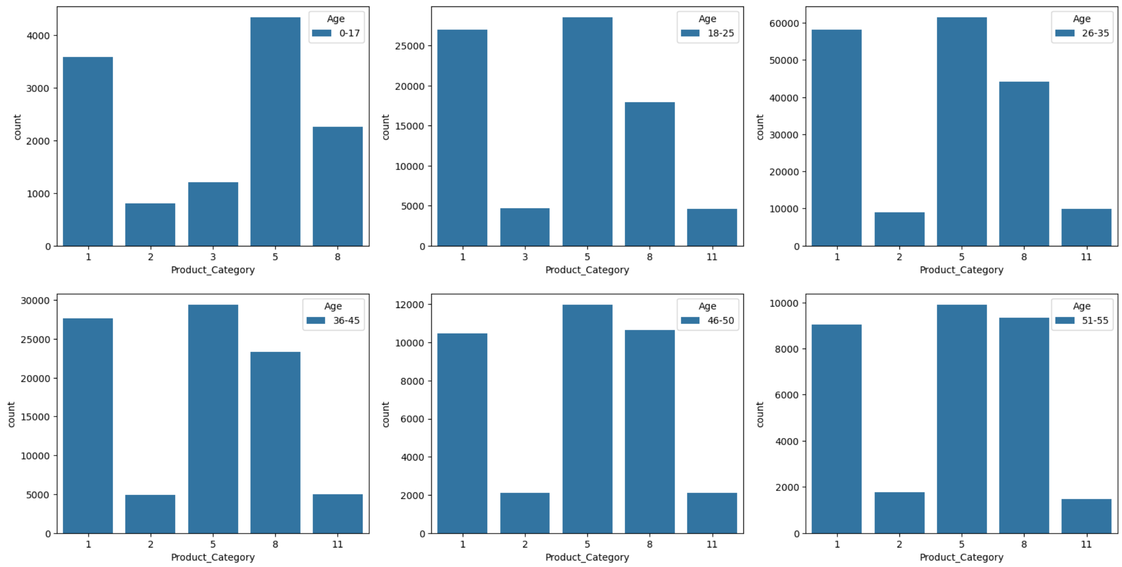
[ ]:
```
# top 10 product categories for age group 0-17
index1 = df1[df1['Age']=='0-17']['Product_Category'].value_counts().head(5).
 ↪index
age_cat1 = df1[df1['Product_Category'].isin(index1)]


# top 10 product categories for age group 18-25
index2 = df1[df1['Age']=='18-25']['Product_Category'].value_counts().head(5).
 ↪index
age_cat2 = df1[df1['Product_Category'].isin(index2)]


# top 10 product categories for age group '26-35'
index3 = df1[df1['Age']=='26-35']['Product_Category'].value_counts().head(5).
 ↪index
```

```
age_cat3 = df1[df1['Product_Category'].isin(index3)]

# top 10 product categories for age group '36-45'

index4 = df1[df1['Age']=='36-45']['Product_Category'].value_counts().head(5).
 ↪index
age_cat4 = df1[df1['Product_Category'].isin(index4)]


# top 10 product categories for age group '46-50'
index5 = df1[df1['Age']=='46-50']['Product_Category'].value_counts().head(5).
 ↪index
age_cat5 = df1[df1['Product_Category'].isin(index5)]


# top 10 product categories for age group '51-55'
index6 = df1[df1['Age']=='51-55']['Product_Category'].value_counts().head(5).
 ↪index
age_cat6 = df1[df1['Product_Category'].isin(index6)]
```

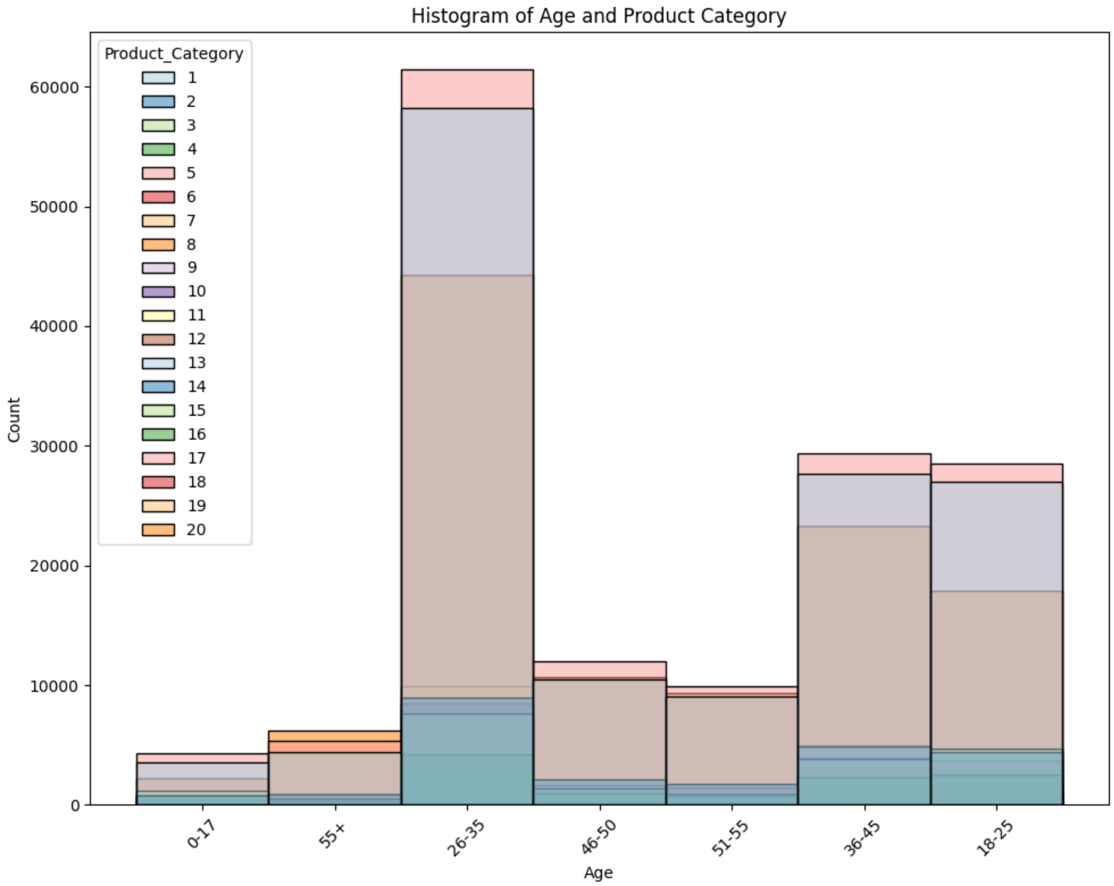
[ ]:
```
fig = plt.figure(figsize = (20,10))
plt.subplot(2,3,1)
sns.countplot(x='Product_Category',data␣
 ↪=age_cat1[age_cat1['Age']=='0-17'],hue='Age')


plt.subplot(2,3,2)
sns.countplot(x='Product_Category',data␣
 ↪=age_cat2[age_cat2['Age']=='18-25'],hue='Age')

plt.subplot(2,3,3)

sns.countplot(x='Product_Category',data␣
 ↪=age_cat3[age_cat3['Age']=='26-35'],hue='Age')

plt.subplot(2,3,4)

sns.countplot(x='Product_Category',data␣
 ↪=age_cat4[age_cat4['Age']=='36-45'],hue='Age')

plt.subplot(2,3,5)

sns.countplot(x='Product_Category',data␣
 ↪=age_cat5[age_cat5['Age']=='46-50'],hue='Age')

plt.subplot(2,3,6)
```

```
sns.countplot(x='Product_Category',data␣
 ↪=age_cat6[age_cat6['Age']=='51-55'],hue='Age')
```

