Business Case: Walmart - Confidence Interval and CLT

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import statistics
```

Getting know more about Data

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).

In [2]:
```python
sales=pd.read_csv("D://walmart_data.txt")
```

In [3]:
```python
sales
```

Out[3]:

	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 × 10 columns

In [4]:
```python
sales.shape
```

Out[4]:
```
(550068, 10)
```

In [5]:
```python
sales.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
```

In [6]:
```python
sales.describe()
```

Out[6]:

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

In [7]:
```python
sales.isnull()
```

Out[7]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
550063	False	False	False	False	False	False	False	False	False	False
550064	False	False	False	False	False	False	False	False	False	False
550065	False	False	False	False	False	False	False	False	False	False
550066	False	False	False	False	False	False	False	False	False	False
550067	False	False	False	False	False	False	False	False	False	False

550068 rows × 10 columns

In [8]:
```python
sales.isnull().describe()
```

Out[8]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
count	550068	550068	550068	550068	550068	550068	550068	550068	550068	550068
unique	1	1	1	1	1	1	1	1	1	1
top	False	False	False	False	False	False	False	False	False	False
freq	550068	550068	550068	550068	550068	550068	550068	550068	550068	550068

In [9]:
```python
sales.isnull().sum().sum()
```

Out[9]:
```
0
```

In [10]:
```python
sales
```

Out[10]:

	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 × 10 columns

Outliers (using boxplot, "describe" method by checking the difference between mean and median, isnull etc.)

In [11]:
```python
P25 = sales['Purchase'].quantile(0.25)
P50 = sales['Purchase'].quantile(0.50)
P75 = sales['Purchase'].quantile(0.75)
```

In [12]:
```python
IQR = P75 - P25
IQR
```

Out[12]:
```
6231.0
```

In [13]:
```python
lower = max(0,P25-(1.5*IQR))
lower
```

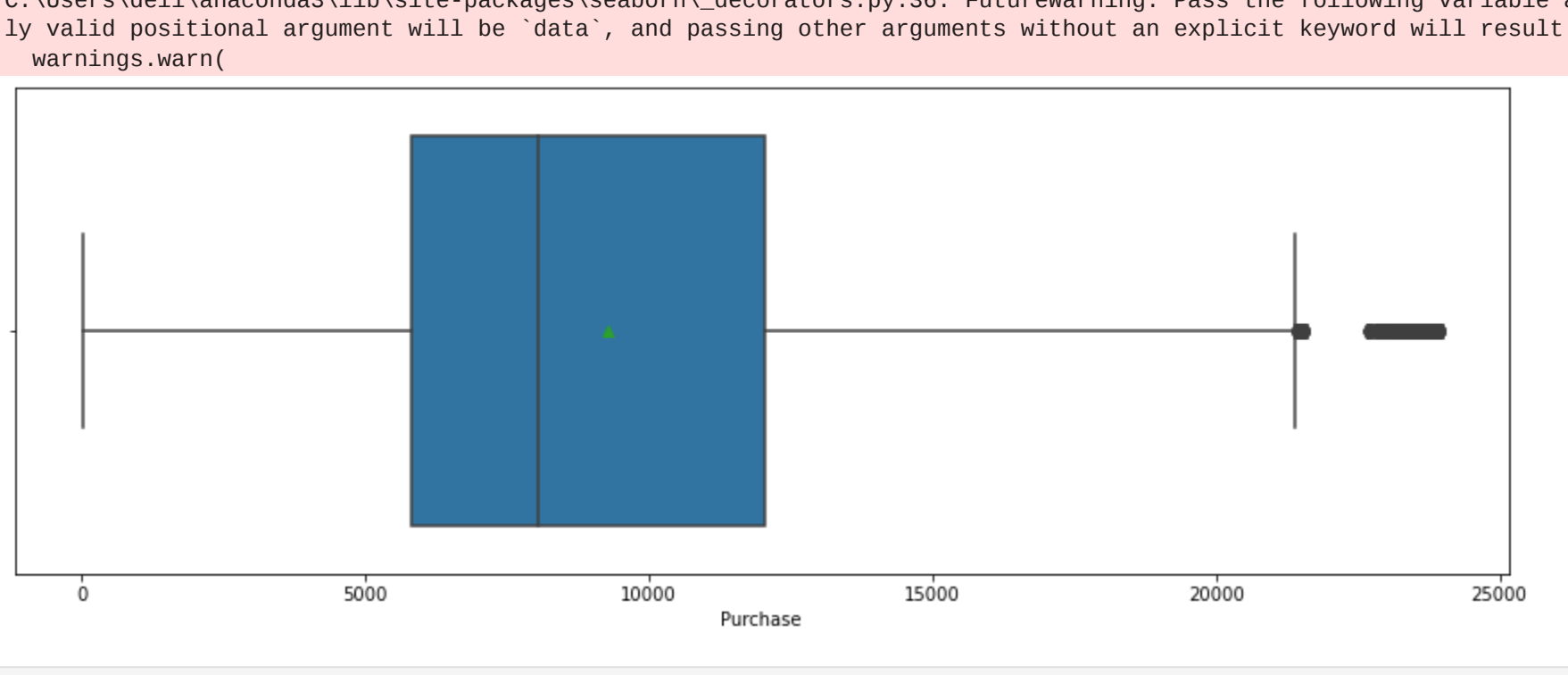
Out[13]:
```
0
```

In [14]:
```python
upper = P75 + (1.5*IQR)
upper
```

Out[14]:
```
21400.5
```

In [15]:
```python
plt.figure(figsize=(15,5))
sns.boxplot(sales['Purchase'],orient = "h",showmeans= True)
plt.show()
```

