business-problem

July 15, 2024

Business Problem

- As a data analyst at Walmart my primary objective is to analyze customer purchase behavior during Black Friday sales to drive strategic decision-making and improve business performance.
- I am tasked with investigating whether there are clear differences in spending habits between male and female customers.
- My goal is to find out if women spend more than men during Black Friday at Walmart.

Walmart

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

The dataset has the following features:

User_ID: User ID

Product_ID: Product ID

Gender: Sex of User

Age: Age in bins

Occupation: Occupation(Masked)

City_Category: Category of the City (A,B,C)

StayInCurrentCityYears: Number of years stay in current city

Marital_Status: Marital Status

ProductCategory: Product Category (Masked)

Purchase: Purchase Amount

Objectives of the Project

- Perform EDA on the given dataset and find insights.
- Provide Useful Insights and Business recommendations that can help the business to grow.

Importing libraries

```
[ ]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     import warnings
     warnings.filterwarnings('ignore')
     import gdown as gd
     from scipy.stats import norm,stats
     from scipy.stats import chi2,chisquare,chi2_contingency
```

Loading the data

```
[ ]: !gdown 1X0pQprMJ-rup8d-gSQ0bA8HKXaZD4cac
```

```
Downloading…
From: https://drive.google.com/uc?id=1X0pQprMJ-rup8d-gSQ0bA8HKXaZD4cac
To: /content/walmart_data.csv
100% 23.0M/23.0M [00:00<00:00, 175MB/s]
```

```
[ ]: df=pd.read_csv("walmart_data.csv")
```

Basic Obervation

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

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

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

These are the first 5 rows of the dataset.

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

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

```
[ ]: df.ndim
```

```
[ ]: 2
```

In the Walmart dataset, there are 5,50068 rows and 10 columns. It has two dimensions.

[ ]:
```
df.columns
```

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

[ ]:
```
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
```

The data types include object (for text/string data) and int64 (for integer data).

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

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

             Purchase
count  550068.000000
mean     9263.968713
std      5023.065394
min        12.000000
```

```
25%       5823.000000
50%       8047.000000
75%      12054.000000
max      23961.000000
```

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

[ ]:
```
       Product_ID  Gender     Age City_Category Stay_In_Current_City_Years
count      550068  550068  550068        550068                     550068
unique       3631       2       7             3                          5
top     P00265242       M   26-35             B                          1
freq         1880  414259  219587        231173                     193821
```

- There are 3631 unique Product_ID in the dataset. P00265242 is the most sold Product_ID.
- The top people purchasing are in the age range of 26-35.
- Males are top in purchasing.
- 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.

Data Cleaning

[ ]:
```
df.isnull().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
```

There are no missing values in this dataset.

Non-Graphical Analysis

[ ]:
```
df['User_ID'].nunique()
```

[ ]: 5891

[ ]:
```
df['Product_ID'].nunique()
```

[ ]: 3631

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

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

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

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

[ ]:
```
df['Occupation'].unique()
```

[ ]:
```
array([10, 16, 15,  7, 20,  9,  1, 12, 17,  0,  3,  4, 11,  8, 19,  2, 18,
        5, 14, 13,  6])
```

[ ]:
```python
df['City_Category'].value_counts()
```

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

[ ]:
```python
Stay_In_Current_City_Years_counts = df['Stay_In_Current_City_Years'].
 ↪value_counts()
percentage_Stay_In_Current_City_Years_counts =␣
 ↪(Stay_In_Current_City_Years_counts / len(df)) * 100
print(f"Stay_In_Current_City_Years count :␣
 ↪\n{Stay_In_Current_City_Years_counts}\nStay_In_Current_City_Years percentage␣
 ↪: \n{percentage_Stay_In_Current_City_Years_counts}")
```

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

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

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

```
Name: count, dtype: float64
```

[ ]:
```
df['Product_Category'].nunique()
```

[ ]: 20

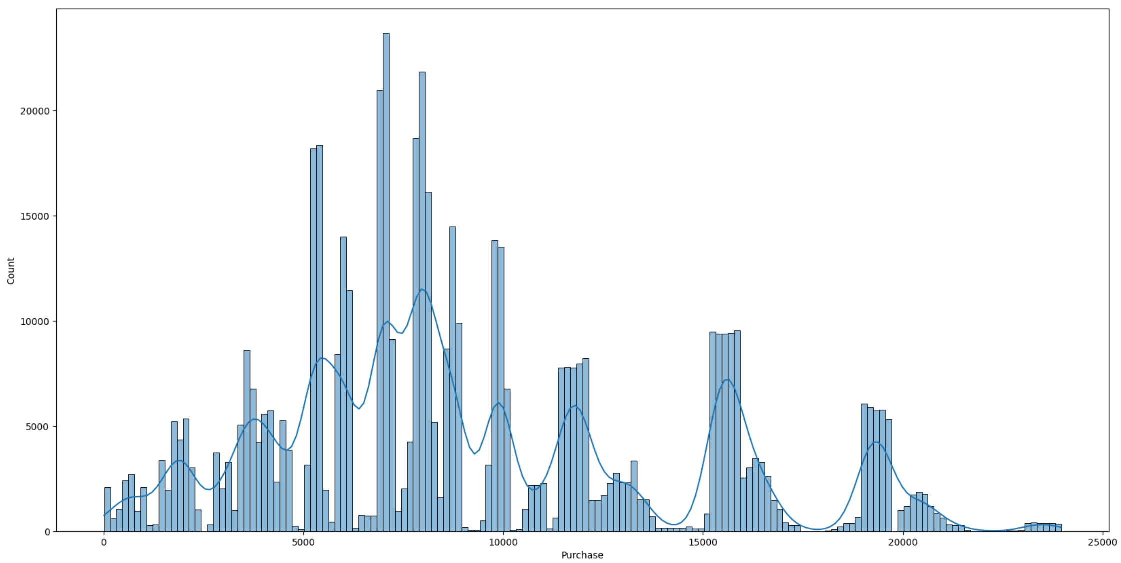
[ ]:
```
df['Purchase'].nunique()
```

[ ]: 18105

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

Visual Analysis - Univariate & Bivariate

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

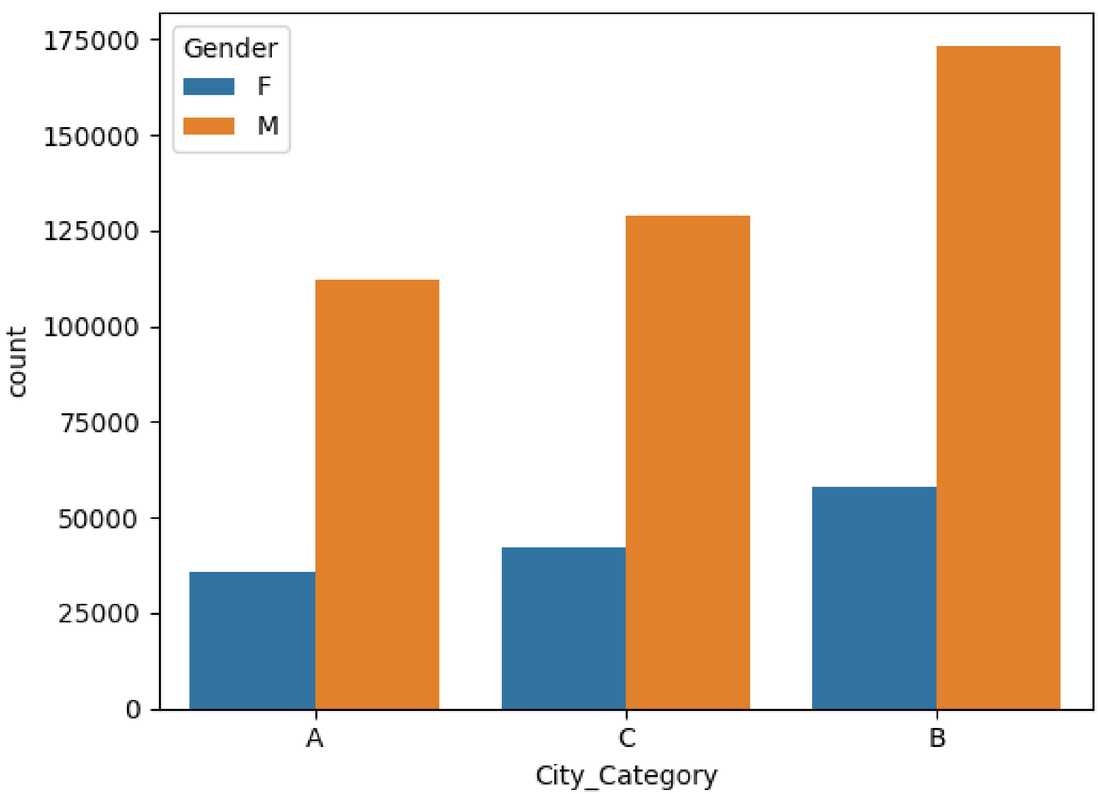


[ ]:
```
contingency_table = pd.crosstab(df['Gender'], df['City_Category'])

# Perform the Chi-Square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Print the results
```

```
print(f"Chi-Square statistic: {chi2}")
print(f"P-value: {p_value}")
print(f"Degrees of freedom: {dof}")
if p_value < 0.05:
    print("Reject H0")
    print("there is a significant association between the 'Gender' and␣
 ↪'City_Category' variables.")
else:
    print("Fail to reject H0")
    print("there is not significant association between the 'Gender' and␣
 ↪'City_Category' variables.")
```

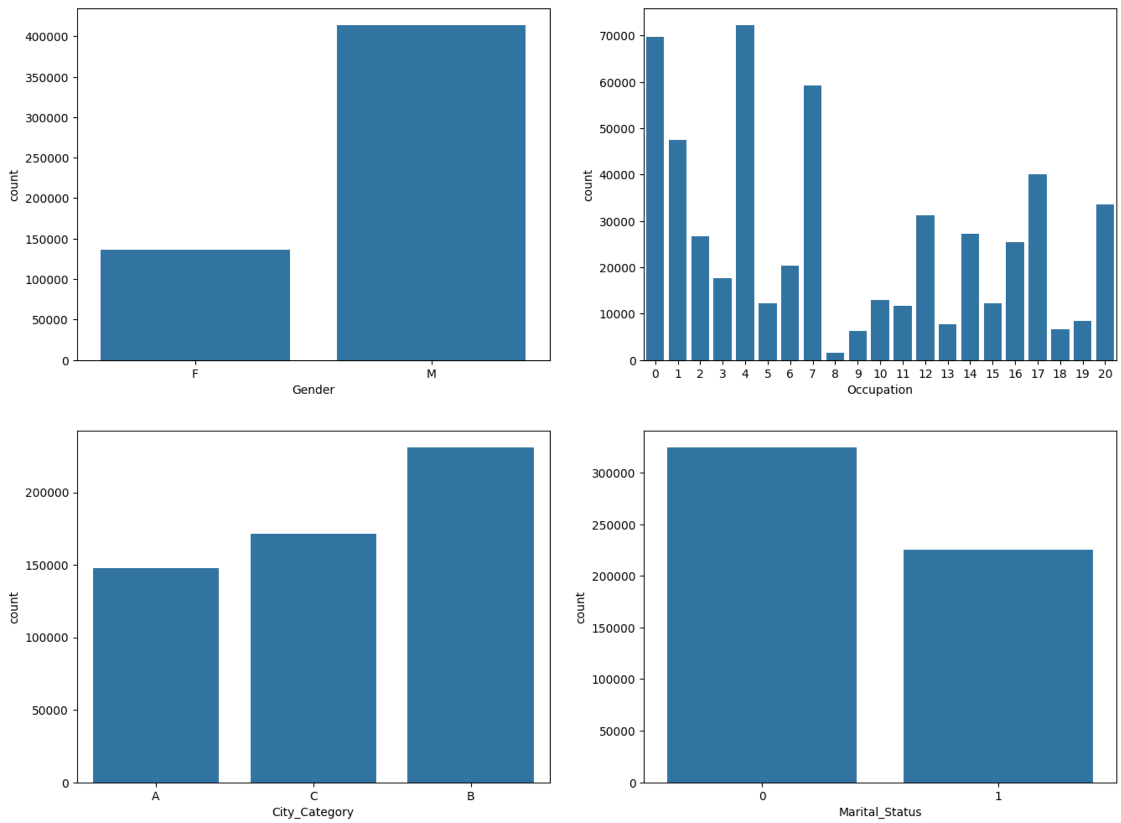
```
Chi-Square statistic: 33.58382571304351
P-value: 5.097590042852447e-08
Degrees of freedom: 2
Reject H0
there is a significant association between the 'Gender' and 'City_Category'
variables.
```

[ ]:
```
sns.countplot(x='City_Category',hue='Gender', data=df)
plt.show()
```



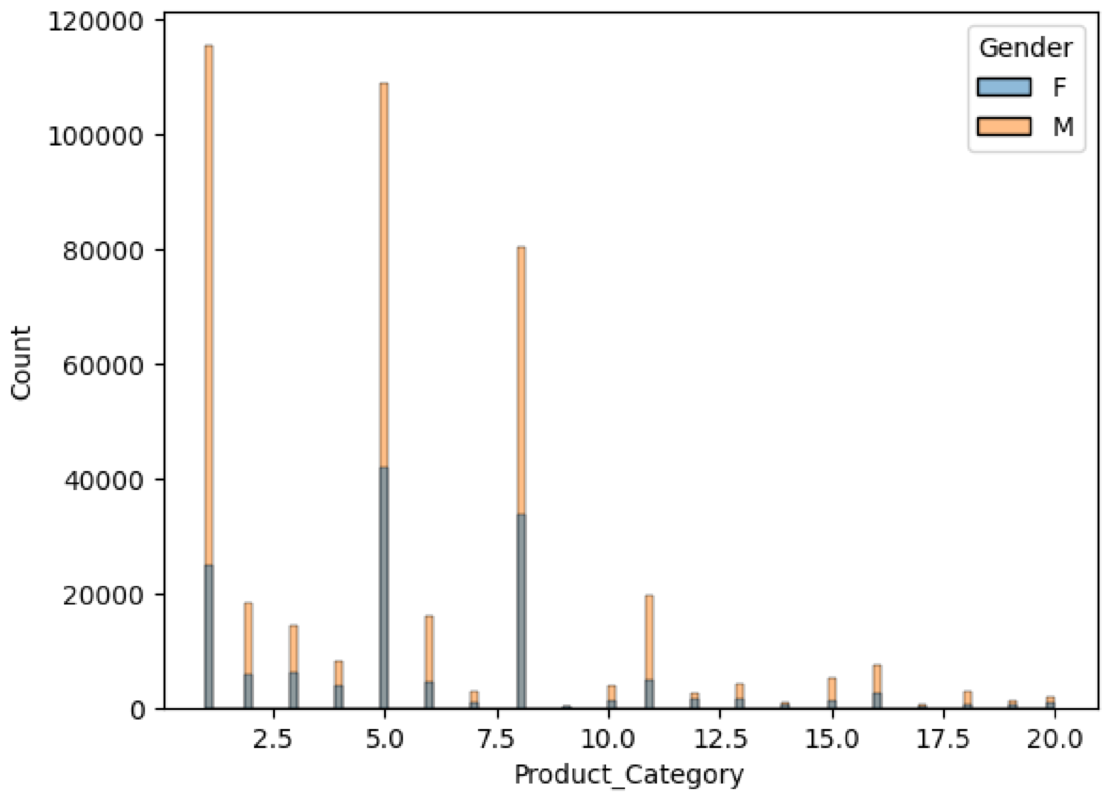
- The Chi-Square statistic obtained is 33.58382571304351, with a p-value of 5.097590042852447e-08. This indicates a significant association between 'Gender' and 'City_Category'
- The degrees of freedom for the Chi-Square test are 2, which is calculated as (number of rows - 1) * (number of columns - 1)
- Since the p-value is less than the chosen significance level of 0.05, we can reject the null hypothesis and conclude that there is a significant association between 'Gender' and 'City_Category'
- The significant association suggests that the distribution of 'City_Category' is not independent of 'Gender'. This means that the two variables are related to each other in some way.

[ ]:
```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(16, 12))
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()
```



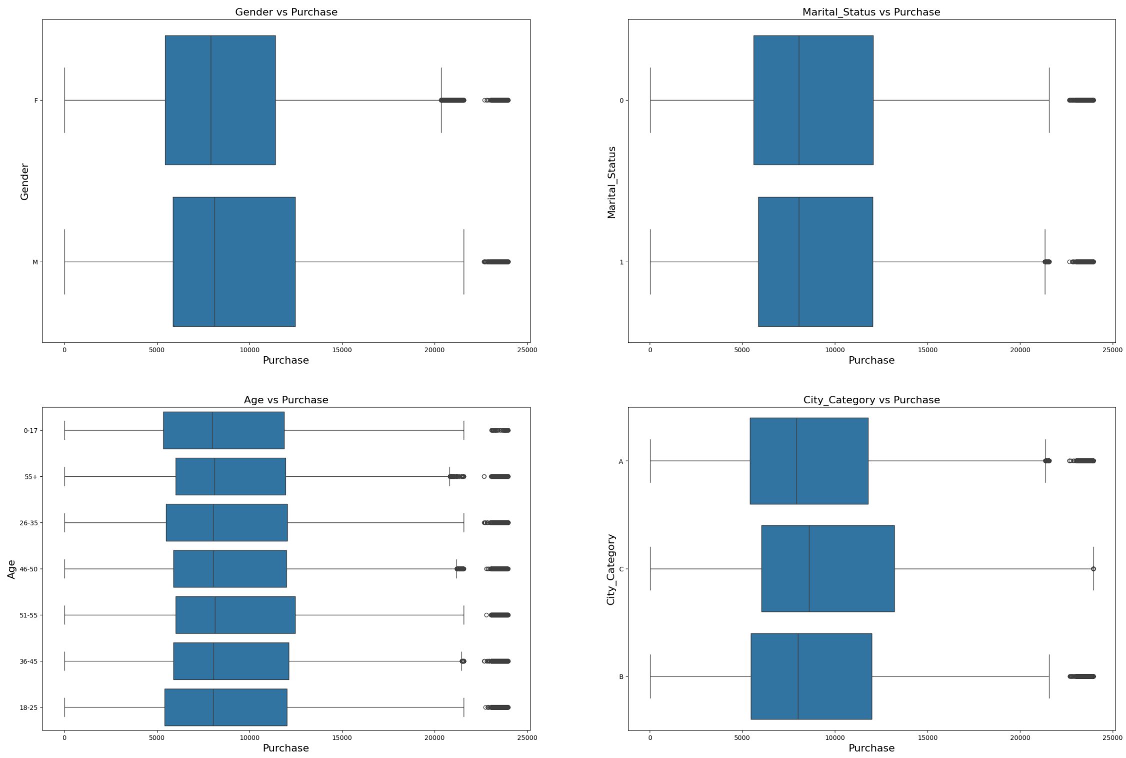
[ ]:
```
#  Preferred Product Categories for Different Genders
sns.histplot(data=df, x='Product_Category', hue='Gender')
```

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

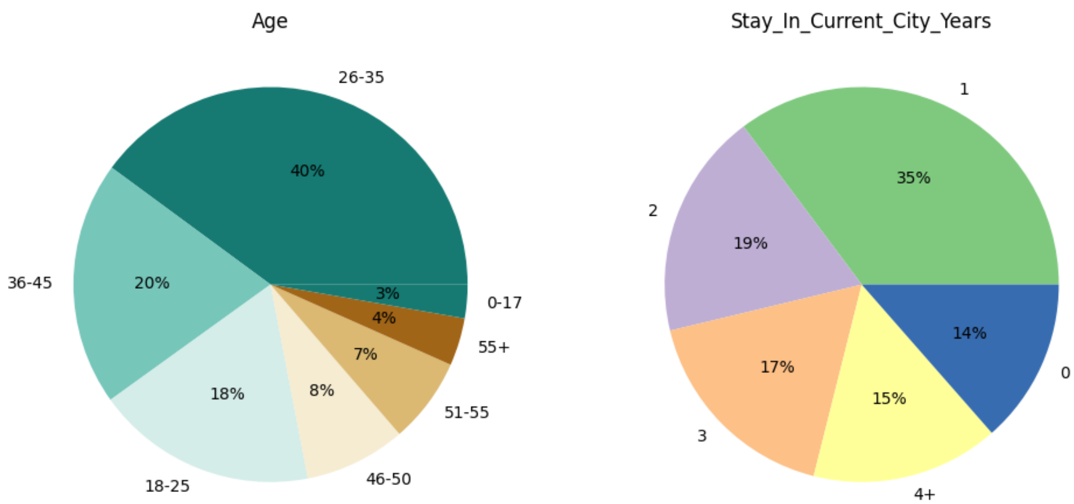


[ ]:
```
fig1, axs=plt.subplots(nrows=2,ncols=2, figsize=(30,20))

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

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

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

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

```
plt.show()
```



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

data = df['Age'].value_counts(normalize=True)*100
palette_color = sns.color_palette('BrBG_r')
axs[0].pie(x=data.values, labels=data.index, autopct='%.
↪0f%%',colors=palette_color)
axs[0].set_title("Age")

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



- Most of the users are Male
- There are 20 different types of Occupation and Product_Category
- More users belong to B City_Category
- More users are Single as compare to Married
- Product_Category - 1, 5, 8, & 11 have highest purchasing frequency.

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

[ ]:
```
      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]
```

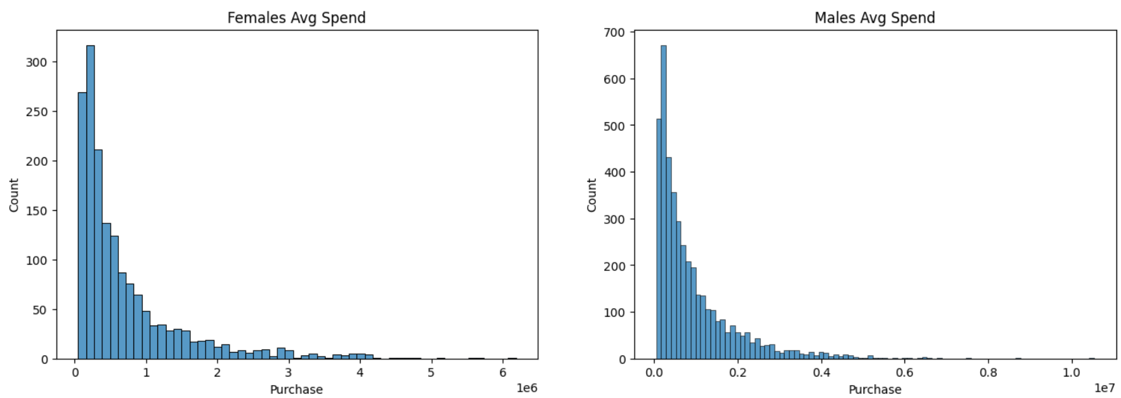
[ ]:
```
avgamt_gender['Gender'].value_counts()
```

[ ]:
```
Gender
M    4225
```

```
F    1666
Name: count, dtype: int64
```

[ ]:
```
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(avgamt_gender[avgamt_gender['Gender']=='M']['Purchase'],␣
 ↪ax=axs[1]).set_title("Males Avg Spend")
```

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

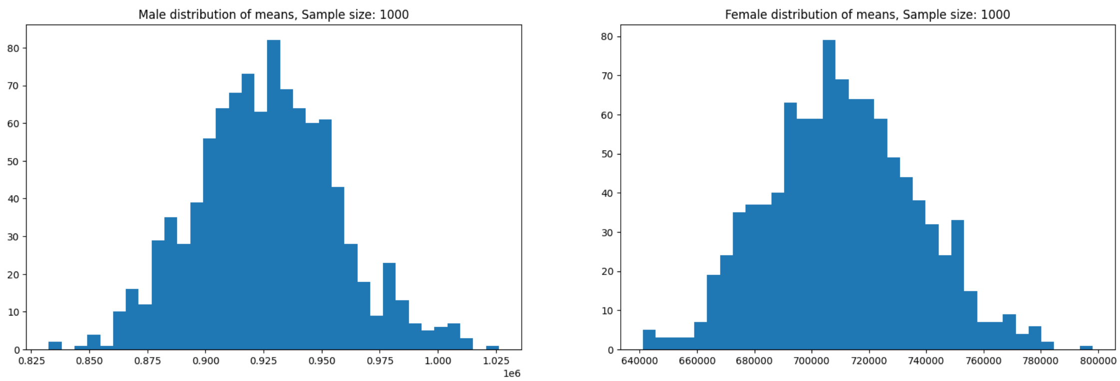


Average amount spend by males are higher than females.

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

[ ]:
```
#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)
```

```
[ ]: 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()
```




```
[ ]: #Taking the value for z at 90% confidence interval as:
z90=1.645 #90% 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: 926398.02
Sample avg spend amount for Female: 710996.08

Sample std for Male: 30413.12
Sample std for Female: 25901.72

Sample std error for Male: 961.75
Sample std error for Female: 819.08

Male_CI:  [924815.9422896195, 927980.0911623808]
Female_CI:  [709648.6874877517, 712343.4751022484]
```

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

[ ]:
```
#Taking the value for z at 95% confidence interval as:
z95=1.960 #95% 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: 926398.02
Sample avg spend amount for Female: 710996.08

Sample std for Male: 30413.12
Sample std for Female: 25901.72

Sample std error for Male: 961.75
Sample std error for Female: 819.08

Male_CI:  [924512.9918656317, 928283.0415863686]
Female_CI:  [709390.6759076404, 712601.4866823597]
```

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

[ ]:
```
#Taking the value for z at  99% confidence interval as:
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: 926398.02
Sample avg spend amount for Female: 710996.08

Sample std for Male: 30413.12
Sample std for Female: 25901.72

Sample std error for Male: 961.75
Sample std error for Female: 819.08
```

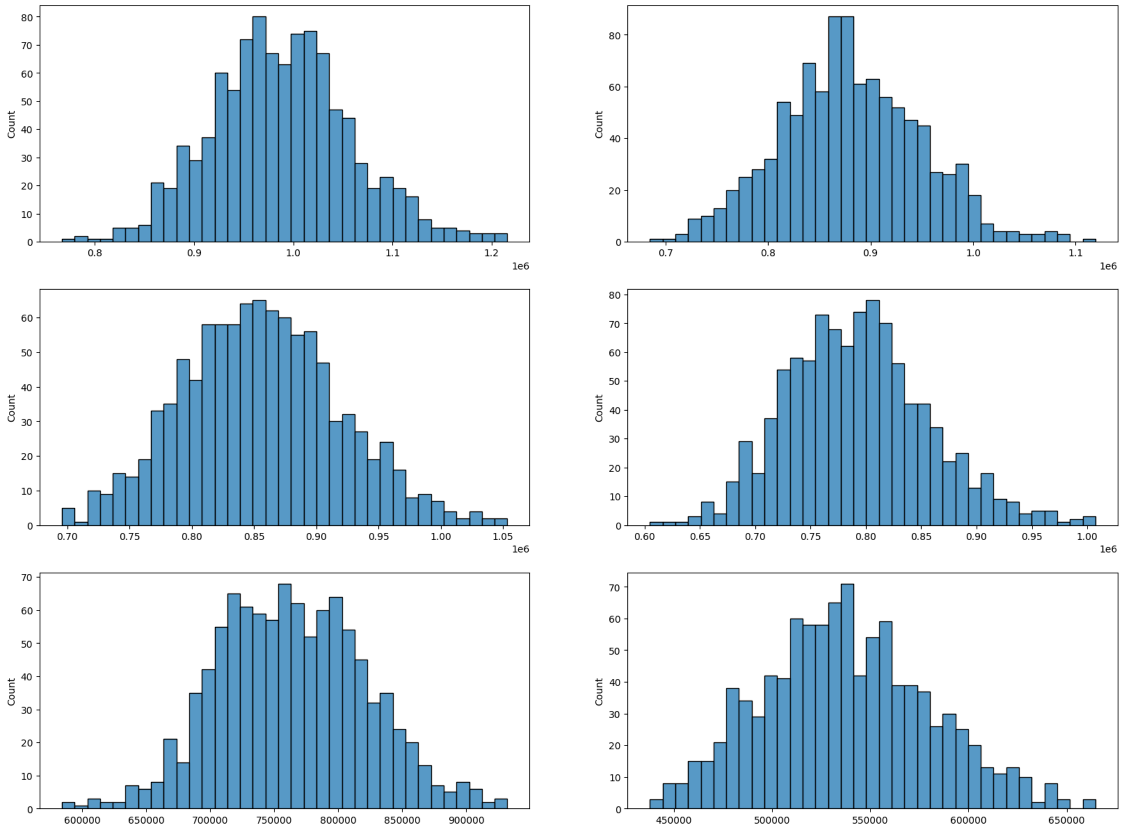
```
Male_CI:  [923920.5554809445, 928875.4779710558]
Female_CI:  [708886.1199287559, 713106.0426612442]
```

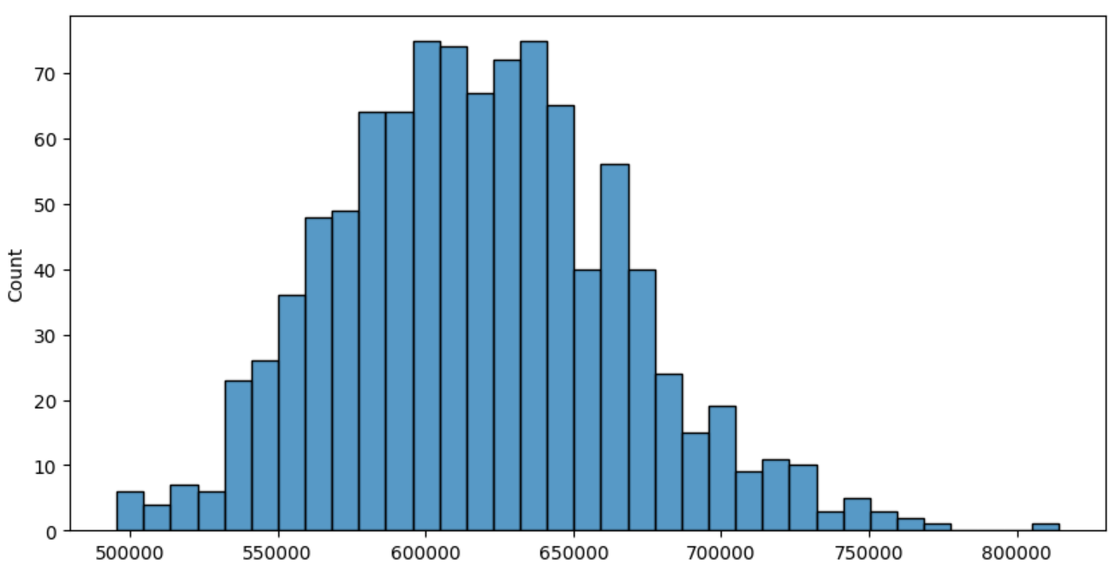
[ ]:
```python
avgamt_age = df.groupby(['User_ID', 'Age'])[['Purchase']].sum()
avgamt_age = avgamt_age.reset_index()

avgamt_age['Age'].value_counts()
```

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

[ ]:
```python
sample_size = 200
num_repitions = 1000

all_sample_means = {}

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

for i in age_intervals:
    for j in range(num_repitions):

        mean = avgamt_age[avgamt_age['Age']==i].sample(sample_size,␣
 ↪replace=True)['Purchase'].mean()
        all_sample_means[i].append(mean)



fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(20, 15))

sns.histplot(all_sample_means['26-35'],bins=35,ax=axis[0,0])
sns.histplot(all_sample_means['36-45'],bins=35,ax=axis[0,1])
sns.histplot(all_sample_means['18-25'],bins=35,ax=axis[1,0])
sns.histplot(all_sample_means['46-50'],bins=35,ax=axis[1,1])
sns.histplot(all_sample_means['51-55'],bins=35,ax=axis[2,0])
sns.histplot(all_sample_means['55+'],bins=35,ax=axis[2,1])

plt.show()
```

```
plt.figure(figsize=(10, 5))
sns.histplot(all_sample_means['0-17'],bins=35)
plt.show()
```





The means sample seems to be normally distributed for all age groups. Also, we can see the mean of the sample means are closer to the population mean as per central limit theorem.

Calculating 90% confidence interval for avg expenses for different age groups for sample size 200:

[ ]:
```
z90=1.645 #90% Confidence Interval

sample_size = 200
num_repitions = 1000

all_population_means={}
all_sample_means = {}

age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']
for i in age_intervals:
    all_sample_means[i] = []
    all_population_means[i]=[]
    population_mean=avgamt_age[avgamt_age['Age']==i]['Purchase'].mean()
    all_population_means[i].append(population_mean)

print("All age group population mean: \n", all_population_means)
print("\n")

for i in age_intervals:
    for j in range(num_repitions):

        mean = avgamt_age[avgamt_age['Age']==i].sample(sample_size,␣
 ↪replace=True)['Purchase'].mean()
        all_sample_means[i].append(mean)


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

    new_df = avgamt_age[avgamt_age['Age']==val]

    std_error = z90*new_df['Purchase'].std()/np.sqrt(len(new_df))
    sample_mean = new_df['Purchase'].mean()
    lower_lim = sample_mean - std_error
    upper_lim = sample_mean + std_error

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

```
All age group population mean:
 {'26-35': [989659.3170969313], '36-45': [879665.7103684661], '18-25':
[854863.119738073], '46-50': [792548.7815442561], '51-55': [763200.9230769231],
```

```
'55+': [539697.2446236559], '0-17': [618867.8119266055]}
```

```
For age 26-35 confidence interval of means: (952206.28, 1027112.35)
For age 36-45 confidence interval of means: (832398.89, 926932.53)
For age 18-25 confidence interval of means: (810187.65, 899538.59)
For age 46-50 confidence interval of means: (726209.00, 858888.57)
For age 51-55 confidence interval of means: (703772.36, 822629.48)
For age 55+ confidence interval of means: (487032.92, 592361.57)
For age 0-17 confidence interval of means: (542320.46, 695415.16)
```

Calculating 95% confidence interval for avg expenses for different age groups for sample size 200:

[ ]:
```
z95=1.960 #95% Confidence Interval

sample_size = 200
num_repitions = 1000

all_means = {}

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

for i in age_intervals:
    for j in range(num_repitions):
        mean = avgamt_age[avgamt_age['Age']==i].sample(sample_size,␣
 ↪replace=True)['Purchase'].mean()
        all_means[i].append(mean)
for val in ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']:

    new_df = avgamt_age[avgamt_age['Age']==val]

    std_error = z95*new_df['Purchase'].std()/np.sqrt(len(new_df))
    sample_mean = new_df['Purchase'].mean()
    lower_lim = sample_mean - std_error
    upper_lim = sample_mean + std_error

    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)
```

Calculating 99% confidence interval for avg expenses for different age groups for sample size 200:

```
[ ]: z99=2.576 #99% Confidence Interval

     sample_size = 200
     num_repitions = 1000

     all_means = {}

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

     for i in age_intervals:
         for j in range(num_repitions):
             mean = avgamt_age[avgamt_age['Age']==i].sample(sample_size,␣
      ↪replace=True)['Purchase'].mean()
             all_means[i].append(mean)
     for val in ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']:

         new_df = avgamt_age[avgamt_age['Age']==val]

         std_error = z99*new_df['Purchase'].std()/np.sqrt(len(new_df))
         sample_mean = new_df['Purchase'].mean()
         lower_lim = sample_mean - std_error
         upper_lim = sample_mean + std_error

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

```
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)
```

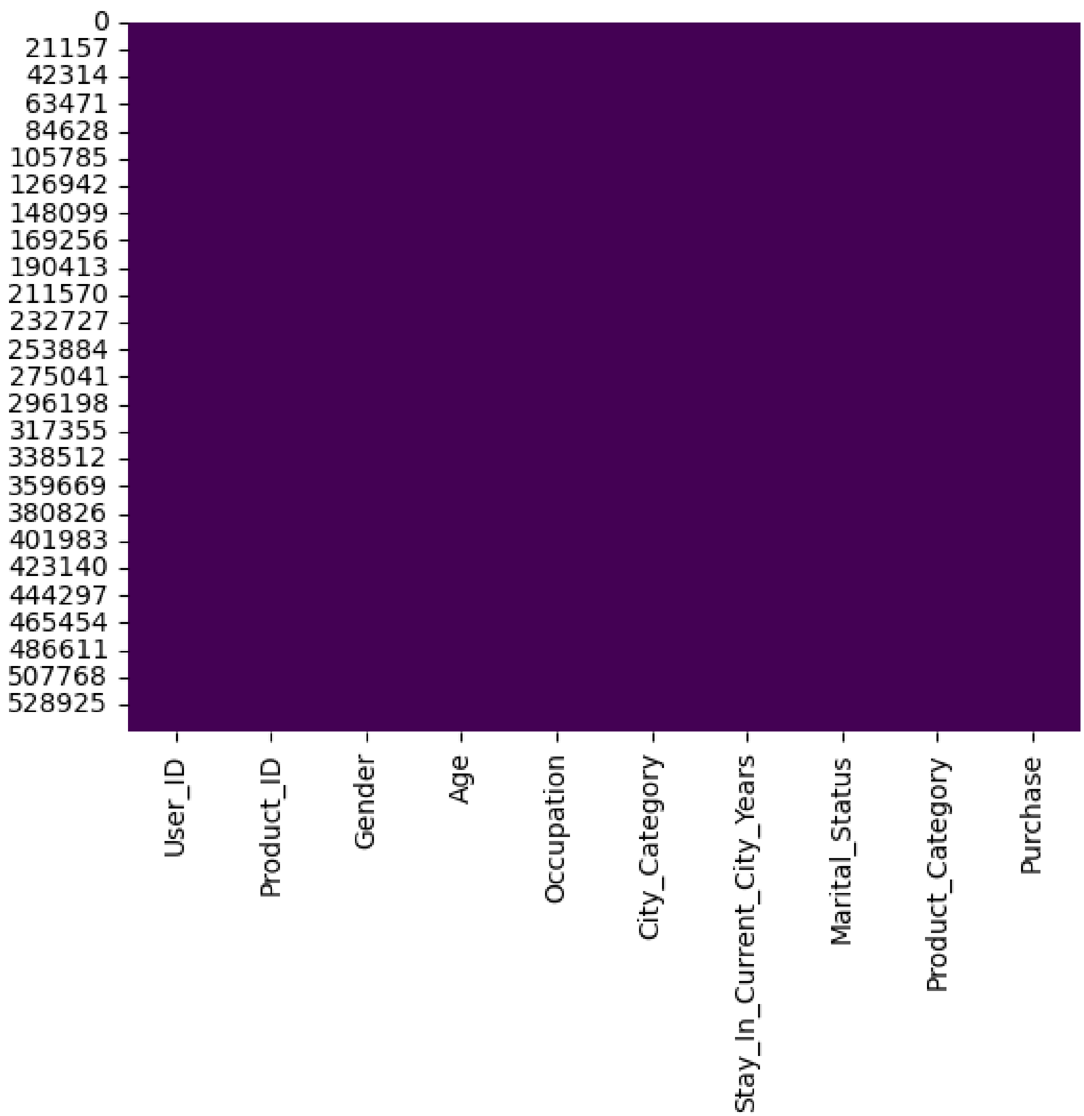
We can see the sample means are closer to the population mean for the differnt age groups. And, with greater confidence interval we have the upper limit and lower limit range increases. As we have seen for gender and marital status, by increasing the sample size we can have the mean of the sample means closer to the population.

Missing Value & Outlier check

```
[ ]: df.isnull().sum().sort_values(ascending=False)
```

[ ]:
```
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
```

[ ]:
```
sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.show()
```



Business Insights

- 80% of the users are between the age 18-50 (40%: 26-35, 18%: 18-25, 20%: 36-45).
- 75% of the users are Male and 25% are Female.
- 60% Single, 40% Married.
- 35% Staying in the city from 1 year, 18% from 2 years, 17% from 3 years.
- Total of 20 product categories in this dataset.
- There are 20 differnent types of occupations in the city.
- Most of the users are Male.
- More users belong to B City_Category.
- More users are Single as compare to Married.
- Product_Category - 1, 5, 8, & 11 have highest purchasing frequency.

Report

With reference to the above data, at a 95% confidence interval: * The highest average amount spent by 26- to 35 year old customers will lie between 945034.42 and 1034284.21. * The average amount spent by 36- to 45-year-old customers will lie between 823347.80 and 935983.62 . * The average amount spent by 18- to 25-year-old customers will lie between 801632.78 and 908093.46 . * The average amount spent by 46- to 50-year-old customers will lie between 713505.63 and 871591.93 . * The average amount spent by 51- to 55-year-old customers will lie between 692392.43 and 834009.42 . * The average amount spent by 55+ age group customers will lie between 476948.26 and 602446.23 * The lowest average amount spent by 0 to 17-year-old customers will lie between 527662.46 and 710073.17 . * From the above data, it is clear that the age group 26 to 35 spends more compared to other age categories. * Age groups above 55 and below 0 to 17 spend very little compared to others. * Confidence intervals for average 26- to 35-year-old and 36- to 45-year-old spending are not overlapping. * With respect to the above data, the company should target the age category between 26 and 35, as they spend more money compared to others.

Recommendations

- Men spent more money than women, company can focus on retaining the male customers and getting more male customers.
- 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.
- Unmarried customers spend more money than married customers, So company should focus on acquisition of Unmarried customers.
- 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.
- 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.
- 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.
- Some of the Product category like 19,20,13 have very less purchase. Company can think of dropping it.
- The top products should be given focus in order to maintain the quality in order to further increase the sales of those products.

- We have highest frequency of purchase order between 5k and 10k, company can focus more on these mid range products to increase the sales.