Question 1

Defining the problem statement

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 [1018]:

```
##importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
```

In [1019]:

```
##Loading the data
walmart = pd.read_csv("walmart.csv")
walmart.head()
```

Out[1019]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchas
0	1000001	P00069042	F	0-17	10	A	2	0	3	837
1	1000001	P00248942	F	0-17	10	A	2	0	1	1520
2	1000001	P00087842	F	0-17	10	A	2	0	12	142
3	1000001	P00085442	F	0-17	10	A	2	0	12	105
4	1000002	P00285442	M	55+	16	C	4+	0	8	796

Analyzing the basic metrics

Observations on shape of data

In [1020]:

```
walmart.shape ## 550068 rows and 10 columns
```

Out[1020]:

```
(550068, 10)
```

In [1021]:

```
walmart.ndim
```

Out[1021]:

```
2
```

In [1022]:

```
walmart.size
```

Out[1022]:

```
5500680
```

data types of all the attributes

In [1023]:

```
walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64 
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

- User_ID, Occupation, Marital_Status,Product_Category,Purchase are numerical
- Rest all are non numerical or categorical in the original data
- No **null** values

conversion of categorical attributes to 'category'

In [1024]:

```
walmart["Marital_Status"] = walmart["Marital_Status"].astype('object')
```

In [1025]:

```
walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  object
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(4), object(6)
memory usage: 42.0+ MB
```

statistical summary

In [1026]:

```
walmart.describe() ##numerical attributes
```

Out[1026]:

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

In [1027]:

```
walmart[["User_ID","Occupation","Product_Category","Purchase"]].mean()
```

Out[1027]:

```
User_ID             1.003029e+06
Occupation          8.076707e+00
Product_Category    5.404270e+00
Purchase            9.263969e+03
dtype: float64
```

In [1028]:

```
walmart[["User_ID","Occupation","Product_Category","Purchase"]].median()
```

Out[1028]:

```
User_ID             1003077.0
Occupation                7.0
Product_Category          5.0
Purchase               8047.0
dtype: float64
```

Occupation and Purchase have outliers present since mean and median have considerable difference

In [1029]:

```
walmart.mode()
```

Out[1029]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchas
0	1001680	P00265242	M	26-35	4	B	1	0	5	701

- Most **frequent buyer** is 1001680
- Most of the buyers are **male** and hail fron **city category B**
- Most of the buyers have **occupation 4**
- Most of the buyers are **unmarried**
- **Product_category 5** with product_id P00265242 is most frequently bought

Non-Graphical Analysis: Value counts and unique attributes

In [1030]:

```
walmart.nunique()
```

Out[1030]:

```
User_ID                        5891
Product_ID                     3631
Gender                            2
Age                               7
Occupation                       21
City_Category                     3
Stay_In_Current_City_Years        5
Marital_Status                    2
Product_Category                 20
Purchase                      18105
dtype: int64
```

In [1031]:

```
walmart["User_ID"].value_counts()
```

Out[1031]:

```
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
```

1001680 is the most frequent buyer

In [1032]:

```
walmart["Product_ID"].value_counts()
```

Out[1032]:

```
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
```

- P00265242 is the most common product

In [1033]:

```
walmart["Product_ID"].nunique()
```

Out[1033]:

```
3631
```

- Only 3631 product_id's are unique

In [1034]:

```
walmart["Gender"].value_counts()
```

Out[1034]:

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

- Most of the buyers are Males

In [1035]:

```
walmart["Age"].value_counts()
```

Out[1035]:

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

- Most of the buyers are in Age group 26 to 35 and 36-45

In [1036]:

```
walmart["Occupation"].value_counts()
```

Out[1036]:

```
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
```

- Most of the buyers have occupation 4

In [1037]:

```
walmart["City_Category"].value_counts()
```

Out[1037]:

```
B    231173
C    171175
A    147720
Name: City_Category, dtype: int64
```

- Most of the buyers stay in city category B

In [1038]:

```
walmart["Stay_In_Current_City_Years"].value_counts()
```

Out[1038]:

```
1     193821
2     101838
3      95285
4+     84726
0      74398
Name: Stay_In_Current_City_Years, dtype: int64
```

- Most number of buyers have been staying for less than 2 years in the current city

In [1039]:

```
walmart["Marital_Status"].value_counts()
```

Out[1039]:

```
0    324731
1    225337
Name: Marital_Status, dtype: int64
```

- Mostly buyers are unmarried

In [1040]:

```
walmart["Product_Category"].value_counts()
```

Out[1040]:

```
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
```

- Product category 5 is the most bought along with 1,8

In [1041]:

```
walmart["Purchase"].value_counts()
```

Out[1041]:

```
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 [1042]:

```
walmart["Purchase"].value_counts()[:50]
```

Out[1042]:

```
7011    191
7193    188
6855    187
6891    184
7012    183
6960    183
6879    182
7166    182
7027    182
6868    180
7165    180
6883    180
6858    179
7093    178
6931    178
7089    178
7185    178
6923    178
7114    177
7188    177
7085    176
6908    176
7060    176
7167    175
6973    175
6928    175
6949    175
7146    175
7159    175
6904    174
7010    174
7962    174
6952    174
7192    174
7034    174
6862    173
7047    173
7067    172
7108    172
7049    172
6930    172
7081    172
7028    172
6978    172
7110    171
6938    171
7024    171
7046    171
7026    171
7083    171
Name: Purchase, dtype: int64
```

- Most of the purchase are in the range 6k to 7k

Visual Analysis - Univariate & Bivariate

Univariate

In [1043]:

```
walmart.head()
```

Out[1043]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchas
0	1000001	P00069042	F	0-17	10	A	2	0	3	837
1	1000001	P00248942	F	0-17	10	A	2	0	1	1520
2	1000001	P00087842	F	0-17	10	A	2	0	12	142
3	1000001	P00085442	F	0-17	10	A	2	0	12	105
4	1000002	P00285442	M	55+	16	C	4+	0	8	796

In [1044]:

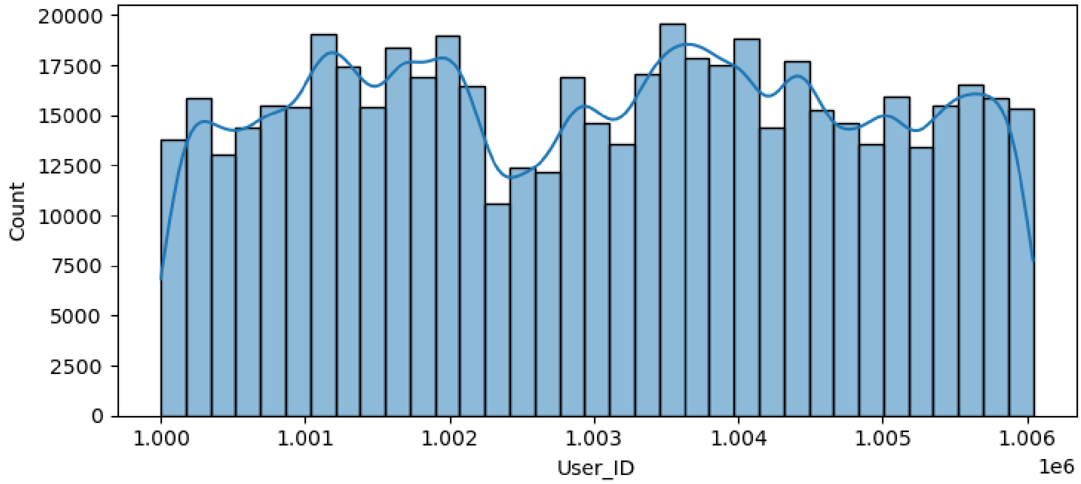
```
walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  object
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(4), object(6)
memory usage: 42.0+ MB
```

Continous Univariate

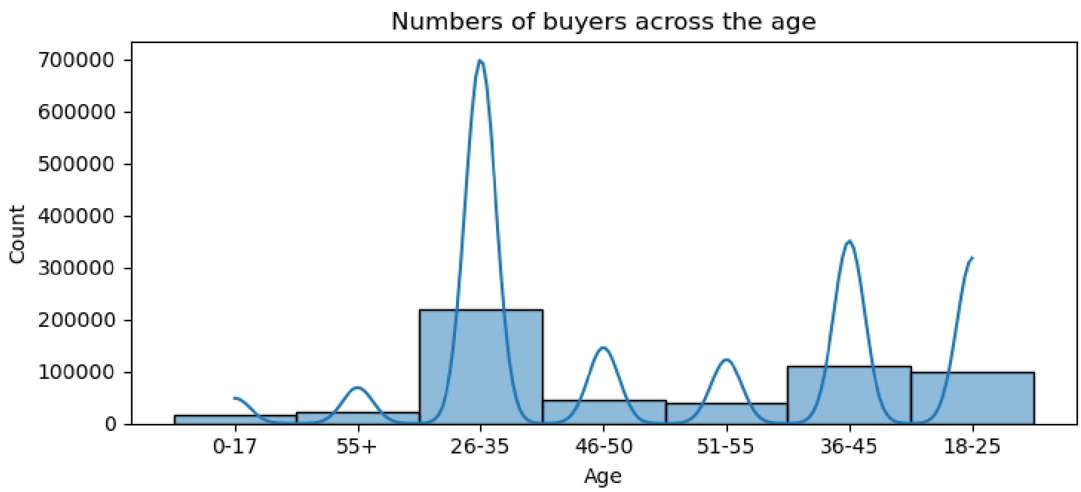
In [1045]:

```
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True
sns.histplot(data=walmart["User_ID"],bins=35,kde=True)
plt.show()
```



In [1046]:

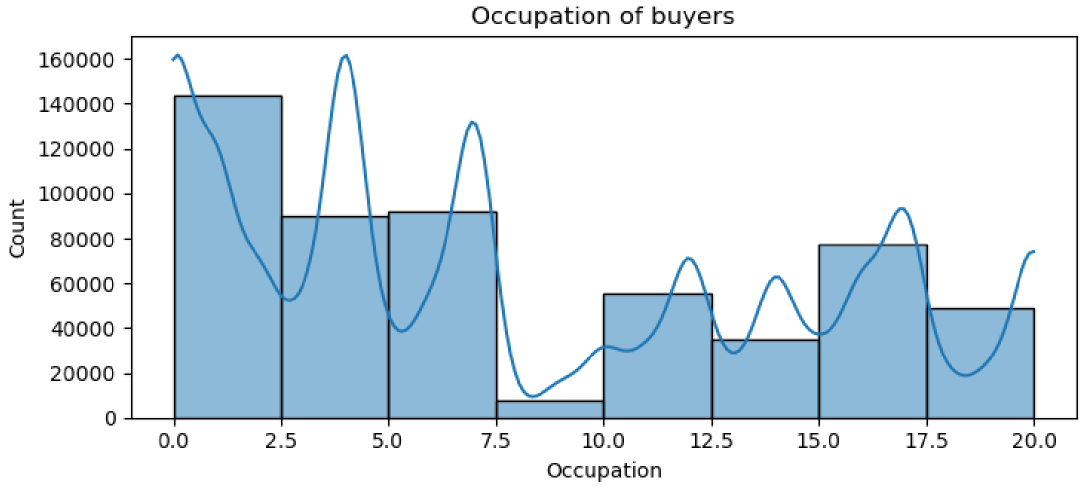
```
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True
sns.histplot(data=walmart["Age"],bins=14,kde=True)
plt.title("Numbers of buyers across the age")
plt.show()
```



- Most of the users are between 26-35 & 36-45

In [1047]:

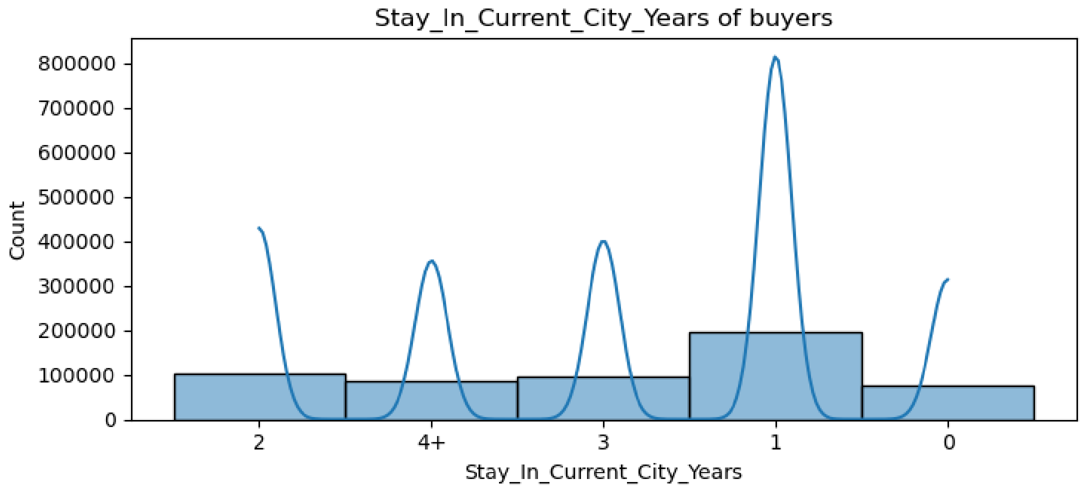
```
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True
sns.histplot(data=walmart["Occupation"],binwidth=2.5,kde=True)
plt.title("Occupation of buyers")
plt.show()
```



- Most of the buyers belong to occupation 4

In [1048]:

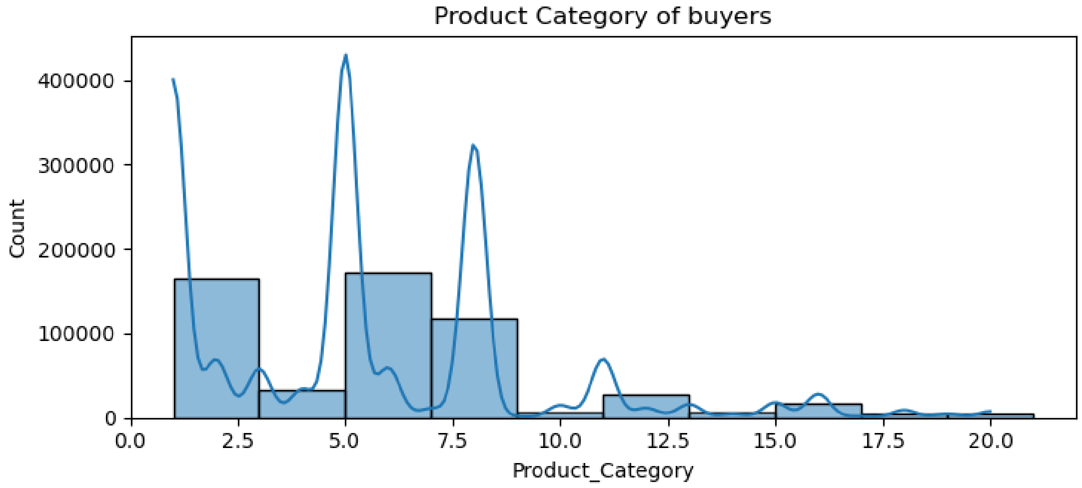
```
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True
sns.histplot(data=walmart["Stay_In_Current_City_Years"],kde=True)
plt.title("Stay_In_Current_City_Years of buyers")
plt.show()
```



- Most of buyers have stayed only for 1 year in the current city
- Very less buyers have been staying in the city for less than a year

In [1049]:

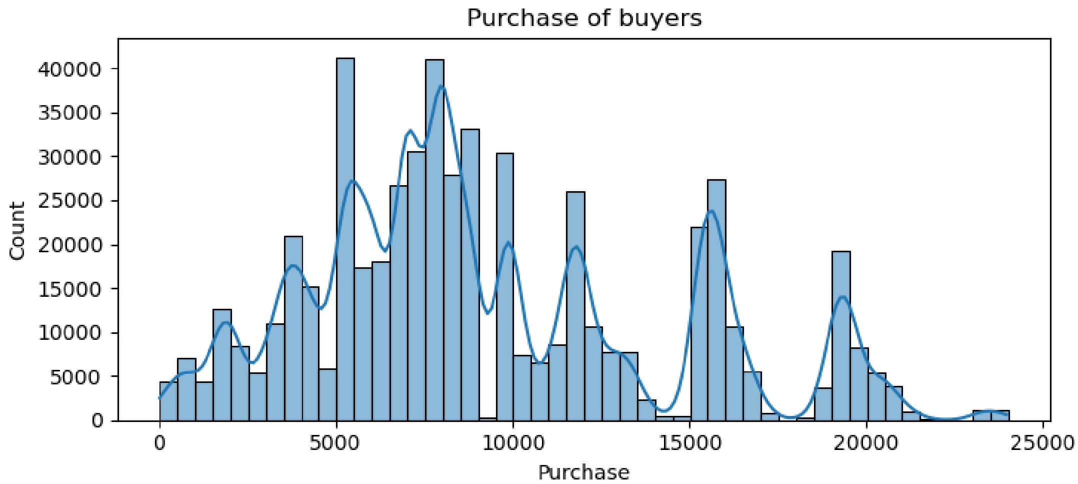
```
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True
sns.histplot(data=walmart["Product_Category"],binwidth=2,kde=True)
plt.title("Product Category of buyers")
plt.show()
```



- Product Category 5 is the most bought

In [1050]:

```
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True
sns.histplot(data=walmart["Purchase"],binwidth=500,kde=True)
plt.title("Purchase of buyers")
plt.show()
```



- Most of the buyers spent within 5000 to 12000

Categorical Univariate

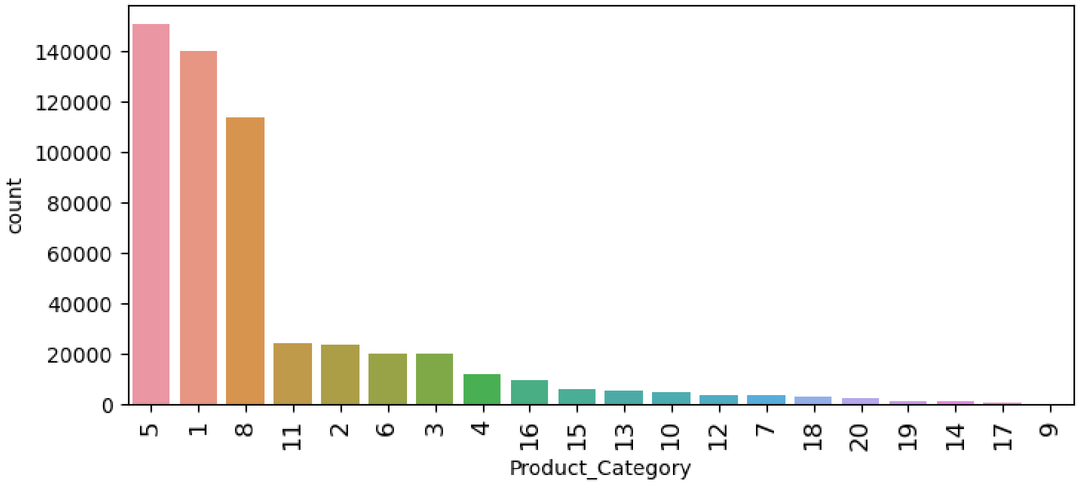
In [1051]:

```
walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  object
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(4), object(6)
memory usage: 42.0+ MB
```

In [1052]:

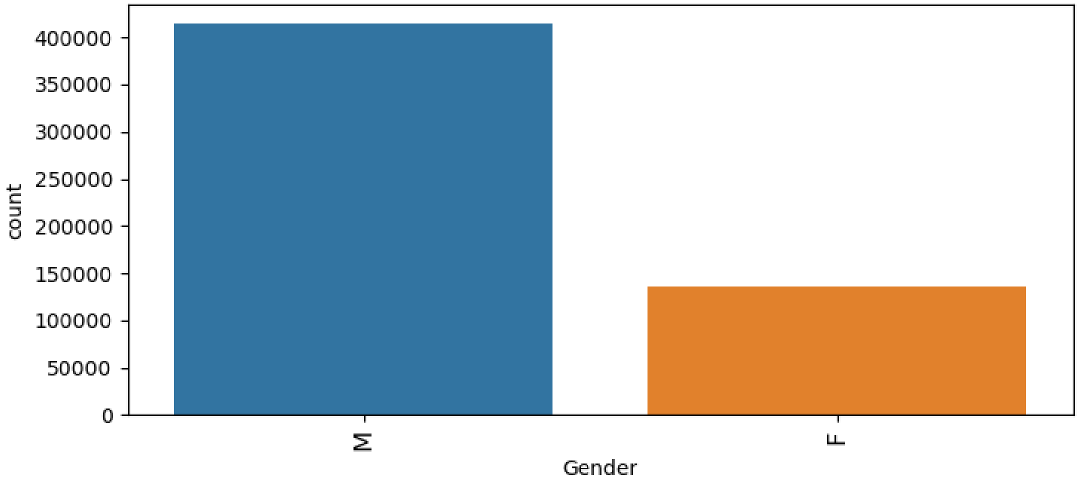
```
sns.countplot(data=walmart,x="Product_Category",order = walmart['Product_Category'].value_counts().index)
plt.xticks(rotation=90,fontsize=12) ## to avoid overlapping labels
plt.show()
```



- Product Category 5,1,8 are the most bought

In [1053]:

```
sns.countplot(data=walmart,x="Gender",order = walmart['Gender'].value_counts().index)
plt.xticks(rotation=90,fontsize=12) ## to avoid overlapping labels
plt.show()
```



In [1054]:

```
walmart.groupby("Gender").count()["User_ID"]
```

Out[1054]:

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

In [1055]:

```
(135809/(135809+414259))*100
```

Out[1055]:

```
24.689492935418894
```

In [1056]:

```
414259/(135809+414259)
```

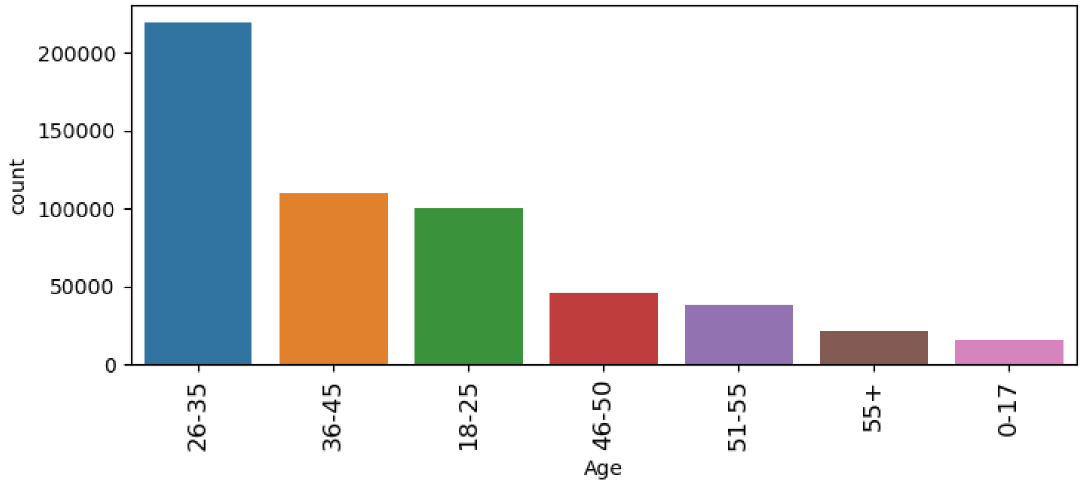
Out[1056]:

```
0.7531050706458111
```

- Only 24 % of buyers are females, and 75% buyers are males

In [1057]:

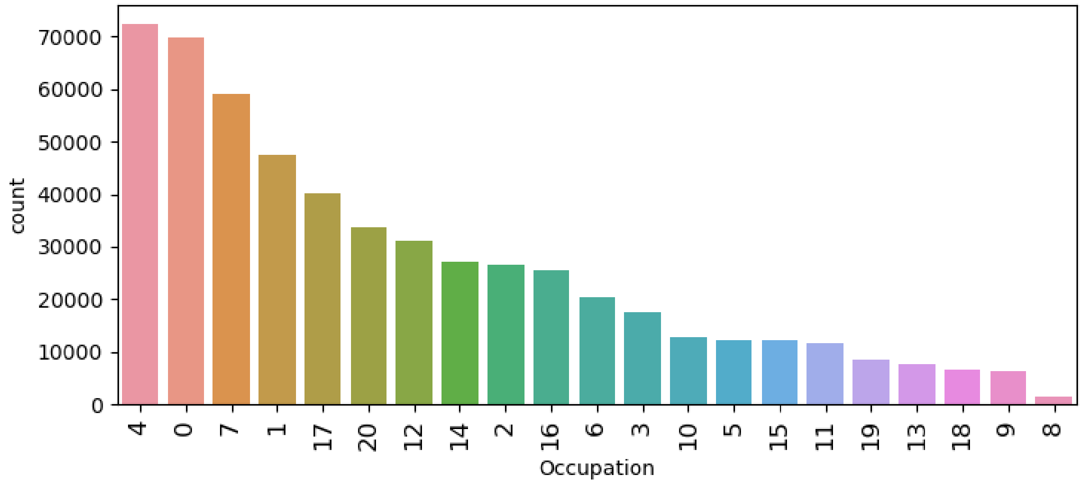
```
sns.countplot(data=walmart,x="Age",order = walmart['Age'].value_counts().index)
plt.xticks(rotation=90,fontsize=12) ## to avoid overlapping labels
plt.show()
```



- 26-25 age group has most number of buyers

In [1058]:

```
sns.countplot(data=walmart,x="Occupation",order = walmart['Occupation'].value_counts().index)
plt.xticks(rotation=90,fontsize=12) ## to avoid overlapping labels
plt.show()
```



- Most of the buyers have occptaion 4 or occupation 0

In [1059]:

```
walmart.loc[walmart["Occupation"]==0]["Gender"].value_counts()
```

Out[1059]:

```
M    51526
F    18112
Name: Gender, dtype: int64
```

In [1060]:

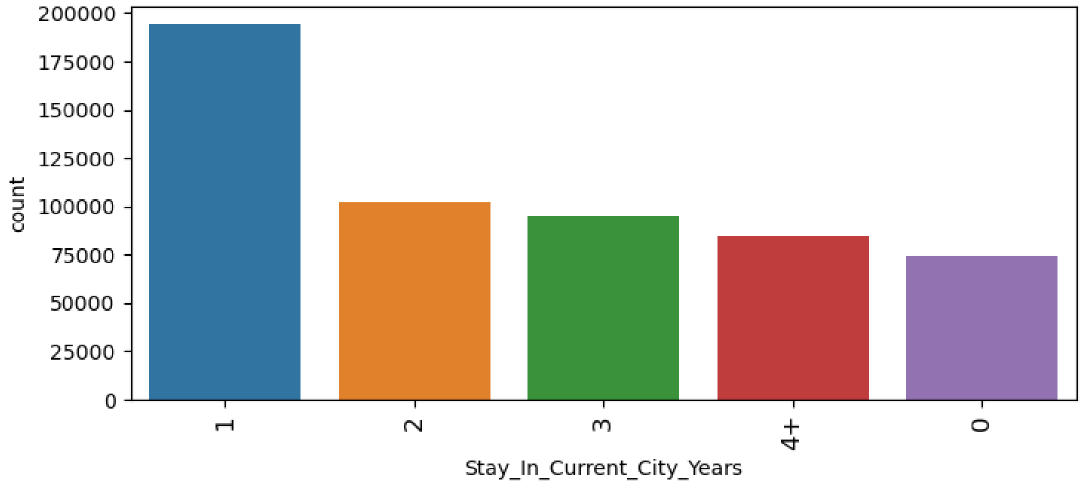
```
walmart.loc[walmart["Occupation"]==4]["Gender"].value_counts()
```

Out[1060]:

```
M    54472
F    17836
Name: Gender, dtype: int64
```

In [1061]:

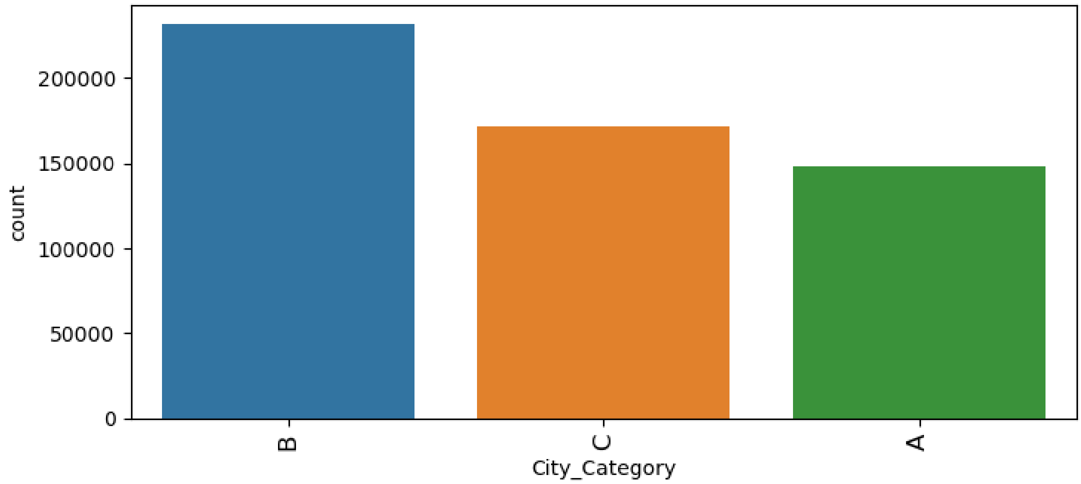
```
sns.countplot(data=walmart,x="Stay_In_Current_City_Years",order = walmart['Stay_In_Current_City_Years'].value_counts().inde:
plt.xticks(rotation=90,fontsize=12) ## to avoid overlapping labels
plt.show()
```



- Most of the buyers are have been living in the current city for 1-2 years

In [1062]:

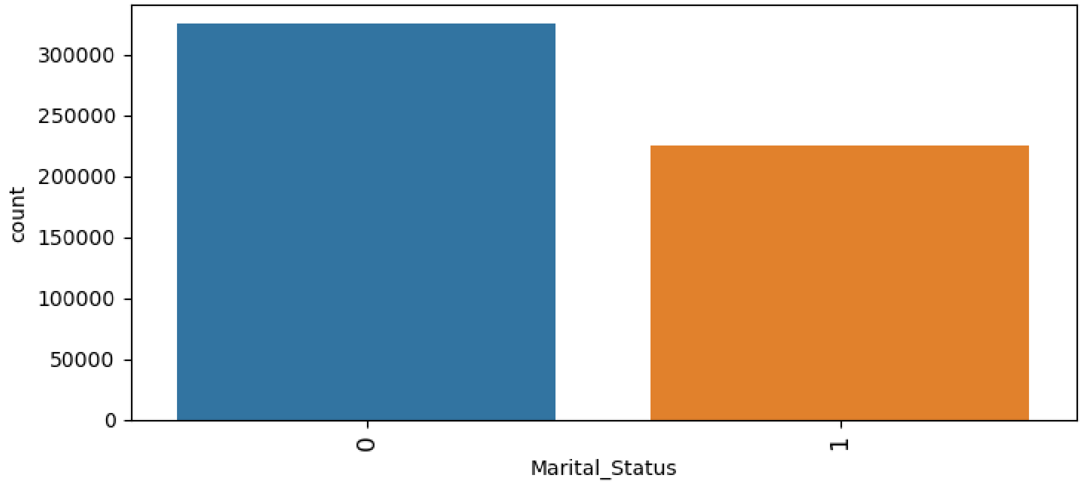
```
sns.countplot(data=walmart,x="City_Category",order = walmart['City_Category'].value_counts().index)
plt.xticks(rotation=90,fontsize=12) ## to avoid overlapping labels
plt.show()
```



- Most of the belong to city category B

In [1063]:

```
sns.countplot(data=walmart,x="Marital_Status",order = walmart['Marital_Status'].value_counts().index)
plt.xticks(rotation=90,fontsize=12) ## to avoid overlapping labels
plt.show()
```



- Most of the buyers are unmarried

In [1064]:

```
walmart.groupby("Marital_Status").count()["User_ID"]
```

Out[1064]:

```
Marital_Status
0    324731
1    225337
Name: User_ID, dtype: int64
```

In [1065]:

```
324731/(324731+225337)
```

Out[1065]:

```
0.5903470116421969
```

In [1066]:

```
225337/(324731+225337)
```

Out[1066]:

```
0.40965298835780306
```

- 59% of buyers are married 40% are unmarried

In [1067]:

```
pd.crosstab(index=walmart['Gender'], columns=[walmart['Marital_Status']],normalize=True,margins=True)
```

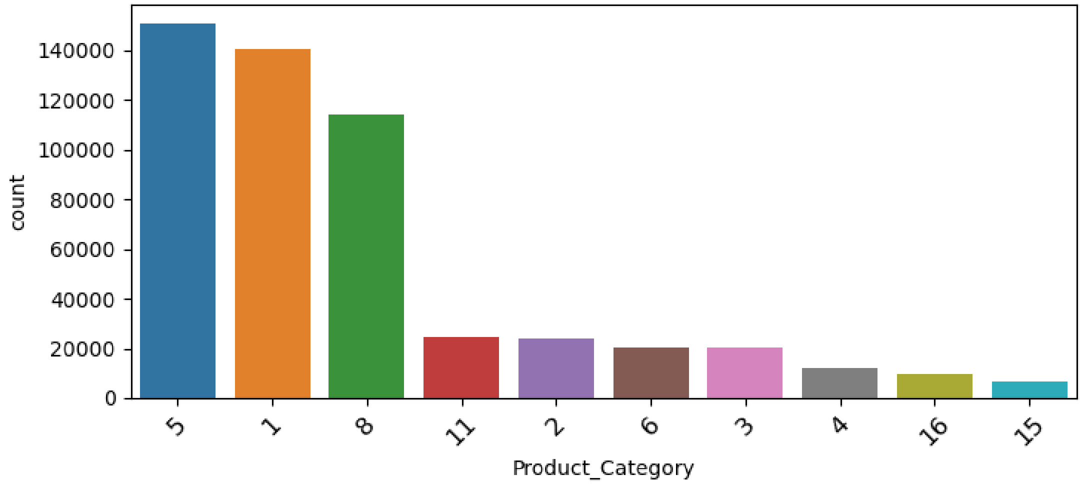
Out[1067]:

Marital_Status	0	1	All
Gender			
F	0.143293	0.103602	0.246895
M	0.447054	0.306051	0.753105
All	0.590347	0.409653	1.000000

Top 10 product categories

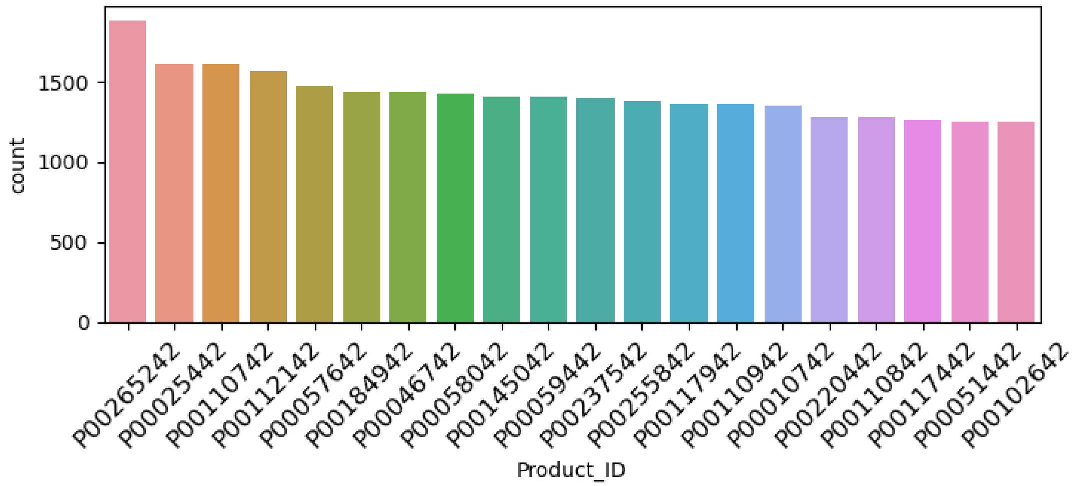
In [1068]:

```
walmart_top10_product_category = walmart["Product_Category"].value_counts()[:10]
top10_productcategory=walmart[walmart["Product_Category"].isin(walmart_top10_product_category.index)]
sns.countplot(data=top10_productcategory,x="Product_Category",order = top10_productcategory['Product_Category'].value_count
plt.xticks(rotation=45,fontsize=12) ## to avoid overlapping labels
plt.show()
```



In [1069]:

```
##Top 20 product_ID
walmart_top20_product_id = walmart["Product_ID"].value_counts()[:20]
top20_productid=walmart[walmart["Product_ID"].isin(walmart_top20_product_id.index)]
sns.countplot(data=top20_productid,x="Product_ID",order = top20_productid['Product_ID'].value_counts().index)
plt.xticks(rotation=45,fontsize=12) ## to avoid overlapping labels
plt.show()
```



- 'P00265242', 'P00025442', 'P00110742' are most sold product_id's

Bivariate

Bivariate Analaysis of two categorical variables

In [1070]:

```
walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  object
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(4), object(6)
memory usage: 42.0+ MB
```

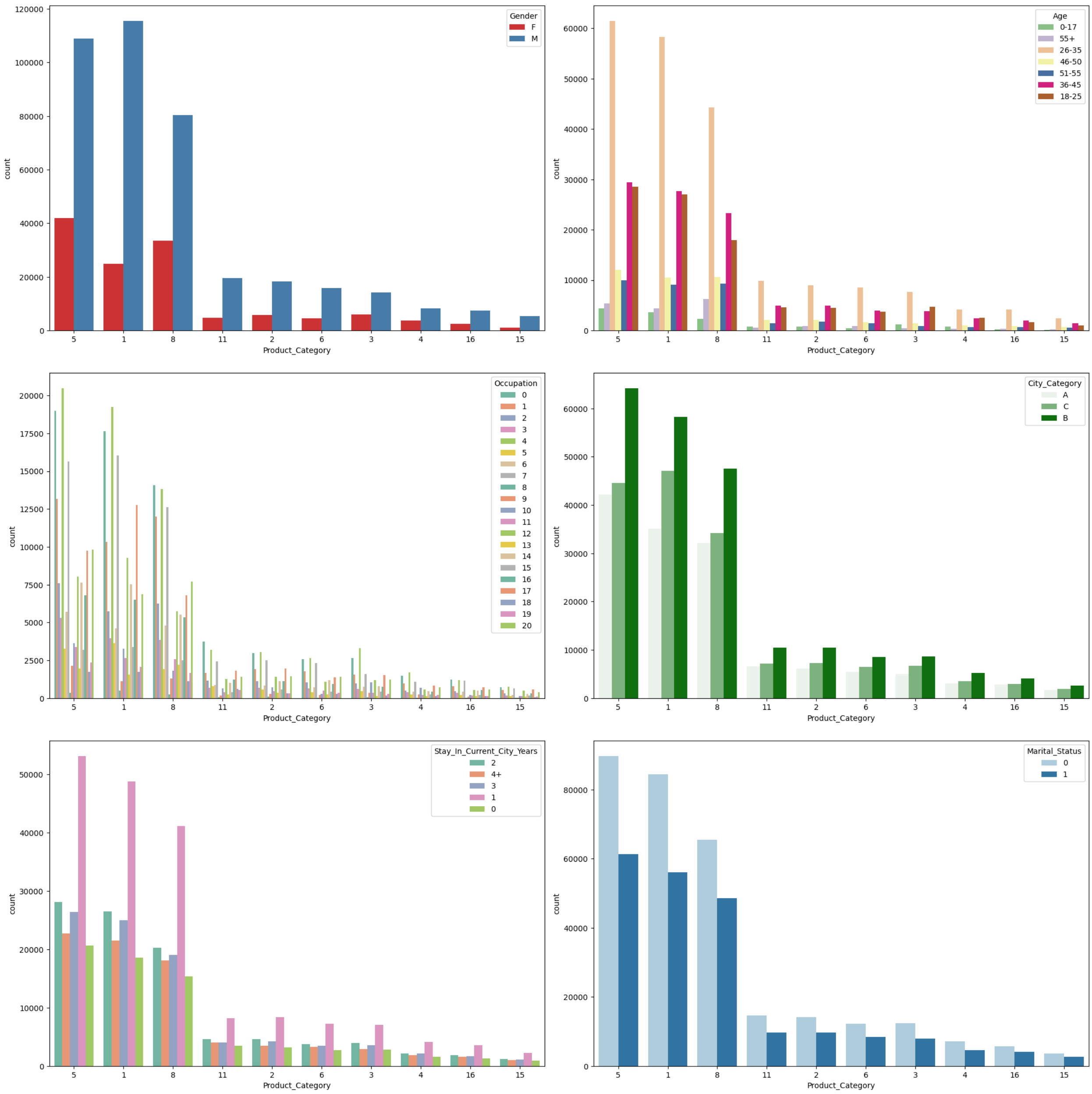
Top product_category

In [1071]:

```
##Top 10 product_category
walmart_top10_product_category = walmart["Product_Category"].value_counts()[:10]
top10_productcategory=walmart[walmart["Product_Category"].isin(walmart_top10_product_category.index)]
```

In [1072]:

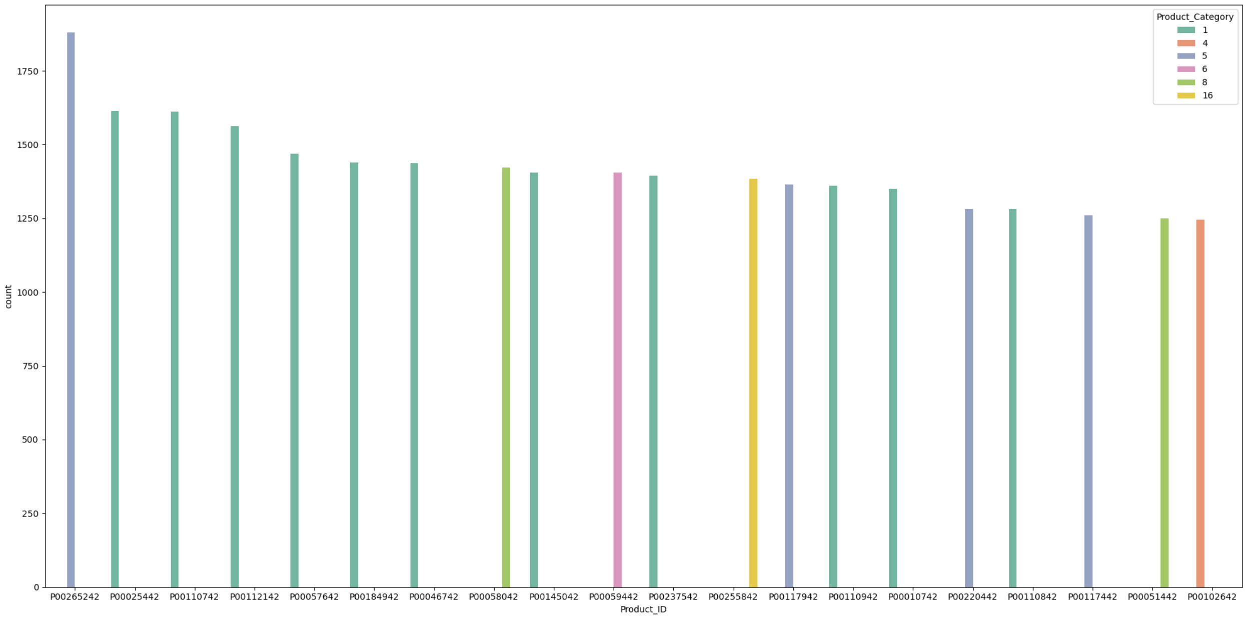
```
plt.figure(figsize=(20,20))
plt.subplot(3,2,1) ##1 shows the position
sns.countplot(data=top10_productcategory,x='Product_Category',hue='Gender',order=top10_productcategory['Product_Category'].v
plt.subplot(3,2,2)
sns.countplot(data=top10_productcategory,x='Product_Category',hue='Age',order=top10_productcategory['Product_Category'].valu
plt.subplot(3,2,3)
sns.countplot(data=top10_productcategory,x='Product_Category',hue='Occupation',order=top10_productcategory['Product_Category
plt.subplot(3,2,4)
sns.countplot(data=top10_productcategory,x='Product_Category',hue='City_Category',order=top10_productcategory['Product_Categ
plt.subplot(3,2,5)
sns.countplot(data=top10_productcategory,x='Product_Category',hue='Stay_In_Current_City_Years',order=top10_productcategory[
plt.subplot(3,2,6)
sns.countplot(data=top10_productcategory,x='Product_Category',hue='Marital_Status',order=top10_productcategory['Product_Cat
plt.show()
```



- Most of the buyers acroos the top 10 product category are males, female buyers are very less comparably
- Most of the buyers are between 26-35 years of age,very less buyers in age group 0 to 17 years of age
- Most of the buyers have occupation 4 , occupation 8 buyers are very few
- Most of the buyers for these product category belong to city category B
- Most of the buyers for these product category have been the city for 0-1 years
- Most of the buyers are unmarried

In [1073]:

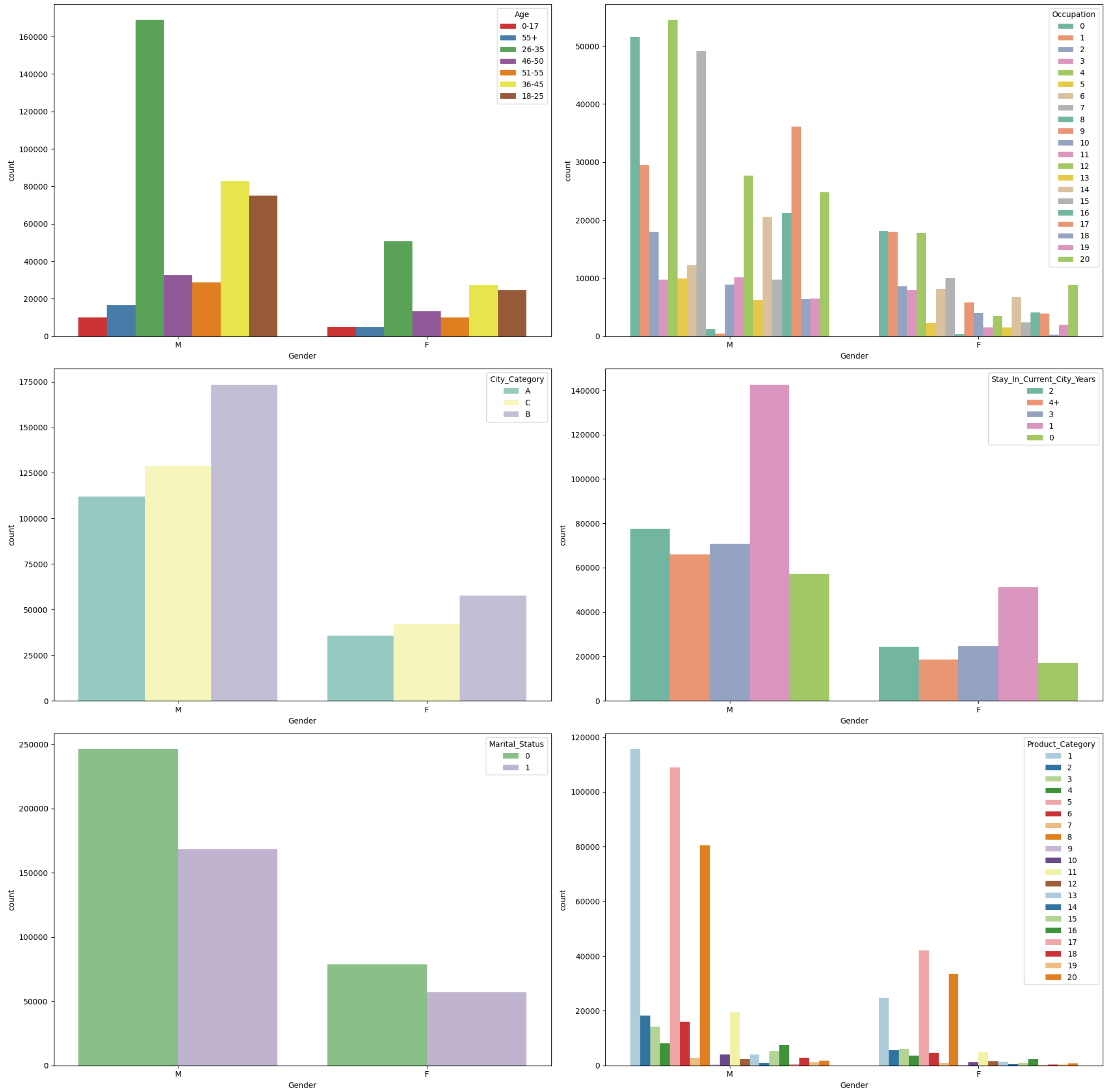
```
plt.figure(figsize=(20,10))
sns.countplot(data=top20_productid,x='Product_ID',hue='Product_Category',order=top20_productid['Product_ID'].value_counts()
plt.show()
```



- Product category 5 is the most popular amongst top 5 productids
- Product category 1 is common for most of the product_id and is also more frequently bought along with 5

In [1074]:

```
plt.figure(figsize=(20,20))
plt.subplot(3,2,1) ##1 shows the position
sns.countplot(data=walmart,x='Gender',hue='Age',order=walmart['Gender'].value_counts().index,palette="Set1")
plt.subplot(3,2,2)
sns.countplot(data=walmart,x='Gender',hue='Occupation',order=walmart['Gender'].value_counts().index,palette="Set2")
plt.subplot(3,2,3)
sns.countplot(data=walmart,x='Gender',hue='City_Category',order=walmart['Gender'].value_counts().index,palette="Set3")
plt.subplot(3,2,4)
sns.countplot(data=walmart,x='Gender',hue='Stay_In_Current_City_Years',order=walmart['Gender'].value_counts().index,palette
plt.subplot(3,2,5)
sns.countplot(data=walmart,x='Gender',hue='Marital_Status',order=walmart['Gender'].value_counts().index,palette="Accent")
plt.subplot(3,2,6)
sns.countplot(data=walmart,x='Gender',hue='Product_Category',order=walmart['Gender'].value_counts().index,palette="Paired")
plt.show()
```



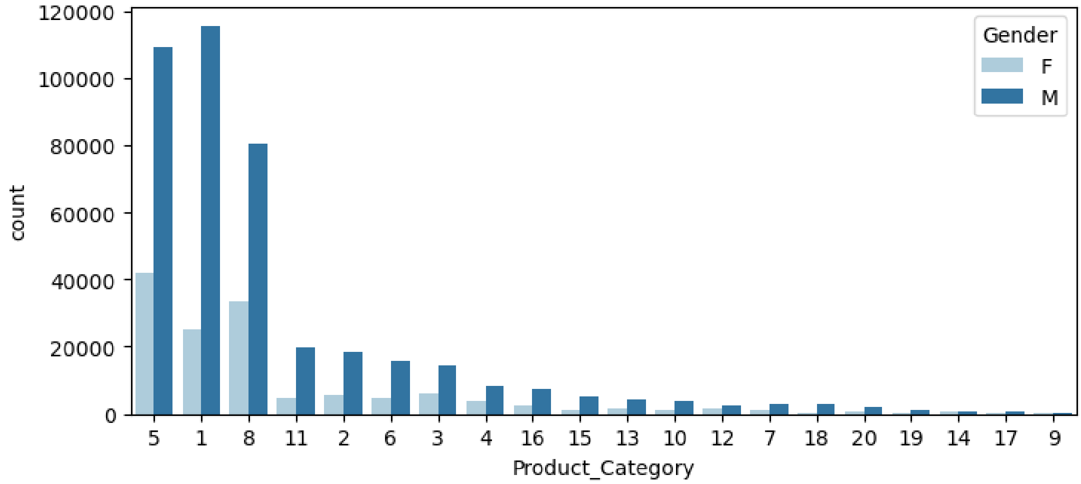
- Age group 26-45 have most number of buyers both across males and female
- Male buyers are mostly from occuptaion 4 and city category B, same hold true for female buyers
- Most of the male and female buyers are unmarried.

In [1075]:

```
sns.countplot(data=walmart,x='Product_Category',hue='Gender',order=walmart['Product_Category'].value_counts().index,palette=
```

Out[1075]:

```
<Axes: xlabel='Product_Category', ylabel='count'>
```



- Most of the females buyers buy product_category 5 whereas the most of male buyers buy product category 1

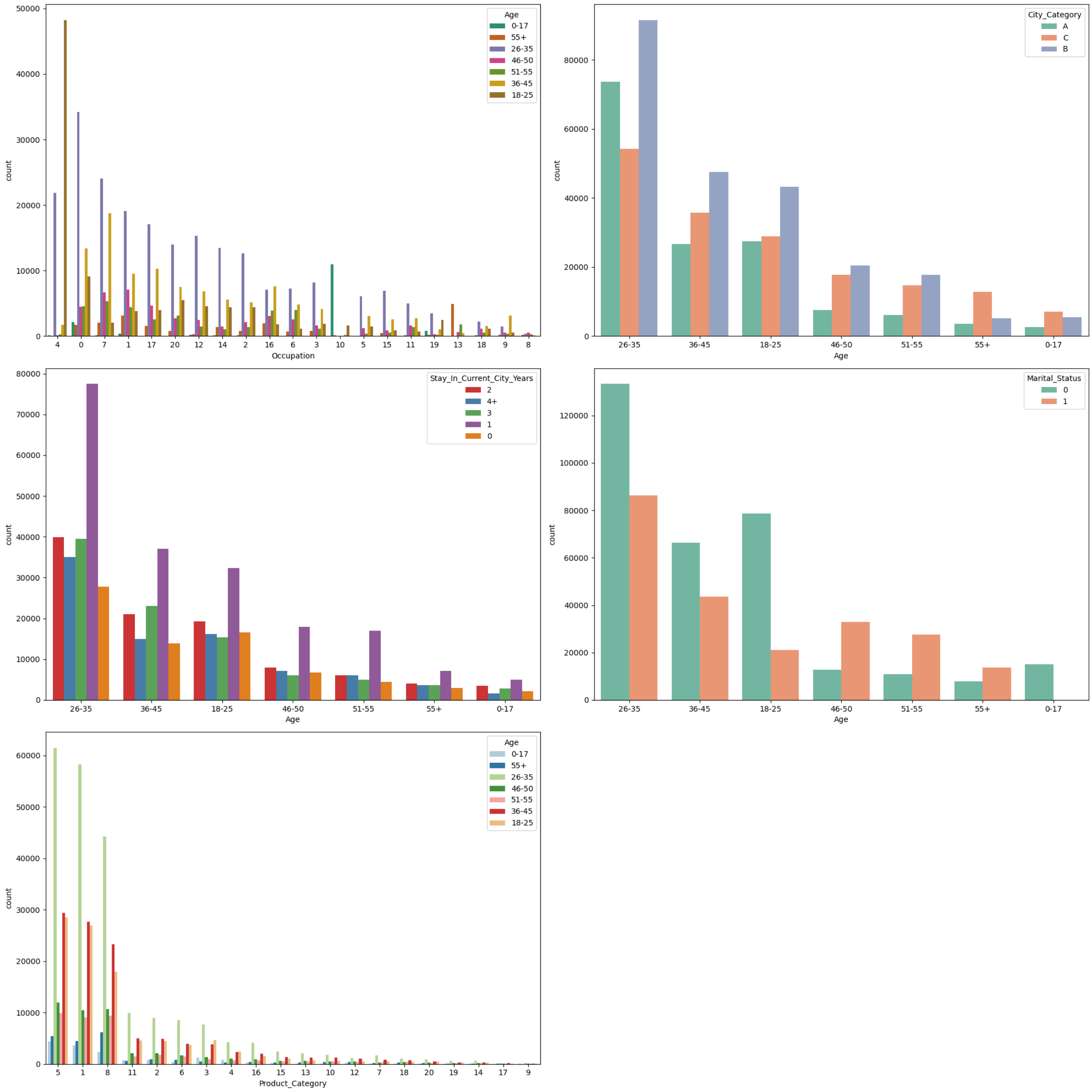
In [1076]:

```
walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  object
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(4), object(6)
memory usage: 42.0+ MB
```

In [1077]:

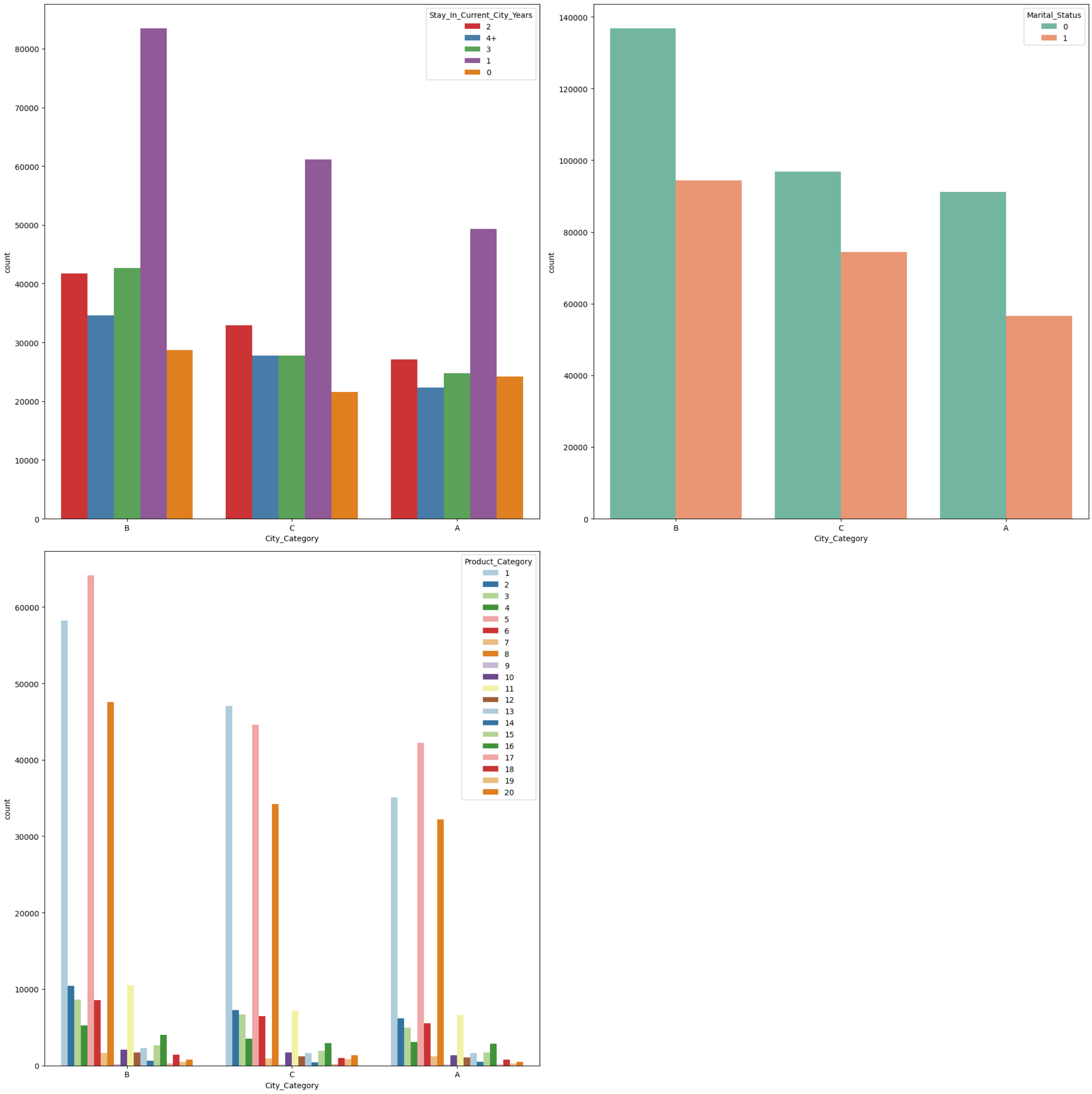
```
plt.figure(figsize=(20,20))
plt.subplot(3,2,1) ##1 shows the position
sns.countplot(data=walmart,x='Occupation',hue='Age',order=walmart['Occupation'].value_counts().index,palette="Dark2")
plt.subplot(3,2,2)
sns.countplot(data=walmart,x='Age',hue='City_Category',order=walmart['Age'].value_counts().index,palette="Set2")
plt.subplot(3,2,3)
sns.countplot(data=walmart,x='Age',hue='Stay_In_Current_City_Years',order=walmart['Age'].value_counts().index,palette="Set1'
plt.subplot(3,2,4)
sns.countplot(data=walmart,x='Age',hue='Marital_Status',order=walmart['Age'].value_counts().index,palette="Set2")
plt.subplot(3,2,5)
sns.countplot(data=walmart,x='Product_Category',hue='Age',order=walmart['Product_Category'].value_counts().index,palette="Pa
plt.show()
```



- Buyers in the age group 0-17,18-25 are mostly unmarried and 46-55+ are mostly married
- Product category 5,1,8 are most popular amongsth age group 26-25
- Preference of city category for age group 26-35 is B, A,C , whereas for age group 36-45, B,C,A

In [1078]:

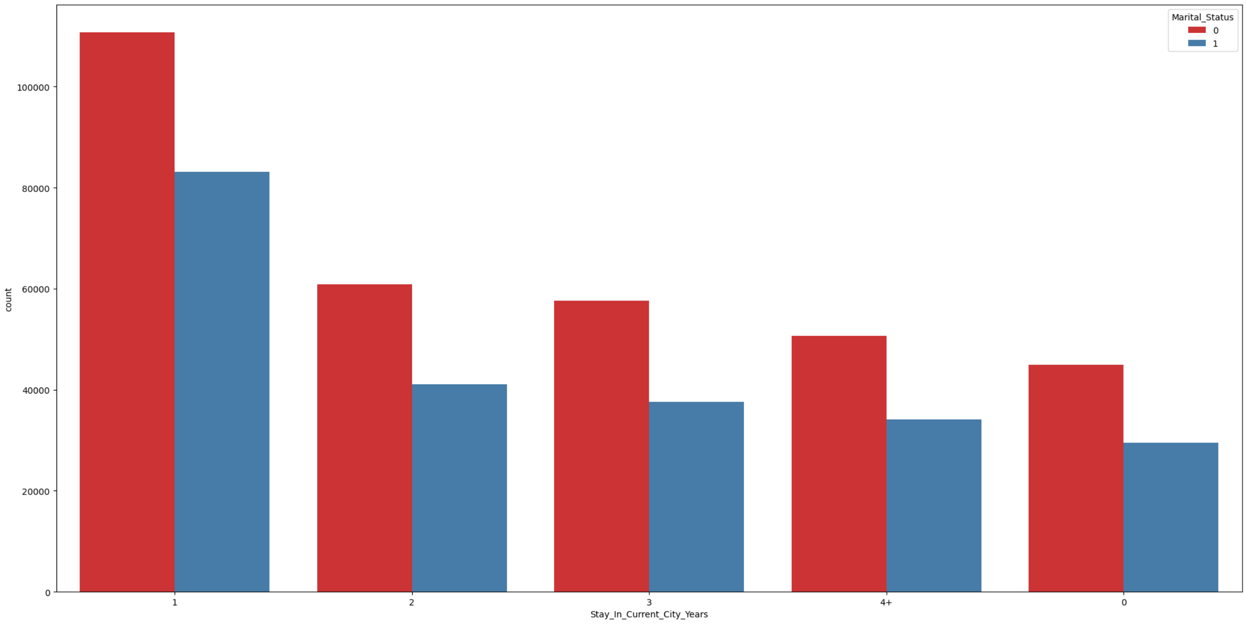
```
plt.figure(figsize=(20,20))
plt.subplot(2,2,1) ##1 shows the position
sns.countplot(data=walmart,x='City_Category',hue='Stay_In_Current_City_Years',order=walmart['City_Category'].value_counts()
plt.subplot(2,2,2)
sns.countplot(data=walmart,x='City_Category',hue='Marital_Status',order=walmart['City_Category'].value_counts().index,palet
plt.subplot(2,2,3)
sns.countplot(data=walmart,x='City_Category',hue='Product_Category',order=walmart['City_Category'].value_counts().index,pal
plt.show()
```

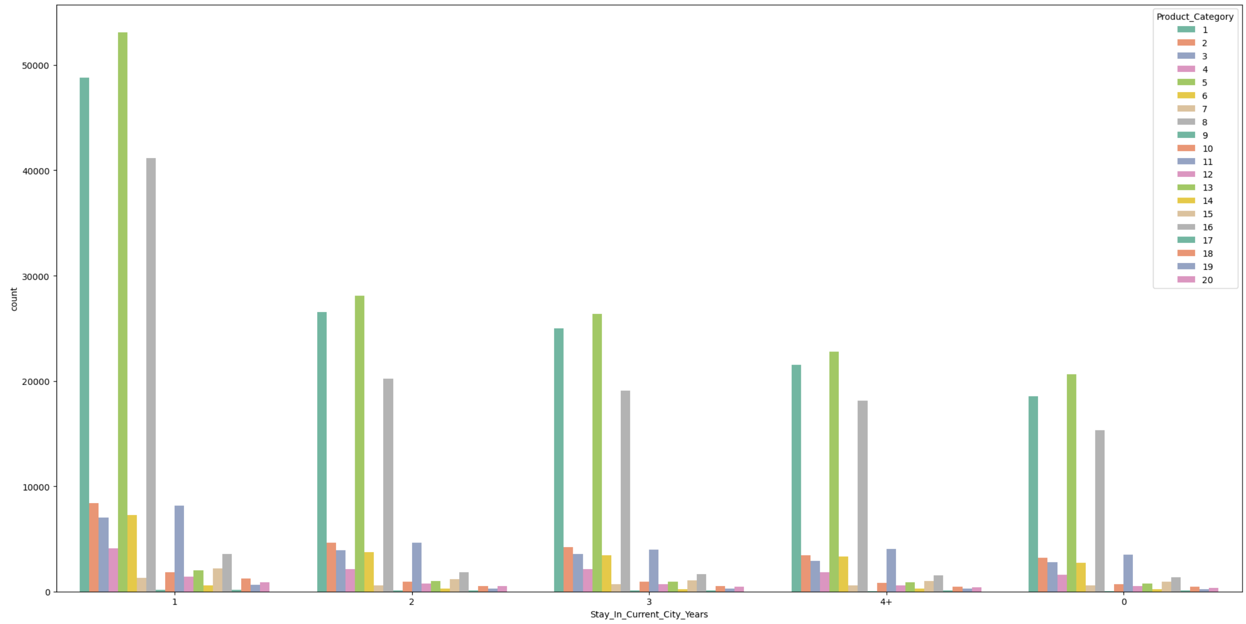


- City Category B,A has 5 as most bought product category whereas 1 is the most bought product category in city category C

In [1079]:

```
plt.figure(figsize=(20,20))
plt.subplot(2,1,1) ##1 shows the position
sns.countplot(data=walmart,x='Stay_In_Current_City_Years',hue='Marital_Status',order=walmart['Stay_In_Current_City_Years'].v
plt.subplot(2,1,2)
sns.countplot(data=walmart,x='Stay_In_Current_City_Years',hue='Product_Category',order=walmart['Stay_In_Current_City_Years']
plt.show()
```





Bivariate analysis for Categorical-Numerical or Numerical-Categorical

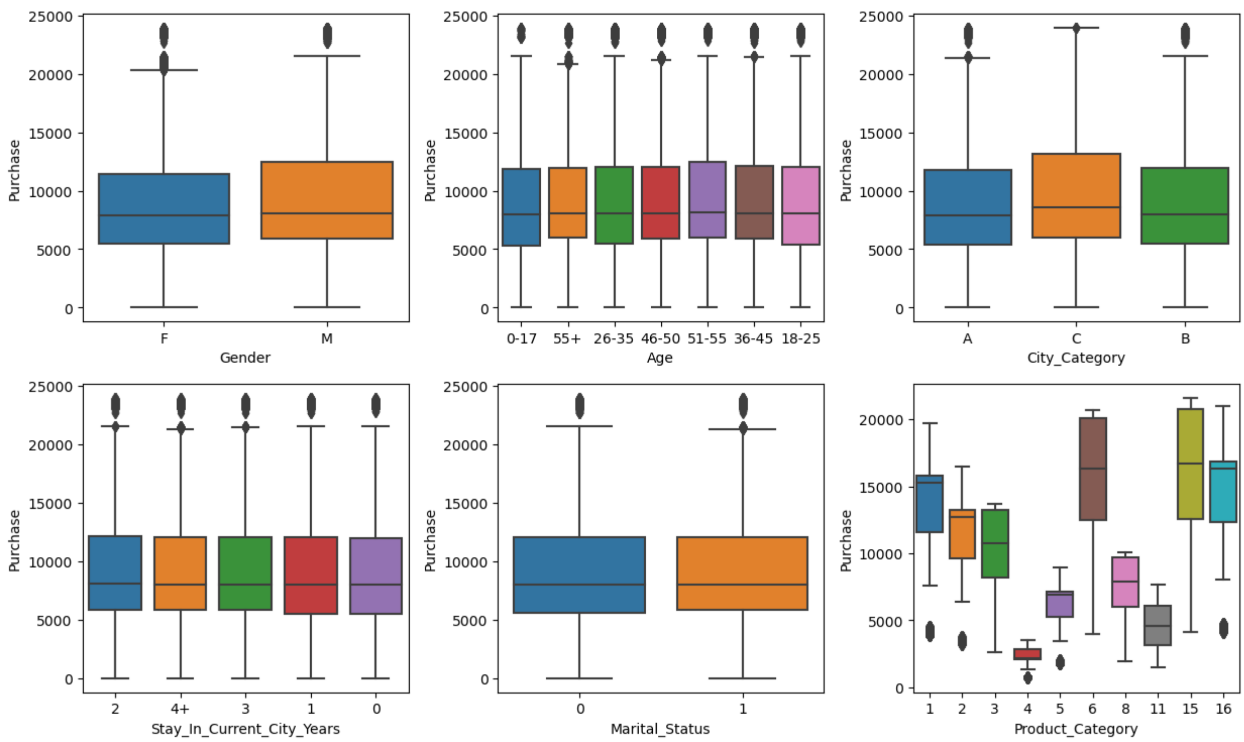
In [1080]:

```
walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  object
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(4), object(6)
memory usage: 42.0+ MB
```

In [1081]:

```
plt.rcParams["figure.figsize"] = [12.50, 7.50]
plt.rcParams["figure.autolayout"] = True
plt.subplot(2,3,1) ##1 shows the position
sns.boxplot(data=walmart,x='Gender',y='Purchase')
plt.subplot(2,3,2)
sns.boxplot(data=walmart,x='Age',y='Purchase')
plt.subplot(2,3,3)
sns.boxplot(data=walmart,x='City_Category',y='Purchase')
plt.subplot(2,3,4)
sns.boxplot(data=walmart,x='Stay_In_Current_City_Years',y='Purchase')
plt.subplot(2,3,5)
sns.boxplot(data=walmart,x='Marital_Status',y='Purchase')
plt.subplot(2,3,6)
sns.boxplot(data=top10_productcategory,x='Product_Category',y='Purchase')
plt.show()
```

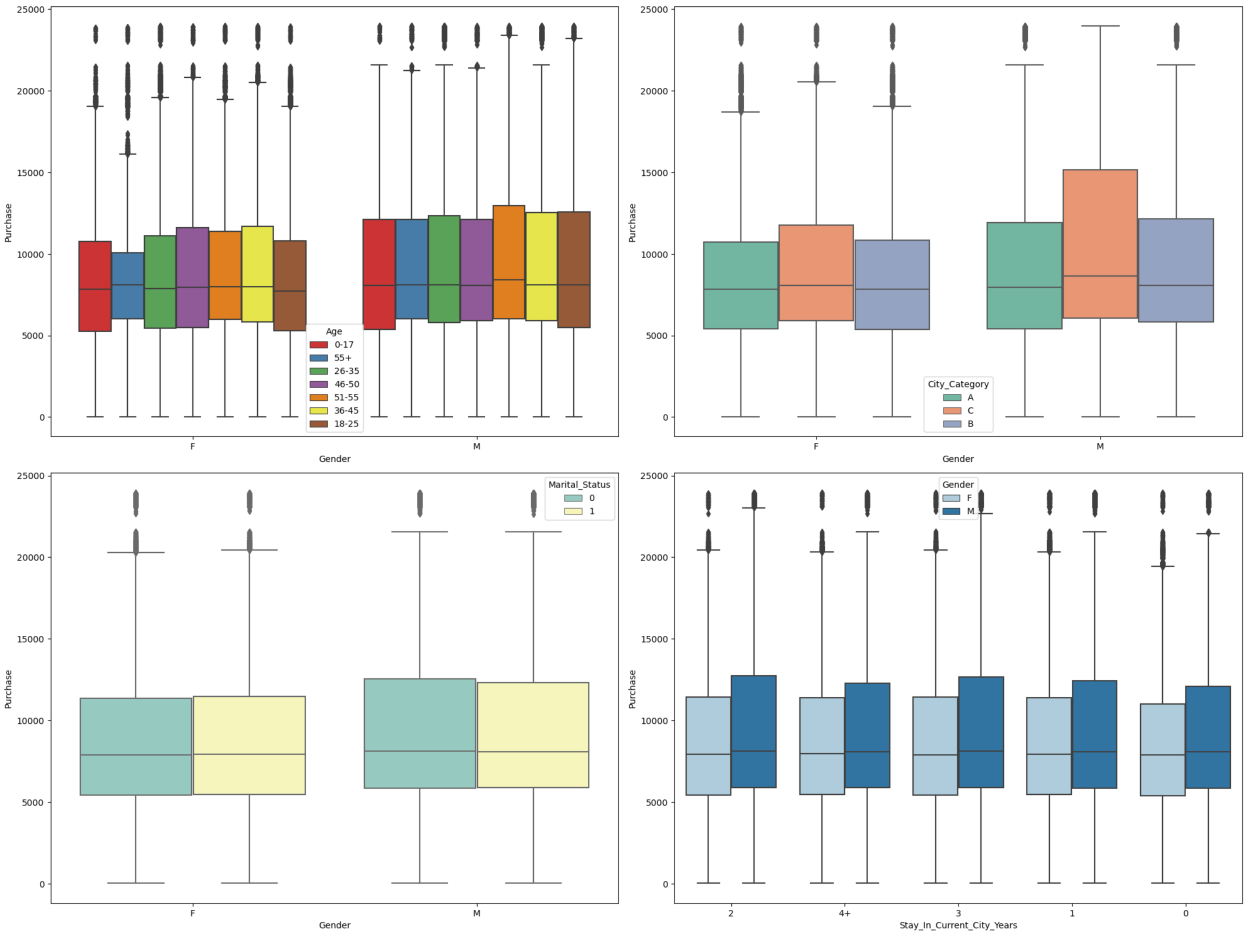


- Median purchase value for male is higher than for female, more number of outliers in female purchases
- Median purchase value of buyer from city category C is higher and has lesser outliers
- Product category 15 ,6 have the highest number of buyers and purchase amount is also high with very less outliers

Mutivariate Analysis

In [1082]:

```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20, 15))
plt.rcParams["figure.autolayout"] = True
fig.subplots_adjust(top=1.5)
sns.boxplot(data=walmart, y='Purchase', x='Gender', hue='Age', palette='Set1', ax=axs[0,0])
sns.boxplot(data=walmart, y='Purchase', x='Gender', hue='City_Category', palette='Set2', ax=axs[0,1])
sns.boxplot(data=walmart, y='Purchase', x='Gender', hue='Marital_Status', palette='Set3', ax=axs[1,0])
sns.boxplot(data=walmart, y='Purchase', x='Stay_In_Current_City_Years', hue='Gender', palette='Paired', ax=axs[1,1])
plt.show()
```



- Female purchases has most outliers

Missing Value & Outlier Detection

In [1083]:

```
walmart.isnull().sum()
```

Out[1083]:

```
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
```

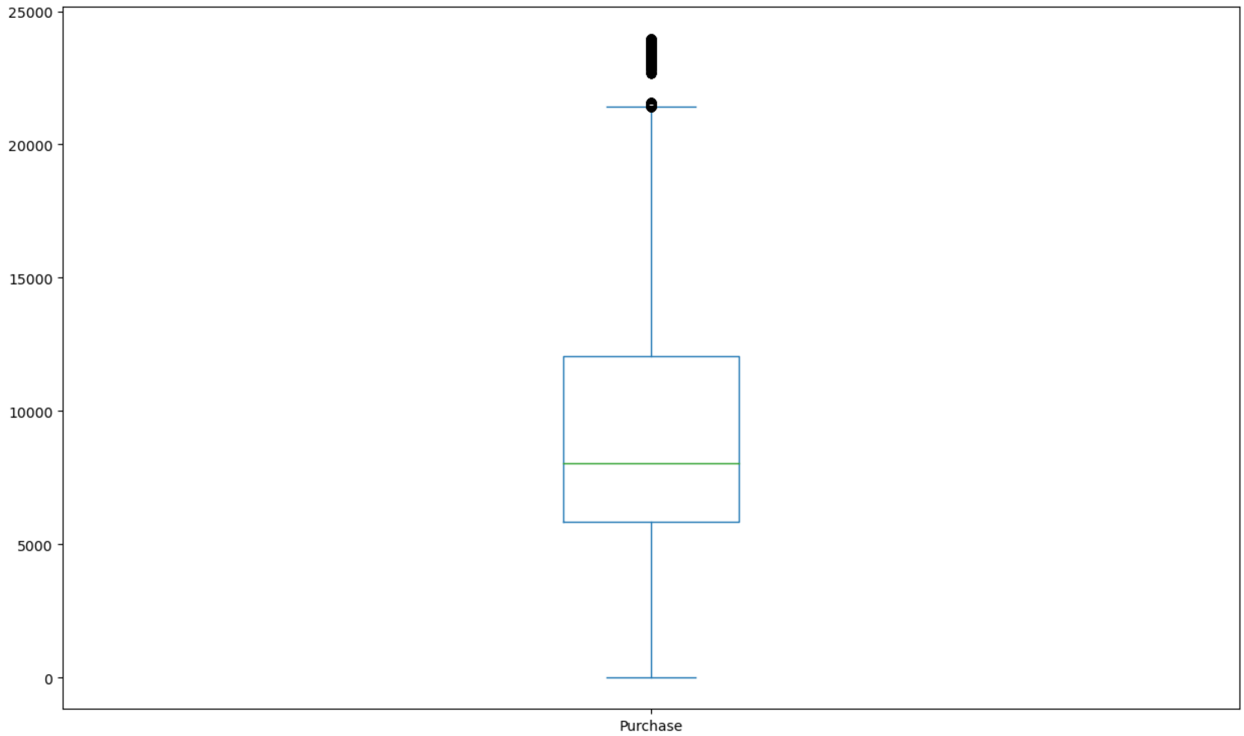
- No null values in the data

In [1084]:

```
walmart["Purchase"].plot(kind='box')
```

Out[1084]:

```
<Axes: >
```



In [1085]:

```
p_25_purchase = np.percentile(walmart['Purchase'],25) ##25th percentile
p_50_purchase = np.percentile(walmart['Purchase'],50) ##50 percentile
p_75_purchase = np.percentile(walmart['Purchase'],75) ##75 percentile
IQR_purchase = p_75_purchase - p_25_purchase
upper_purchase =  p_75_purchase + 1.5*IQR_purchase
lower_purchase = max(p_25_purchase - 1.5*IQR_purchase,0)
purchase_outliers=walmart.loc[walmart['Purchase'] > upper_purchase]["Purchase"]
percent_ouliers_purchase=(len(purchase_outliers)/len(walmart['Purchase']))*100
print(f"Upper Whisker: {upper_purchase} \nLower Whisker: {lower_purchase}\nIQR: {IQR_purchase}\nOutlier in Purchase:{purcha
```

```
Upper Whisker: 21400.5
Lower Whisker: 0
IQR: 6231.0
Outlier in Purchase:[23603 23792 23233 ... 23529 23663 23496]
Number of Outliers in Purchase:2677
percent_ouliers_purchase:0.49
```

Answering questions

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

In [1086]:

```
df_purchase=walmart.groupby(['User_ID', 'Gender'])[['Purchase']].sum().reset_index()
```

Female customers/buyers

In [1087]:

```
df_purchase_female=df_purchase[df_purchase["Gender"]=="F"]
print("Female population mean :" ,df_purchase_female["Purchase"].mean())
print("Female population standard deviation :" ,df_purchase_female["Purchase"].std())
```

```
Female population mean : 712024.3949579832
Female population standard deviation : 807370.7261464577
```

Male customers/buyers

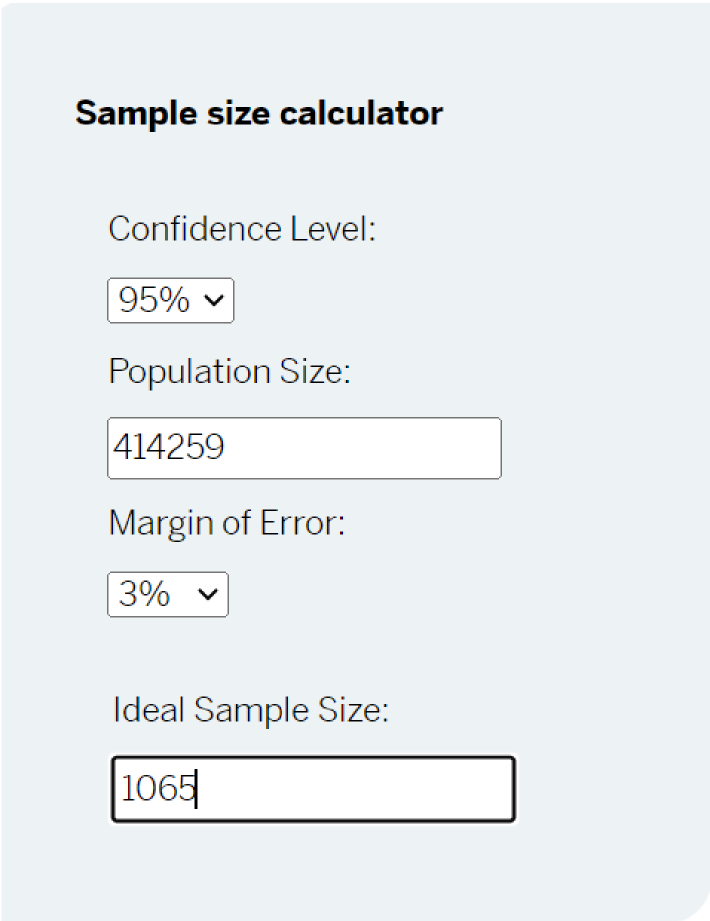
In [1088]:

```
df_purchase_male=df_purchase[df_purchase["Gender"]=="M"]
print("Male population mean :" ,df_purchase_male["Purchase"].mean())
print("Male population standard deviation :" ,df_purchase_male["Purchase"].std())
```

```
Male population mean : 925344.4023668639
Male population standard deviation : 985830.100795388
```

- Males spend more per transaction than females
- Males earning more than females could be reason behind this behavior , or possiblities are that females prefer buying online

Let us assume the confidence interval to be 95% and desired marging of error to be plus-minus 3

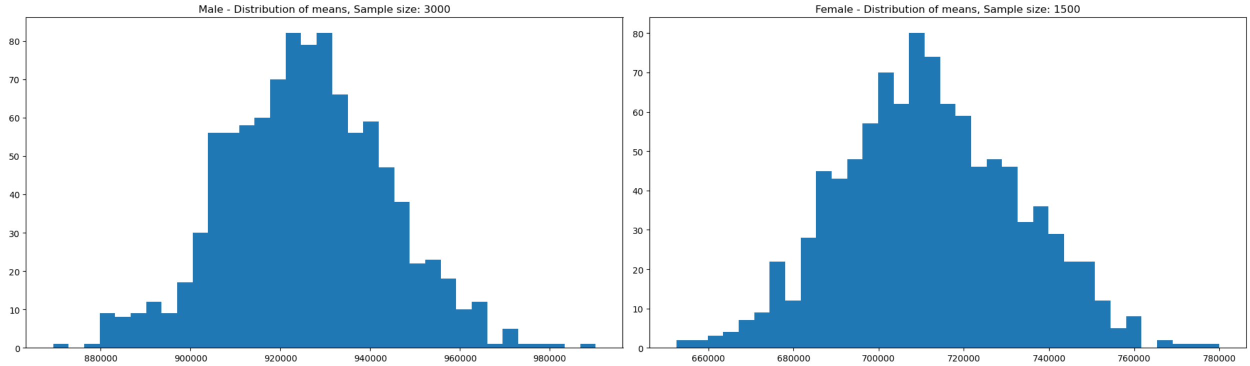


In [1089]:

```
genders = ["M", "F"]

male_sample_size = 3000
female_sample_size = 1500
num_repetitions = 1000
male_means = []
female_means = []
diff =[]

for _ in range(num_repetitions):
    male_mean = df_purchase_male["Purchase"].sample(male_sample_size, replace=True).mean()
    female_mean = df_purchase_female["Purchase"].sample(female_sample_size, replace=True).mean()
    male_means.append(male_mean)
    female_means.append(female_mean)
    diff.append(male_mean-female_mean)
```

In [1090]:

```
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))

axis[0].hist(male_means, bins=35)
axis[1].hist(female_means, 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()
```



In [1091]:

```
print("Population mean - Mean of sample means of amount spend for Male: {:.2f}".format(np.mean(male_means)))
print("Population mean - Mean of sample means of amount spend for Female: {:.2f}".format(np.mean(female_means)))

print("\nMale - Sample mean: {:.2f}, Sample std: {:.2f}".format(df_purchase_male['Purchase'].mean(), df_purchase_male['Purc
print("Female - Sample mean: {:.2f}, Sample std: {:.2f}".format(df_purchase_female['Purchase'].mean(), df_purchase_female['
```

```
Population mean - Mean of sample means of amount spend for Male: 925731.27
Population mean - Mean of sample means of amount spend for Female: 712134.89

Male - Sample mean: 925344.40, Sample std: 985830.10
Female - Sample mean: 712024.39, Sample std: 807370.73
```

Calculating the 95% confidence interval

In [1092]:

```
z=norm.ppf(.975)
z ## z_score
```

Out[1092]:

```
1.959963984540054
```

In [1093]:

```
male_margin_of_error_clt = z*df_purchase_male['Purchase'].std()/np.sqrt(len(df_purchase_male))
male_sample_mean = df_purchase_male['Purchase'].mean()
male_lower_lim = male_sample_mean - male_margin_of_error_clt
male_upper_lim = male_sample_mean + male_margin_of_error_clt

female_margin_of_error_clt = z*df_purchase_female['Purchase'].std()/np.sqrt(len(df_purchase_female))
female_sample_mean = df_purchase_female['Purchase'].mean()
female_lower_lim = female_sample_mean - female_margin_of_error_clt
female_upper_lim = female_sample_mean + female_margin_of_error_clt

diff_margin_of_error_clt = z*(np.std(diff))/np.sqrt(len(diff))
diff_sample_mean = np.mean(diff)
diff_lower_lim = diff_sample_mean - diff_margin_of_error_clt
diff_upper_lim = diff_sample_mean + diff_margin_of_error_clt



print("Male confidence interval of means: ({:.2f}, {:.2f})".format(male_lower_lim, male_upper_lim))
print("Female confidence interval of means: ({:.2f}, {:.2f})".format(female_lower_lim, female_upper_lim))
print("confidence interval of difference of male and female means: ({:.2f}, {:.2f})".format(diff_lower_lim, diff_upper_lim)
```

```
Male confidence interval of means: (895618.38, 955070.43)
Female confidence interval of means: (673255.48, 750793.30)
confidence interval of difference of male and female means: (211935.10, 215257.65)
```

Now we can infer about the population that, **95% of the times**:

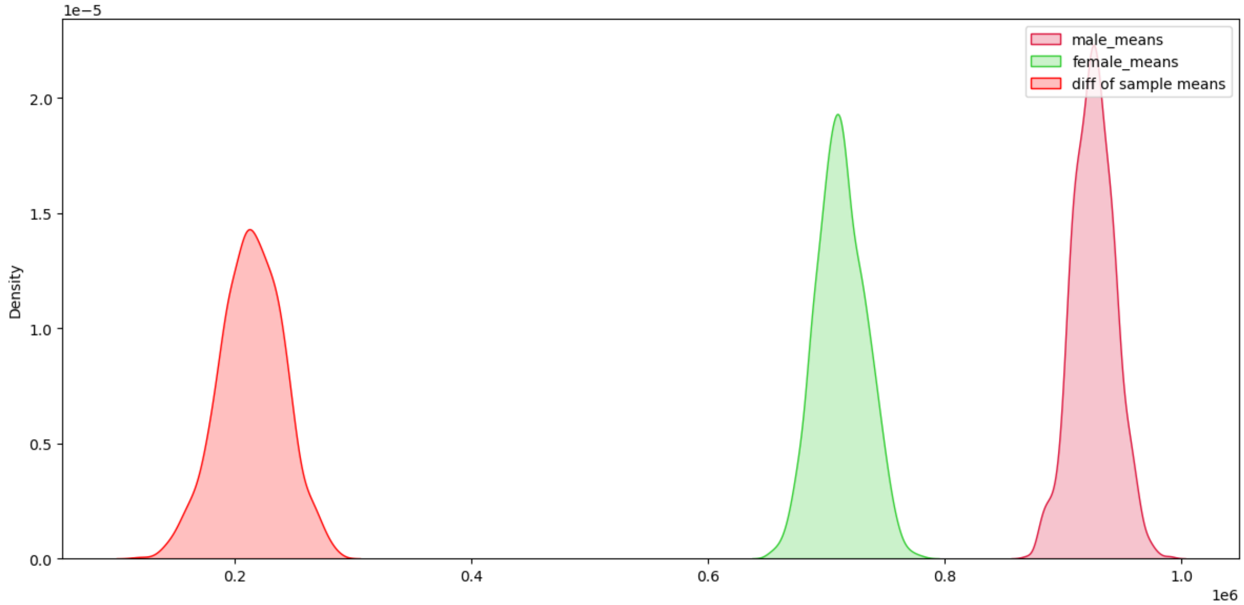
1. Average amount spend by male customer will lie in between: **(895617.83, 955070.97)**
2. Average amount spend by female customer will lie in between: **(673254.77, 750794.02)**

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

Lets see

In [1094]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

fig, ax = plt.subplots(figsize=(12, 6))
sns.kdeplot(data=male_means,
            color='crimson', label='male_means', fill=True, ax=ax)
sns.kdeplot(data=female_means,
            color='limegreen', label='female_means', fill=True, ax=ax)
sns.kdeplot(data=diff,
            color='red', label='diff of sample means', fill=True, ax=ax)
ax.legend()
plt.tight_layout()
plt.show()
```



- Neither the means nor the confidence interval are overlapping
- Also as per the article https://towardsdatascience.com/why-overlapping-confidence-intervals-mean-nothing-about-statistical-significance-48360559900a (https://towardsdatascience.com/why-overlapping-confidence-intervals-mean-nothing-about-statistical-significance-48360559900a) ,**If the 95% CI of the difference contains 0, then there is no difference in age between groups. If it doesn't contain 0, then there is a statistically significant difference between groups**
- As it turns out the difference is statistically significant since the 95% CI (shaded red region) doesn't contain 0.
- More lucrative discounts to lure female customers, special promotional offers on International Women's Day , Mother's Day , Daugther's Day, and on their occassions like Anniversay , Birthdays etc

Are married/unmarried spending more money per transaction than men? Why or Why not?

In [1095]:

```
walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  object
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(4), object(6)
memory usage: 42.0+ MB
```

In [1096]:

```
df_purchase_maritalstatus=walmart.groupby(['User_ID', 'Marital_Status'])[['Purchase']].sum().reset_index()
df_purchase_married=df_purchase_maritalstatus[df_purchase_maritalstatus["Marital_Status"]==1]
print("Married population mean :" ,df_purchase_married["Purchase"].mean())
print("Married population standard deviation :" ,df_purchase_married["Purchase"].std())
df_purchase_unmarried=df_purchase_maritalstatus[df_purchase_maritalstatus["Marital_Status"]==0]
print("Unmarried population mean :" ,df_purchase_unmarried["Purchase"].mean())
print("Unmarried population standard deviation :" ,df_purchase_unmarried["Purchase"].std())
```

```
Married population mean : 843526.7966855295
Married population standard deviation : 935352.1158252308
Unmarried population mean : 880575.7819724905
Unmarried population standard deviation : 949436.249555238
```

- Unmarried people spend more than married people, unmarried people generally focus less on savings(lesser liabilities) hence they might be spending more

Let us assume the confidence interval to be 95% and desired marging of error to be plus-minus 3

In [1097]:

```
df_purchase_maritalstatus["Marital_Status"].value_counts()
```

Out[1097]:

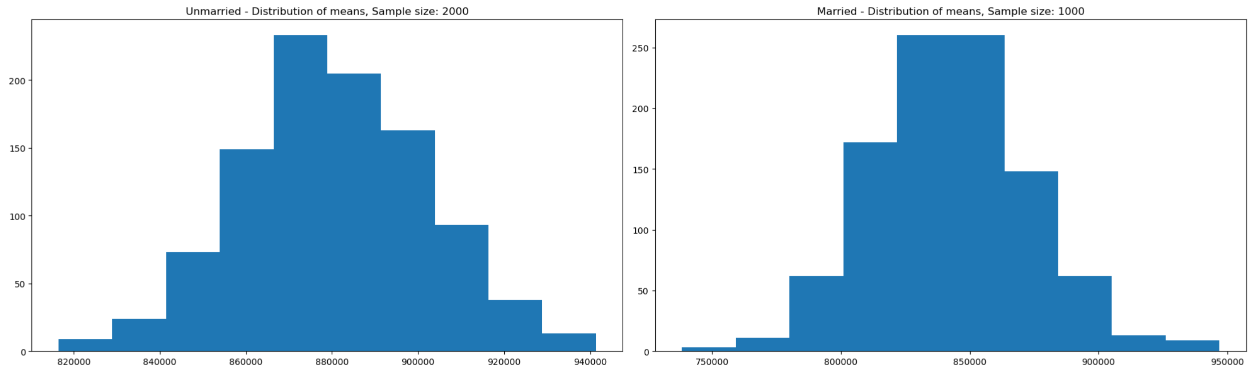
```
0    3417
1    2474
Name: Marital_Status, dtype: int64
```

In [1098]:

```
unmarried_sample_size = 2000
married_sample_size = 1000
num_repetitions = 1000
unmarried_means = []
married_means = []
diff_ms =[]

for _ in range(num_repetitions):
    unmarried_mean = df_purchase_unmarried["Purchase"].sample(unmarried_sample_size, replace=True).mean()
    married_mean = df_purchase_married["Purchase"].sample(married_sample_size, replace=True).mean()
    unmarried_means.append(unmarried_mean)
    married_means.append(married_mean)
    diff_ms.append(unmarried_mean-married_mean)
```

In [1099]:

```
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))

axis[0].hist(unmarried_means)
axis[1].hist(married_means)
axis[0].set_title("Unmarried - Distribution of means, Sample size: 2000")
axis[1].set_title("Married - Distribution of means, Sample size: 1000")

plt.show()
print("Population mean - Mean of sample means of amount spend for Married: {:.2f}".format(np.mean(married_means)))
print("Population mean - Mean of sample means of amount spend for Unmarried: {:.2f}".format(np.mean(unmarried_means)))
```




```
Population mean - Mean of sample means of amount spend for Married: 842832.85
Population mean - Mean of sample means of amount spend for Unmarried: 880427.27
```

Calculating the 95% confidence interval

In [1100]:

```
unmarried_margin_of_error_clt = z*df_purchase_unmarried['Purchase'].std()/np.sqrt(len(df_purchase_unmarried))
unmarried_sample_mean = df_purchase_unmarried['Purchase'].mean()
unmarried_lower_lim = unmarried_sample_mean - unmarried_margin_of_error_clt
unmarried_upper_lim = unmarried_sample_mean + unmarried_margin_of_error_clt

married_margin_of_error_clt = z*df_purchase_married['Purchase'].std()/np.sqrt(len(df_purchase_married))
married_sample_mean = df_purchase_married['Purchase'].mean()
married_lower_lim = married_sample_mean - married_margin_of_error_clt
married_upper_lim = married_sample_mean + married_margin_of_error_clt

diff_ms_margin_of_error_clt = z*(np.std(diff_ms))/np.sqrt(len(diff_ms))
diff_ms_sample_mean = np.mean(diff_ms)
diff_ms_lower_lim = diff_ms_sample_mean - diff_ms_margin_of_error_clt
diff_ms_upper_lim = diff_ms_sample_mean + diff_ms_margin_of_error_clt



print("Unmarried confidence interval of means: ({:.2f}, {:.2f})".format(unmarried_lower_lim, unmarried_upper_lim))
print("Married confidence interval of means: ({:.2f}, {:.2f})".format(married_lower_lim, married_upper_lim))
print("confidence interval of difference of married and unmarried means: ({:.2f}, {:.2f})".format(diff_ms_lower_lim, diff_ms
```

```
Unmarried confidence interval of means: (848741.77, 912409.80)
Married confidence interval of means: (806669.51, 880384.08)
confidence interval of difference of married and unmarried means: (35337.69, 39851.16)
```

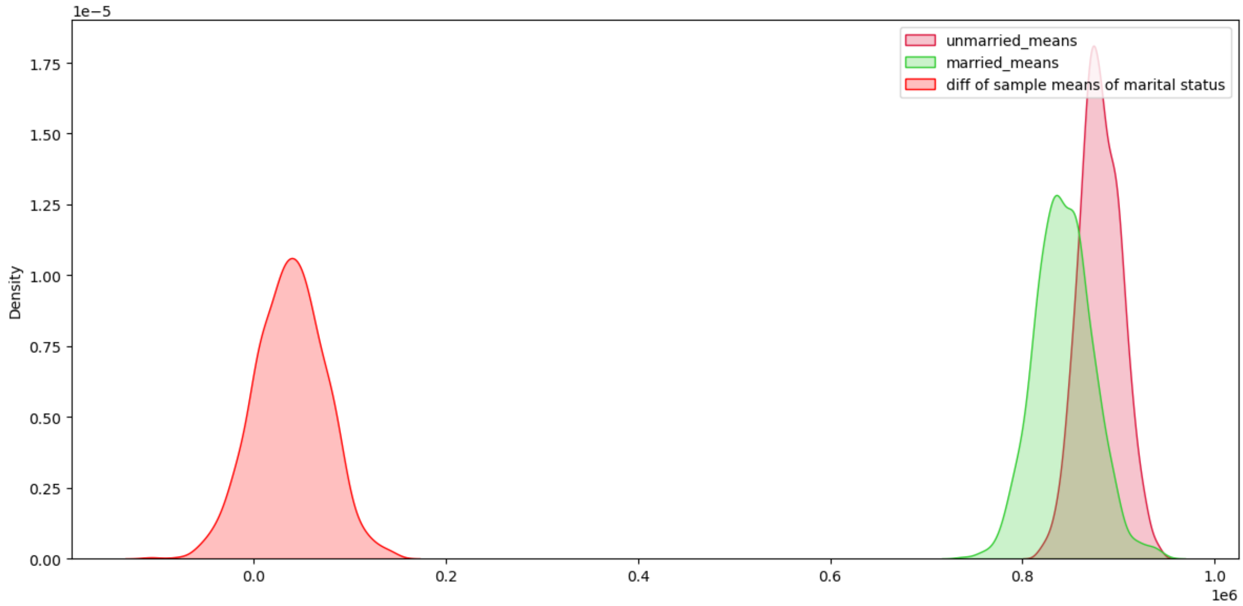
Now we can infer about the population that, **95% of the times**:

1. Average amount spend by unmarried customer will lie in between: **(848741.77, 912409.80)**
2. Average amount spend by married customer will lie in between: **(806669.51, 880384.08)**

Are confidence intervals of average unmarried and married spending overlapping? How can Walmart leverage this conclusion to make changes or improvements?

In [1101]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

fig, ax = plt.subplots(figsize=(12, 6))
sns.kdeplot(data=unmarried_means,
            color='crimson', label='unmarried_means', fill=True, ax=ax)
sns.kdeplot(data=married_means,
            color='limegreen', label='married_means', fill=True, ax=ax)
sns.kdeplot(data=diff_ms,
            color='red', label='diff of sample means of marital status', fill=True, ax=ax)
ax.legend()
plt.tight_layout()
plt.show()
```



- Married and unmarried sample means overlap with each other
- Also as per the article https://towardsdatascience.com/why-overlapping-confidence-intervals-mean-nothing-about-statistical-significance-48360559900a (https://towardsdatascience.com/why-overlapping-confidence-intervals-mean-nothing-about-statistical-significance-48360559900a) ,**If the 95% CI of the difference contains 0, then there is no difference in age between groups. If it doesn't contain 0, then there is a statistically significant difference between groups**
- As it turns out the difference is **statistically insignificant** since the 95% CI (shaded red region) contain 0.

Are younger/older spending more money per transaction than men? Why or Why not?

In [1102]:

```
df_purchase_age["Age"].unique()
```

Out[1102]:

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

In [1103]:

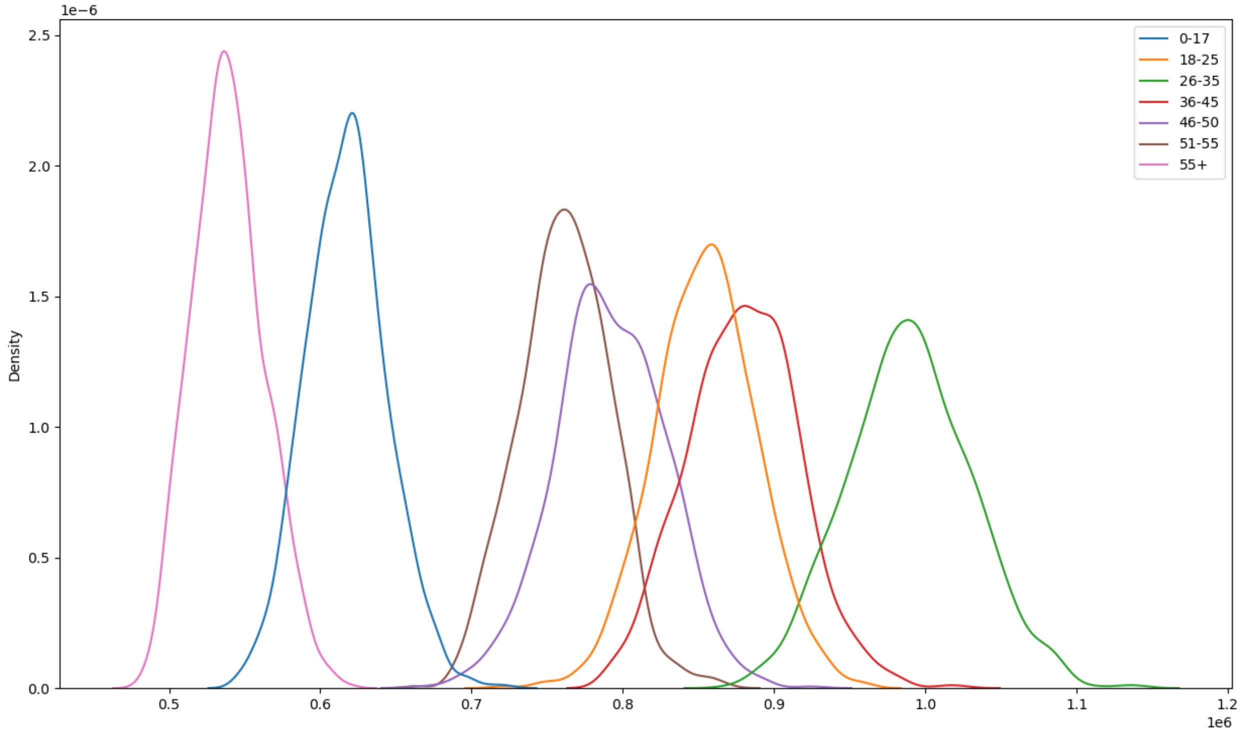
```
df_purchase_age=walmart.groupby(['User_ID', 'Age'])[['Purchase']].sum().reset_index()
sample_size = 700
num_repetitions = 1000

all_means = {}

age_intervals = ['0-17','18-25','26-35','36-45', '46-50', '51-55', '55+']
for age_interval in age_intervals:
    all_means[age_interval] = []

for age_interval in age_intervals:
    for _ in range(num_repetitions):
        mean = df_purchase_age[df_purchase_age['Age']==age_interval].sample(sample_size, replace=True)['Purchase'].mean()
        all_means[age_interval].append(mean)
```

In [1104]:

```
sns.kdeplot(data=all_means)
```

Out[1104]:

```
<Axes: ylabel='Density'>
```



In [1105]:

```
for val in ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']:

    new_df = df_purchase_age[df_purchase_age['Age']==val]

    margin_of_error_clt = 1.96*new_df['Purchase'].std()/np.sqrt(len(new_df))
    sample_mean = new_df['Purchase'].mean()
    lower_lim = sample_mean - margin_of_error_clt
    upper_lim = sample_mean + margin_of_error_clt

    print("For age {} --> confidence interval of means: ({:.2f}, {:.2f})".format(val, lower_lim, upper_lim))
```

```
For age 26-35 --> confidence interval of means: (945034.42, 1034284.21)
For age 36-45 --> confidence interval of means: (823347.80, 935983.62)
For age 18-25 --> confidence interval of means: (801632.78, 908093.46)
For age 46-50 --> confidence interval of means: (713505.63, 871591.93)
For age 51-55 --> confidence interval of means: (692392.43, 834009.42)
For age 55+ --> confidence interval of means: (476948.26, 602446.23)
For age 0-17 --> confidence interval of means: (527662.46, 710073.17)
```

- Customer between age 18-45 are the potential customer, to retain them , monthly/annual memberships plans can be launched with additional discount and wallet cashback
- Home delivery options to senior citizen (51-55+) so that they buy more

- Survey for Mid age customers can be conducted to understand the mindset and offers they are looking for under product category 5,1,8