Problem statement: In this case study we are going to analyze the customer purchase behavior based on the gender and other factors provided to us in form of the dataset and how based on those factors Walmart can use it to make better business decisions. The data provided to us is from a black friday sale and we need to draw insights and recommendations based on the spending habits of different genders

In [1]:

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

In [2]:

```
df=pd.read_csv('C:/Prakruthi/DSML/WALMART - Case study/walmart_data.csv')
```

In [3]:

```
df.head(5)
```

Out[3]:

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

In [4]:

```
#Checking if there are missing values:
df.isnull().sum()
```

Out[4]:

```
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
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
User_ID                       550068 non-null int64
Product_ID                    550068 non-null object
Gender                        550068 non-null object
Age                           550068 non-null object
Occupation                    550068 non-null int64
City_Category                 550068 non-null object
Stay_In_Current_City_Years    550068 non-null object
Marital_Status                550068 non-null int64
Product_Category              550068 non-null int64
Purchase                      550068 non-null int64
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

In [6]:

```
df['User_ID'].nunique()
df['Product_ID'].nunique()
df.groupby('Product_ID')['Product_ID'].value_counts().sort_values(ascending=False)
df['Age'].nunique()
df.groupby('Age')['Age'].value_counts().sort_values(ascending=False)
df['Occupation'].nunique()
df['Purchase'].max()
df['Purchase'].min()
```

Out[6]:

```
12
```

Basic metrics: We have a total of 550069 rows which represent the number of transactions(which will also be our population dataset) and 10 attributes/columns which are going to use for our gender based analysis and help to draw better conclusions.

Below are some of the metrics based on the givem dataset

1. Since there are no missing values, no action renuired
2. There are a total of 3631 products and product P00265242 has been sold the most i.e 1880 times
3. There are a total of 7 age groups. Most purchases have been done by the age group 26-35
4. There are a total of 21 occupation types
5. The price of the purchases range from 12 to 23961 which means a wide range of products have been purchased

In [7]:

```
df["User_ID"] = df["User_ID"].astype('object')
df["Product_ID"] = df["Product_ID"].astype('object')
df["Gender"] = df["Gender"].astype('object')
df["City_Category"] = df["City_Category"].astype('object')
df["Stay_In_Current_City_Years"] = df["Stay_In_Current_City_Years"].astype('object')
df["Marital_Status"] = df["Marital_Status"].astype('object')
df["Occupation"] = df["Occupation"].astype('object')
df["Product_Category"] = df["Product_Category"].astype('object')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
User_ID                       550068 non-null object
Product_ID                    550068 non-null object
Gender                        550068 non-null object
Age                           550068 non-null object
Occupation                    550068 non-null object
City_Category                 550068 non-null object
Stay_In_Current_City_Years    550068 non-null object
Marital_Status                550068 non-null object
Product_Category              550068 non-null object
Purchase                      550068 non-null int64
dtypes: int64(1), object(9)
memory usage: 42.0+ MB
```

In [8]:

```
df.describe(include='all')
```

Out[8]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_Ci
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	

Some additional observations

1. A total of 5891 users are present
2. 3 city categories are available for analysis. Tier-A, Tier-B, Tier-C
3. 2 marital statuses are available. 0 and 1 which is unmarried and married

In [9]:

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

Out[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
```

Here we have the below observations

1. As observed before, the age group 26-35 have the highest percentage of users with 34.85% customers belonging to this age group
2. Age group 18-45 comprise of almost 73% of users which makes it the target age group for marketing

In [10]:

```
df_gender = df.groupby(['User_ID'])['Gender'].unique()
df_gender.value_counts()/len(df_age)
```

Out[10]:

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

Percentage of male customers are way more than female customers at 71.71% and 28.28% respectively

Using contingency table to check the number of male versus female customers across different age groups

In [11]:

```
male_female_age = pd.crosstab(index=df["Gender"], columns=df["Age"], margins=True)
male_female_age
```

Out[11]:

Age	0-17	18-25	26-35	36-45	46-50	51-55	55+	All
Gender								
F	5083	24628	50752	27170	13199	9894	5083	135809
M	10019	75032	168835	82843	32502	28607	16421	414259
All	15102	99660	219587	110013	45701	38501	21504	550068

As we have observed before, here as well we can see that the no. of customers are highest in both males and females between the age group 26-35

In [12]:

```
male_female_purchase = pd.crosstab(index=df["Gender"], columns=df["Marital_Status"], margir
male_female_purchase
```

Out[12]:

Marital_Status	0	1	All
Gender			
F	78821	56988	135809
M	245910	168349	414259
All	324731	225337	550068

from the above table we can observe that the no. of unmarried customers are more than the married customers

In [13]:

```
df.groupby('Gender')['Purchase'].sum()
```

Out[13]:

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

Here, we can observe that the purchase amount for Male customers is more than 3 times than that of Female customers

In [14]:

```
df.groupby(['Marital_Status','Gender'])['Purchase'].sum()
```

Out[14]:

```
Marital_Status  Gender
0               F          684154127
                M         2324773320
1               F          502078515
                M         1584806780
Name: Purchase, dtype: int64
```

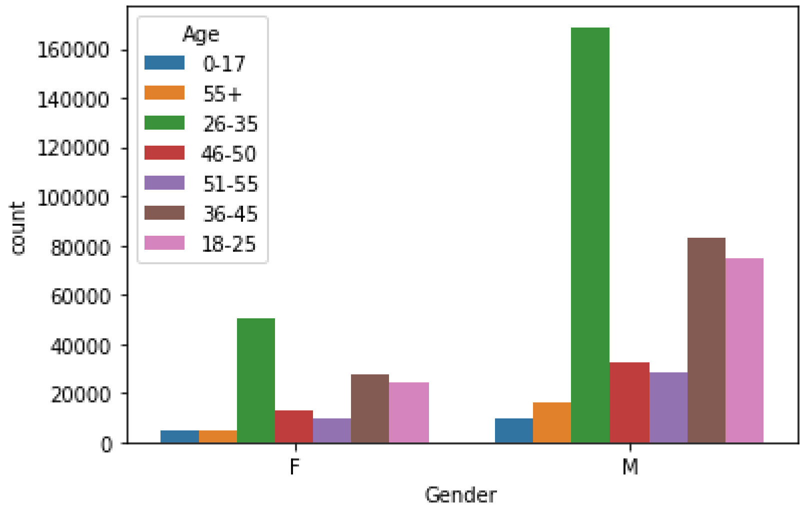
Here it can be observed that the unmarried Male customers have the highest purchase amount compared to the other categories

In [15]:

```
sns.countplot(data=df,
 x="Gender",hue='Age')
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d04655780>
```



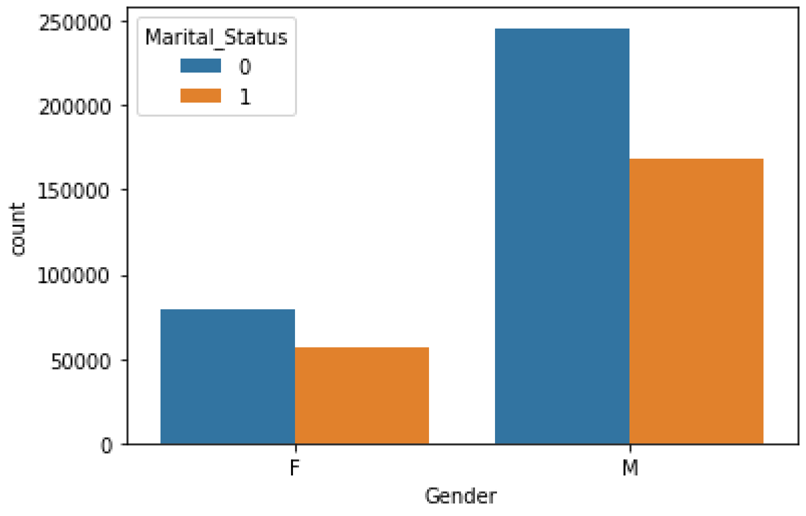
Here from the above countplot, we can clearly see that the number of Male customers in the age group 26-35 is way higher than any other age and also more than the females

In [16]:

```
sns.countplot(data=df,
 x="Gender",hue='Marital_Status')
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d047f7240>
```



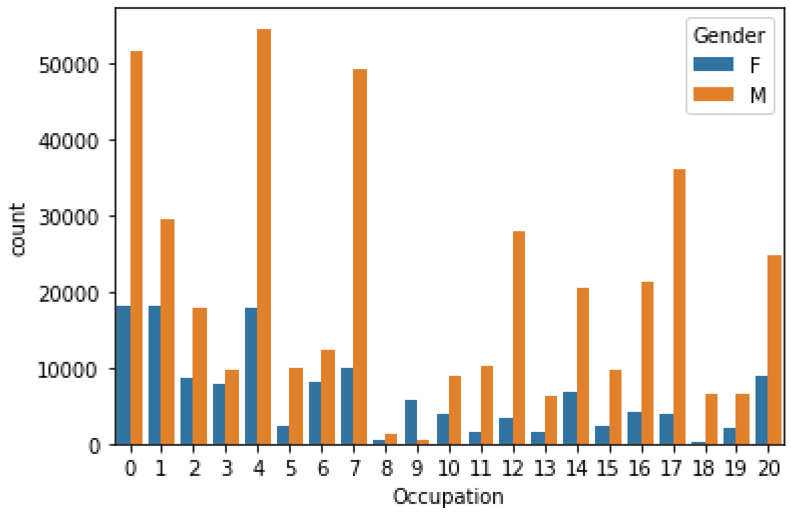
Here we can observe that the unmarried customers have more purchases than the married customers for both genders

In [17]:

```
sns.countplot(data=df,
 x="Occupation",hue='Gender')
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d048aa358>
```



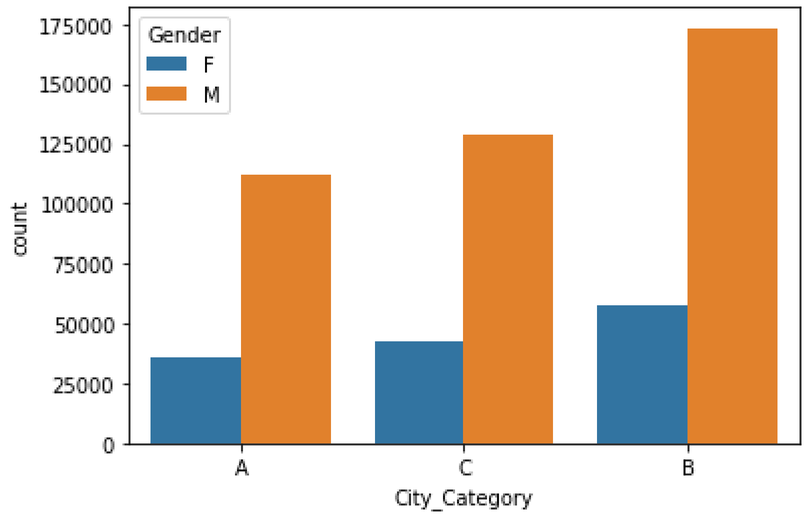
From this countplot we can observe that the occupations 0, 4 and 7 have almost the highest purchases for both genders

In [18]:

```
sns.countplot(data=df,
 x="City_Category",hue='Gender')
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d087172e8>
```



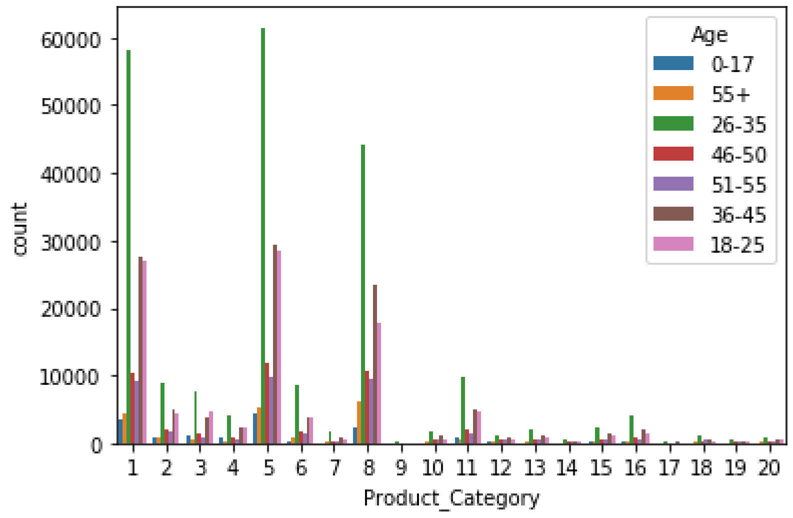
From this countplot we can see that TIER-B cities have the highest purchases for both genders

In [19]:

```
sns.countplot(data=df,
 x="Product_Category",hue='Age')
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d08720b70>
```

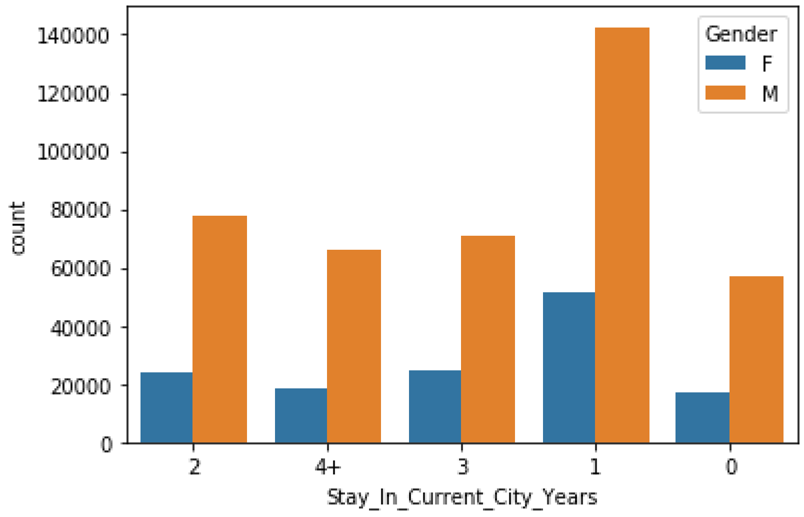


In [20]:

```
sns.countplot(data=df,
 x="Stay_In_Current_City_Years",hue='Gender')
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d088daeb8>
```



Product categories 1,5 and 8 have a lot of purchases compared to other product categories

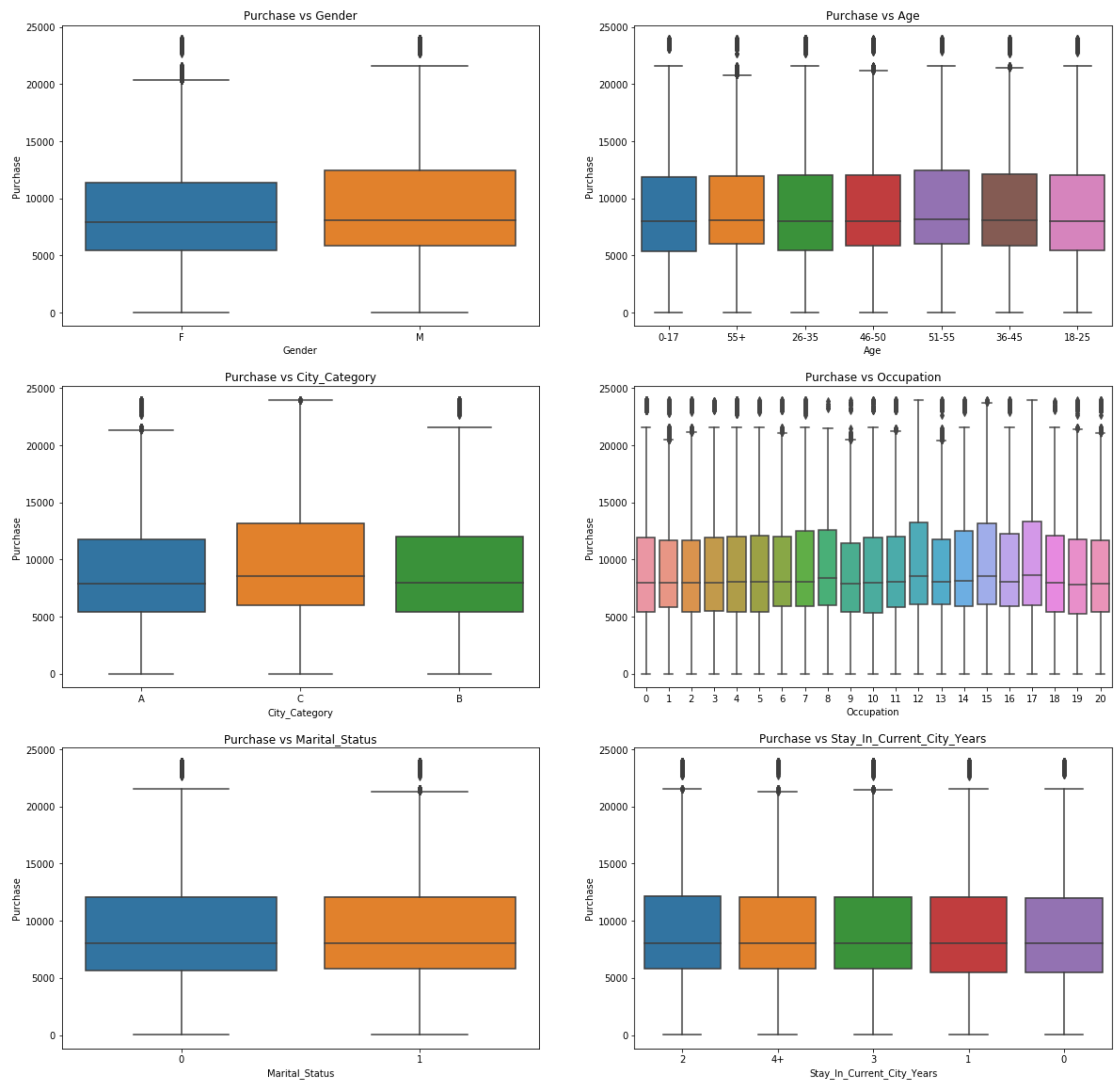
We can observe that the TIER-B cities have the highest purchases for both the genders

In [21]:

```
fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(20, 20))
sns.boxplot(data=df, y='Purchase',x='Gender', ax=axis[0,0]).set_title('Purchase vs Gender')
sns.boxplot(data=df, y='Purchase',x='Age',ax=axis[0,1]).set_title('Purchase vs Age')
sns.boxplot(data=df, y='Purchase',x='City_Category',ax=axis[1,0]).set_title('Purchase vs Ci
sns.boxplot(data=df, y='Purchase',x='Occupation', ax=axis[1,1]).set_title('Purchase vs Occu
sns.boxplot(data=df, y='Purchase',x='Marital_Status',ax=axis[2,0]).set_title('Purchase vs M
sns.boxplot(data=df, y='Purchase',x='Stay_In_Current_City_Years',ax=axis[2,1]).set_title('P
```

Out[21]:

```
Text(0.5, 1.0, 'Purchase vs Stay_In_Current_City_Years')
```



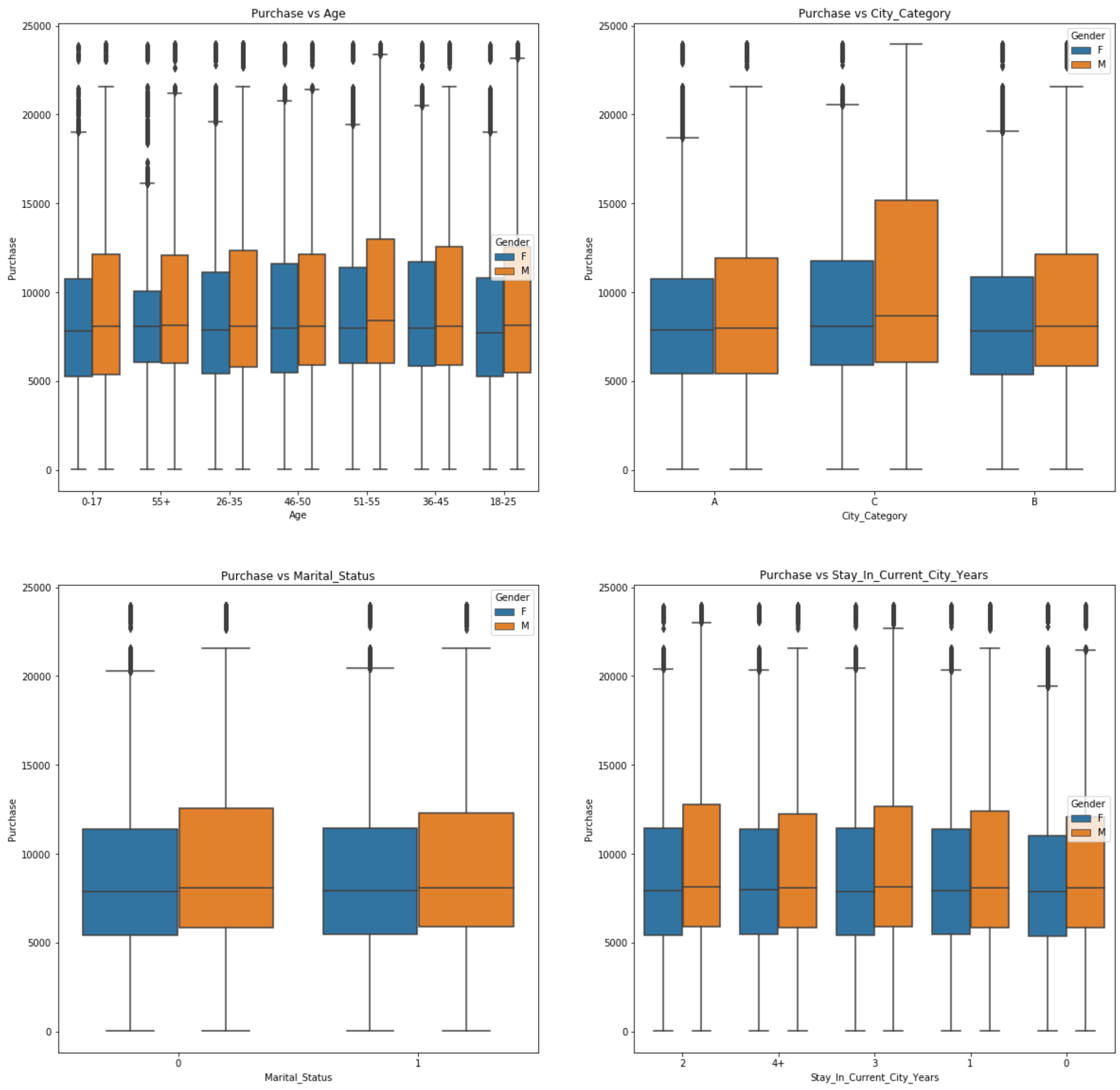
The median of purchase and also the 25th and 75th percentile for all the different categories are somewhat similar which means the purchase amounts for are more or less in the same range for different categories/attributes

In [22]:

```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(20, 20))
sns.boxplot(data=df, y='Purchase',x='Age',hue='Gender',ax=axis[0,0]).set_title('Purchase vs
sns.boxplot(data=df, y='Purchase',x='City_Category',hue='Gender',ax=axis[0,1]).set_title('P
sns.boxplot(data=df, y='Purchase',x='Marital_Status',hue='Gender',ax=axis[1,0]).set_title('
sns.boxplot(data=df, y='Purchase',x='Stay_In_Current_City_Years',hue='Gender',ax=axis[1,1])
```

Out[22]:

```
Text(0.5, 1.0, 'Purchase vs Stay_In_Current_City_Years')
```



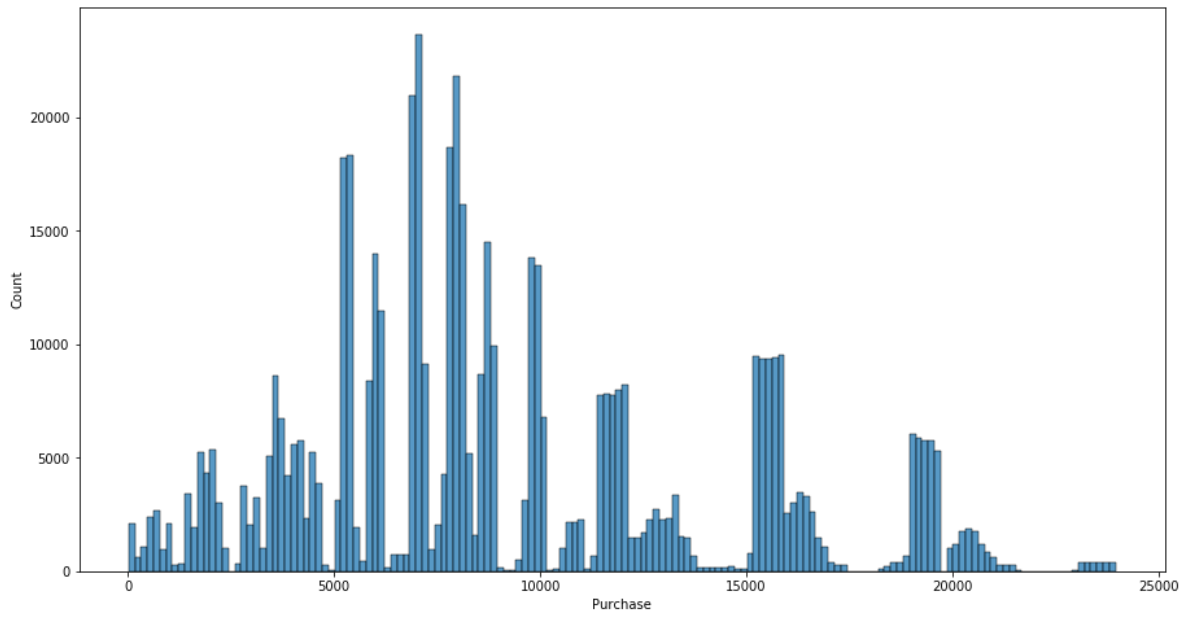
We can observe above that with respect to all other categories, Males have higher purchases compared to females

In [23]:

```
plt.figure(figsize=(15,8))
sns.histplot(df["Purchase"])
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d09bc0358>
```



The purchases follow a somewhat normal distribution for the entire population data provided to us

