In [1]:

```
import pandas as pd
import seaborn as sbn
import matplotlib.pyplot as plt
```

In [2]:

```
df=pd.read_csv('/Users/suraaj/Desktop/Datasets/walmart.csv')
```

In [3]:

```
df.head(10)
```

Out[3]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	M
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+	
5	1000003	P00193542	M	26-35	15	A	3	
6	1000004	P00184942	M	46-50	7	B	2	
7	1000004	P00346142	M	46-50	7	B	2	
8	1000004	P0097242	M	46-50	7	B	2	
9	1000005	P00274942	M	26-35	20	A	1	

In [4]:

```
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
```

In [5]:

```
#Rows calculation
for i in ['Gender','Marital_Status','Age']:
    print(df[i].value_counts(normalize=True)*100)
    print('-'*50)
```

```
M    75.310507
F    24.689493
Name: Gender, dtype: float64
--------------------------------------------------
0    59.034701
1    40.965299
Name: Marital_Status, dtype: float64
--------------------------------------------------
26-35    39.919974
36-45    19.999891
18-25    18.117760
46-50     8.308246
51-55     6.999316
55+       3.909335
0-17      2.745479
Name: Age, dtype: float64
--------------------------------------------------
```

In [6]:

```
#Rows calculation
for i in ['Gender','Marital_Status','Age']:
    print(df[i].value_counts())
    print('-'*50)
```

```
M    414259
F    135809
Name: Gender, dtype: int64
--------------------------------------------------
0    324731
1    225337
Name: Marital_Status, dtype: int64
--------------------------------------------------
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
--------------------------------------------------
```

In [7]:

```
df.groupby('Gender')['User_ID'].nunique()
```

Out[7]:

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

Population:

Males and Females: 50 Mn

In [8]:

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

Out[8]:

Gender	count	mean	std	min	25%	50%	75%	max
F	135809.0	8734.565765	4767.233289	12.0	5433.0	7914.0	11400.0	23959.0
M	414259.0	9437.526040	5092.186210	12.0	5863.0	8098.0	12454.0	23961.0

Median of purchases for Women: 7914

Median of purchases for male: 8098

[7000-8000] women

[7900-8500] male

We can't conculude anything if the confidence interval overlapps

In [9]:

```
df.sample(300).groupby('Gender')['Purchase'].describe()
```

Out[9]:

Gender	count	mean	std	min	25%	50%	75%	max
F	76.0	8632.447368	5056.999842	477.0	5300.25	7851.0	10887.5	20732.0
M	224.0	9673.133929	5111.146199	567.0	5866.50	8132.0	12595.5	23451.0

In [10]:

```
male_sample_means=[df[df['Gender']=='M']['Purchase'].sample(300).mean() for i in ran
```

In [11]:

```
female_sample_means=[df[df['Gender']=='F']['Purchase'].sample(300).mean() for i in r
```

In [12]:

```
import numpy as np
np.mean(male_sample_means)
```

Out[12]:

```
9437.830646666669
```

In [13]:

```
np.mean(female_sample_means)
```

Out[13]:

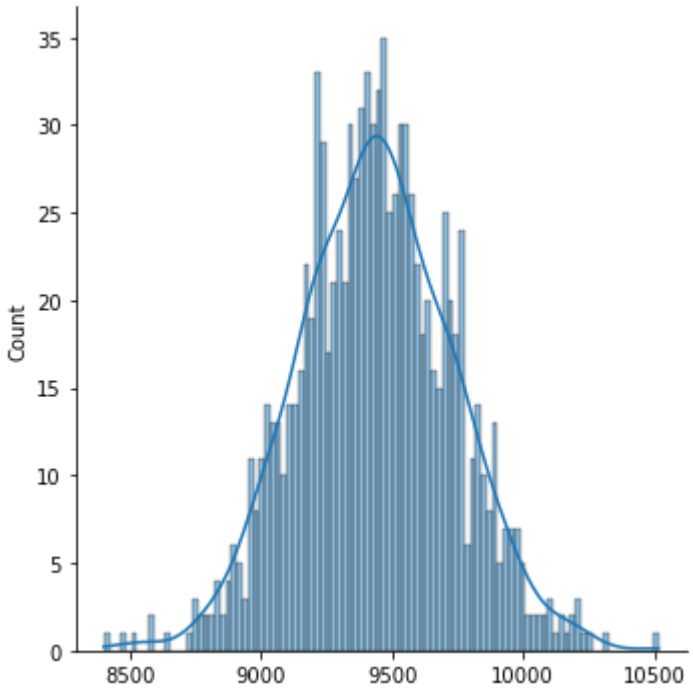
```
8748.099113333334
```

In [14]:

```
sbn.displot(male_sample_means, bins=100, kde=True)
```

Out[14]:

```
<seaborn.axisgrid.FacetGrid at 0x7fef38020760>
```

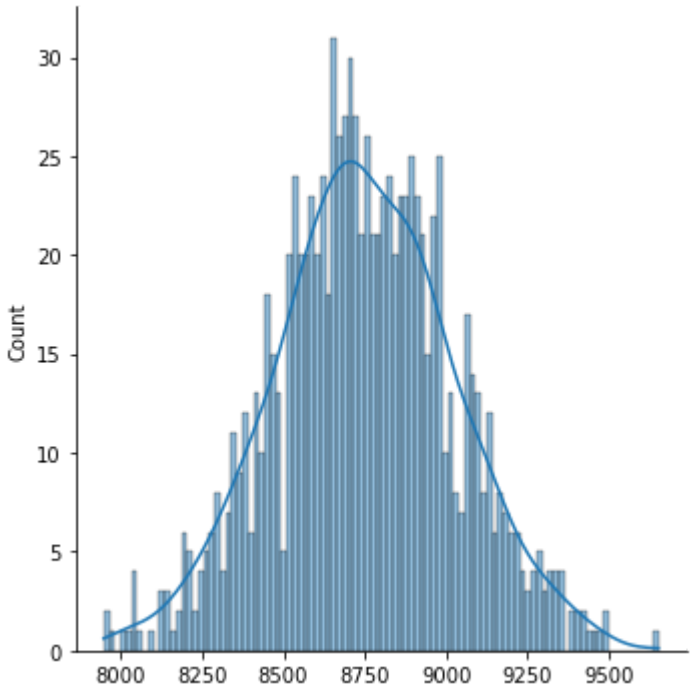


In [15]:

```
sbn.displot(female_sample_means, bins=100, kde=True)
```

Out[15]:

```
<seaborn.axisgrid.FacetGrid at 0x7fef3806f070>
```



In [16]:

```
#Confidence intervals for males and females
```

In [17]:

```
np.mean(male_sample_means), np.std(male_sample_means)
```

Out[17]:

```
(9437.830646666669, 293.1319550842886)
```

In [18]:

```
np.mean(female_sample_means), np.std(female_sample_means)
```

Out[18]:

```
(8748.099113333334, 274.81914326531273)
```

In [19]:

```
#what will be the Z score if the confidence interval is 95% and it is a one sided ta
#what will be the Z score if the confidence interval is 95% and it is a two sided ta
```

In [31]:

```
Lower_limit_males= np.mean(male_sample_means)- (np.std(male_sample_means)/np.sqrt(10
Upper_limit_males= np.mean(male_sample_means)+ (np.std(male_sample_means)/np.sqrt(10
```

In [32]:

```
Lower_limit_females= np.mean(female_sample_means)- (np.std(female_sample_means)/np.s
Upper_limit_females= np.mean(female_sample_means)+ (np.std(female_sample_means)/np.s
```

In [ ]:

```
#should we divide by np.sqrt(1000) or should we not divide by np.sqrt(1000)
```

In [33]:

```
(Lower_limit_males,Upper_limit_males)
```

Out[33]:

```
(9425.686537107593, 9464.06037622574)
```

In [34]:

```
Lower_limit_females,Upper_limit_females
```

Out[34]:

```
(8730.410335386987, 8764.22300461301)
```

In [ ]:

```
#with 95% confidence we are able to find the confidence interval of purchases done b
```

Result: Spending of males (for pop) is greater than spending of females

Recommendations:

In [ ]:

```
#marries and unmarried
```

In [ ]:

```
#age bins
```

In [20]:

```
Lower_limit_males= np.mean(male_sample_means)- (np.std(male_sample_means))*1.96
Upper_limit_males= np.mean(male_sample_means)+ (np.std(male_sample_means))*1.96
```

In [21]:

```
(Lower_limit_males,Upper_limit_males)
```

Out[21]:

```
(8863.292014701463, 10012.369278631875)
```

In [23]:

```
Lower_limit_females= np.mean(female_sample_means)- (np.std(female_sample_means))*1.9
Upper_limit_females= np.mean(female_sample_means)+ (np.std(female_sample_means))*1.9
```

In [24]:

```
Lower_limit_females,Upper_limit_females
```

Out[24]:

```
(8209.45359253332, 9286.744634133347)
```

In [25]:

```
# Get the corresponding values of 2.5th and 97.5th percentiles
conf_interval = np.percentile(male_sample_means,[2.5,97.5])

# Print the interval
print("The confidence interval: ",conf_interval)
```

```
The confidence interval:  [8881.55875    9999.80791667]
```

In [26]:

```
# Get the corresponding values of 2.5th and 97.5th percentiles
conf_interval = np.percentile(female_sample_means,[2.5,97.5])

# Print the interval
print("The confidence interval: ",conf_interval)
```

```
The confidence interval:  [8201.90858333 9303.368     ]
```

In [ ]:

```

```