In [87]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [88]:
```
df=pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094")
```

In [89]:
```
df.head()
```

Out[89]:

	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 [90]:
```
df.shape
```

Out[90]:
```
(550068, 10)
```

In [91]:
```
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 [92]:
```
df.describe()
```

Out[92]:

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

In [93]:
```
df.isna().sum()
```

Out[93]:
```
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 [94]:
```
df.dtypes
```

Out[94]:
```
User_ID                        int64
Product_ID                    object
Gender                        object
Age                           object
Occupation                     int64
City_Category                 object
Stay_In_Current_City_Years    object
Marital_Status                 int64
Product_Category               int64
Purchase                       int64
dtype: object
```

In [95]:
```
#changing data types to category
for i in df.columns[:-1]:
    df[i]=df[i].astype("category")
df.dtypes
```

Out[95]:
```
User_ID                       category
Product_ID                    category
Gender                        category
Age                           category
Occupation                    category
City_Category                 category
Stay_In_Current_City_Years    category
Marital_Status                category
Product_Category              category
Purchase                         int64
dtype: object
```

In [96]:
```
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  category
 1   Product_ID                  550068 non-null  category
 2   Gender                      550068 non-null  category
 3   Age                         550068 non-null  category
 4   Occupation                  550068 non-null  category
 5   City_Category               550068 non-null  category
 6   Stay_In_Current_City_Years  550068 non-null  category
 7   Marital_Status              550068 non-null  category
 8   Product_Category            550068 non-null  category
 9   Purchase                    550068 non-null  int64   
dtypes: category(9), int64(1)
memory usage: 10.3 MB
```

In [97]:
```
df["User_ID"].nunique()
```

Out[97]: 5891

In [98]:
```
df["Product_ID"].nunique()
```

Out[98]: 3631

In [99]:
```
df["Gender"].nunique()
```

Out[99]: 2

In [100]:
```
df["Age"].nunique()
```
Out[100]:
```
7
```

In [101]:
```
df["Occupation"].nunique()
```
Out[101]:
```
21
```

In [102]:
```
df["City_Category"].nunique()
```
Out[102]:
```
3
```

In [103]:
```
df["Stay_In_Current_City_Years"].nunique()
```
Out[103]:
```
5
```

In [104]:
```
df["Marital_Status"].nunique()
```
Out[104]:
```
2
```

In [105]:
```
df["Product_Category"].nunique()
```
Out[105]:
```
20
```

In [106]:
```
df["Purchase"].nunique()
```
Out[106]:
```
18105
```

In [107]:
```
df["User_ID"].value_counts()
```
Out[107]:
```
1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
           ... 
1002111       7
1005391       7
1002690       7
1005608       7
1000708       6
Name: User_ID, Length: 5891, dtype: int64
```

In [108]:
```
df["Product_ID"].value_counts()
```

Out[108]:
```
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
             ...
P00068742       1
P00012342       1
P00162742       1
P00091742       1
P00231642       1
Name: Product_ID, Length: 3631, dtype: int64
```

In [109]:
```
df["Gender"].value_counts()
```

Out[109]:
```
M    414259
F    135809
Name: Gender, dtype: int64
```

In [110]:
```
df["Age"].value_counts()
```

Out[110]:
```
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 [111]:
```
df["Occupation"].value_counts()
```

Out[111]:
```
4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: Occupation, dtype: int64
```

In [112]:
```
df["City_Category"].value_counts()
```

Out[112]:
```
B    231173
C    171175
A    147720
Name: City_Category, dtype: int64
```

In [113]:
```
df["Stay_In_Current_City_Years"].value_counts()
```

Out[113]:
```
1     193821
2     101838
3      95285
4+     84726
0      74398
Name: Stay_In_Current_City_Years, dtype: int64
```

In [114]:
```
df["Marital_Status"].value_counts()
```

Out[114]:
```
0    324731
1    225337
Name: Marital_Status, dtype: int64
```

In [115]:
```
df["Marital_Status"]=df["Marital_Status"].replace({0:"unmarried",1:"married"})
```

In [116]:
```
df["Marital_Status"].unique()
```

Out[116]:
```
['unmarried', 'married']
Categories (2, object): ['unmarried', 'married']
```

In [117]:
```
df["Marital_Status"].value_counts()
```

Out[117]:
```
unmarried    324731
married      225337
Name: Marital_Status, dtype: int64
```

In [118]:
```
df["Product_Category"].value_counts()
```

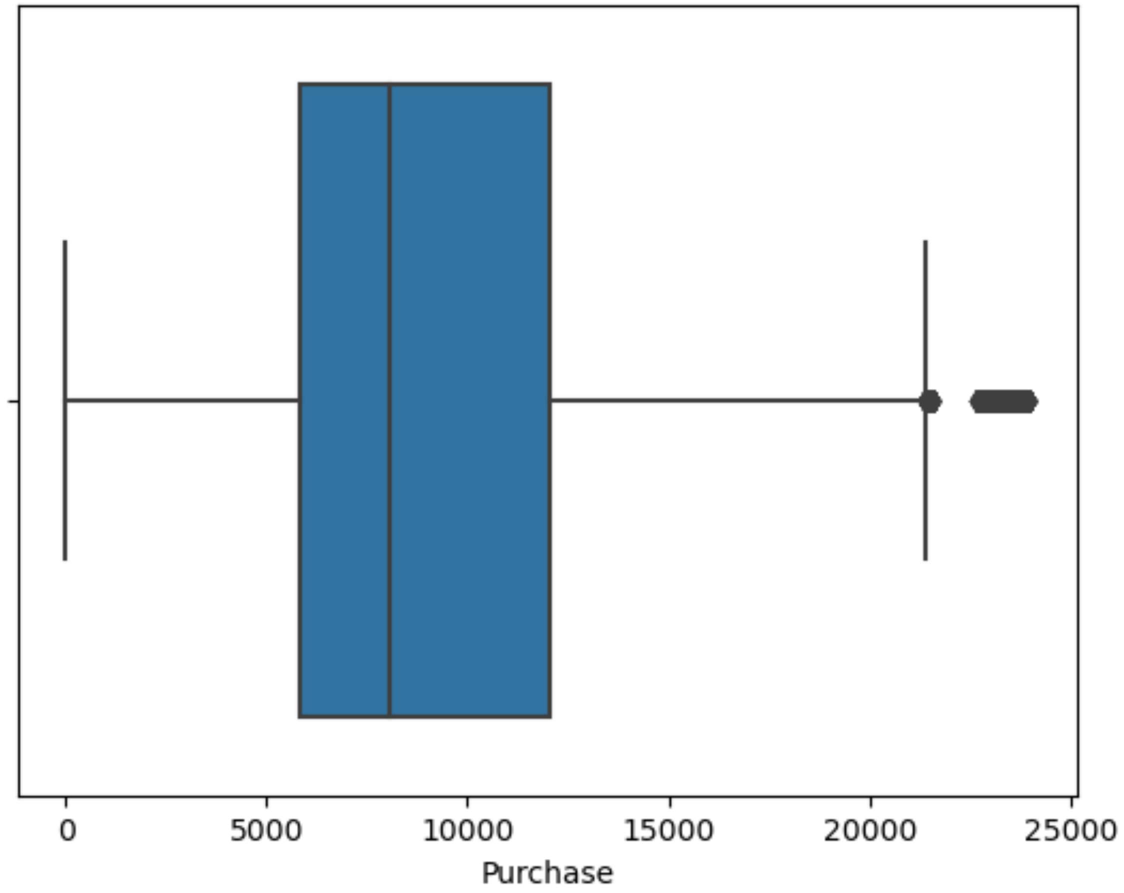
Out[118]:
```
5     150933
1     140378
8     113925
11     24287
2      23864
6      20466
3      20213
4      11753
16      9828
15      6290
13      5549
10      5125
12      3947
7       3721
18      3125
20      2550
19      1603
14      1523
17       578
9        410
Name: Product_Category, dtype: int64
```

In [119]:
```
df["Purchase"].value_counts()
```

Out[119]:
```
7011     191
7193     188
6855     187
6891     184
7012     183
        ...
23491      1
18345      1
3372       1
855        1
21489      1
Name: Purchase, Length: 18105, dtype: int64
```

In [124]:
```
sns.boxplot(x = "Purchase",data = df)
```

Out[124]: <Axes: xlabel='Purchase'>



In [121]:
```
df.head()
```

Out[121]:

	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	unmarried	3	8370
1	1000001	P00248942	F	0-17	10	A	2	unmarried	1	15200
2	1000001	P00087842	F	0-17	10	A	2	unmarried	12	1422
3	1000001	P00085442	F	0-17	10	A	2	unmarried	12	1057
4	1000002	P00285442	M	55+	16	C	4+	unmarried	8	7969

In [122]:
```
"""
Above data has no null values.
From the box plot we can assume purchase categeory might have outliers in it.
"""
```

Out[122]: '\nAbove data has no null values.\nFrom the box plot we can assume purchase categeory might have outliers in it.\n'

In [ ]:

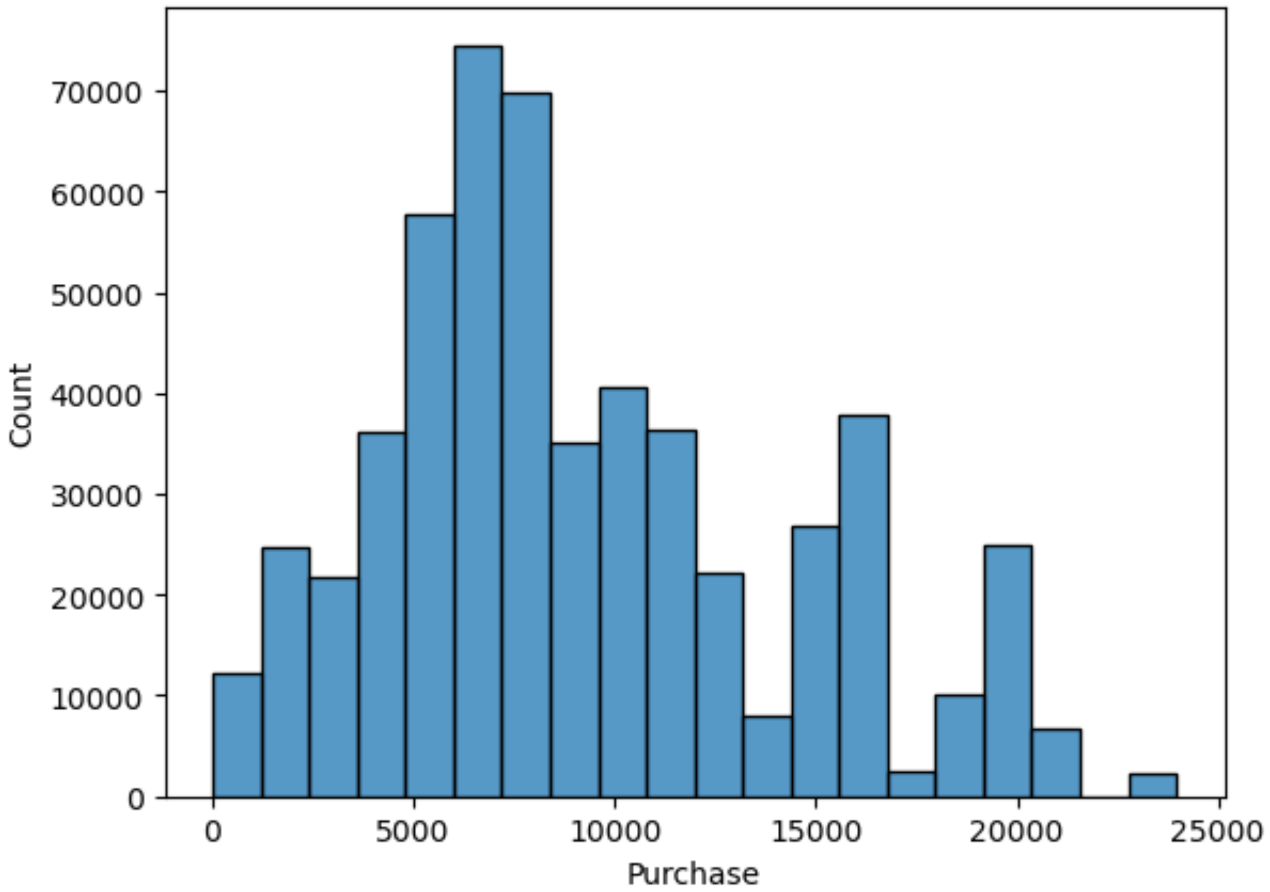
```
#Univariate Analysis
```

In [125]:

```
sns.histplot(x="Purchase",data=df,bins=20)
```

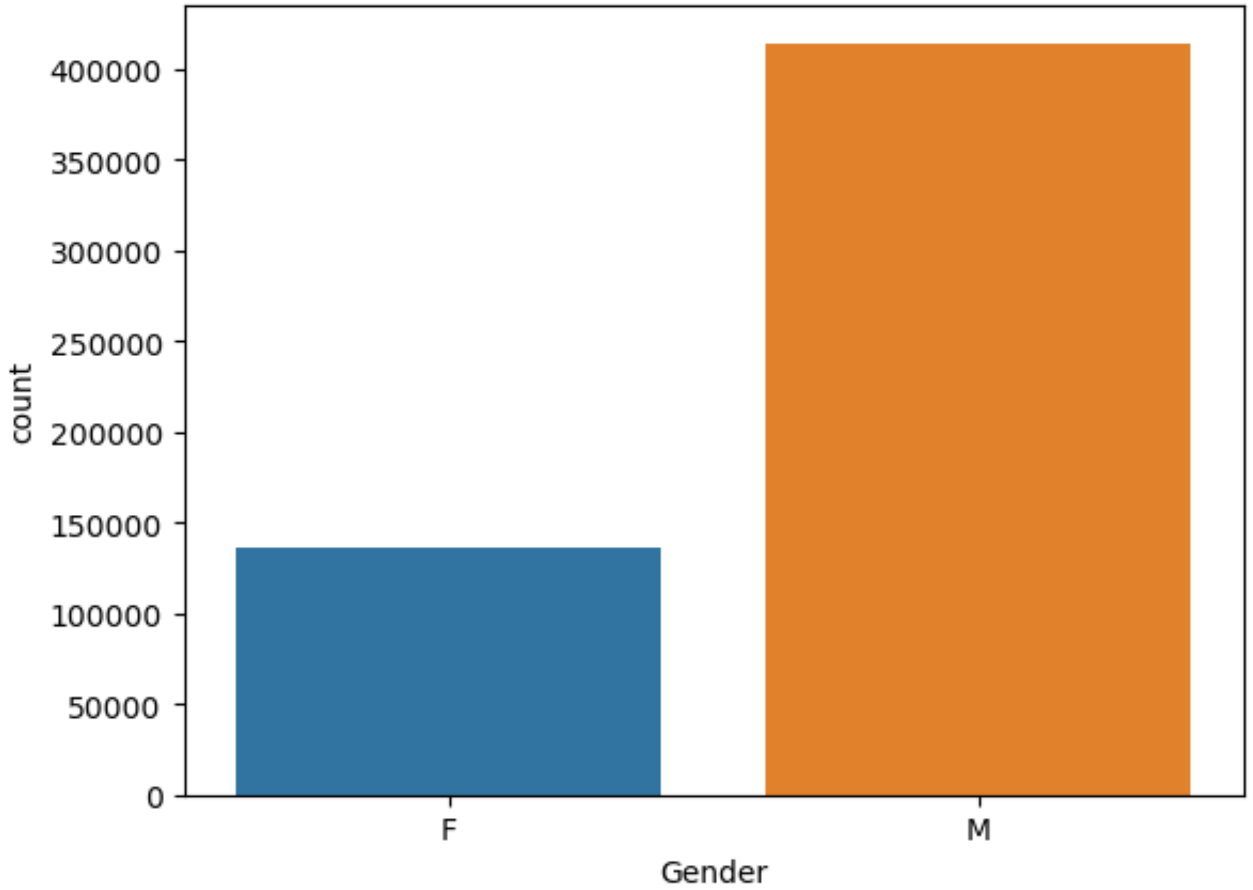
Out[125]:

```
<Axes: xlabel='Purchase', ylabel='Count'>
```



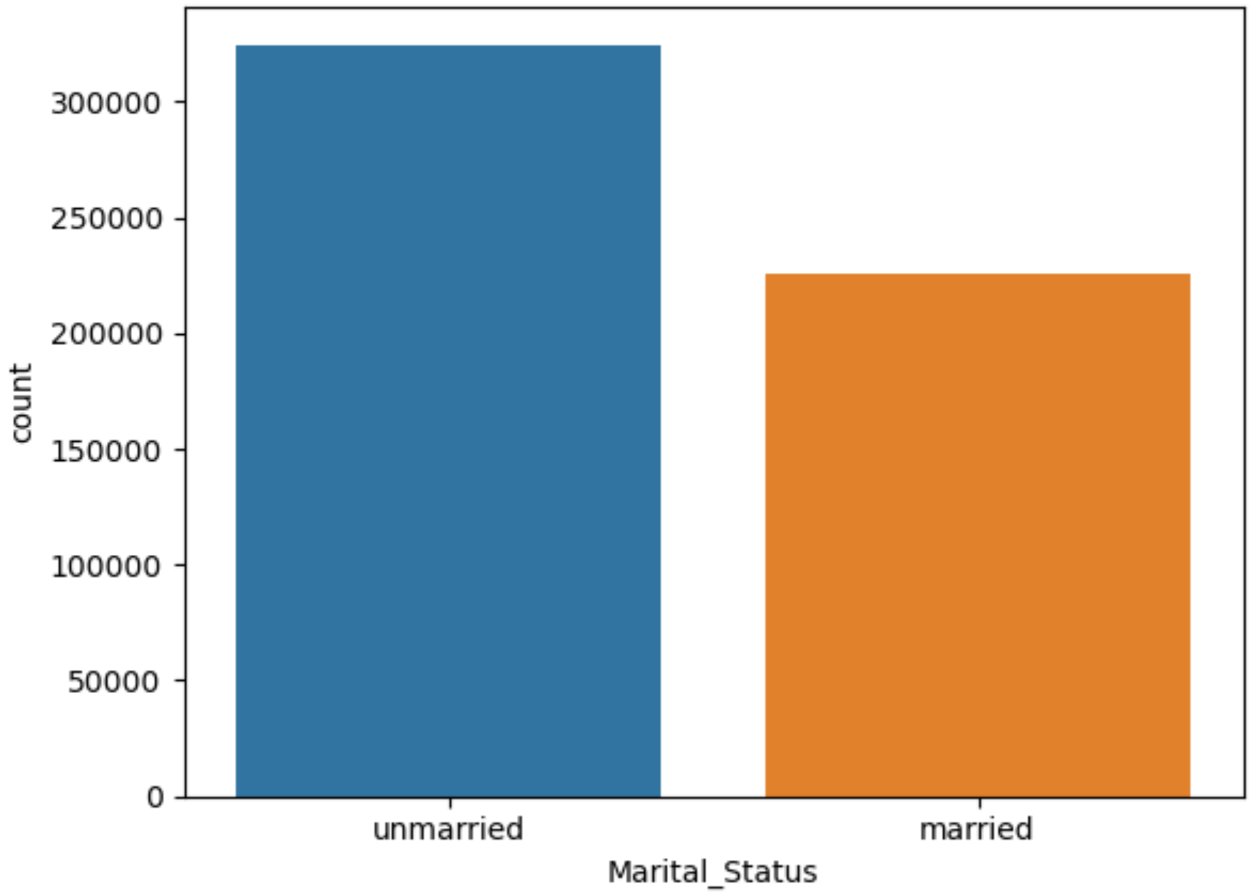
In [127]:
```
sns.countplot(x="Gender",data=df)
```

Out[127]: <Axes: xlabel='Gender', ylabel='count'>



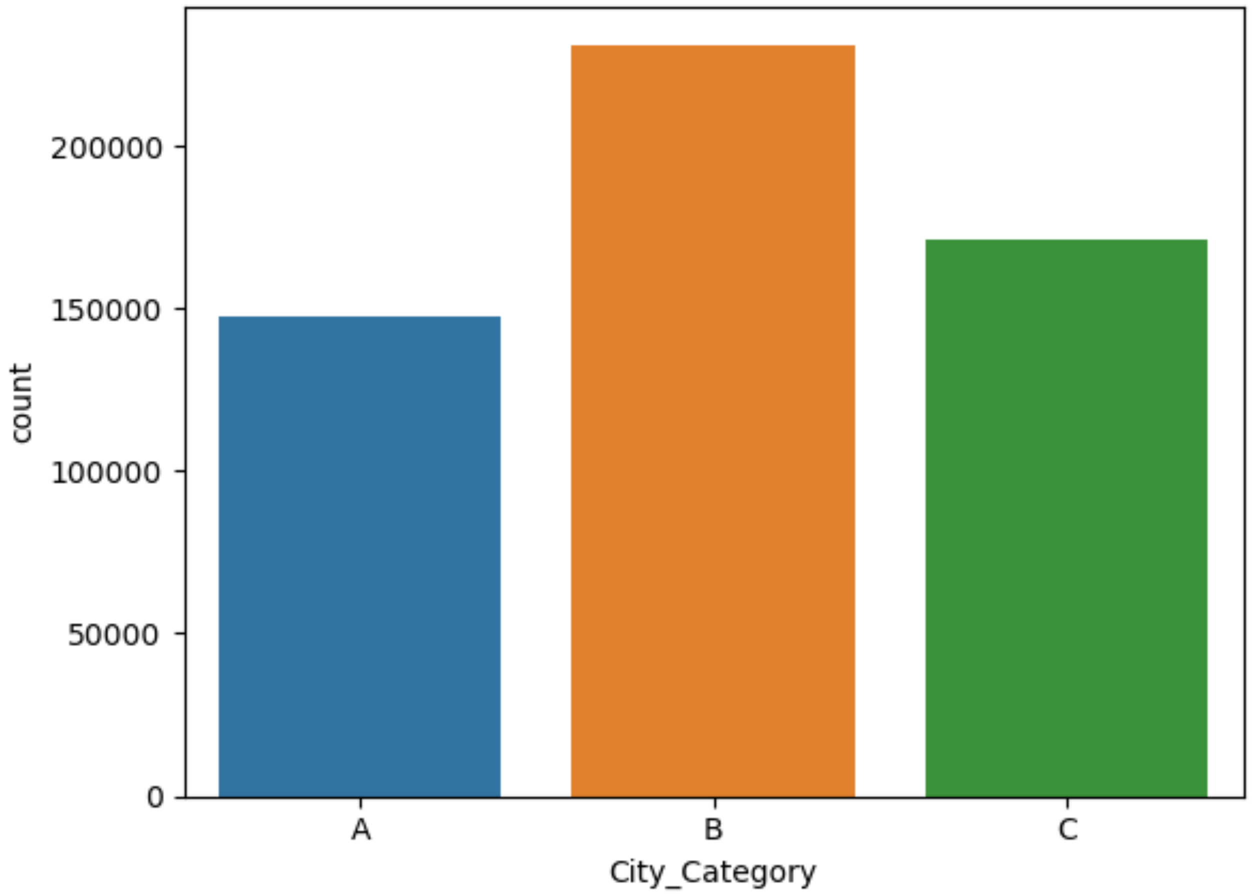
In [129]:
```
sns.countplot(x="Marital_Status",data=df)
```

Out[129]: <Axes: xlabel='Marital_Status', ylabel='count'>



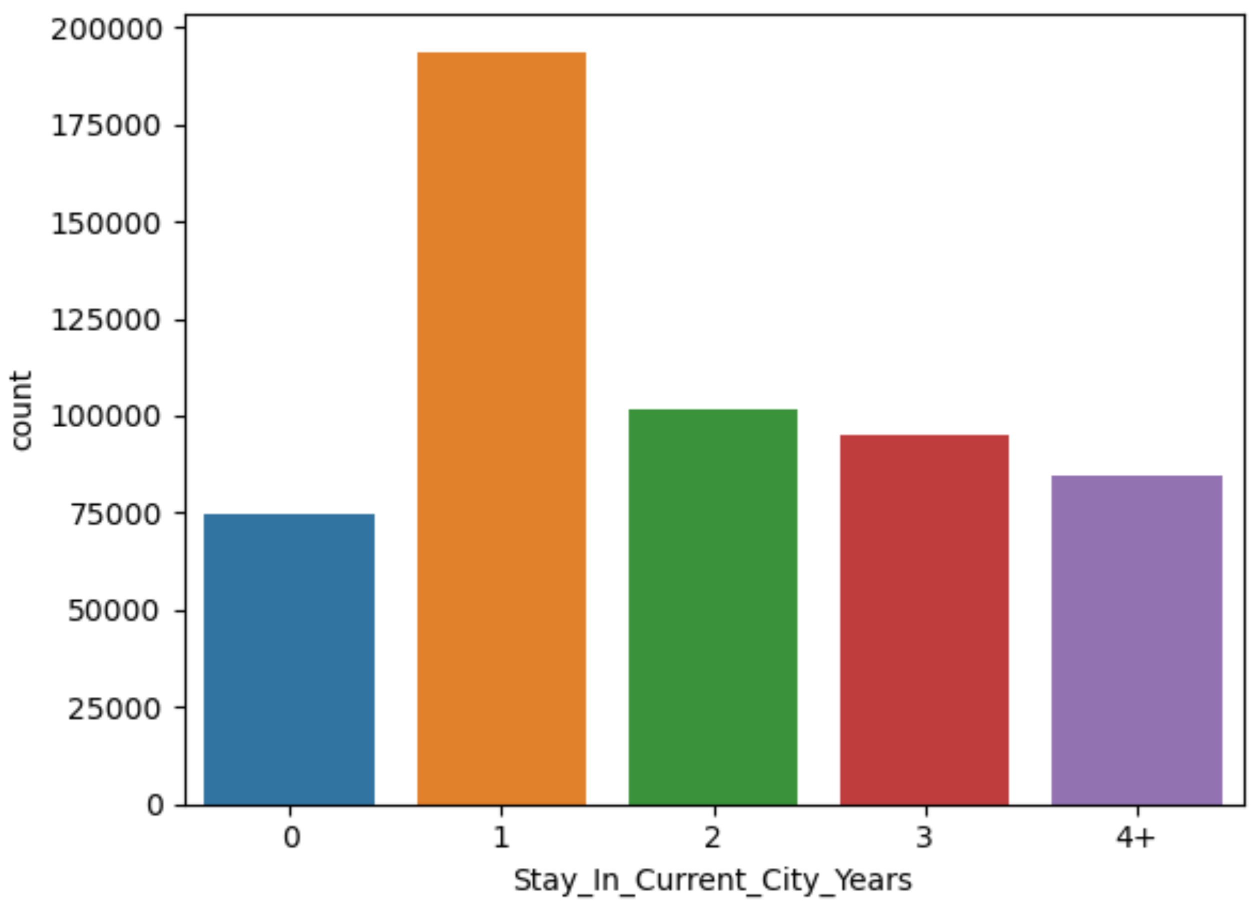
In [130]:
```
sns.countplot(x="City_Category",data=df)
```

Out[130]:
```
<Axes: xlabel='City_Category', ylabel='count'>
```



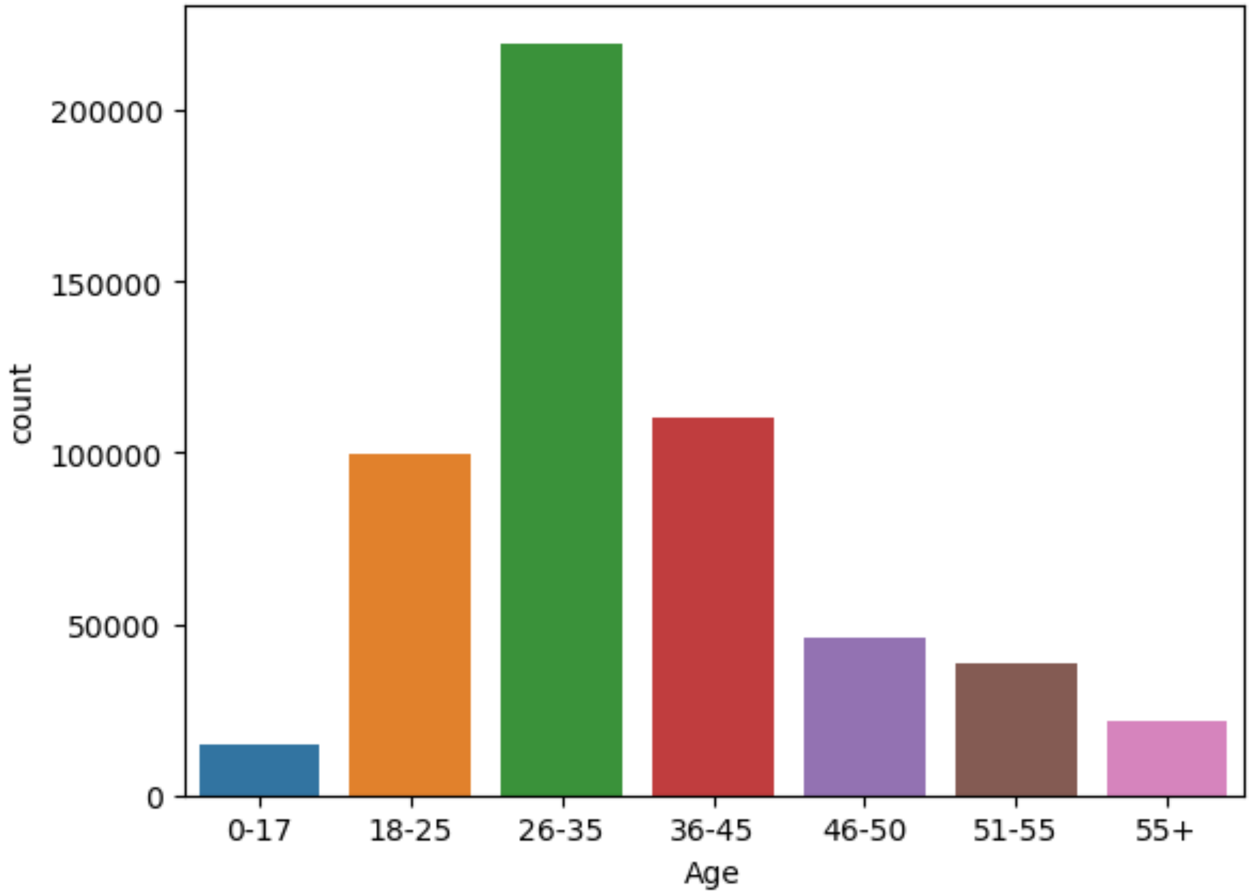
In [131]:
```
sns.countplot(x="Stay_In_Current_City_Years",data=df)
```

Out[131]:
```
<Axes: xlabel='Stay_In_Current_City_Years', ylabel='count'>
```



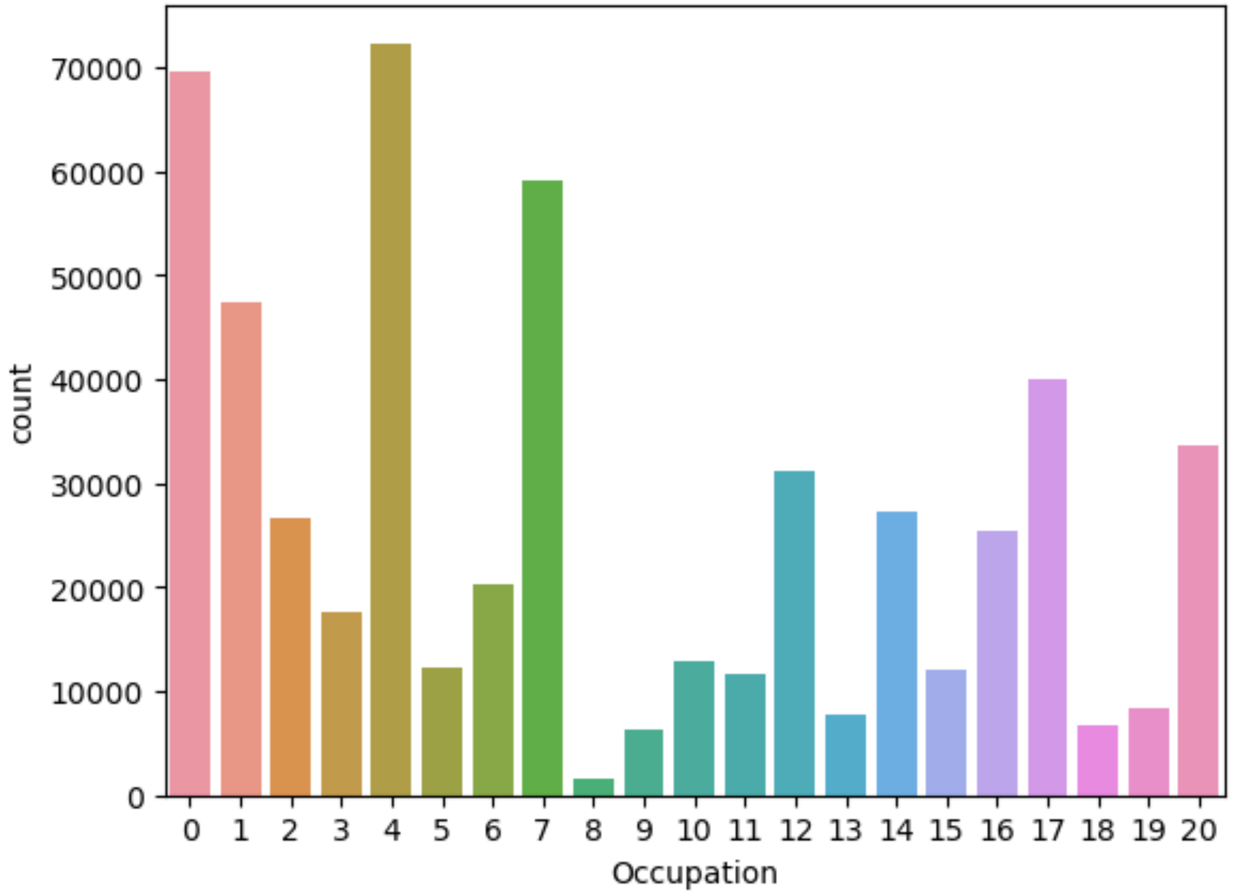
In [132]:
```
sns.countplot(x="Age",data=df)
```

Out[132]: <Axes: xlabel='Age', ylabel='count'>



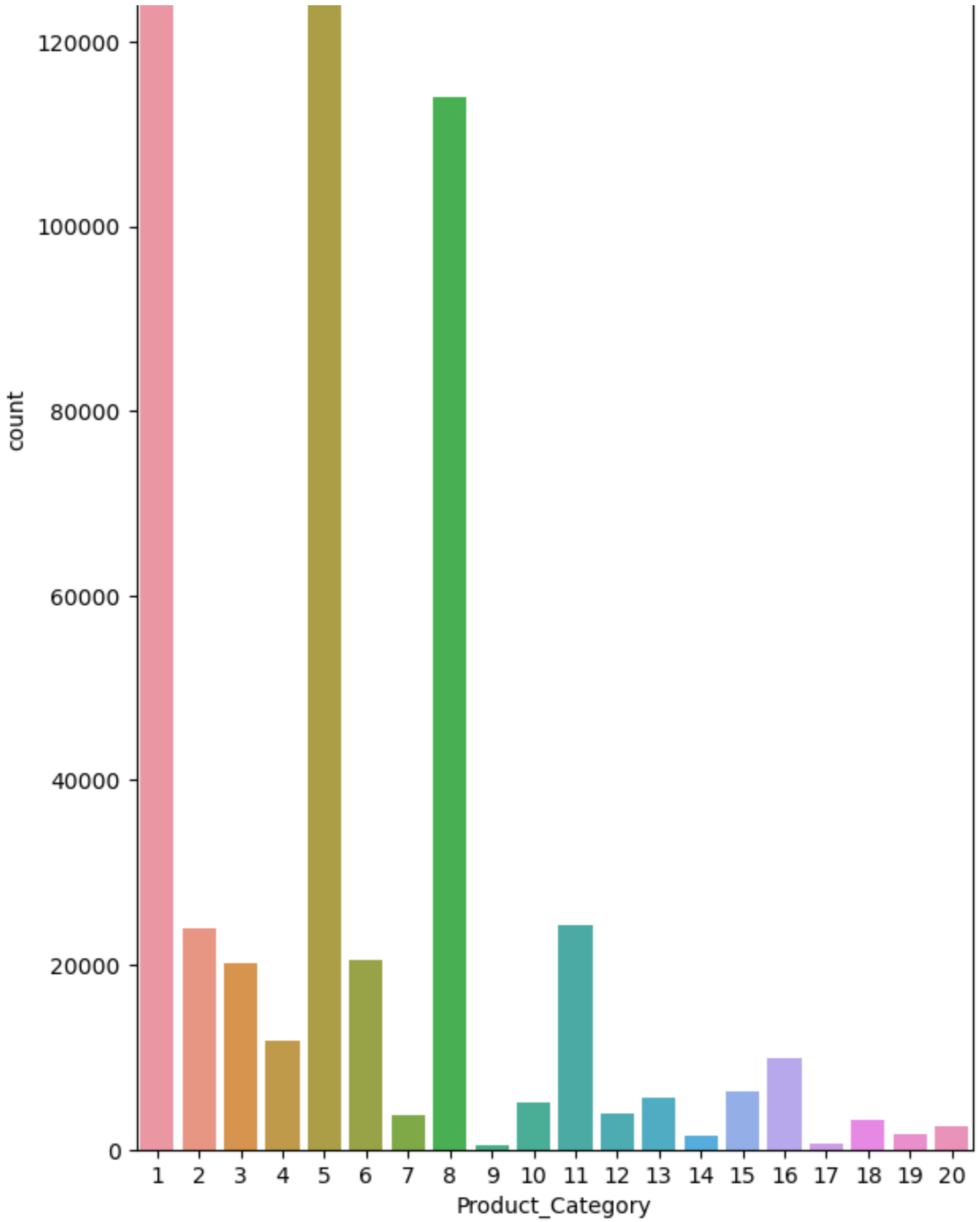
In [133]:
```
sns.countplot(x="Occupation",data=df)
```

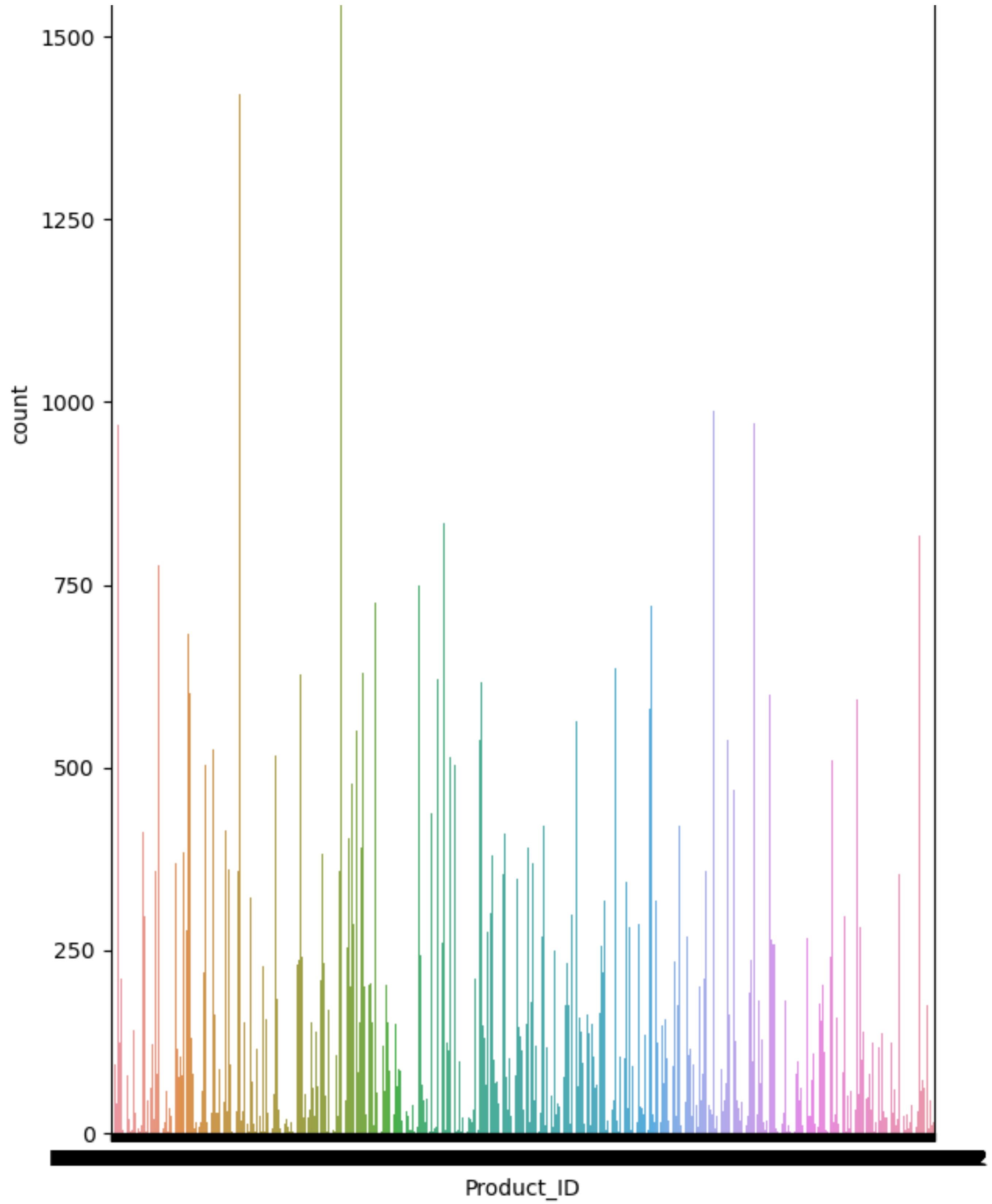
Out[133]: <Axes: xlabel='Occupation', ylabel='count'>



In [148]:
```
fig = plt.figure(figsize=(15,12))

plt.subplot(1,2,1)
sns.countplot(x="Product_Category", data=df)

plt.subplot(1,2,2)
sns.countplot(x="Product_ID", data=df)
```

Out[148]: <Axes: xlabel='Product_ID', ylabel='count'>


count
120000
100000
80000
60000
40000
20000
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Product_Category


count
1500
1250
1000
750
500
250
0
Product_ID

In [ ]:
```
"""
Business Insights:
There is a significant difference between male and female customers during the sale.
In marriatal status category unmarried account for higher sales as compared to males.
In city categeory City B account for highest number of sales followed by City C and City A.
Around 50% of customers have stayed in the city for less than one year,which means highest sale of products is
by newcomers.
Customers in the age group of 26-35 account for the highest no of sales which means walmart has highest customer
base of middle aged people.
Also customers with occupation category of 4 followed by 0 and 7 accounts for highest numer of sales on
black friday,which means these occupation category have highest demand for walmart products.
Product category 5 followed by 1 and 8 accounts for majority of sales.
"""
```

In [126]:
```
df.head()
```

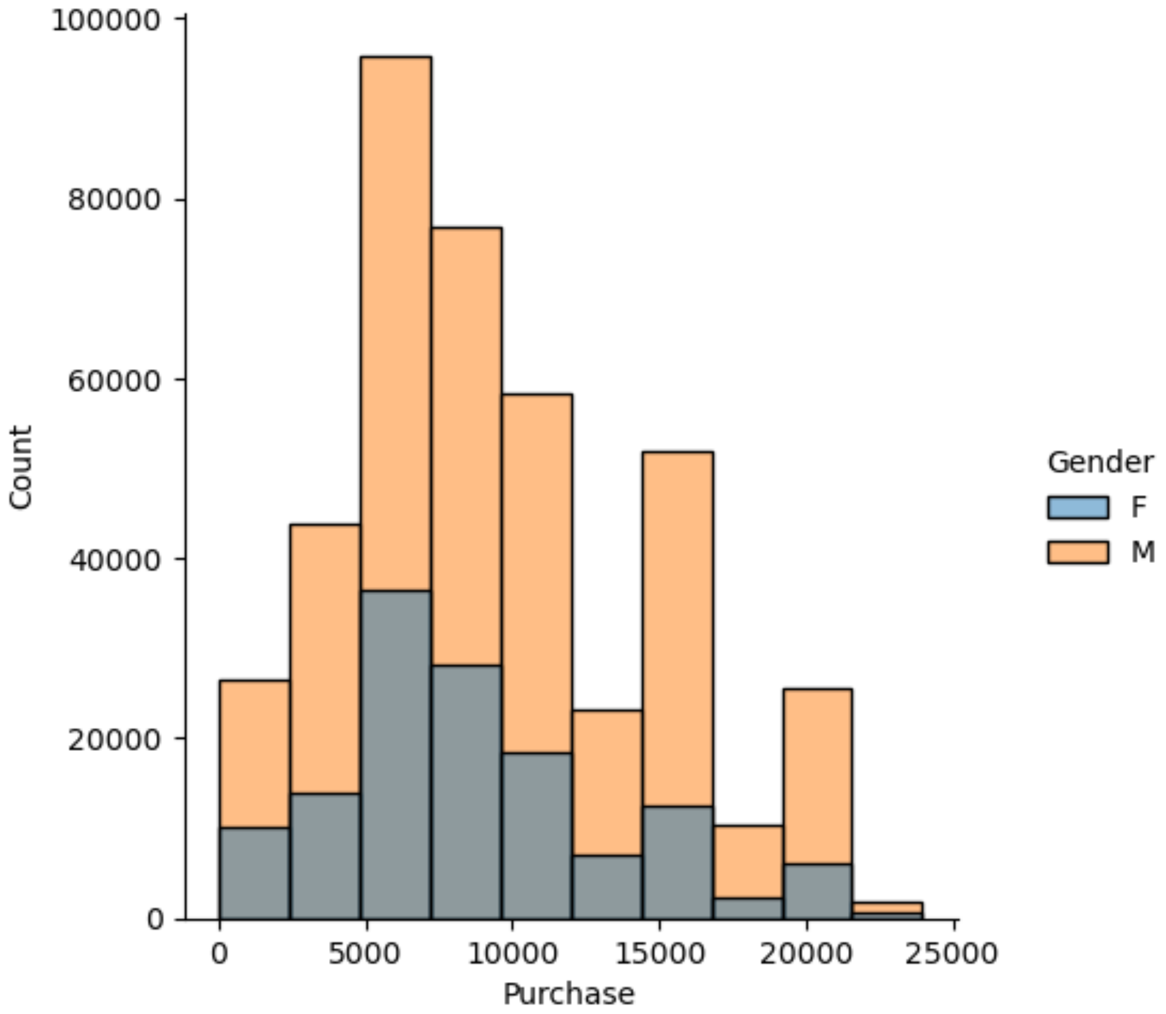
Out[126]:

	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	unmarried	3	8370
1	1000001	P00248942	F	0-17	10	A	2	unmarried	1	15200
2	1000001	P00087842	F	0-17	10	A	2	unmarried	12	1422
3	1000001	P00085442	F	0-17	10	A	2	unmarried	12	1057
4	1000002	P00285442	M	55+	16	C	4+	unmarried	8	7969

In [ ]:
```
#Bi Variate Analysis
```

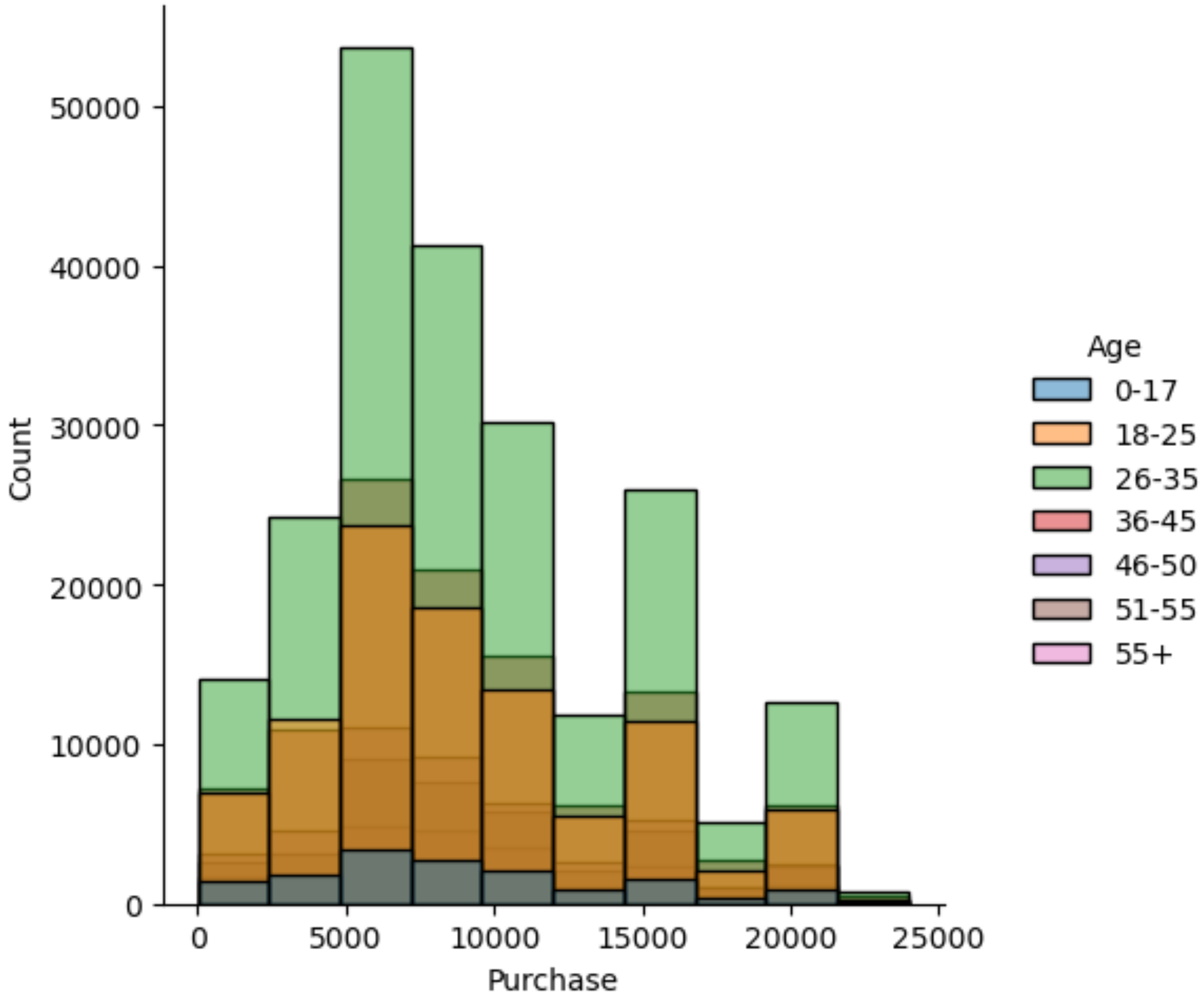
In [246]:
```
sns.displot(x = "Purchase", hue = "Gender", data = df,bins=10)
```

Out[246]: <seaborn.axisgrid.FacetGrid at 0x7f78e6127910>



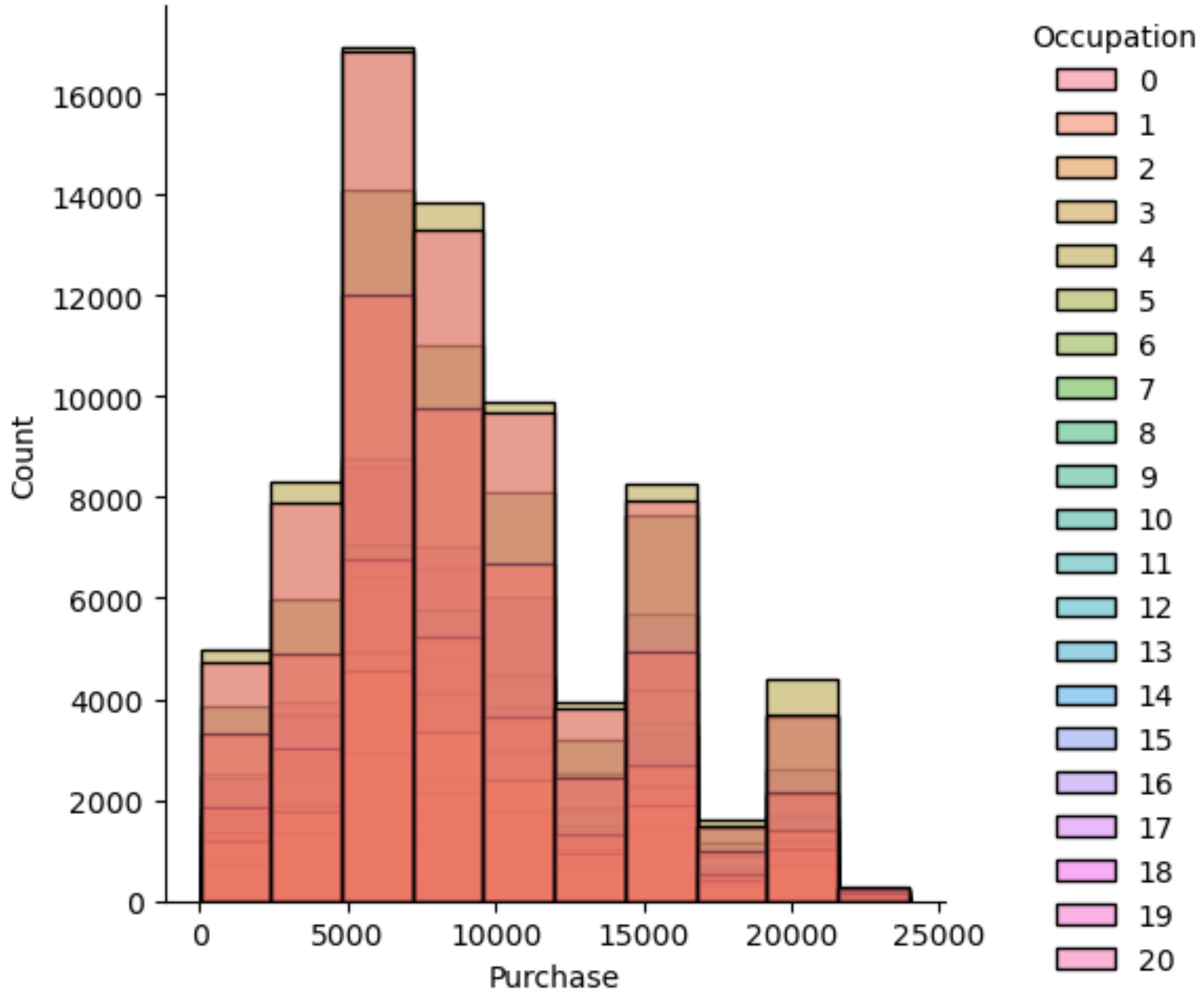
In [157]:
```
sns.displot(x = "Purchase", hue = "Age", data = df,bins=10)
```

Out[157]: <seaborn.axisgrid.FacetGrid at 0x7f791f5c1a50>



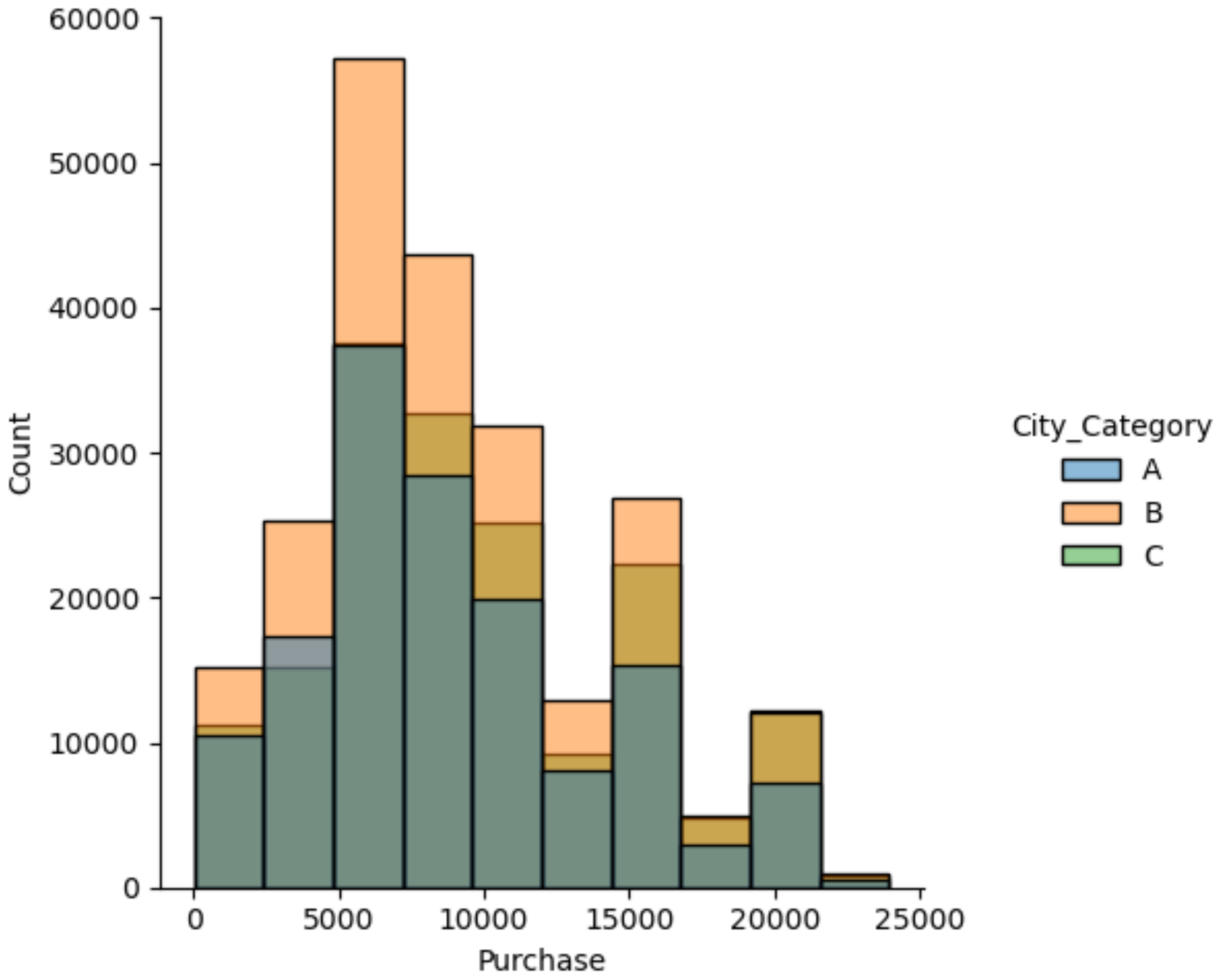
In [158]:
```
sns.displot(x = "Purchase", hue = "Occupation", data = df,bins=10)
```

Out[158]: <seaborn.axisgrid.FacetGrid at 0x7f790a859930>



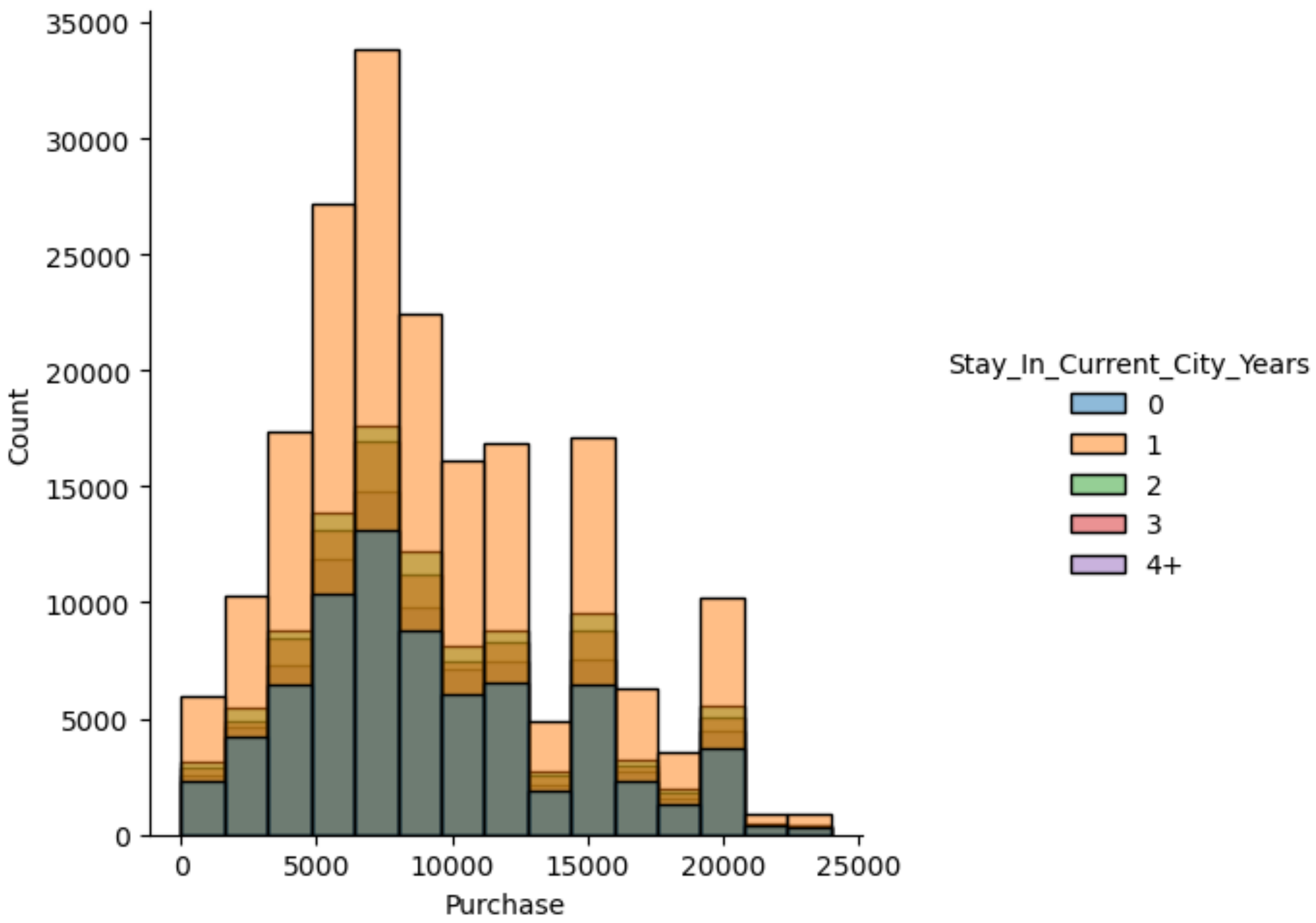
In [160]:
```
sns.displot(x = "Purchase", hue = "City_Category", data = df,bins=10)
```

Out[160]: <seaborn.axisgrid.FacetGrid at 0x7f794cee4220>



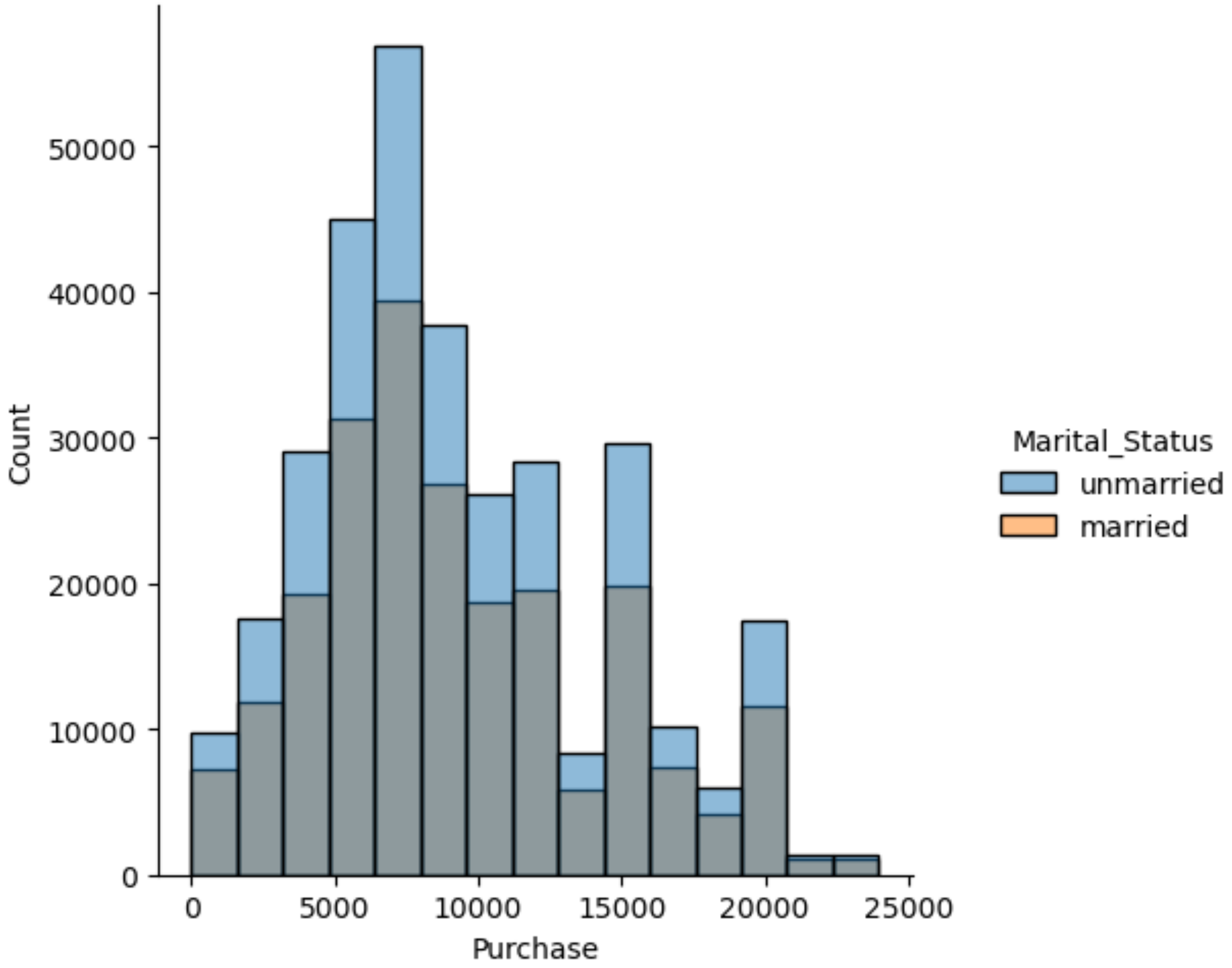
In [161]:
```
sns.displot(x = "Purchase", hue = "Stay_In_Current_City_Years", data = df,bins=15)
```

Out[161]: <seaborn.axisgrid.FacetGrid at 0x7f790bbf5bd0>



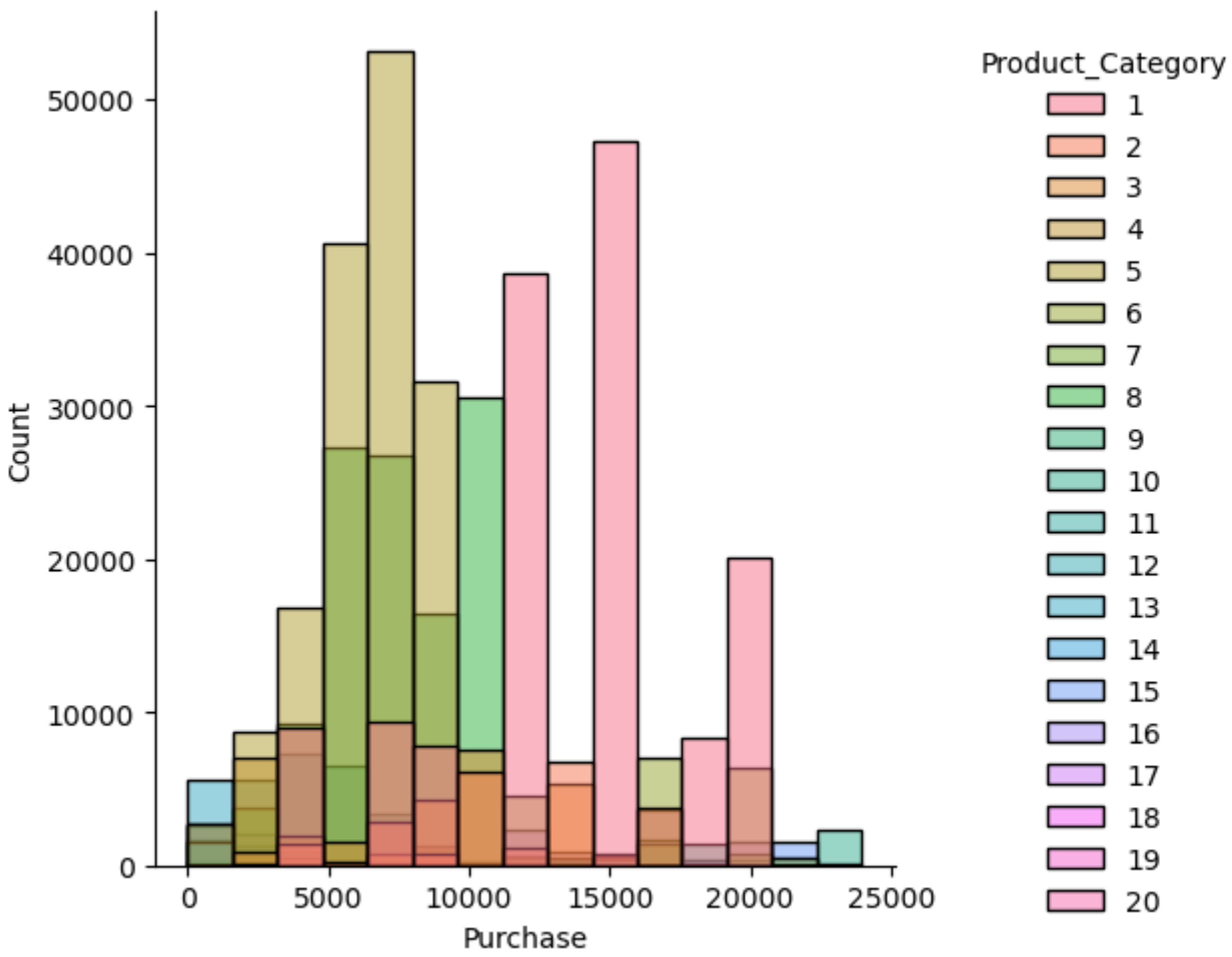
In [164]:
```
sns.displot(x = "Purchase", hue = "Marital_Status", data = df,bins=15)
```

Out[164]: <seaborn.axisgrid.FacetGrid at 0x7f7949c43070>



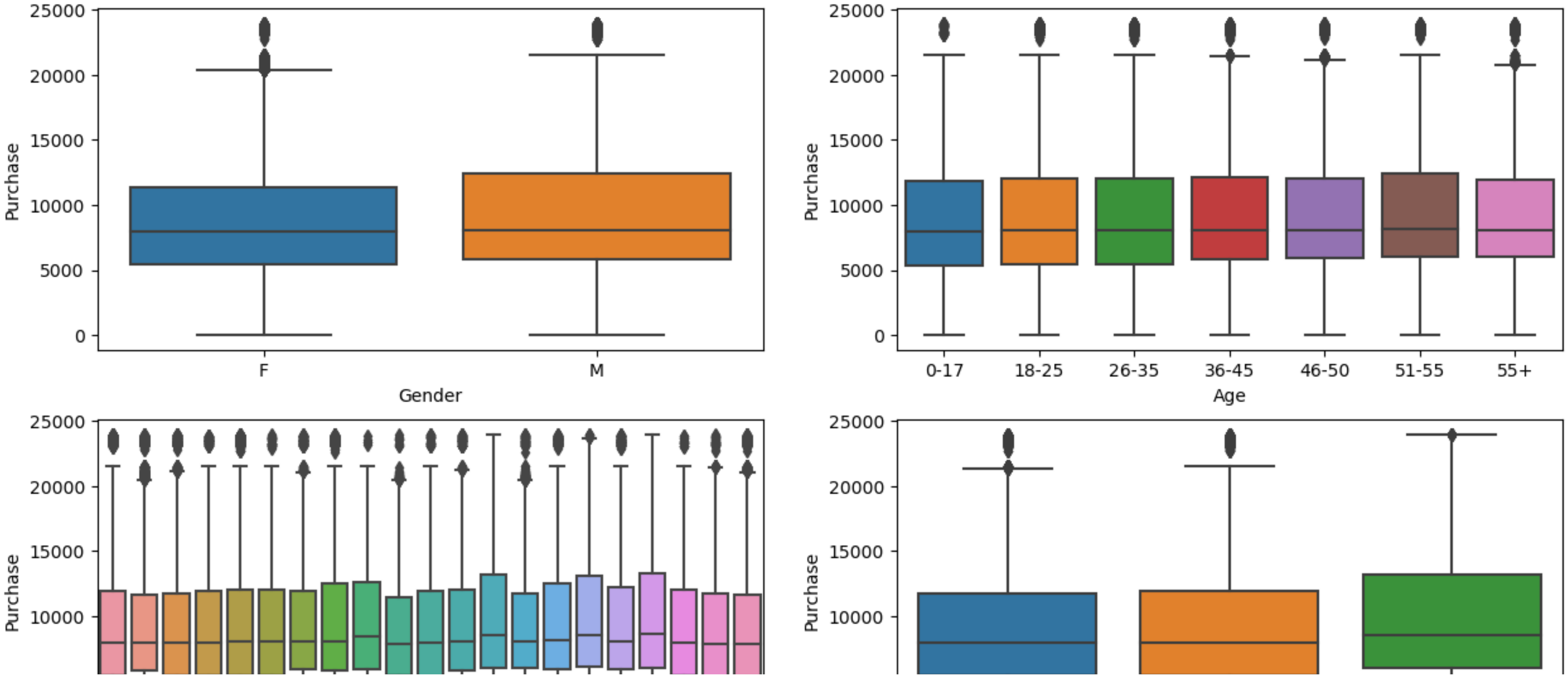
In [165]:
```
sns.displot(x = "Purchase", hue = "Product_Category", data = df,bins=15)
```

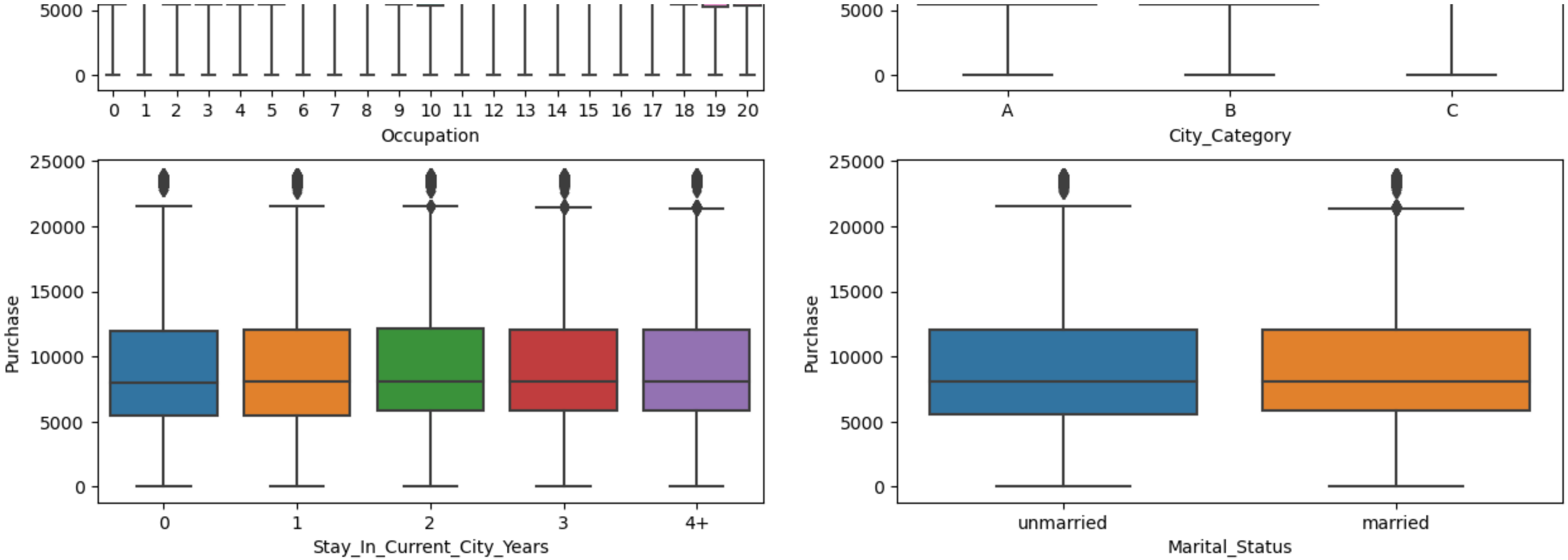
Out[165]: <seaborn.axisgrid.FacetGrid at 0x7f78f52acc40>



In [166]:

```
fig = plt.figure(figsize=(15,12))

plt.subplot(3,2,1)
sns.boxplot(x="Gender",y="Purchase", data=df)

plt.subplot(3,2,2)
sns.boxplot(x="Age", y="Purchase",data=df)

plt.subplot(3,2,3)
sns.boxplot(x="Occupation",y="Purchase", data=df)

plt.subplot(3,2,4)
sns.boxplot(x="City_Category",y="Purchase", data=df)

plt.subplot(3,2,5)
sns.boxplot(x="Stay_In_Current_City_Years",y="Purchase", data=df)

plt.subplot(3,2,6)
sns.boxplot(x="Marital_Status",y="Purchase", data=df)
plt.show()
```





In [ ]:
```
#All box plots are overlapping so we cant directly say about clear winner in any categorical variable.
#so we will now go for confidence interval to tell about clear winner with confidence and significance.
```

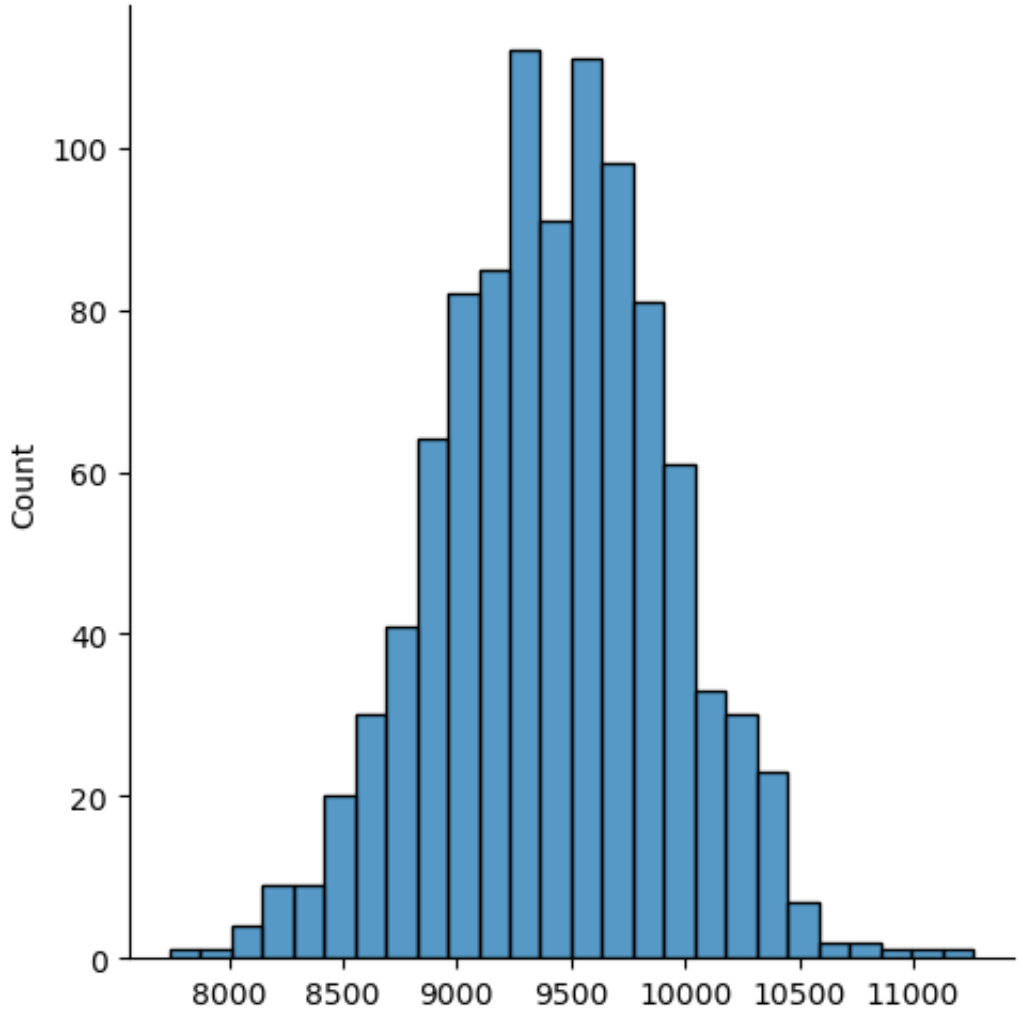
In [167]:
```
#We will use bootstrapping here.
```

In [168]:
```
sample_size = 100
iterations = 1000
male_df = df[df["Gender"] == "M"]
male_sample_df = [ male_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [169]:
```
sample_size = 100
iterations = 1000
female_df = df[df["Gender"] == "F"]
female_sample_df = [ female_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

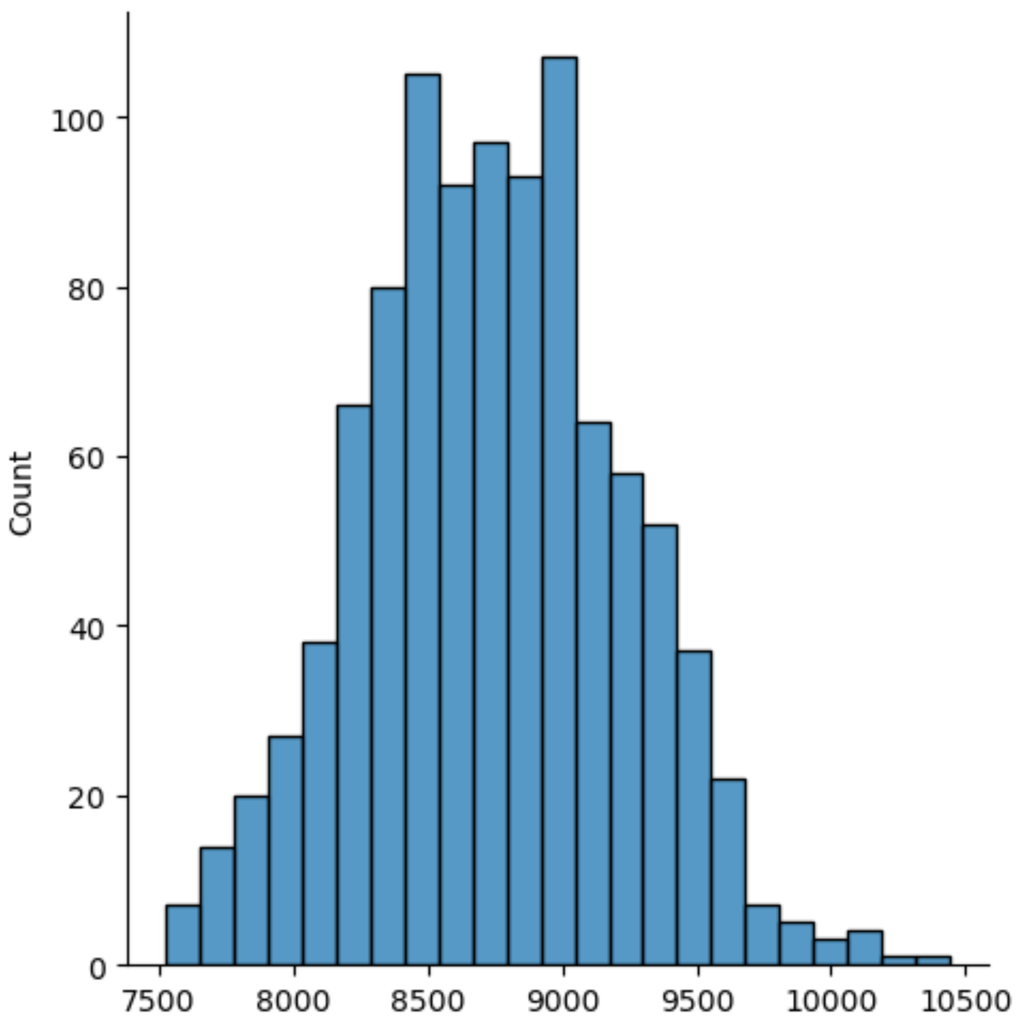
In [170]:
```
sns.displot(male_sample_df)
```

Out[170]: <seaborn.axisgrid.FacetGrid at 0x7f78f9e61420>



In [172]:
```
sns.displot(female_sample_df)
```

Out[172]: <seaborn.axisgrid.FacetGrid at 0x7f78e3f56f50>



In [173]:
```
male_confidence_interval = np.percentile(male_sample_df, [2.5 , 97.5])
female_confidence_interval = np.percentile(female_sample_df, [2.5 , 97.5])
```

In [174]:
```
male_confidence_interval
```

Out[174]: array([8439.103, 10378.646])

In [175]:
```
female_confidence_interval
```

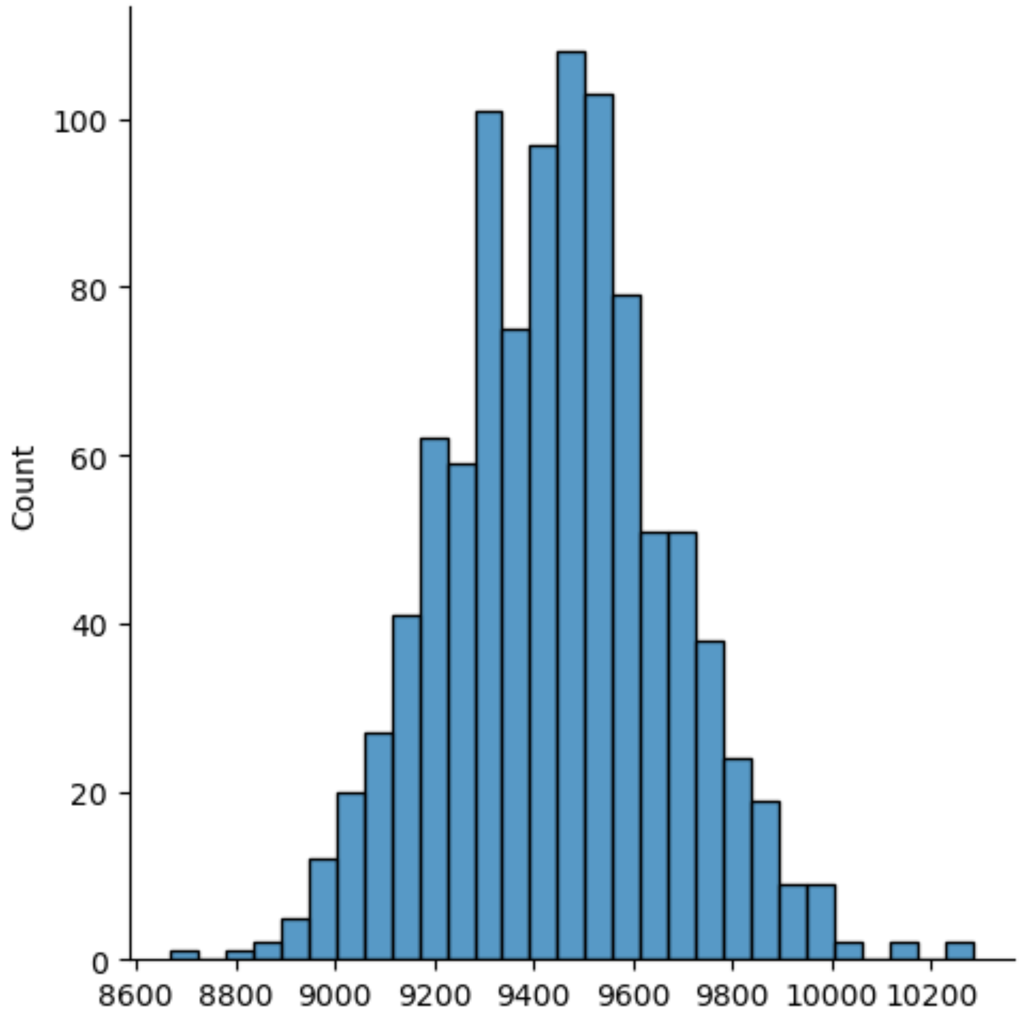
Out[175]: array([7802.43325, 9618.53525])

In [207]:
```
sample_size = 500
iterations = 1000
male_df = df[df["Gender"] == "M"]
male_sample_df = [ male_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [208]:
```
sample_size = 500
iterations = 1000
female_df = df[df["Gender"] == "F"]
female_sample_df = [ female_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

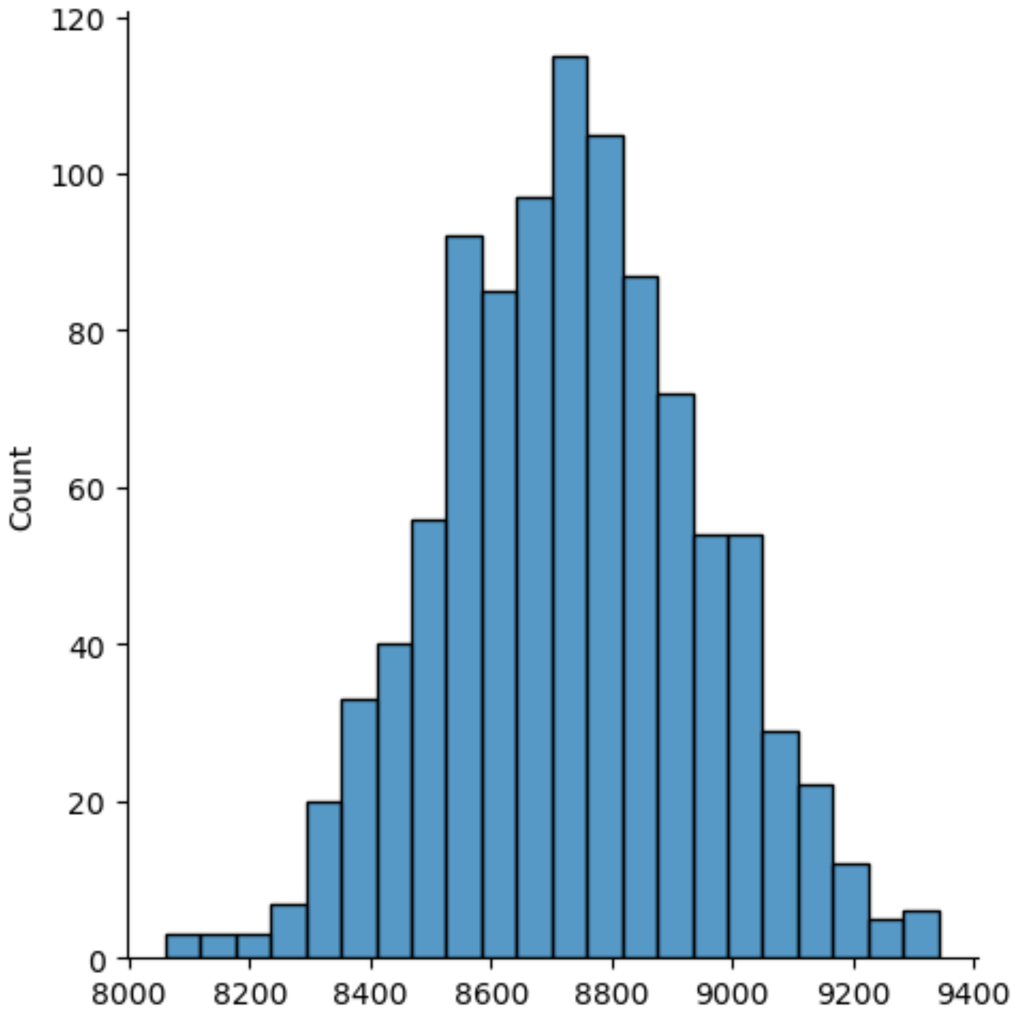
In [211]:
```
sns.displot(male_sample_df)
```

Out[211]: <seaborn.axisgrid.FacetGrid at 0x7f78e614d0f0>



In [210]:
```
sns.displot(female_sample_df)
```

Out[210]: <seaborn.axisgrid.FacetGrid at 0x7f78e4deb9a0>



In [212]:
```
male_confidence_interval = np.percentile(male_sample_df, [2.5 , 97.5])
female_confidence_interval = np.percentile(female_sample_df, [2.5 , 97.5])
```

In [213]:
```
male_confidence_interval
```

Out[213]: array([9024.26585, 9891.7583])

In [214]:
```
female_confidence_interval
```

Out[214]: array([8337.2697 , 9160.58785])

In [216]:
```
sample_size = 1000
iterations = 1000
male_df = df[df["Gender"] == "M"]
male_sample_df = [ male_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [220]:
```
sample_size = 1000
iterations = 1000
female_df = df[df["Gender"] == "F"]
female_sample_df = [ female_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [221]:
```
male_confidence_interval = np.percentile(male_sample_df, [2.5 , 97.5])
female_confidence_interval = np.percentile(female_sample_df, [2.5 , 97.5])
```

In [222]:
```
male_confidence_interval
```

Out[222]:
```
array([9336.9207225, 9535.7513675])
```

In [223]:
```
female_confidence_interval
```

Out[223]:
```
array([8427.67125, 9039.58415])
```

In [224]:
```
sample_size = 10000
iterations = 1000
male_df = df[df["Gender"] == "M"]
male_sample_df = [ male_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [225]:
```
sample_size = 10000
iterations = 1000
female_df = df[df["Gender"] == "F"]
female_sample_df = [ female_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [226]:
```
male_confidence_interval = np.percentile(male_sample_df, [2.5 , 97.5])
female_confidence_interval = np.percentile(female_sample_df, [2.5 , 97.5])
```

In [227]:
```
male_confidence_interval
```

Out[227]:
```
array([9338.3187825, 9546.4340225])
```

In [228]:
```
female_confidence_interval
```

Out[228]:
```
array([8646.464475, 8827.00131 ])
```

In [ ]:
```
"""
With increase in sample size confidence intervel is getting more and more narrower and we are getting more
accurate data.
With 95% confidence interval mean purchase amount of the male ranges between 9333.8 and 9546.43.
With 95% confidence interval mean purchase amount of the female ranges between 8646.46 and 8827.00.
We can clearly see that both intervals are not overlapping with each other,so we can say female customers
are spending less as compared to male customers.

"""
```

In [229]:

```
#Married vs Unmarried users
```

In [232]:

```
sample_size = 500
iterations = 1000
married_df = df[df["Marital_Status"] == "married"]
married_sample_df = [ married_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [233]:

```
sample_size = 500
iterations = 1000
unmarried_df = df[df["Marital_Status"] == "unmarried"]
unmarried_sample_df = [ unmarried_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [234]:

```
married_confidence_interval = np.percentile(married_sample_df, [2.5 , 97.5])
unmarried_confidence_interval = np.percentile(unmarried_sample_df, [2.5 , 97.5])
```

In [236]:

```
married_confidence_interval
```

Out[236]:

```
array([8812.0044, 9663.3935])
```

In [237]:

```
unmarried_confidence_interval
```

Out[237]:

```
array([8800.51545, 9722.62575])
```

In [230]:

```
df.head()
```

Out[230]:

	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	unmarried	3	8370
1	1000001	P00248942	F	0-17	10	A	2	unmarried	1	15200
2	1000001	P00087842	F	0-17	10	A	2	unmarried	12	1422
3	1000001	P00085442	F	0-17	10	A	2	unmarried	12	1057
4	1000002	P00285442	M	55+	16	C	4+	unmarried	8	7969

In [231]:

```
df["Marital_Status"].value_counts()
```

Out[231]:

```
unmarried    324731
married      225337
Name: Marital_Status, dtype: int64
```

In [238]:
```
sample_size = 10000
iterations = 1000
married_df = df[df["Marital_Status"] == "married"]
married_sample_df = [ married_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [239]:
```
sample_size = 10000
iterations = 1000
unmarried_df = df[df["Marital_Status"] == "unmarried"]
unmarried_sample_df = [ unmarried_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [240]:
```
married_confidence_interval = np.percentile(married_sample_df, [2.5 , 97.5])
unmarried_confidence_interval = np.percentile(unmarried_sample_df, [2.5 , 97.5])
```

In [241]:
```
married_confidence_interval
```

Out[241]:
```
array([9170.9309575, 9359.0792725])
```

In [242]:
```
unmarried_confidence_interval
```

Out[242]:
```
array([9171.0173925, 9361.713885 ])
```

In [243]:
```
"""
With increase in sample size confidence intervel is not having much of effect.
With 95% confidence interval mean purchase amount of the married customers ranges between 9170.93 and 9359.07.
With 95% confidence interval mean purchase amount of the unmarried customers ranges between 9171.01 and 9361.71.
We can clearly see that both intervals are clearly overlapping with each other,so we can say both type of customers
are spending almost equal with unmarried customers spending a bit more as compared to married customers.

"""
```

Out[243]:
```
'\nWith increase in sample size confidence intervel is not having much of effect.\nWith 95% confidence interval mean purchase amount of the married custom
ers ranges between 9170.93 and 9359.07.\nWith 95% confidence interval mean purchase amount of the unmarried customers ranges between 9171.01 and 9361.71.\
nWe can clearly see that both intervals are clearly overlapping with each other,so we can say both type of customers\nare spending almost equal with unmar
ried customers spending a bit more as compared to married customers.\n\n'
```

In [244]:
```
#Age Group
```

In [247]:
```
df["Age"].value_counts()
```

Out[247]:
```
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 [249]:
```
sample_size = 10000
iterations = 1000
a0_17_df = df[df["Age"] == "0-17"]
a0_17_sample_df = [ a0_17_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [250]:
```
sample_size = 10000
iterations = 1000
a18_25_df = df[df["Age"] == "18-25"]
a18_25_sample_df = [ a18_25_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [251]:
```
sample_size = 10000
iterations = 1000
a26_35_df = df[df["Age"] == "26-35"]
a26_35_sample_df = [ a26_35_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [252]:
```
sample_size = 10000
iterations = 1000
a36_45_df = df[df["Age"] == "36-45"]
a36_45_sample_df = [ a36_45_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [253]:
```
sample_size = 10000
iterations = 1000
a46_50_df = df[df["Age"] == "46-50"]
a46_50_sample_df = [ a46_50_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [254]:
```
sample_size = 10000
iterations = 1000
a51_55_df = df[df["Age"] == "51-55"]
a51_55_sample_df = [ a51_55_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [256]:
```
sample_size = 10000
iterations = 1000
a55_df = df[df["Age"] == "55+"]
a55_sample_df = [ a55_df.sample(sample_size , replace = True)["Purchase"].mean() for i in range(iterations)]
```

In [260]:
```
a0_17_confidence_interval = np.percentile(a0_17_sample_df, [2.5 , 97.5])
```

In [261]:
```
a18_25_confidence_interval = np.percentile(a18_25_sample_df, [2.5 , 97.5])
```

In [262]:
```
a26_35_confidence_interval = np.percentile(a26_35_sample_df, [2.5 , 97.5])
```

In [263]:
```
a36_45_confidence_interval = np.percentile(a36_45_sample_df , [2.5 , 97.5])
```

In [264]:
```
a46_50_confidence_interval = np.percentile(a46_50_sample_df, [2.5 , 97.5])
```

In [265]:
```
a51_55_confidence_interval = np.percentile(a51_55_sample_df, [2.5 , 97.5])
```

In [273]:
```
a55_confidence_interval = np.percentile(a55_sample_df, [2.5 , 97.5])
```

In [267]:
```
a0_17_confidence_interval
```

Out[267]:
```
array([8837.2225875, 9032.4972425])
```

In [268]:
```
a18_25_confidence_interval
```

Out[268]:
```
array([9065.0404425, 9272.3409175])
```

In [269]:
```
a26_35_confidence_interval
```

Out[269]:
```
array([9151.43606 , 9347.910045])
```

In [270]:
```
a36_45_confidence_interval
```

Out[270]:
```
array([9242.8121725, 9426.7840225])
```

In [271]:
```
a46_50_confidence_interval
```

Out[271]:
```
array([9113.6262175, 9310.30055  ])
```

In [272]:
```
a51_55_confidence_interval
```

Out[272]:
```
array([9443.0649575, 9633.906115 ])
```

In [274]:
```
a55_confidence_interval
```

Out[274]:
```
array([9243.5747975, 9433.01808  ])
```

In [275]:
```
"""
With 95% confidence interval mean purchase amount of the age group 00-17 ranges between 8837.22 and 9032.49.
With 95% confidence interval mean purchase amount of the age group 18-25 ranges between 9065.04 and 9272.34.
With 95% confidence interval mean purchase amount of the age group 26-35 ranges between 9151.43 and 9347.91.
With 95% confidence interval mean purchase amount of the age group 36-45 ranges between 9242.81 and 9426.78.
With 95% confidence interval mean purchase amount of the age group 46-50 ranges between 9113.62 and 9310.30.
With 95% confidence interval mean purchase amount of the age group 51-55 ranges between 9443.06 and 9633.90.
With 95% confidence interval mean purchase amount of the age group 55+ ranges between 9243.57 and 9433.01.
We can infer that confidence interval of age group 0-17 doesn't overlap with any other age group,so amount
spent by them is least.
Also confidence interval of age group 51-55 doesnt overlap with any other,so amount spent by them is highest.
Rest all other age groups are overlapping so we can't say with 95% confidence about their distinctions.
"""
```

Out[275]:
```
"\nWith 95% confidence interval mean purchase amount of the age group 00-17 ranges between 8837.22 and 9032.49.\nWith 95% confidence interval mean purchas
e amount of the age group 18-25 ranges between 9065.04 and 9272.34.\nWith 95% confidence interval mean purchase amount of the age group 26-35 ranges betwe
en 9151.43 and 9347.91.\nWith 95% confidence interval mean purchase amount of the age group 36-45 ranges between 9242.81 and 9426.78.\nWith 95% confidence
interval mean purchase amount of the age group 46-50 ranges between 9113.62 and 9310.30.\nWith 95% confidence interval mean purchase amount of the age gro
up 51-55 ranges between 9443.06 and 9633.90.\nWith 95% confidence interval mean purchase amount of the age group 55+ ranges between 9243.57 and 9433.01.\n
We can infer that confidence interval of age group 0-17 doesn't overlap with any other age group,so amount\nspent by them is least.\nAlso confidence inter
val of age group 51-55 doesnt overlap with any other,so amount spent by them is highest.\nRest all other age groups are overlapping so we can't say with 9
5% confidence about their distinctions.\n"
```

In [ ]: