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

import warnings
warnings.filterwarnings('ignore')
```

```python
data.head()
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	4.761905	26.1415	9.1
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.8200	9.6
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.2155	7.4

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.2880	8.4
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	4.761905	30.2085	5.3

```
data.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
```

```
data['Date']=pd.to_datetime(data['Date'])
```

```
data1=data.copy()
```

```
data1['day']=(data1['Date']).dt.day
data1['month']=(data1['Date']).dt.month
data1['year']=(data1['Date']).dt.year
```

```
data1.head()
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating	day	month	year
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	2019-01-05	13:08	Ewallet	522.83	4.761905	26.1415	9.1	5	1	2019

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating	day	month	year
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	2019-03-08	10:29	Cash	76.40	4.761905	3.8200	9.6	8	3	2019
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	2019-03-03	13:23	Credit card	324.31	4.761905	16.2155	7.4	3	3	2019
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	2019-01-27	20:33	Ewallet	465.76	4.761905	23.2880	8.4	27	1	2019
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	604.3785	2019-02-	10:37	Ewallet	604.17	4.761905	30.2085	5.3	8	2	2019

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating	day	month	year
											08									

```
data1.columns
```

Out[10]:

```
Index(['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender',
       'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Date'
,
       'Time', 'Payment', 'cogs', 'gross margin percentage', 'gross income
',
       'Rating', 'day', 'month', 'year'],
      dtype='object')
```

```
data_col=['Branch', 'City', 'Customer type', 'Gender',
       'Product line','Quantity','Payment','month','year']
```

```
for i in data_col:
    print(i + ': ')
    print(data1[i].unique())
```

```
Branch:
['A' 'C' 'B']
City:
['Yangon' 'Naypyitaw' 'Mandalay']
Customer type:
['Member' 'Normal']
Gender:
['Female' 'Male']
```

```
Product line:
['Health and beauty' 'Electronic accessories' 'Home and lifestyle'
 'Sports and travel' 'Food and beverages' 'Fashion accessories']
Quantity:
[ 7  5  8  6 10  2  3  4  1  9]
Payment:
['Ewallet' 'Cash' 'Credit card']
month:
[1 3 2]
year:
[2019]
```

```
data2=data1.copy()
```

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
```

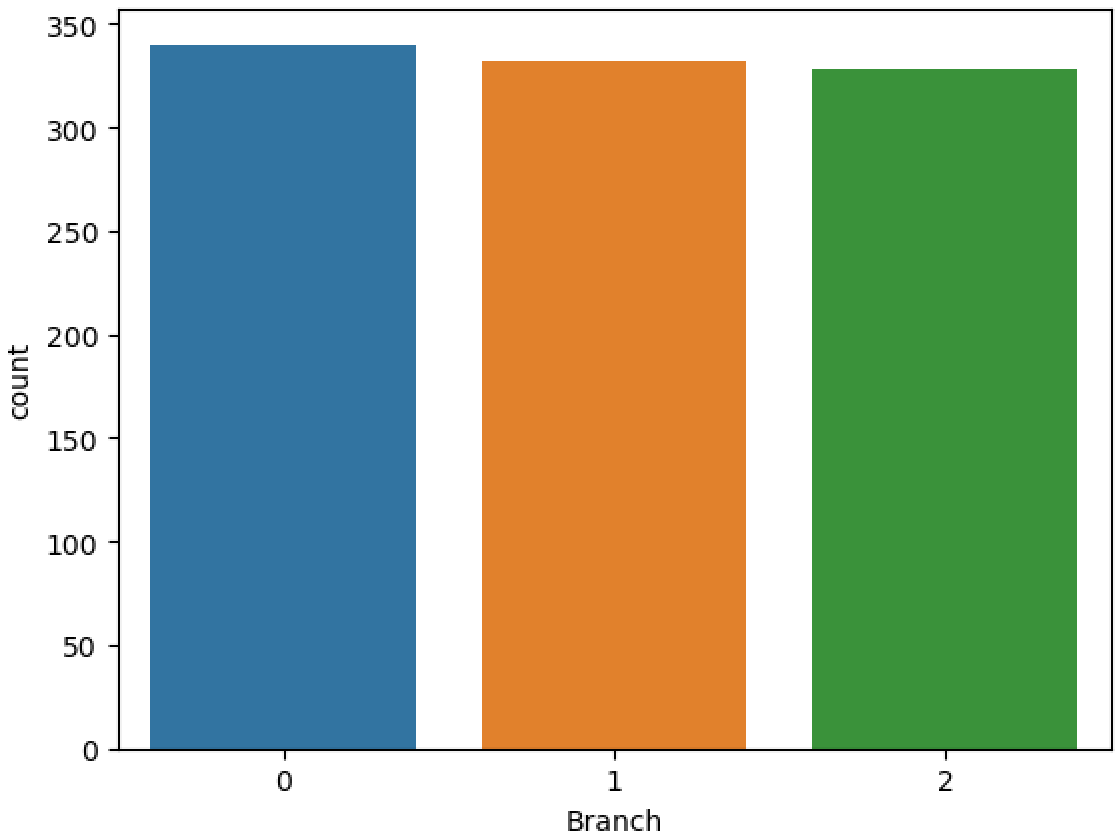
```
label_data=['Branch', 'City', 'Customer type', 'Gender',
       'Product line','Payment']
for j in label_data:
    data2[j]=le.fit_transform(data2[j])
```

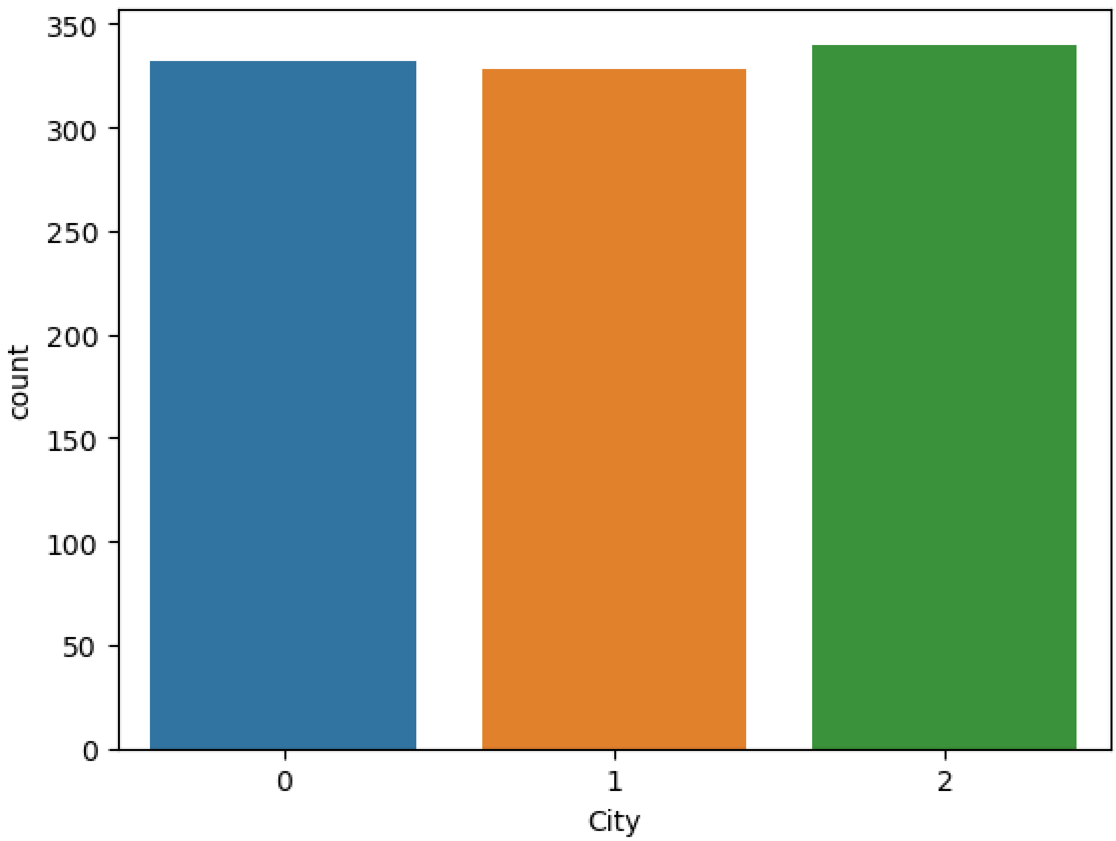
```
data2.head()
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating	day	month	year
0	750-67-8428	0	2	0	0	3	74.69	7	26.1415	548.9715	2019-01-05	13:08	2	522.83	4.761905	26.1415	9.1	5	1	2019
1	226-31-3081	2	1	1	0	0	15.28	5	3.8200	80.2200	2019-03-08	10:29	0	76.40	4.761905	3.8200	9.6	8	3	2019
2	631-41-3108	0	2	1	1	4	46.33	7	16.2155	340.5255	2019-03-03	13:23	1	324.31	4.761905	16.2155	7.4	3	3	2019
3	123-19-1176	0	2	0	1	3	58.22	8	23.2880	489.0480	2019-01-27	20:33	2	465.76	4.761905	23.2880	8.4	27	1	2019

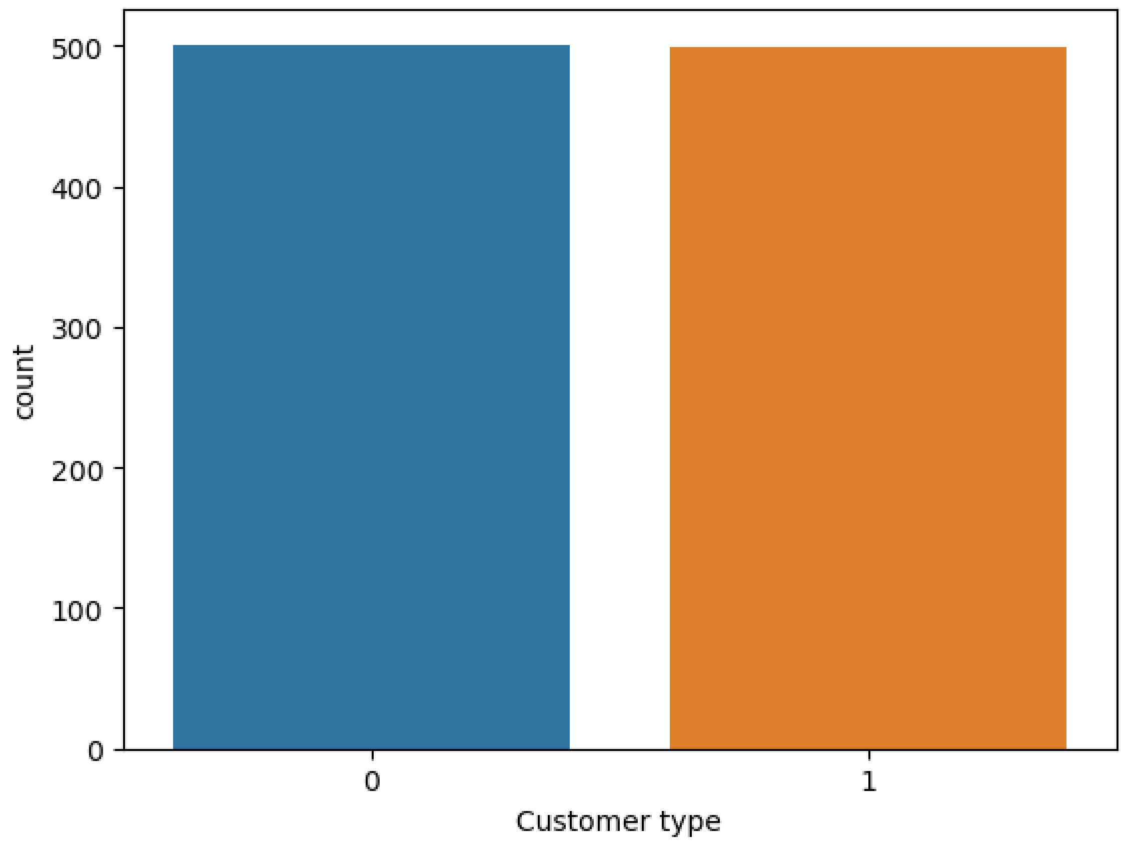
	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating	day	month	year
4	373-73-7910	0	2	1	1	5	86.31	7	30.2085	634.3785	2019-02-08	10:37	2	604.17	4.761905	30.2085	5.3	8	2	2019

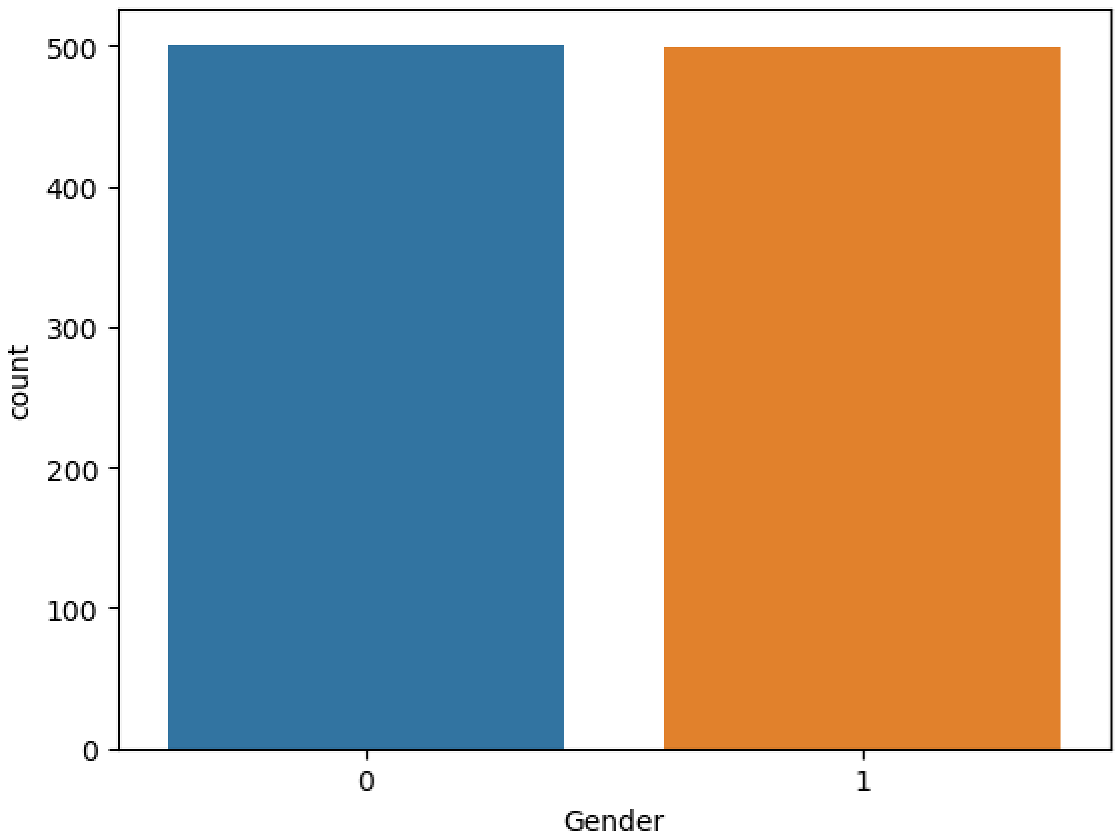
```
for k in label_data:
    sns.countplot(x=data2[k])
    plt.show()
```


350
300
250
200
150
100
50
0
count
0
1
2
Branch

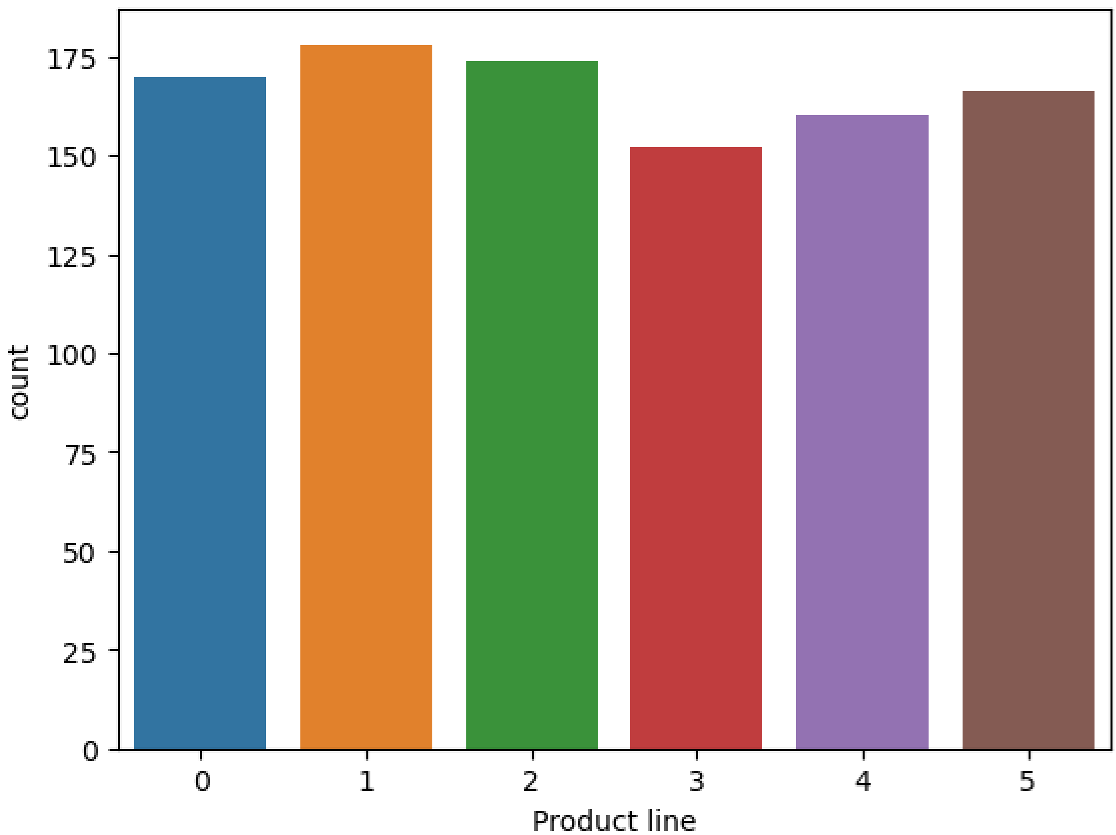


350
300
250
200
150
100
50
0
count
0
1
2
City

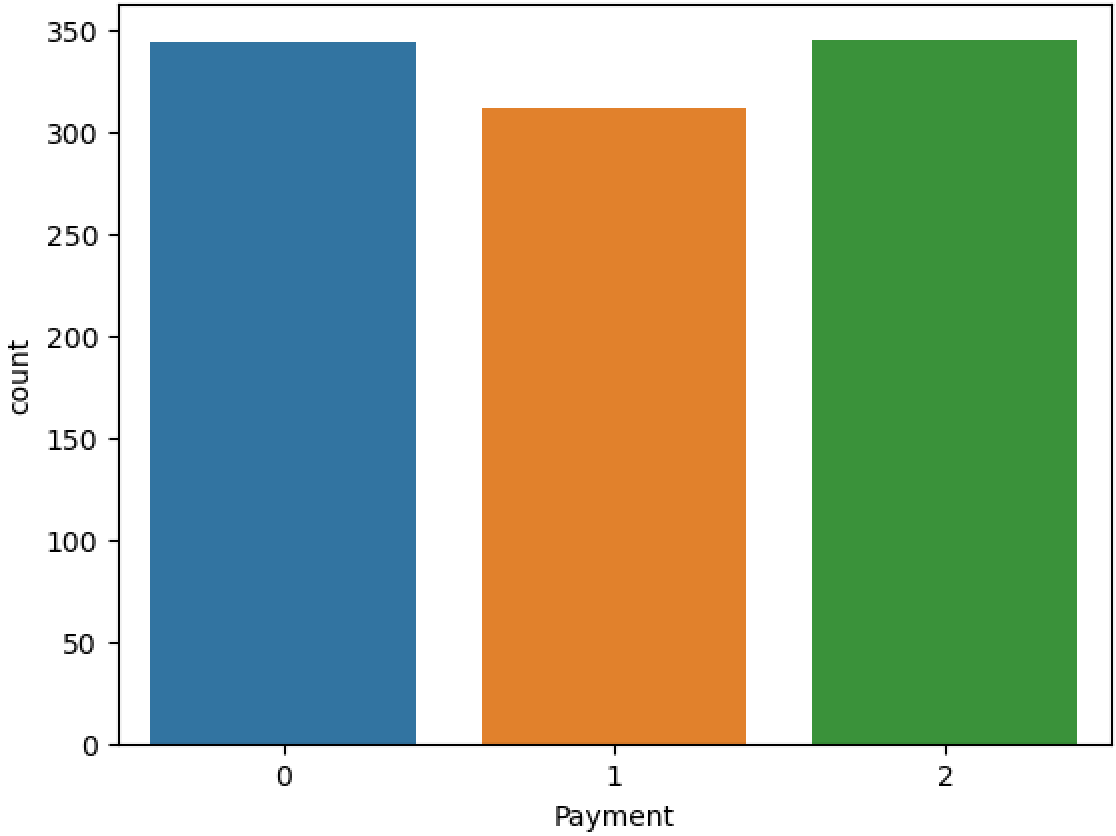

count
500
400
300
200
100
0
0
1
Customer type



500
400
300
200
100
0
count
0
1
Gender

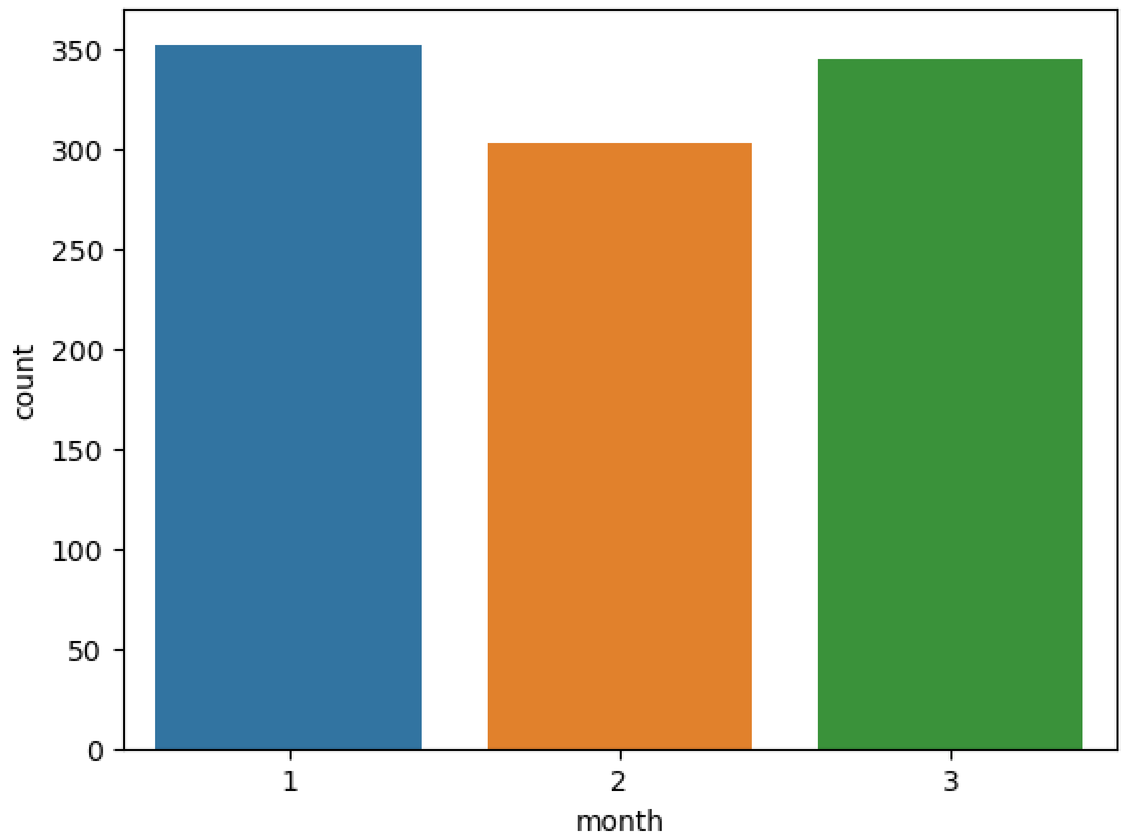


175
150
125
100
75
50
25
0
count
0
1
2
3
4
5
Product line




```
sns.countplot(x=data2['month'])
```

```
<AxesSubplot:xlabel='month', ylabel='count'>
```




```
data3=data2.copy()
```

```
data3.drop(['Invoice ID', 'Date',
       'Time', 'day', 'month', 'year'],axis=1,inplace=True)
```

```
data3.head()
```

	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Payment	cogs	gross margin percentage	gross income	Rating
0	0	2	0	0	3	74.69	7	26.1415	548.9715	2	522.83	4.761905	26.1415	9.1
1	2	1	1	0	0	15.28	5	3.8200	80.2200	0	76.40	4.761905	3.8200	9.6
2	0	2	1	1	4	46.33	7	16.2155	340.5255	1	324.31	4.761905	16.2155	7.4
3	0	2	0	1	3	58.22	8	23.2880	489.0480	2	465.76	4.761905	23.2880	8.4
4	0	2	1	1	5	86.31	7	30.2085	634.3785	2	604.17	4.761905	30.2085	5.3

```
data3.columns
```

```
Index(['Branch', 'City', 'Customer type', 'Gender', 'Product line',
       'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Payment', 'cogs',
       'gross margin percentage', 'gross income', 'Rating'],
      dtype='object')
```

```
data3 = data3.reindex(columns=['Branch', 'City', 'Customer type', 'Gender',
'Product line',
       'Unit price', 'Quantity', 'Tax 5%', 'Payment', 'cogs',
       'gross margin percentage', 'gross income', 'Rating', 'Total'])
```

```
x=data3.drop(['Total'],axis=1)
y=data3[['Total']]
```

```
x.shape,y.shape, data3.shape
```

```
((1000, 13), (1000, 1), (1000, 14))
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4,random_stat
e=42)
```

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((600, 13), (400, 13), (600, 1), (400, 1))
```

```
from sklearn.preprocessing import StandardScaler
stand= StandardScaler()
x_train=stand.fit_transform(x_train)
x_test=stand.transform(x_test)
```

```
from sklearn.ensemble import RandomForestRegressor
rfc=RandomForestRegressor(n_estimators=100)
```

```
rfc.fit(x_train,y_train)
```

```
RandomForestRegressor()
```

```python
from sklearn.metrics import r2_score
```

```python
rfc_y_predict=rfc.predict(x_test)
```

```python
r2_score(y_test, rfc_y_predict)
```

```
0.9999350661657065
```

```python
for z in range(0,20):
    diff=y_test.values[z]-rfc_y_predict[z]
    print(y_test.values[z] , ' ', rfc_y_predict[z],' ', diff )
```

```
[523.971]   521.2057199999999   [2.76528]
[616.98]   616.4257049999995   [0.554295]
[408.7335]   408.61380000000014   [0.1197]
[135.3555]   135.63669000000002   [-0.28119]
[45.927]   44.60925000000004   [1.31775]
[618.975]   620.2895999999995   [-1.3146]
[127.827]   127.36531499999994   [0.461685]
[731.6925]   731.7491999999995   [-0.0567]
[450.1035]   451.34858999999994   [-1.24509]
[138.1275]   137.48920500000008   [0.638295]
[422.73]   422.38307999999995   [0.34692]
[463.428]   462.1820699999998   [1.24593]
[212.7825]   213.20018999999982   [-0.41769]
[252.252]   252.18563999999998   [0.06636]
[290.0835]   289.80965999999984   [0.27384]
[331.128]   329.26487999999995   [1.86312]
[587.664]   587.83557   [-0.17157]
[216.846]   216.92013000000009   [-0.07413]
[757.365]   757.3464150000001   [0.018585]
[185.094]   185.0490599999999   [0.04494]
```