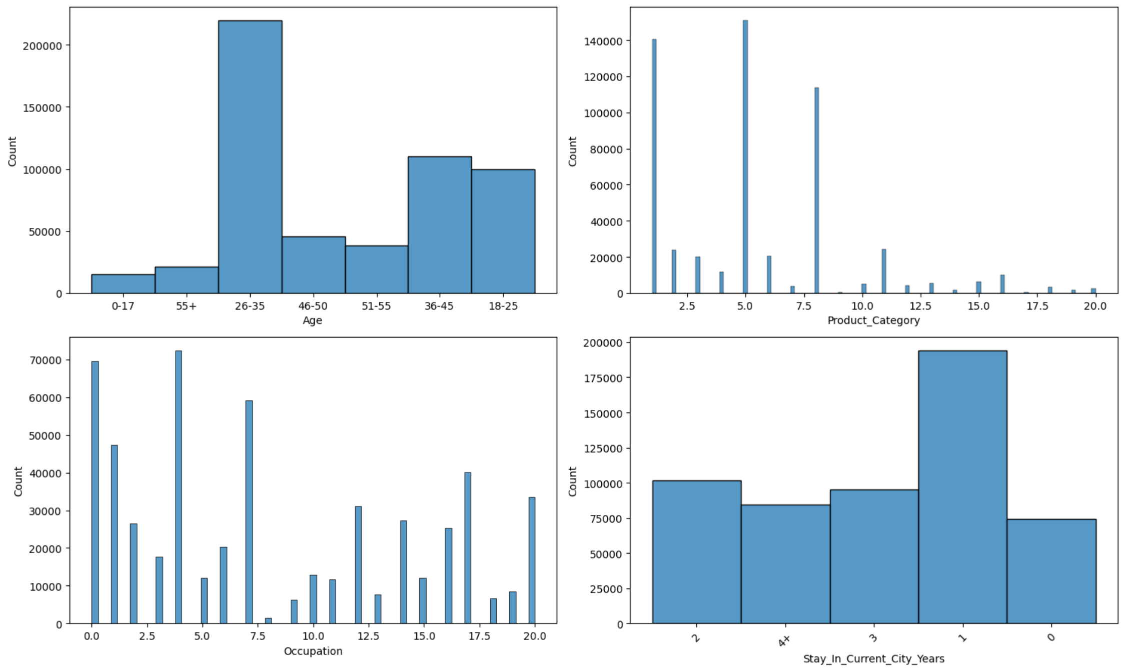
[ ]: <Axes: xlabel='Product_Category', ylabel='count'>



[ ]:
```
colors = ['skyblue', 'orange', 'green', 'red', 'purple', 'brown', 'pink',␣
 ↪'gray']

plt.figure(figsize=(10, 8))
sns.histplot(data=df1, x='Age', hue='Product_Category',palette='Paired')
plt.title('Histogram of Age and Product Category')
plt.xlabel('Age')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[ ]:
```
fig, axis = plt.subplots(nrows = 2, ncols = 2, figsize =(15,9))
# fig.subplots_adjust(top=1.2)

sns.histplot(data = df1 , x ='Age',  ax =axis[0,0] )
sns.histplot(data = df1 , x ='Occupation',  ax =axis[1,0] )
sns.histplot(data = df1 , x ='Product_Category', ax =axis[0,1] )
sns.histplot(data = df1 , x ='Stay_In_Current_City_Years',  ax =axis[1,1] )
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[ ]:
```
df1.groupby(['Product_Category','Product_ID'])['Age'].count()
```

[ ]:
```
Product_Category  Product_ID
1                 P00000642     512
                  P00000942      55
                  P00001042     503
                  P00001542      69
                  P00002042      93
                                ...
19                P00370293     785
                  P00370853     818
20                P00371644     899
                  P00372445     837
                  P00375436     814
Name: Age, Length: 3631, dtype: int64
```

[ ]:
```
df1.groupby(['Product_Category','Age']).size()
```

[ ]:
```
Product_Category  Age
1                 0-17      3585
                  18-25    26962
                  26-35    58249
                  36-45    27648
                  46-50    10474
                            ...
```

```
20                26-35      898
                  36-45      506
                  46-50      227
                  51-55      200
                  55+        160
Length: 140, dtype: int64
```

[ ]:
```
df1.groupby(by = ['Product_Category'])['Product_ID'].nunique().
 ↪sort_values(ascending=False)
```

[ ]:
```
Product_Category
8     1047
5      967
1      493
11     254
2      152
6      119
7      102
16      98
3       90
4       88
14      44
15      44
13      35
18      30
10      25
12      25
17      11
20       3
9        2
19       2
Name: Product_ID, dtype: int64
```

[ ]:
```
df1.head(10)
```

[ ]:
```
   User_ID Product_ID Gender   Age  Occupation City_Category  \
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
5  1000003  P00193542      M 26-35          15             A
6  1000004  P00184942      M 46-50           7             B
7  1000004  P00346142      M 46-50           7             B
8  1000004   P0097242      M 46-50           7             B
9  1000005  P00274942      M 26-35          20             A
```

```
  Stay_In_Current_City_Years  Marital_Status  Product_Category  Purchase
0                          2               0                 3      8370
1                          2               0                 1     15200
2                          2               0                12      1984
3                          2               0                12      1984
4                         4+               0                 8      7969
5                          3               0                 1     15227
6                          2               1                 1     19215
7                          2               1                 1     15854
8                          2               1                 1     15686
9                          1               1                 8      7871
```

[ ]:
```
df2 = df1[['Age','Marital_Status','Purchase','Gender']]
df2['Marital_Status']=df2['Marital_Status'].apply(lambda x: 'Married' if x==1␣
↪else 'Unmarried')
```

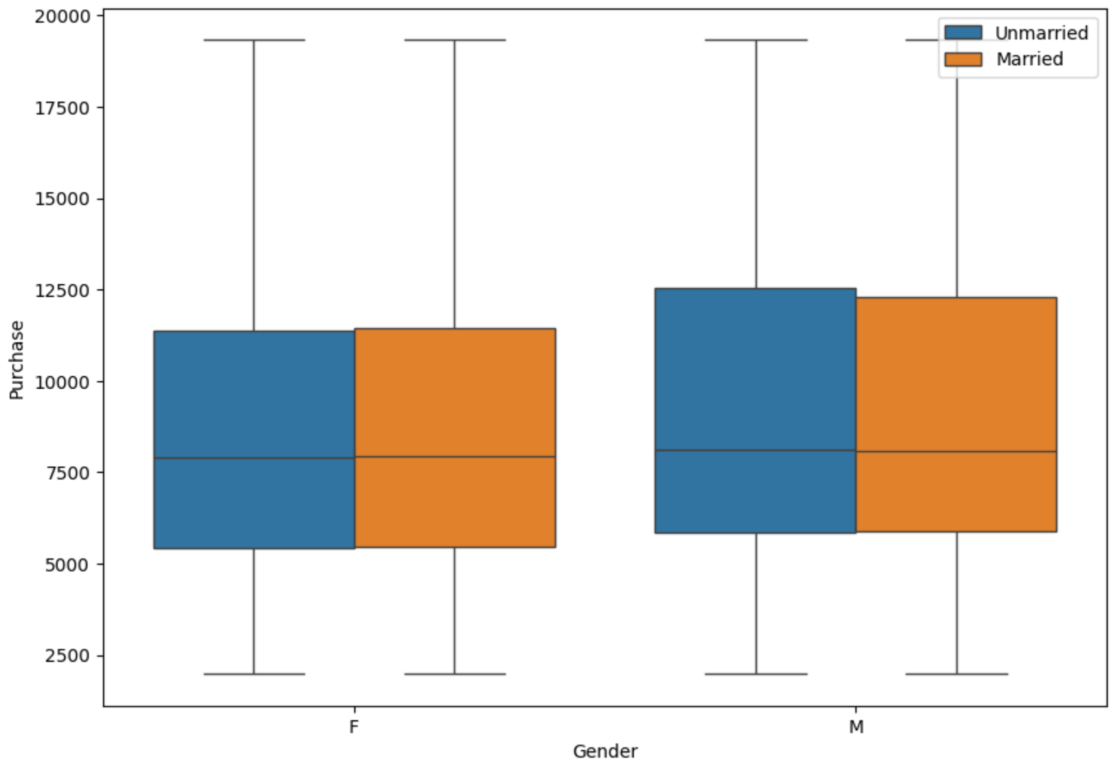
```
<ipython-input-84-2dfc777d0ddd>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2['Marital_Status']=df2['Marital_Status'].apply(lambda x: 'Married' if x==1
else 'Unmarried')
```

4.1 b. Is there a relationship between age, marital status, and the amount spent?