```
C:\Users\dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [16]:
```python
outliers = sales[sales['Purchase']>upper]
outliers
```

Out[16]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
343	1000058	P00117642	M	26-35	2	B	3	0	10	23603
375	1000062	P00119342	F	36-45	3	A	1	0	10	23792
652	1000126	P00087042	M	18-25	9	B	1	0	10	23233
736	1000139	P00159542	F	26-35	20	C	2	0	10	23595
1041	1000175	P00052842	F	26-35	2	B	1	0	10	23341
...	...	...	...	...	...	...	...	...	...	...
544488	1005815	P00116142	M	26-35	20	B	1	0	10	23753
544704	1005847	P00085342	F	18-25	4	B	2	0	10	23724
544743	1005852	P00202242	F	26-35	1	A	0	1	10	23529
545663	1006002	P00116142	M	51-55	0	C	1	1	10	23663
545787	1006018	P00052842	M	36-45	1	C	3	0	10	23496

2677 rows × 10 columns

In [17]:
```python
statistics.mean(sales['Purchase'])
```

Out[17]:
```
9263.968712959126
```

In [18]:
```python
statistics.median(sales['Purchase'])
```

Out[18]:
```
8047.0
```

In [19]:
```python
product25=sales['Product_Category'].quantile(0.25)
product50=sales['Product_Category'].quantile(0.50)
product75=sales['Product_Category'].quantile(0.75)
product_IQR = product75 - product25
product_lower = max(0,product25-(1.5*product_IQR))
product_upper = product75+(1.5*product_IQR)
```

In [20]:
```python
statistics.mean(sales['Product_Category'])
```

Out[20]:
```
5.404270017525106
```

In [21]:
```python
statistics.median(sales['Product_Category'])
```

Out[21]:
```
5.0
```

In [22]:
```python
outlier_product = sales[sales['Product_Category']>product_upper]
outlier_product
```

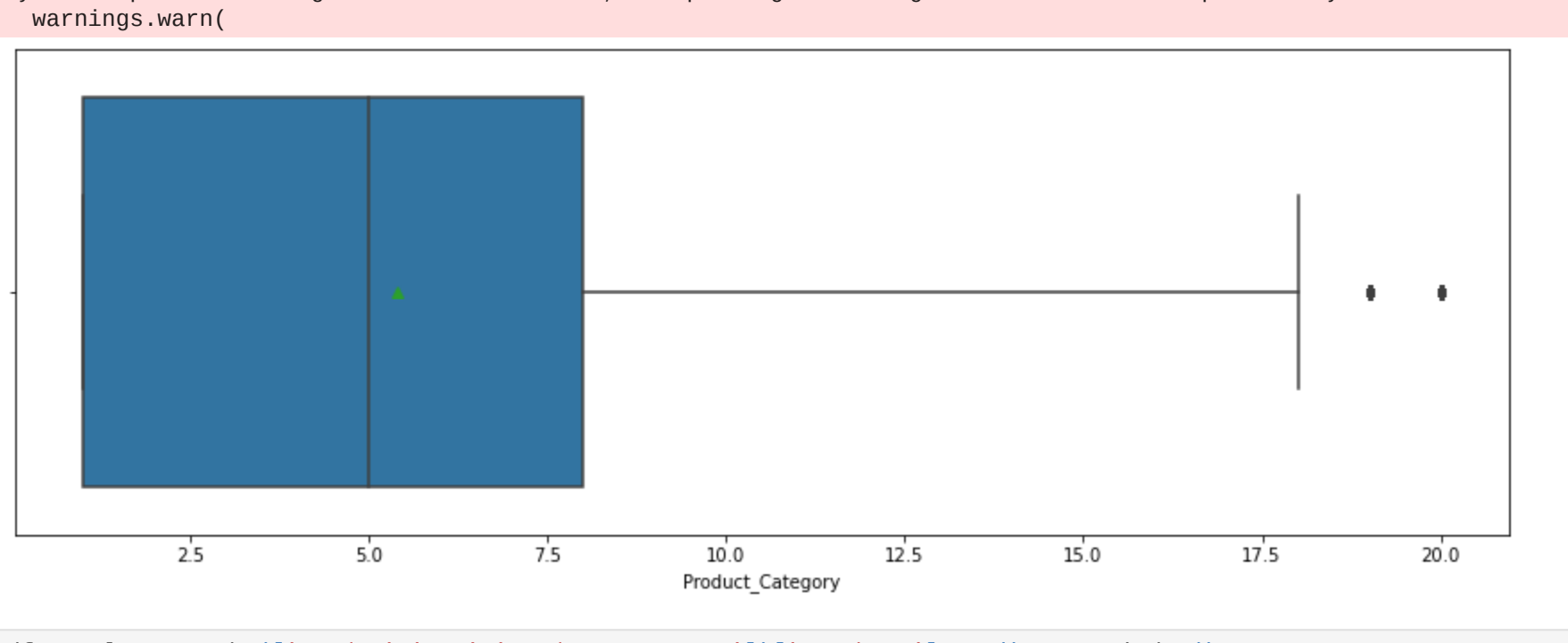
Out[22]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
545915	1000001	P00375436	F	0-17	10	A	2	0	20	612
545916	1000002	P00372445	M	55+	16	C	4+	0	20	119
545917	1000004	P00375436	M	46-50	7	B	2	1	20	481
545918	1000006	P00375436	F	51-55	9	A	1	0	20	480
545919	1000007	P00372445	M	36-45	1	B	1	1	20	241
...	...	...	...	...	...	...	...	...	...	...
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

4153 rows × 10 columns

In [23]:
```python
plt.figure(figsize=(15,5))
sns.boxplot(sales['Product_Category'],orient = "h",showmeans= True)
plt.show()
```

```
C:\Users\dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [24]:
```python
df_1=sales.groupby(['Gender','Age','Product_Category'])['Purchase'].sum().reset_index()
df_2=df_1[df_1['Purchase']>5000000]
```

In [25]:
```python
df_2.mean(axis='index')
```

```
C:\Users\dell\AppData\Local\Temp\ipykernel_17168\2325815496.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df_2.mean(axis='index')
```

Out[25]:
```
Product_Category    6.647619e+00
Purchase            4.668389e+07
dtype: float64
```

In [26]:
```python
sales['Purchase'].unique()
```

Out[26]:
```
array([ 8370, 15200,  1422, ...,   135,   123,   613], dtype=int64)
```

In [27]:
```python
sales['Product_Category'].unique()
```

Out[27]:
```
array([ 3,  1, 12,  8,  5,  4,  2,  6, 14, 11, 13, 15,  7, 16, 18, 10, 17,
        9, 20, 19], dtype=int64)
```

In [28]:
```python
sales['Marital_Status'].unique()
```

Out[28]:
```
array([0, 1], dtype=int64)
```

In [29]:
```python
df_2=sales.groupby(['User_ID'])['Gender'].unique()
df_2.value_counts()/len(df_2)
```

Out[29]:
```
[M]    0.717196
[F]    0.282804
Name: Gender, dtype: float64
```

In [30]:
```python
df_2=sales.groupby(['User_ID'])['Marital_Status'].unique()
df_2.value_counts()/len(df_2)
```

Out[30]:
```
[0]    0.580037
[1]    0.419963
Name: Marital_Status, dtype: float64
```

In [31]:
```python
df_2=sales.groupby(['User_ID'])['City_Category'].unique()
df_2.value_counts()/len(df_2)
```

Out[31]:
```
[C]    0.532847
[B]    0.289764
[A]    0.177389
Name: City_Category, dtype: float64
```

In [32]:
```python
df_2=pd.DataFrame(sales.groupby(['Gender'])[['Purchase']].sum())
df_2['percent'] = (df_2['Purchase'] / df_2['Purchase'].sum()) * 100
df_2
```

Out[32]:

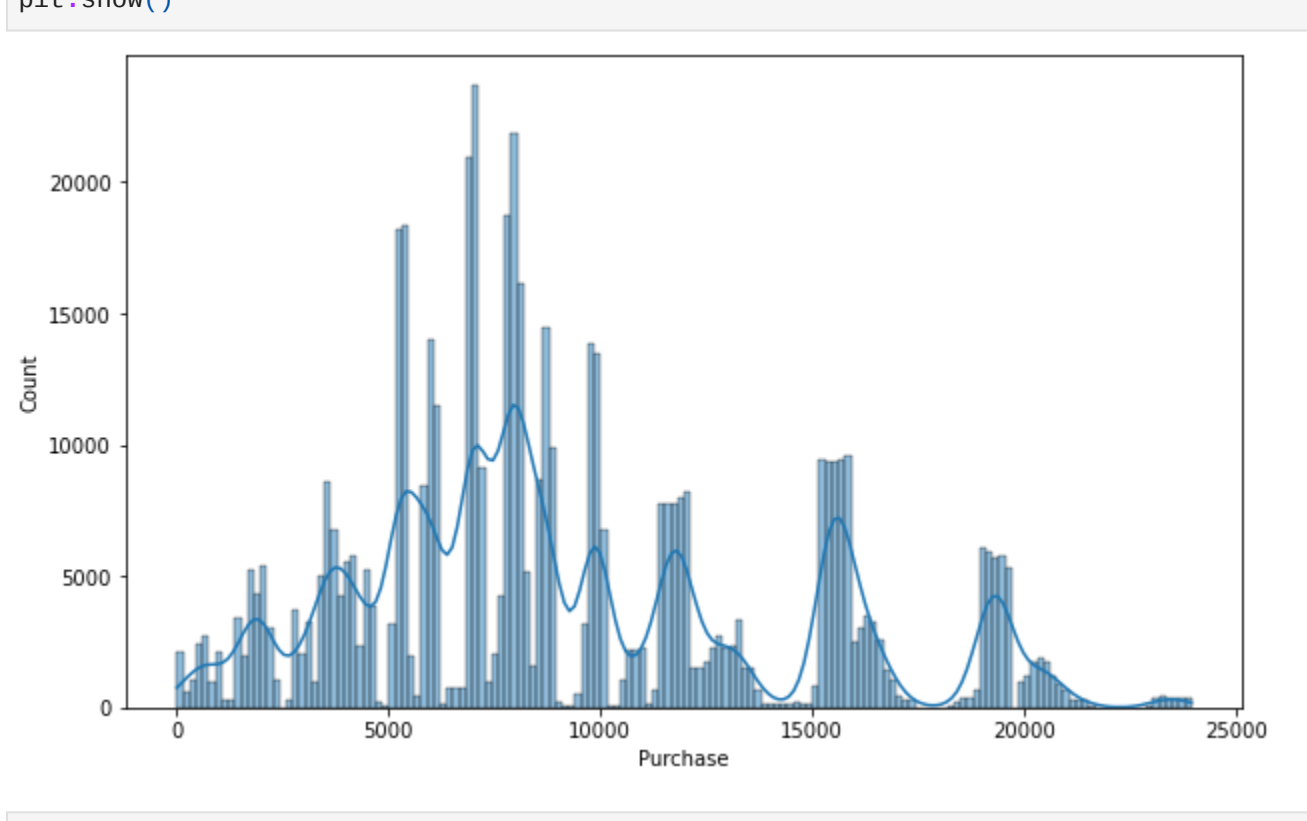
Gender	Purchase	percent
F	1186232642	23.278576
M	3909580100	76.721424

In [33]:
```python
df_2=pd.DataFrame(sales.groupby(['Age'])[['Purchase']].sum())
df_2['percent'] = (df_2['Purchase'] / df_2['Purchase'].sum()) * 100
df_2
```

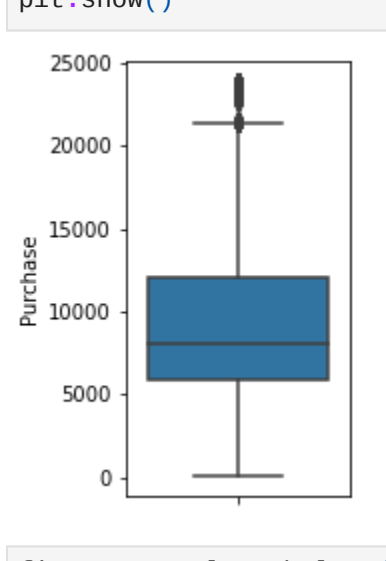
Out[33]:

Age	Purchase	percent
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

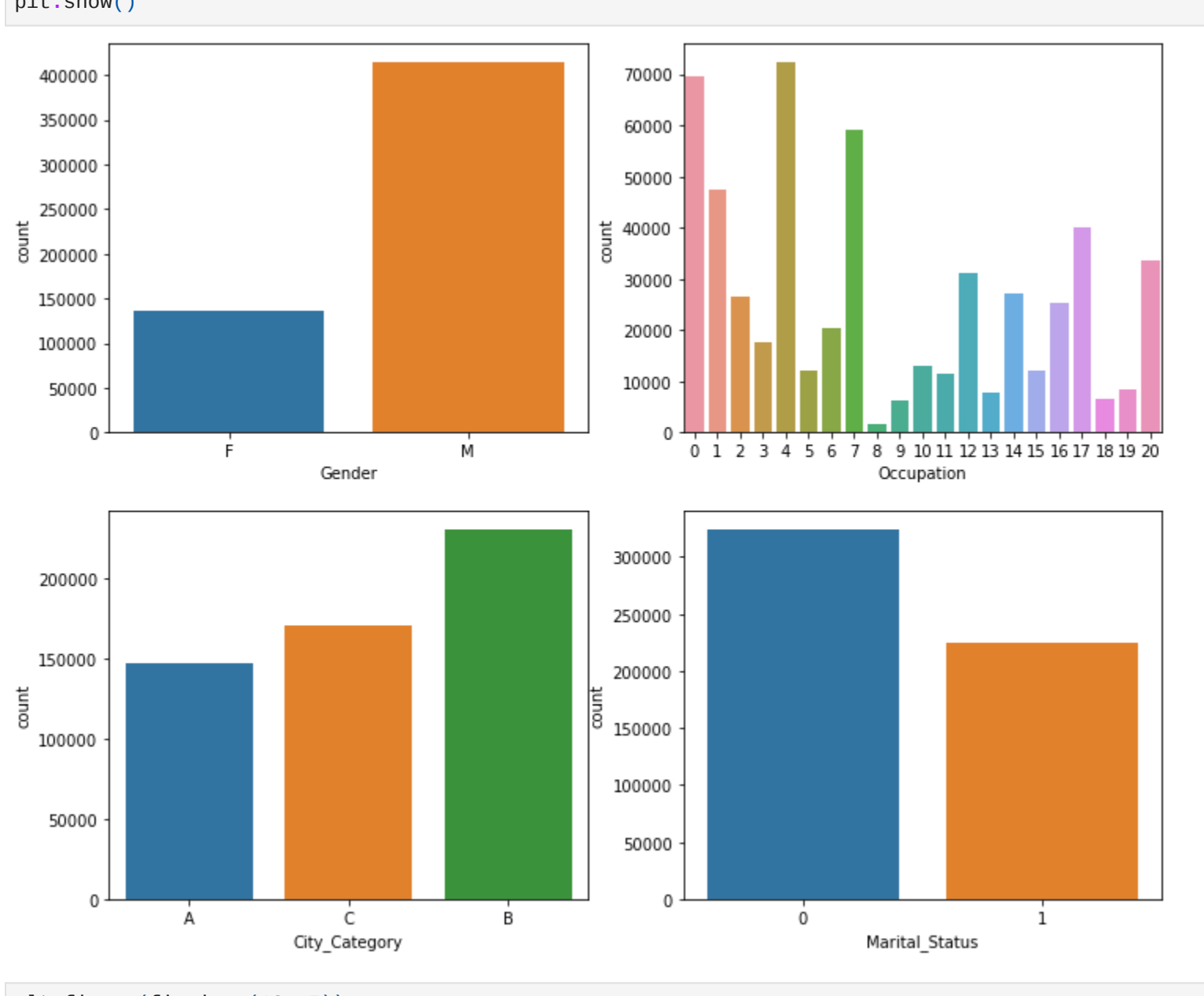
In [34]:
```python
plt.figure(figsize=(10, 6))
sns.histplot(data=sales, x="Purchase", kde=True)
plt.show()
```



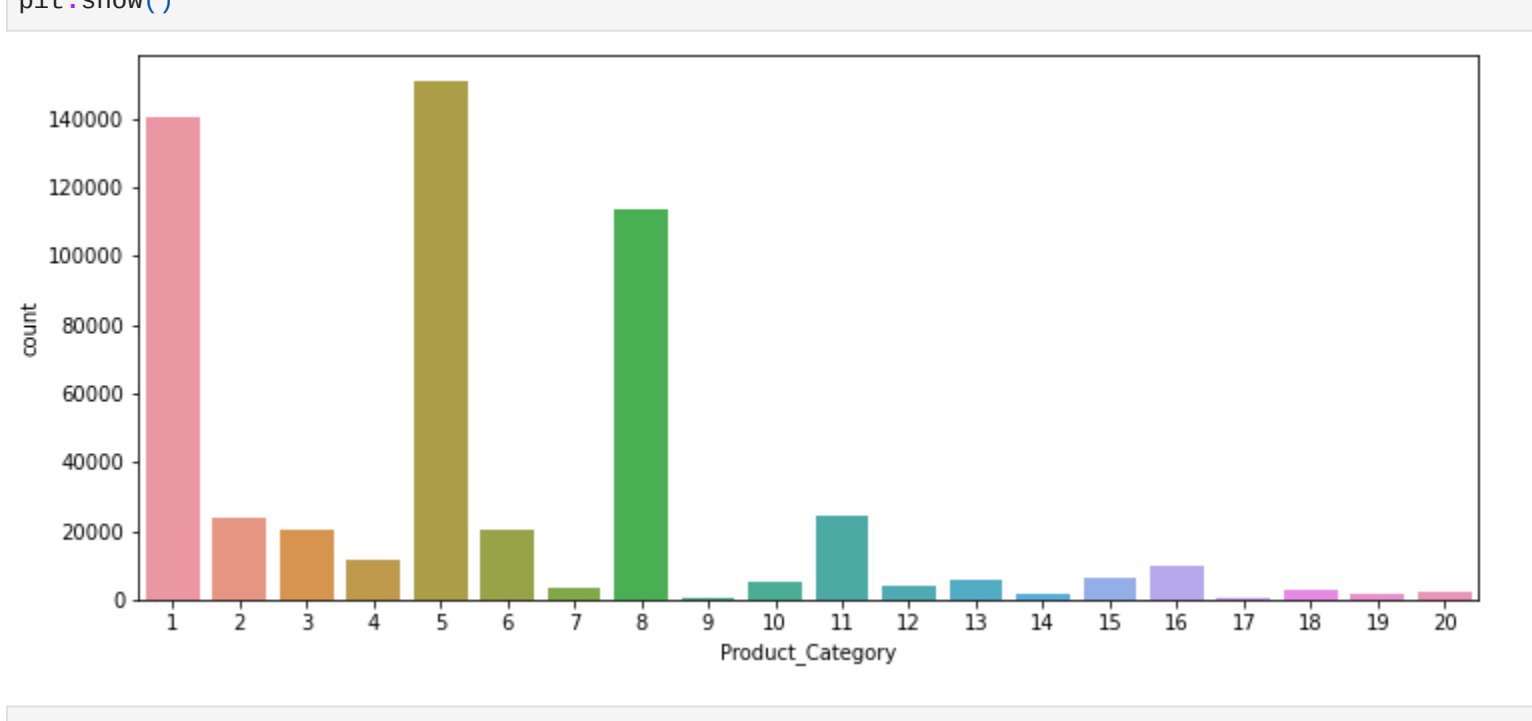
In [35]:
```python
plt.figure(figsize=(2, 4))
sns.boxplot(data=sales, y='Purchase')
plt.show()
```



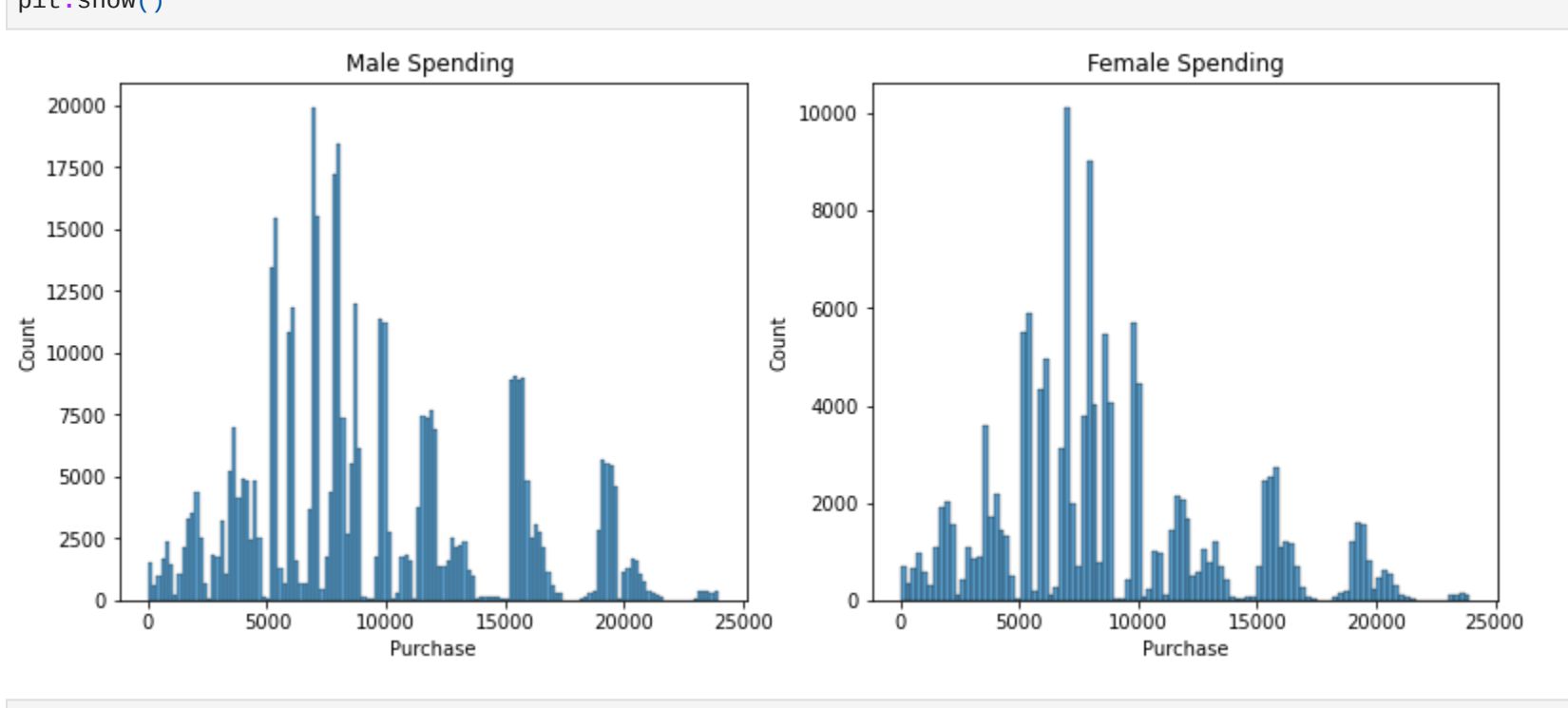
In [36]:
```python
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(12, 10))
sns.countplot(data=sales, x='Gender', ax=axs[0,0])
sns.countplot(data=sales, x='Occupation', ax=axs[0,1])
sns.countplot(data=sales, x='City_Category', ax=axs[1,0])
sns.countplot(data=sales, x='Marital_Status', ax=axs[1,1])
plt.show()
```



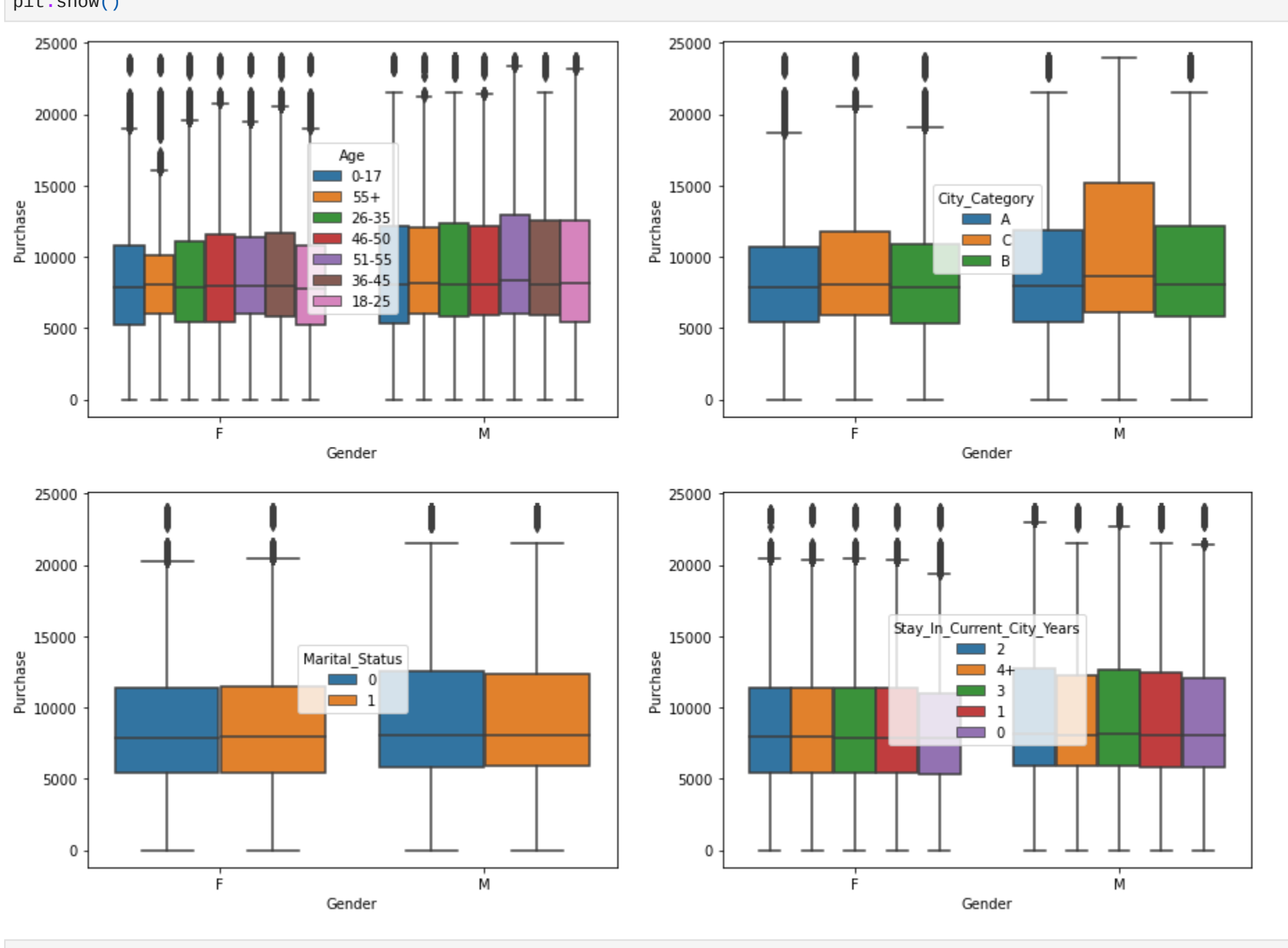
In [37]:
```python
plt.figure(figsize=(12, 5))
sns.countplot(data=sales, x='Product_Category')
plt.show()
```



In [38]:
```python
#bivairate
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(13,5))
sns.histplot(data=sales[sales['Gender']=='M']['Purchase'], ax=axs[0]).set_title("Male Spending ")
sns.histplot(data=sales[sales['Gender']=='F']['Purchase'], ax=axs[1]).set_title("Female Spending")
plt.show()
```

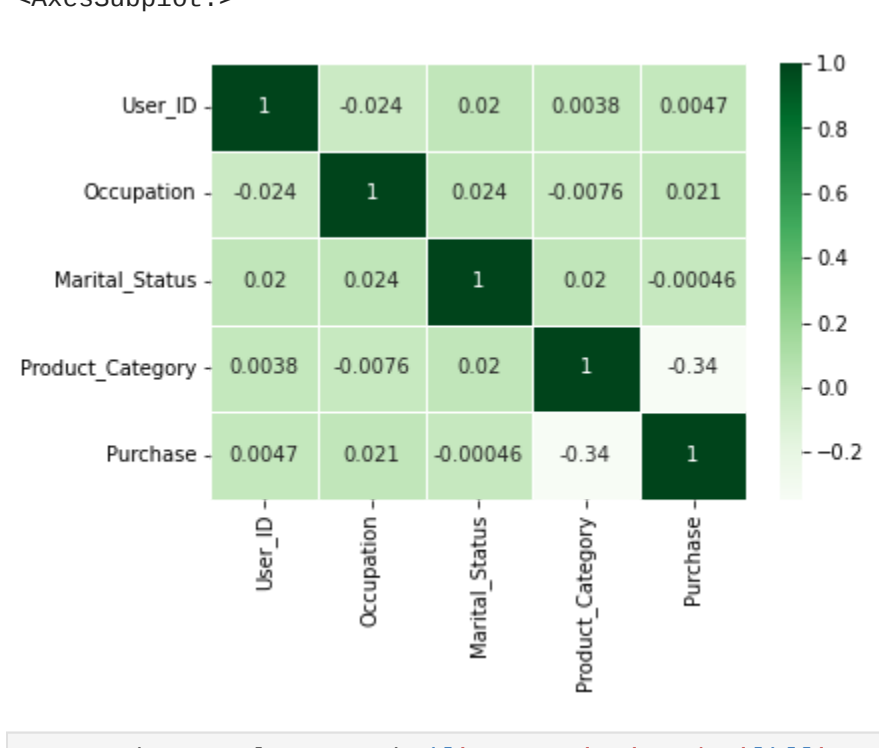


In [39]:
```python
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(15, 6))
fig.subplots_adjust(top=1.5)
sns.boxplot(data=sales, y='Purchase', x='Gender', hue='Age', ax=axs[0,0])
sns.boxplot(data=sales, y='Purchase', x='Gender', hue='City_Category', ax=axs[0,1])

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

plt.show()
```



In [40]:
```python
sns.heatmap(sales.corr(), annot=True, cmap="Greens", linewidth=0.5)
```

Out[40]:
```
<AxesSubplot:>
```



In [41]:
```python
avg_gender = sales.groupby(['User_ID', 'Gender'])[['Purchase']].sum()
avg_gender = avg_gender.reset_index()
avg_gender
```

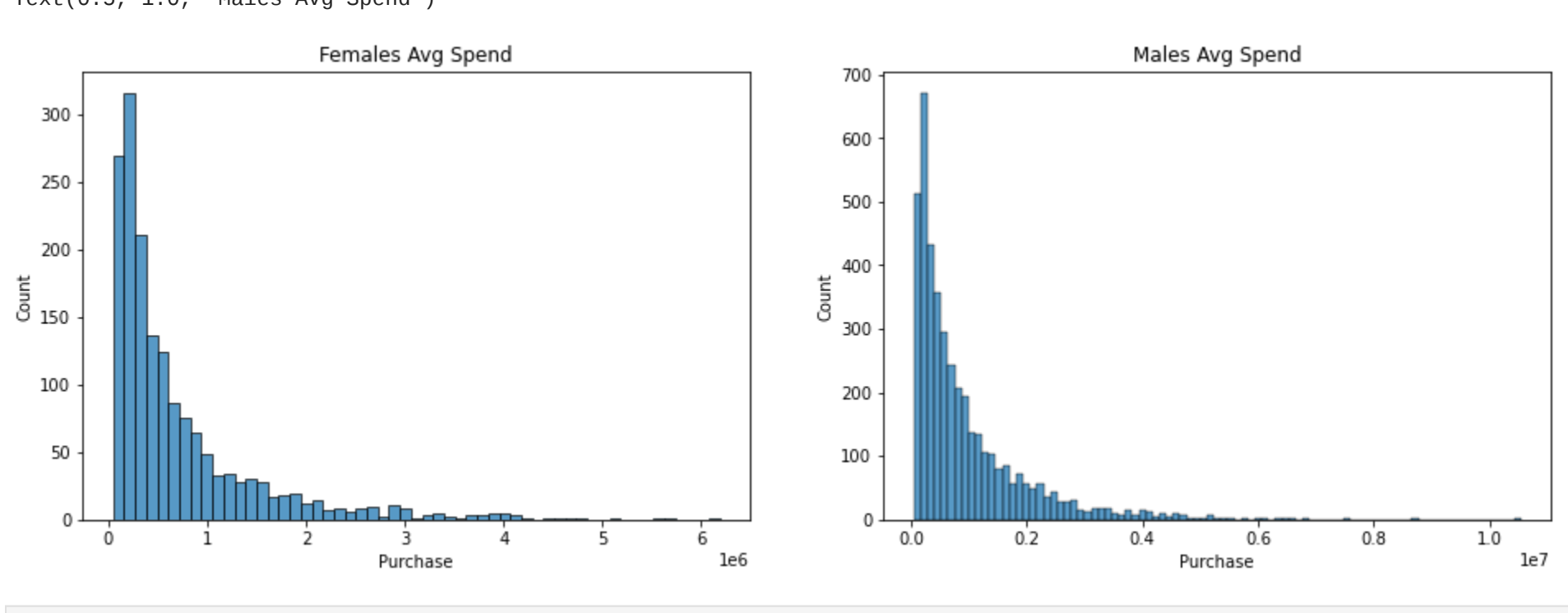
Out[41]:

	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 × 3 columns

In [42]:
```python
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(16,5))

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

Out[42]:
```
Text(0.5, 1.0, 'Males Avg Spend')
```



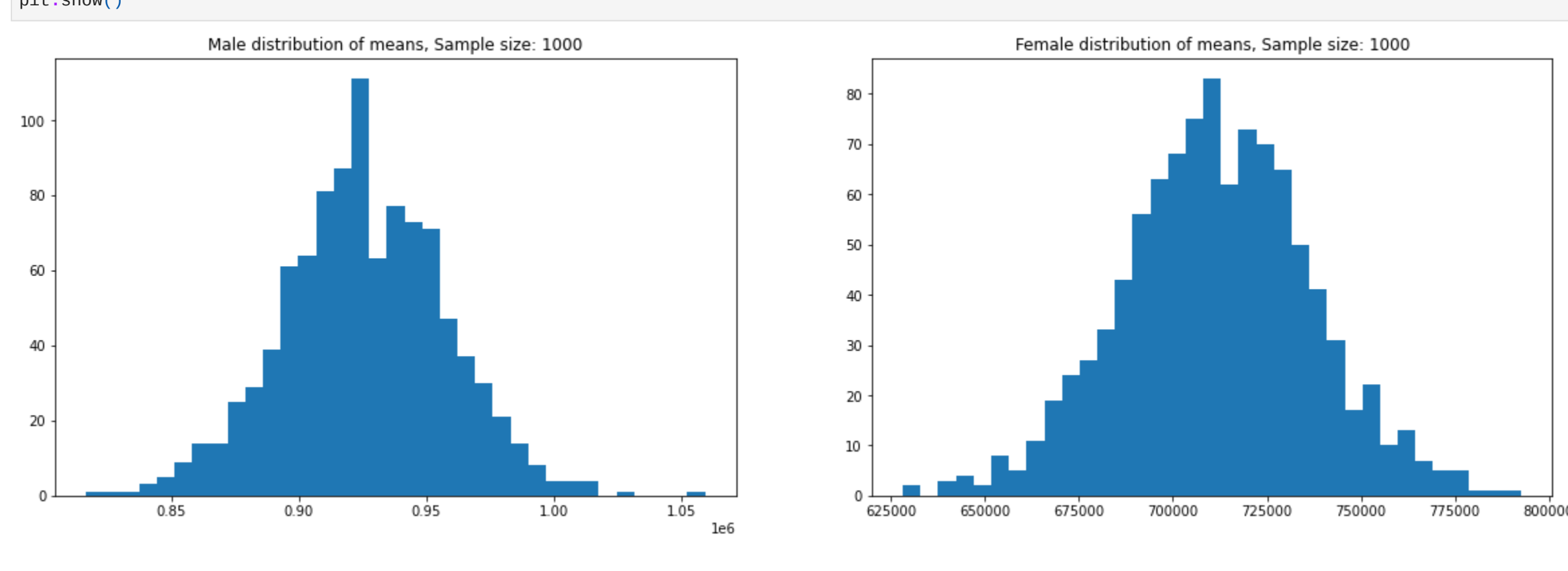
In [43]:
```python
avg_male = avg_gender[avg_gender['Gender']=='M']
avg_female = avg_gender[avg_gender['Gender']=='F']
```

In [44]:
```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 = 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)
```

In [45]:
```python
#Final insight
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)))

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: 925714.29
Sample avg spend amount for Female: 711438.62

Sample std for Male: 32102.82
Sample std for Female: 25292.68

Sample std error for Male: 1015.18
Sample std error for Female: 799.82

Male_CI:  [924044.3196454166, 927384.2628745834]
Female_CI:  [710122.9126074797, 712754.3361485203]
```

In [46]:
```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()
```



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 overcome and uppers of female purchase CI are lesser than lower limits of male purchase CI. This proves that men usually spend more than women Males might be doing the purchase for females.Salary can be a factor in less purchase We also need to see whether male-based products were sold more than women products to clearly identify difference in spending pattern.

1. Confidence intervals and distribution of the mean of the expenses by female and male customers. Ans:At 99% Confidence Interval with sample size 1000 Average amount spend by male customers lie in the lower parameters Average amount spend by female customers lie in higher parameter
2. 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 notover come.Attracting more female customers are chances to increase female purchases
3. Results when the same activity is performed for Married vs Unmarried Ans: 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]
4. Results when the same activity is performed for Age Ans:its really based up on the diffrent factors and the ratios that are more act on trends and presonal prefarnce and taste pattrens

Recomdations

1.Female expenditure is less than Males but the sale of women-base products are more selling compare to other domain products 2.certain category products have high rage of demand comapre to others focus shold be more on them and appealing on the trends 3.accoding to age group of 20-35 in few cities have highest expenditure those might age group produts night attract more business 4.married couples spend less un married couple spend more so more focus on products which unmarried couple attract 5.The top products should be given focus in order to maintain the quality in order to further increase the sales of those products.

In [47]:

In [ ]: