EXPERIMENT -4

AIM

To perform EDA on the given data set.

Explanation

The primary aim with exploratory analysis is to examine the data for distribution, outliers and anomalies to direct specific testing of your hypothesis.

ALGORITHM

STEP 1:

Import the required packages(pandas,numpy,seaborn).

STEP 2:

Read the given csv file.

STEP 3:

Convert the file into a dataframe and get information of the data.

STEP 4:

Remove the non numerical data columns using drop() method.

STEP 5:

Replace the null values using (.fillna).

STEP 6:

returns object containing counts of unique values using (value_counts()).

STEP 7:

Plot the counts in the form of Histogram or Bar Graph.

STEP 8:

find the pairwise correlation of all columns in the dataframe(.corr()).

STEP 9:

Save the final data set into the file.

In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
```

In [3]:
```
import seaborn as sns
```

In [5]:
```
df=pd.read_csv("supermarket.csv")
df
```

Out[5]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	T
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3
...	...	...	...	...	...	...	...	...	...	
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175	42.3
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.4
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.5920	33.4
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.2910	69.1
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.9190	649.2

1000 rows × 17 columns

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
```

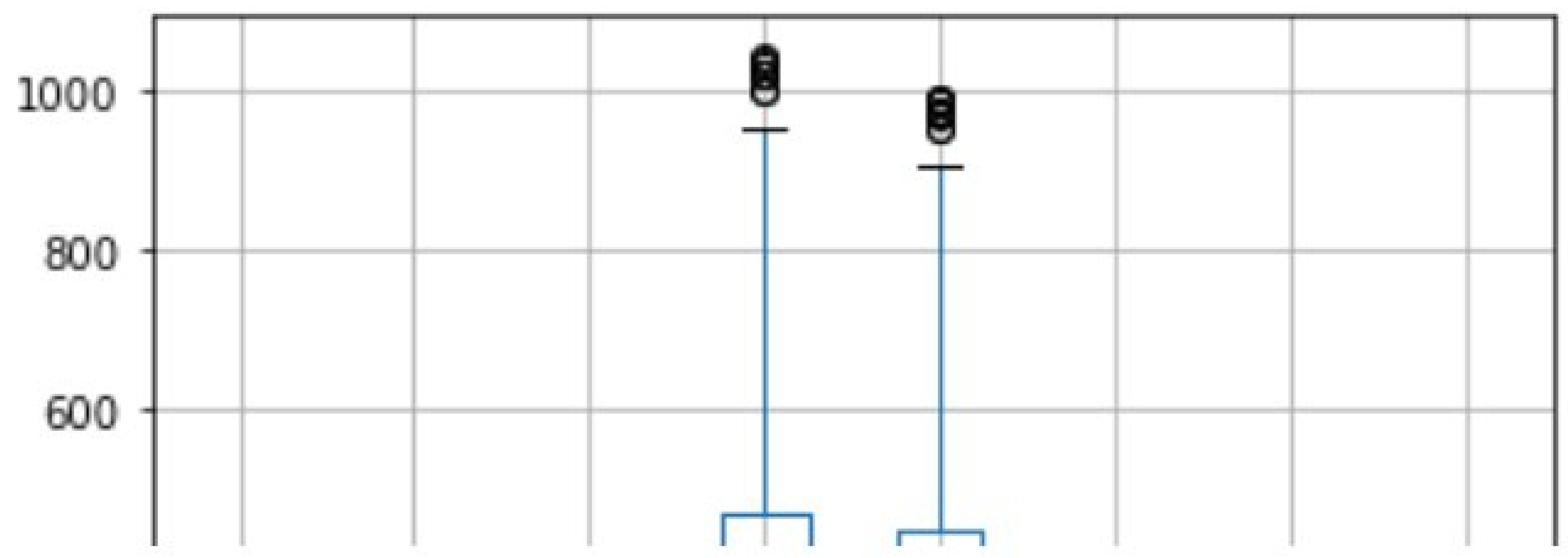
```
 0   Invoice ID               1000 non-null   object 
 1   Branch                   1000 non-null   object 
 2   City                     1000 non-null   object 
 3   Customer type            1000 non-null   object 
 4   Gender                   1000 non-null   object 
 5   Product line             1000 non-null   object 
 6   Unit price               1000 non-null   float64
 7   Quantity                 1000 non-null   int64  
 8   Tax 5%                   1000 non-null   float64
 9   Total                    1000 non-null   float64
 10  Date                     1000 non-null   object 
 11  Time                     1000 non-null   object 
 12  Payment                  1000 non-null   object 
 13  cogs                     1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income             1000 non-null   float64
 16  Rating                   1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
```

In [30]:
```
df.isnull().info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Invoice ID               1000 non-null   bool 
 1   Branch                   1000 non-null   bool 
 2   City                     1000 non-null   bool 
 3   Customer type            1000 non-null   bool 
 4   Gender                   1000 non-null   bool 
 5   Product line             1000 non-null   bool 
 6   Unit price               1000 non-null   bool 
 7   Quantity                 1000 non-null   bool 
 8   Tax 5%                   1000 non-null   bool 
 9   Total                    1000 non-null   bool 
 10  Date                     1000 non-null   bool 
 11  Time                     1000 non-null   bool 
 12  Payment                  1000 non-null   bool 
 13  cogs                     1000 non-null   bool 
 14  gross margin percentage  1000 non-null   bool 
 15  gross income             1000 non-null   bool 
 16  Rating                   1000 non-null   bool 
dtypes: bool(17)
memory usage: 16.7 KB
```

In [27]:
```
df.boxplot()
```

Out[27]:
```
<AxesSubplot:>
```



In [12]:
```
df["Quantity"].value_counts()
```

Out[12]:
```
10    119
1     112
4     109
7     102
5     102
6      98
9      92
2      91
3      90
8      85
Name: Quantity, dtype: int64
```

In [13]:
```
df["Gender"].value_counts()
```

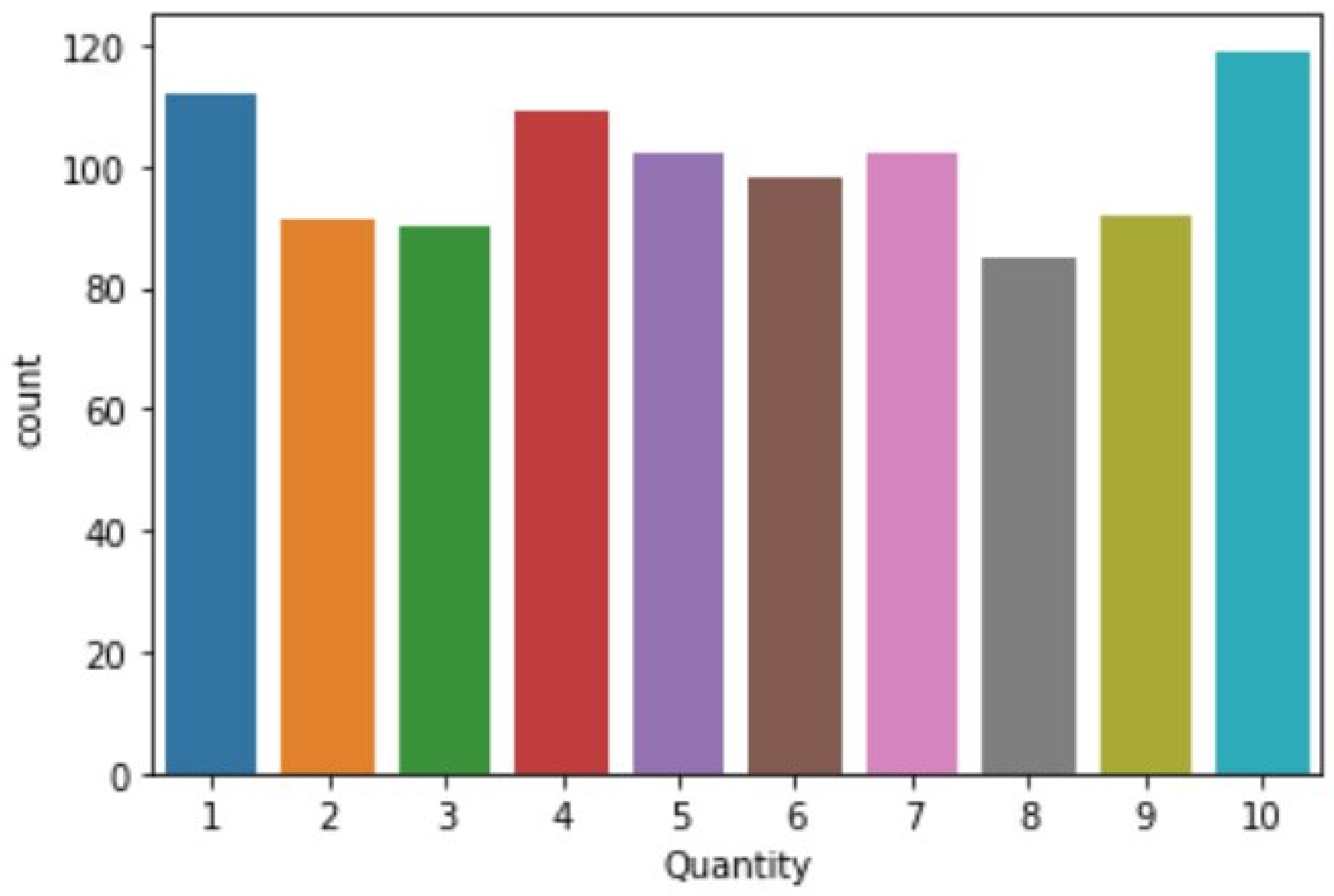
Out[13]:
```
Female    501
Male      499
Name: Gender, dtype: int64
```

In [20]:
```
df["Customer type"].value_counts()
```

Out[20]:
```
Member    501
Normal    499
Name: Customer type, dtype: int64
```

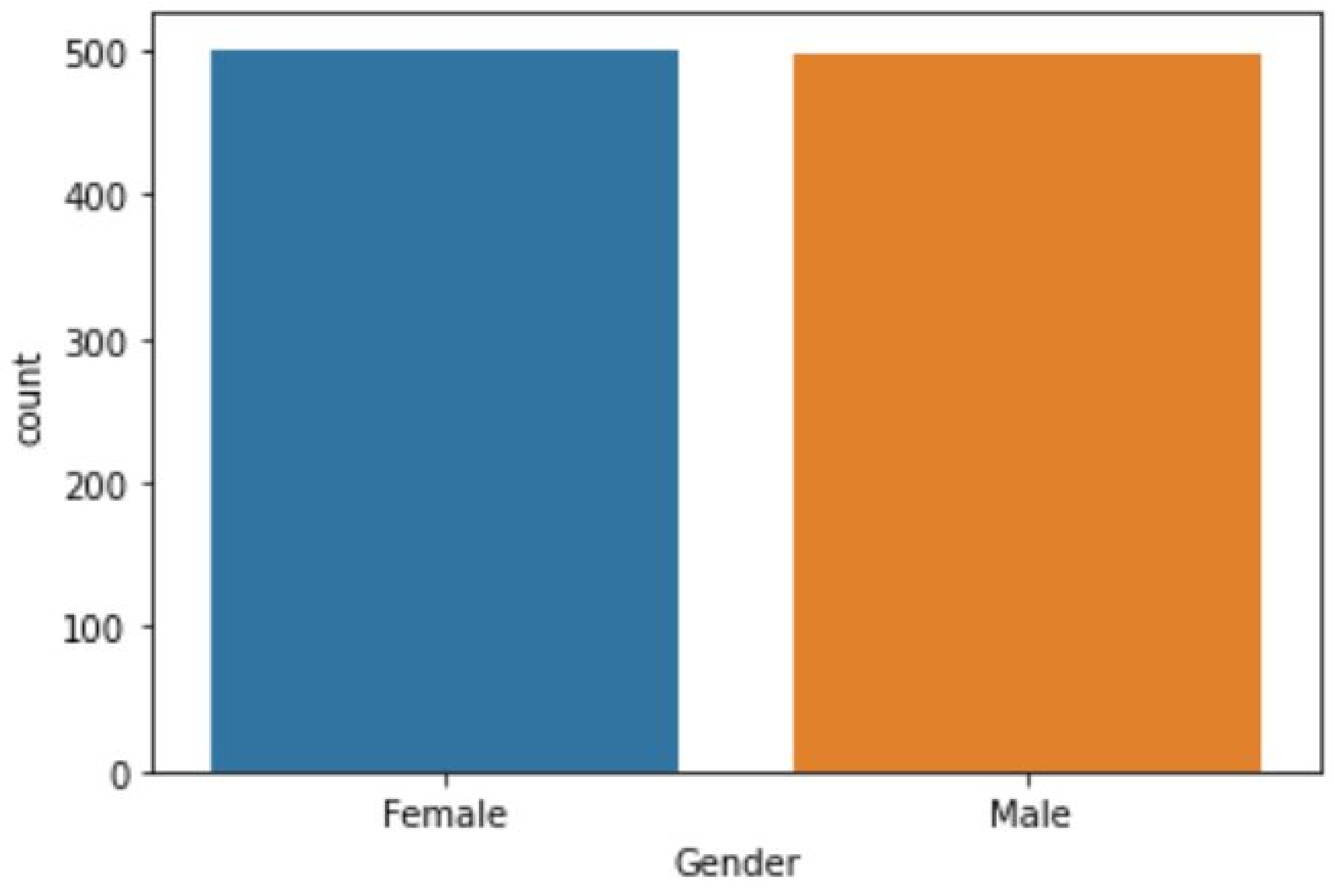
In [14]:
```
sns.countplot(x="Quantity",data=df)
```

Out[14]:
```
<AxesSubplot:xlabel='Quantity', ylabel='count'>
```



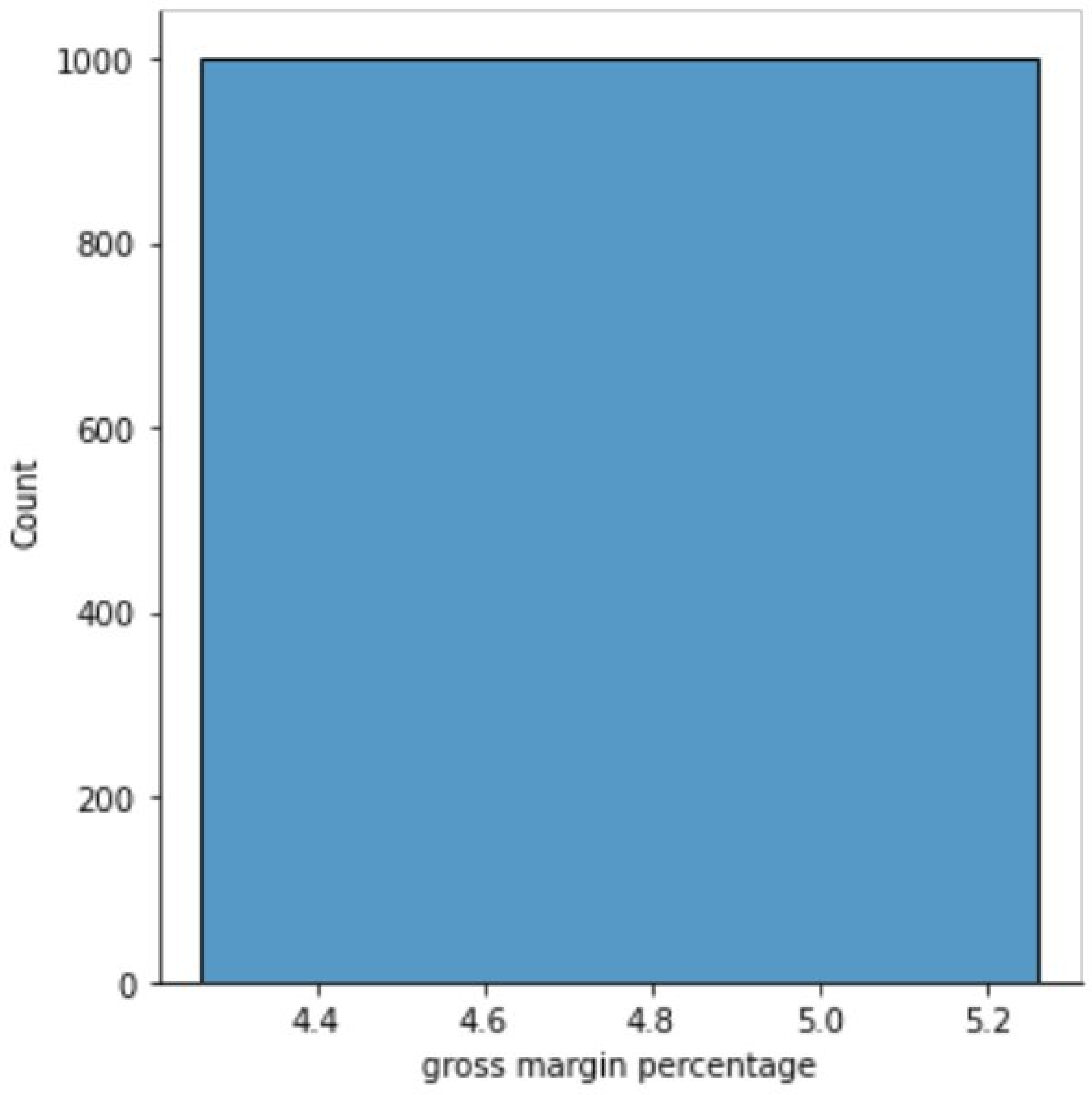
In [15]:
```
sns.countplot(x="Gender",data=df)
```

Out[15]:
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



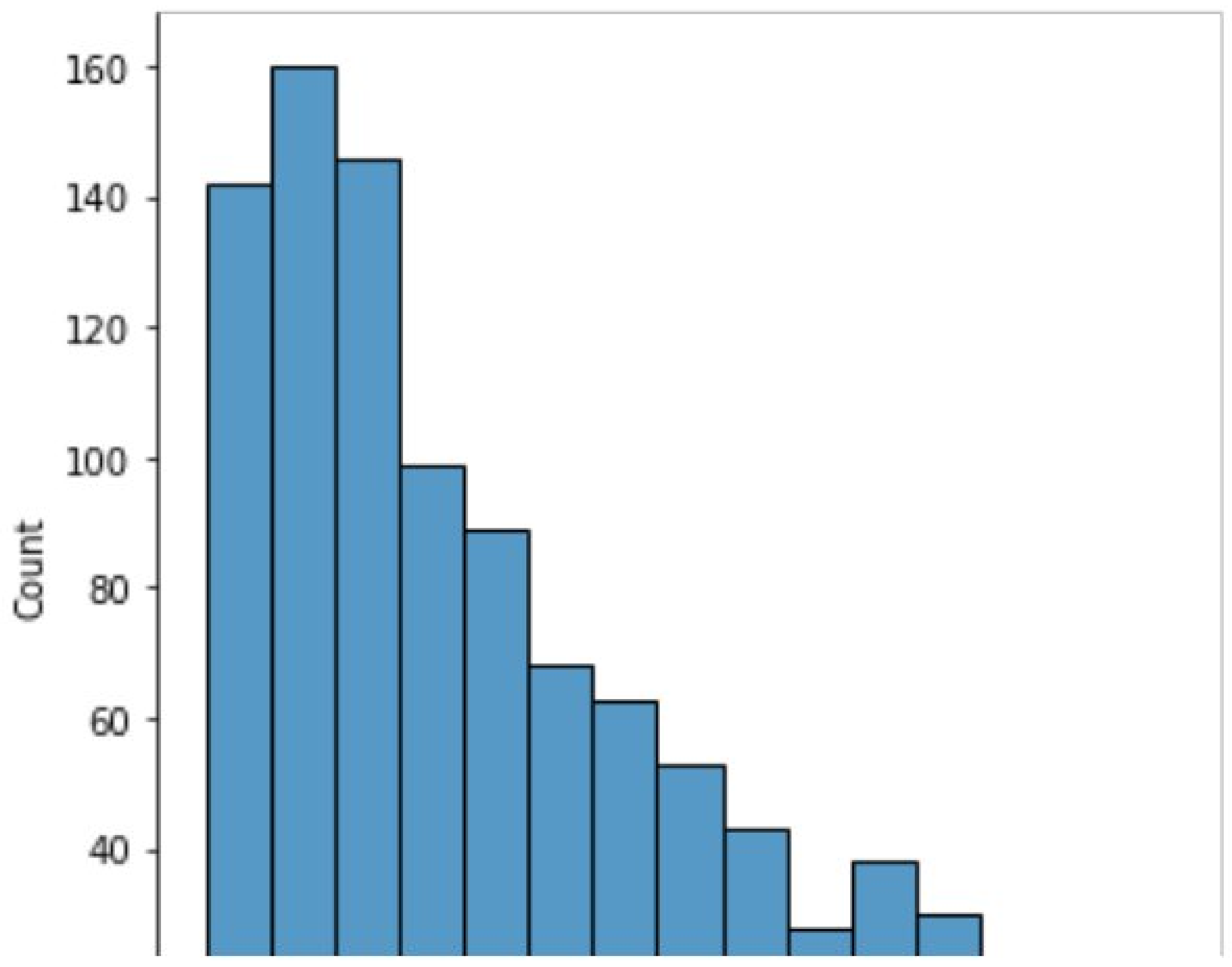
In [16]:
```
sns.displot(df["gross margin percentage"])
```

Out[16]:
```
<seaborn.axisgrid.FacetGrid at 0x1425b89afa0>
```



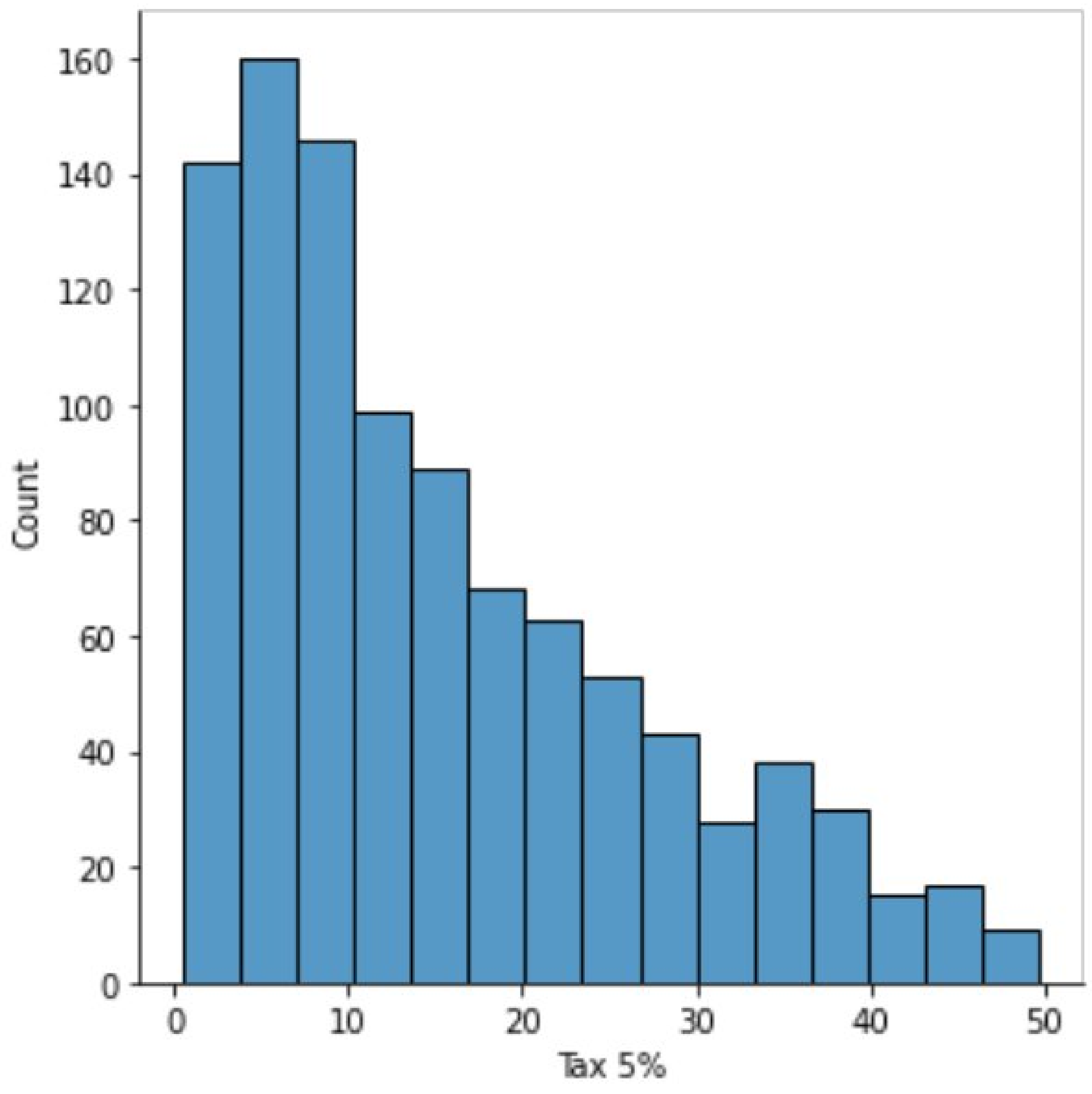
In [17]:
```
sns.displot(df["Total"])
```

Out[17]:
```
<seaborn.axisgrid.FacetGrid at 0x1425bbf6b80>
```



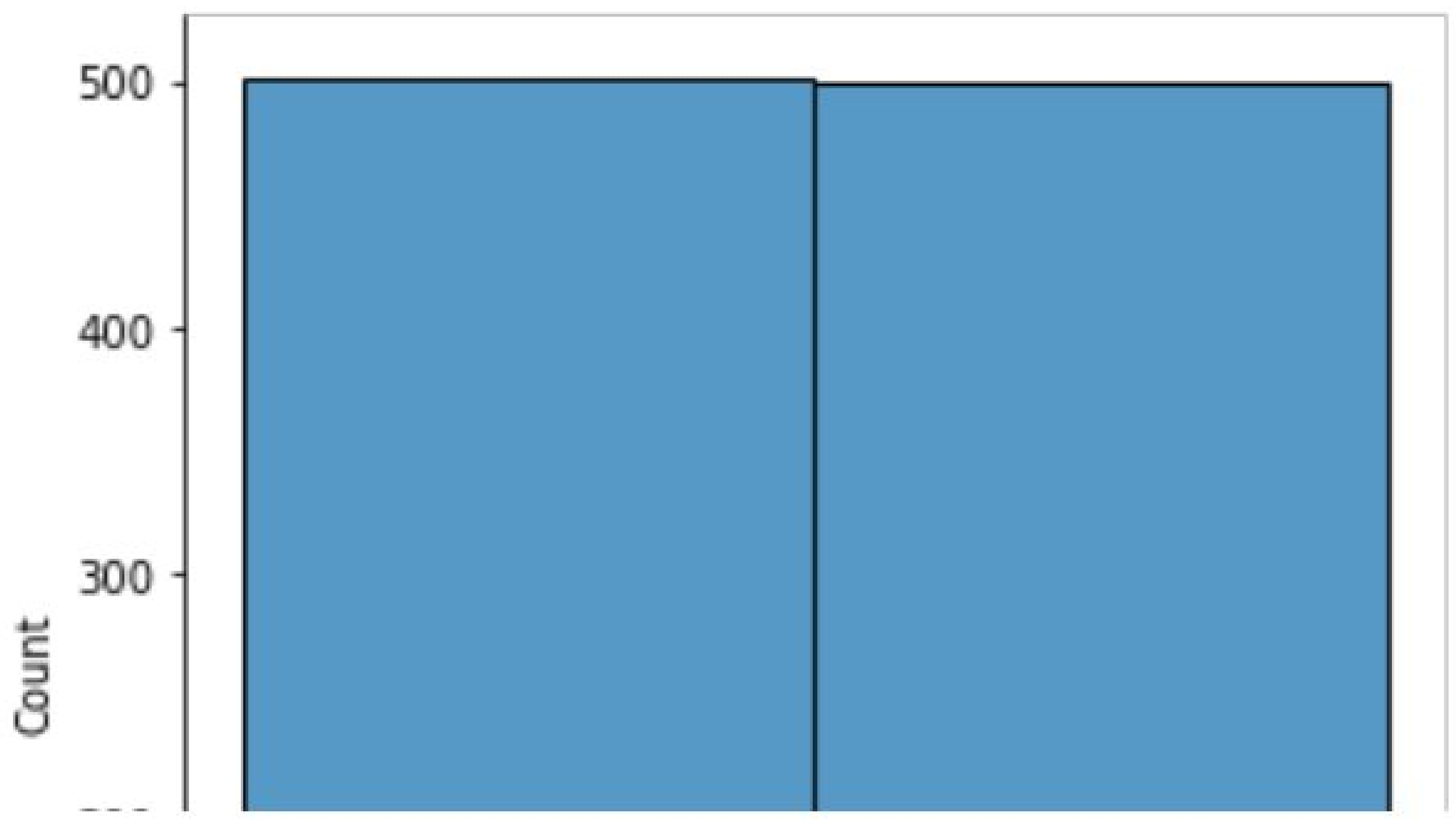
```
In [19]:  sns.displot(df["Tax 5%"])
```

```
Out[19]:  <seaborn.axisgrid.FacetGrid at 0x1425bd57f70>
```



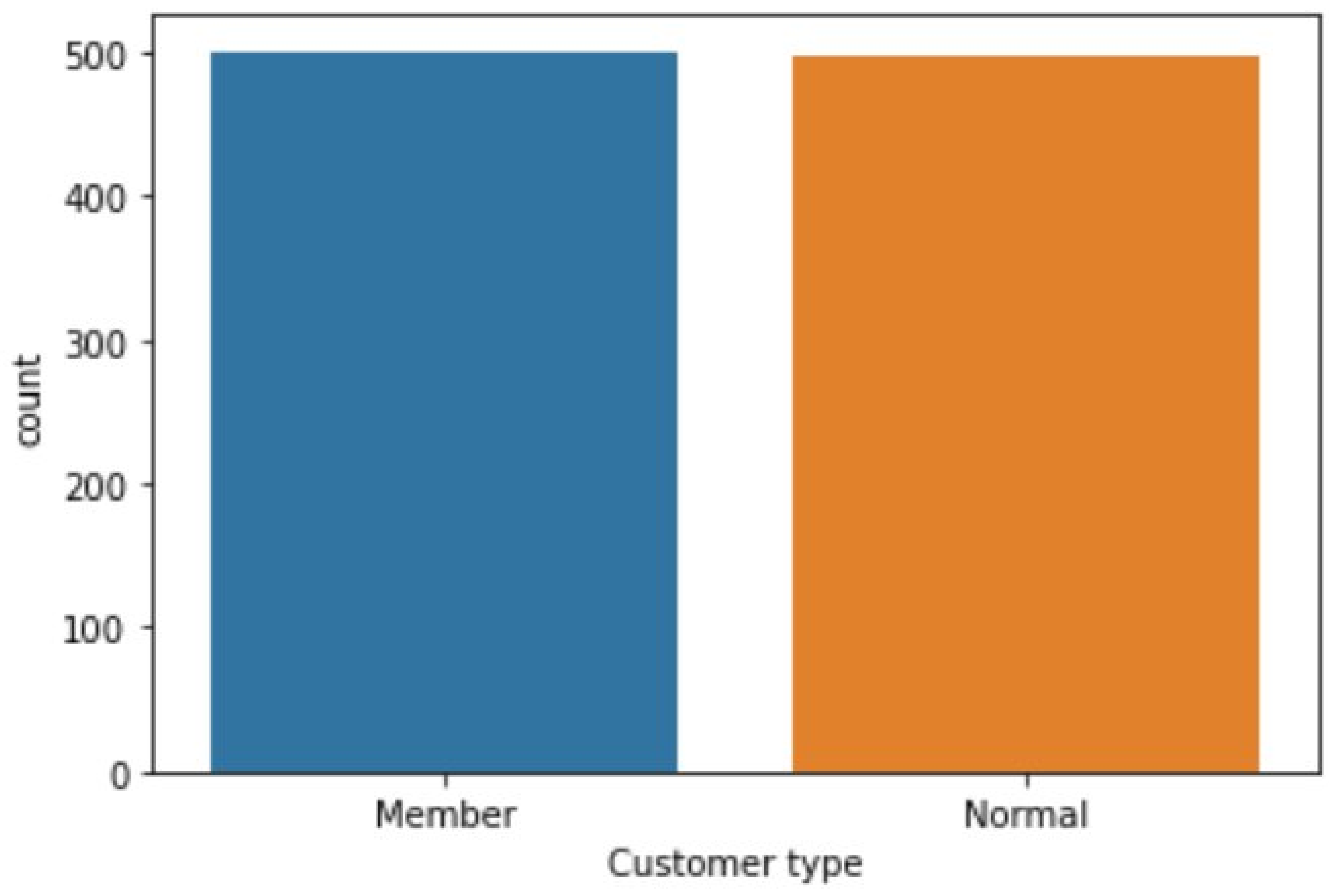
```
In [25]:  sns.displot(df["Customer type"])
```

```
Out[25]:  <seaborn.axisgrid.FacetGrid at 0x1425bffe580>
```



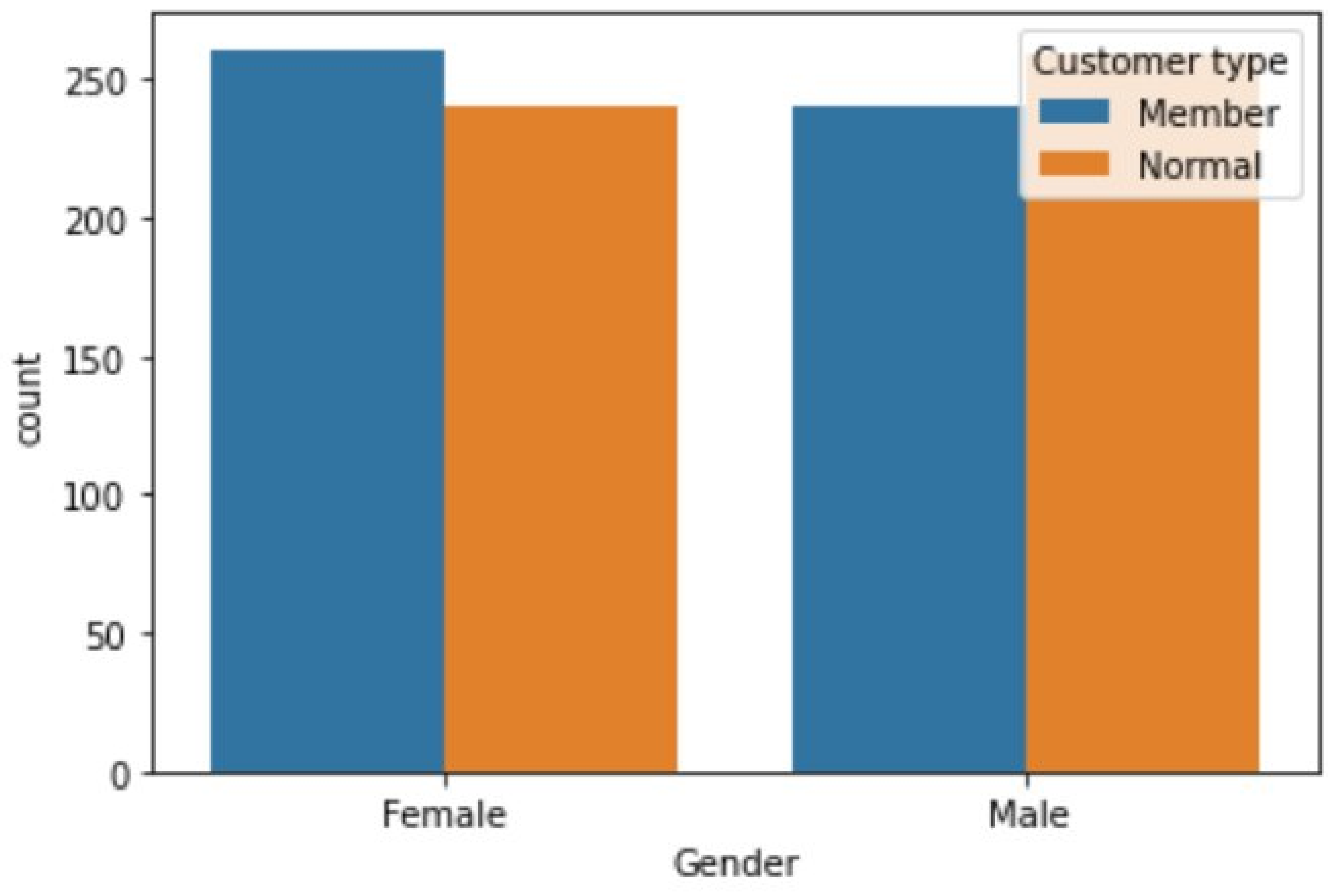
In [21]:
```
sns.countplot(x="Customer type",data=df)
```

Out[21]:
```
<AxesSubplot:xlabel='Customer type', ylabel='count'>
```



In [23]:
```
sns.countplot(x="Gender",hue="Customer type",data=df)
```

Out[23]:
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



In [26]:
```
pd.crosstab(df["Gender"],df["Customer type"])
```

Out[26]:

Customer type	Member	Normal
Gender		
Female	261	240
Male	240	259

In [32]:
```
pd.crosstab(df["Payment"],df["Customer type"])
```

Out[32]:

Customer type	Member	Normal
Payment		
Cash	168	176
Credit card	172	139
Ewallet	161	184

In [ ]:
```
df.drop
```

In [29]:
```
df.corr()
```

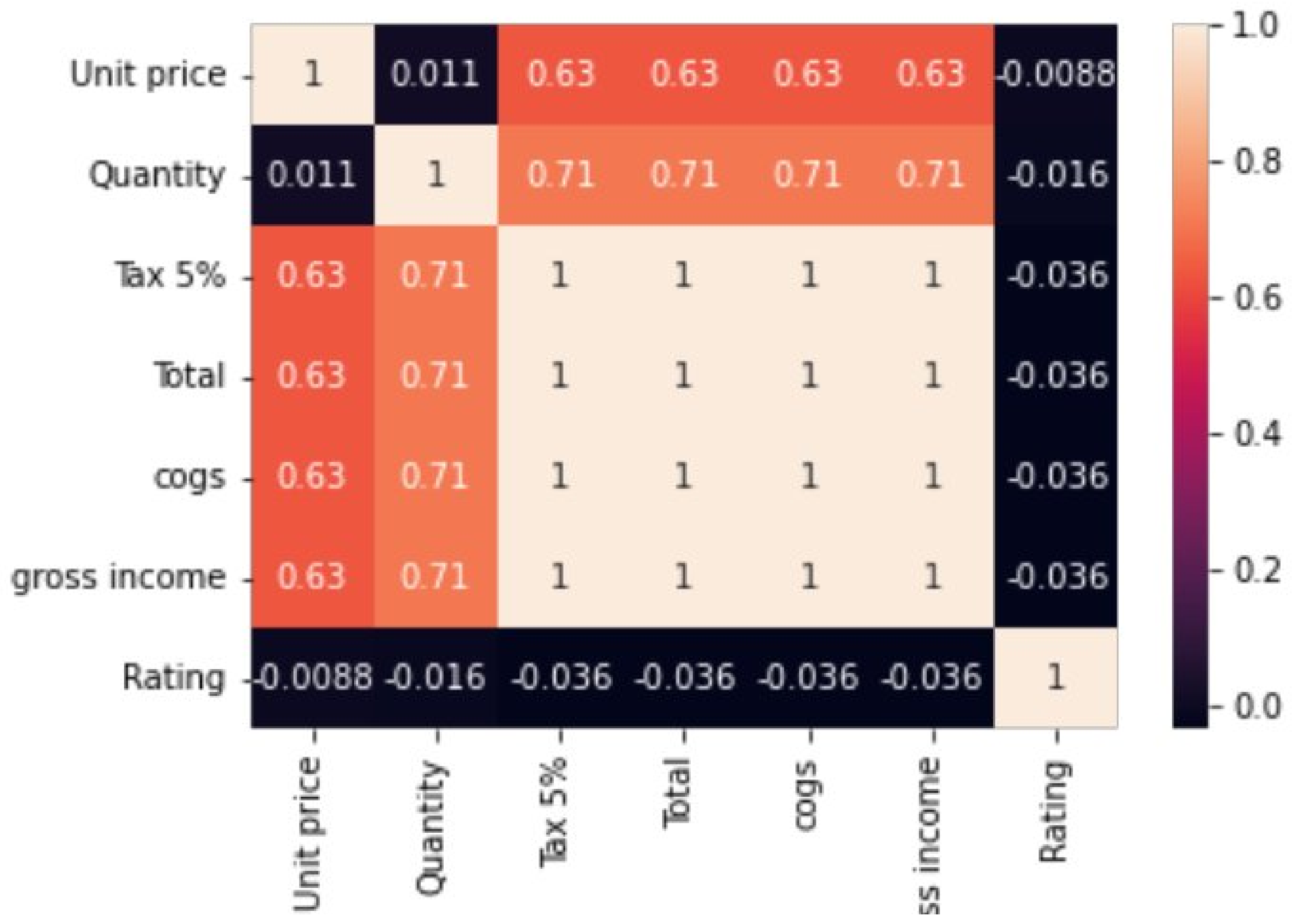
Out[29]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
Unit price	1.000000	0.010778	0.633962	0.633962	0.633962	NaN	0.633962	-0.008778
Quantity	0.010778	1.000000	0.705510	0.705510	0.705510	NaN	0.705510	-0.015815
Tax 5%	0.633962	0.705510	1.000000	1.000000	1.000000	NaN	1.000000	-0.036442
Total	0.633962	0.705510	1.000000	1.000000	1.000000	NaN	1.000000	-0.036442
cogs	0.633962	0.705510	1.000000	1.000000	1.000000	NaN	1.000000	-0.036442
gross margin percentage	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
gross income	0.633962	0.705510	1.000000	1.000000	1.000000	NaN	1.000000	-0.036442
Rating	-0.008778	-0.015815	-0.036442	-0.036442	-0.036442	NaN	-0.036442	1.000000

In [33]:
```
df.drop("gross margin percentage",axis=1,inplace=True)
```

In [34]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[34]:
```
<AxesSubplot:>
```



```
In [ ]: df.drop("Cabin",axis=1,inplace=True)
```