In [24]:

```
sns.heatmap(df.corr(), annot=True)
```

Out[24]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d09fc1588>
```

1
Purchase
Purchase
1.100
1.075
1.050
1.025
1.000
0.975
0.950
0.925
0.900

We cannot really draw any significant correlation between the categories using the heatmap or pairplots

In [25]:

```
total_purchase_gender = df.groupby(['User_ID', 'Gender'])[['Purchase']].sum()
total_purchase_gender = total_purchase_gender.reset_index()
total_purchase_gender.head(10)
```

Out[25]:

	User_ID	Gender	Purchase
0	1000001	F	334093
1	1000002	M	810472
2	1000003	M	341635
3	1000004	M	206468
4	1000005	M	821001
5	1000006	F	379930
6	1000007	M	234668
7	1000008	M	796593
8	1000009	M	594099
9	1000010	F	2169510

this dataframe can be used further for our analysis to answer many questions

In [26]:

```
 # Getting the total number of males and females in the population data. this can further b
total_purchase_gender['Gender'].value_counts()
```

Out[26]:

```
M    4225
F    1666
Name: Gender, dtype: int64
```

We can clearly observe from above that the population of males are almost 3 times more than the females. This will be our population data for the analysis

In [27]:

```
#Checking the total spendings between males and females
total_purchase_gender.groupby(['Gender'])['Purchase'].sum()
```

Out[27]:

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

As we can see the total male purchases are more than thrice than that of females. This could also be due to the fact that the female population given to us is much lesser compared to males

In [28]:

```
#Checking the average spendings between males and females
total_purchase_gender.groupby(['Gender'])['Purchase'].mean()
```

Out[28]:

```
Gender
F    712024.394958
M    925344.402367
Name: Purchase, dtype: float64
```

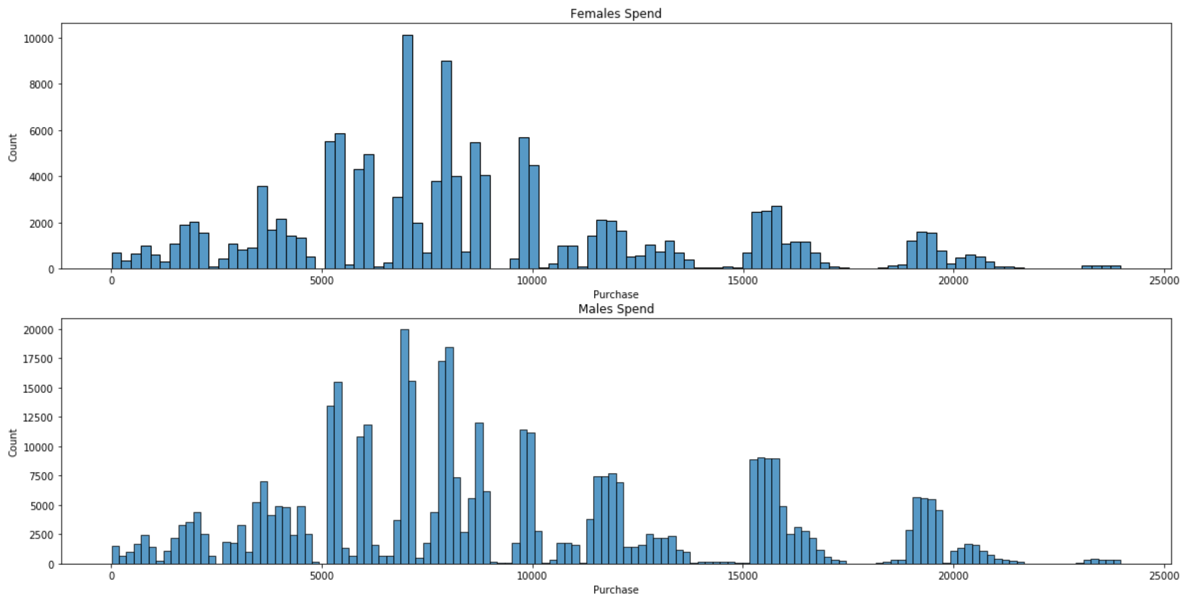
Although the the female population given to us is much lesser compared to males, its interesting to note that the average purchases of females are not very much less compared to males. This can mean that females are spending somewhat the same way as men.

In [29]:

```
#Graphical analysis of the male and female spendings for population
fig, axis = plt.subplots(nrows=2, ncols=1, figsize=(20,10))

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

Out[29]:

```
Text(0.5, 1.0, 'Males Spend')
```



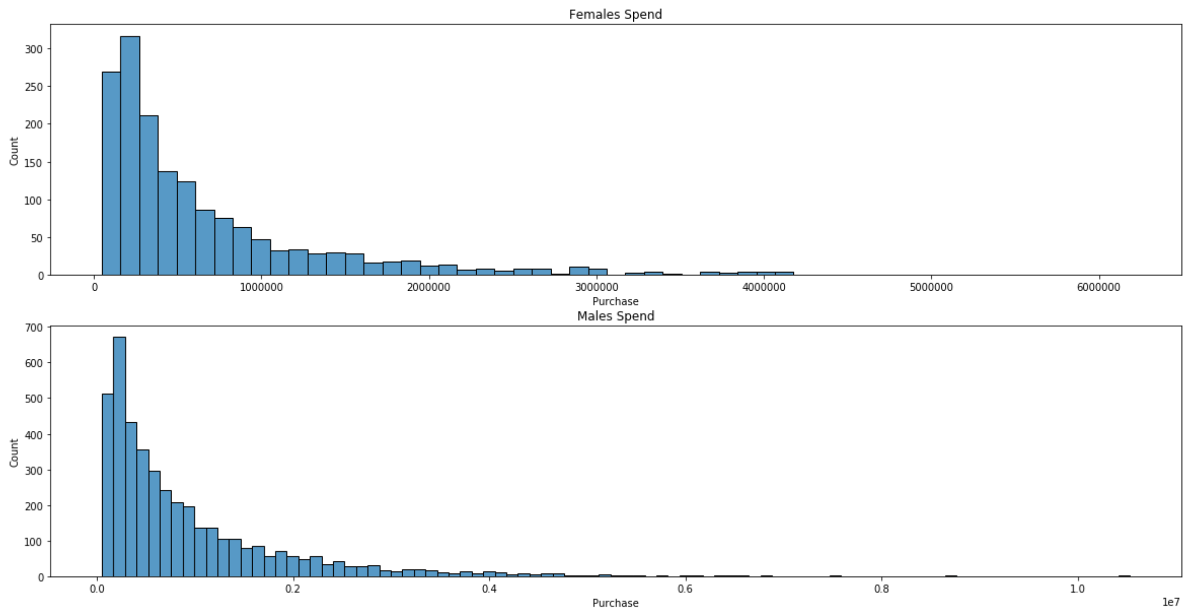
The distribution for both genders are similar with the spending between 5000-1000 being the highest for both Males and Females. Also we can note that the average purchase done by Males is twice as that of females

In [30]:

```
#Graphical analysis of the male and female spendings for population per User
fig, axis = plt.subplots(nrows=2, ncols=1, figsize=(20,10))

sns.histplot(data=total_purchase_gender[total_purchase_gender['Gender']=='F']['Purchase'],
sns.histplot(data=total_purchase_gender[total_purchase_gender['Gender']=='M']['Purchase'],
```

Out[30]:

```
Text(0.5, 1.0, 'Males Spend')
```



The average purchase done by Males are much more higher per user than Females

In [31]:

```
total_male = total_purchase_gender[total_purchase_gender['Gender']=='M']
total_female = total_purchase_gender[total_purchase_gender['Gender']=='F']
```

In [32]:

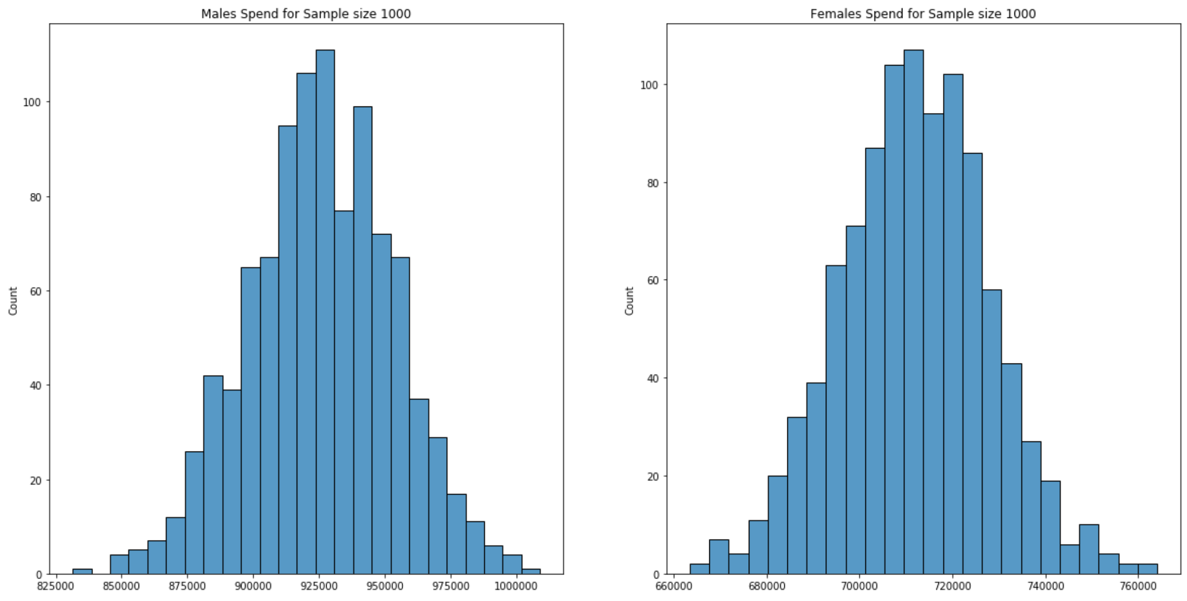
```
#Sampling to check how we can draw inferences based on our samples w.r.t population. Sample
num_samples = 1000
rep = 1000

collect_sample_means_male = []
collect_sample_means_female = []

for person in range(rep):
    sample_mean_male = total_male['Purchase'].sample(num_samples).mean()
    collect_sample_means_male.append(sample_mean_male)
    sample_mean_female = total_female['Purchase'].sample(num_samples).mean()
    collect_sample_means_female.append(sample_mean_female)
```

In [33]:

```
#Graphical representation of the male and female spendings/purchase for sample
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20,10))

sns.histplot(data=collect_sample_means_male, ax=axis[0]).set_title("Males Spend for Sample
sns.histplot(data=collect_sample_means_female, ax=axis[1]).set_title("Females Spend for Sam
```

Out[33]:

```
Text(0.5, 1.0, 'Females Spend for Sample size 1000')
```



In [34]:

```
#Now let's calculate 90% confidence interval for a sample size of 1000 to analyze how the a
z_90=1.645 #90% Confidence Interval

print("Population mean purchase amount for Males:",total_male['Purchase'].mean())
print("Population mean purchase amount for Females:",total_female['Purchase'].mean())
print("Sample mean purchase amount for Males:",np.mean(collect_sample_means_male))
print("Sample mean purchase amount for Females:",np.mean(collect_sample_means_male))
print("Population standard deviation for Males:",np.std(total_male['Purchase']))
print("Population standard deviation for Females:",np.std(total_female['Purchase']))
print("Sample std error for Male:",np.std(total_male['Purchase'])/np.sqrt(1000))
print("Sample std error for Female:",np.std(total_female['Purchase'])/np.sqrt(1000))

sample_mean_male=np.mean(collect_sample_means_male)
sample_mean_female=np.mean(collect_sample_means_female)

population_std_male=np.std(total_male['Purchase'])
population_std_female=np.std(total_female['Purchase'])

sample_std_error_male=population_std_male/np.sqrt(1000)
sample_std_error_female=population_std_female/np.sqrt(1000)

Upper_Limit_male=z_90*sample_std_error_male + sample_mean_male
Lower_Limit_male=sample_mean_male - z_90*sample_std_error_male

Upper_Limit_female=z_90*sample_std_error_female + sample_mean_female
Lower_Limit_female=sample_mean_female - z_90*sample_std_error_female

print("CI for males: ",[Lower_Limit_male,Upper_Limit_male])
print("CI for females: ",[Lower_Limit_female,Upper_Limit_female])
```

```
Population mean purchase amount for Males: 925344.4023668639
Population mean purchase amount for Females: 712024.3949579832
Sample mean purchase amount for Males: 925679.314421
Sample mean purchase amount for Females: 925679.314421
Population standard deviation for Males: 985713.4276071227
Population standard deviation for Females: 807128.3816336752
Sample std error for Male: 31170.995514500053
Sample std error for Female: 25523.640501280293
CI for males:  [874403.0267996474, 976955.6020423525]
CI for females:  [669757.3380943941, 753730.1153436062]
```

In [35]:

```
#Now let's calculate 95% confidence interval for a sample size of 1000 to analyze how the a
z_95=1.960 #95% Confidence Interval

print("Population mean purchase amount for Males:",total_male['Purchase'].mean())
print("Population mean purchase amount for Females:",total_female['Purchase'].mean())
print("Sample mean purchase amount for Males:",np.mean(collect_sample_means_male))
print("Sample mean purchase amount for Females:",np.mean(collect_sample_means_male))
print("Population standard deviation for Males:",np.std(total_male['Purchase']))
print("Population standard deviation for Females:",np.std(total_female['Purchase']))
print("Sample std error for Male:",np.std(total_male['Purchase'])/np.sqrt(1000))
print("Sample std error for Female:",np.std(total_female['Purchase'])/np.sqrt(1000))

sample_mean_male=np.mean(collect_sample_means_male)
sample_mean_female=np.mean(collect_sample_means_female)

population_std_male=np.std(total_male['Purchase'])
population_std_female=np.std(total_female['Purchase'])

sample_std_error_male=population_std_male/np.sqrt(1000)
sample_std_error_female=population_std_female/np.sqrt(1000)

Upper_Limit_male=z_95*sample_std_error_male + sample_mean_male
Lower_Limit_male=sample_mean_male - z_95*sample_std_error_male

Upper_Limit_female=z_95*sample_std_error_female + sample_mean_female
Lower_Limit_female=sample_mean_female - z_95*sample_std_error_female

print("CI for males: ",[Lower_Limit_male,Upper_Limit_male])
print("CI for females: ",[Lower_Limit_female,Upper_Limit_female])
```

```
Population mean purchase amount for Males: 925344.4023668639
Population mean purchase amount for Females: 712024.3949579832
Sample mean purchase amount for Males: 925679.314421
Sample mean purchase amount for Females: 925679.314421
Population standard deviation for Males: 985713.4276071227
Population standard deviation for Females: 807128.3816336752
Sample std error for Male: 31170.995514500053
Sample std error for Female: 25523.640501280293
CI for males:  [864584.1632125799, 986774.46562942]
CI for females:  [661717.3913364907, 761770.0621015095]
```

In [36]:

```
#Now let's calculate 99% confidence interval for a sample size of 1000 to analyze how the a
z_99=2.576 #99% Confidence Interval

print("Population mean purchase amount for Males:",total_male['Purchase'].mean())
print("Population mean purchase amount for Females:",total_female['Purchase'].mean())
print("Sample mean purchase amount for Males:",np.mean(collect_sample_means_male))
print("Sample mean purchase amount for Females:",np.mean(collect_sample_means_male))
print("Population standard deviation for Males:",np.std(total_male['Purchase']))
print("Population standard deviation for Females:",np.std(total_female['Purchase']))
print("Sample std error for Male:",np.std(total_male['Purchase'])/np.sqrt(1000))
print("Sample std error for Female:",np.std(total_female['Purchase'])/np.sqrt(1000))

sample_mean_male=np.mean(collect_sample_means_male)
sample_mean_female=np.mean(collect_sample_means_female)

population_std_male=np.std(total_male['Purchase'])
population_std_female=np.std(total_female['Purchase'])

sample_std_error_male=population_std_male/np.sqrt(1000)
sample_std_error_female=population_std_female/np.sqrt(1000)

Upper_Limit_male=z_99*sample_std_error_male + sample_mean_male
Lower_Limit_male=sample_mean_male - z_99*sample_std_error_male

Upper_Limit_female=z_99*sample_std_error_female + sample_mean_female
Lower_Limit_female=sample_mean_female - z_99*sample_std_error_female

print("CI for males: ",[Lower_Limit_male,Upper_Limit_male])
print("CI for females: ",[Lower_Limit_female,Upper_Limit_female])
```

```
Population mean purchase amount for Males: 925344.4023668639
Population mean purchase amount for Females: 712024.3949579832
Sample mean purchase amount for Males: 925679.314421
Sample mean purchase amount for Females: 925679.314421
Population standard deviation for Males: 985713.4276071227
Population standard deviation for Females: 807128.3816336752
Sample std error for Male: 31170.995514500053
Sample std error for Female: 25523.640501280293
CI for males:  [845382.8299756479, 1005975.7988663521]
CI for females:  [645994.8287877021, 777492.6246502982]
```

Further we are going to analyze the different confidence intervals for a slightly bigger sample size

In [37]:

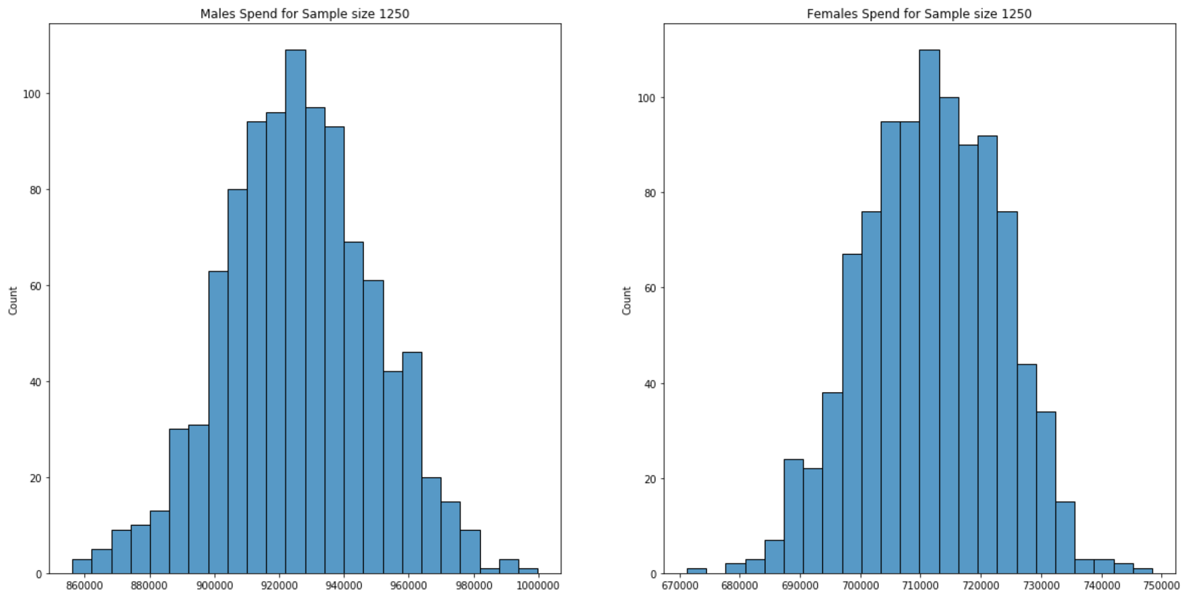
```
#Sampling to check how we can draw inferences based on our samples w.r.t population. Sample
num_samples = 1250
rep = 1000

collect_sample_means_male = []
collect_sample_means_female = []

for person in range(rep):
    sample_mean_male = total_male['Purchase'].sample(num_samples).mean()
    collect_sample_means_male.append(sample_mean_male)
    sample_mean_female = total_female['Purchase'].sample(num_samples).mean()
    collect_sample_means_female.append(sample_mean_female)
```

In [38]:

```
#Graphical representation of the male and female spendings/purchase for sample
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20,10))

sns.histplot(data=collect_sample_means_male, ax=axis[0]).set_title("Males Spend for Sample
sns.histplot(data=collect_sample_means_female, ax=axis[1]).set_title("Females Spend for Sam
```

Out[38]:

```
Text(0.5, 1.0, 'Females Spend for Sample size 1250')
```



In [39]:

```
#Now let's calculate 90% confidence interval for a sample size of 1250 to analyze how the a
z_90=1.645 #90% Confidence Interval

print("Population mean purchase amount for Males:",total_male['Purchase'].mean())
print("Population mean purchase amount for Females:",total_female['Purchase'].mean())
print("Sample mean purchase amount for Males:",np.mean(collect_sample_means_male))
print("Sample mean purchase amount for Females:",np.mean(collect_sample_means_male))
print("Population standard deviation for Males:",np.std(total_male['Purchase']))
print("Population standard deviation for Females:",np.std(total_female['Purchase']))
print("Sample std error for Male:",np.std(total_male['Purchase'])/np.sqrt(1250))
print("Sample std error for Female:",np.std(total_female['Purchase'])/np.sqrt(1250))

sample_mean_male=np.mean(collect_sample_means_male)
sample_mean_female=np.mean(collect_sample_means_female)

population_std_male=np.std(total_male['Purchase'])
population_std_female=np.std(total_female['Purchase'])

sample_std_error_male=population_std_male/np.sqrt(1250)
sample_std_error_female=population_std_female/np.sqrt(1250)

Upper_Limit_male=z_90*sample_std_error_male + sample_mean_male
Lower_Limit_male=sample_mean_male - z_90*sample_std_error_male

Upper_Limit_female=z_90*sample_std_error_female + sample_mean_female
Lower_Limit_female=sample_mean_female - z_90*sample_std_error_female

print("CI for males: ",[Lower_Limit_male,Upper_Limit_male])
print("CI for females: ",[Lower_Limit_female,Upper_Limit_female])
```

```
Population mean purchase amount for Males: 925344.4023668639
Population mean purchase amount for Females: 712024.3949579832
Sample mean purchase amount for Males: 926021.8527751999
Sample mean purchase amount for Females: 926021.8527751999
Population standard deviation for Males: 985713.4276071227
Population standard deviation for Females: 807128.3816336752
Sample std error for Male: 27880.185958705257
Sample std error for Female: 22829.038077651814
CI for males:  [880158.9468731298, 971884.75867727]
CI for females:  [674200.3922862628, 749307.9275617372]
```

In [40]:

```
#Now let's calculate 95% confidence interval for a sample size of 1250 to analyze how the a
z_95=1.960 #95% Confidence Interval

print("Population mean purchase amount for Males:",total_male['Purchase'].mean())
print("Population mean purchase amount for Females:",total_female['Purchase'].mean())
print("Sample mean purchase amount for Males:",np.mean(collect_sample_means_male))
print("Sample mean purchase amount for Females:",np.mean(collect_sample_means_male))
print("Population standard deviation for Males:",np.std(total_male['Purchase']))
print("Population standard deviation for Females:",np.std(total_female['Purchase']))
print("Sample std error for Male:",np.std(total_male['Purchase'])/np.sqrt(1250))
print("Sample std error for Female:",np.std(total_female['Purchase'])/np.sqrt(1250))

sample_mean_male=np.mean(collect_sample_means_male)
sample_mean_female=np.mean(collect_sample_means_female)

population_std_male=np.std(total_male['Purchase'])
population_std_female=np.std(total_female['Purchase'])

sample_std_error_male=population_std_male/np.sqrt(1250)
sample_std_error_female=population_std_female/np.sqrt(1250)

Upper_Limit_male=z_95*sample_std_error_male + sample_mean_male
Lower_Limit_male=sample_mean_male - z_95*sample_std_error_male

Upper_Limit_female=z_95*sample_std_error_female + sample_mean_female
Lower_Limit_female=sample_mean_female - z_95*sample_std_error_female

print("CI for males: ",[Lower_Limit_male,Upper_Limit_male])
print("CI for females: ",[Lower_Limit_female,Upper_Limit_female])
```

```
Population mean purchase amount for Males: 925344.4023668639
Population mean purchase amount for Females: 712024.3949579832
Sample mean purchase amount for Males: 926021.8527751999
Sample mean purchase amount for Females: 926021.8527751999
Population standard deviation for Males: 985713.4276071227
Population standard deviation for Females: 807128.3816336752
Sample std error for Male: 27880.185958705257
Sample std error for Female: 22829.038077651814
CI for males:  [871376.6882961376, 980667.0172542622]
CI for females:  [667009.2452918025, 756499.0745561976]
```

In [41]:

```
#Now let's calculate 99% confidence interval for a sample size of 1250 to analyze how the a
z_99=2.576 #99% Confidence Interval

print("Population mean purchase amount for Males:",total_male['Purchase'].mean())
print("Population mean purchase amount for Females:",total_female['Purchase'].mean())
print("Sample mean purchase amount for Males:",np.mean(collect_sample_means_male))
print("Sample mean purchase amount for Females:",np.mean(collect_sample_means_male))
print("Population standard deviation for Males:",np.std(total_male['Purchase']))
print("Population standard deviation for Females:",np.std(total_female['Purchase']))
print("Sample std error for Male:",np.std(total_male['Purchase'])/np.sqrt(1250))
print("Sample std error for Female:",np.std(total_female['Purchase'])/np.sqrt(1250))

sample_mean_male=np.mean(collect_sample_means_male)
sample_mean_female=np.mean(collect_sample_means_female)

population_std_male=np.std(total_male['Purchase'])
population_std_female=np.std(total_female['Purchase'])

sample_std_error_male=population_std_male/np.sqrt(1250)
sample_std_error_female=population_std_female/np.sqrt(1250)

Upper_Limit_male=z_99*sample_std_error_male + sample_mean_male
Lower_Limit_male=sample_mean_male - z_99*sample_std_error_male

Upper_Limit_female=z_99*sample_std_error_female + sample_mean_female
Lower_Limit_female=sample_mean_female - z_99*sample_std_error_female

print("CI for males: ",[Lower_Limit_male,Upper_Limit_male])
print("CI for females: ",[Lower_Limit_female,Upper_Limit_female])
```

```
Population mean purchase amount for Males: 925344.4023668639
Population mean purchase amount for Females: 712024.3949579832
Sample mean purchase amount for Males: 926021.8527751999
Sample mean purchase amount for Females: 926021.8527751999
Population standard deviation for Males: 985713.4276071227
Population standard deviation for Females: 807128.3816336752
Sample std error for Male: 27880.185958705257
Sample std error for Female: 22829.038077651814
CI for males:  [854202.4937455752, 997841.2118048246]
CI for females:  [652946.5578359689, 770561.7620120312]
```

We need to further analyze how the Marital status affects the purchase ability and draw inference based on that

In [42]:

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

In [43]:

```
total_unmarried = total_purchase_marital[total_purchase_marital['Marital_Status']==0]
total_married = total_purchase_marital[total_purchase_marital['Marital_Status']==1]
```

In [44]:

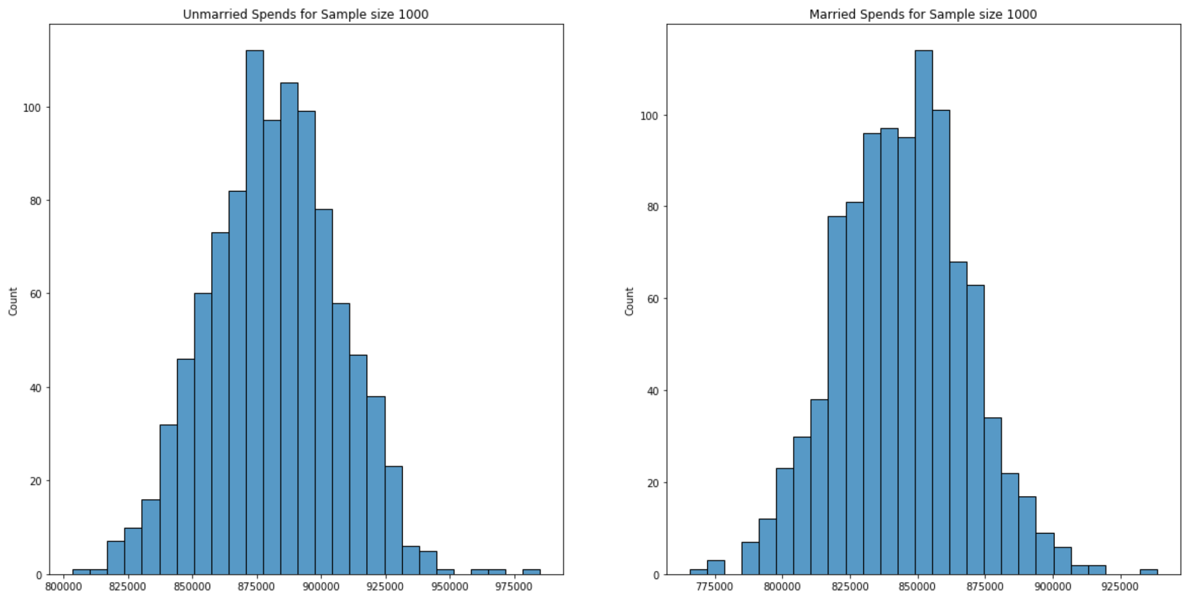
```
#Sampling to check how we can draw inferences based on our samples w.r.t population. Sample
num_samples = 1000
rep = 1000

collect_sample_means_unmarried = []
collect_sample_means_married = []

for person in range(rep):
    sample_mean_unmarried = total_unmarried['Purchase'].sample(num_samples).mean()
    collect_sample_means_unmarried.append(sample_mean_unmarried)
    sample_mean_married = total_married['Purchase'].sample(num_samples).mean()
    collect_sample_means_married.append(sample_mean_married)
```

In [45]:

```
#Graphical representation of the married and unmarried customers spendings/purchase for sam
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20,10))

sns.histplot(data=collect_sample_means_unmarried, ax=axis[0]).set_title("Unmarried Spends f
sns.histplot(data=collect_sample_means_married, ax=axis[1]).set_title("Married Spends for S
```

Out[45]:

```
Text(0.5, 1.0, 'Married Spends for Sample size 1000')
```



In [46]:

```
#Now let's calculate 90% confidence interval for a sample size of 1000 to analyze how the a
z_90=1.645 #90% Confidence Interval

print("Population mean purchase amount for Unmarried customers:",total_unmarried['Purchase'
print("Population mean purchase amount for Married customers:",total_married['Purchase'].me
print("Sample mean purchase amount for Unmarried customers:",np.mean(collect_sample_means_u
print("Sample mean purchase amount for Married customers:",np.mean(collect_sample_means_mar
print("Population standard deviation for Unmarried customers:",np.std(total_unmarried['Purc
print("Population standard deviation for Married customers:",np.std(total_married['Purchase
print("Sample std error for Male:",np.std(total_male['Purchase'])/np.sqrt(1000))
print("Sample std error for Female:",np.std(total_female['Purchase'])/np.sqrt(1000))

sample_mean_unmarried=np.mean(collect_sample_means_unmarried)
sample_mean_married=np.mean(collect_sample_means_married)

population_std_unmarried=np.std(total_unmarried['Purchase'])
population_std_married=np.std(total_married['Purchase'])

sample_std_error_unmarried=population_std_unmarried/np.sqrt(1000)
sample_std_error_married=population_std_married/np.sqrt(1000)

Upper_Limit_unmarried=z_90*sample_std_error_unmarried + sample_mean_unmarried
Lower_Limit_unmarried=sample_mean_unmarried - z_90*sample_std_error_unmarried

Upper_Limit_married=z_90*sample_std_error_married + sample_mean_married
Lower_Limit_married=sample_mean_married - z_90*sample_std_error_married

print("CI for Unmarried customers: ",[Lower_Limit_unmarried,Upper_Limit_unmarried])
print("CI for Married customers: ",[Lower_Limit_married,Upper_Limit_married])
```

```
Population mean purchase amount for Unmarried customers: 880575.7819724905
Population mean purchase amount for Married customers: 843526.7966855295
Sample mean purchase amount for Unmarried customers: 881542.575107
Sample mean purchase amount for Married customers: 844521.386365
Population standard deviation for Unmarried customers: 949297.3110530594
Population standard deviation for Married customers: 935163.0603173219
Sample std error for Male: 31170.995514500053
Sample std error for Female: 25523.640501280293
CI for Unmarried customers:  [832160.6344775634, 930924.5157364365]
CI for Married customers:  [795874.7019324861, 893168.0707975139]
```

In [47]:

```
#Now let's calculate 95% confidence interval for a sample size of 1000 to analyze how the a
z_95=1.960 #95% Confidence Interval

print("Population mean purchase amount for Unmarried customers:",total_unmarried['Purchase'
print("Population mean purchase amount for Married customers:",total_married['Purchase'].me
print("Sample mean purchase amount for Unmarried customers:",np.mean(collect_sample_means_u
print("Sample mean purchase amount for Married customers:",np.mean(collect_sample_means_mar
print("Population standard deviation for Unmarried customers:",np.std(total_unmarried['Purc
print("Population standard deviation for Married customers:",np.std(total_married['Purchase
print("Sample std error for Male:",np.std(total_male['Purchase'])/np.sqrt(1000))
print("Sample std error for Female:",np.std(total_female['Purchase'])/np.sqrt(1000))

sample_mean_unmarried=np.mean(collect_sample_means_unmarried)
sample_mean_married=np.mean(collect_sample_means_married)

population_std_unmarried=np.std(total_unmarried['Purchase'])
population_std_married=np.std(total_married['Purchase'])

sample_std_error_unmarried=population_std_unmarried/np.sqrt(1000)
sample_std_error_married=population_std_married/np.sqrt(1000)

Upper_Limit_unmarried=z_95*sample_std_error_unmarried + sample_mean_unmarried
Lower_Limit_unmarried=sample_mean_unmarried - z_95*sample_std_error_unmarried

Upper_Limit_married=z_95*sample_std_error_married + sample_mean_married
Lower_Limit_married=sample_mean_married - z_95*sample_std_error_married

print("CI for Unmarried customers: ",[Lower_Limit_unmarried,Upper_Limit_unmarried])
print("CI for Married customers: ",[Lower_Limit_married,Upper_Limit_married])
```

```
Population mean purchase amount for Unmarried customers: 880575.7819724905
Population mean purchase amount for Married customers: 843526.7966855295
Sample mean purchase amount for Unmarried customers: 881542.575107
Sample mean purchase amount for Married customers: 844521.386365
Population standard deviation for Unmarried customers: 949297.3110530594
Population standard deviation for Married customers: 935163.0603173219
Sample std error for Male: 31170.995514500053
Sample std error for Female: 25523.640501280293
CI for Unmarried customers:  [822704.5181868203, 940380.6320271796]
CI for Married customers:  [786559.3793815792, 902483.3933484209]
```

In [48]:

```
#Now let's calculate 99% confidence interval for a sample size of 1000 to analyze how the c
z_99=2.576 #99% Confidence Interval

print("Population mean purchase amount for Unmarried customers:",total_unmarried['Purchase'
print("Population mean purchase amount for Married customers:",total_married['Purchase'].me
print("Sample mean purchase amount for Unmarried customers:",np.mean(collect_sample_means_u
print("Sample mean purchase amount for Married customers:",np.mean(collect_sample_means_mar
print("Population standard deviation for Unmarried customers:",np.std(total_unmarried['Purc
print("Population standard deviation for Married customers:",np.std(total_married['Purchase
print("Sample std error for Male:",np.std(total_male['Purchase'])/np.sqrt(1000))
print("Sample std error for Female:",np.std(total_female['Purchase'])/np.sqrt(1000))

sample_mean_unmarried=np.mean(collect_sample_means_unmarried)
sample_mean_married=np.mean(collect_sample_means_married)

population_std_unmarried=np.std(total_unmarried['Purchase'])
population_std_married=np.std(total_married['Purchase'])

sample_std_error_unmarried=population_std_unmarried/np.sqrt(1000)
sample_std_error_married=population_std_married/np.sqrt(1000)

Upper_Limit_unmarried=z_99*sample_std_error_unmarried + sample_mean_unmarried
Lower_Limit_unmarried=sample_mean_unmarried - z_99*sample_std_error_unmarried

Upper_Limit_married=z_99*sample_std_error_married + sample_mean_married
Lower_Limit_married=sample_mean_married - z_99*sample_std_error_married

print("CI for Unmarried customers: ",[Lower_Limit_unmarried,Upper_Limit_unmarried])
print("CI for Married customers: ",[Lower_Limit_married,Upper_Limit_married])
```

```
Population mean purchase amount for Unmarried customers: 880575.7819724905
Population mean purchase amount for Married customers: 843526.7966855295
Sample mean purchase amount for Unmarried customers: 881542.575107
Sample mean purchase amount for Married customers: 844521.386365
Population standard deviation for Unmarried customers: 949297.3110530594
Population standard deviation for Married customers: 935163.0603173219
Sample std error for Male: 31170.995514500053
Sample std error for Female: 25523.640501280293
CI for Unmarried customers:  [804212.5574404781, 958872.5927735218]
CI for Married customers:  [768342.7486153613, 920700.0241146388]
```

In [49]:

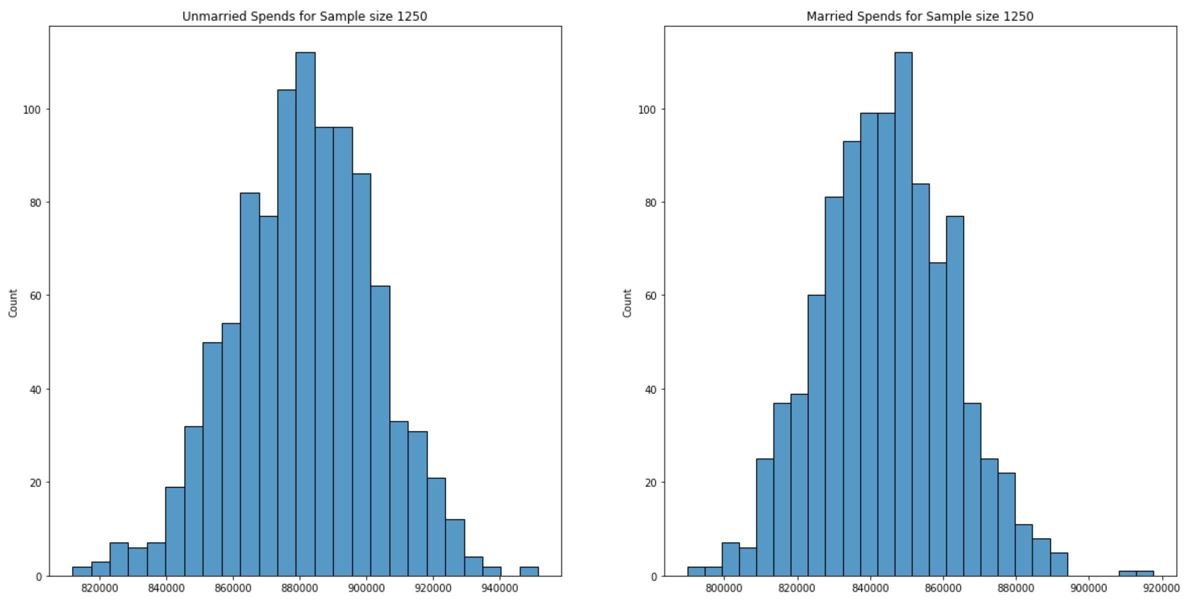
```
#Sampling to check how we can draw inferences based on our samples w.r.t population. Sample
num_samples = 1250
rep = 1000

collect_sample_means_unmarried = []
collect_sample_means_married = []

for person in range(rep):
    sample_mean_unmarried = total_unmarried['Purchase'].sample(num_samples).mean()
    collect_sample_means_unmarried.append(sample_mean_unmarried)
    sample_mean_married = total_married['Purchase'].sample(num_samples).mean()
    collect_sample_means_married.append(sample_mean_married)
```

In [50]:

```
#Graphical representation of the married and unmarried customers spendings/purchase for sam
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20,10))

sns.histplot(data=collect_sample_means_unmarried, ax=axis[0]).set_title("Unmarried Spends f
sns.histplot(data=collect_sample_means_married, ax=axis[1]).set_title("Married Spends for S
```

Out[50]:

```
Text(0.5, 1.0, 'Married Spends for Sample size 1250')
```



In [51]:

```
#Now let's calculate 90% confidence interval for a sample size of 1250 to analyze how the a
z_90=1.645 #90% Confidence Interval

print("Population mean purchase amount for Unmarried customers:",total_unmarried['Purchase'
print("Population mean purchase amount for Married customers:",total_married['Purchase'].me
print("Sample mean purchase amount for Unmarried customers:",np.mean(collect_sample_means_u
print("Sample mean purchase amount for Married customers:",np.mean(collect_sample_means_mar
print("Population standard deviation for Unmarried customers:",np.std(total_unmarried['Purc
print("Population standard deviation for Married customers:",np.std(total_married['Purchase
print("Sample std error for Male:",np.std(total_male['Purchase'])/np.sqrt(1250))
print("Sample std error for Female:",np.std(total_female['Purchase'])/np.sqrt(1250))

sample_mean_unmarried=np.mean(collect_sample_means_unmarried)
sample_mean_married=np.mean(collect_sample_means_married)

population_std_unmarried=np.std(total_unmarried['Purchase'])
population_std_married=np.std(total_married['Purchase'])

sample_std_error_unmarried=population_std_unmarried/np.sqrt(1250)
sample_std_error_married=population_std_married/np.sqrt(1250)

Upper_Limit_unmarried=z_90*sample_std_error_unmarried + sample_mean_unmarried
Lower_Limit_unmarried=sample_mean_unmarried - z_90*sample_std_error_unmarried

Upper_Limit_married=z_90*sample_std_error_married + sample_mean_married
Lower_Limit_married=sample_mean_married - z_90*sample_std_error_married

print("CI for Unmarried customers: ",[Lower_Limit_unmarried,Upper_Limit_unmarried])
print("CI for Married customers: ",[Lower_Limit_married,Upper_Limit_married])
```

```
Population mean purchase amount for Unmarried customers: 880575.7819724905
Population mean purchase amount for Married customers: 843526.7966855295
Sample mean purchase amount for Unmarried customers: 881438.0545504
Sample mean purchase amount for Married customers: 844307.5135736
Population standard deviation for Unmarried customers: 949297.3110530594
Population standard deviation for Married customers: 935163.0603173219
Sample std error for Male: 27880.185958705257
Sample std error for Female: 22829.038077651814
CI for Unmarried customers:  [837269.5041070885, 925606.6049937115]
CI for Married customers:  [800796.5962651672, 887818.4308820327]
```

In [52]:

```
#Now let's calculate 95% confidence interval for a sample size of 1000 to analyze how the a
z_95=1.960 #95% Confidence Interval

print("Population mean purchase amount for Unmarried customers:",total_unmarried['Purchase'
print("Population mean purchase amount for Married customers:",total_married['Purchase'].me
print("Sample mean purchase amount for Unmarried customers:",np.mean(collect_sample_means_u
print("Sample mean purchase amount for Married customers:",np.mean(collect_sample_means_mar
print("Population standard deviation for Unmarried customers:",np.std(total_unmarried['Purc
print("Population standard deviation for Married customers:",np.std(total_married['Purchase
print("Sample std error for Male:",np.std(total_male['Purchase'])/np.sqrt(1250))
print("Sample std error for Female:",np.std(total_female['Purchase'])/np.sqrt(1250))

sample_mean_unmarried=np.mean(collect_sample_means_unmarried)
sample_mean_married=np.mean(collect_sample_means_married)

population_std_unmarried=np.std(total_unmarried['Purchase'])
population_std_married=np.std(total_married['Purchase'])

sample_std_error_unmarried=population_std_unmarried/np.sqrt(1250)
sample_std_error_married=population_std_married/np.sqrt(1250)

Upper_Limit_unmarried=z_95*sample_std_error_unmarried + sample_mean_unmarried
Lower_Limit_unmarried=sample_mean_unmarried - z_95*sample_std_error_unmarried

Upper_Limit_married=z_95*sample_std_error_married + sample_mean_married
Lower_Limit_married=sample_mean_married - z_95*sample_std_error_married

print("CI for Unmarried customers: ",[Lower_Limit_unmarried,Upper_Limit_unmarried])
print("CI for Married customers: ",[Lower_Limit_married,Upper_Limit_married])
```

```
Population mean purchase amount for Unmarried customers: 880575.7819724905
Population mean purchase amount for Married customers: 843526.7966855295
Sample mean purchase amount for Unmarried customers: 881438.0545504
Sample mean purchase amount for Married customers: 844307.5135736
Population standard deviation for Unmarried customers: 949297.3110530594
Population standard deviation for Married customers: 935163.0603173219
Sample std error for Male: 27880.185958705257
Sample std error for Female: 22829.038077651814
CI for Unmarried customers:  [828811.6965753905, 934064.4125254095]
CI for Married customers:  [792464.7184827015, 896150.3086644985]
```

In [53]:

```
#Now let's calculate 99% confidence interval for a sample size of 1250 to analyze how the a
z_99=2.576 #99% Confidence Interval

print("Population mean purchase amount for Unmarried customers:",total_unmarried['Purchase'
print("Population mean purchase amount for Married customers:",total_married['Purchase'].me
print("Sample mean purchase amount for Unmarried customers:",np.mean(collect_sample_means_u
print("Sample mean purchase amount for Married customers:",np.mean(collect_sample_means_mar
print("Population standard deviation for Unmarried customers:",np.std(total_unmarried['Purc
print("Population standard deviation for Married customers:",np.std(total_married['Purchase
print("Sample std error for Male:",np.std(total_male['Purchase'])/np.sqrt(1250))
print("Sample std error for Female:",np.std(total_female['Purchase'])/np.sqrt(1250))

sample_mean_unmarried=np.mean(collect_sample_means_unmarried)
sample_mean_married=np.mean(collect_sample_means_married)

population_std_unmarried=np.std(total_unmarried['Purchase'])
population_std_married=np.std(total_married['Purchase'])

sample_std_error_unmarried=population_std_unmarried/np.sqrt(1250)
sample_std_error_married=population_std_married/np.sqrt(1250)

Upper_Limit_unmarried=z_99*sample_std_error_unmarried + sample_mean_unmarried
Lower_Limit_unmarried=sample_mean_unmarried - z_99*sample_std_error_unmarried

Upper_Limit_married=z_99*sample_std_error_married + sample_mean_married
Lower_Limit_married=sample_mean_married - z_99*sample_std_error_married

print("CI for Unmarried customers: ",[Lower_Limit_unmarried,Upper_Limit_unmarried])
print("CI for Married customers: ",[Lower_Limit_married,Upper_Limit_married])
```

```
Population mean purchase amount for Unmarried customers: 880575.7819724905
Population mean purchase amount for Married customers: 843526.7966855295
Sample mean purchase amount for Unmarried customers: 881438.0545504
Sample mean purchase amount for Married customers: 844307.5135736
Population standard deviation for Unmarried customers: 949297.3110530594
Population standard deviation for Married customers: 935163.0603173219
Sample std error for Male: 27880.185958705257
Sample std error for Female: 22829.038077651814
CI for Unmarried customers:  [812271.984068959, 950604.125031841]
CI for Married customers:  [776171.2685969905, 912443.7585502095]
```

We can further analyze how the the different age groups affects the purchase ability and draw inference based on that

In [54]:

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

In [55]:

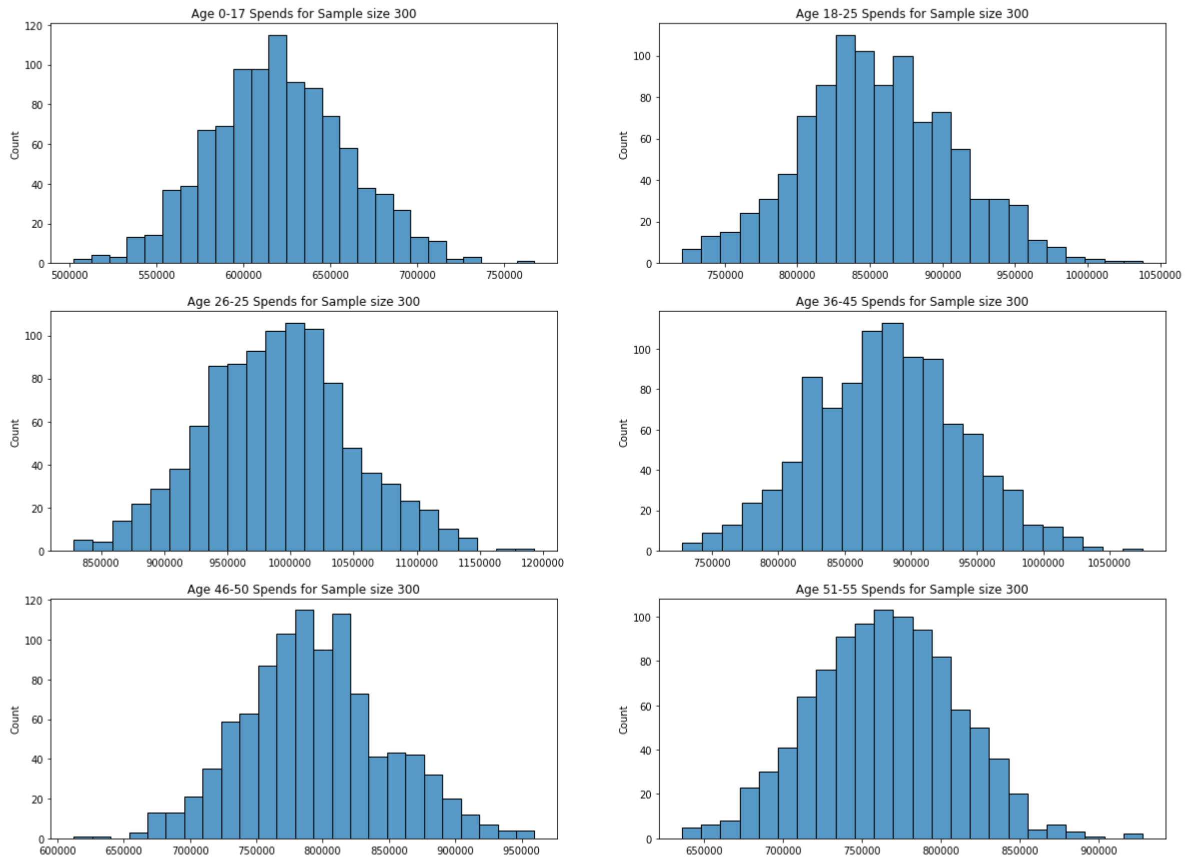
```
#Sampling to check how we can draw inferences based on our samples w.r.t population. Sample
num_samples = 300 #taking a smaller sample size since the age groups bins are more
rep = 1000

collect_all_sample_means_age = {}
ages = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

for age_bin in ages:
    collect_all_sample_means_age[age_bin] = []

for i in ages:
    for person in range(rep):

        sample_mean = total_purchase_age[total_purchase_age['Age']==i].sample(num_samples,
        collect_all_sample_means_age[i].append(sample_mean)
```

In [56]:

```
#Graphical representation of the different age groups spendings/purchase for sample

fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(20, 15))
sns.histplot(collect_all_sample_means_age['0-17'],ax=axis[0,0]).set_title("Age 0-17 Spends
sns.histplot(collect_all_sample_means_age['18-25'],ax=axis[0,1]).set_title("Age 18-25 Spend
sns.histplot(collect_all_sample_means_age['26-35'],ax=axis[1,0]).set_title("Age 26-25 Spend
sns.histplot(collect_all_sample_means_age['36-45'],ax=axis[1,1]).set_title("Age 36-45 Spend
sns.histplot(collect_all_sample_means_age['46-50'],ax=axis[2,0]).set_title("Age 46-50 Spend
sns.histplot(collect_all_sample_means_age['51-55'],ax=axis[2,1]).set_title("Age 51-55 Spend

plt.figure(figsize=(10, 5))
sns.histplot(collect_all_sample_means_age['55+']).set_title("Age 55+ Spends for Sample size
```

Out[56]:

```
Text(0.5, 1.0, 'Age 55+ Spends for Sample size 300')
```



Age 55+ Spends for Sample size 300

In [57]:

```
collect_all_sample_means_age = {}
ages = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
z_90=1.645 #90% Confidence Interval

for i in ages:
    age_df = total_purchase_age[total_purchase_age['Age']==i]
    sample_mean = np.mean(age_df["Purchase"])
    population_std = np.std(age_df["Purchase"])
    sample_std_error = population_std/np.sqrt(300)

    Upper_Limit = z_90*sample_std_error + sample_mean
    Lower_Limit = sample_mean - z_90*sample_std_error
    print("Sample mean of age group {} for sample size 300 is : {:.2f}".format(i,sample_mea
    print("90% Confidence interval for age group {} for sample size 300 : ({:.2f}, {:.2f})"
```

```
Sample mean of age group 0-17 for sample size 300 is : 618867.81
90% Confidence interval for age group 0-17 for sample size 300 : (553765.05,
683970.57)
Sample mean of age group 18-25 for sample size 300 is : 854863.12
90% Confidence interval for age group 18-25 for sample size 300 : (770569.6
2, 939156.62)
Sample mean of age group 26-35 for sample size 300 is : 989659.32
90% Confidence interval for age group 26-35 for sample size 300 : (891706.9
2, 1087611.72)
Sample mean of age group 36-45 for sample size 300 is : 879665.71
90% Confidence interval for age group 36-45 for sample size 300 : (786480.9
3, 972850.49)
Sample mean of age group 46-50 for sample size 300 is : 792548.78
90% Confidence interval for age group 46-50 for sample size 300 : (704372.5
9, 880724.98)
Sample mean of age group 51-55 for sample size 300 is : 763200.92
90% Confidence interval for age group 51-55 for sample size 300 : (688029.0
8, 838372.77)
Sample mean of age group 55+ for sample size 300 is : 539697.24
90% Confidence interval for age group 55+ for sample size 300 : (481131.61,
598262.88)
```

In [58]:

```
collect_all_sample_means_age = {}
ages = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
z_95=1.960 #95% Confidence Interval

for i in ages:
    age_df = total_purchase_age[total_purchase_age['Age']==i]
    sample_mean = np.mean(age_df["Purchase"])
    population_std = np.std(age_df["Purchase"])
    sample_std_error = population_std/np.sqrt(300)

    Upper_Limit = z_95*sample_std_error + sample_mean
    Lower_Limit = sample_mean - z_95*sample_std_error
    print("Sample mean of age group {} for sample size 300 is : {:.2f}".format(i,sample_mea
    print("95% Confidence interval for age group {} for sample size 300 : ({:.2f}, {:.2f})"
```

```
Sample mean of age group 0-17 for sample size 300 is : 618867.81
95% Confidence interval for age group 0-17 for sample size 300 : (541298.56,
696437.06)
Sample mean of age group 18-25 for sample size 300 is : 854863.12
95% Confidence interval for age group 18-25 for sample size 300 : (754428.3
1, 955297.93)
Sample mean of age group 26-35 for sample size 300 is : 989659.32
95% Confidence interval for age group 26-35 for sample size 300 : (872950.0
8, 1106368.56)
Sample mean of age group 36-45 for sample size 300 is : 879665.71
95% Confidence interval for age group 36-45 for sample size 300 : (768637.0
3, 990694.39)
Sample mean of age group 46-50 for sample size 300 is : 792548.78
95% Confidence interval for age group 46-50 for sample size 300 : (687487.7
8, 897609.78)
Sample mean of age group 51-55 for sample size 300 is : 763200.92
95% Confidence interval for age group 51-55 for sample size 300 : (673634.4
7, 852767.38)
Sample mean of age group 55+ for sample size 300 is : 539697.24
95% Confidence interval for age group 55+ for sample size 300 : (469916.91,
609477.58)
```

In [59]:

```
collect_all_sample_means_age = {}
ages = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
z_99=2.576 #99% Confidence Interval

for i in ages:
    age_df = total_purchase_age[total_purchase_age['Age']==i]
    sample_mean = np.mean(age_df["Purchase"])
    population_std = np.std(age_df["Purchase"])
    sample_std_error = population_std/np.sqrt(300)

    Upper_Limit = z_99*sample_std_error + sample_mean
    Lower_Limit = sample_mean - z_99*sample_std_error
    print("Sample mean of age group {} for sample size 300 is : {:.2f}".format(i,sample_mea
    print("99% Confidence interval for age group {} for sample size 300: ({:.2f}, {:.2f})".
```

```
Sample mean of age group 0-17 for sample size 300 is : 618867.81
99% Confidence interval for age group 0-17 for sample size 300: (516919.66,
720815.97)
Sample mean of age group 18-25 for sample size 300 is : 854863.12
99% Confidence interval for age group 18-25 for sample size 300: (722863.08,
986863.16)
Sample mean of age group 26-35 for sample size 300 is : 989659.32
99% Confidence interval for age group 26-35 for sample size 300: (836270.03,
1143048.60)
Sample mean of age group 36-45 for sample size 300 is : 879665.71
99% Confidence interval for age group 36-45 for sample size 300: (733742.31,
1025589.11)
Sample mean of age group 46-50 for sample size 300 is : 792548.78
99% Confidence interval for age group 46-50 for sample size 300: (654468.61,
930628.95)
Sample mean of age group 51-55 for sample size 300 is : 763200.92
99% Confidence interval for age group 51-55 for sample size 300: (645485.01,
880916.83)
Sample mean of age group 55+ for sample size 300 is : 539697.24
99% Confidence interval for age group 55+ for sample size 300: (447985.95, 6
31408.54)
```

In [60]:

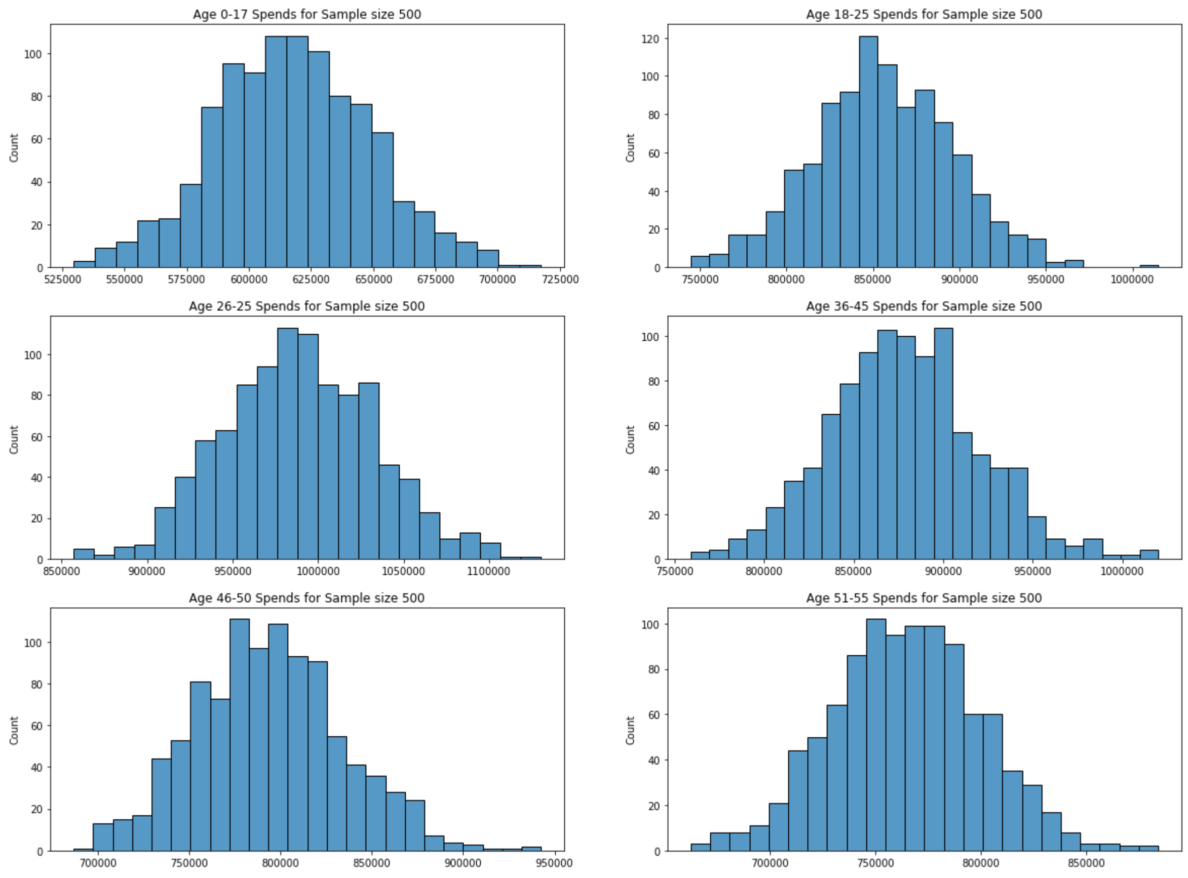
```
#Sampling to check how we can draw inferences based on our samples w.r.t population. Sample
num_samples = 500 #taking a smaller sample size since the age groups bins are more
rep = 1000

collect_all_sample_means_age = {}
ages = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

for age_bin in ages:
    collect_all_sample_means_age[age_bin] = []

for i in ages:
    for person in range(rep):

        sample_mean = total_purchase_age[total_purchase_age['Age']==i].sample(num_samples,
        collect_all_sample_means_age[i].append(sample_mean)
```

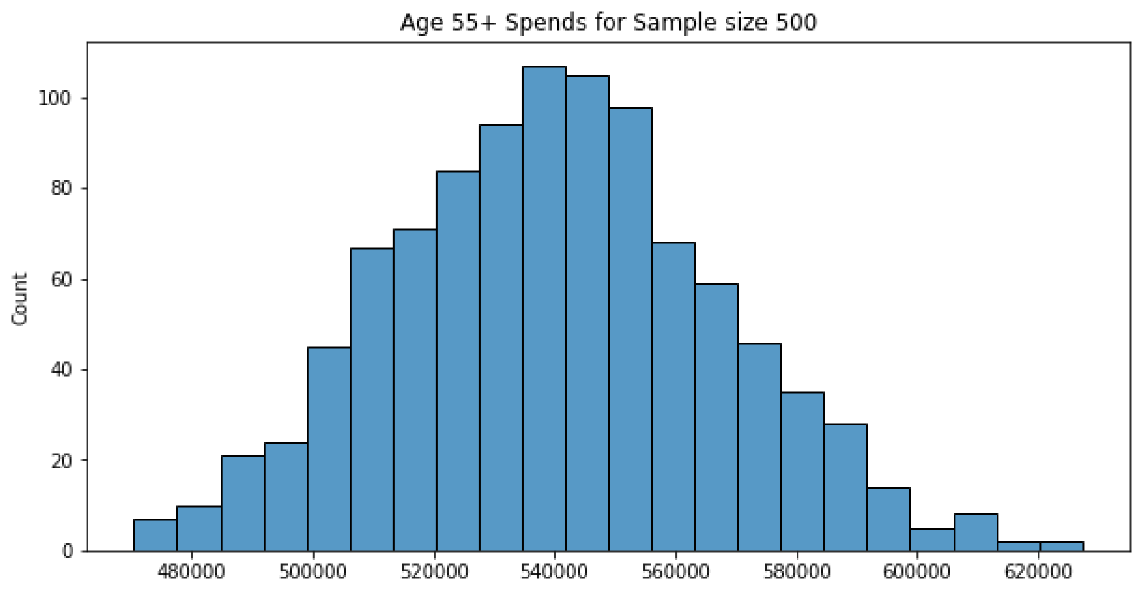
In [61]:

```
#Graphical representation of the different age groups spendings/purchase for sample

fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(20, 15))
sns.histplot(collect_all_sample_means_age['0-17'],ax=axis[0,0]).set_title("Age 0-17 Spends 
sns.histplot(collect_all_sample_means_age['18-25'],ax=axis[0,1]).set_title("Age 18-25 Spend
sns.histplot(collect_all_sample_means_age['26-35'],ax=axis[1,0]).set_title("Age 26-25 Spend
sns.histplot(collect_all_sample_means_age['36-45'],ax=axis[1,1]).set_title("Age 36-45 Spend
sns.histplot(collect_all_sample_means_age['46-50'],ax=axis[2,0]).set_title("Age 46-50 Spend
sns.histplot(collect_all_sample_means_age['51-55'],ax=axis[2,1]).set_title("Age 51-55 Spend

plt.figure(figsize=(10, 5))
sns.histplot(collect_all_sample_means_age['55+']).set_title("Age 55+ Spends for Sample size
```

Out[61]:

```
Text(0.5, 1.0, 'Age 55+ Spends for Sample size 500')
```





In [62]:

```
collect_all_sample_means_age = {}
ages = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
z_90=1.645 #90% Confidence Interval

for i in ages:
    age_df = total_purchase_age[total_purchase_age['Age']==i]
    sample_mean = np.mean(age_df["Purchase"])
    population_std = np.std(age_df["Purchase"])
    sample_std_error = population_std/np.sqrt(500)

    Upper_Limit = z_90*sample_std_error + sample_mean
    Lower_Limit = sample_mean - z_90*sample_std_error
    print("Sample mean of age group {} for sample size 500 is : {:.2f}".format(i,sample_mea
    print("90% Confidence interval for age group {} for sample size 500: ({:.2f}, {:.2f})".
```

```
Sample mean of age group 0-17 for sample size 500 is : 618867.81
90% Confidence interval for age group 0-17 for sample size 500: (568439.43,
669296.19)
Sample mean of age group 18-25 for sample size 500 is : 854863.12
90% Confidence interval for age group 18-25 for sample size 500: (789569.65,
920156.59)
Sample mean of age group 26-35 for sample size 500 is : 989659.32
90% Confidence interval for age group 26-35 for sample size 500: (913785.72,
1065532.92)
Sample mean of age group 36-45 for sample size 500 is : 879665.71
90% Confidence interval for age group 36-45 for sample size 500: (807485.09,
951846.33)
Sample mean of age group 46-50 for sample size 500 is : 792548.78
90% Confidence interval for age group 46-50 for sample size 500: (724247.79,
860849.77)
Sample mean of age group 51-55 for sample size 500 is : 763200.92
90% Confidence interval for age group 51-55 for sample size 500: (704973.06,
821428.78)
Sample mean of age group 55+ for sample size 500 is : 539697.24
90% Confidence interval for age group 55+ for sample size 500: (494332.50, 5
85061.99)
```

In [63]:

```
collect_all_sample_means_age = {}
ages = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
z_95=1.960 #95% Confidence Interval

for i in ages:
    age_df = total_purchase_age[total_purchase_age['Age']==i]
    sample_mean = np.mean(age_df["Purchase"])
    population_std = np.std(age_df["Purchase"])
    sample_std_error = population_std/np.sqrt(500)

    Upper_Limit = z_95*sample_std_error + sample_mean
    Lower_Limit = sample_mean - z_95*sample_std_error
    print("Sample mean of age group {} for sample size 500 is : {:.2f}".format(i,sample_mea
    print("95% Confidence interval for age group {} for sample size 500: ({:.2f}, {:.2f})".
```

```
Sample mean of age group 0-17 for sample size 500 is : 618867.81
95% Confidence interval for age group 0-17 for sample size 500: (558782.93,
678952.69)
Sample mean of age group 18-25 for sample size 500 is : 854863.12
95% Confidence interval for age group 18-25 for sample size 500: (777066.65,
932659.59)
Sample mean of age group 26-35 for sample size 500 is : 989659.32
95% Confidence interval for age group 26-35 for sample size 500: (899256.73,
1080061.91)
Sample mean of age group 36-45 for sample size 500 is : 879665.71
95% Confidence interval for age group 36-45 for sample size 500: (793663.27,
965668.15)
Sample mean of age group 46-50 for sample size 500 is : 792548.78
95% Confidence interval for age group 46-50 for sample size 500: (711168.88,
873928.68)
Sample mean of age group 51-55 for sample size 500 is : 763200.92
95% Confidence interval for age group 51-55 for sample size 500: (693823.05,
832578.80)
Sample mean of age group 55+ for sample size 500 is : 539697.24
95% Confidence interval for age group 55+ for sample size 500: (485645.63, 5
93748.86)
```

In [64]:

```
collect_all_sample_means_age = {}
ages = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
z_99=2.576 #99% Confidence Interval

for i in ages:
    age_df = total_purchase_age[total_purchase_age['Age']==i]
    sample_mean = np.mean(age_df["Purchase"])
    population_std = np.std(age_df["Purchase"])
    sample_std_error = population_std/np.sqrt(500)

    Upper_Limit = z_99*sample_std_error + sample_mean
    Lower_Limit = sample_mean - z_99*sample_std_error
    print("Sample mean of age group {} for sample size 500 is : {:.2f}".format(i,sample_mea
    print("99% Confidence interval for age group {} for sample size 500: ({:.2f}, {:.2f})".
```

```
Sample mean of age group 0-17 for sample size 500 is : 618867.81
99% Confidence interval for age group 0-17 for sample size 500: (539899.11,
697836.51)
Sample mean of age group 18-25 for sample size 500 is : 854863.12
99% Confidence interval for age group 18-25 for sample size 500: (752616.33,
957109.91)
Sample mean of age group 26-35 for sample size 500 is : 989659.32
99% Confidence interval for age group 26-35 for sample size 500: (870844.49,
1108474.15)
Sample mean of age group 36-45 for sample size 500 is : 879665.71
99% Confidence interval for age group 36-45 for sample size 500: (766633.93,
992697.49)
Sample mean of age group 46-50 for sample size 500 is : 792548.78
99% Confidence interval for age group 46-50 for sample size 500: (685592.34,
899505.22)
Sample mean of age group 51-55 for sample size 500 is : 763200.92
99% Confidence interval for age group 51-55 for sample size 500: (672018.57,
854383.27)
Sample mean of age group 55+ for sample size 500 is : 539697.24
99% Confidence interval for age group 55+ for sample size 500: (468657.98, 6
10736.51)
```

Based on the analysis so far, we can now answer some important questions below

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

ans. No, women are not spending more money per transaction than men. If anything, the spending pattern is similar per transaction. Although, average spending when it comes to purchases per user is more for Men.

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

ans. 90% CI for a sample size - 1000

```
 CI for males:  [874103.8804056474, 976656.4556483526]
 CI for females:  [669939.9079123939, 753912.6851616061]
 This means we are confident that 90% of our sample means will lie in the range [8
74103.8804056474, 976656.4556483526] for males and in the range [669939.907912393
9, 753912.6851616061] for females

 95% CI for a sample size - 1000

 CI for males:  [864285.01681858, 986475.3192354201]
 CI for females:  [661899.9611544906, 761952.6319195094]
 This means we are confident that 95% of our sample means will lie in the range [8
64285.01681858, 986475.3192354201] for males and in the range [661899.9611544906,
 761952.6319195094] for females

 99% CI for a sample size - 1000

 CI for males:  [845083.6835816479, 1005676.6524723521]
 CI for females:  [646177.398605702, 777675.194468298]
 This means we are confident that 99% of our sample means will lie in the range [8
45083.6835816479, 1005676.6524723521] for males and in the range [646177.39860570
2, 777675.194468298] for females

 90% CI for a sample size - 1250

 CI for males:  [880374.02870753, 972099.8405116702]
 CI for females:  [673842.1899862627, 748949.7252617371]
 This means we are confident that 90% of our sample means will lie in the range [8
80374.02870753, 972099.8405116702] for males and in the range [673842.1899862627,
 748949.7252617371] for females

 95% CI for a sample size - 1250

 CI for males:  [871591.7701305378, 980882.0990886624]
 CI for females:  [666651.0429918023, 756140.8722561975]
 This means we are confident that 95% of our sample means will lie in the range [8
71591.7701305378, 980882.0990886624] for males and in the range [666651.042991802
3, 756140.8722561975] for females

 99% CI for sample size -1250

 CI for males:  [854417.5755799754, 998056.2936392248]
 CI for females:  [652588.3555359688, 770203.559712031]
 This means we are confident that 95% of our sample means will lie in the range [8
54417.5755799754, 998056.2936392248] for males and in the range [652588.355535968
8, 770203.559712031] for females

 Here we can note that the confidence interval becomes smaller as the sample size
 increases
```

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

ans. Yes, as we can see above, the confidence intervals of males and females are overlapping. It means that there is a high degree of uncertainty or variability in the data, and it is difficult to draw definitive conclusions about any differences between the two groups. However, it is important to note that the overlap of confidence intervals does not necessarily mean that there is no difference between the groups. It just means that the data is not precise enough to be confident in any differences that may exist.

4. Results when the same activity is performed for Married vs Unmarried ans. 90% CI for a sample size - 1000

   CI for Unmarried customers: [829918.3019815636, 928682.1832404366]

   CI for Married customers: [796191.3833064862, 893484.752171514]

   This means we are confident that 90% of our sample means will lie in the range [829918.3019815636, 928682.1832404366] for unmarried customers and in the range [796191.3833064862, 893484.752171514] for married customers

   95% CI for a sample size - 1000

   CI for Unmarried customers: [820462.1856908204, 938138.2995311797]

   CI for Married customers: [786876.0607555793, 902800.0747224209]

   This means we are confident that 95% of our sample means will lie in the range [820462.1856908204, 938138.2995311797] for unmarried customers and in the range [786876.0607555793, 902800.0747224209] for married customers

   99% CI for a sample size - 1000

   CI for Unmarried customers: [801970.2249444783, 956630.2602775219]

   CI for Married customers: [768659.4299893613, 921016.7054886388]

   This means we are confident that 99% of our sample means will lie in the range [801970.2249444783, 956630.2602775219] for unmarried customers and in the range [768659.4299893613, 921016.7054886388] for married customers

   90% CI for a sample size - 1250

   CI for Unmarried customers: [836594.2250366886, 924931.3259233115]

   CI for Married customers: [799967.0254235673, 886988.8600404328]

   This means we are confident that 90% of our sample means will lie in the range [836594.2250366886, 924931.3259233115] for unmarried customers and in the range [799967.0254235673, 886988.8600404328] for married customers

   95% CI for a sample size - 1250

   CI for Unmarried customers: [828136.4175049906, 933389.1334550095]

   CI for Married customers: [791635.1476411015, 895320.7378228985]

   This means we are confident that 95% of our sample means will lie in the range [828136.4175049906, 933389.1334550095] for unmarried customers and in the range [791635.1476411015, 895320.7378228985] for married customers

   99% CI for a sample size - 1250

CI for Unmarried customers: [811596.7049985591, 949928.845961441]

CI for Married customers: [775341.6977553905, 911614.1877086095]

This means we are confident that 99% of our sample means will lie in the range [811596.7049985591, 949928.845961441] for unmarried customers and in the range [775341.6977553905, 911614.1877086095] for married customers

Here we can note that the confidence interval becomes smaller as the sample size increases

5. Results when the same activity is performed for Age ans. 90% Confidence interval for age group 0-17 for sample size 300 : (553765.05, 683970.57) 90% Confidence interval for age group 18-25 for sample size 300 : (770569.62, 939156.62) 90% Confidence interval for age group 26-35 for sample size 300 : (891706.92, 1087611.72) 90% Confidence interval for age group 36-45 for sample size 300 : (786480.93, 972850.49) 90% Confidence interval for age group 46-50 for sample size 300 : (704372.59, 880724.98) 90% Confidence interval for age group 51-55 for sample size 300 : (688029.08, 838372.77) 90% Confidence interval for age group 55+ for sample size 300 : (481131.61, 598262.88)

```
 95% Confidence interval for age group 0-17 for sample size 300 : (541298.56, 6964
37.06)
 95% Confidence interval for age group 18-25 for sample size 300 : (754428.31, 955
297.93)
 95% Confidence interval for age group 26-35 for sample size 300 : (872950.08, 110
6368.56)
 95% Confidence interval for age group 36-45 for sample size 300 : (768637.03, 990
694.39)
 95% Confidence interval for age group 46-50 for sample size 300 : (687487.78, 897
609.78)
 95% Confidence interval for age group 51-55 for sample size 300 : (673634.47, 852
767.38)
 95% Confidence interval for age group 55+ for sample size 300 : (469916.91, 60947
7.58)


 99% Confidence interval for age group 0-17 for sample size 300: (516919.66, 72081
5.97)
 99% Confidence interval for age group 18-25 for sample size 300: (722863.08, 9868
63.16)
 99% Confidence interval for age group 26-35 for sample size 300: (836270.03, 1143
048.60)
 99% Confidence interval for age group 36-45 for sample size 300: (733742.31, 1025
589.11)
 99% Confidence interval for age group 46-50 for sample size 300: (654468.61, 9306
28.95)
 99% Confidence interval for age group 51-55 for sample size 300: (645485.01, 8809
16.83)
 99% Confidence interval for age group 55+ for sample size 300: (447985.95, 63140
8.54)


 90% Confidence interval for age group 0-17 for sample size 500: (568439.43, 66929
6.19)
 90% Confidence interval for age group 18-25 for sample size 500: (789569.65, 9201
56.59)
 90% Confidence interval for age group 26-35 for sample size 500: (913785.72, 1065
532.92)
 90% Confidence interval for age group 36-45 for sample size 500: (807485.09, 9518
46.33)
 90% Confidence interval for age group 46-50 for sample size 500: (724247.79, 8608
49.77)
 90% Confidence interval for age group 51-55 for sample size 500: (704973.06, 8214
28.78)
 90% Confidence interval for age group 55+ for sample size 500: (494332.50, 58506
1.99)


 95% Confidence interval for age group 0-17 for sample size 500: (558782.93, 67895
2.69)
 95% Confidence interval for age group 18-25 for sample size 500: (777066.65, 9326
59.59)
```

```
 95% Confidence interval for age group 26-35 for sample size 500: (899256.73, 1080
061.91)
 95% Confidence interval for age group 36-45 for sample size 500: (793663.27, 9656
68.15)
 95% Confidence interval for age group 46-50 for sample size 500: (711168.88, 8739
28.68)
 95% Confidence interval for age group 51-55 for sample size 500: (693823.05, 8325
78.80)
 95% Confidence interval for age group 55+ for sample size 500: (485645.63, 59374
8.86)


 99% Confidence interval for age group 0-17 for sample size 500: (539899.11, 69783
6.51)
 99% Confidence interval for age group 18-25 for sample size 500: (752616.33, 9571
09.91)
 99% Confidence interval for age group 26-35 for sample size 500: (870844.49, 1108
474.15)
 99% Confidence interval for age group 36-45 for sample size 500: (766633.93, 9926
97.49)
 99% Confidence interval for age group 46-50 for sample size 500: (685592.34, 8995
05.22)
 99% Confidence interval for age group 51-55 for sample size 500: (672018.57, 8543
83.27)
 99% Confidence interval for age group 55+ for sample size 500: (468657.98, 61073
6.51)

 Here we can note that the confidence interval becomes smaller as the sample size
 increases
```

In [ ]: