walmart-case-study

July 5, 2023

#Walmart Business Case Study

Business Problem:

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

[72]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[73]:
```
df = pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/
↪000/001/293/original/walmart_data.csv")
```

[74]:
```
df.head()
```

[74]:
```
   User_ID Product_ID Gender   Age  Occupation City_Category  \
0  1000001  P00069042      F  0-17          10             A
1  1000001  P00248942      F  0-17          10             A
2  1000001  P00087842      F  0-17          10             A
3  1000001  P00085442      F  0-17          10             A
4  1000002  P00285442      M   55+          16             C

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

[75]:
```
df.shape
```

[75]:
```
(550068, 10)
```

[76]:
```
df.describe(include = "all")
```

[76]:

```
                User_ID Product_ID  Gender     Age     Occupation City_Category  \
count   5.500680e+05     550068  550068  550068  550068.000000        550068
unique           NaN       3631       2       7            NaN             3
top              NaN  P00265242       M   26-35            NaN             B
freq             NaN       1880  414259  219587            NaN        231173
mean    1.003029e+06        NaN     NaN     NaN       8.076707           NaN
std     1.727592e+03        NaN     NaN     NaN       6.522660           NaN
min     1.000001e+06        NaN     NaN     NaN       0.000000           NaN
25%     1.001516e+06        NaN     NaN     NaN       2.000000           NaN
50%     1.003077e+06        NaN     NaN     NaN       7.000000           NaN
75%     1.004478e+06        NaN     NaN     NaN      14.000000           NaN
max     1.006040e+06        NaN     NaN     NaN      20.000000           NaN

       Stay_In_Current_City_Years  Marital_Status  Product_Category  \
count                      550068   550068.000000     550068.000000
unique                          5             NaN               NaN
top                             1             NaN               NaN
freq                       193821             NaN               NaN
mean                          NaN        0.409653          5.404270
std                           NaN        0.491770          3.936211
min                           NaN        0.000000          1.000000
25%                           NaN        0.000000          1.000000
50%                           NaN        0.000000          5.000000
75%                           NaN        1.000000          8.000000
max                           NaN        1.000000         20.000000

             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
```

[77]:

```
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
```

[78]:
```
#Checking the null values
df.isnull().sum()/len(df) * 100
```

[78]:
```
User_ID                       0.0
Product_ID                    0.0
Gender                        0.0
Age                           0.0
Occupation                    0.0
City_Category                 0.0
Stay_In_Current_City_Years    0.0
Marital_Status                0.0
Product_Category              0.0
Purchase                      0.0
dtype: float64
```

#Initial Observations:

1. There are no missing values in the data.
2. There are 3631 unique product IDs in the dataset. P00265242 is the most sold Product ID.
3. There are 7 unique age groups and most of the purchase belongs to age 26-35 group.
4. There are 3 unique citi categories with category B being the highest.
5. unique values for Stay_in_current_citi_years with 1 being the highest.
6. The difference between mean and median seems to be significant for purchase that suggests outliers in the data.
7. Minimum & Maximum purchase is 12 and 23961 suggests the purchasing behaviour is quite spread over a aignificant range of values. Mean is 9264 and 75% of purchase is of less than or equal to 12054. It suggest most of the purchase is not more than 12k.
8. Few categorical variable are of integer data type. It can be converted to character type.
9. Out of 550068 data points, 414259's gender is Male and rest are the female. Male purchase count is much higher than female.
10. Standard deviation for purchase have significant value which suggests data is more spread out for this attribute.

[79]:
```
df.groupby("Marital_Status")["Purchase"].sum()
```

[79]:
```
Marital_Status
0    3008927447
1    2086885295
Name: Purchase, dtype: int64
```

[80]:
```
df.groupby("Gender")["Purchase"].sum()
```

[80]:
```
Gender
F    1186232642
M    3909580100
Name: Purchase, dtype: int64
```

[81]:
```
df.groupby("Product_Category")["Purchase"].sum()
```

[81]:
```
Product_Category
1     1910013754
2      268516186
3      204084713
4       27380488
5      941835229
6      324150302
7       60896731
8      854318799
9        6370324
10     100837301
11     113791115
12       5331844
13       4008601
14      20014696
15      92969042
16     145120612
17       5878699
18       9290201
19         59378
20        944727
Name: Purchase, dtype: int64
```

[82]:
```
columns=['User_ID','Occupation', 'Marital_Status', 'Product_Category']
df[columns]=df[columns].astype('object')
```

[83]:
```
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  object
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  object
 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  object
 9   Purchase                    550068 non-null  int64
dtypes: int64(1), object(9)
memory usage: 42.0+ MB
```

[84]:

```python
# Checking how categorical variables contributes to the entire data
categ_cols = ['Gender', 'Age', 'City_Category', 'Stay_In_Current_City_Years',␣
 ↪'Marital_Status']
df[categ_cols].melt().groupby(['variable', 'value'])[['value']].count()/len(df)
```

[84]:

```
                                     value
variable                   value
Age                        0-17   0.027455
                           18-25  0.181178
                           26-35  0.399200
                           36-45  0.199999
                           46-50  0.083082
                           51-55  0.069993
                           55+    0.039093
City_Category              A      0.268549
                           B      0.420263
                           C      0.311189
Gender                     F      0.246895
                           M      0.753105
Marital_Status             0      0.590347
                           1      0.409653
Stay_In_Current_City_Years 0      0.135252
                           1      0.352358
                           2      0.185137
                           3      0.173224
                           4+     0.154028
```

#Observations:

1. 40% of the purchase done by aged 26-35 and 78% purchase are done by the customers aged between the age 18-45 (40%: 26-35, 18%: 18-25, 20%: 36-45)
2. 75% of the purchase count are done by Male and 25% by Female
3. 60% Single, 40% Married contributes to the purchase count.

4. 35% Staying in the city from 1 year, 18% from 2 years, 17% from 3 years
5. There are 20 product categories in total.
6. There are 20 different types of occupations in the city.

[85]:
```
#Checking how the data is spread basis distinct users

df2=df.groupby(['User_ID'])['Age'].unique()
df2.value_counts()/len(df2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.
↪PyObjectHashTable.map_locations()

TypeError: unhashable type: 'numpy.ndarray'
```

```
Exception ignored in: 'pandas._libs.index.IndexEngine._call_map_locations'
Traceback (most recent call last):
  File "pandas\_libs\hashtable_class_helper.pxi", line 5231, in
pandas._libs.hashtable.PyObjectHashTable.map_locations
TypeError: unhashable type: 'numpy.ndarray'
```

[85]:
```
[26-35]    0.348498
[36-45]    0.198099
[18-25]    0.181463
[46-50]    0.090137
[51-55]    0.081650
[55+]      0.063147
[0-17]     0.037006
Name: Age, dtype: float64
```

#Observation:

1. We can see 35% of the users are aged 26-35. 73% of users are aged between 18-45.
2. From the previous observation we saw 40% of the purchase are done by users aged 26-35. And, we have 35% of users aged between 26-35 and they are contributing 40% of total purchase count.So, we can infer users aged 26-35 are more frequent customers.

[86]:
```
df2=df.groupby(['User_ID'])['Gender'].unique()
df2.value_counts()/len(df2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.
↪PyObjectHashTable.map_locations()

TypeError: unhashable type: 'numpy.ndarray'
```

```
Exception ignored in: 'pandas._libs.index.IndexEngine._call_map_locations'
Traceback (most recent call last):
  File "pandas\_libs\hashtable_class_helper.pxi", line 5231, in
pandas._libs.hashtable.PyObjectHashTable.map_locations
TypeError: unhashable type: 'numpy.ndarray'
```

[86]:
```
[M]    0.717196
[F]    0.282804
Name: Gender, dtype: float64
```

#Observation:

1. We have 72% male users and 28% female users. Combining with previous observations we can see 72% of male users contributing to 75% of the purchase count and 28% of female users are contributing to 25% of the purchase count.

[87]:
```
df2=df.groupby(['User_ID'])['Marital_Status'].unique()
df2.value_counts()/len(df2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.
↪PyObjectHashTable.map_locations()

TypeError: unhashable type: 'numpy.ndarray'
```

```
Exception ignored in: 'pandas._libs.index.IndexEngine._call_map_locations'
Traceback (most recent call last):
  File "pandas\_libs\hashtable_class_helper.pxi", line 5231, in
pandas._libs.hashtable.PyObjectHashTable.map_locations
TypeError: unhashable type: 'numpy.ndarray'
```

[87]:
```
[0]    0.580037
[1]    0.419963
Name: Marital_Status, dtype: float64
```

#Observation:

1. We have 58% of the single users and 42% of married users. Combining with previous observation, single users contributes more as 58% of the single contributes to the 60% of the purchase count.

[88]:
```
df2=df.groupby(['User_ID'])['City_Category'].unique()
df2.value_counts()/len(df2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.
↪PyObjectHashTable.map_locations()
```

```
TypeError: unhashable type: 'numpy.ndarray'
```

```
Exception ignored in: 'pandas._libs.index.IndexEngine._call_map_locations'
Traceback (most recent call last):
  File "pandas\_libs\hashtable_class_helper.pxi", line 5231, in
pandas._libs.hashtable.PyObjectHashTable.map_locations
TypeError: unhashable type: 'numpy.ndarray'
```

[88]:
```
[C]    0.532847
[B]    0.289764
[A]    0.177389
Name: City_Category, dtype: float64
```

#Observation:

1. 53% of the users belong to city category C whereas 29% to category B and 18% belong to category A. Combining from the previous observation category B purchase count is 42% and Category C purchase count is 31%. We can clearly see category B are more actively purchasing inspite of the fact they are only 28% of the total users. On the other hand, we have 53% of category C users but they only contribute 31% of the total purchase count.

[89]:
```
#Checking the age group distribution in different city categories
pd.
↪crosstab(index=df["City_Category"],columns=df["Age"],margins=True,normalize="index")
```

[89]:
```
Age                0-17     18-25     26-35     36-45     46-50     51-55  \
City_Category
A              0.017222  0.186400  0.499222  0.180185  0.051496  0.041288
B              0.023511  0.187076  0.396171  0.205898  0.088272  0.076743
C              0.041612  0.168705  0.316974  0.209131  0.103333  0.085649
All            0.027455  0.181178  0.399200  0.199999  0.083082  0.069993

Age                 55+
City_Category
A              0.024188
B              0.022330
C              0.074596
All            0.039093
```

[90]:
```
#Checking how genders are contributing towards toatl purchase amount
df2=pd.DataFrame(df.groupby(['Gender'])[['Purchase']].sum())

df2['percent'] = (df2['Purchase'] /
                  df2['Purchase'].sum()) * 100
df2
```

[90]:
```
            Purchase    percent
Gender
F         1186232642  23.278576
M         3909580100  76.721424
```

Observation:

We can see male(72% of the population) contributes to more than 76% of the total purchase amount whereas female(28% of the population) contributes 23% of the total purchase amount.

[91]:
```
#Checking how purchase value are spread among differnt age categories
df2=pd.DataFrame(df.groupby(['Age'])[['Purchase']].sum())

df2['percent'] = (df2['Purchase'] /
                  df2['Purchase'].sum()) * 100
df2
```

[91]:
```
         Purchase    percent
Age
0-17    134913183   2.647530
18-25   913848675  17.933325
26-35  2031770578  39.871374
36-45  1026569884  20.145361
46-50   420843403   8.258612
51-55   367099644   7.203947
55+     200767375   3.939850
```

1. We can see the net purchase amount spread is similar to the purchase count spread among the different age groups.

[92]:
```
df2=pd.DataFrame(df.groupby(['Marital_Status'])['Purchase'].sum())

df2['percent'] = (df2['Purchase'] /
                  df2['Purchase'].sum()) * 100
df2
```

[92]:
```
                  Purchase    percent
Marital_Status
0               3008927447  59.047057
1               2086885295  40.952943
```

Observations:

1. Single users are contributing 59% towards the total purchase amount in comparison to 41% by married users.

[93]:
```
df2=pd.DataFrame(df.groupby(['City_Category'])['Purchase'].sum())

df2['percent'] = (df2['Purchase'] /
```

```
                  df2['Purchase'].sum()) * 100
df2
```

[93]:

City_Category	Purchase	percent
A	1316471661	25.834381
B	2115533605	41.515136
C	1663807476	32.650483

Observations:

1. City_category contribution to the total purchase amount is also similar to their contribution towards Purchase count. Still, combining with previous observation we can City_category C although has percentage purchase count of 31% but they contribute more in terms of purchase amount i.e. 32.65%. We can infer City category C purchase higher value products.

[94]:
```
df2=pd.DataFrame(df.groupby(['Stay_In_Current_City_Years'])[['Purchase']].sum())

df2['percent'] = (df2['Purchase'] /
                  df2['Purchase'].sum()) * 100
df2
```

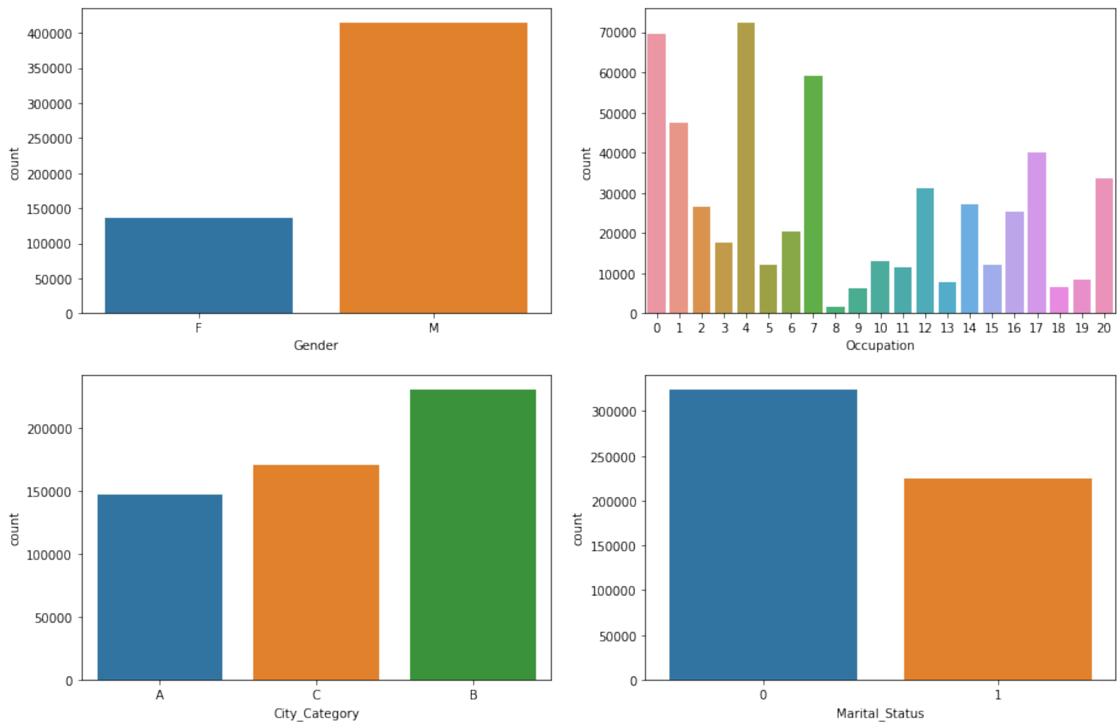
[94]:

Stay_In_Current_City_Years	Purchase	percent
0	682979229	13.402754
1	1792872533	35.183250
2	949173931	18.626547
3	884902659	17.365290
4+	785884390	15.422160

Univariate Analysis:

We can explore the distribution of the data for the quantitative attributes using histplot.

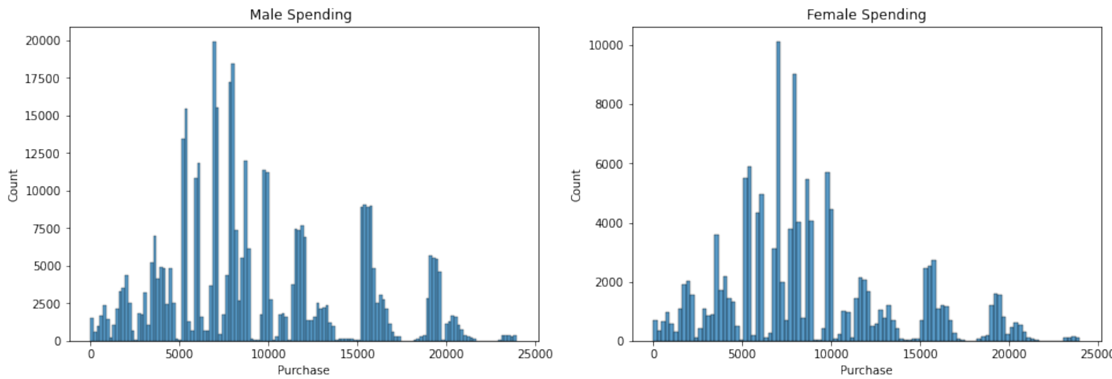
[95]:
```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))
sns.countplot(data=df, x='Gender', ax=axs[0,0])
sns.countplot(data=df, x='Occupation', ax=axs[0,1])
sns.countplot(data=df, x='City_Category', ax=axs[1,0])
sns.countplot(data=df, x='Marital_Status', ax=axs[1,1])
plt.show()
```



#Observations:

1. We can clearly see from the graphs above the purchases done by males are much higher than females.
2. We have 21 occupations categories. Occupation category 4, 0, and 7 are with higher number of purchases and category 8 with the lowest number of purchaes.
3. The purchases are highest from City category B.
4. Single customer purchases are higher than married users.

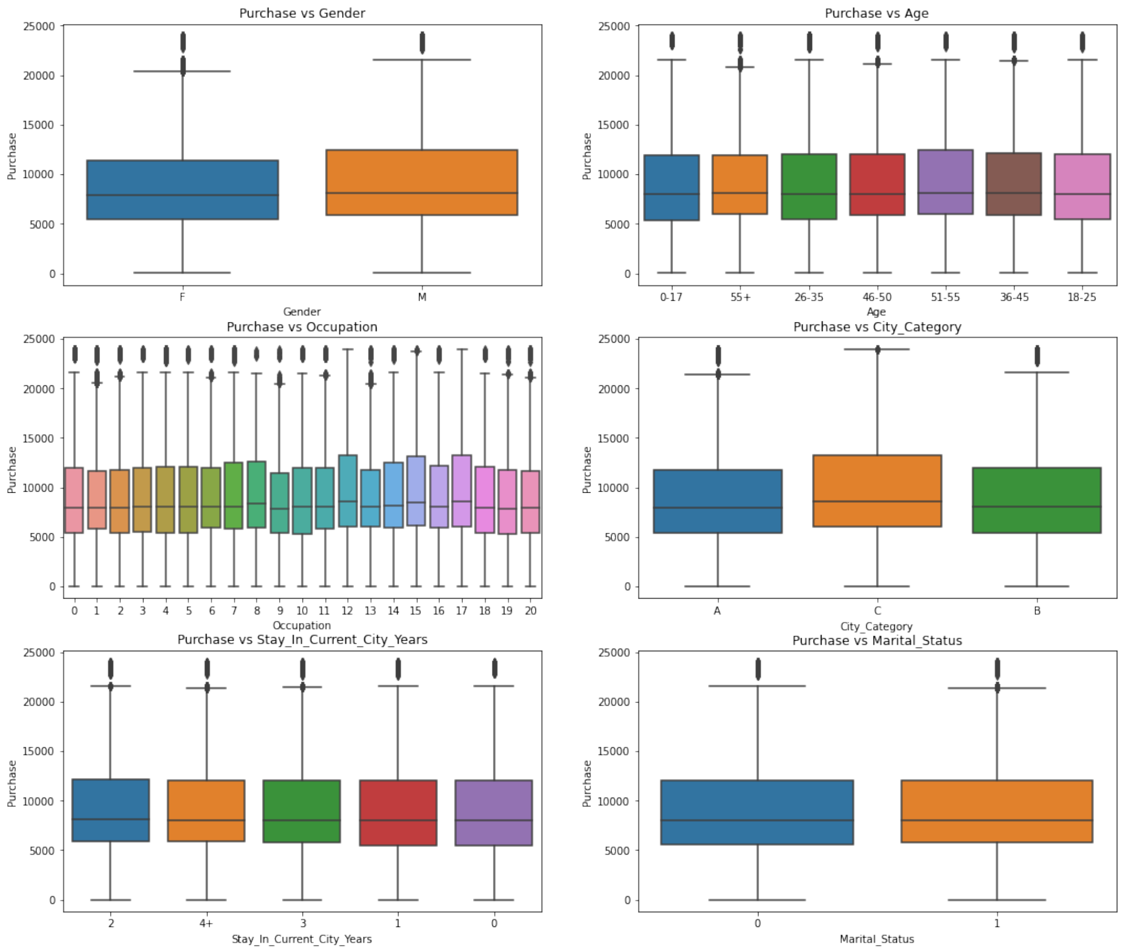
[96]:
```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(16,5))
sns.histplot(data=df[df['Gender']=='M']['Purchase'], ax=axs[0]).set_title("Male Spending ")
sns.histplot(data=df[df['Gender']=='F']['Purchase'], ax=axs[1]).set_title("Female Spending")
plt.show()
```

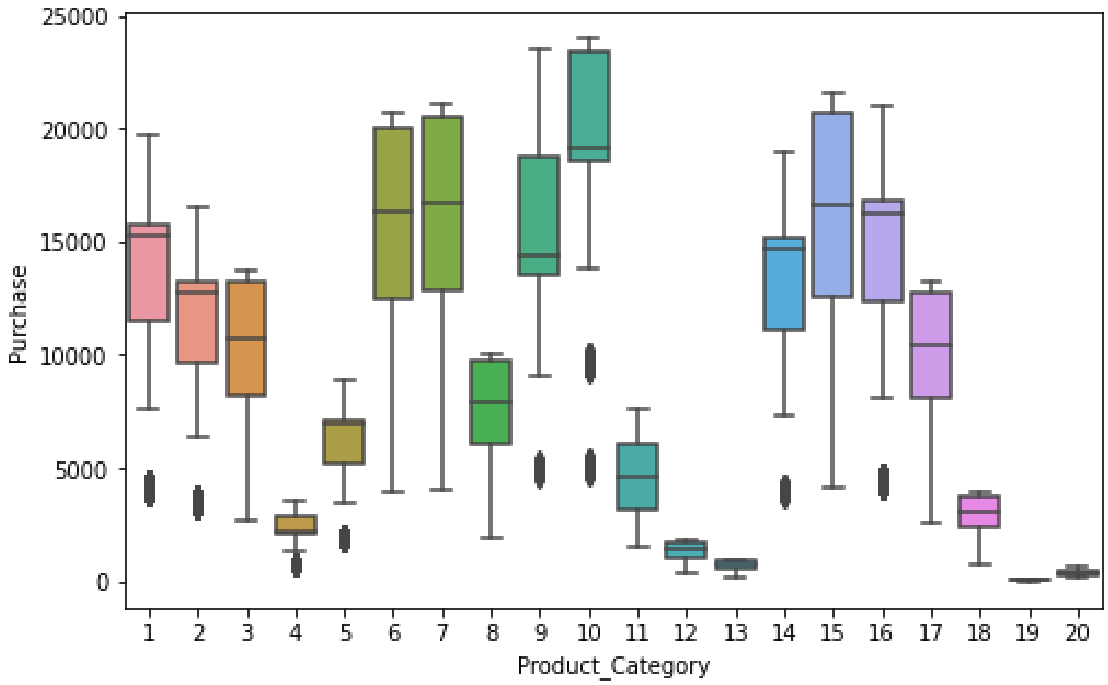


#Observations:

1. From the above histplot, we can clearly see spending behaviour is very much similar in nature for both males and females as the maximum purchase count are between the purchase value range of 5000-10000 for both. But, the purchase count are more in case of males.

[97]:
```
attr = ['Gender', 'Age', 'Occupation', 'City_Category',␣
 ↪'Stay_In_Current_City_Years', 'Marital_Status']

fig, axs = plt.subplots(nrows=3, ncols=2, figsize=(18, 10))
fig.subplots_adjust(top=1.3)
count = 0
for row in range(3):
    for col in range(2):
        sns.boxplot(data=df, y='Purchase', x=attr[count], ax=axs[row, col],)
        axs[row,col].set_title(f"Purchase vs {attr[count]}")
        count += 1
plt.show()

plt.figure(figsize=(8, 5))
sns.boxplot(data=df, y='Purchase', x='Product_Category')
plt.show()
```

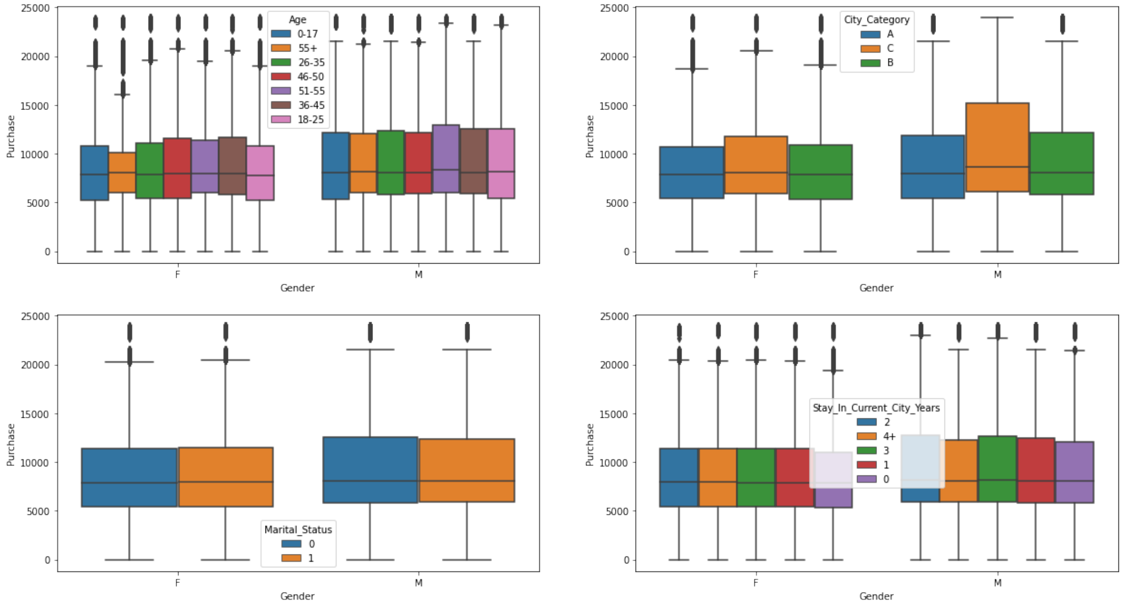

Purchase vs Gender
Purchase
Gender
F
M
Purchase vs Age
Purchase
Age
0-17
55+
26-35
46-50
51-55
36-45
18-25
Purchase vs Occupation
Purchase
Occupation
Purchase vs City_Category
Purchase
City_Category
A
C
B
Purchase vs Stay_In_Current_City_Years
Purchase
Stay_In_Current_City_Years
2
4+
3
1
0
Purchase vs Marital_Status
Purchase
Marital_Status
0
1



Observations:

1. The spending behaviour for males and females are similar as we had seen from the above histplot. Males purchasing value are in the little higher range than females.
2. Among differnt age categories, we see similar purchase behaviour. For all age groups, most of the purchases are of the values between 5k to 12k with all have some outliers.
3. Among different occupation as well, we see similar purchasing behaviour in terms of the purchase values.
4. Similarly for City category, stay in current city years, marital status - we see the users spends mostly in the range of 5k to 12k.
5. We see variations among product categories. Product category 10 products are the costliest ones. Also, there are few outliers for some of the product categories.

[98]:
```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20, 6))
fig.subplots_adjust(top=1.5)
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()
```

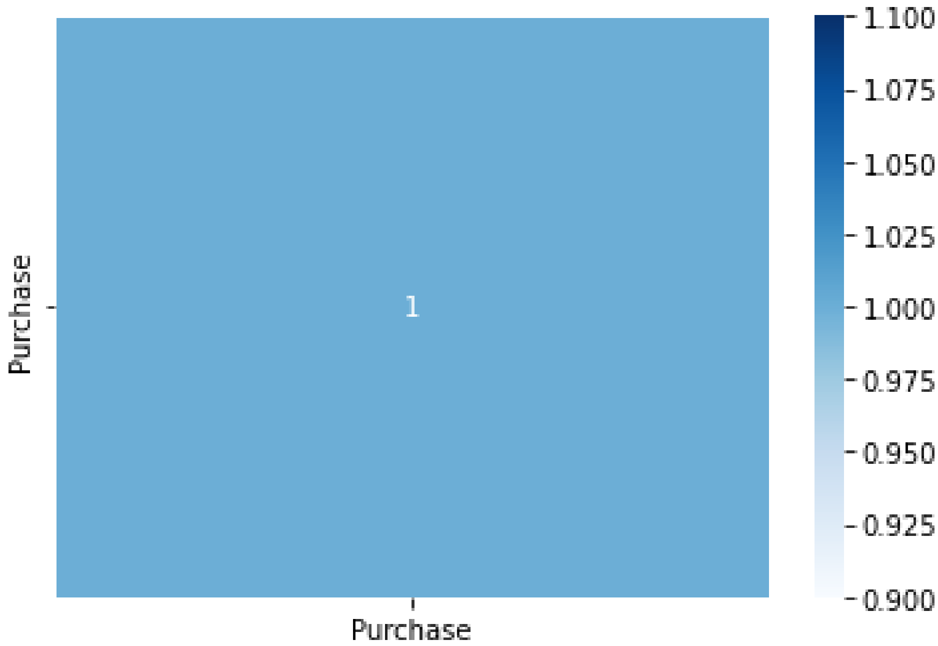


Observations:

1. The purchasing pattern is very much similar for males and females even among diffent age groups.
2. The purchasing behaviour of males and females basis different citi categories is also similar in nature. Still, males from city category B tends to purchase costlier products in comparison to females.
3. Males and females spending behaviour remains similar even when take into account their marital status.
4. Purchase values are similar for males and females basis Stay_in_current_city_years. Although, Males buy slightly high value products.

[99]:
```python
sns.heatmap(df.corr(), annot=True, cmap="Blues", linewidth=.5)
```

[99]: <AxesSubplot:>



#Observations:

1. From the above correlation plot, we can see the correlation is not significant between any pair of variables.

[100]:
```
avgamt_gender = df.groupby(['User_ID', 'Gender'])[['Purchase']].sum()
avgamt_gender = avgamt_gender.reset_index()
avgamt_gender
```

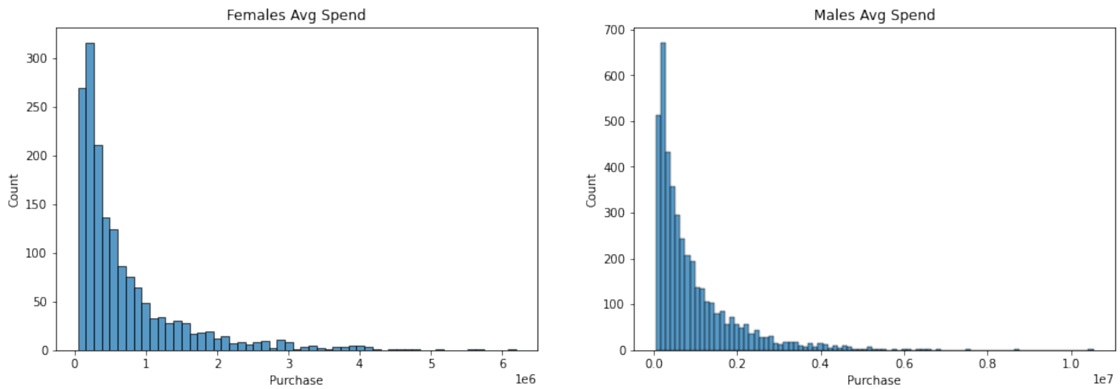
[100]:
```
      User_ID Gender  Purchase
0     1000001      F    334093
1     1000002      M    810472
2     1000003      M    341635
3     1000004      M    206468
4     1000005      M    821001
…         …     …         …
5886  1006036      F   4116058
5887  1006037      F   1119538
5888  1006038      F     90034
5889  1006039      F    590319
5890  1006040      M   1653299

[5891 rows x 3 columns]
```

[101]:
```
# Gender wise count in the entire data
avgamt_gender['Gender'].value_counts()
```

[101]:
```
M    4225
F    1666
Name: Gender, dtype: int64
```

[102]:
```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(16,5))

sns.histplot(data=avgamt_gender[avgamt_gender['Gender']=='F']['Purchase'],␣
↪ax=axs[0]).set_title("Females Avg Spend")
sns.histplot(data=avgamt_gender[avgamt_gender['Gender']=='M']['Purchase'],␣
↪ax=axs[1]).set_title("Males Avg Spend")
```

[102]:
```
Text(0.5, 1.0, 'Males Avg Spend')
```



Observations:

1. Average amount spend by males are higher than females.

[103]:
```
avgamt_gender.groupby(['Gender'])[['Purchase']].mean()
```

[103]:
```
             Purchase
Gender
F       712024.394958
M       925344.402367
```

[104]:
```
avgamt_gender.groupby(['Gender'])['Purchase'].sum()
```

[104]:
```
Gender
F    1186232642
M    3909580100
Name: Purchase, dtype: int64
```

Observations:

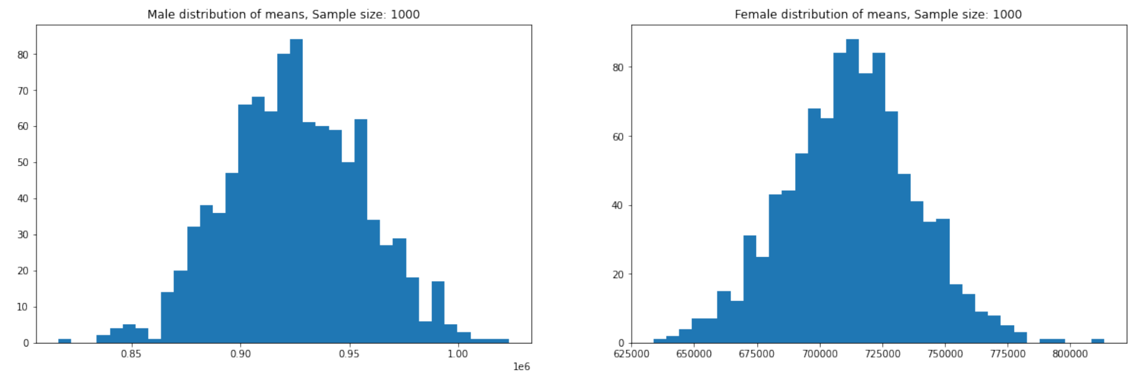
1. Average amount for the males is 925344 for the entire population whereas it's much lesser for females(712024).

2. Total amount spend by males is around 4 billion whereas for females it's 1.2 billion.

[105]:
```python
avgamt_male = avgamt_gender[avgamt_gender['Gender']=='M']
avgamt_female = avgamt_gender[avgamt_gender['Gender']=='F']
```

[106]:
```python
#Finding the sample(sample size=1000) for avg purchase amount for males and females
genders = ["M", "F"]

sample_size = 1000

num_repitions = 1000
male_means = []
female_means = []

for i in range(num_repitions):
    male_mean = avgamt_male.sample(sample_size, replace=True)['Purchase'].mean()
    female_mean = avgamt_female.sample(sample_size, replace=True)['Purchase'].mean()

    male_means.append(male_mean)
    female_means.append(female_mean)
```

[107]:
```python
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: 1000")
axis[1].set_title("Female distribution of means, Sample size: 1000")

plt.show()
```



Observations:

1. The means sample seems to be normally distributed for both males and females. Also, we

can see the mean of the sample means are closer to the population mean as per central limit theorem.

[108]:
```
#Taking the values for z at 90%, 95% and 99% confidence interval as:
z90=1.645 #90% Confidence Interval
z95=1.960 #95% Confidence Interval
z99=2.576 #99% Confidence Interval

print("Population avg spend amount for Male: {:.2f}".
 ↪format(avgamt_male['Purchase'].mean()))
print("Population avg spend amount for Female: {:.2f}\n".
 ↪format(avgamt_female['Purchase'].mean()))

print("Sample avg spend amount for Male: {:.2f}".format(np.mean(male_means)))
print("Sample avg spend amount for Female: {:.2f}\n".format(np.
 ↪mean(female_means)))

print("Sample std for Male: {:.2f}".format(pd.Series(male_means).std()))
print("Sample std for Female: {:.2f}\n".format(pd.Series(female_means).std()))

print("Sample std error for Male: {:.2f}".format(pd.Series(male_means).std()/np.
 ↪sqrt(1000)))
print("Sample std error for Female: {:.2f}\n".format(pd.Series(female_means).
 ↪std()/np.sqrt(1000)))

sample_mean_male=np.mean(male_means)
sample_mean_female=np.mean(female_means)

sample_std_male=pd.Series(male_means).std()
sample_std_female=pd.Series(female_means).std()

sample_std_error_male=sample_std_male/np.sqrt(1000)
sample_std_error_female=sample_std_female/np.sqrt(1000)

Upper_Limit_male=z90*sample_std_error_male + sample_mean_male
Lower_Limit_male=sample_mean_male - z90*sample_std_error_male

Upper_Limit_female=z90*sample_std_error_female + sample_mean_female
Lower_Limit_female=sample_mean_female - z90*sample_std_error_female

print("Male_CI: ",[Lower_Limit_male,Upper_Limit_male])
print("Female_CI: ",[Lower_Limit_female,Upper_Limit_female])
```

```
Population avg spend amount for Male: 925344.40
Population avg spend amount for Female: 712024.39

Sample avg spend amount for Male: 924911.11
Sample avg spend amount for Female: 712504.28
```

```
Sample std for Male: 31274.49
Sample std for Female: 25831.10

Sample std error for Male: 988.99
Sample std error for Female: 816.85

Male_CI:  [923284.2284755156, 926537.9933984844]
Female_CI:  [711160.5588799753, 713847.9985180247]
```

#Observation:

Now using the Confidence interval at 90%, we can say that:

Average amount spend by male customers lie in the range 9,22,940.71 - 9,26,225.18

Average amount spend by female customers lie in range 7,10,425.64 - 7,13,064.55

Observation:

Now using the Confidence interval at 95%, we can say that:

Average amount spend by male customers lie in the range 9,22,626.24 - 9,26,539.65

Average amount spend by female customers lie in range 7,10,172.98 - 7,13,317.21

[112]:
```
#Taking the values for z at 90%, 95% and 99% confidence interval as:
z90=1.645 #90% Confidence Interval
z95=1.960 #95% Confidence Interval
z99=2.576 #99% Confidence Interval

print("Population avg spend amount for Male: {:.2f}".
 ↪format(avgamt_male['Purchase'].mean()))
print("Population avg spend amount for Female: {:.2f}\n".
 ↪format(avgamt_female['Purchase'].mean()))

print("Sample avg spend amount for Male: {:.2f}".format(np.mean(male_means)))
print("Sample avg spend amount for Female: {:.2f}\n".format(np.
 ↪mean(female_means)))

print("Sample std for Male: {:.2f}".format(pd.Series(male_means).std()))
print("Sample std for Female: {:.2f}\n".format(pd.Series(female_means).std()))

print("Sample std error for Male: {:.2f}".format(pd.Series(male_means).std()/np.
 ↪sqrt(1000)))
print("Sample std error for Female: {:.2f}\n".format(pd.Series(female_means).
 ↪std()/np.sqrt(1000)))

sample_mean_male=np.mean(male_means)
sample_mean_female=np.mean(female_means)
```

```
sample_std_male=pd.Series(male_means).std()
sample_std_female=pd.Series(female_means).std()

sample_std_error_male=sample_std_male/np.sqrt(1000)
sample_std_error_female=sample_std_female/np.sqrt(1000)

Upper_Limit_male=z95*sample_std_error_male + sample_mean_male
Lower_Limit_male=sample_mean_male - z95*sample_std_error_male

Upper_Limit_female=z95*sample_std_error_female + sample_mean_female
Lower_Limit_female=sample_mean_female - z95*sample_std_error_female

print("Male_CI: ",[Lower_Limit_male,Upper_Limit_male])
print("Female_CI: ",[Lower_Limit_female,Upper_Limit_female])
```

```
Population avg spend amount for Male: 925344.40
Population avg spend amount for Female: 712024.39

Sample avg spend amount for Male: 924911.11
Sample avg spend amount for Female: 712504.28

Sample std for Male: 31274.49
Sample std for Female: 25831.10

Sample std error for Male: 988.99
Sample std error for Female: 816.85

Male_CI:  [922972.6977914016, 926849.5240825984]
Female_CI:  [710903.2508295238, 714105.3065684763]
```

Observation:

Now using the Confidence interval at 95%, we can say that:

Average amount spend by male customers lie in the range 9,22,626.24 - 9,26,539.65

Average amount spend by female customers lie in range 7,10,172.98 - 7,13,317.21

Calculating 99% confidence interval for sample size 1000:

[113]:
```
#Taking the values for z at 90%, 95% and 99% confidence interval as:
z90=1.645 #90% Confidence Interval
z95=1.960 #95% Confidence Interval
z99=2.576 #99% Confidence Interval

print("Population avg spend amount for Male: {:.2f}".
 ↪format(avgamt_male['Purchase'].mean()))
print("Population avg spend amount for Female: {:.2f}\n".
 ↪format(avgamt_female['Purchase'].mean()))
```

```
print("Sample avg spend amount for Male: {:.2f}".format(np.mean(male_means)))
print("Sample avg spend amount for Female: {:.2f}\n".format(np.
 ↪mean(female_means)))

print("Sample std for Male: {:.2f}".format(pd.Series(male_means).std()))
print("Sample std for Female: {:.2f}\n".format(pd.Series(female_means).std()))

print("Sample std error for Male: {:.2f}".format(pd.Series(male_means).std()/np.
 ↪sqrt(1000)))
print("Sample std error for Female: {:.2f}\n".format(pd.Series(female_means).
 ↪std()/np.sqrt(1000)))

sample_mean_male=np.mean(male_means)
sample_mean_female=np.mean(female_means)

sample_std_male=pd.Series(male_means).std()
sample_std_female=pd.Series(female_means).std()

sample_std_error_male=sample_std_male/np.sqrt(1000)
sample_std_error_female=sample_std_female/np.sqrt(1000)

Upper_Limit_male=z99*sample_std_error_male + sample_mean_male
Lower_Limit_male=sample_mean_male - z99*sample_std_error_male

Upper_Limit_female=z99*sample_std_error_female + sample_mean_female
Lower_Limit_female=sample_mean_female - z99*sample_std_error_female

print("Male_CI: ",[Lower_Limit_male,Upper_Limit_male])
print("Female_CI: ",[Lower_Limit_female,Upper_Limit_female])
```

```
Population avg spend amount for Male: 925344.40
Population avg spend amount for Female: 712024.39

Sample avg spend amount for Male: 924911.11
Sample avg spend amount for Female: 712504.28

Sample std for Male: 31274.49
Sample std for Female: 25831.10

Sample std error for Male: 988.99
Sample std error for Female: 816.85

Male_CI:  [922363.4822313563, 927458.7396426437]
Female_CI:  [710400.0706419741, 714608.4867560259]
```

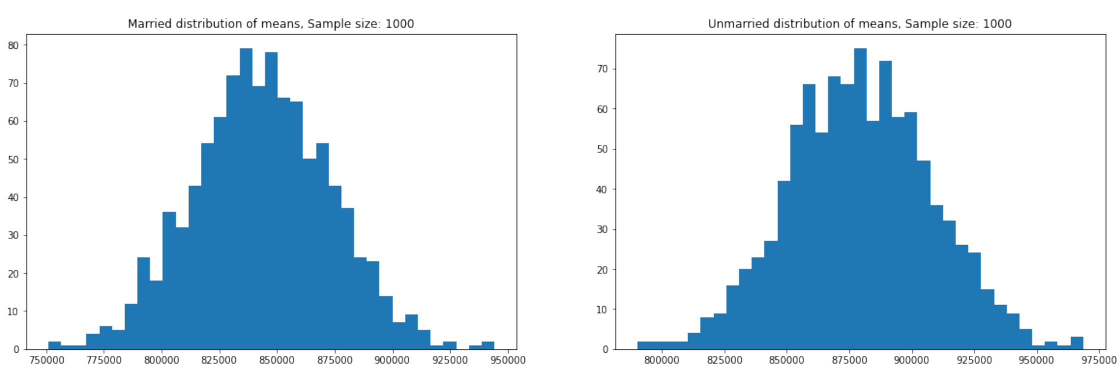
Observation:

Now using the Confidence interval at 99%, we can say that:

Average amount spend by male customers lie in the range 9,22,011.28 - 9,27,154.61

Average amount spend by female customers lie in range 7,09,678.88 - 7,13,811.31

[114]:
```
avg_Marital = df.groupby(['User_ID', 'Marital_Status'])[['Purchase']].sum()
avg_Marital = avg_Marital.reset_index()

avgamt_married = avg_Marital[avg_Marital['Marital_Status']==1]
avgamt_single = avg_Marital[avg_Marital['Marital_Status']==0]

sample_size = 1000
num_repitions = 1000
married_means = []
single_means = []

for i in range(num_repitions):
    avg_married = avg_Marital[avg_Marital['Marital_Status']==1].
 ↪sample(sample_size, replace=True)['Purchase'].mean()
    avg_single = avg_Marital[avg_Marital['Marital_Status']==0].
 ↪sample(sample_size, replace=True)['Purchase'].mean()

    married_means.append(avg_married)
    single_means.append(avg_single)


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

axis[0].hist(married_means, bins=35)
axis[1].hist(single_means, bins=35)
axis[0].set_title("Married distribution of means, Sample size: 1000")
axis[1].set_title("Unmarried distribution of means, Sample size: 1000")

plt.show()
```



Observations:

The means sample seems to be normally distributed for both married and singles. Also, we can see the mean of the sample means are closer to the population mean as per central limit theorem.

[115]:
```
avg_Marital['Marital_Status'].value_counts()
```

[115]:
```
0    3417
1    2474
Name: Marital_Status, dtype: int64
```

[116]:
```
#Taking the values for z at 90%, 95% and 99% confidence interval as:
z90=1.645 #90% Confidence Interval
z95=1.960 #95% Confidence Interval
z99=2.576 #99% Confidence Interval

print("Population avg spend amount for Married: {:.2f}".
 ↪format(avgamt_married['Purchase'].mean()))
print("Population avg spend amount for Single: {:.2f}\n".
 ↪format(avgamt_single['Purchase'].mean()))

print("Sample avg spend amount for Married: {:.2f}".format(np.
 ↪mean(married_means)))
print("Sample avg spend amount for Single: {:.2f}\n".format(np.
 ↪mean(single_means)))

print("Sample std for Married: {:.2f}".format(pd.Series(married_means).std()))
print("Sample std for Single: {:.2f}\n".format(pd.Series(single_means).std()))

print("Sample std error for Married: {:.2f}".format(pd.Series(married_means).
 ↪std()/np.sqrt(1000)))
print("Sample std error for Single: {:.2f}\n".format(pd.Series(single_means).
 ↪std()/np.sqrt(1000)))

sample_mean_married=np.mean(married_means)
sample_mean_single=np.mean(single_means)

sample_std_married=pd.Series(married_means).std()
sample_std_single=pd.Series(single_means).std()

sample_std_error_married=sample_std_married/np.sqrt(1000)
sample_std_error_single=sample_std_single/np.sqrt(1000)

Upper_Limit_married=z90*sample_std_error_male + sample_mean_married
Lower_Limit_married=sample_mean_married - z90*sample_std_error_married

Upper_Limit_single=z90*sample_std_error_single + sample_mean_single
Lower_Limit_single=sample_mean_single - z90*sample_std_error_single
```

```
print("Married_CI: ",[Lower_Limit_married,Upper_Limit_married])
print("Single_CI: ",[Lower_Limit_single,Upper_Limit_single])
```

```
Population avg spend amount for Married: 843526.80
Population avg spend amount for Single: 880575.78

Sample avg spend amount for Married: 843587.47
Sample avg spend amount for Single: 879483.28

Sample std for Married: 29635.85
Sample std for Single: 28723.71

Sample std error for Married: 937.17
Sample std error for Single: 908.32

Married_CI:  [842045.8313458387, 845214.3550084843]
Single_CI:  [877989.0910974769, 880977.475542523]
```

[117]:

```
#Taking the values for z at 90%, 95% and 99% confidence interval as:
z90=1.645 #90% Confidence Interval
z95=1.960 #95% Confidence Interval
z99=2.576 #99% Confidence Interval

print("Population avg spend amount for Married: {:.2f}".
 ↪format(avgamt_married['Purchase'].mean()))
print("Population avg spend amount for Single: {:.2f}\n".
 ↪format(avgamt_single['Purchase'].mean()))

print("Sample avg spend amount for Married: {:.2f}".format(np.
 ↪mean(married_means)))
print("Sample avg spend amount for Single: {:.2f}\n".format(np.
 ↪mean(single_means)))

print("Sample std for Married: {:.2f}".format(pd.Series(married_means).std()))
print("Sample std for Single: {:.2f}\n".format(pd.Series(single_means).std()))

print("Sample std error for Married: {:.2f}".format(pd.Series(married_means).
 ↪std()/np.sqrt(1000)))
print("Sample std error for Single: {:.2f}\n".format(pd.Series(single_means).
 ↪std()/np.sqrt(1000)))

sample_mean_married=np.mean(married_means)
sample_mean_single=np.mean(single_means)

sample_std_married=pd.Series(married_means).std()
sample_std_single=pd.Series(single_means).std()
```

```
sample_std_error_married=sample_std_married/np.sqrt(1000)
sample_std_error_single=sample_std_single/np.sqrt(1000)

Upper_Limit_married=z95*sample_std_error_male + sample_mean_married
Lower_Limit_married=sample_mean_married - z95*sample_std_error_married

Upper_Limit_single=z95*sample_std_error_single + sample_mean_single
Lower_Limit_single=sample_mean_single - z95*sample_std_error_single

print("Married_CI: ",[Lower_Limit_married,Upper_Limit_married])
print("Single_CI: ",[Lower_Limit_single,Upper_Limit_single])
```

```
Population avg spend amount for Married: 843526.80
Population avg spend amount for Single: 880575.78

Sample avg spend amount for Married: 843587.47
Sample avg spend amount for Single: 879483.28

Sample std for Married: 29635.85
Sample std for Single: 28723.71

Sample std error for Married: 937.17
Sample std error for Single: 908.32

Married_CI:  [841750.6234562546, 845525.8856925983]
Single_CI:  [877702.9691825256, 881263.5974574742]
```

[118]:
```
#Taking the values for z at 90%, 95% and 99% confidence interval as:
z90=1.645 #90% Confidence Interval
z95=1.960 #95% Confidence Interval
z99=2.576 #99% Confidence Interval

print("Population avg spend amount for Married: {:.2f}".
 ↪format(avgamt_married['Purchase'].mean()))
print("Population avg spend amount for Single: {:.2f}\n".
 ↪format(avgamt_single['Purchase'].mean()))

print("Sample avg spend amount for Married: {:.2f}".format(np.
 ↪mean(married_means)))
print("Sample avg spend amount for Single: {:.2f}\n".format(np.
 ↪mean(single_means)))

print("Sample std for Married: {:.2f}".format(pd.Series(married_means).std()))
print("Sample std for Single: {:.2f}\n".format(pd.Series(single_means).std()))

print("Sample std error for Married: {:.2f}".format(pd.Series(married_means).
 ↪std()/np.sqrt(1000)))
```

```
print("Sample std error for Single: {:.2f}\n".format(pd.Series(single_means).
 ↪std()/np.sqrt(1000)))

sample_mean_married=np.mean(married_means)
sample_mean_single=np.mean(single_means)

sample_std_married=pd.Series(married_means).std()
sample_std_single=pd.Series(single_means).std()

sample_std_error_married=sample_std_married/np.sqrt(1000)
sample_std_error_single=sample_std_single/np.sqrt(1000)

Upper_Limit_married=z99*sample_std_error_male + sample_mean_married
Lower_Limit_married=sample_mean_married - z99*sample_std_error_married

Upper_Limit_single=z99*sample_std_error_single + sample_mean_single
Lower_Limit_single=sample_mean_single - z99*sample_std_error_single

print("Married_CI: ",[Lower_Limit_married,Upper_Limit_married])
print("Single_CI: ",[Lower_Limit_single,Upper_Limit_single])
```

```
Population avg spend amount for Married: 843526.80
Population avg spend amount for Single: 880575.78

Sample avg spend amount for Married: 843587.47
Sample avg spend amount for Single: 879483.28

Sample std for Married: 29635.85
Sample std for Single: 28723.71

Sample std error for Married: 937.17
Sample std error for Single: 908.32

Married_CI:  [841173.3280277348, 846135.1012526436]
Single_CI:  [877143.4418821766, 881823.1247578233]
```

Observation:

For married and singles, it can be seen with larger sample size the sample mean gets closer to tthe population mean. And at greater confidence interval, the range increases.

Recommendations:

1. Men spent more money than women, company can focus on retaining the male customers and getting more male customers.
2. Product_Category - 1, 5, 8 have highest purchasing frequency. it means these are the products in these categories are in more demand. Company can focus on selling more of these products.
3. Unmarried customers spend more money than married customers, So company should focus

on acquisition of Unmarried customers.

4. Customers in the age 26-35 spend more money than the others, So company should focus on acquisition of customers who are in the age 26-35.
5. We have more customers aged 26-35 in the city category B and A, company can focus more on these customers for these cities to increase the business.
6. Male customers living in City_Category C spend more money than other male customers living in B or C, Selling more products in the City_Category C will help the company increase the revenue.
7. Some of the Product category like 19,20,13 have very less purchase. Company can think of dropping it.
8. The top 10 users who have purchased more company should give more offers and discounts so that they can be retained and can be helpful for companies business.
9. The occupation which are contributing more company can think of offering credit cards or other benefits to those customers by liasing with some financial partners to increase the sales.
10. The top products should be given focus in order to maintain the quality in order to further increase the sales of those products.

Question:

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

Ans: No. CI's of male and female do not overlap and upper limits of female purchase CI are lesser than lower limits of male purchase CI. This proves that men usually spend more than women (NOTE: as per data 77% contibutions are from men and only 23% purchases are from women).

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

At 99% Confidence Interval with sample size 1000

Average amount spend by male customers lie in the range 9,22,011.28 - 9,27,154.61

Average amount spend by female customers lie in range 7,09,678.88 - 7,13,811.31

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

Ans: No. Confidence intervals of average male and female spending are not overlapping. This trend can be changed via introducing female centric marketing strategies by Walmart so that more female customers are attracted to increase female purchases to achieve comparable statistics close to 50%.

4. Results when the same activity is performed for Married vs Unmarried

At 99% Confidence Interval with sample size 1000

Average amount spend by married customers lie in the range: [841059.6309378392, 845078.140167503] Average amount spend by unmarried customers lie in the range: [879093.3492016713, 884078.6782803286]

5. Results when the same activity is performed for Age

At 99% Confidence Interval with sample size 200

For age 26-35 confidence interval of means: (931009.46,1048309.18) For age 36-45 confidence interval of means: (805647.89, 953683.53) For age 18-25 confidence interval of means: (784903.24, 924823.00) For age 46-50 confidence interval of means: (688663.50, 896434.06) For age 51-55 confidence interval of means: (670138.33, 856263.52) For age 55+ confidence interval of means: (457227.15, 622167.34) For age 0-17 confidence interval of means: (498997.92, 738737.71)

```
[ ]:
```