```
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmar
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/wa
To: /content/walmart_data.csv?1641285094
100% 23.0M/23.0M [00:00<00:00, 28.4MB/s]
```

Introduction

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

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).

```
#importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Basic Analysis

```
#fetching the data frame and viewing the df
df = pd.read_csv('/content/walmart_data.csv?1641285094')
df.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Yea
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
[illegible]	[illegible]	[illegible]	[illegible]	0-	[illegible]	[illegible]	

```
#information about df
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
```

```
#basic desrciptive stats about df
df.describe()
```

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

```
#changing the data type of product userid, category, marital status and occupation
col = ['User_ID', 'Occupation', 'Marital_Status', 'Product_Category']
df[col] = df[col].astype('object')
```

```
#description inclusive of object data types
df.describe(include = 'all')
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Curren
count	550068.0	550068	550068	550068	550068.0	550068	
unique	5891.0	3631	2	7	21.0	3	
top	1001680.0	P00265242	M	26-35	4.0	B	
freq	1026.0	1880	414259	219587	72308.0	231173	
mean	NaN	NaN	NaN	NaN	NaN	NaN	
std	NaN	NaN	NaN	NaN	NaN	NaN	
min	NaN	NaN	NaN	NaN	NaN	NaN	
25%	NaN	NaN	NaN	NaN	NaN	NaN	
50%	NaN	NaN	NaN	NaN	NaN	NaN	

```
#to find the count df NaN values
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
```

```
# fetching the frequency of every single column
for i in df.columns:
  print(i, ':')
  print()
  print(df[i].value_counts())
  print()
```

```
User_ID :

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

Product_ID :

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

Gender :

M    414259
F    135809
Name: Gender, dtype: int64

Age :

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

Occupation :

4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
```

```
#fetching the shape of dataset
df.shape
```

```
(550068, 10)
```

```
# fetching the probbaility of every single column
for i in df.columns:
  print(i, ':')
  print()
```

```
    print(df[i].value_counts(normalize = True))
    print()
```

```
3372      0.000002
855       0.000002
21489     0.000002
Name: Purchase, Length: 18105, dtype: float64
```

```
#to find the no of unique values in each column
for i in df.columns:
  print(i, ':', df[i].nunique())
  print()
```

```
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
```

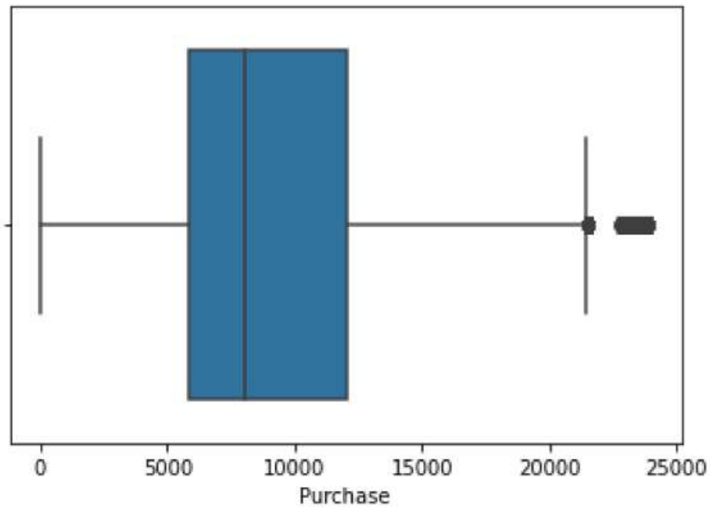
Insights

1. Dataset is pretty clean with no null or Nan values and has got 10 columns and 5+lakh rows
2. There are 5891 unique user ids with 1001680 purchasing the most
3. There are 3631 unique product ids with product id P00265242 being the most purchased
4. More males have purchased the products during black friday
5. People under the age group 26-35 has purchased the most in about 7 available age categories
6. People living in city B and people who are living for an year has purchased more
7. People with the Occupation 4 has purchased more products under about 21 occupation categories present
8. People with marital status 0 and product category 5 has purchased more under about 20 product categories that are available

Univariate Analysis

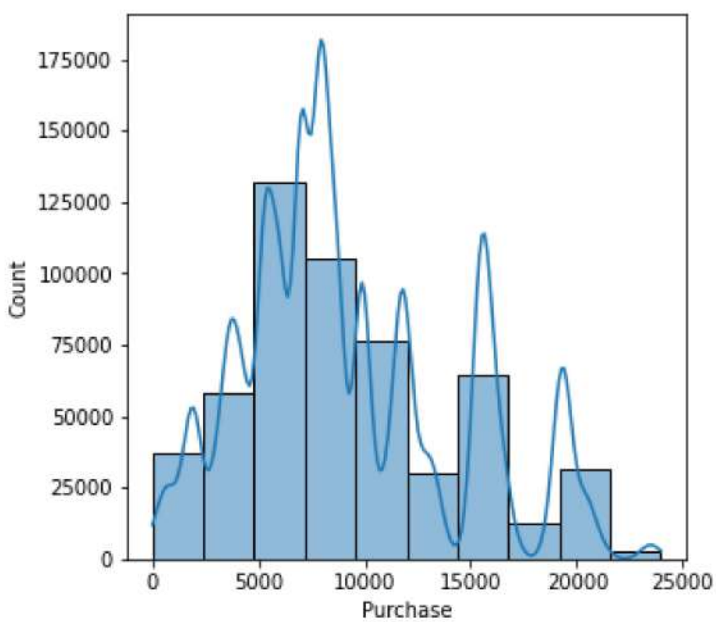
```
#boxplot
sns.boxplot(df['Purchase'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass tł
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc83a909340>
```



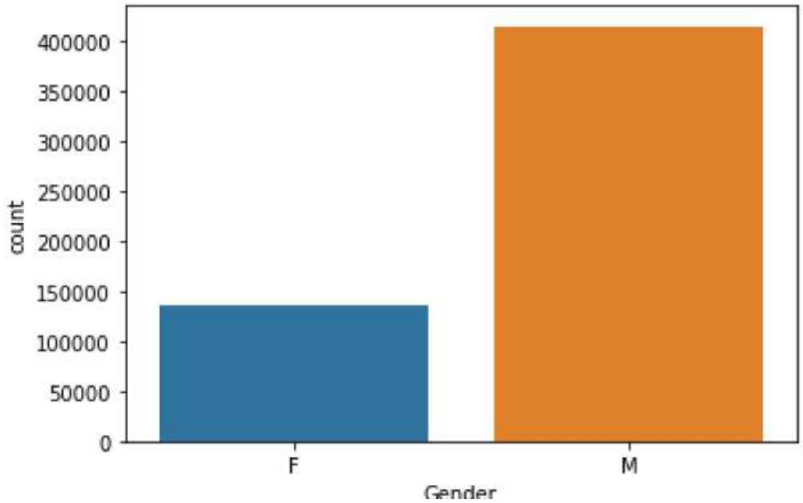

```
#histplot with kde
plt.figure(figsize = (5,5))
sns.histplot(df['Purchase'], kde = True, bins = 10)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc83aeff9a0>
```



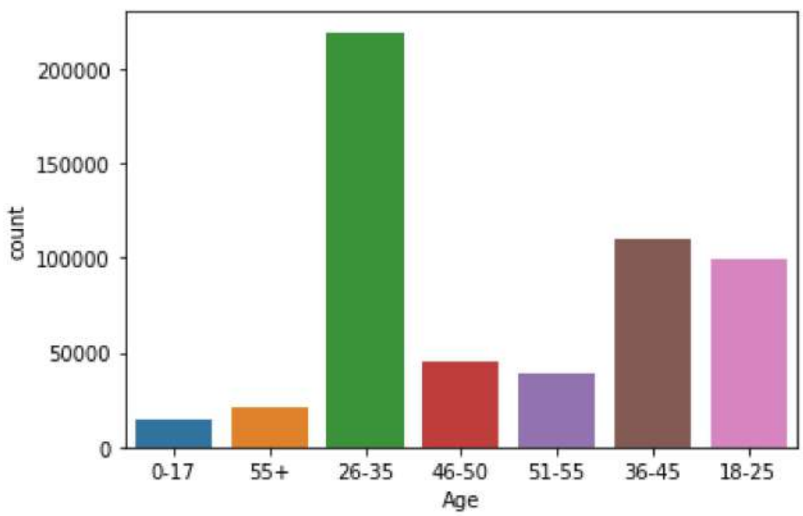

```
#countplot of gender
sns.countplot(df['Gender'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc83af07130>
```




```
#countplot of age
sns.countplot(df['Age'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc83ae39550>
```

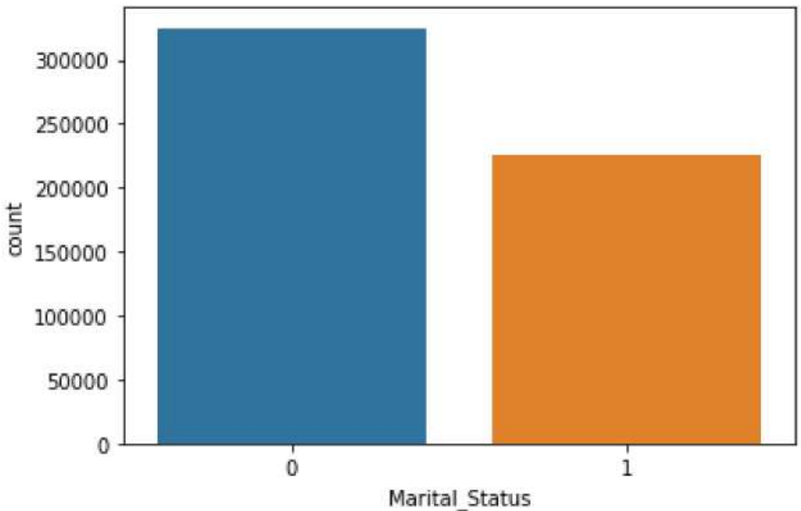



```
#countplot of occupation
sns.countplot(df['Occupation'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass tl
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc83ad97d30>
```


```
#countplot of marital status
sns.countplot(df['Marital_Status'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass tl
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc840e92e80>
```

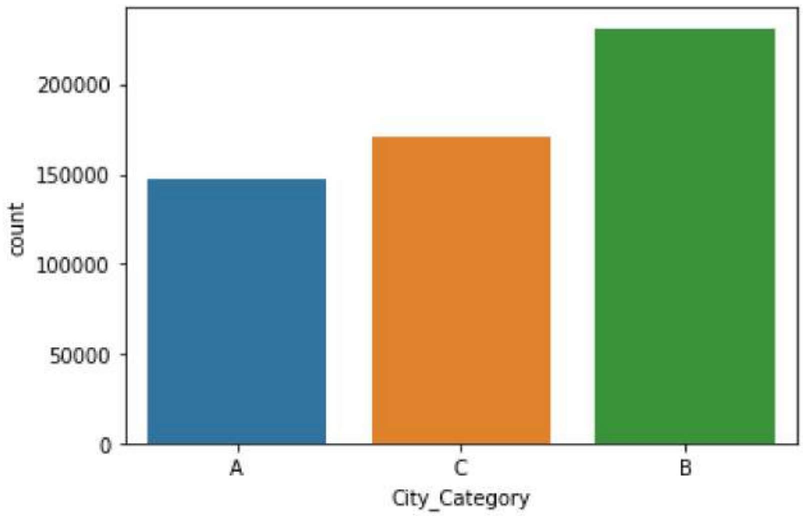



```
#countplot of product category
sns.countplot(df['Product_Category'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
```

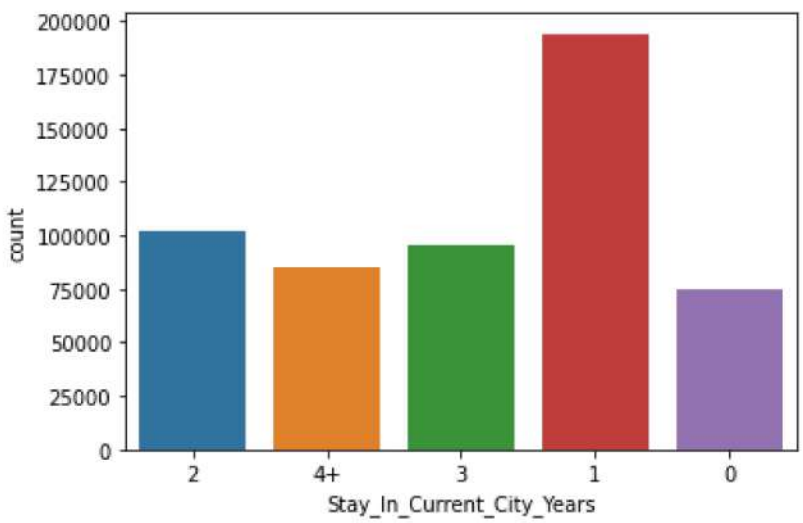
```
#countplot of city category
sns.countplot(df['City_Category'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc83ac3db50>
```




```
#countplot of current city stay
sns.countplot(df['Stay_In_Current_City_Years'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc83ab8ecd0>
```



Insights

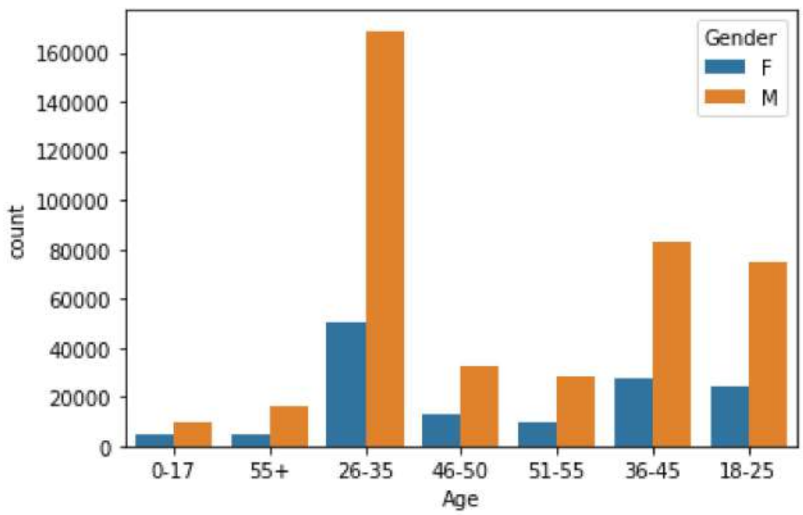
1. 75% males purchase during the black friday sales

2. People in the age group 26-35 purchase more followed by 36-45 and 18-25
3. People in the occupation 4 followed by 0 and 7 purchase the most
4. More unmarried purchase than the married
5. Products under the product category 5 followed by 1 and 8 are being the most purchased
6. People living in city B purchase more products followed by city C and city A
7. People living in the city for an year purchase more products during the sale followed by people living for 2 years, 3 years, 4+ years and finally 0 year

Bivariate Analysis

```
#contplot of age wrt gender
sns.countplot(df['Age'], hue = df['Gender'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc807304370>
```

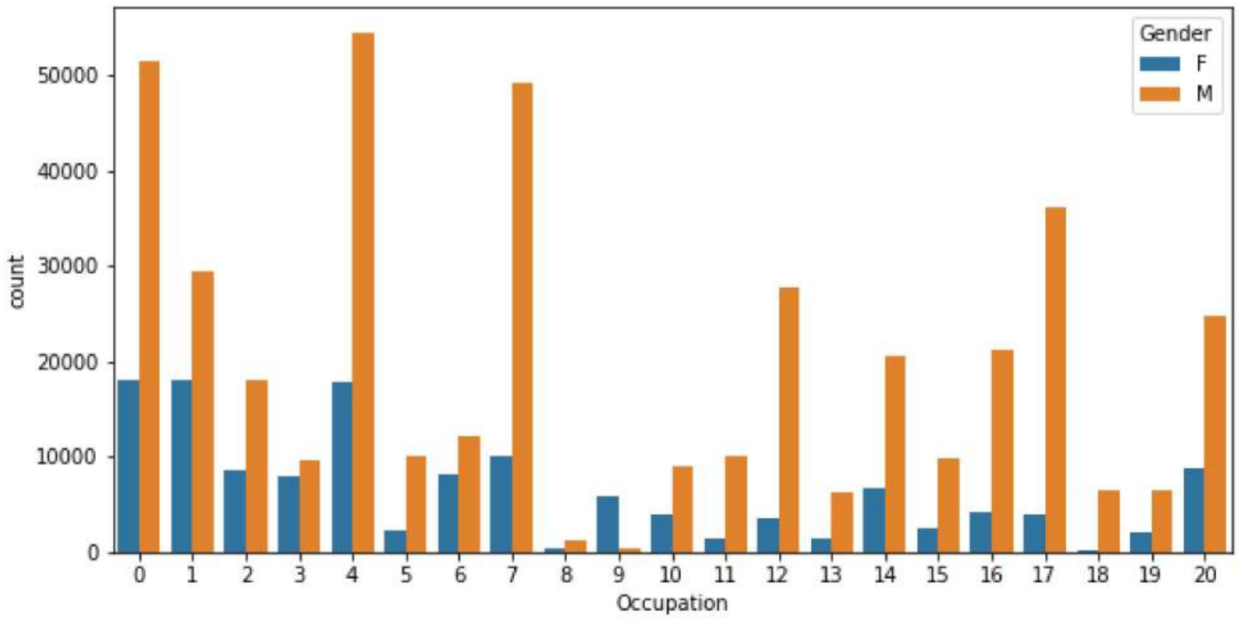



```
#countplot of marital status wrt gender
sns.countplot(df['Marital_Status'], hue = df['Gender'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc818e4fd00>
```


```
#countplot of marital status wrt gender
plt.figure(figsize = (10, 5))
sns.countplot(df['Occupation'], hue = df['Gender'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc804396df0>
```

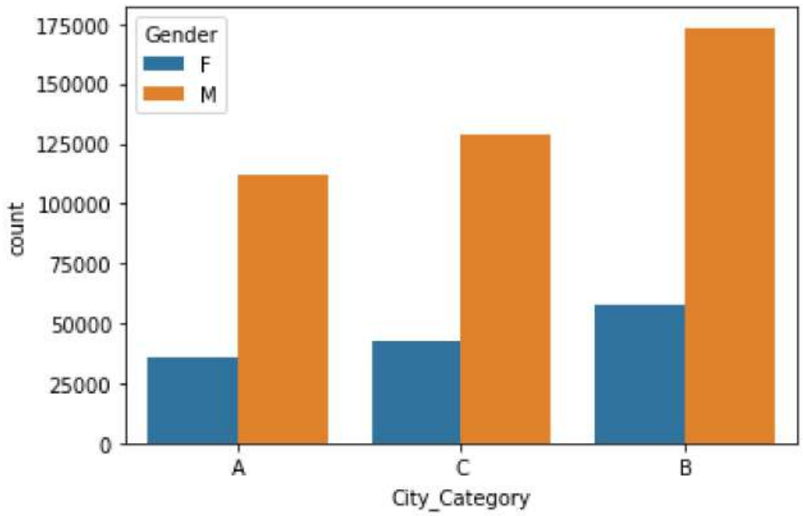



```
#countplot of product category wrt gender
plt.figure(figsize = (10, 5))
sns.countplot(df['Product_Category'], hue = df['Gender'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass tl
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc80e41a8b0>
```


```
#countplot of city category wrt gender
sns.countplot(df['City_Category'], hue = df['Gender'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass tl
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc80eb5f1f0>
```

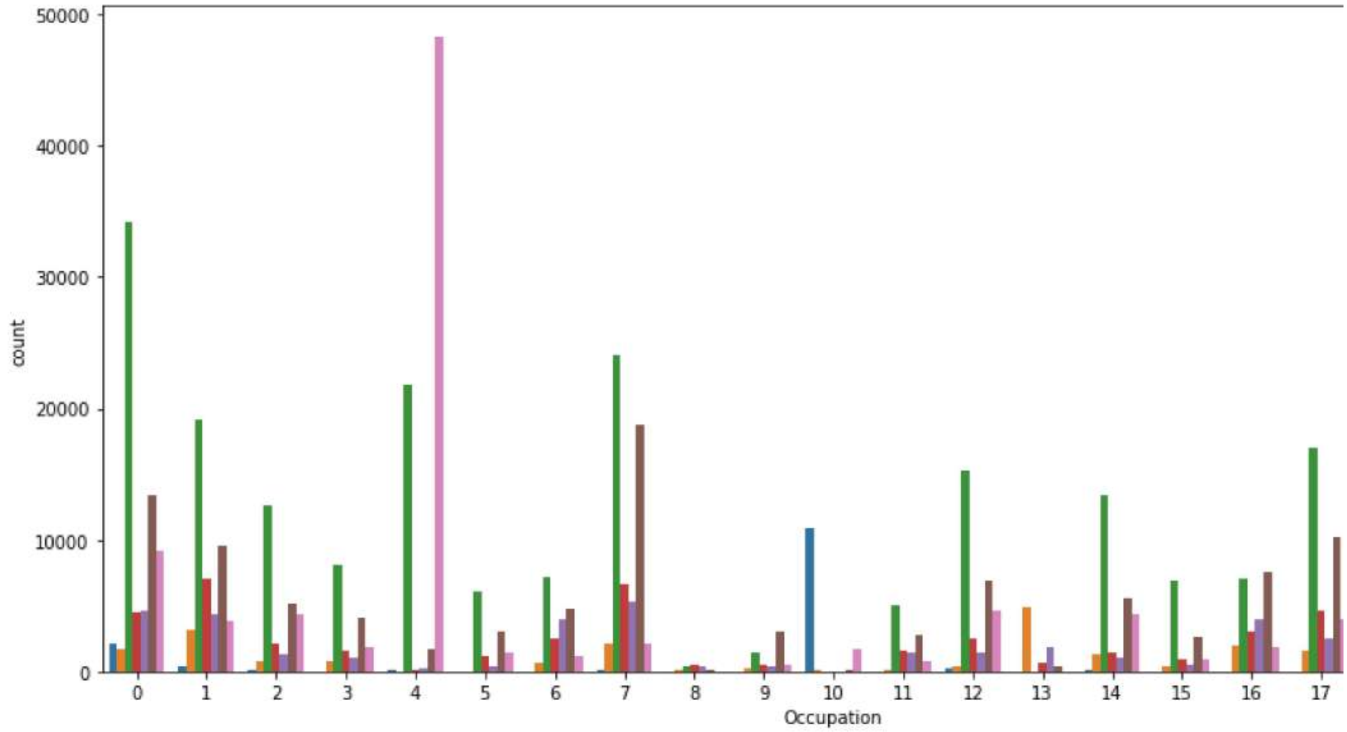



```
#countplot of stay in city wrt gender
sns.countplot(df['Stay_In_Current_City_Years'], hue = df['Gender'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
```

```
#countplot of occupation wrt age
plt.figure(figsize = (15,7))
sns.countplot(df['Occupation'], hue = df['Age'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc80eb56460>
```

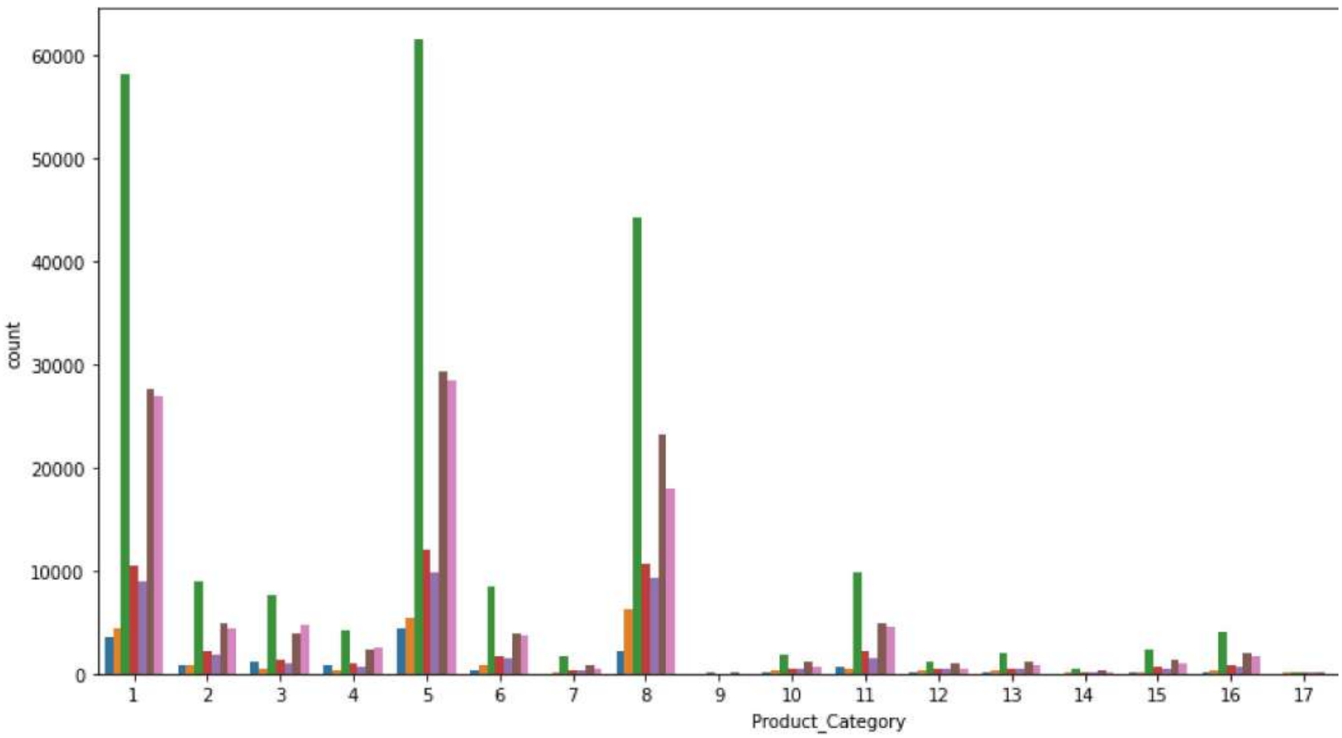



```
#countplot of marital status wrt age
sns.countplot(df['Marital_Status'], hue = df['Age'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc803f16ee0>
```

```
#countplot of product category wrt age
plt.figure(figsize = (15,7))
sns.countplot(df['Product_Category'], hue = df['Age'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc8074b6b80>
```

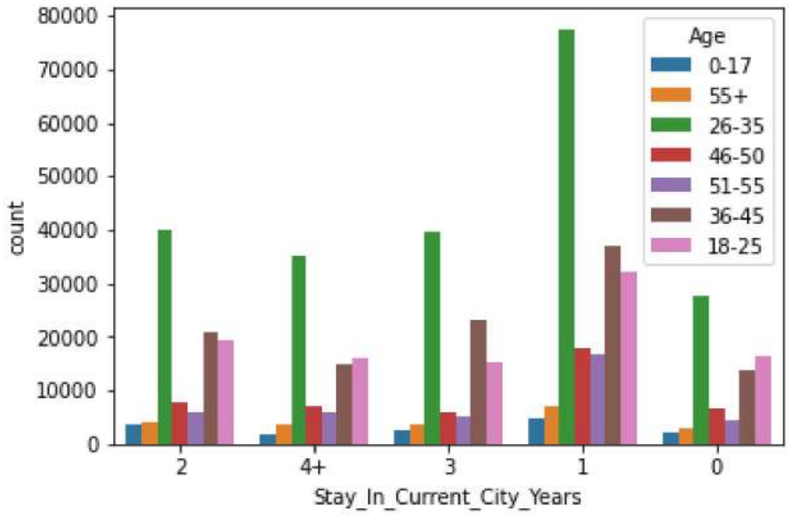



```
#countplot of city category wrt age
sns.countplot(df['City_Category'], hue = df['Age'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass tl
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc803b4f1c0>
```


```
#countplot of stay in city wrt age
sns.countplot(df['Stay_In_Current_City_Years'], hue = df['Age'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass tl
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc803ae6ca0>
```

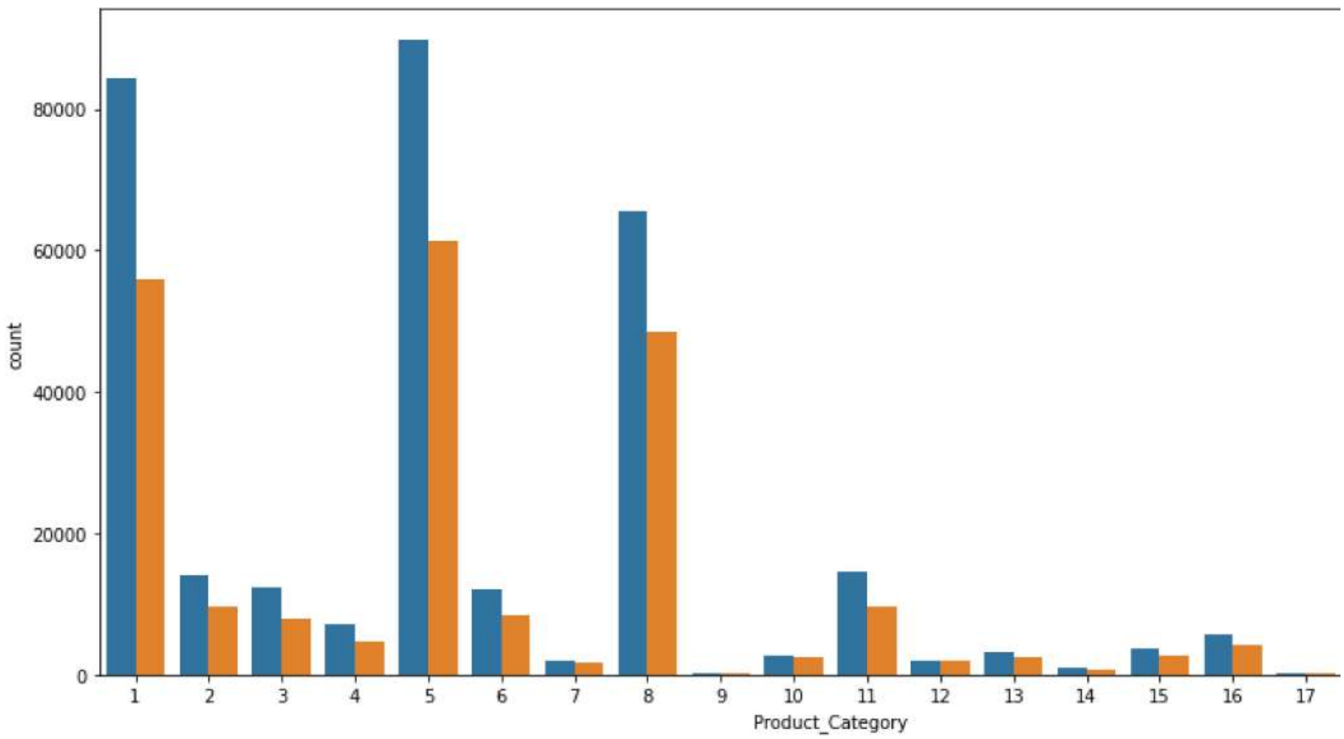



```
#countplot of occupation wrt marital status
plt.figure(figsize = (15,7))
sns.countplot(df['Occupation'], hue = df['Marital_Status'])
```

```
local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fol
nings.warn(
lotlib.axes._subplots.AxesSubplot at 0x7fc8041fbf10>
```

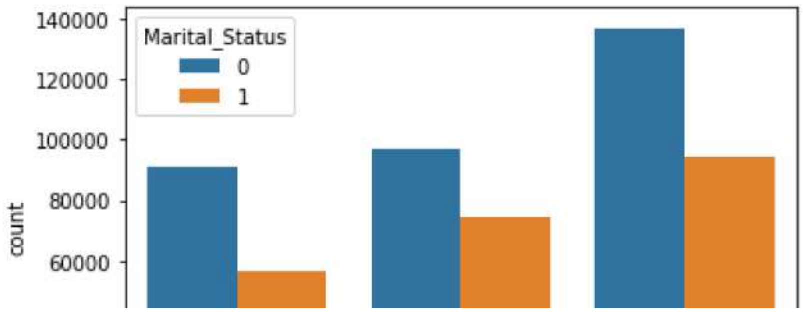
```
#countplot of occupation wrt marital status
plt.figure(figsize = (15,7))
sns.countplot(df['Product_Category'], hue = df['Marital_Status'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc803dd7340>
```



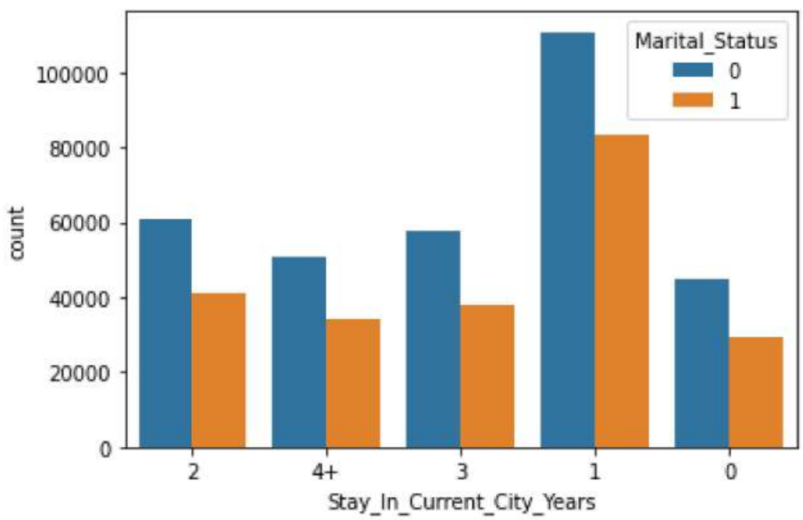

```
#countplot of city category wrt marital status
sns.countplot(df['City_Category'], hue = df['Marital_Status'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc803b95e80>
```




```
#countplot of stay in years wrt marital status
sns.countplot(df['Stay_In_Current_City_Years'], hue = df['Marital_Status'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fc80383b3a0>
```

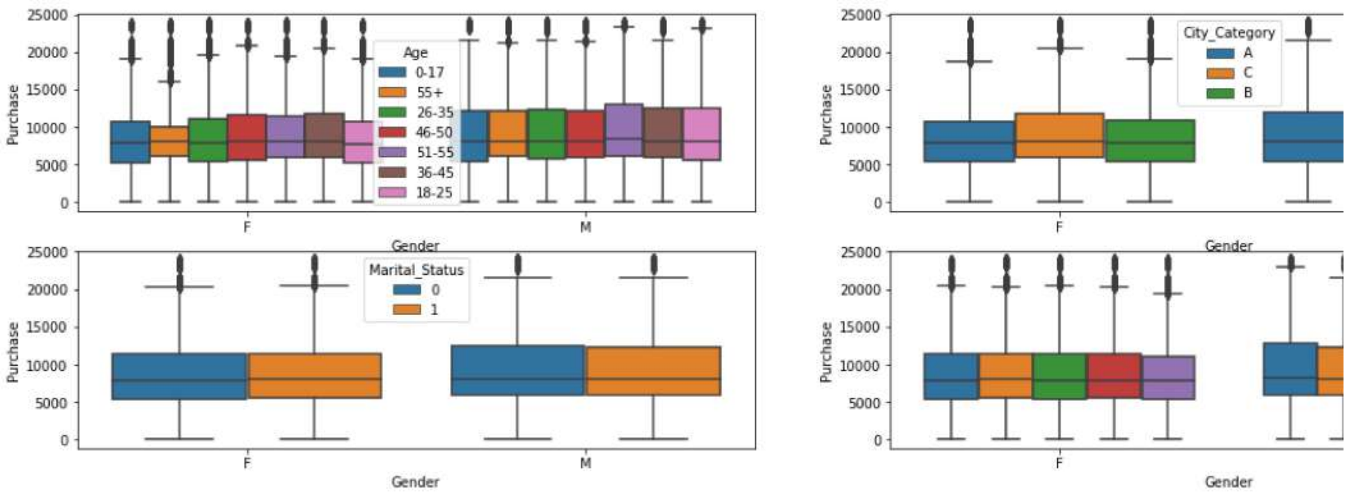


Insights

1. More single males under the age group 26-35 in the occupation 4,0 and 7 buy most products under the product category 1, 5 and 8 from the city B staying for an year
2. People in the age group 18-25 are mostly working in occupation 4
3. People in age group 0-17 are mostly working in occupation 10
4. People in age group 26-35 are mostly working in occupation 0,7,4,1,17,12,20,14,2,3
5. More people are single in the age group 26-35 followed by 18-25
6. Most people in the age group 26-35 buy products in the category 5,1 and 8
7. City B has more people in the age group 26-35 while a has more in the age group 18- 25
8. Most of the people in occupation 4 are single

Multivariate Analysis

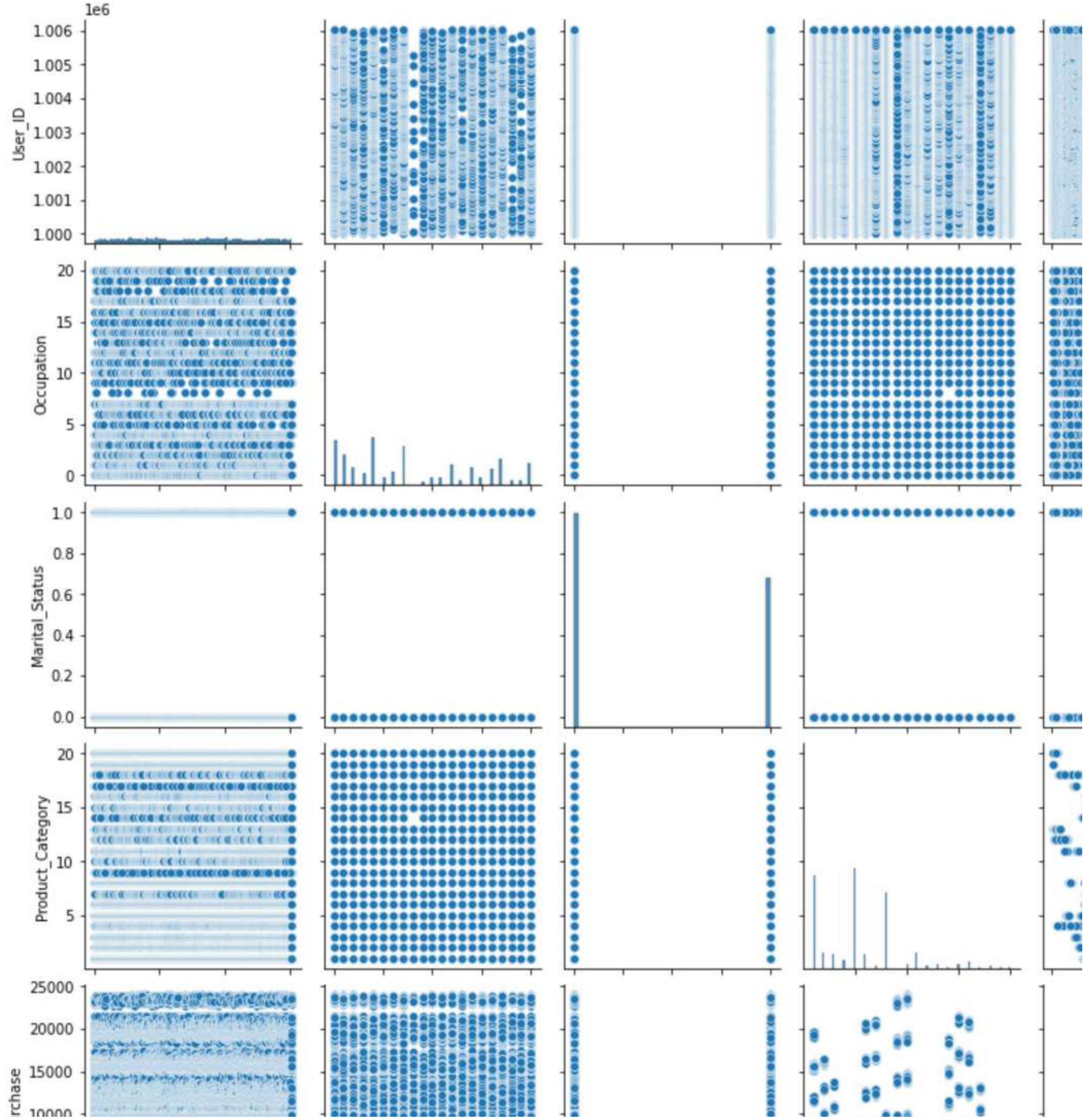
```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20, 6))
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Age', ax=axs[0,0])
sns.boxplot(data=df, y='Purchase', x='Gender', hue='City_Category', ax=axs[0,1])

sns.boxplot(data=df, y='Purchase', x='Gender', hue='Marital_Status', ax=axs[1,0])
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Stay_In_Current_City_Years', ax=axs[1,1])

plt.show()
```



Pair Plot

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7fc802ea9220>
```



Analysis

```
#Total purchase based on gender, age and marital status
df1 = df.groupby(['Gender','Age','Marital_Status'])['Purchase'].sum()
df1
```

```
Gender  Age    Marital_Status
F       0-17   0                  42385978
        18-25  0                 155646801
               1                  49829041
        26-35  0                 258528663
```

```
                 1             184447570
        36-45    0             150275279
                 1              93163684
        46-50    0              27740352
                 1              88966512
        51-55    0              32708873
                 1              56757124
        55+      0              16868181
                 1              28914584
M       0-17     0              92527205
        18-25    0             568273801
                 1             140099032
        26-35    0             974801439
                 1             613992906
        36-45    0             473835481
                 1             309295440
        46-50    0              85918008
                 1             218218531
        51-55    0              71083521
                 1             206550126
        55+      0              58333865
                 1              96650745
Name: Purchase, dtype: int64
```

```
#Total purchase based on gender
df2 = df.groupby(['Gender'])['Purchase'].sum()
df2
```

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

```
#Avg purchase based on gender
df3 = df.groupby(['Gender'])['Purchase'].mean()
df3
```

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

```
#Avg purchase based on age
df4 = df.groupby(['Age'])['Purchase'].mean()
df4
```

```
Age
0-17     8933.464640
18-25    9169.663606
26-35    9252.690633
36-45    9331.350695
46-50    9208.625697
```

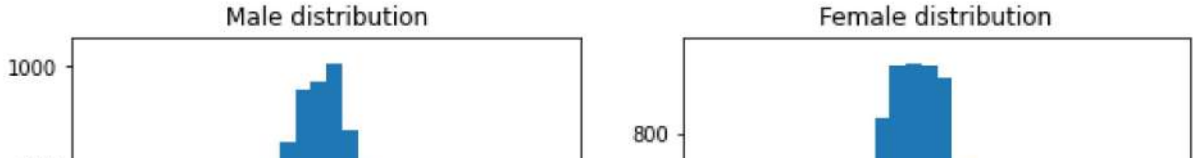
```
51-55    9534.808031
55+      9336.280459
Name: Purchase, dtype: float64
```

```
#Avg purchase based on Marital status
df5 = df.groupby(['Marital_Status'])['Purchase'].mean()
df5
```

```
Marital_Status
0    9265.907619
1    9261.174574
Name: Purchase, dtype: float64
```

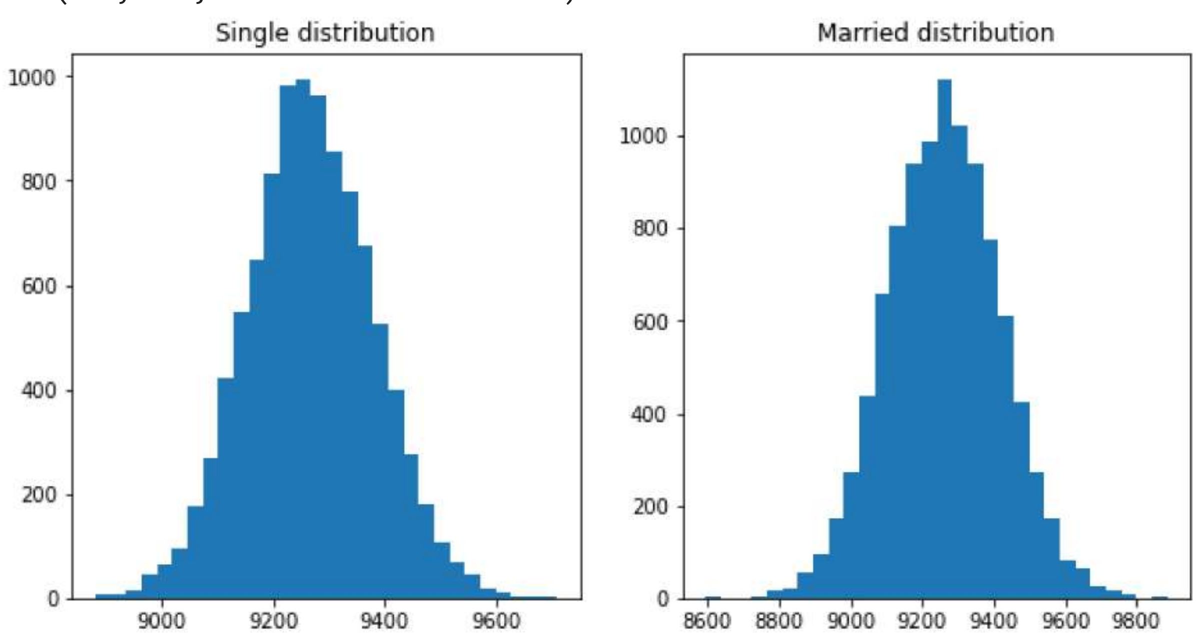
```
#calculating means for hist plot based on gender
df_m = df.loc[df['Gender'] == 'M']
df_f = df.loc[df['Gender'] == 'F']

male_sample_size = 2000
female_sample_size = 1000
m_mean = []
f_mean = []

for j in range(10000):
    male_mean = df_m.sample(male_sample_size, replace=True)['Purchase'].mean()
    female_mean = df_f.sample(female_sample_size, replace=True)['Purchase'].mean()


    m_mean.append(male_mean)
    f_mean.append(female_mean)


#hist of distributions
fig, axs = plt.subplots(nrows = 1, ncols = 2, figsize=(10, 5))
axs[0].hist(m_mean, bins = 30)
axs[1].hist(f_mean, bins = 30)
axs[0].set_title('Male distribution')
axs[1].set_title('Female distribution')
```

```
Text(0.5, 1.0, 'Female distribution')
```




```
#calculating means for hist plot based on marital status
df_s = df.loc[df['Marital_Status'] == 0]
df_ma = df.loc[df['Marital_Status'] == 1]

s_sample_size = 2000
ma_sample_size = 1000
s_mean = []
ma_mean = []

for j in range(10000):
    s_means = df_s.sample(s_sample_size, replace=True)['Purchase'].mean()
    ma_means = df_ma.sample(ma_sample_size, replace=True)['Purchase'].mean()

    s_mean.append(s_means)
    ma_mean.append(ma_means)


#hist of distributions
fig, axs = plt.subplots(nrows = 1, ncols = 2, figsize=(10, 5))
axs[0].hist(s_mean, bins = 30)
axs[1].hist(ma_mean, bins = 30)
axs[0].set_title('Single distribution')
axs[1].set_title('Married distribution')
```

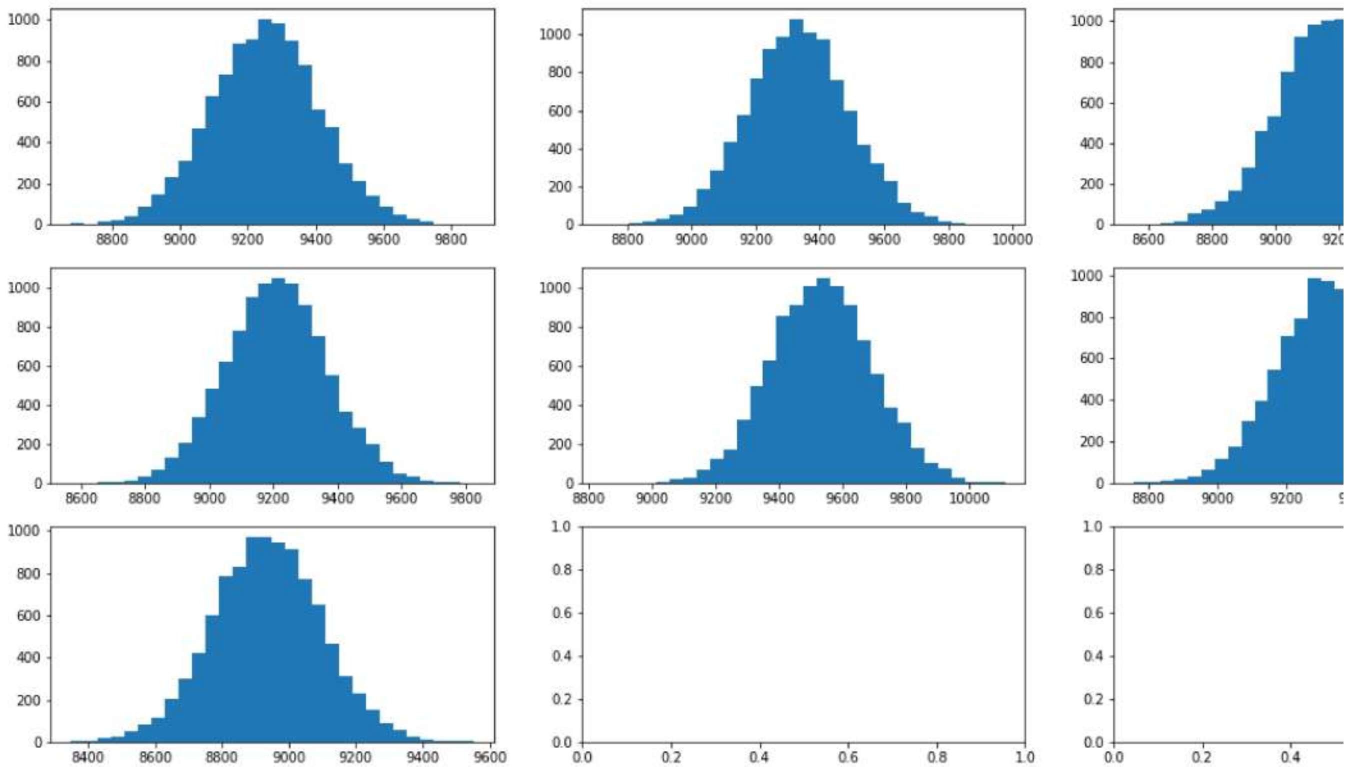
```
Text(0.5, 1.0, 'Married distribution')
```




```
#calculating means for hist plot based on Age
df_26 = df.loc[df['Age'] == '26-35']
```

```
df_36 = df.loc[df['Age'] == '36-45']
df_18 = df.loc[df['Age'] == '18-25']
df_46 = df.loc[df['Age'] == '46-50']
df_51 = df.loc[df['Age'] == '51-55']
df_55 = df.loc[df['Age'] == '55+']
df_17 = df.loc[df['Age'] == '0-17']

mean26 = []
mean36 = []
mean18 = []
mean46 = []
mean51 = []
mean55 = []
mean17 = []

for j in range(10000):
    means26 = df_26.sample(1000, replace=True)['Purchase'].mean()
    means36 = df_36.sample(1000, replace=True)['Purchase'].mean()
    means18 = df_18.sample(1000, replace=True)['Purchase'].mean()
    means46 = df_46.sample(1000, replace=True)['Purchase'].mean()
    means51 = df_51.sample(1000, replace=True)['Purchase'].mean()
    means55 = df_55.sample(1000, replace=True)['Purchase'].mean()
    means17 = df_17.sample(1000, replace=True)['Purchase'].mean()

    mean26.append(means26)
    mean36.append(means36)
    mean18.append(means18)
    mean46.append(means46)
    mean51.append(means51)
    mean55.append(means55)
    mean17.append(means17)

#hist of distributions
fig, axs = plt.subplots(nrows = 3, ncols = 3, figsize=(20, 10))
axs[0,0].hist(mean26, bins = 30)
axs[0,1].hist(mean36, bins = 30)
axs[0,2].hist(mean18, bins = 30)
axs[1,0].hist(mean46, bins = 30)
axs[1,1].hist(mean51, bins = 30)
axs[1,2].hist(mean55, bins = 30)
axs[2,0].hist(mean17, bins = 30)
```

```
(array([  3.,   4.,  15.,  25.,  48.,  83., 111., 201., 299., 422., 597.,
        786., 829., 973., 967., 946., 910., 771., 653., 466., 312., 226.,
        155.,  91.,  58.,  22.,  11.,   8.,   5.,   3.]),
 array([8347.613, 8387.754, 8427.895, 8468.036, 8508.177, 8548.318,
        8588.459, 8628.6  , 8668.741, 8708.882, 8749.023, 8789.164,
        8829.305, 8869.446, 8909.587, 8949.728, 8989.869, 9030.01 ,
        9070.151, 9110.292, 9150.433, 9190.574, 9230.715, 9270.856,
        9310.997, 9351.138, 9391.279, 9431.42 , 9471.561, 9511.702,
        9551.843]),
 <a list of 30 Patch objects>)
```



```
#90% CI based on gender for male
np.percentile(m_mean, [5,95])
```

```
array([9250.019075, 9623.655475])
```

```
#95% CI based on gender for male
np.percentile(m_mean, [2.5,97.5])
```

```
array([9219.0369625, 9660.858575 ])
```

```
#99% CI based on gender for male
np.percentile(m_mean, [0.5,99.5])
```

```
array([9148.142745 , 9730.1469425])
```

```
#90% CI based on gender for female
np.percentile(f_mean, [5,95])
```

```
array([8487.106 , 8984.2443])
```

```
#95% CI based on gender for female
np.percentile(f_mean, [2.5,97.5])
```

```
array([8439.76335 , 9033.580775])
```

```
#99% CI based on gender for female
np.percentile(f_mean, [0.5,99.5])
```

```
array([8349.7483  , 9128.479475])
```

```
#90% CI based on marital status for single
print(np.percentile(s_mean, [5,95]))
#95% CI based on marital status for single
print(np.percentile(s_mean, [2.5,97.5]))
#99% CI based on marital status for single
print(np.percentile(s_mean, [0.5,99.5]))
```

```
[9085.9624  9455.09305]
[9050.8329125 9491.5221625]
[8980.26803  9559.647385]
```

```
#90% CI based on marital status for married
print(np.percentile(ma_mean, [5,95]))
#95% CI based on marital status for married
print(np.percentile(ma_mean, [2.5,97.5]))
#99% CI based on marital status for married
print(np.percentile(ma_mean, [0.5,99.5]))
```

```
[9006.62545 9519.916  ]
[8957.116425 9567.961125]
[8866.88303  9673.773885]
```

```
#90% CI based on age for group 26-35
print(np.percentile(mean26, [5,95]))
#95% CI based on age for group 26-35
print(np.percentile(mean26, [2.5,97.5]))
#99% CI based on age for group 26-35
print(np.percentile(mean26, [0.5,99.5]))
```

```
[8986.9044  9514.56395]
[8940.21775  9569.520675]
[8848.981645 9663.991295]
```

```
#90% CI based on age for group 36-45
print(np.percentile(mean36, [5,95]))
#95% CI based on age for group 36-45
print(np.percentile(mean36, [2.5,97.5]))
#99% CI based on age for group 36-45
print(np.percentile(mean36, [0.5,99.5]))
```

```
[9075.3269  9597.04255]
[9028.0629  9642.67705]
[8918.957725 9751.150905]
```

```
#90% CI based on age for group 18-25
print(np.percentile(mean18, [5,95]))
#95% CI
print(np.percentile(mean18, [2.5,97.5]))
#99% CI
print(np.percentile(mean18, [0.5,99.5]))
```

```
[8903.46385 9437.18455]
[8844.7129 9486.5602]
[8744.507215 9579.459135]
```

```
#90% CI based on age for group 46-50
print(np.percentile(mean46, [5,95]))
#95% CI
print(np.percentile(mean46, [2.5,97.5]))
#99% CI
print(np.percentile(mean46, [0.5,99.5]))
```

```
[8948.54585 9474.20435]
[8899.385875 9522.539425]
[8810.1534   9623.680405]
```

```
#90% CI based on age for group 51-55
print(np.percentile(mean51, [5,95]))
#95% CI
print(np.percentile(mean51, [2.5,97.5]))
#99% CI
print(np.percentile(mean51, [0.5,99.5]))
```

```
[9275.7987  9800.23595]
[9222.291525 9850.47275 ]
[9124.50299 9938.29212]
```

```
#90% CI based on age for group 55+
print(np.percentile(mean55, [5,95]))
#95% CI
print(np.percentile(mean55, [2.5,97.5]))
#99% CI
print(np.percentile(mean55, [0.5,99.5]))
```

```
[9079.4316 9595.4218]
[9028.2728  9644.83735]
[8932.62729  9748.174945]
```

```
#90% CI based on age for group 0-17
print(np.percentile(mean17, [5,95]))
#95% CI
print(np.percentile(mean17, [2.5,97.5]))
#99% CI
print(np.percentile(mean17, [0.5,99.5]))
```

```
[8670.4007  9203.38705]
[8618.18275  9252.826525]
[8512.586355 9348.71895 ]
```

Insights

1. Average money spent by female is 8734.56 and male is 9437.53
2. Average money spent by single is 9265.61 and married is 9261.17
3. Average money spent by people on various age groups is as follows: 0-17 : 8933.464640, 18-25 : 9169.663606, 26-35 : 9252.690633, 36-45 : 9331.350695, 46-50 : 9208.625697, 51-55 : 9534.808031, 55 plus : 9336.280459
4. 90% CI based on gender for male is ([9250.019075, 9623.655475]), 95% CI is ([9219.0369625, 9660.858575]) and 99% is ([9148.142745 , 9730.1469425])
5. 90%, 95% and 99% CI based on gender for female is as follows: ([8487.106 , 8984.2443]), ([8487.106 , 8984.2443]), ([8349.7483 , 9128.479475])
6. 90%, 95% and 99% CI based on marital status for single is as follows: [9085.9624,9455.09305], [9050.8329125,9491.5221625], [8980.26803,9559.647385]
7. 90%, 95% and 99% CI based on marital status for married is as follows: [9006.62545,9519.916], [8957.116425,9567.961125], [8866.88303,9673.773885]
8. 90%, 95% and 99% CI for age group 26-35 is as follows: [8986.9044,9514.56395], [8940.21775,9569.520675], [8848.981645,9663.991295]
9. 90%, 95% and 99% CI for age group 36-45 is as follows: [9075.3269,9597.04255], [9028.0629,9642.67705], [8918.957725,9751.150905]
10. 90%, 95% and 99% CI for age group 18-25 is as follows: [8903.46385,9437.18455], [8844.7129,9486.5602], [8744.507215,9579.459135]

11. 90%, 95% and 99% CI for age group 46-50 is as follows: [8948.54585,9474.20435], [8899.385875,9522.539425], [8810.1534,9623.680405]
12. 90%, 95% and 99% CI for age group 51-55 is as follows: [9275.7987,9800.23595], [9222.291525,9850.47275], [9124.50299,9938.29212]
13. 90%, 95% and 99% CI for age group 55 plus is as follows: [9079.4316,9595.4218], [9028.2728,9644.83735], [8932.62729,9748.174945]
14. 90%, 95% and 99% CI for age group 0-17 plus is as follows: [8670.4007,9203.38705], [8618.18275,9252.826525], [8512.586355,9348.71895]

Recommendations

1. The product P00265242 can be stocked up more since it is the most purchased
2. More single males purchase during the black friday sales under the age group 26-35. So, the products that attract this group could be made more available
3. City B can be stocked up with products that attract more in the age group 26-35 and city A could be stocked up by products that attract the age group 18-25. However, city B could alwasy be stocked up with more products since there are more people purchasing over there
4. More products under the category 1,5 and 8 could be made most available since they are the most sold out
5. Products that attract people in the occupation 4,0 and 7 can be made more available
6. Products could be mixed and dislayed so that none could miss a product and would be more likely to get more