# Walmart_Case_Study

May 5, 2024

**BUSINESS CASE STUDY : 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.

## 1 Importing all the libraries for analyzing the case study

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

## 2 1.Defining Problem Statement and Analyzing basic metrics

### Problem Statement

The Management team in the company 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).

[6]:
```
df=pd.read_csv('walmart_data.CSV')
df
```

[6]:
```
        User_ID Product_ID Gender   Age  Occupation City_Category  \
0       1000001  P00069042      F  0-17        10.0             A
1       1000001  P00248942      F  0-17        10.0             A
2       1000001  P00087842      F  0-17        10.0             A
```

```
3      1000001  P00085442      F   0-17        10.0            A
4      1000002  P00285442      M    55+        16.0            C
…         …        …     …    …       …          …
75128  1005575  P00176942      M  36-45         7.0            C
75129  1005575  P00189242      M  36-45         7.0            C
75130  1005575  P00077942      M  36-45         7.0            C
75131  1005575  P00258842      M  36-45         7.0            C
75132  1005575    P002163    NaN    NaN         NaN          NaN

      Stay_In_Current_City_Years  Marital_Status  Product_Category  Purchase
0                              2             0.0               3.0    8370.0
1                              2             0.0               1.0   15200.0
2                              2             0.0              12.0    1422.0
3                              2             0.0              12.0    1057.0
4                             4+             0.0               8.0    7969.0
…                          …             …               …       …
75128                          2             0.0               5.0    1972.0
75129                          2             0.0               5.0    5425.0
75130                          2             0.0               5.0    5198.0
75131                          2             0.0               5.0    5430.0
75132                        NaN             NaN               NaN       NaN

[75133 rows x 10 columns]
```

**Observations on shape of data**

[ ]:
```
df.shape
```

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

**Data types of all the attributes**

[ ]:
```
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
```

**Null values in the given dataframe**

```
[ ]: df.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
```

**conversion of categorical attributes to 'category' (If required)**

converting categorical attributes to the 'category' data type, when appropriate, can lead to improved memory usage, faster computations, and better performance of analytical techniques

```
[ ]: columns=['Occupation','Marital_Status','Product_Category']
     df[columns]=df[columns].astype('object')
     df.dtypes
```

```
[ ]: User_ID                        int64
     Product_ID                    object
     Gender                        object
     Age                           object
     Occupation                    object
     City_Category                 object
     Stay_In_Current_City_Years    object
     Marital_Status                object
     Product_Category              object
     Purchase                       int64
     dtype: object
```

**statistical summary**

```
[ ]: df.describe(include='all')
```

```
[ ]:             User_ID Product_ID  Gender     Age  Occupation City_Category  \
     count   5.500680e+05     550068  550068  550068    550068.0        550068
     unique           NaN       3631       2       7        21.0             3
     top              NaN  P00265242       M   26-35         4.0             B
```

```
freq             NaN        1880  414259  219587     72308.0         231173
mean    1.003029e+06         NaN     NaN     NaN         NaN            NaN
std     1.727592e+03         NaN     NaN     NaN         NaN            NaN
min     1.000001e+06         NaN     NaN     NaN         NaN            NaN
25%     1.001516e+06         NaN     NaN     NaN         NaN            NaN
50%     1.003077e+06         NaN     NaN     NaN         NaN            NaN
75%     1.004478e+06         NaN     NaN     NaN         NaN            NaN
max     1.006040e+06         NaN     NaN     NaN         NaN            NaN

       Stay_In_Current_City_Years  Marital_Status  Product_Category  \
count                      550068        550068.0          550068.0
unique                          5             2.0              20.0
top                             1             0.0               5.0
freq                       193821        324731.0          150933.0
mean                          NaN             NaN               NaN
std                           NaN             NaN               NaN
min                           NaN             NaN               NaN
25%                           NaN             NaN               NaN
50%                           NaN             NaN               NaN
75%                           NaN             NaN               NaN
max                           NaN             NaN               NaN

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

** Observation from above table:**

1) The top people purchasing are in the age range of 26–35.
2) Males are top in purchasing
3) The average purchase is 9263.96 and the maximum purchase is 23961, so the average value is sensitive to outliers, but the fact that the mean is so small compared to the maximum value indicates the maximum value is an outlier.

## 3 Non-Graphical Analysis: Value counts and unique attributes

**Value counts**

```
[ ]: gender_counts = df['Gender'].value_counts()
     percentage_gender_counts = (gender_counts / len(df)) * 100
     print(f"Gender count : \n{gender_counts} \nGender percentage :␣
      ↪\n{percentage_gender_counts}")
```

```
Gender count :
Gender
M    414259
F    135809
Name: count, dtype: int64
Gender percentage :
Gender
M    75.310507
F    24.689493
Name: count, dtype: float64
```

```
[ ]: Age_counts = df['Age'].value_counts()
     percentage_Age_counts = (Age_counts / len(df)) * 100
     print(f"Age count : \n{Age_counts} \nAge percentage :␣
      ↪\n{percentage_Age_counts}")
```

```
Age count :
Age
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: count, dtype: int64
Age percentage :
Age
26-35    39.919974
36-45    19.999891
18-25    18.117760
46-50     8.308246
51-55     6.999316
55+       3.909335
0-17      2.745479
Name: count, dtype: float64
```

```
[ ]: Stay_In_Current_City_Years_counts = df['Stay_In_Current_City_Years'].
      ↪value_counts()
     percentage_Stay_In_Current_City_Years_counts =␣
      ↪(Stay_In_Current_City_Years_counts / len(df)) * 100
```

```
print(f"Stay_In_Current_City_Years count : \n{␣
↪Stay_In_Current_City_Years_counts}
      \nStay_In_Current_City_Years percentage :␣
↪\n{percentage_Stay_In_Current_City_Years_counts}")
```

```
Stay_In_Current_City_Years count :
Stay_In_Current_City_Years
1     193821
2     101838
3      95285
4+     84726
0      74398
Name: count, dtype: int64
Stay_In_Current_City_Years percentage :
Stay_In_Current_City_Years
1     35.235825
2     18.513711
3     17.322404
4+    15.402823
0     13.525237
Name: count, dtype: float64
```

[ ]:
```
Marital_Status_counts = df['Marital_Status'].value_counts()
percentage_Marital_Status_counts = (Marital_Status_counts / len(df)) * 100
print(f"Marital_Status count : \n{Marital_Status_counts} \nMarital_Status␣
↪percentage :\n{percentage_Marital_Status_counts}")
```

```
Marital_Status count :
Marital_Status
0    324731
1    225337
Name: count, dtype: int64
Marital_Status percentage :
Marital_Status
0    59.034701
1    40.965299
Name: count, dtype: float64
```

**Insights**

1) 75% of users are male and 25% are female.
2) Users ages 26–35 are 40%, users ages 36–45 are 20%, users ages 18–25 are 18%, and very low users ages ( 0–17 & 55+ )are 5%.
3) 35% stay in a city for 1 year, 18% stay in a city for 2 years, 17% stay in a city for 3 years, and 15% stay in a city for 4+ years.
4) 60% of users are single, and 40% are married.

**Unique attributes :**

```
[ ]: unique_category_count = df['Product_Category'].nunique()
     print('Unique Product_Category count:',unique_category_count)
```

```
Unique Product_Category count: 20
```

```
[ ]: unique_City_Category_count = df['City_Category'].nunique()
     print('Unique City_Category count:',unique_City_Category_count)
```

```
Unique City_Category count: 3
```

```
[ ]: unique_Product_ID_count = df['Product_ID'].nunique()
     print('Unique Product_ID count:',unique_Product_ID_count)
```

```
Unique Product_ID count: 3631
```

```
[ ]: unique_User_ID_count = df['User_ID'].nunique()
     print('Unique User_ID count:',unique_User_ID_count)
```

```
Unique User_ID count: 5891
```

## 4 Insights :

1) The total product category count is 20 unique products.
2) The total number of unique city categories is three.
3) The total number of unique product IDs is 3631.
4) The total number of unique user IDs is 5891

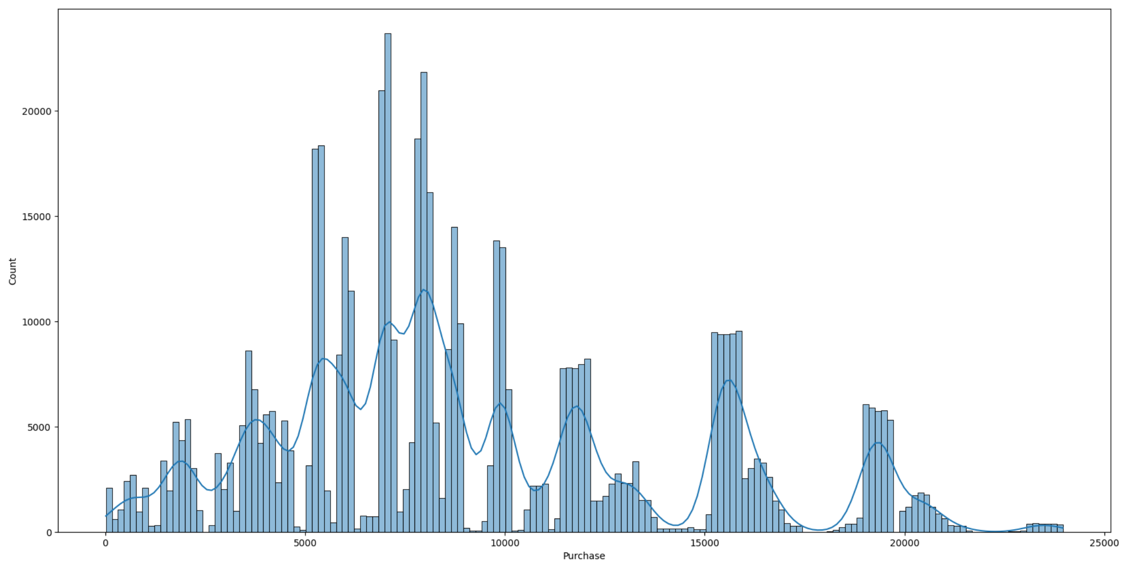
## 5 Visual Analysis - Univariate & Bivariate

Visual analysis of data is a crucial step in exploring and understanding your datasets. Python provides several powerful libraries for creating visualizations, with matplotlib and seaborn being two of the most popular ones.Here I have used various plots like - Line plot - Box plot - Heatmap - Pie Chart - Bar Chart

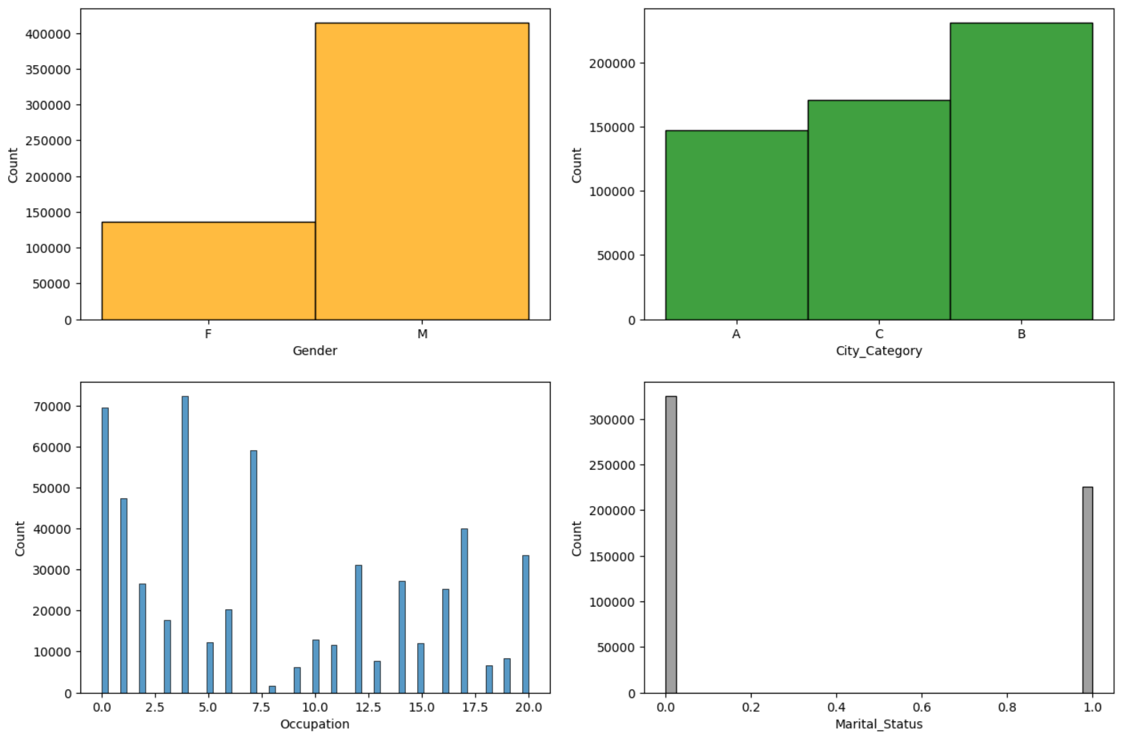
## 6 Univariate

Univariate analysis is a statistical method that involves the analysis of a single variable at a time. It aims to understand the distribution, central tendency, dispersion, and other properties of a single variable without considering its relationship with other variables

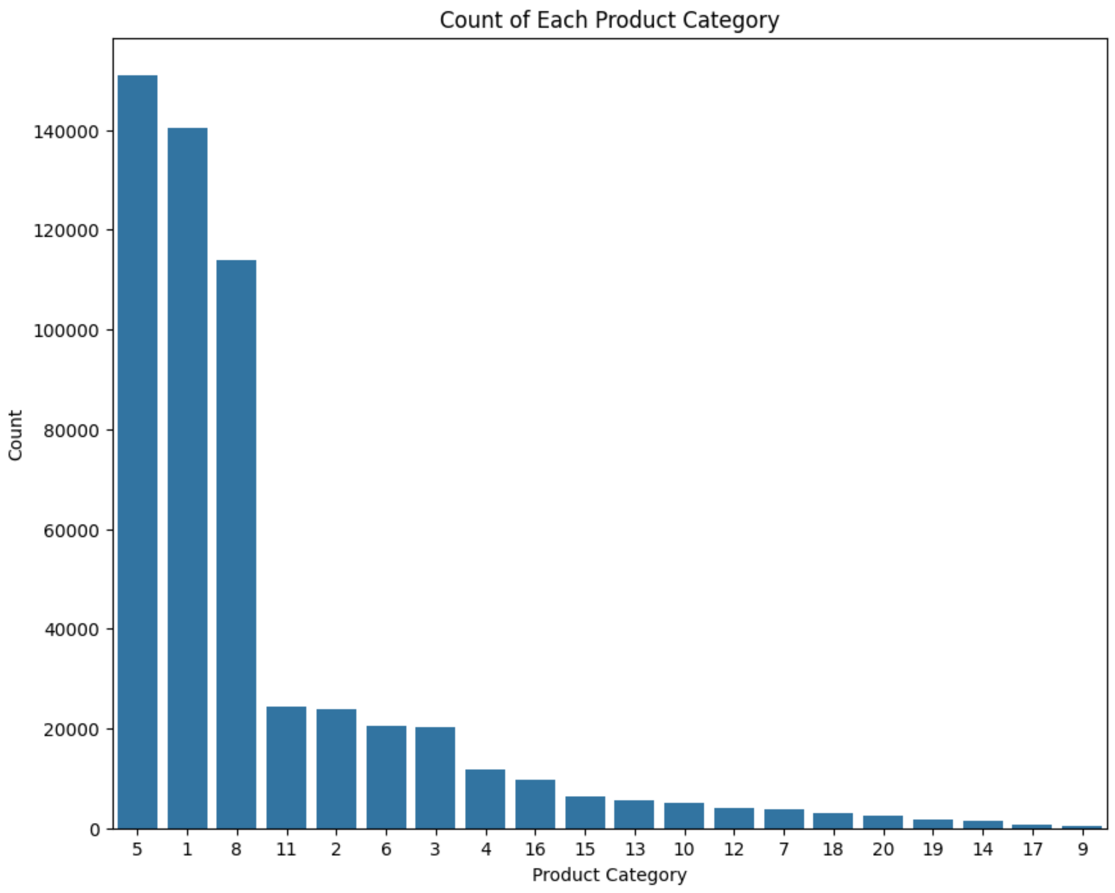
```
[ ]: plt.figure(figsize=(20,10))
     sns.histplot(data=df, x='Purchase', kde=True)
     plt.show()
```



[ ]:
```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(15,10))
sns.histplot(data=df, x='Gender', ax=axis[0,0],color = "orange")
sns.histplot(data=df, x='City_Category', ax=axis[0,1],color = "green")
sns.histplot(data=df, x='Occupation', ax=axis[1,0])
sns.histplot(data=df, x='Marital_Status',ax=axis[1,1],color = "grey")
plt.show()
```



```
[ ]: plt.figure(figsize=(10, 8))

     sns.countplot(data=df, x='Product_Category', order=df['Product_Category'].
      ↪value_counts().index)
     plt.xlabel('Product Category')
     plt.ylabel('Count')
     plt.title('Count of Each Product Category')
     plt.show()
```

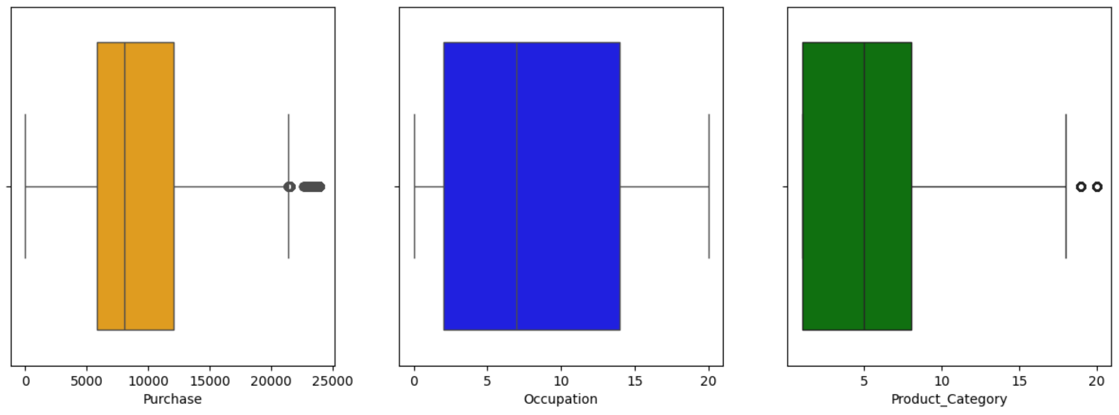


**Insights:**

1) The product categories 5, 1, and 8 have the highest purchase.
2) Male purchasing power outnumbers female purchasing power.
3) More users belongs in the B city region.
4) Max users are single.
5) The maximum purchase ranges from 5000 to 15000.

**#Outliers detection using BoxPlots:**

```
[ ]: fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(15,2))
     fig.subplots_adjust(top=2)
     sns.boxplot(data=df, x='Purchase', ax=axis[0],color = "orange")
     sns.boxplot(data=df, x='Occupation', ax=axis[1],color = "blue")
     sns.boxplot(data=df, x='Product_Category', ax=axis[2],color = "GREEN")
     plt.show()
```



**Insights:**

1) Purchases have outliers.
2) The occupation does not have any outliers.
3) Product categories have some outliers, but most of the products are purchased in the range 1 to 8.

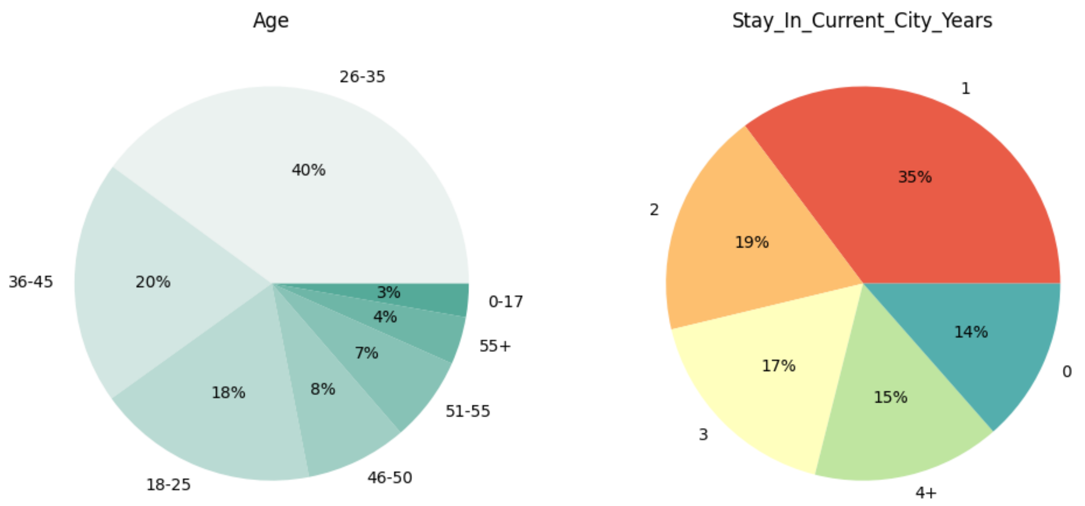
## 7 Using pie chart:

Analyzing the variation in purchases with the following,

1. Gender vs Purchase
2. Martial_Status vs Purchase
3. Age vs Purchase
4. City_Category vs Purchase

```
[ ]: unique_colors_age = sns.color_palette("light:#5A9", len(df['Age'].unique()))
     unique_colors_city_years = sns.color_palette("Spectral",␣
      ↪len(df['Stay_In_Current_City_Years'].unique()))

     fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(12, 8))

     data_age = df['Age'].value_counts(normalize=True) * 100
```

```
axs[0].pie(x=data_age.values, labels=data_age.index, autopct='%.0f%%',␣
  ↪colors=unique_colors_age)
axs[0].set_title("Age")

data_city_years = df['Stay_In_Current_City_Years'].value_counts(normalize=True)␣
  ↪* 100
axs[1].pie(x=data_city_years.values, labels=data_city_years.index, autopct='%.
  ↪0f%%', colors=unique_colors_city_years)
axs[1].set_title("Stay_In_Current_City_Years")
plt.show()
```



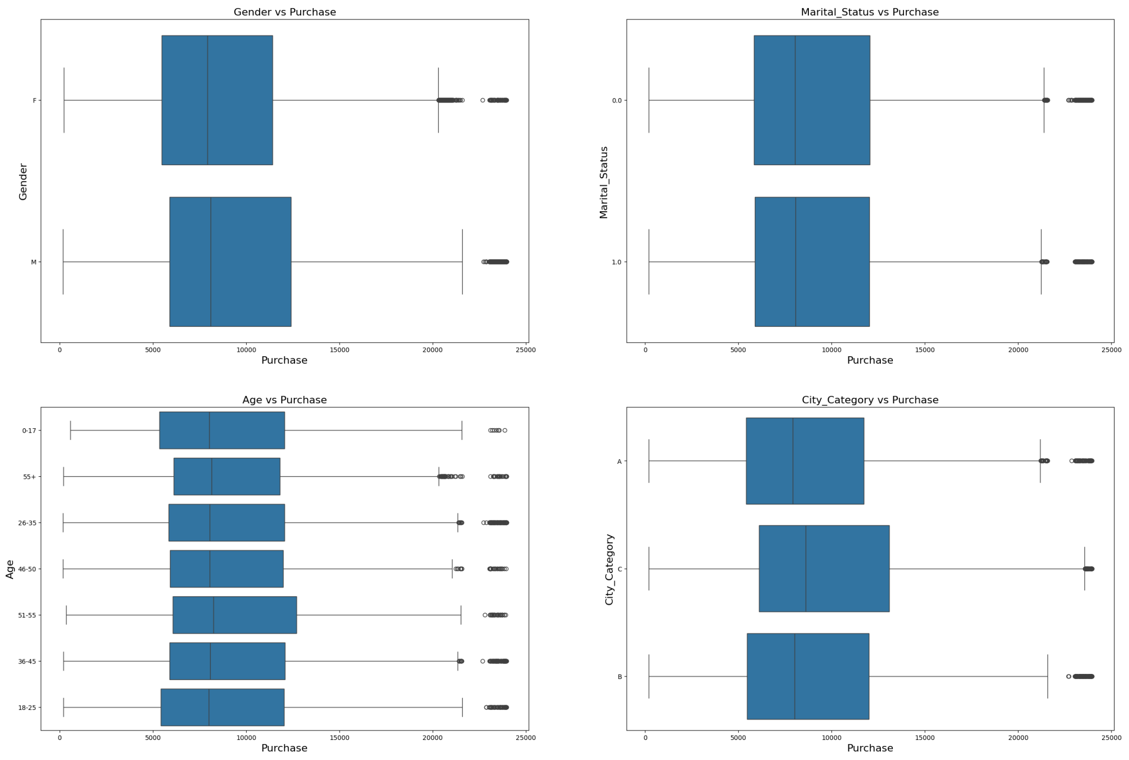
**Insights :** 1) Users ages 26–35 are 40%, users ages 36–45 are 20%, users ages 18–25 are 18%, users ages 46–50 are 8%, users ages 51–55 are 7%, users ages 55+ are 4%, and very low users ages 0–17 are 2%.

2) 35% stay in a city for 1 year, 19% stay in a city for 2 years, 17% stay in a city for 3 years, and 15% stay in a city for 4+ years.

## 8 Bivariate Analysis:

[8]:
```
fig1, axs=plt.subplots(nrows=2,ncols=2, figsize=(30,20))

sns.boxplot(data=df, y='Gender',x ='Purchase',orient='h',ax=axs[0,0])
axs[0,0].set_title("Gender vs Purchase", fontsize=16)
axs[0,0].set_xlabel("Purchase", fontsize=16)
axs[0,0].set_ylabel("Gender", fontsize=16)

sns.boxplot(data=df, y='Marital_Status',x ='Purchase',orient='h',ax=axs[0,1])
axs[0,1].set_title("Marital_Status vs Purchase", fontsize=16)
```

```
axs[0,1].set_xlabel("Purchase", fontsize=16)
axs[0,1].set_ylabel("Marital_Status", fontsize=16)

sns.boxplot(data=df, y='Age',x ='Purchase',orient='h',ax=axs[1,0])
axs[1,0].set_title("Age vs Purchase", fontsize=16)
axs[1,0].set_xlabel("Purchase", fontsize=16)
axs[1,0].set_ylabel("Age", fontsize=16)

sns.boxplot(data=df, y='City_Category',x ='Purchase',orient='h',ax=axs[1,1])
axs[1,1].set_title("City_Category vs Purchase", fontsize=16)
axs[1,1].set_xlabel("Purchase", fontsize=16)
axs[1,1].set_ylabel("City_Category", fontsize=16)
plt.show()
```



**insight**

1) **Gender vs. Purchase**

    a) The median for males and females is almost equal.

    b) Females have more outliers compared to males.

    c) Males purchased more compared to females.

2) **Martial Status vs. Purchase**

    a) The median for married and single people is almost equal.
    b) Outliers are present in both records.

3) **Age vs. Purchase**
    a) The median for all age groups is almost equal.
    b) Outliers are present in all age groups.

4) **City Category vs. Purchase**
    a) The C city region has very low outliers compared to other cities.
    b) A and B city region medians are almost the same.

## 9 Using pandas quantile funtion detecting number of outliers from purchase

[9]:
```
q1 = df["Purchase"].quantile(0.25)
q3 = df["Purchase"].quantile(0.75)
IQR = q3-q1
outliers = df["Purchase"][((df["Purchase"]<(q1-1.5*IQR)) |␣
 ↪(df["Purchase"]>(q3+1.5*IQR)))]
print("number of outliers: "+ str(len(outliers)))
print("max outlier value:"+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))
```

```
number of outliers: 381
max outlier value:23958.0
min outlier value: 21314.0
```

#4. How does gender affect the amount spent?

## 10 Are women spending more money per transaction than men? Why or Why not?

[10]:
```
avg_by_gender = df.groupby('Gender')['Purchase'].mean()
print(f'Average purchase of male and female : \n{avg_by_gender}')
```

```
Average purchase of male and female :
Gender
F    8780.111166
M    9464.059671
Name: Purchase, dtype: float64
```

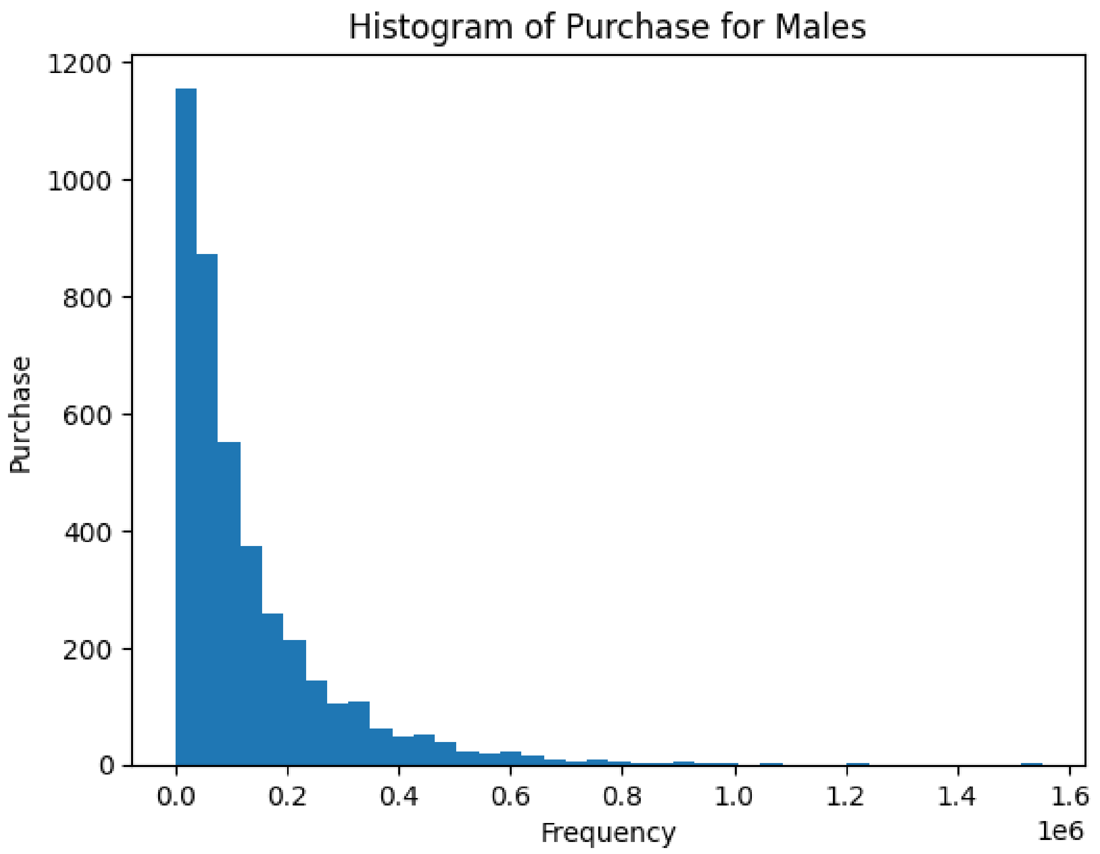
[11]:
```
agg_df = df.groupby(['User_ID', 'Gender'])[['Purchase']].agg({'Purchase':␣
 ↪['sum', 'mean']})
agg_df = agg_df.reset_index()
agg_df = agg_df.sort_values(by='User_ID', ascending=False)

print(f"Top 10 purchase from male and female\n{agg_df.head(10)}")
```

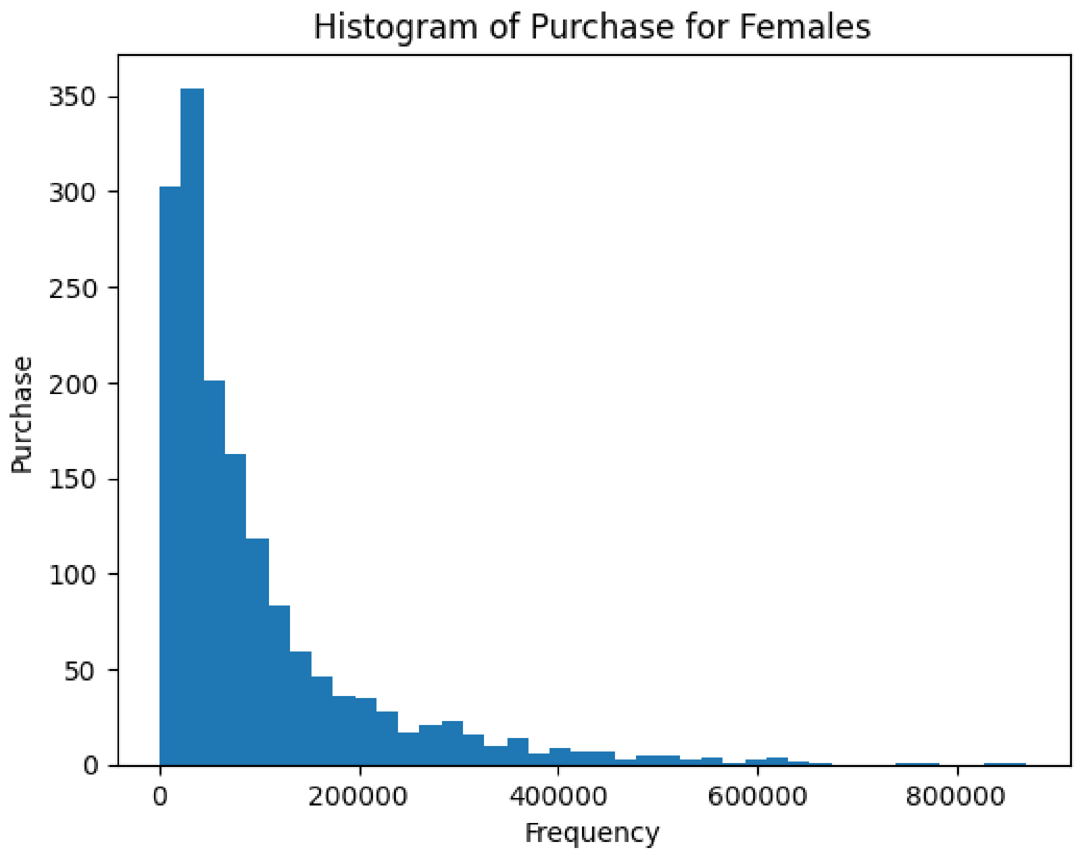
```
Top 10 purchase from male and female
      User_ID Gender  Purchase
                           sum          mean
5691  1006040      M   95780.0   8707.272727
5690  1006039      F   50364.0   6295.500000
5689  1006037      F   86597.0  10824.625000
5688  1006036      F  196339.0   6770.310345
5687  1006035      F   42357.0   8471.400000
5686  1006034      M   40575.0  20287.500000
5685  1006033      M   65827.0  16456.750000
5684  1006032      M   27517.0   6879.250000
5683  1006031      F   31748.0   6349.600000
5682  1006030      M   41033.0  10258.250000
```

[12]:
```
Gender_wise_count=agg_df['Gender'].value_counts()
print(f'Each gender wise count : \n{Gender_wise_count}')
```

```
Each gender wise count :
Gender
M    4102
F    1590
Name: count, dtype: int64
```

[13]:
```
sum_by_gender = df.groupby(['User_ID', 'Gender'])['Purchase'].sum()
sum_by_gender = sum_by_gender.reset_index()
sum_by_gender = sum_by_gender.sort_values(by='User_ID', ascending=False)

# MALE data representation through a histogram
male_data = sum_by_gender[sum_by_gender['Gender']=='M']['Purchase']
plt.hist(male_data, bins=40)
plt.ylabel('Purchase')
plt.xlabel('Frequency')
plt.title('Histogram of Purchase for Males')
plt.show()

# FEMALE data representation through a histogram
Female_data = sum_by_gender[sum_by_gender['Gender']=='F']['Purchase']
plt.hist(Female_data, bins=40)
plt.ylabel('Purchase')
plt.xlabel('Frequency')
plt.title('Histogram of Purchase for Females')
plt.show()
```


Histogram of Purchase for Males
Purchase
1200
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Frequency
1e6



[14]:
```
Mean_by_gender = df.groupby(['User_ID', 'Gender'])['Purchase'].sum()
Mean_by_gender = Mean_by_gender.reset_index()
Mean_by_gender = Mean_by_gender.sort_values(by='User_ID', ascending=False)
Male_cust_avg = Mean_by_gender[Mean_by_gender['Gender']=='M']['Purchase'].mean()
Female_cust_avg = Mean_by_gender[Mean_by_gender['Gender']=='F']['Purchase'].
 ↪mean()
print(f'Male customer average spent amount: {Male_cust_avg}')
print(f'Female customer average spent amount: {Female_cust_avg}')
```

```
Male customer average spent amount: 131460.9107752316
Female customer average spent amount: 100242.36352201257
```

## 11 insight

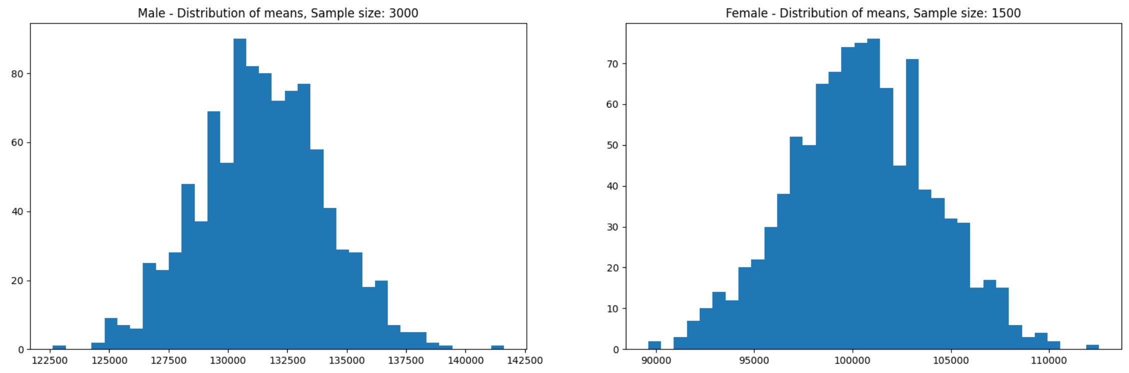
1) Male customers spend more money than female customers.
2) The highest purchase has been made from this user id: 1006040, and the gender is male.
3) Most of the females also purchase, but they don't spend a lot more.

## 12 Confidence intervals and distribution of the mean of the expenses by female and male customers.

[15]:
```
# filtering gender wise dataframe
male_df = sum_by_gender[sum_by_gender['Gender']=='M']
female_df = sum_by_gender[sum_by_gender['Gender']=='F']

# Taking random sample size from dataframe
male_sample_size = 3000
female_sample_size = 1000
num_repitions = 1000

# Taking random sample from male and female dataframe
random_sample_male = male_df.sample(n=male_sample_size)
random_sample_female = female_df.sample(n=female_sample_size)

# Taking mean value from random sample male and female dataframe
male_means = random_sample_male['Purchase'].mean()
print(f'Population mean: random male samples mean purchase value: {male_means}')
female_means = random_sample_female['Purchase'].mean()
print(f'Population mean: random Female samples mean purchase value :␣
 ↪{female_means}')

# Taking sample mean from filtered male dataframe
Male_sample_mean = round(male_df['Purchase'].mean(),2)
print(f'Sample means of Male purchase : {Male_sample_mean}')
Male_std_value = round(male_df['Purchase'].std(),2)
print(f'Sample STD of Male purchase : {Male_std_value}')

# Taking sample mean from filtered female dataframe
Female_sample_mean = round(female_df['Purchase'].mean(),2)
print(f'Sample means of Female purchase : {Female_sample_mean}')
Female_std_value = round(female_df['Purchase'].std(),2)
print(f'Sample STD of Female purchase : {Female_std_value}')

# taking blank list to creat histogram
male_means1 = []
female_means1 = []

# using for loop to create again mean value for histogram
for _ in range(num_repitions):
    male_mean2 = male_df.sample(male_sample_size,replace=True)['Purchase'].
 ↪mean()
    female_mean2 = female_df.
 ↪sample(female_sample_size,replace=True)['Purchase'].mean()
    male_means1.append(male_mean2)
    female_means1.append(female_mean2)
```

```
# making histogram to check visually distribution mean for male and female
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))
axis[0].hist(male_means1, bins=35)
axis[1].hist(female_means1, bins=35)
axis[0].set_title("Male - Distribution of means, Sample size: 3000")
axis[1].set_title("Female - Distribution of means, Sample size: 1500")
plt.show()
```

```
Population mean: random male samples mean purchase value: 133134.58833333335
Population mean: random Female samples mean purchase value : 97064.698
Sample means of Male purchase : 131460.91
Sample STD of Male purchase : 145367.25
Sample means of Female purchase : 100242.36
Sample STD of Female purchase : 115709.32
```



## 13 Insight

1) The average amount spent by male customers is 925344.4.
2) The average amount spent by female customers is 712024.

39.

3) Male customers have made more purchases than female customers.

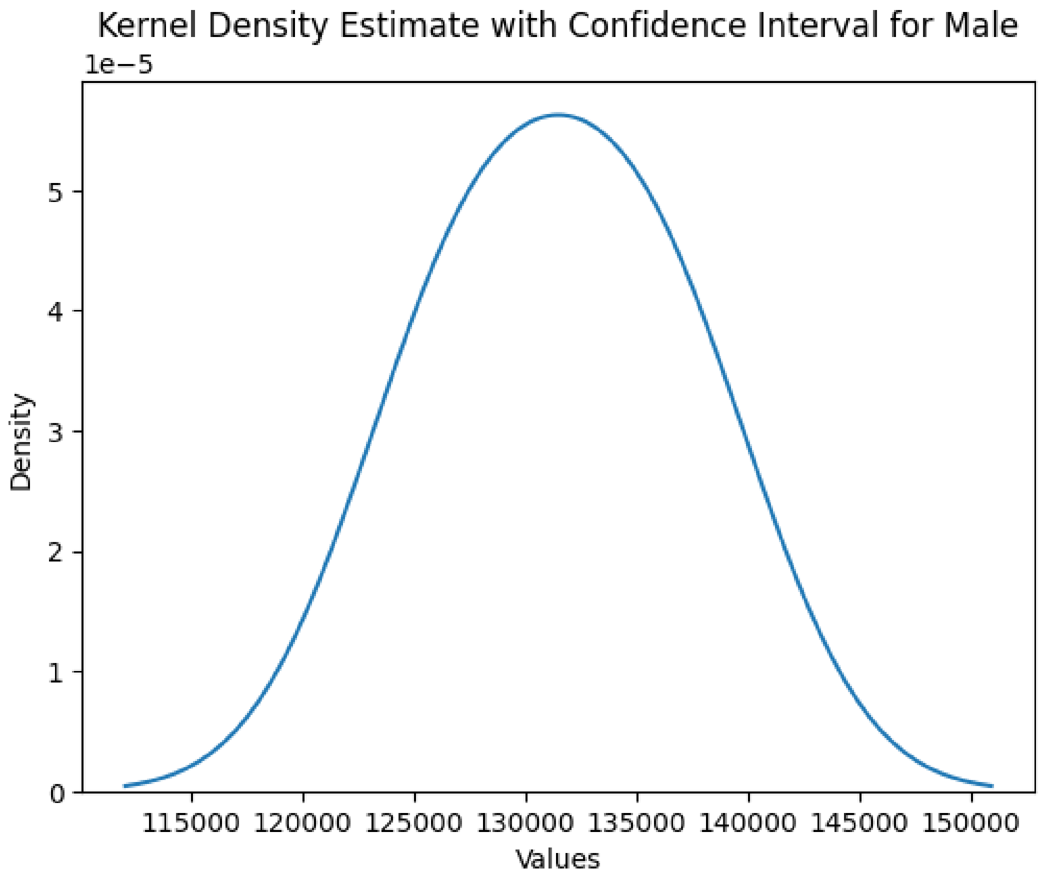
## 14 Are confidence intervals of average male and female spending overlapping? How can company leverage this conclusion to make changes or improvements?

[16]:
```
#sample size
sample_size = 3000
# Confidence level ( 95% confidence interval)
```

```
confidence_level = 0.95
# Calculate the margin of error using the z-distribution for male
z_critical = stats.norm.ppf((1 + confidence_level) / 2)
margin_of_error = z_critical * (Male_std_value / np.sqrt(sample_size))
# Calculate the margin of error using the z-distribution for female
z_critical = stats.norm.ppf((1 + confidence_level) / 2)
margin_of_error = z_critical * (Female_std_value / np.sqrt(sample_size))
```

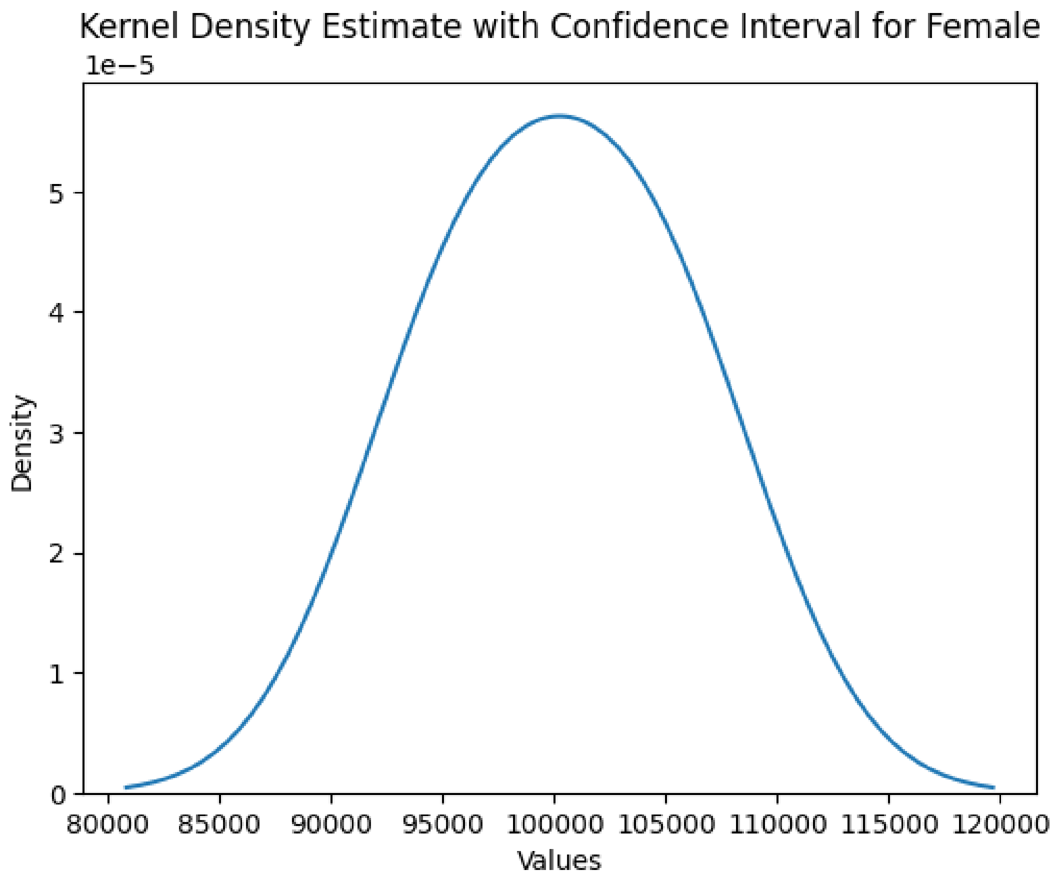
[17]:
```
# Calculate the confidence interval for male and presenting it on the graph
Male_confidence_interval = (Male_sample_mean - margin_of_error,␣
 ↪Male_sample_mean + margin_of_error)
print("Confidence Interval 95% Male:", Male_confidence_interval)
sns.kdeplot(Male_confidence_interval)
plt.xlabel('Values')
plt.ylabel('Density')
plt.title('Kernel Density Estimate with Confidence Interval for Male')
plt.show()
```

```
Confidence Interval 95% Male: (127320.38124565012, 135601.4387543499)
```



```
[18]: # Calculate the confidence interval for female and presenting it on the graph
Female_confidence_interval = (Female_sample_mean - margin_of_error,␣
 ↪Female_sample_mean + margin_of_error)
print("Confidence Interval 95% Female:", Female_confidence_interval)
sns.kdeplot(Female_confidence_interval)
plt.xlabel('Values')
plt.ylabel('Density')
plt.title('Kernel Density Estimate with Confidence Interval for Female')
plt.show()
```

```
Confidence Interval 95% Female: (96101.83124565012, 104382.88875434989)
```



## 15 Insight

1) With reference to the above data, at a 95% confidence interval:
    a) The average amount spent by male customers will lie between 896453.54 and 954235.25.
    b) The average amount spent by female customers will lie between 683133.53 and 740915.24.

2) Confidence intervals for average male and female spending are not overlapping.
3) With respect to the above data, company should target more male customers, as they spend a lot compared to females.

#5. How does Marital_Status affect the amount spent?

## 16 Results when the same activity is performed for Married vs Unmarried

[19]:
```
sum_by_Marital_Status = df.groupby(['User_ID', 'Marital_Status'])['Purchase'].
  ↪sum()
sum_by_Marital_Status = sum_by_Marital_Status.reset_index()
sum_by_Marital_Status = sum_by_Marital_Status.sort_values(by='User_ID',␣
  ↪ascending=False)
Married_cust_avg =␣
  ↪sum_by_Marital_Status[sum_by_Marital_Status['Marital_Status']==1]['Purchase'].
  ↪mean()
print(f'Married customer average spent amount: {Married_cust_avg}')
```

```
Married customer average spent amount: 119107.68117154812
```

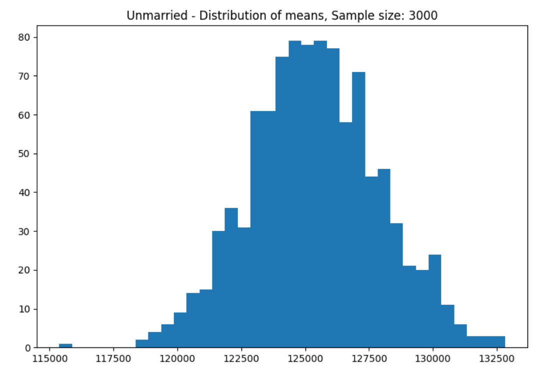
[20]:
```
sum_by_Marital_Status = df.groupby(['User_ID', 'Marital_Status'])['Purchase'].
  ↪sum()
sum_by_Marital_Status = sum_by_Marital_Status.reset_index()
sum_by_Marital_Status = sum_by_Marital_Status.sort_values(by='User_ID',␣
  ↪ascending=False)
Unmarried_cust_avg =␣
  ↪sum_by_Marital_Status[sum_by_Marital_Status['Marital_Status']==0]['Purchase'].
  ↪mean()
print(f'Unmarried customer average spent amount: {Unmarried_cust_avg}')
```

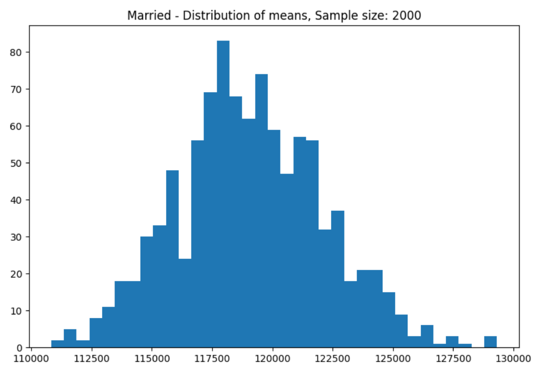
```
Unmarried customer average spent amount: 125369.67171411266
```

[21]:
```
# filtering Marital Status wise dataframe
Unmarried_df = sum_by_Marital_Status[sum_by_Marital_Status['Marital_Status']==0]
Married_df = sum_by_Marital_Status[sum_by_Marital_Status['Marital_Status']==1]

# Taking random sample size from dataframe
Unmarried_sample_size = 3000
Married_sample_size = 2000
num_repitions = 1000

# Taking random sample from unmarried and married dataframe
random_sample_Unmarried = Unmarried_df.sample(n=Unmarried_sample_size)
random_sample_Married = Married_df.sample(n=Married_sample_size)
```

```
# Taking mean value from random sample unmarried and married dataframe
Unmarried_means = random_sample_Unmarried['Purchase'].mean()
print(f'Population mean: random Unmarried samples mean purchase value:␣
  ↪{Unmarried_means}')
Married_means = random_sample_Married['Purchase'].mean()
print(f'Population mean: random Married samples mean purchase value :␣
  ↪{Married_means}')

# Taking sample mean from filtered unmarried dataframe
Unmarried_sample_mean = round(Unmarried_df['Purchase'].mean(),2)
print(f'Sample means of Unmarried purchase : {Unmarried_sample_mean}')
Unmarried_std_value = round(Unmarried_df['Purchase'].std(),2)
print(f'Sample STD of Unmarried purchase : {Unmarried_std_value}')

# Taking sample mean from filtered Married dataframe
Married_sample_mean = round(Married_df['Purchase'].mean(),2)
print(f'Sample means of Married purchase : {Married_sample_mean}')
Married_std_value = round(Married_df['Purchase'].std(),2)
print(f'Sample STD of Married purchase : {Married_std_value}')

# taking blank list to creat histogram
Unmarried_means1 = []
Married_means1 = []

# using for loop to create again mean value for histogram
for _ in range(num_repitions):
    Unmarried_mean2 = Unmarried_df.
  ↪sample(Unmarried_sample_size,replace=True)['Purchase'].mean()
    Married_mean2 = Married_df.
  ↪sample(Married_sample_size,replace=True)['Purchase'].mean()
    Unmarried_means1.append(Unmarried_mean2)
    Married_means1.append(Married_mean2)

# # making histogram to check visually distribution mean for Unmarried and␣
  ↪Married
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))
axis[0].hist(Unmarried_means1, bins=35)
axis[1].hist(Married_means1, bins=35)
axis[0].set_title("Unmarried - Distribution of means, Sample size: 3000")
axis[1].set_title("Married - Distribution of means, Sample size: 2000")
plt.show()
```

```
Population mean: random Unmarried samples mean purchase value: 125466.231
Population mean: random Married samples mean purchase value : 119763.7
Sample means of Unmarried purchase : 125369.67
Sample STD of Unmarried purchase : 140498.9
Sample means of Married purchase : 119107.68
Sample STD of Married purchase : 135459.22
```





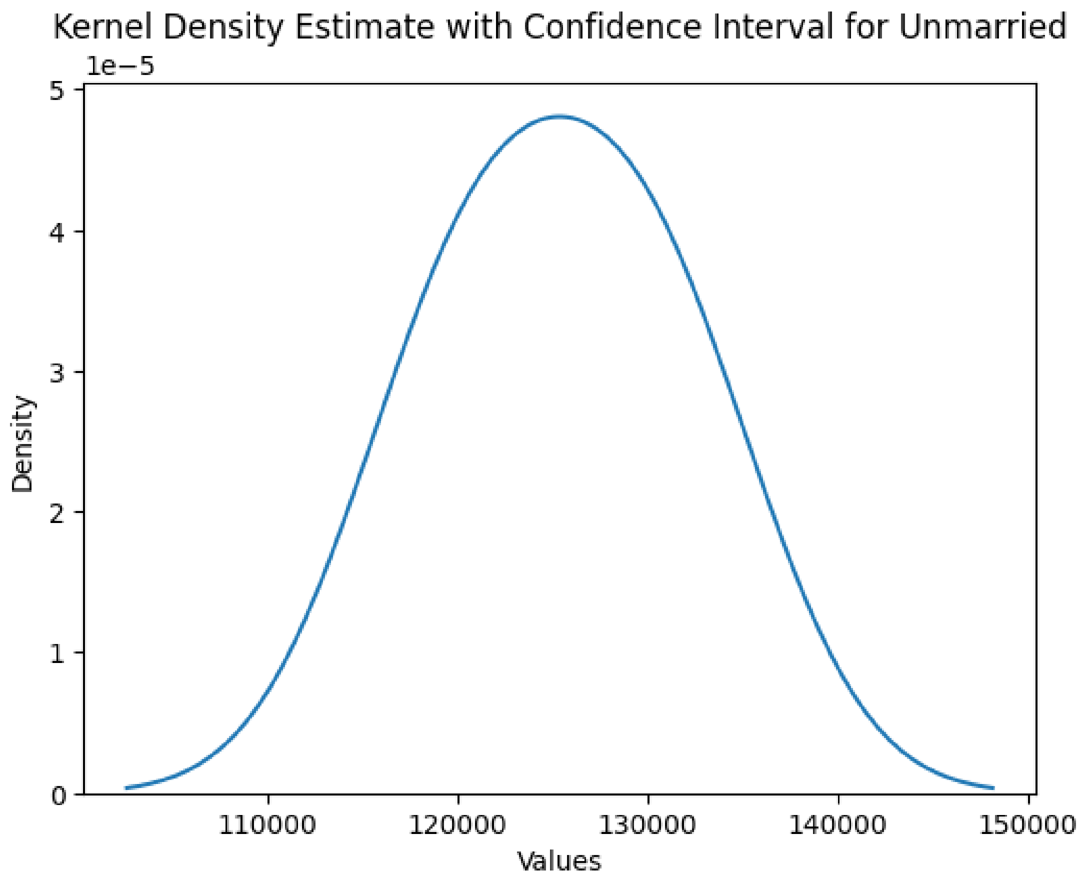
## 17 Insight

1) Unmarried customer average sent amount: 880575.7819724905
2) Married customer average sent amount: 843526.7966855295
3) Unmarried customers spend more than married customers.

[22]:
```
#sample size
sample_size = 3000
# Confidence level ( 95% confidence interval)
confidence_level = 0.95
# Calculate the margin of error using the z-distribution for male
z_critical = stats.norm.ppf((1 + confidence_level) / 2)  # Z-score for the␣
 ↪desired confidence level
margin_of_error = z_critical * (Unmarried_std_value / np.sqrt(sample_size))
# Calculate the margin of error using the z-distribution for female
z_critical = stats.norm.ppf((1 + confidence_level) / 2)  # Z-score for the␣
 ↪desired confidence level
margin_of_error = z_critical * (Married_std_value / np.sqrt(sample_size))
```

[23]:
```
# Calculate the confidence interval for Unmarried and presenting it on the graph
Unmarried_confidence_interval = (Unmarried_sample_mean - margin_of_error,␣
 ↪Unmarried_sample_mean + margin_of_error)
print("Confidence Interval 95% Unmarried:", Unmarried_confidence_interval)
sns.kdeplot(Unmarried_confidence_interval)
plt.xlabel('Values')
plt.ylabel('Density')
plt.title('Kernel Density Estimate with Confidence Interval for Unmarried')
plt.show()
```

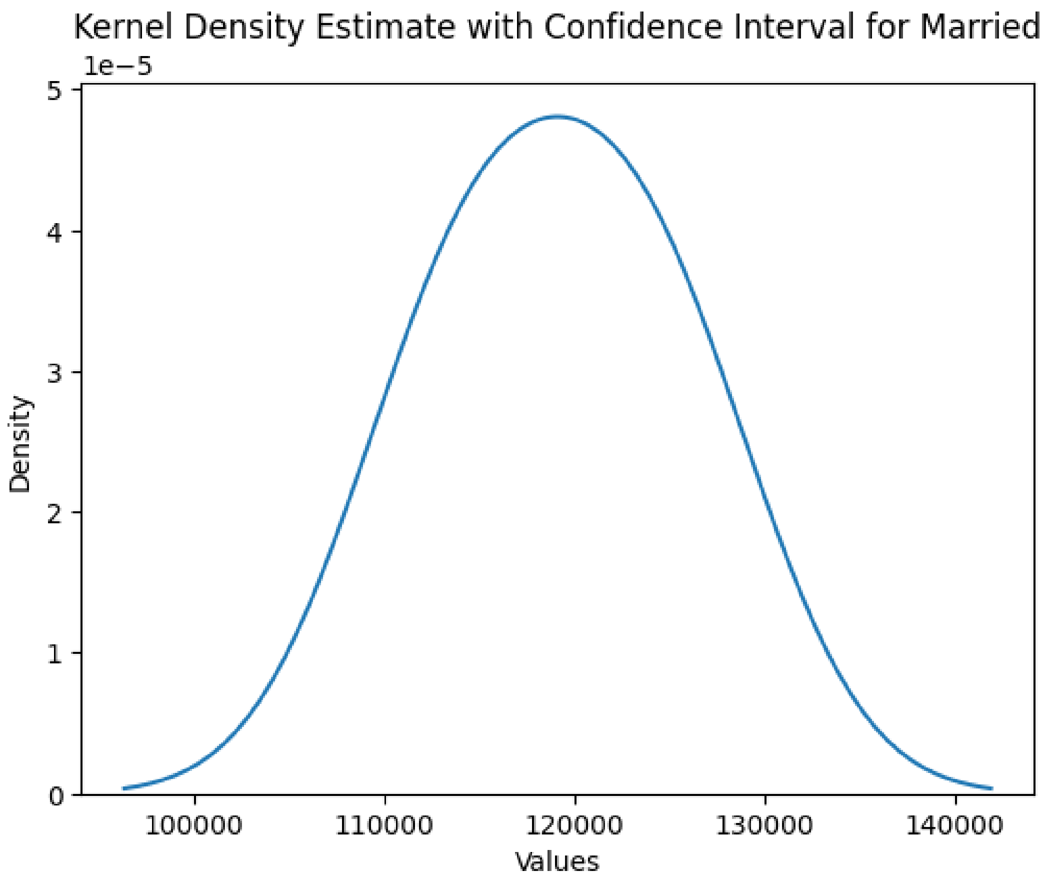
```
Confidence Interval 95% Unmarried: (120522.41313727012, 130216.92686272987)
```



[24]:

```
# Calculate the confidence interval for female and presenting it on the graph
Married_confidence_interval = (Married_sample_mean - margin_of_error,␣
 ↪Married_sample_mean + margin_of_error)
print("Confidence Interval 95% Married:", Married_confidence_interval)
sns.kdeplot(Married_confidence_interval)
plt.xlabel('Values')
plt.ylabel('Density')
plt.title('Kernel Density Estimate with Confidence Interval for Married')
plt.show()
```

```
Confidence Interval 95% Married: (114260.42313727012, 123954.93686272987)
```



## 18 Insight

1) With reference to the above data, at a 95% confidence interval:

   a) The average amount spent by an unmarried customer will lie between 847105.2492916514 and 914046.3107083486.

   b) The average amount spent by a married customer will lie between 810056.2692916514 and 876997.3307083487.

2) Confidence intervals for average unmarried and married spending are overlapping.

3) With respect to the above data, company should target more unmarried customers, as they spend a lot compared to married customers.

#6. How does Age affect the amount spent?

## 19 Results when the same activity is performed for Age

[30]:
```
def calculate_age_group_means_and_confidence_intervals(df):
    sum_by_age = df.groupby(['User_ID', 'Age'])['Purchase'].sum().reset_index()
    sum_by_age = sum_by_age.sort_values(by='User_ID', ascending=False)
    # Create dict and filtering data age group wise
    age_groups = {
        'Age_0_17': sum_by_age[sum_by_age['Age'] == '0-17'],
        'Age_18_25': sum_by_age[sum_by_age['Age'] == '18-25'],
        'Age_26_35': sum_by_age[sum_by_age['Age'] == '26-35'],
        'Age_36_45': sum_by_age[sum_by_age['Age'] == '36-45'],
        'Age_46_50': sum_by_age[sum_by_age['Age'] == '46-50'],
        'Age_51_55': sum_by_age[sum_by_age['Age'] == '51-55'],
        'Age_55+': sum_by_age[sum_by_age['Age'] == '55+']
    }
    # Define sample sizes and number of repetitions
    sample_sizes = {
        'Age_0_17': 200,
        'Age_18_25': 1000,
        'Age_26_35': 2000,
        'Age_36_45': 1000,
        'Age_46_50': 500,
        'Age_51_55': 400,
        'Age_55+': 300
    }
    num_repitions = 1000
    # Create a dictionary to store results
    results = {}
    # Perform random sampling and calculate means for each age group
    for age_group, age_df in age_groups.items():
        sample_size = sample_sizes.get(age_group, 0)
        sample_means = []
        for _ in range(num_repitions):
            random_sample = age_df.sample(n=sample_size,replace=True)
            sample_mean = random_sample['Purchase'].mean()
            sample_means.append(sample_mean)
        # Calculate the population mean, sample mean, and standard deviation
        population_mean = age_df['Purchase'].mean()
        sample_mean_mean = sum(sample_means) / len(sample_means)
        sample_mean_std = pd.Series(sample_means).std()
        # Calculate the confidence interval using the z-distribution
        confidence_level = 0.95  # 95% confidence interval
        z_critical = stats.norm.ppf((1 + confidence_level) / 2)  # Z-score for␣
↪the desired confidence level
        margin_of_error = z_critical * (age_df['Purchase'].std() / np.
↪sqrt(sample_size))
        lower_bound = sample_mean_mean - margin_of_error
        upper_bound = sample_mean_mean + margin_of_error
        results[age_group] = {
            'Population Mean': population_mean,
            'Sample Mean Mean': sample_mean_mean,
            'Sample Mean Std': sample_mean_std,
            'Confidence Interval': (lower_bound, upper_bound)
        }
    return results
results = calculate_age_group_means_and_confidence_intervals(df)
for age_group, metrics in results.items():
    print(f'{age_group} average spent value, random mean value, std value and␣
 ↪Confidence Interval:')
    print(f'{age_group} customer average spent amount: {metrics["Population␣
 ↪Mean"]}')
    print(f'Random Sample Mean : {metrics["Sample Mean Mean"]}')
    print(f'Sample Mean Std: {metrics["Sample Mean Std"]}')
    print(f'Confidence Interval: {metrics["Confidence Interval"]}')
    print()
```

```
Age_0_17 average spent value, random mean value, std value and Confidence
Interval:
Age_0_17 customer average spent amount: 85943.85446009389
Random Sample Mean : 85660.51138999997
Sample Mean Std: 6921.515376666196
Confidence Interval: (72244.90125230767, 99076.12152769227)

Age_18_25 average spent value, random mean value, std value and Confidence
Interval:
Age_18_25 customer average spent amount: 124589.04461687682
Random Sample Mean : 124575.73826900005
Sample Mean Std: 4124.878257779727
Confidence Interval: (116203.43262723908, 132948.04391076104)

Age_26_35 average spent value, random mean value, std value and Confidence
Interval:
Age_26_35 customer average spent amount: 138576.3559236948
Random Sample Mean : 138614.54756649997
Sample Mean Std: 3364.0053357113093
Confidence Interval: (132057.03541206103, 145172.0597209389)

Age_36_45 average spent value, random mean value, std value and Confidence
Interval:
Age_36_45 customer average spent amount: 124412.07780725021
Random Sample Mean : 124399.47062600011
Sample Mean Std: 4559.304634848914
Confidence Interval: (115472.77395354124, 133326.16729845898)
```

```
Age_46_50 average spent value, random mean value, std value and Confidence
Interval:
Age_46_50 customer average spent amount: 110150.29354207436
Random Sample Mean : 110022.25591199986
Sample Mean Std: 6055.963509791998
Confidence Interval: (98301.20257628469, 121743.30924771503)

Age_51_55 average spent value, random mean value, std value and Confidence
Interval:
Age_51_55 customer average spent amount: 111011.80652173913
Random Sample Mean : 110887.32599999987
Sample Mean Std: 5840.5572653592935
Confidence Interval: (99147.55713978056, 122627.09486021918)

Age_55+ average spent value, random mean value, std value and Confidence
Interval:
Age_55+ customer average spent amount: 78458.40677966102
Random Sample Mean : 78429.24523999995
Sample Mean Std: 5136.442205543787
Confidence Interval: (68143.08746973018, 88715.40301026971)
```

## 20 Insight

1) With reference to the above data, at a 95% confidence interval:
   a) The highest average amount spent by 26- to 35-year-old customers will lie between 944419.9990 and 1034842.9516.
   b) The average amount spent by 36- to 45-year-old customers will lie between 819003.0902 and 940678.8198.
   c) The average amount spent by 18- to 25-year-old customers will lie between 799594.4375 and 909664.7362.
   d) The average amount spent by 46- to 50-year-old customers will lie between 711215.1004 and 874125.3830.
   e) The average amount spent by 51- to 55-year-old customers will lie between 685670.0292 and 840962.3353.
   f) The average amount spent by 55+ age group customers will lie between 470454.5225 and 610200.5797.
   g) The lowest average amount spent by 0 to 17-year-old customers will lie between 524534.4423 and 714973.3156.
2) From the above data, it is clear that the age group 26 to 35 spends more compared to other age categories.
3) Age groups above 55 and below 0 to 17 spend very little compared to others.

4) Confidence intervals for average 26- to 35-year-old and 36- to 45-year-old spending are not overlapping.

5) With respect to the above data, the company should target the age category between 26 and 35, as they spend more money compared to others.

## 21 Recommendations

1) Men spend more money than women, so the company should focus on retaining male customers and getting more male customers.

2) Product Category: 5, 1, and 8 have the highest purchasing frequency. It means the products in these categories are liked more by customers. The company can focus on selling more of these products.

3) Product Category: 11, 2, and 6, 3 have almost close competition in purchasing.

   The company can focus on selling more of these products.

4) Unmarried customers spend more money compared to married customers. So the company should focus on retaining the unmarried customers and getting more unmarried customers.

5) 86% of purchases are done by customers whose ages are between 18 and 45. So the company should focus on the acquisition of customers who are aged 18–45.

6) Customers living in City_Category C spend more money than other customers living in B or A. Selling more products in City Category C will help the company increase sales.

```
[ ]:
```