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 how the spending habits differ between male and female customers.

Analysing Basic metrics

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

In [2]:
```
df = pd.read_csv(r'C:\Users\walmart_data.csv')
```

In [3]:
```
df.head()
```

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

In [4]:
```
df.shape
```

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

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64 
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

There are no missing values in the data.

In [119…

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

Out[119]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	
count	550068.0	550068	550068	550068	550068.0	550068	550068	550068.0	550068.0	5
unique	5891.0	3631	2	7	21.0	3	5	2.0	20.0	
top	1001680.0	P00265242	M	26-35	4.0	B	1	0.0	5.0	
freq	1026.0	1880	414259	219587	72308.0	231173	193821	324731.0	150933.0	
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

In [10]:
```
df['Age'].unique()
```

Out[10]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

In [12]:
```
df['Occupation'].unique()
```

Out[12]:
```
array([10, 16, 15,  7, 20,  9,  1, 12, 17,  0,  3,  4, 11,  8, 19,  2, 18,
        5, 14, 13,  6], dtype=int64)
```

In [13]:
```
df['City_Category'].unique()
```

Out[13]:
```
array(['A', 'C', 'B'], dtype=object)
```

In [14]:
```
df['Stay_In_Current_City_Years'].unique()
```

Out[14]:
```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

In [15]:
```
df['Product_Category'].unique()
```

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

There are 7 unique age groups and most of the purchase belongs to age 26-35 group.

There are 3 unique city categories.

There are 5 unique values for Stay_in_current_citi_years.

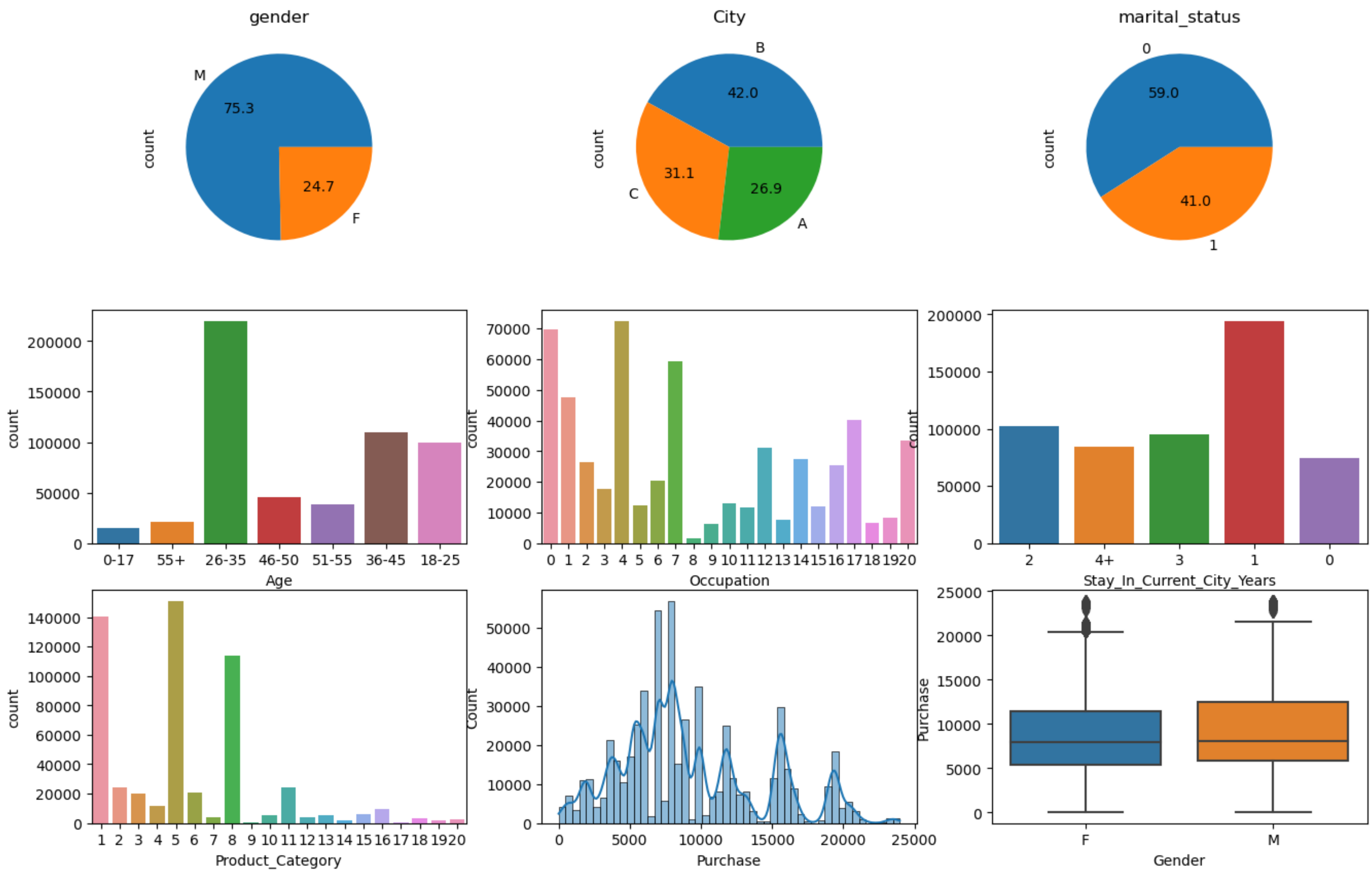
Standard deviation for Purchase is quite high suggesting widespread data with outliers.

There are 5891 unique customer IDs .

The customers belong to 21 distinct occupation.

There are 20 unique product categories

In [36]:
```
fig, axis = plt.subplots(nrows=3, ncols=3, figsize=(16, 10))
df['Gender'].value_counts().plot(kind='pie',autopct="%.1f",ax = axis[0][0]).set_title('gender')
df['City_Category'].value_counts().plot(kind='pie',autopct="%.1f",ax = axis[0][1]).set_title('City')
df['Marital_Status'].value_counts().plot(kind='pie',autopct="%.1f",ax = axis[0][2]).set_title('marital_status')
sns.countplot(x= 'Age',data =df,ax = axis[1][0])
sns.countplot(x= 'Occupation',data =df,ax = axis[1][1])
sns.countplot(x= 'Stay_In_Current_City_Years',data =df,ax = axis[1][2])
sns.countplot(x= 'Product_Category',data =df,ax = axis[2][0])
sns.histplot(x= 'Purchase',data =df,kde = True,bins = 50,ax = axis[2][1])
sns.boxplot(x='Gender',y='Purchase',data = df,ax=axis[2][2])
plt.show()
```


gender
count
M
75.3
24.7
F
City
count
B
42.0
31.1
C
26.9
A
marital_status
count
0
59.0
41.0
1
count
Age
0-17
55+
26-35
46-50
51-55
36-45
18-25
count
Occupation
count
Stay_In_Current_City_Years
2
4+
3
1
0
count
Product_Category
Count
Purchase
Purchase
Gender
F
M

Male purchase more products than females.

The city with highest purchases is B.

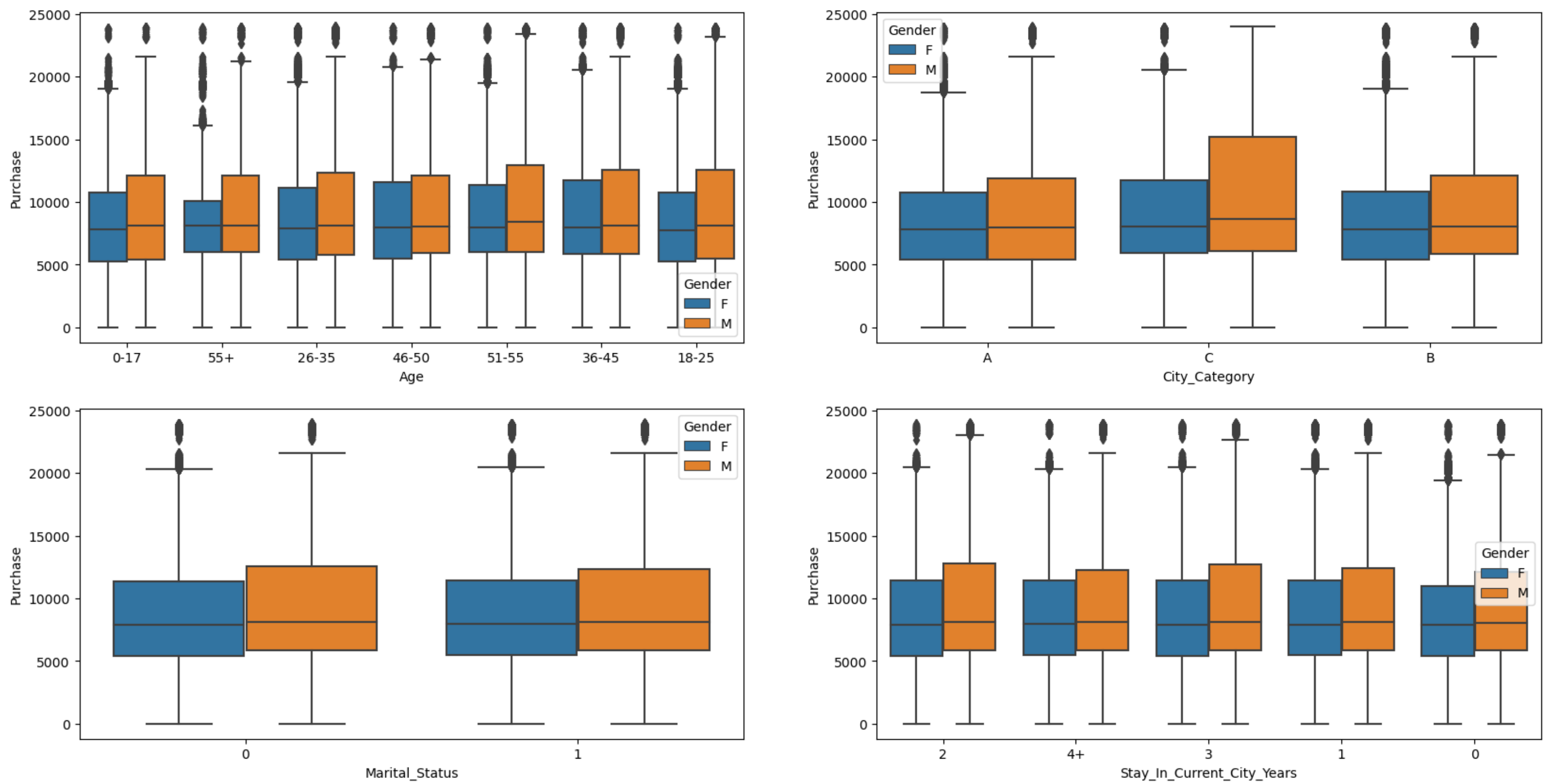
Married people purchase more products than unmarried.

Most of the buyers are in the age group 26-35.

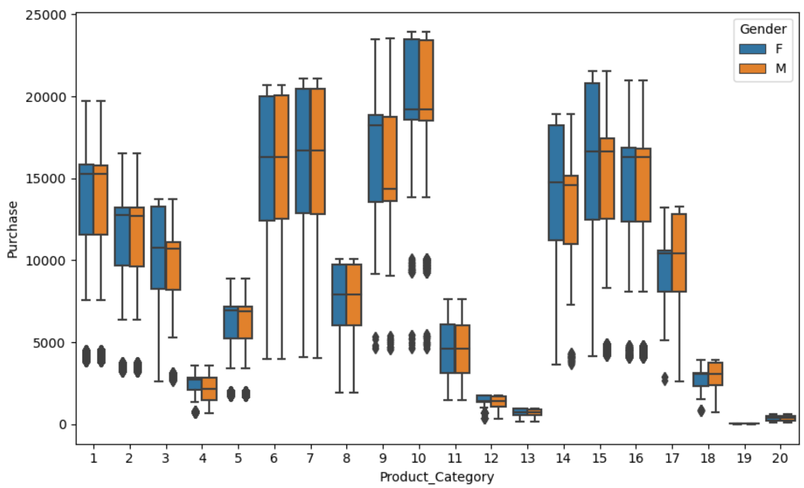
Most of the products are purchased with purchase amount 8000-9000

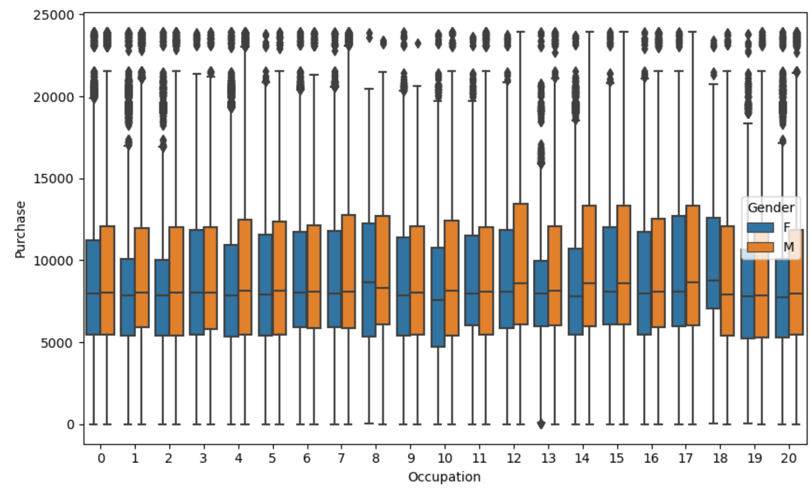
In [38]:

```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20, 10))

sns.boxplot(data=df, y='Purchase', hue='Gender', x='Age', ax=axs[0,0])
sns.boxplot(data=df, y='Purchase', hue='Gender', x='City_Category', ax=axs[0,1])

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

plt.show()
```



In [43]:
```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(22, 6))

sns.boxplot(data=df, y='Purchase', hue='Gender', x='Product_Category', ax=axs[0])
sns.boxplot(data=df, y='Purchase', hue='Gender', x='Occupation', ax=axs[1])
plt.show()
```





1.The spending behaviour for males and females are similar as we had seen from the above histplot. Males purchasing value are in higher range.

2.there are few outliers for some of the product categories

In [44]:
```
avg_gender = df.groupby(['User_ID', 'Gender'])[['Purchase']].sum()
avg_gender = avg_gender.reset_index()
avg_gender
```

Out[44]:

	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 [45]:

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

In [46]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  object
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  object
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  object
 8   Product_Category            550068 non-null  object
 9   Purchase                    550068 non-null  int64 
dtypes: int64(1), object(9)
memory usage: 42.0+ MB
```

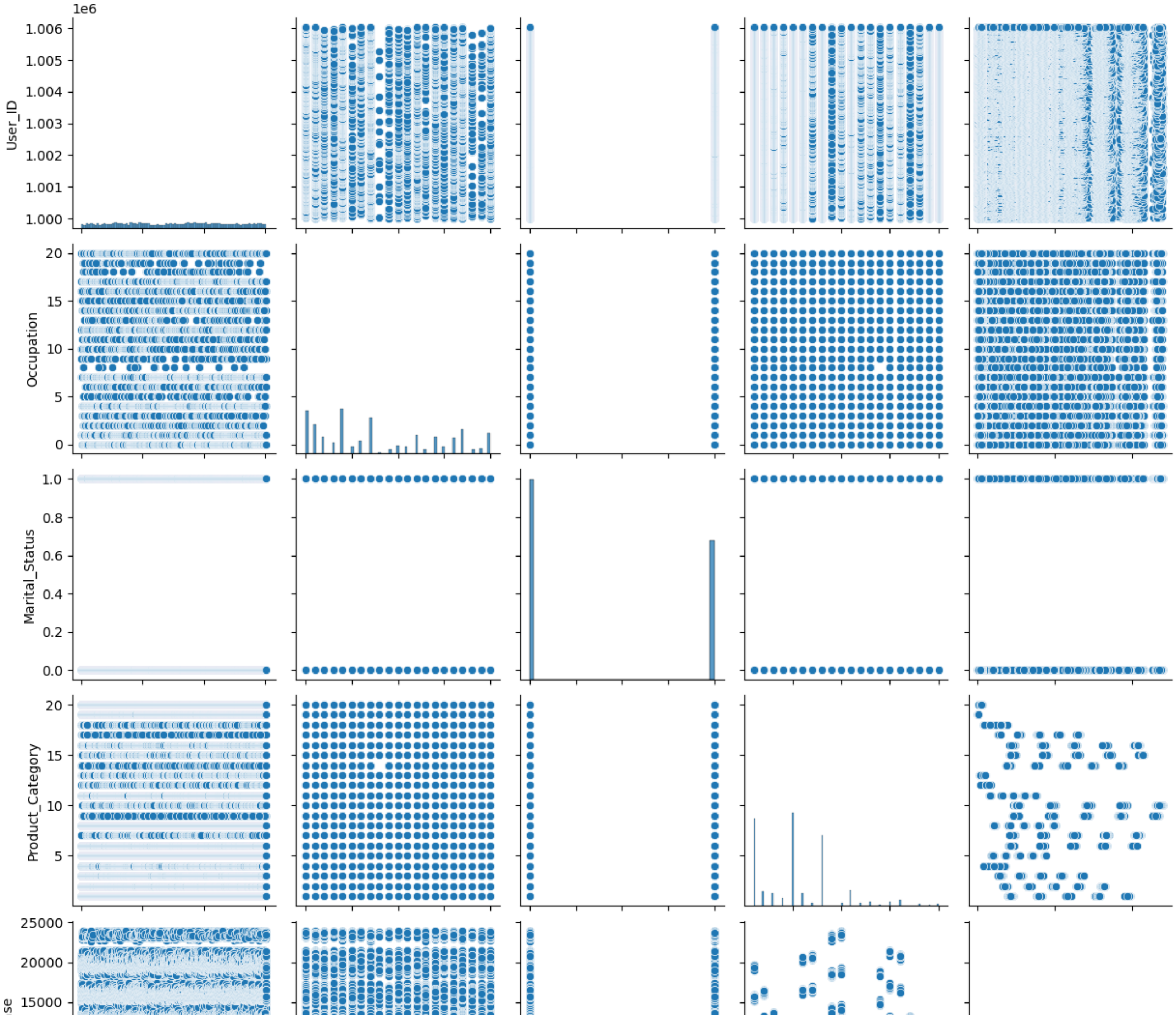
In [49]:

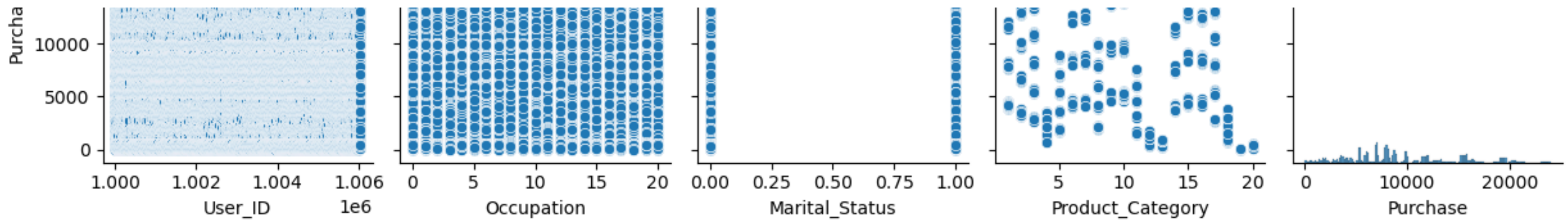
```
sns.pairplot(df)
```

```
C:\Users\cardi\anaconda3\conda-meta\anacond\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layou
t has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[49]:

```
<seaborn.axisgrid.PairGrid at 0x2206773d850>
```

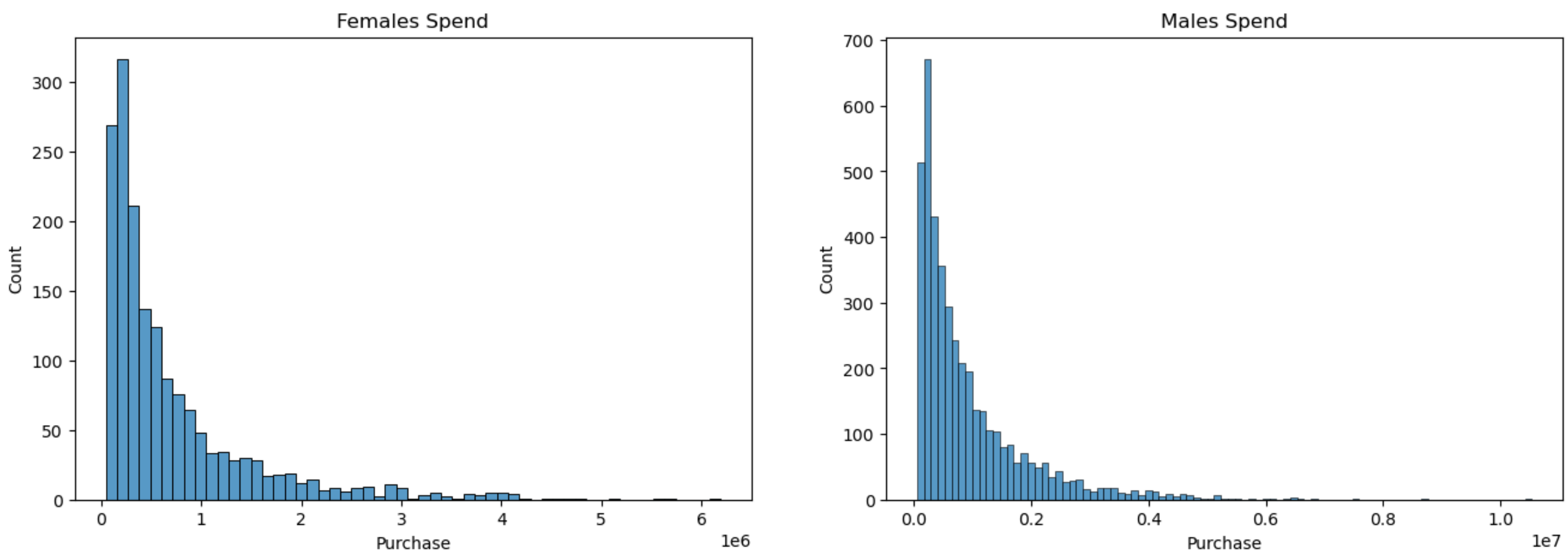

1e6
User_ID
1.006
1.005
1.004
1.003
1.002
1.001
1.000
Occupation
20
15
10
5
0
Marital_Status
1.0
0.8
0.6
0.4
0.2
0.0
Product_Category
20
15
10
5
25000
20000
15000
se



The correlation between the categories is very low.

In [50]:
```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 Spend")
sns.histplot(data=avg_gender[avg_gender['Gender']=='M']['Purchase'], ax=axs[1]).set_title("Males Spend")
```

Out[50]:
```
Text(0.5, 1.0, 'Males Spend')
```



The amount spend by males is higher than females

In [51]:
```python
avg_gender.groupby(['Gender'])[['Purchase']].mean()
```

Out[51]:

	Purchase
Gender	
F	712024.394958
M	925344.402367

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

In [53]:
```
avg_gender.groupby(['Gender'])['Purchase'].sum()
```

Out[53]:
```
Gender
F    1186232642
M    3909580100
Name: Purchase, dtype: int64
```

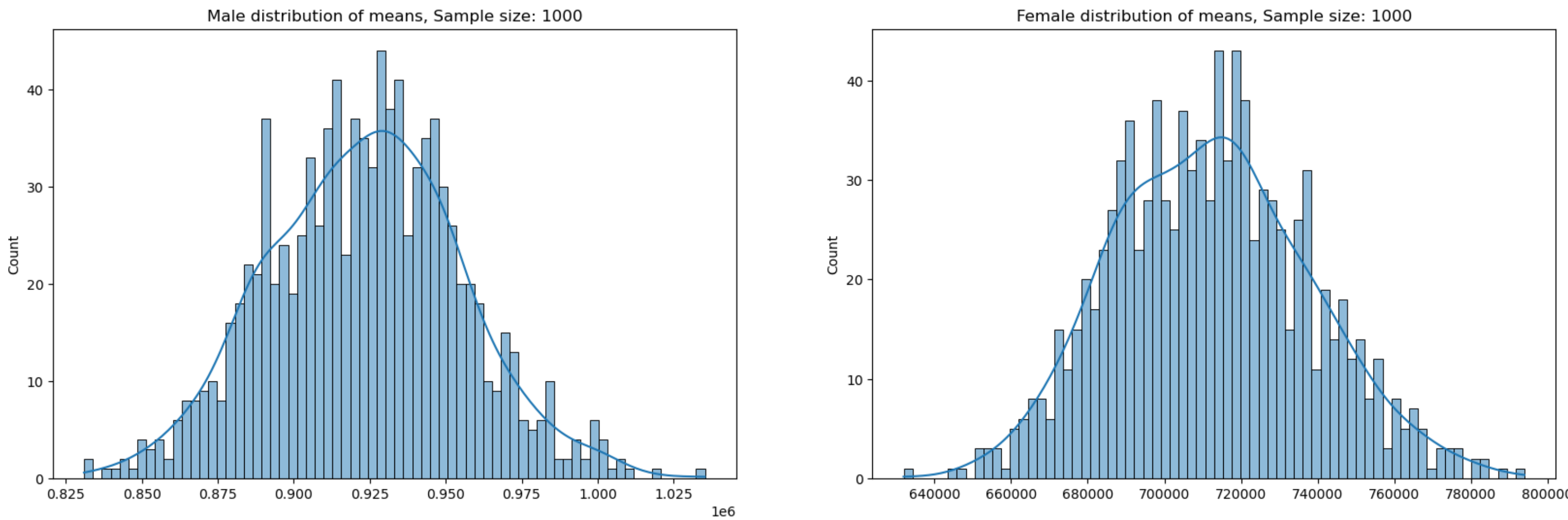
Total amount spend by males is around 4 billion whereas for females it's 1.2 billion

In [54]:
```
avg_male = avg_gender[avg_gender['Gender']=='M']
avg_female = avg_gender[avg_gender['Gender']=='F']
```

In [55]:
```
sample_size = 1000
rep = 1000
male_means = []
female_means = []

for i in range(rep):
    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 [59]:
```
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))

sns.histplot(male_means, bins=70,kde=True,ax =axis[0])
sns.histplot(female_means, bins=70 , kde=True,ax=axis[1])
axis[0].set_title("Male distribution of means, Sample size: 1000")
axis[1].set_title("Female distribution of means, Sample size: 1000")
```

Out[59]:
```
Text(0.5, 1.0, 'Female distribution of means, Sample size: 1000')
```

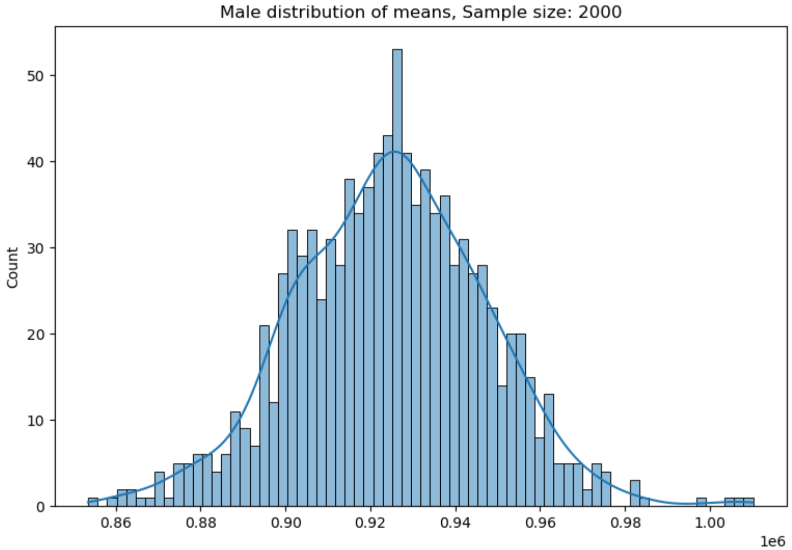


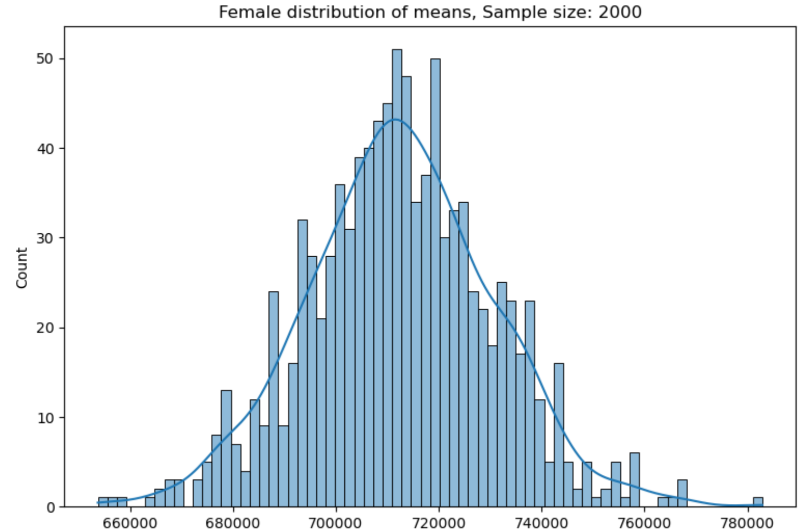
In [61]:

```python
sample_size = 2000
rep = 1000
male_means = []
female_means = []

for i in range(rep):
    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)
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))

sns.histplot(male_means, bins=70,kde=True,ax =axis[0])
sns.histplot(female_means, bins=70 , kde=True,ax=axis[1])
axis[0].set_title("Male distribution of means, Sample size: 2000")
axis[1].set_title("Female distribution of means, Sample size: 2000")
```

Out[61]:

```
Text(0.5, 1.0, 'Female distribution of means, Sample size: 2000')
```





The mean sample is normally distributed for both males and females. Also, we can see the mean of the sample means are closer to the population mean 925344 & 712024 respectively.

Calculating 90% confidence interval for sample size 1000

In [62]:
```
z90=1.645 #90% Confidence Interval
z95=1.960 #95% Confidence Interval
z99=2.576 #99% Confidence Interval
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])
```

```
Male_CI:  [923479.2312120051, 925800.9537359948]
Female_CI:  [711563.2720763895, 713436.8431226104]
```

In [63]:
```
sample_mean_male
```

Out[63]:
```
924640.092474
```

In [64]:
```
sample_mean_female
```

Out[64]:
```
712500.0575994999
```

In [65]:
```
sample_std_male
```

Out[65]:
```
22315.90051891312
```

In [66]:
```
sample_std_female
```

Out[66]:
```
18008.364328882617
```

In [67]:
```
sample_std_error_male
```

Out[67]:
```
705.6907367749891
```

In [68]:
```
sample_std_error_female
```

Out[68]:
```
569.4744821339863
```

Calculating 95% confidence interval for sample size 1000

In [69]:
```
sample_mean_male=np.mean(male_means)
sample_mean_female=np.mean(female_means)

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

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

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

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

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

```
Male_CI:  [923256.938629921, 926023.246318079]
Female_CI:  [711383.8876145174, 713616.2275844825]
```

Calculating 99% confidence interval for sample size 1000

In [70]:
```
sample_mean_male=np.mean(male_means)
sample_mean_female=np.mean(female_means)

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

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

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

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

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

```
Male_CI:  [922822.2331360676, 926457.9518119323]
Female_CI:  [711033.0913335228, 713967.0238654771]
```

With 90% confidence interval, we can say that:

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

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

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

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

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

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

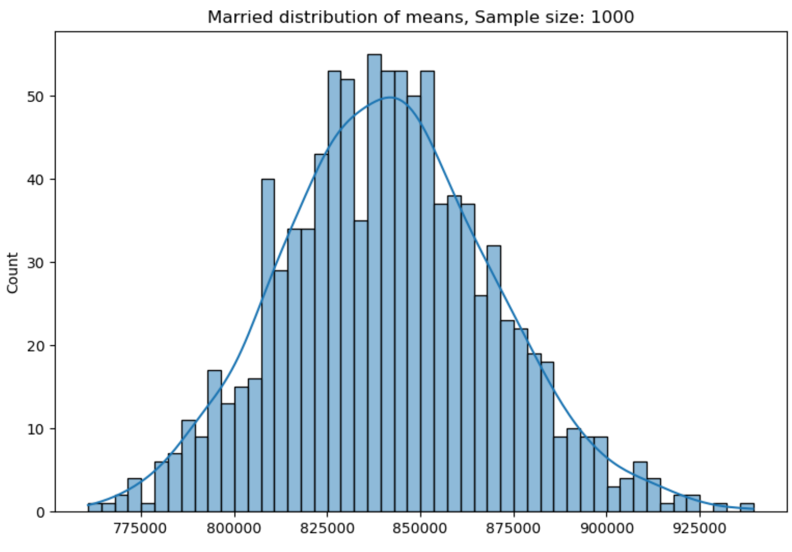
Average amount spend by male customers lie in the range 9,22,011.28 - 9,27,154.61

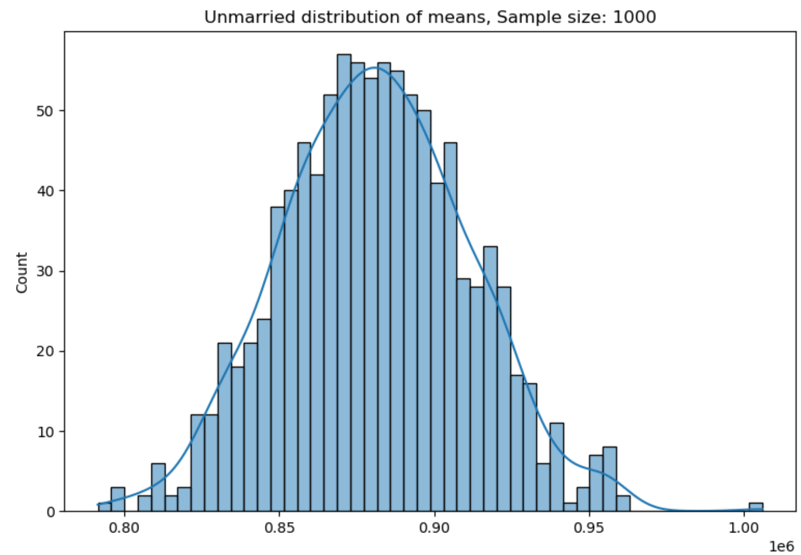
Average amount spend by female customers lie in range 7,09,678.88 - 7,13,811.31

Confidence interval considering marital status

In [73]:

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

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

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

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

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


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

sns.histplot(married_means, bins=50,kde=True,ax= axis[0])
sns.histplot(single_means, bins=50,kde=True,ax=axis[1])
axis[0].set_title("Married distribution of means, Sample size: 1000")
axis[1].set_title("Unmarried distribution of means, Sample size: 1000")

plt.show()
```





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

Calculating 90% confidence interval for avg expenses for married/single for sample size 1000:

In [74]:
```
sample_mean_married=np.mean(married_means)
sample_mean_single=np.mean(single_means)

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

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

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

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

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

```
Married_CI:  [839788.5512888249, 842427.5026689948]
Single_CI:  [879637.2257680065, 882766.3390859935]
```

Calculating 95% confidence interval for avg expenses for married/single for sample size 1000:

In [76]:

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

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

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

Upper_Limit_married=z90*sample_std_error_male + sample_mean_married
Lower_Limit_married=sample_mean_married - z95*sample_std_error_married

Upper_Limit_single=z90*sample_std_error_single + sample_mean_single
Lower_Limit_single=sample_mean_single - z95*sample_std_error_single

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

```
Married_CI:  [839505.5127555572, 842427.5026689948]
Single_CI:  [879337.6298120291, 882766.3390859935]
```

Calculating 99% confidence interval for avg expenses for married/single for sample size 1000:

In [77]:
```
sample_mean_married=np.mean(married_means)
sample_mean_single=np.mean(single_means)

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

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

Upper_Limit_married=z90*sample_std_error_male + sample_mean_married
Lower_Limit_married=sample_mean_married - z99*sample_std_error_married

Upper_Limit_single=z90*sample_std_error_single + sample_mean_single
Lower_Limit_single=sample_mean_single - z99*sample_std_error_single

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

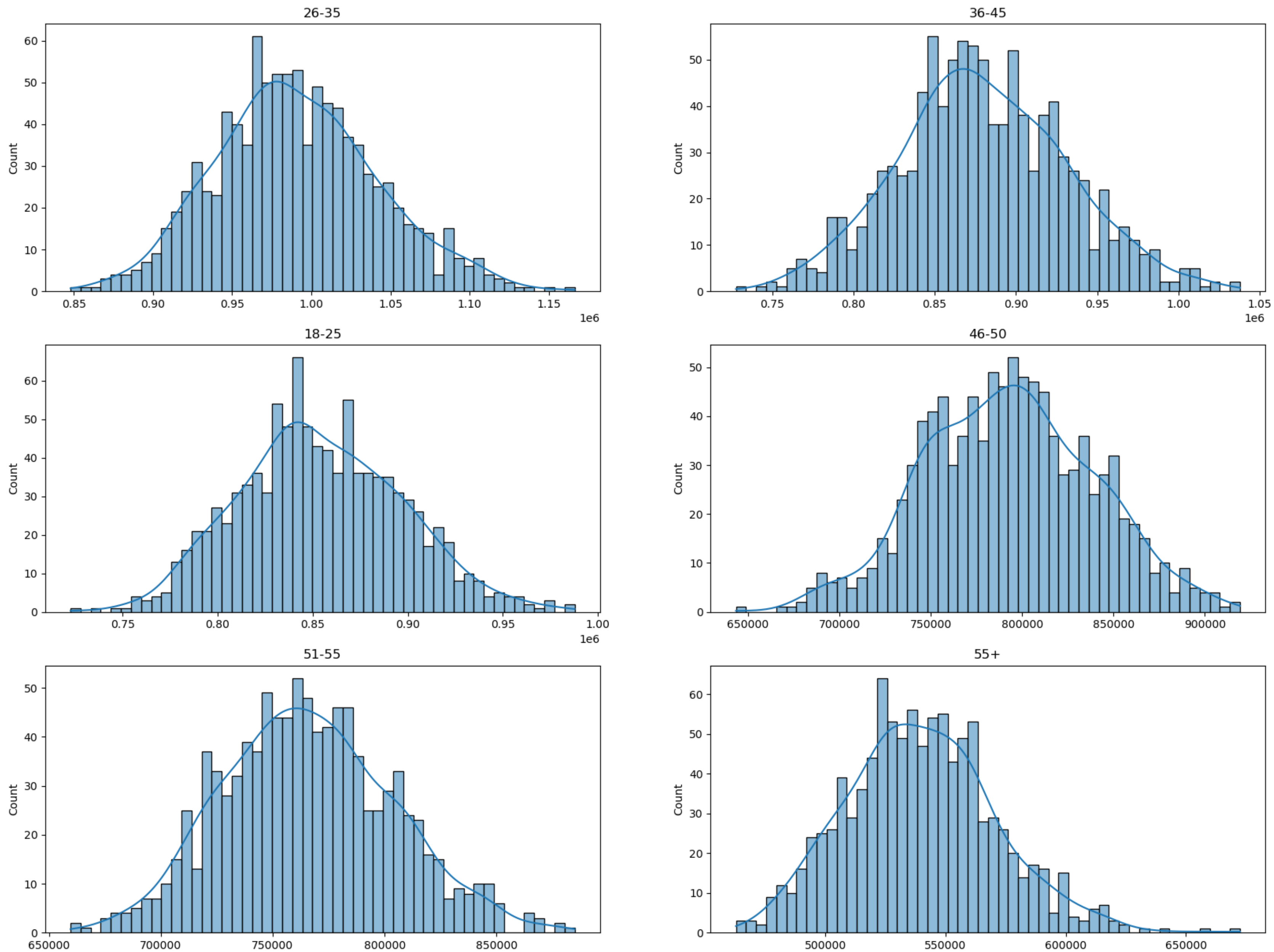
```
Married_CI:  [838952.0151793895, 842427.5026689948]
Single_CI:  [878751.7532758954, 882766.3390859935]
```

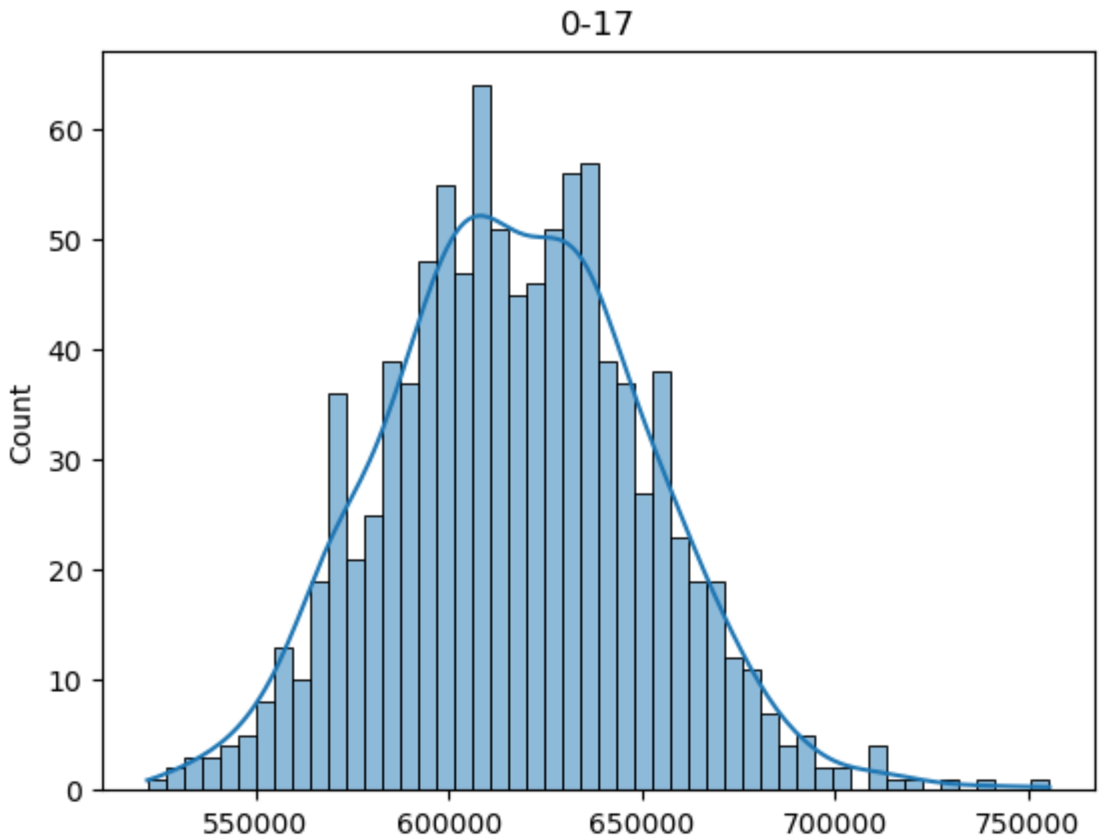
Confidence interval considering age

In [80]:
```
avg_age = df.groupby(['User_ID', 'Age'])[['Purchase']].sum()
avg_age = avg_age.reset_index()
sample_size = 400
num_repitions = 1000

all_sample_means = {}

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

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

        mean = avg_age[avg_age['Age']==i].sample(sample_size, replace=True)['Purchase'].mean()
        all_sample_means[i].append(mean)
```

In [116…

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

sns.histplot(all_sample_means['26-35'],bins=50,kde=True,ax=axis[0,0]).set_title('26-35')
sns.histplot(all_sample_means['36-45'],bins=50,kde=True,ax=axis[0,1]).set_title('36-45')
sns.histplot(all_sample_means['18-25'],bins=50,kde=True,ax=axis[1,0]).set_title('18-25')
sns.histplot(all_sample_means['46-50'],bins=50,kde=True,ax=axis[1,1]).set_title('46-50')
sns.histplot(all_sample_means['51-55'],bins=50,kde=True,ax=axis[2,0]).set_title('51-55')
sns.histplot(all_sample_means['55+'],bins=50,kde=True,ax=axis[2,1]).set_title('55+')
plt.show()
sns.histplot(all_sample_means['0-17'],bins=50,kde=True).set_title('0-17')

plt.show()
```


26-35
36-45
18-25
46-50
51-55
55+
Count
1e6


0-17
Count
0
10
20
30
40
50
60
550000
600000
650000
700000
750000

In [122…

```
df2 = df.groupby(['User_ID', 'Age'])[['Purchase']].sum()
df2 = df2.reset_index()
df2
sample_size = 400
num_repitions = 1000
all_means = {}
age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+',
'0-17']
for age_interval in age_intervals:
    all_means[age_interval] = []
for age_interval in age_intervals:
    for _ in range(num_repitions):
        amt = df2[df2['Age']==age_interval].sample(sample_size,
replace=True)['Purchase'].mean()
        all_means[age_interval].append(amt)
for val in ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']:
    new_df = df2[df2['Age']==val]
    std_error = z90*new_df['Purchase'].std()/np.sqrt(len(new_df))
    sample_mean = new_df['Purchase'].mean()
    lower_lim = sample_mean - std_error
    upper_lim = sample_mean + std_error
    print("For age {} confidence interval of 90% mean: ({:.2f}, {:.2f})".format(val, lower_lim, upper_lim))
```

```
For age 26-35 confidence interval of 90% mean: (952206.28, 1027112.35)
For age 36-45 confidence interval of 90% mean: (832398.89, 926932.53)
For age 18-25 confidence interval of 90% mean: (810187.65, 899538.59)
For age 46-50 confidence interval of 90% mean: (726209.00, 858888.57)
For age 51-55 confidence interval of 90% mean: (703772.36, 822629.48)
For age 55+ confidence interval of 90% mean: (487032.92, 592361.57)
For age 0-17 confidence interval of 90% mean: (542320.46, 695415.16)
```

In [121…

```
df2 = df.groupby(['User_ID', 'Age'])[['Purchase']].sum()
df2 = df2.reset_index()
df2
sample_size = 400
num_repitions = 1000
all_means = {}
age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+',
'0-17']
for age_interval in age_intervals:
    all_means[age_interval] = []
for age_interval in age_intervals:
    for _ in range(num_repitions):
        amt = df2[df2['Age']==age_interval].sample(sample_size,
replace=True)['Purchase'].mean()
        all_means[age_interval].append(amt)
for val in ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']:
    new_df = df2[df2['Age']==val]
    std_error = z95*new_df['Purchase'].std()/np.sqrt(len(new_df))
    sample_mean = new_df['Purchase'].mean()
    lower_lim = sample_mean - std_error
    upper_lim = sample_mean + std_error
    print("For age {} confidence interval of 95% means: ({:.2f}, {:.2f})".format(val, lower_lim, upper_lim))
```

```
For age 26-35 confidence interval of 95% means: (945034.42, 1034284.21)
For age 36-45 confidence interval of 95% means: (823347.80, 935983.62)
For age 18-25 confidence interval of 95% means: (801632.78, 908093.46)
For age 46-50 confidence interval of 95% means: (713505.63, 871591.93)
For age 51-55 confidence interval of 95% means: (692392.43, 834009.42)
For age 55+ confidence interval of 95% means: (476948.26, 602446.23)
For age 0-17 confidence interval of 95% means: (527662.46, 710073.17)
```

In [123…

```
df2 = df.groupby(['User_ID', 'Age'])[['Purchase']].sum()
df2 = df2.reset_index()
df2
sample_size = 400
num_repitions = 1000
all_means = {}
age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+',
'0-17']
for age_interval in age_intervals:
    all_means[age_interval] = []
for age_interval in age_intervals:
    for _ in range(num_repitions):
        amt = df2[df2['Age']==age_interval].sample(sample_size,
replace=True)['Purchase'].mean()
        all_means[age_interval].append(amt)
for val in ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']:
    new_df = df2[df2['Age']==val]
    std_error = z99*new_df['Purchase'].std()/np.sqrt(len(new_df))
    sample_mean = new_df['Purchase'].mean()
    lower_lim = sample_mean - std_error
    upper_lim = sample_mean + std_error
    print("For age {} confidence interval of 99% means({:.2f}, {:.2f})".format(val, lower_lim, upper_lim))
```

```
For age 26-35 confidence interval of 99% means(931009.46, 1048309.18)
For age 36-45 confidence interval of 99% means(805647.89, 953683.53)
For age 18-25 confidence interval of 99% means(784903.24, 924823.00)
For age 46-50 confidence interval of 99% means(688663.50, 896434.06)
For age 51-55 confidence interval of 99% means(670138.33, 856263.52)
For age 55+ confidence interval of 99% means(457227.15, 622167.34)
For age 0-17 confidence interval of 99% means(498997.92, 738737.71)
```

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

Recommendations

1.Men spent more money than women, company should focus on retaining the male customers and getting more female customers.

2.Product_Category - 1, 5, 8 have highest purchasing frequency. it means these are the products in these categories are in more demand. Company can focus on selling more of these products.

3.Company should focus on acquisition of Unmarried customers.

4.Customers in the age 26-35 spend more money than the others, company should focus on acquisition of young customers.

5.We have more customers aged 26-35 in the city category B and A, company can focus more on these customers for these cities.

6.Male customers living in City_Category C spend more money than other male customers living in B or C, Selling more products in the City_Category C will help the company increase the revenue.

7.Some of the Product category like 19,20,13 have very less purchase , company should discontinue those products.

8.The occupation which are contributing more company can think of offering benefits to those customers to increase the sales.

9.People who are staying in city for an year have contributed more to the total purchase amount. Company can focus on such customer base who are neither too old nor too new residents in the city.

10.We have highest frequency of purchase order between 7k and 10k, company can focus more on these mid range products to increase the sales

Answers to the questions

-------Are women spending more money per transaction than men?

No.upper limit of female purchase CI is less than lower limit of male purchase CI. Reasons for the above could be: 1.Males have higher salary than females. 2.There are more male oriented products in the store than female. 3.Cost of female oriented products is high. 4.In married couples males are doing the shopping for female.

-------Confidence intervals and distribution of the mean of the expenses by female and male customer:

At 99% CI with sample size of 1000 Avg amount spend by males lie in the range - 9,22,822.23 - 9,26,457 Avg amount spend by females lie in the range - 7,11,033.09 - 7,13,967.02

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

No. Walmart can focus more on women based products.

------Results when the same activity is performed for Married vs Unmarried

At 99% Confidence Interval with sample size 1000

Average amount spend by married customers lie in the range: 8,38,952 - 8,42,427 Average amount spend by unmarried customers lie in the range: 8,78,751 - 8,82,766

------Results when the same activity is performed for Age:

At 99% Confidence Interval with sample size 400

For age 26-35 confidence interval of 99% means(931009.46 -1048309.18) For age 36-45 confidence interval of 99% means(805647.89 - 953683.53) For age 18-25 confidence interval of 99% means(784903.24 - 924823.00) For age 46-50 confidence interval of 99% means(688663.50 - 896434.06) For age 51-55 confidence interval of 99% means(670138.33 - 856263.52) For age 55+ confidence interval of 99% means(457227.15 - 622167.34) For age 0-17 confidence interval of 99% means(498997.92 - 738737.71)

In [ ]: