abhishek-walmart-case-study

August 21, 2023

###Business Case Study : Walmart Confidence Interval and CLT.

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
df = pd.read_csv("original_walmart_data.csv")
dtest= pd.read_csv("original_walmart_data.csv")
df
```

[2]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	7969
…	…	…	…	…	…	…	…	…	…	…
550063	1006033	P00372445	M	51-55	13	B	1	1	20	368
550064	1006035	P00375436	F	26-35	1	C	3	0	20	371
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	137
550066	1006038	P00375436	F	55+	1	C	2	0	20	365
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	490

[550068 rows x 10 columns]

[3]:
```
df.isnull().sum()
```

[3]:
```
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
```

[4]:
```
df.describe(include='all')
```

[4]:

	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
```

[5]:
```
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
```

obeservations

1. Their are no missing value in the given data.
2. Most of the purchase observed by 26-35 age group and their are 7 unique age groups
3. There are 3 unique city categories where city category 'B' is on top.
4. There are 7 unique age groups and most of the purchase belongs to 26-35 age group.
5. their are maximum purchase is 23961 and minimum purchase 12 in purchasing behaviour is quite spread over a aignificant range of values.
6. 75% of the purchase is upto 12054.

[6]:
```
columns=['User_ID','Occupation', 'Marital_Status', 'Product_Category']
df[columns]=df[columns].astype('object')
```

[7]:
```
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
```

[8]:
```
df.describe(include='all')
```

[8]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category
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
75%	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN

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

	Purchase
count	550068.000000
unique	NaN
top	NaN
freq	NaN
mean	9263.968713
std	5023.065394

```
min         12.000000
25%       5823.000000
50%       8047.000000
75%      12054.000000
max      23961.000000
```

[9]:
```
data=df.groupby(['User_ID'])['Age'].unique()
data.value_counts()/len(data)
```

[9]:
```
[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
```

1. we observed almost 35% of users in the age group 26-35 which is highest.

[10]:
```
data=df.groupby(['User_ID'])['Gender'].unique()
data.value_counts()/len(data)
```

[10]:
```
[M]    0.717196
[F]    0.282804
Name: Gender, dtype: float64
```

1. their are 72% male users and 28% female users from the given data.

[11]:
```
data=df.groupby(['User_ID'])['Marital_Status'].unique()
data.value_counts()/len(data)
```

[11]:
```
[0]    0.580037
[1]    0.419963
Name: Marital_Status, dtype: float64
```

[0] indicates single users which is 58%, [1] indicates married users which is 42%.

[12]:
```
data=df.groupby(['User_ID'])['City_Category'].unique()
data.value_counts()/len(data)
```

[12]:
```
[C]    0.532847
[B]    0.289764
[A]    0.177389
Name: City_Category, dtype: float64
```

1. their are A, B, C cities whereas 53% of the users belong to city category C whereas 29% to city category B and 18% belong to city category A.

2. city category B and A contributes less percentage of total population.

[13]:
```
pd.
↪crosstab(index=df["City_Category"],columns=df["Age"],margins=True,normalize="index")
```

[13]:
```
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
```

[14]:
```
df.groupby(['User_ID'])['Purchase'].count().nlargest(10)
```

[14]:
```
User_ID
1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
1003618     767
1001150     752
1001015     740
1005795     729
1005831     727
Name: Purchase, dtype: int64
```

[15]:
```
df.groupby(['User_ID'])['Purchase'].sum().nlargest(10)
```

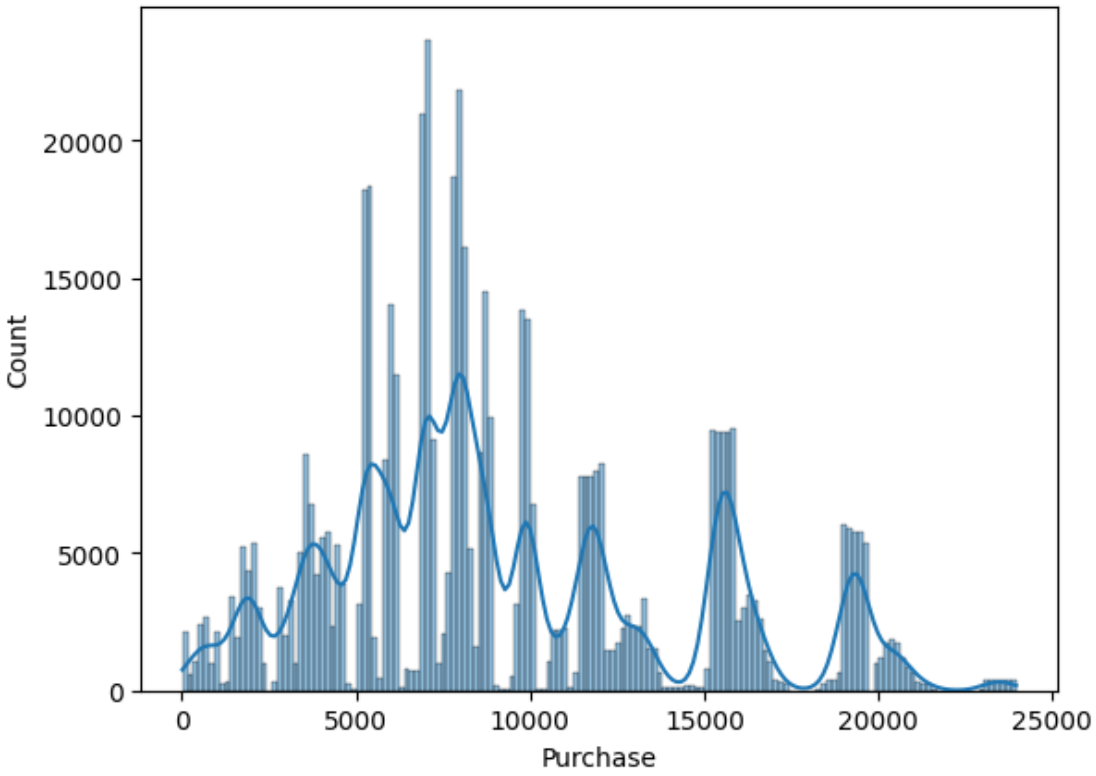
[15]:
```
User_ID
1004277    10536909
1001680     8699596
1002909     7577756
1001941     6817493
1000424     6573609
1004448     6566245
1005831     6512433
1001015     6511314
1003391     6477160
1001181     6387961
```

```
Name: Purchase, dtype: int64
```

The users with high number of purchases contribute more to the purchase amount.

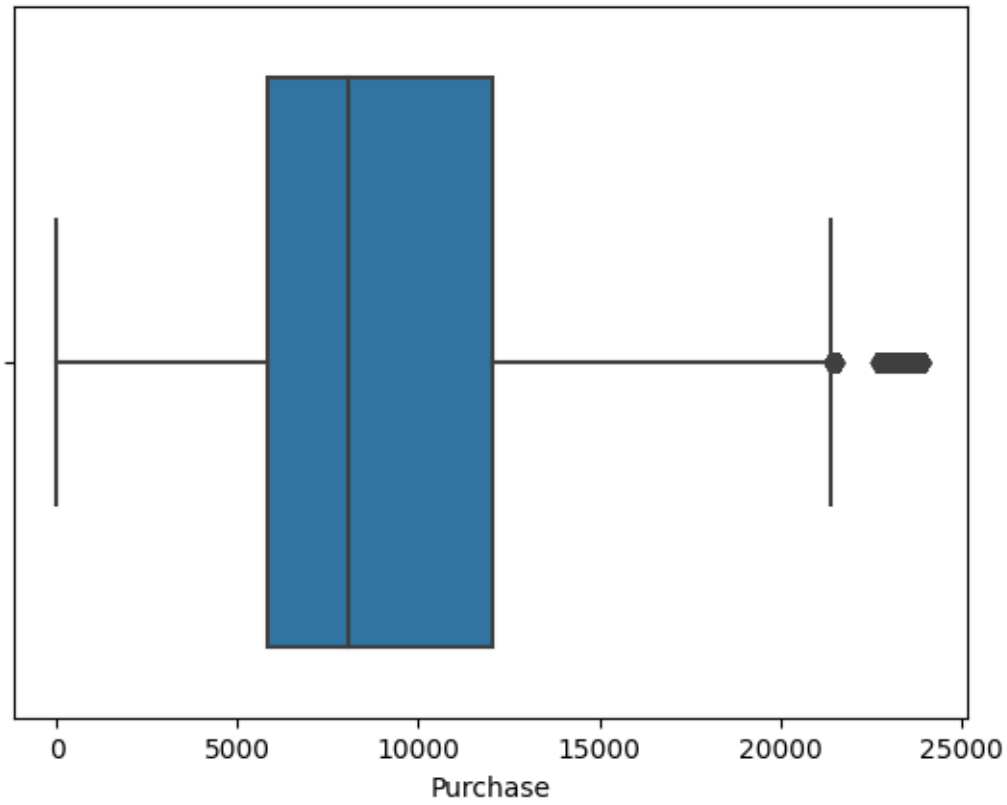
###UNIVARIATE ANALYSIS###

[16]:
```
sns.histplot(data=df, x="Purchase", kde=True)
plt.show()
```



From the initial observation we can observe that purchases are more between 5000 to 10000.

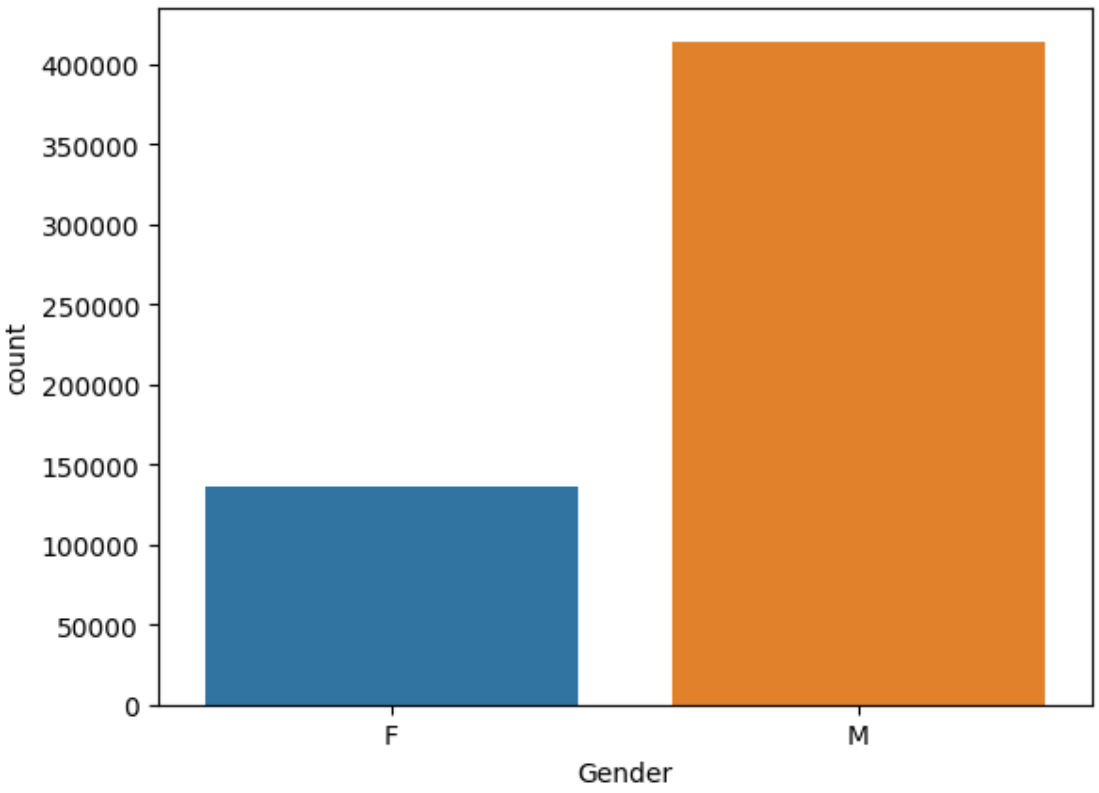
[17]:
```
sns.boxplot(data=df, x='Purchase')
plt.show()
```



from the above boxplot we see the outliers are present in purchase data.

[18]:
```
sns.countplot(data=df, x='Gender')
```

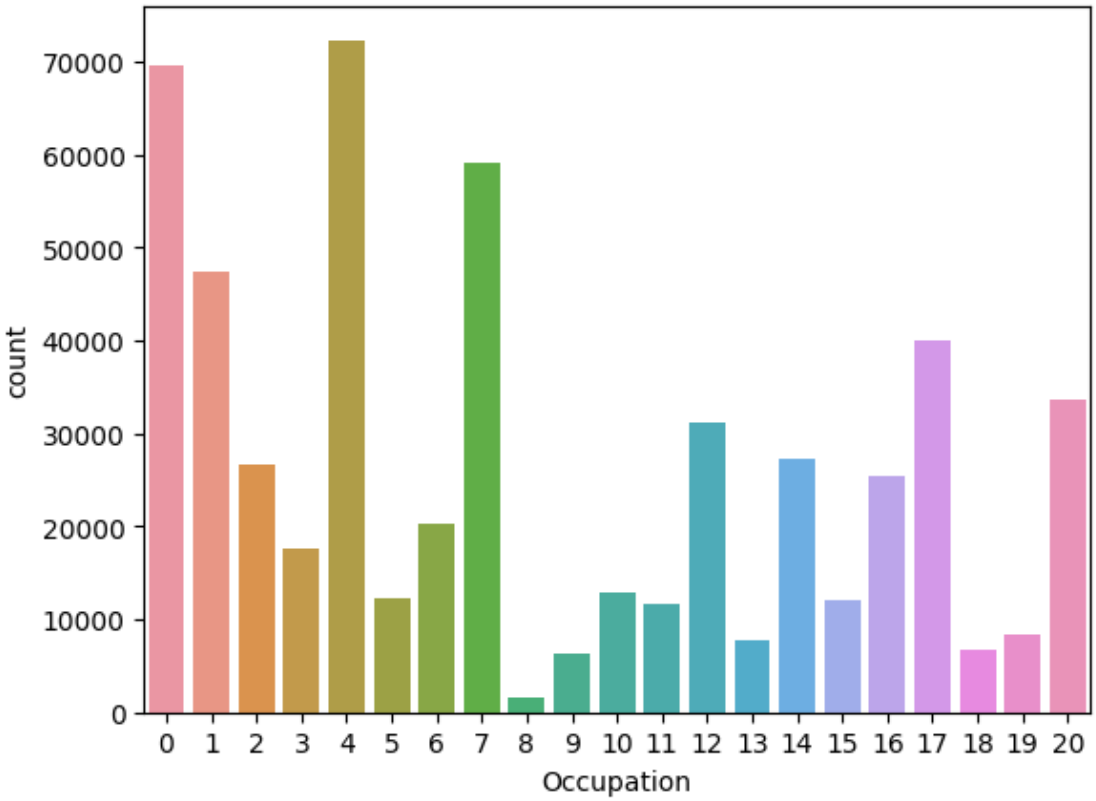
[18]:
```
<Axes: xlabel='Gender', ylabel='count'>
```



purchases done by males are too much higher than females.

[19]:
```
sns.countplot(data=df, x='Occupation')
```

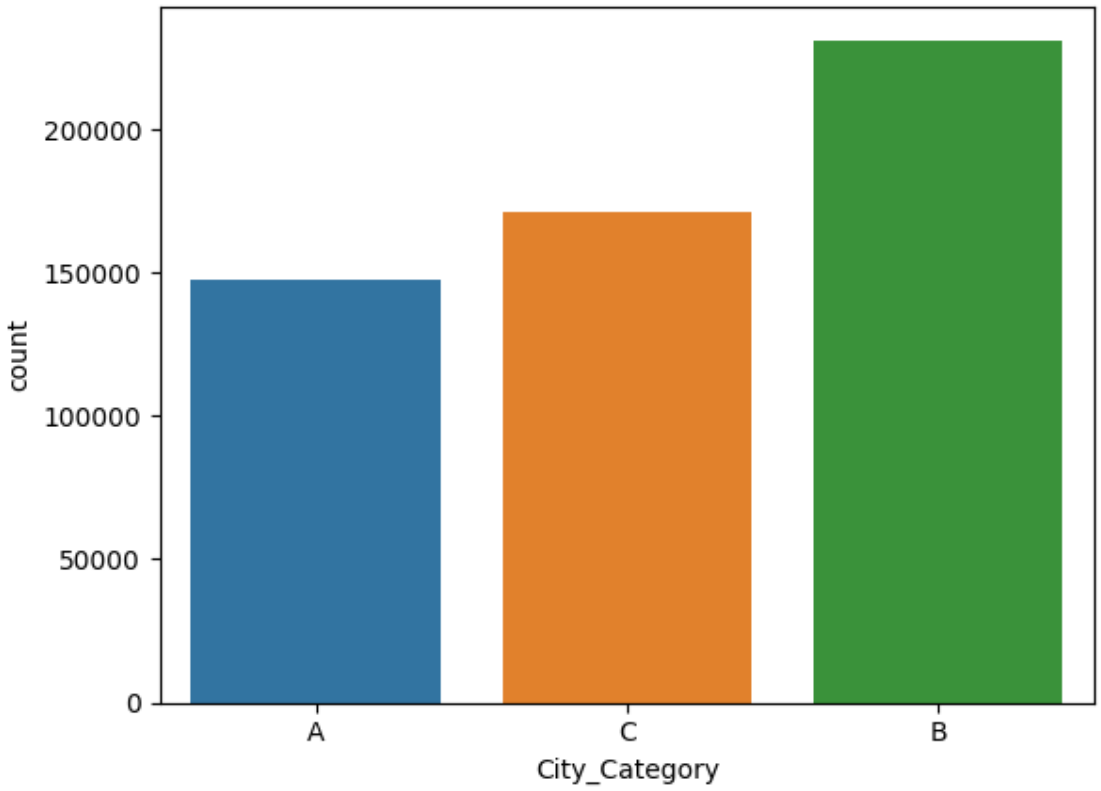
[19]:
```
<Axes: xlabel='Occupation', ylabel='count'>
```



their are total 20 occupations categories, where 4th is highest followed by category 0 and 7.

[20]:
```
sns.countplot(data=df, x='City_Category')
```

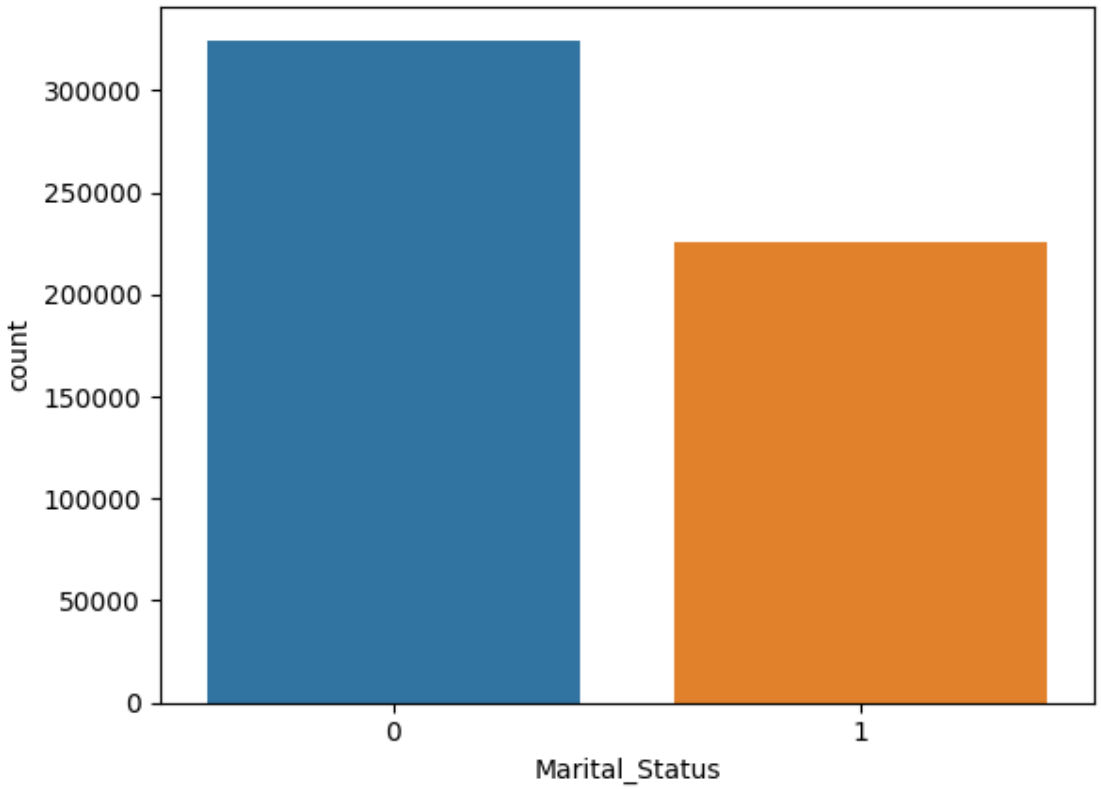
[20]:
```
<Axes: xlabel='City_Category', ylabel='count'>
```



City category B having highest purchase rate.

[21]:
```
sns.countplot(data=df, x='Marital_Status')
```

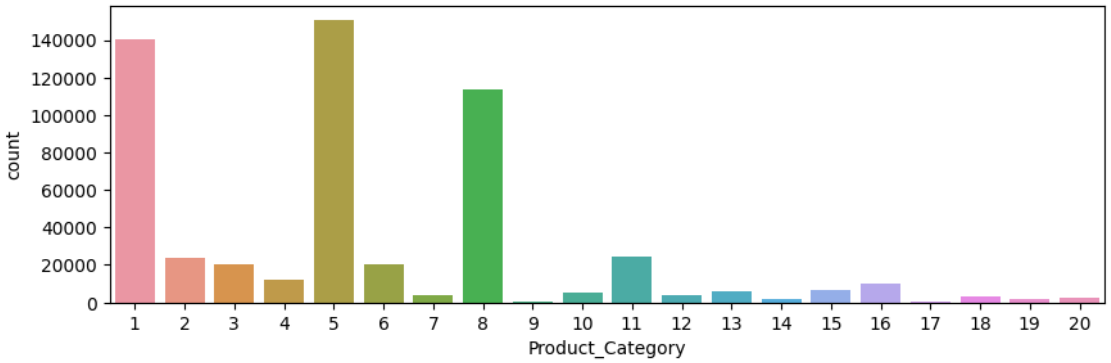
[21]:
```
<Axes: xlabel='Marital_Status', ylabel='count'>
```



single users are more active for purchasing compared to married people.

[22]:
```
plt.figure(figsize=(10, 3))
sns.countplot(data=df, x='Product_Category')
```

[22]:
```
<Axes: xlabel='Product_Category', ylabel='count'>
```

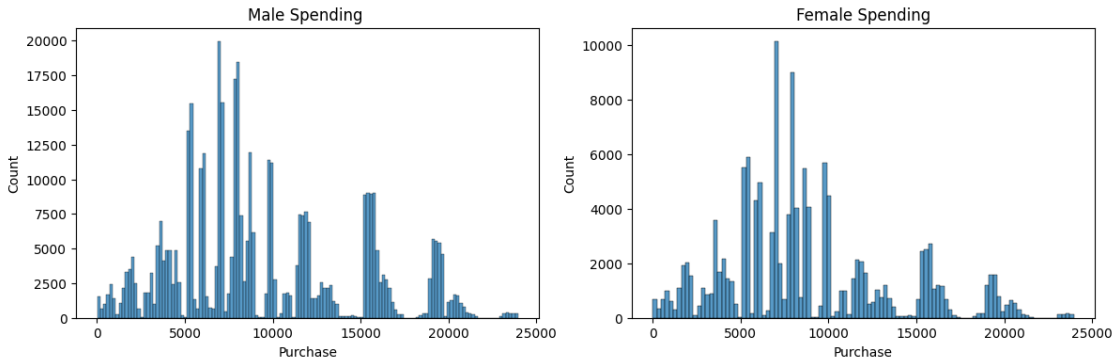


product category is the highest for purchasing followed by 1 aned 8.

###Bivariate Analysis####

[23]:
```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(14,4))
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")
```

[23]:
```
Text(0.5, 1.0, 'Female Spending')
```

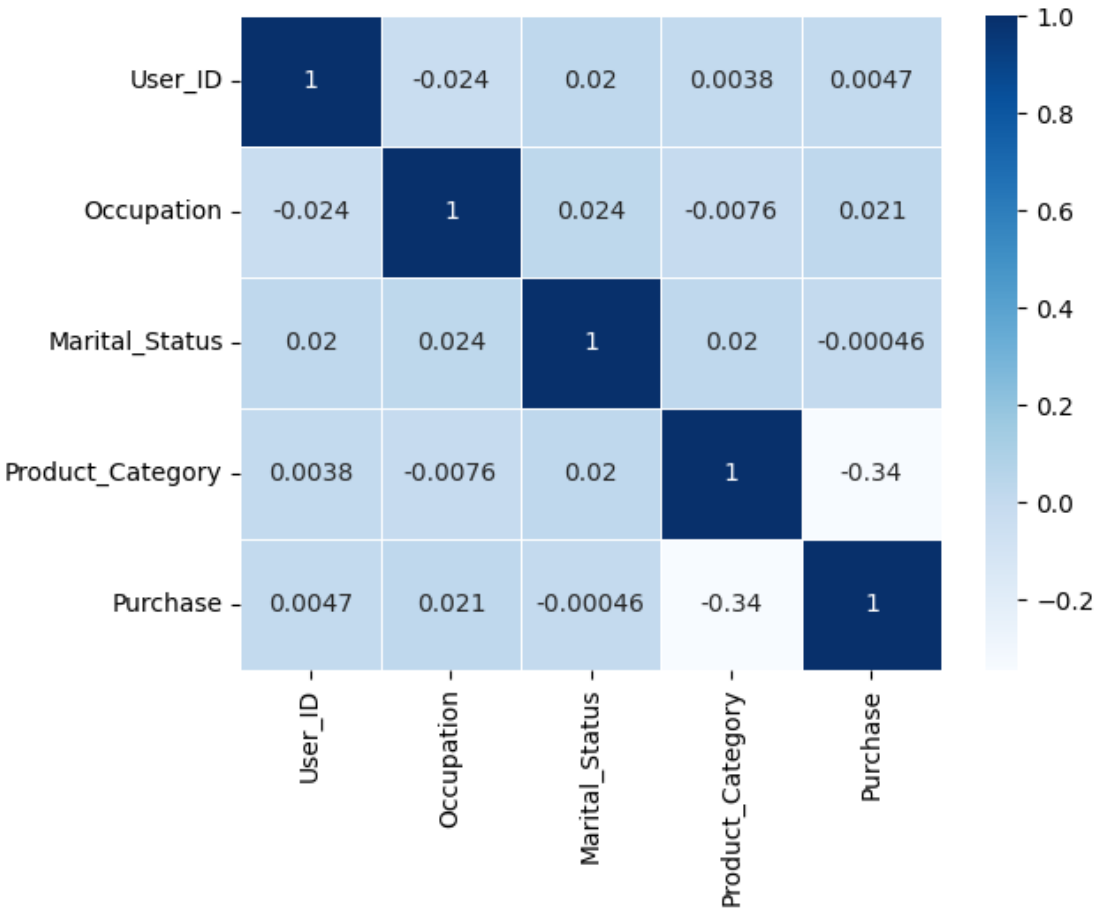


From the above histplots, we can clearly observed spending behaviour is slightly similar in nature.

[24]:
```
sns.heatmap(dtest.corr(), annot=True, cmap="Blues", linewidth=.5)
```

```
<ipython-input-24-9e9405f7d8a6>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(dtest.corr(), annot=True, cmap="Blues", linewidth=.5)
```

[24]:
```
<Axes: >
```



we can see the correlation is not significant between any pair.

[25]:
```
avg_gender= df.groupby(['User_ID', 'Gender'])[['Purchase']].sum()
avg_gender = avg_gender.reset_index()
```

[26]:
```
avg_gender['Gender'].value_counts()
```

[26]:
```
M    4225
F    1666
Name: Gender, dtype: int64
```

[27]:
```
avg_gender.groupby(['Gender'])[['Purchase']].mean()
```

[27]:
```
            Purchase
Gender
F       712024.394958
M       925344.402367
```

[28]:
```
avg_gender.groupby(['Gender'])['Purchase'].sum()
```

[28]:
```
Gender
F    1186232642
M    3909580100
Name: Purchase, dtype: int64
```

[29]:
```
avg_male = avg_gender[avg_gender['Gender']=='M']
avg_female = avg_gender[avg_gender['Gender']=='F']
```

[30]:
```
genders = ["M", "F"]
sample_size = 1000
nums = 1000
male_means = []
female_means = []

for i in range(nums):
  male_mean = avg_male.sample(sample_size, replace=True)['Purchase'].mean()
female_mean = avg_female.sample(sample_size, replace=True)['Purchase'].mean()
male_means.append(male_mean)
female_means.append(female_mean)
```

#1. For Calculating 90% confidence interval for sample size 1000

[31]:
```
# for taking z value for 90%,  95%,  99% from z-table.

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(avg_male['Purchase'].mean()))
print("Population avg_spend amount for Female: {:.2f}\n".
 ↪format(avg_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)))
```

```
Population avg_spend amount for Male: 925344.40
Population avg_spend amount for Female: 712024.39

Sample avg spend amount for Male: 974239.49
Sample avg spend amount for Female: 713005.55
```

1. Average spending amount by male customers is 925344.40
2. Average spending amount by female customers is 712024.39

#2. For calculating 95% confidence interval for sample size 1000

[32]:
```
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(avg_male['Purchase'].mean()))
print("Population avg_spend amount for Female: {:.2f}\n".
 ↪format(avg_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)))
```

```
Population avg_spend amount for Male: 925344.40
Population avg_spend amount for Female: 712024.39

Sample avg spend amount for Male: 974239.49
Sample avg spend amount for Female: 713005.55
```

1. Average spending amount by male customers is 925344.40 with sample avg is 974239.49
2. Average spending amount by female customers is 925344.40 with sample avg is 713005.55

#1. For calculating 99% confidence interval for sample size 1000

[46]:
```
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(avg_male['Purchase'].mean()))
print("Population avg spend amount for Female: {:.2f}\n".
 ↪format(avg_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)))
```

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

Sample avg spend amount for Male: 974239.49
Sample avg spend amount for Female: 713005.55
```

#Based on marital and single stage.

[55]:
```
avg_Marital = df.groupby(['User_ID', 'Marital_Status'])[['Purchase']].sum()
avg_Marital = avg_Marital.reset_index()

avgam_married = avg_Marital[avg_Marital['Marital_Status']==1]
avgam_single = avg_Marital[avg_Marital['Marital_Status']==0]

sample_size = 1000
nums = 1000
married_means = []
single_means = []

for i in range(nums):
  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)
```

[35]:
```
avg_Marital['Marital_Status'].value_counts()
```

[35]:
```
0    3417
1    2474
Name: Marital_Status, dtype: int64
```

#1. for calculating 90% Confidence Interval for Married/single.

[56]:
```
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(avgam_married['Purchase'].mean()))
print("Population avg spend amount for Single: {:.2f}\n".
 ↪format(avgam_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)))
```

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

Sample avg spend amount for Married: 815316.34
Sample avg spend amount for Single: 879021.39
```

#2. for calculating 95% Confidence Interval for Married/single.

[57]:
```
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(avgam_married['Purchase'].mean()))
print("Population avg spend amount for Single: {:.2f}\n".
 ↪format(avgam_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)))
```

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

Sample avg spend amount for Married: 815316.34
Sample avg spend amount for Single: 879021.39
```

#3. For calculating 99% Confidence Interval for Married/single.

[58]:
```
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(avgam_married['Purchase'].mean()))
print("Population avg spend amount for Single: {:.2f}\n".
 ↪format(avgam_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)))
```

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

Sample avg spend amount for Married: 815316.34
Sample avg spend amount for Single: 879021.39
```

0.0.1 Based on Age

[39]:
```
avg_age = df.groupby(['User_ID', 'Age'])[['Purchase']].sum()
avg_age = avg_age.reset_index()
avg_age['Age'].value_counts()
```

[39]:
```
26-35    2053
36-45    1167
18-25    1069
46-50     531
51-55     481
55+       372
0-17      218
Name: Age, dtype: int64
```

#1. For calculating 90% of confidence interval for age.

[40]:
```
z90=1.645 #90% Confidence Interval
z95=1.960 #95% Confidence Interval
z99=2.576 #99% Confidence Interval

sample_size = 200
nums = 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=avg_age[avg_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(nums):mean = avg_age[avg_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 = avg_age[avg_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)
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)
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)
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)
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)
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)
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)
```

1 2. For calculating 95% of confidence interval for age.

[41]:
```
z90=1.645 #90% Confidence Interval
z95=1.960 #95% Confidence Interval
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(nums):
    mean = avg_age[avg_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 = avg_age[avg_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)
```

2 3. For calculating 99% of confidence interval for age.

[42]:
```
z90=1.645 #90% Confidence Interval
z95=1.960 #95% Confidence Interval
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(nums):
    mean = avg_age[avg_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 = avg_age[avg_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 observe that the population means for above are quite close to the sample mean.

#RECOMMENDATION#

1.Single customers spend more money than married customers, so that walmart should be more focus on that perticuler good preffered by single users. 2. Figure illustrates that mens are more active in purchasing than womens, so company should be retain the mens customers. 3. Fompany should take attention where the business is more for example city category 'C'. Which makes the company profit higher. 4. Age group 26-35 is the most purchaser than other age group, so the company should focus on acquisition. 5. company should provide discounts and offers for given period of time, so that they can attract the customers, example: offering them bank discounts, pay later options. 6. Company should think of the product categories where the sell is very low. 7. As per the data womens product purchasing rate is less, so that company should advertise the products for womens too. 8. Lastly, company should also maintain the quality of their products.