Add blockquote

[ ]:
```
plt.figure(figsize=(10,7))
sns.boxplot(data=df2,x='Gender',y='Purchase',hue='Marital_Status')
plt.legend(loc='upper right')
```

[ ]:
```
<matplotlib.legend.Legend at 0x7a4421823730>
```



4.2 Observation

- The **Median** purchase amount for Female is around **7500** for both **married and Unmarried**
- The **Median** purchase amount for Male is around **7500-7750** for both **married and Unmarried**
- The **Variance** is slightly more for **Males** purchases

5 Are there preferred product categories for different genders?

[ ]:
```
df1[df1['Gender']=='F'].groupby('Gender')['Product_Category'].value_counts().
↪head(5)
```

[ ]:
```
Gender  Product_Category
F       5                   41961
        8                   33558
        1                   24831
        3                    6006
        2                    5658
Name: count, dtype: int64
```

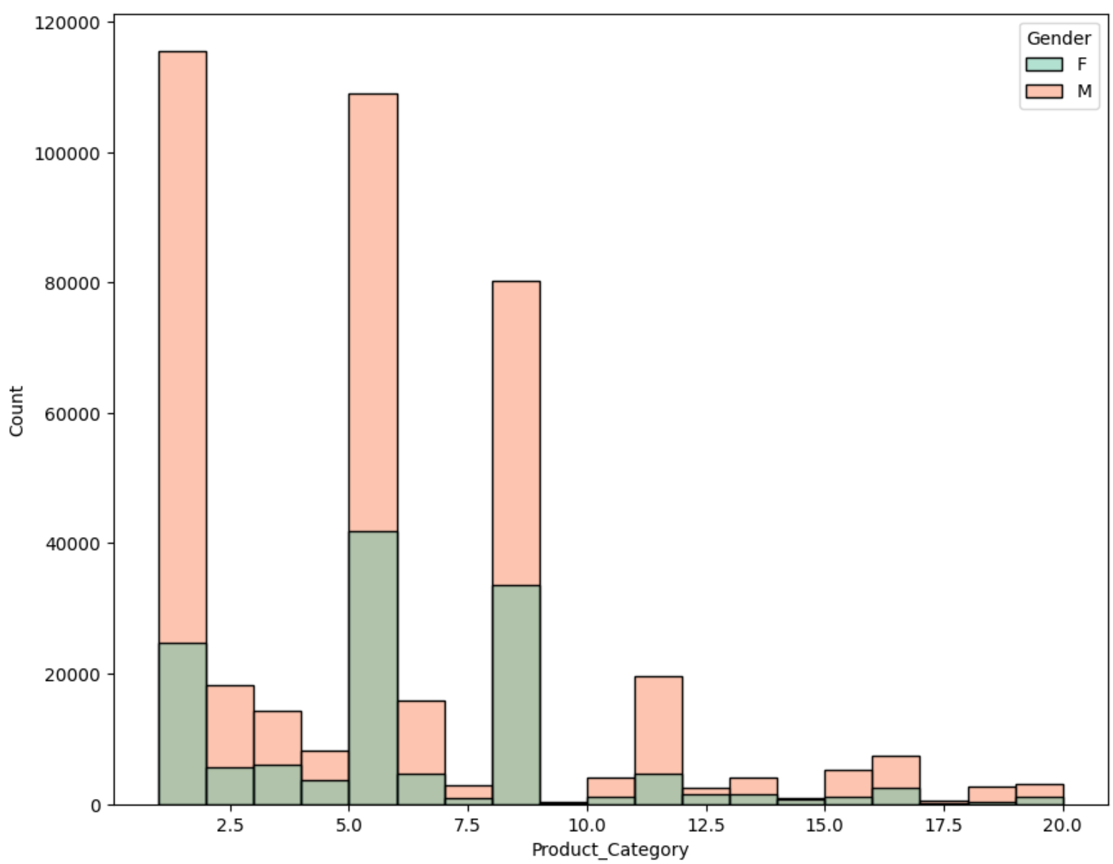
[ ]:
```
df1[df1['Gender']=='M'].groupby('Gender')['Product_Category'].value_counts().
↪head(5)
```

[ ]:
```
Gender  Product_Category
M       1                   115547
        5                   108972
        8                    80367
        11                   19548
        2                    18206
Name: count, dtype: int64
```

[ ]:
```
# pd.crosstab()
```

[ ]:
```
plt.figure(figsize=(10,8))
sns.
 ↪histplot(data=df1,x='Product_Category',hue='Gender',palette='Set2',bins=[1,2,3,4,5,6,7,8,9,
```

[ ]:
```
<Axes: xlabel='Product_Category', ylabel='Count'>
```

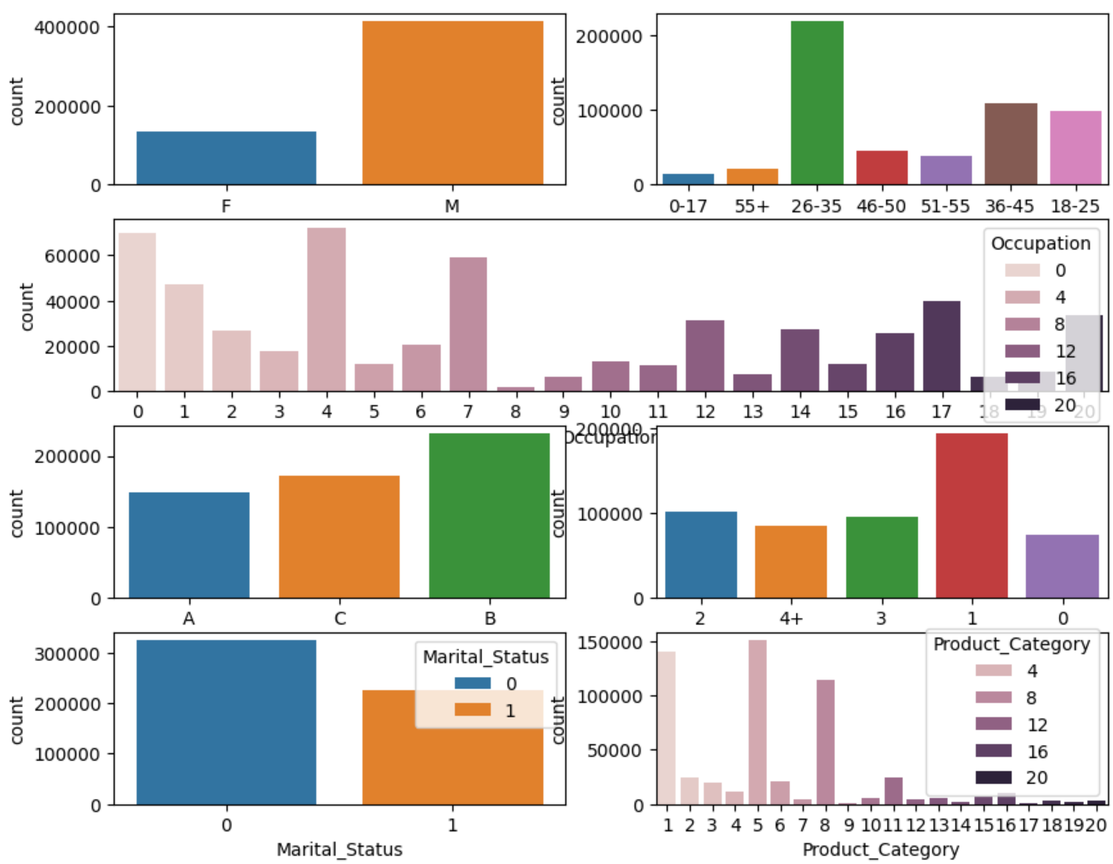


5.1 Observation

- Product Categories **1,5,8,11** are mostly preferred by males
- Product Categories **5,8,1,3 are** mostly preferred by females

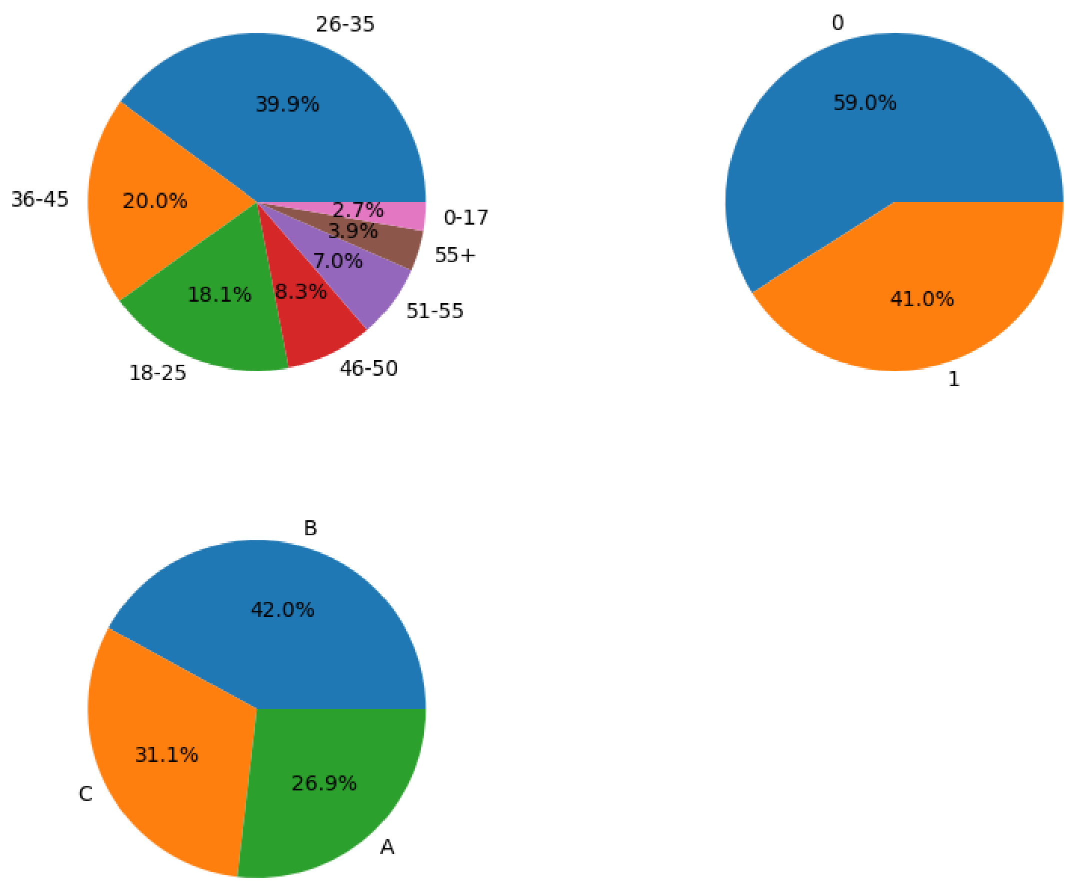
Univariate analysis

```
[ ]: plt.figure(figsize=(10,8))
plt.subplot(4,2,1)
sns.countplot(data=df1,x="Gender", hue='Gender')
plt.subplot(4,2,2)
sns.countplot(data=df1,x="Age", hue="Age")
plt.subplot(4,2,(3,4))
sns.countplot(data=df1,x="Occupation", hue="Occupation")
plt.subplot(4,2,5)
sns.countplot(data=df1,x="City_Category", hue="City_Category")
plt.subplot(4,2,6)
sns.countplot(data=df1,x="Stay_In_Current_City_Years",␣
↪hue="Stay_In_Current_City_Years")

plt.subplot(4,2,7)
sns.countplot(data=df1,x="Marital_Status", hue="Marital_Status")
plt.subplot(4,2,8)
sns.countplot(data=df1,x="Product_Category", hue="Product_Category")

plt.show()
```



[ ]:

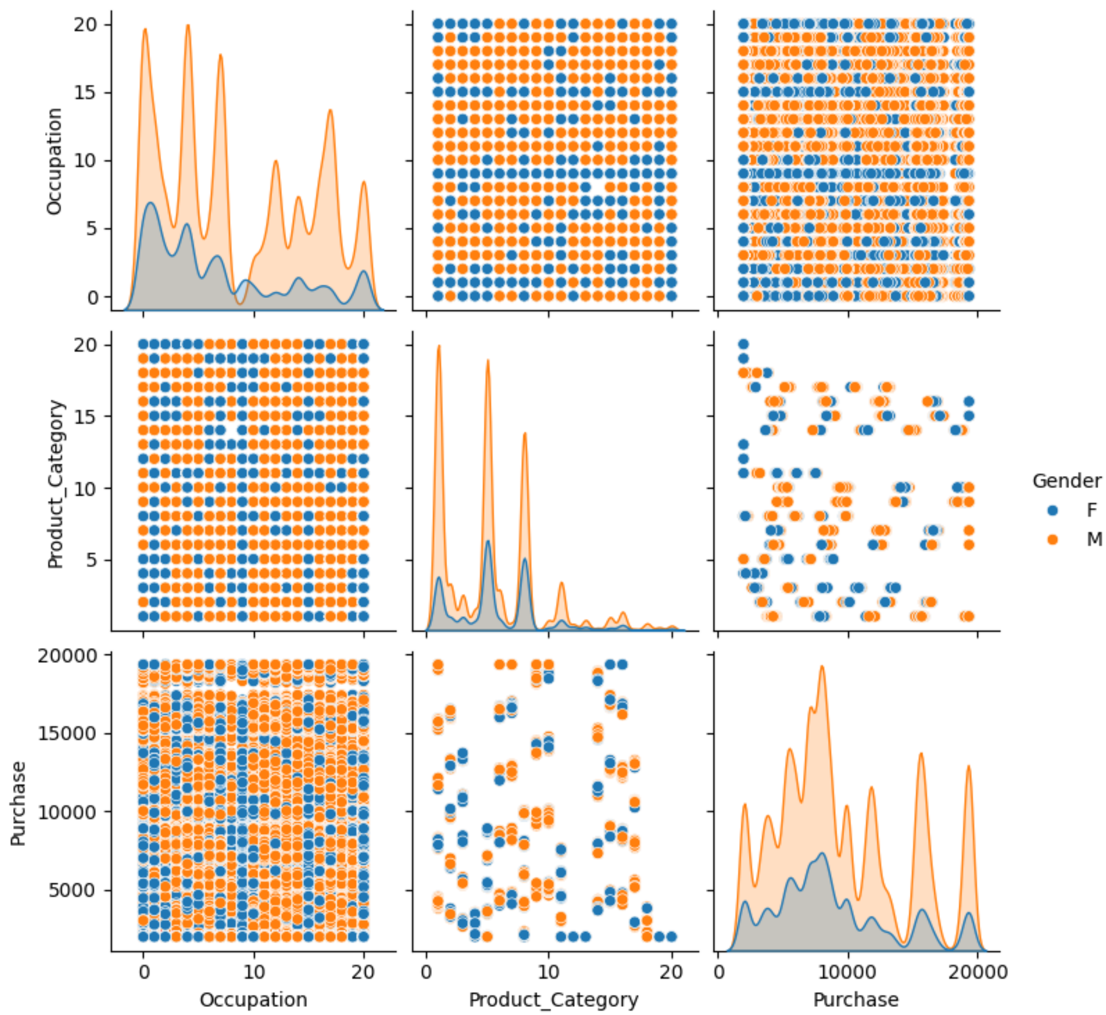
```
plt.figure(figsize=(10,8))
plt.subplot(2,2,1)

plt.pie(df1['Age'].value_counts(),labels=df1['Age'].value_counts().index,autopct='%1.1f%%')
plt.subplot(2,2,2)

plt.pie(df1['Marital_Status'].value_counts(),labels=df1['Marital_Status'].value_counts().index,autopct='%1.1f%%')

plt.subplot(2,2,3)

plt.pie(df1['City_Category'].value_counts(),labels=df1['City_Category'].value_counts().index,autopct='%1.1f%%')
plt.show()
```



Observation: - 26-35 age group occupies large portion of purchase amount with 0-17 being the least. - Unmarried customer occupy 59% of portion in purchasing.

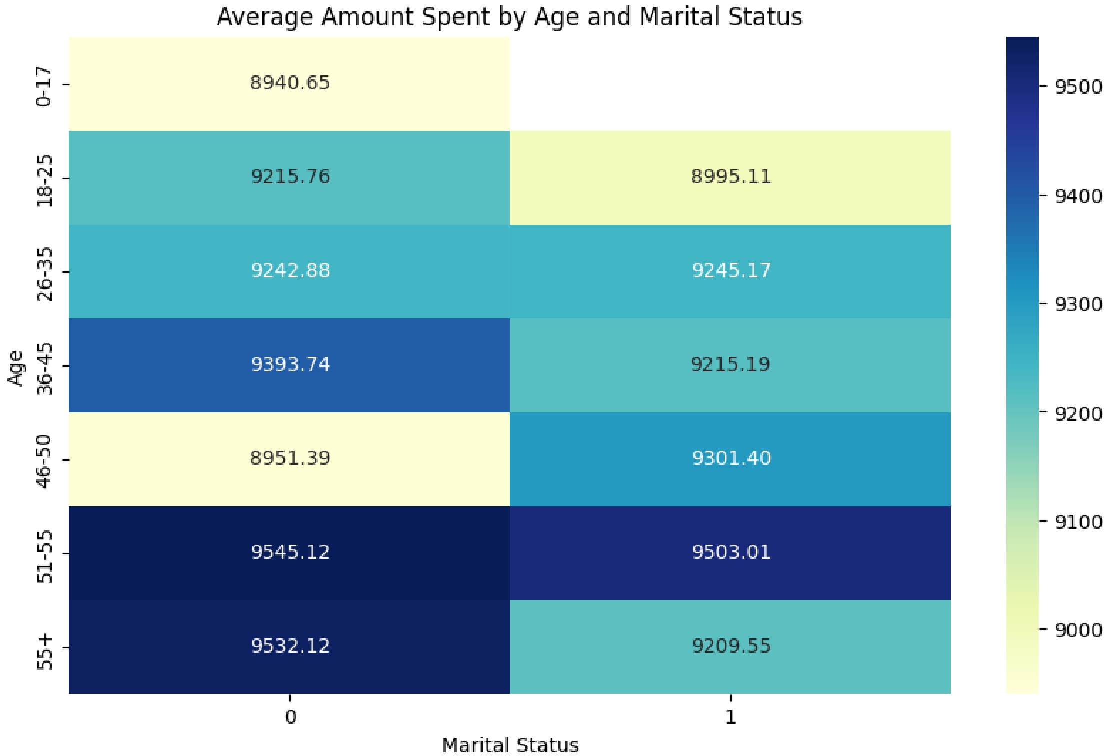
[ ]:
```
sample = df1[['Occupation',
        'Product_Category',
       'Purchase','Gender']]
```

[ ]:
```
sns.pairplot(data=sample,hue='Gender')
plt.show()
```



[ ]:
```
plt.figure(figsize=(10, 6))
pivot_table = df1.pivot_table(index="Age", columns="Marital_Status",␣
 ↪values="Purchase", aggfunc="mean")
sns.heatmap(pivot_table, cmap="YlGnBu", annot=True, fmt=".2f")
plt.title("Average Amount Spent by Age and Marital Status")
plt.xlabel("Marital Status")
```

```
plt.ylabel("Age")
plt.show()
```



Observation - The heatmap visualization offers a clearer representation of the average spending patterns across various age groups and marital status categories.

- The lack of significant variation in spending amounts suggests that age and marital status may not strongly influence the amount spent.

6 4. How does gender affect the amount spent?

[ ]:
```
df=df1.groupby(['User_ID','Gender'])['Purchase'].sum().reset_index()
```

[ ]:
```
# Male
import scipy.stats as stats
from scipy.stats import norm
from statsmodels.stats.weightstats import ztest
male_data = df1[df1['Gender']=='M']['Purchase']
female_data = df1[df1['Gender']=='F']['Purchase']
n1 = len(male_data)
n2 = len(female_data)
male_mean = df[df['Gender']=='M']['Purchase'].mean()
```

```
female_mean = df[df['Gender']=='F']['Purchase'].mean()
CI = 0.95


z1 = norm.ppf(0.025)
z2 = norm.ppf(0.975)

x1_male = z1*(male_data.std()/(np.sqrt(n1))) + male_mean
x2_male = z2*(male_data.std()/(np.sqrt(n1))) + male_mean

x1_female = z1*(female_data.std()/(np.sqrt(n2))) + female_mean
x2_female = z2*(female_data.std()/(np.sqrt(n2))) + female_mean

print(male_mean,female_mean)
print("x1_male and x2_male : ")

print(x1_male,x2_male)
print(f'Range:{x2_male-x1_male}')
print('-'*30)
print("x1_female and x2_female : ")

print(x1_female,x2_female)
print(f'Range:{x2_female-x1_female}')
```

```
924335.9592899408 712185.3523409364
x1_male and x2_male :
924320.9589043823 924350.9596754992
Range:30.00077111693099
------------------------------
x1_female and x2_female :
712160.9035793531 712209.8011025197
Range:48.89752316661179
```

6.1 Bootstraping with different sample sizes

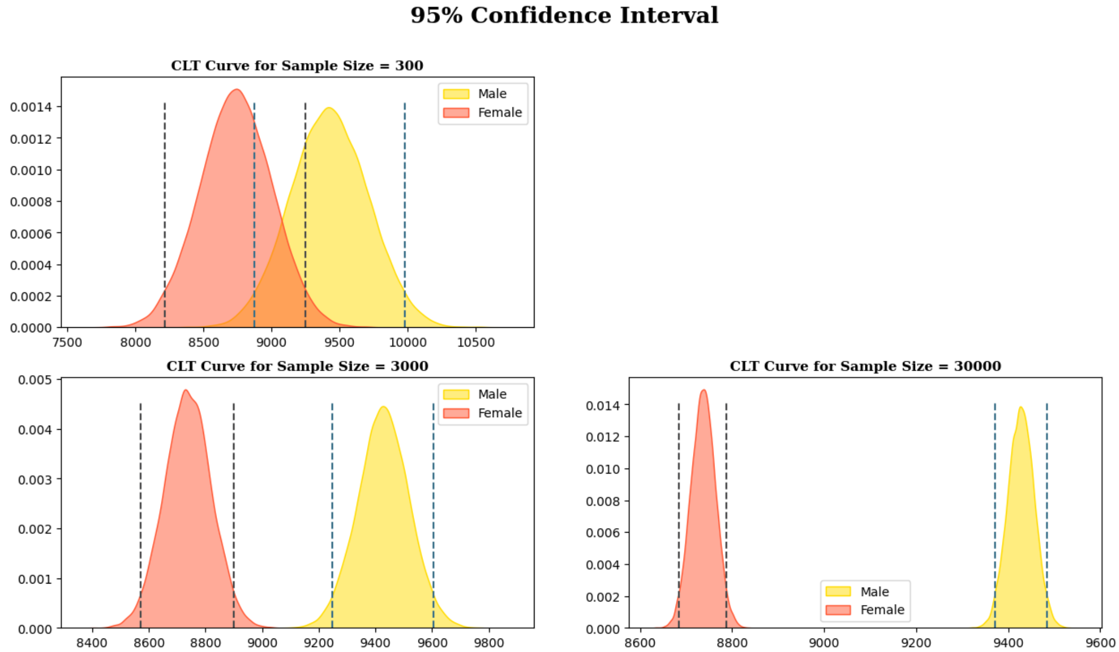
[ ]:
```
len(male_data),len(female_data)
def confidence_interval(data,ci):
    #Converting the list to series
    lower_ci = (100-ci)/2
    upper_ci = (100+ci)/2

    #Calculating lower limit and upper limit of confidence interval
    interval = np.percentile(data,[lower_ci,upper_ci]).round(0)

    return interval
```

[ ]:
```
a=[]
```

```
[ ]: def Bootstraping(ci):

    #setting the plot style
    fig = plt.figure(figsize = (15,8))
    gs = fig.add_gridspec(2,2)

    #creating separate data frames for each gender
    walmart_data_male = df1.loc[df1['Gender'] == 'M','Purchase']
    walmart_data_female = df1.loc[df1['Gender'] == 'F','Purchase']

    #sample sizes and corresponding plot positions
    sample_sizes = [(300,0,0),(3000,1,0),(30000,1,1)]

    #number of samples to be taken from purchase amount
    bootstrap_samples = 20000

    male_samples = {}
    female_samples = {}

    # In each iteration of the loop, "i", "x", "y" will hold the "sample size",␣
↪"row position", "column position" respectively for plotting purposes
    # This allows iterate over different sample sizes and correspondingly place␣
↪the resulting plots in different positions within a grid of subplots
    for i,x,y in sample_sizes:
        male_means = [] #list for collecting the means of male sample
        female_means = [] #list for collecting the means of female sample

        for j in range(bootstrap_samples):
            #creating random 5000 samples of i (sample size)
            male_bootstrapped_samples = np.random.choice(walmart_data_male,size␣
↪= i)
            female_bootstrapped_samples = np.random.
↪choice(walmart_data_female,size = i)

            #calculating mean of those samples
            male_sample_mean = np.mean(male_bootstrapped_samples)
            female_sample_mean = np.mean(female_bootstrapped_samples)

            #appending the mean to the list
            male_means.append(male_sample_mean)
            female_means.append(female_sample_mean)

        #storing the above sample generated
        male_samples[f'{ci}%_{i}'] = male_means
        female_samples[f'{ci}%_{i}'] = female_means

        #creating a temporary dataframe for creating kdeplot
```

```
        temp_walmart_data = pd.DataFrame(data = {'male_means':
↪male_means,'female_means':female_means})

                                                    #plotting kdeplots
        #plot position
        ax = fig.add_subplot(gs[x,y])

        #plots for male and female
        sns.kdeplot(data = temp_walmart_data,x = 'male_means',color ="#FFDC00"␣
↪,fill = True, alpha = 0.5,ax = ax,label = 'Male')
        sns.kdeplot(data = temp_walmart_data,x = 'female_means',color␣
↪="#FF5733" ,fill = True, alpha = 0.5,ax = ax,label = 'Female')

        #calculating confidence intervals for given confidence level(ci)
        m_range = confidence_interval(male_means,ci)
        f_range = confidence_interval(female_means,ci)

        #plotting confidence interval on the distribution
        for k in m_range:
            ax.axvline(x = k,ymax = 0.9, color ="#3A7089",linestyle = '--')

        for k in f_range:
            ax.axvline(x = k,ymax = 0.9, color ="#4b4b4c",linestyle = '--')


        # #removing the axis lines
        # for axislines in ['top','left','right']:
        #     ax.spines[axislines].set_visible(False)

        # adjusting axis labels
        # ax.set_yticks([])
        ax.set_ylabel('')
        ax.set_xlabel('')

        #setting title for visual
        ax.set_title(f'CLT Curve for Sample Size = {i}',{'font':'serif', 'size':
↪11,'weight':'bold'})

        plt.legend()

    #setting title for visual
    fig.suptitle(f'{ci}% Confidence Interval',font = 'serif', size = 18, weight␣
↪= 'bold')

    plt.show()

    return male_samples,female_samples
```

[ ]:
```
male,female = Bootstraping(95)
```



[ ]:
```
np.mean(a)
norm.ppf(0.025),norm.ppf(0.975)
```

```
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:3504:
RuntimeWarning: Mean of empty slice.
  return _methods._mean(a, axis=axis, dtype=dtype,
/usr/local/lib/python3.10/dist-packages/numpy/core/_methods.py:129:
RuntimeWarning: invalid value encountered in scalar divide
  ret = ret.dtype.type(ret / rcount)
```

[ ]: (-1.9599639845400545, 1.959963984540054)

[ ]:
```
for i in male:
  confidence_level = confidence_interval(male[i],95)
  print(f'The confidence interval for ',confidence_level)
  print(f'The range is :{confidence_level[1]-confidence_level[0]} ')
  print('-'*60)
```

```
The confidence interval for  [8873. 9982.]
The range is :1109.0
------------------------------------------------------------
The confidence interval for  [9248. 9606.]
The range is :358.0
------------------------------------------------------------
```

```
The confidence interval for  [9372. 9484.]
The range is :112.0
------------------------------------------------------------
```

```
[ ]: for i in female:
       confidence_level = confidence_interval(female[i],95)
       print(f'The confidence interval for ',confidence_level)
       print(f'The range is :{confidence_level[1]-confidence_level[0]} ')
       print('-'*60)
```

```
The confidence interval for  [8219. 9251.]
The range is :1032.0
------------------------------------------------------------
The confidence interval for  [8572. 8900.]
The range is :328.0
------------------------------------------------------------
The confidence interval for  [8684. 8788.]
The range is :104.0
------------------------------------------------------------
```

a. From the above calculated CLT answer the following questions.

- i. Is the confidence interval computed using the entire dataset wider for one of the genders? Why is this the case?
- ii. How is the width of the confidence interval affected by the sample size?
- iii. Do the confidence intervals for different sample sizes overlap?
- iv. How does the sample size affect the shape of the distributions of the means?

6.2 Observation

- Women tends use less money per transaction as the upper bounds of confidence interval for any sample size is high.
- We can see that as we increase the sample size we are getting the confidence interval range values more narrower and precise.
- We can observe that the except for sample size = 300 there is no overlapping of confidence intervals of both women and men.
- We can observe that as sample size increases the data and confidence interval becomes more narrower and distribution becomes more normally distributed

6.3 Report

a. Report whether the confidence intervals for the average amount spent by males and females (computed using all the data) overlap. How can Walmart leverage this conclusion to make changes or improvements? - From the analysis we can see that except for sample size 300 there is no over lap between male and female.There is statistically significant difference between the average spending per transaction for men and women within the given samples

How can Walmart leverage this conclusion to make changes or improvements?

- Walmart can develop targeted marketing campaigns, loyalty programs, or product bundles tailored to the distinct spending behaviors of male and female customers.
- Pricing and discount strategies can be adjusted based on the data of average spending per transaction by gender.
- The goal is to optimize revenue generation by aligning pricing strategies with the observed spending patterns of different customer segments.

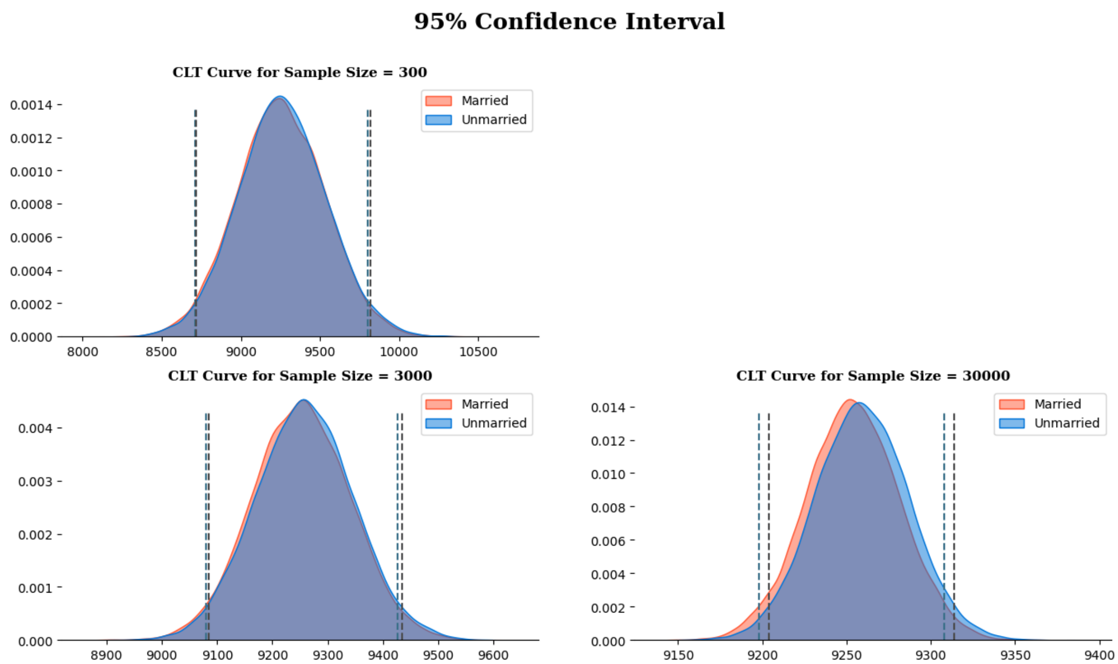
7 5. How does Marital_Status affect the amount spent?

[ ]:
```
def Bootstraping_marital(ci):
    #setting the plot style
    fig = plt.figure(figsize = (15,8))
    gs = fig.add_gridspec(2,2)

    #creating separate data frames
    df_married = df1.loc[df1['Marital_Status'] == 1,'Purchase']
    df_unmarried = df1.loc[df1['Marital_Status'] == 0,'Purchase']

    #sample sizes and corresponding plot positions
    sample_sizes = [(300,0,0),(3000,1,0),(30000,1,1)]

    #number of samples to be taken from purchase amount
    bootstrap_samples = 20000

    married_samples = {}
    unmarried_samples = {}

    for i,x,y in sample_sizes:
        married_means = [] #list for collecting the means of married sample
        unmarried_means = [] #list for collecting the means of unmarried sample

        for j in range(bootstrap_samples):

            #creating random 5000 samples of i sample size
            married_bootstrapped_samples = np.random.choice(df_married,size = i)
            unmarried_bootstrapped_samples = np.random.choice(df_unmarried,size␣
↪= i)

            #calculating mean of those samples
            married_sample_mean = np.mean(married_bootstrapped_samples)
            unmarried_sample_mean = np.mean(unmarried_bootstrapped_samples)

            #appending the mean to the list
```

```
            married_means.append(married_sample_mean)
            unmarried_means.append(unmarried_sample_mean)

        #storing the above sample generated
        married_samples[f'{ci}%_{i}'] = married_means
        unmarried_samples[f'{ci}%_{i}'] = unmarried_means

        #creating a temporary dataframe for creating kdeplot
        temp_df = pd.DataFrame(data = {'married_means':
↪married_means,'unmarried_means':unmarried_means})

                                                        #plotting kdeplots
        #plot position
        ax = fig.add_subplot(gs[x,y])

        #plots for married and unmarried
        sns.kdeplot(data = temp_df,x = 'married_means',color ="#FF5733" ,fill =␣
↪True, alpha = 0.5,ax = ax,label = 'Married')
        sns.kdeplot(data = temp_df,x = 'unmarried_means',color ="#0074D9" ,fill␣
↪= True, alpha = 0.5,ax = ax,label = 'Unmarried')

        #calculating confidence intervals for given confidence level(ci)
        m_range = confidence_interval(married_means,ci)
        u_range = confidence_interval(unmarried_means,ci)

        #plotting confidence interval on the distribution
        for k in m_range:
            ax.axvline(x = k,ymax = 0.9, color ="#3A7089",linestyle = '--')

        for k in u_range:
            ax.axvline(x = k,ymax = 0.9, color ="#4b4b4c",linestyle = '--')


        #removing the axis lines
        for axislines in ['top','left','right']:
            ax.spines[axislines].set_visible(False)

        # adjusting axis labels
        ax.set_ylabel('')
        ax.set_xlabel('')

        #setting title for visual
        ax.set_title(f'CLT Curve for Sample Size = {i}',{'font':'serif', 'size':
↪11,'weight':'bold'})

        plt.legend()
```

```
    #setting title for visual
    fig.suptitle(f'{ci}% Confidence Interval',font = 'serif', size = 18, weight␣
↪= 'bold')

    plt.show()

    return married_samples,unmarried_samples
```

[ ]:
```
m_samp_95,u_samp_95 = Bootstraping_marital(95)
```



[ ]:
```
m_samp_95,u_samp_95
for i in m_samp_95:
  confidence_level = confidence_interval(m_samp_95[i],95)
  print(f'The confidence interval for ',confidence_level)
  print(f'The range is :{confidence_level[1]-confidence_level[0]} ')
  print('-'*60)

for i in u_samp_95:
  confidence_level = confidence_interval(u_samp_95[i],95)
  print(f'The confidence interval for ',confidence_level)
  print(f'The range is :{confidence_level[1]-confidence_level[0]} ')
  print('-'*60)
```

7.1 Observation

- We can observe that there is overlapping of intervals for all kinds of sample sizes which implies that there is no signifcant difference between married and unmarried.
- We can see that as we increase the sample size we are getting the confidence interval range values more narrower and precise.
- We can observe that as sample size increases the data and confidence interval becomes more narrower and distribution becomes more normally distributed

7.2 Report

a. Report whether the confidence intervals for the average amount spent by males and females (computed using all the data) overlap. How can Walmart leverage this conclusion to make changes or improvements? - From the analysis,we can observe that the confidence interval overlap for all the sample sizes. This means that there is no significant difference between the average spending per transaction for married and unmarried customers within the given samples.

How can Walmart leverage this conclusion to make changes or improvements?

- Walmart can avoid the need to allocate marketing resources specifically targeting married or unmarried customers.
- The focus can be on ensuring a consistent and satisfactory shopping experience for both married and unmarried customers.
- Walmart can maintain a diverse range of products and promotions that cater to the varied preferences of a broad customer base.

8 6. How does Age affect the amount spent?

[ ]:
```
def Bootstraping_age(ci):

    #setting the plot style
    fig = plt.figure(figsize = (15,15))
    gs = fig.add_gridspec(4,1)

    #creating separate data frames
    df_1 = df1.loc[df1['Age'] == '0-17','Purchase']
    df_2 = df1.loc[df1['Age'] == '18-25','Purchase']
    df_3 = df1.loc[df1['Age'] == '26-35','Purchase']
    df_4 = df1.loc[df1['Age'] == '36-45','Purchase']
    df_5 = df1.loc[df1['Age'] == '46-50','Purchase']
    df_6 = df1.loc[df1['Age'] == '51-55','Purchase']
    df_7 = df1.loc[df1['Age'] == '55+','Purchase']


    #sample sizes and corresponding plot positions
    sample_sizes = [(300,0),(5000,1),(50000,2)]
```

```
#number of samples to be taken from purchase amount
bootstrap_samples = 20000

samples1,samples2,samples3,samples4,samples5,samples6,samples7 =␣
↪{},{},{},{},{},{},{}

for i,x in sample_sizes:
    l1,l2,l3,l4,l5,l6,l7 = [],[],[],[],[],[],[]

    for j in range(bootstrap_samples):
        #creating random 5000 samples of i sample size
        bootstrapped_samples_1 = np.random.choice(df_1,size = i)
        bootstrapped_samples_2 = np.random.choice(df_2,size = i)
        bootstrapped_samples_3 = np.random.choice(df_3,size = i)
        bootstrapped_samples_4 = np.random.choice(df_4,size = i)
        bootstrapped_samples_5 = np.random.choice(df_5,size = i)
        bootstrapped_samples_6 = np.random.choice(df_6,size = i)
        bootstrapped_samples_7 = np.random.choice(df_7,size = i)

        #calculating mean of those samples
        sample_mean_1 = np.mean(bootstrapped_samples_1)
        sample_mean_2 = np.mean(bootstrapped_samples_2)
        sample_mean_3 = np.mean(bootstrapped_samples_3)
        sample_mean_4 = np.mean(bootstrapped_samples_4)
        sample_mean_5 = np.mean(bootstrapped_samples_5)
        sample_mean_6 = np.mean(bootstrapped_samples_6)
        sample_mean_7 = np.mean(bootstrapped_samples_7)

        #appending the mean to the list
        l1.append(sample_mean_1)
        l2.append(sample_mean_2)
        l3.append(sample_mean_3)
        l4.append(sample_mean_4)
        l5.append(sample_mean_5)
        l6.append(sample_mean_6)
        l7.append(sample_mean_7)

    #storing the above sample generated
    samples1[f'{ci}%_{i}'] = l1
    samples2[f'{ci}%_{i}'] = l2
    samples3[f'{ci}%_{i}'] = l3
    samples4[f'{ci}%_{i}'] = l4
    samples5[f'{ci}%_{i}'] = l5
    samples6[f'{ci}%_{i}'] = l6
    samples7[f'{ci}%_{i}'] = l7
```

```
        #creating a temporary dataframe for creating kdeplot
        temp_df = pd.DataFrame(data = {'0-17':l1,'18-25':l2,'26-35':l3,'36-45':
↪l4,'46-50':l5,'51-55':l6,'55+':l7})

                                                #plotting kdeplots
        #plot position
        ax = fig.add_subplot(gs[x])

        #plots
        for p,q in [('#FF5733', '0-17'), ('#0074D9', '18-25'), ('#FFDC00',␣
↪'26-35'), ('#2ECC40', '36-45'), ('#FF4136', '46-50'), ('#FFD700', '51-55'),␣
↪('#3D9970', '55+')]:

            sns.kdeplot(data = temp_df,x = q,color =p ,fill = True, alpha = 0.
↪5,ax = ax,label = q)

        #removing the axis lines
        for axislines in ['top','left','right']:
            ax.spines[axislines].set_visible(False)

        # adjusting axis labels
        ax.set_ylabel('')
        ax.set_xlabel('')

        #setting title for visual
        ax.set_title(f'CLT Curve for Sample Size = {i}',{'font':'serif', 'size':
↪11,'weight':'bold'})

        plt.legend()

    #setting title for visual
    fig.suptitle(f'{ci}% Confidence Interval',font = 'serif', size = 18, weight␣
↪= 'bold')

    plt.show()

    return samples1,samples2,samples3,samples4,samples5,samples6,samples7
```

[ ]:
```
samples1,samples2,samples3,samples4,samples5,samples6,samples7 =␣
↪Bootstraping_age(95)
```

[ ]:
```
samples1,samples2,samples3,samples4,samples5,samples6,samples7
for i in samples1:
  confidence_level = confidence_interval(samples1[i],95)
  print(f'The confidence interval for {i}',confidence_level)
```

```
  print(f'The range is :{confidence_level[1]-confidence_level[0]} ')
  print('-'*60)
for i in samples2:
  confidence_level = confidence_interval(samples2[i],95)
  print(f'The confidence interval for {i}',confidence_level)
  print(f'The range is :{confidence_level[1]-confidence_level[0]} ')
for i in samples3:
  confidence_level = confidence_interval(samples3[i],95)
  print(f'The confidence interval for {i}',confidence_level)
  print(f'The range is :{confidence_level[1]-confidence_level[0]} ')
  print('-'*60)
for i in samples4:
  confidence_level = confidence_interval(samples4[i],95)
  print(f'The confidence interval for {i} ',confidence_level)
  print(f'The range is :{confidence_level[1]-confidence_level[0]} ')
for i in samples5:
  confidence_level = confidence_interval(samples5[i],95)
  print(f'The confidence interval for {i} ',confidence_level)
  print(f'The range is :{confidence_level[1]-confidence_level[0]} ')
  print('-'*60)
for i in samples6:
  confidence_level = confidence_interval(samples6[i],95)
  print(f'The confidence interval for {i} ',confidence_level)
  print(f'The range is :{confidence_level[1]-confidence_level[0]} ')
for i in samples7:
  confidence_level = confidence_interval(samples7[i],95)
  print(f'The confidence interval for{i} ',confidence_level)
  print(f'The range is :{confidence_level[1]-confidence_level[0]} ')
  print('-'*60)
```

8.1 Observation

- As sample size increases less over lapping in confidence interval is observed and there is significant difference in average per transaction purchase.
- We can see that as we increase the sample size we are getting the confidence interval range values more narrower and precise.
- 0 - 17 : Customers in this age group have the lowest spending per transaction
- 18 - 25, 26 - 35, 46 - 50 : Customers in these age groups have overlapping confidence intervals indicating similar buying characteristics
- 36 - 45, 55+ : Customers in these age groups have overlapping confidence intervals indicating and similar spending patterns
- 51 - 55 : Customers in this age group have the highest spending per transaction

8.2 Report

a. Report whether the confidence intervals for the average amount spent by males and females (computed using all the data) overlap. How can Walmart leverage this conclusion to make changes or improvements? * From the above analysis, we can see that the confidence interval overlap for some of the age groups.

i.e - 18 - 25, 26 - 35, 46 - 50 : Customers in these age groups have overlapping confidence intervals indicating similar buying characteristics

- Identify the 0 - 17 age group as having the lowest spending per transaction. Offer more attractive discounts, coupons, or rewards programs to incentivize higher spending.Tailor product selection and marketing strategies to align with the preferences and needs of this age group.
- Recognize similar buying characteristics among customers in the 18 - 25, 26 - 35, and 46 - 50 age groups, as well as among those in the 36 - 45 and 55+ age groups. Optimize product selection to cater to the preferences of these age groups.

Insights: - The Median purchase amount for Female is around 7500 for both married and Unmarried - The Median purchase amount for Male is around 7500-7750 for both married and Unmarried - The Variance is slightly more for Males purchases - Product Categories 1,5,8,11 are mostly preferred by males - Product Categories 5,8,1,3 are mostly preferred by females - 26-35 age group occupies large portion of purchase amount with 0-17 being the least. - Unmarried customer occupy 59% of portion in purchasing. - The heatmap visualization offers a clearer representation of the average spending patterns across various age groups and marital status categories.

- The lack of significant variation in spending amounts suggests that age and marital status may not strongly influence the amount spent.

8.3 Recomendations

a. Write a detailed recommendation from the analysis that you have done. - Since male customers occupy most portion of black friday sales ,walmart can tailer made marketing strategies and product offereing for males

- With the age group between 26 and 45 contributing to the majority of sales, plan exclusive deals to this demographics to maximize sales.
- Since shoppers aged 18-25, 26-35, and 46-50 share similar buying habits, as do those aged 36-45 and 55+, Walmart can tweak its product offerings to suit their preferences. Additionally, adjusting pricing strategies accordingly can maximize profits.
-

[ ]:

```

```