In [1]:

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```python
data = pd.read_csv('Customer-Churn.csv')
```

In [3]:

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

Out[3]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service
1	5575-GNVDE	Male	0	No	No	34	Yes	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service
4	9237-HQITU	Female	0	No	No	2	Yes	No

5 rows × 21 columns

In [4]:

```python
data.size
```

Out[4]:

```
147903
```

In [5]:

```python
data.shape
```

Out[5]:

```
(7043, 21)
```

In [6]:

```python
data.columns
```

Out[6]:

```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

In [7]:

```python
data.dtypes
```

Out[7]:

```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

In [8]:

```
data.isnull().sum()
```

Out[8]:

```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

In [9]:

```
data.duplicated().sum()
```

Out[9]:

```
0
```

In [10]:

```
data.dtypes
```

Out[10]:

```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

In [11]:

```
data.describe()
```

Out[11]:

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

In [12]:

```
data
```

Out[12]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLin
0	7590-VHVEG	Female	0	Yes	No	1	No	No phor servi
1	5575-GNVDE	Male	0	No	No	34	Yes	N
2	3668-QPYBK	Male	0	No	No	2	Yes	N
3	7795-CFOCW	Male	0	No	No	45	No	No phor servi
4	9237-HQITU	Female	0	No	No	2	Yes	N
...	...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Y
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Y
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phor servi
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Y
7042	3186-AJIEK	Male	0	No	No	66	Yes	N

7043 rows × 21 columns

In [13]:
```
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors='coerce')
```

In [14]:
```
data.dtypes
```

Out[14]:
```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges        float64
Churn                object
dtype: object
```

In [15]:
```
data.isnull().sum()
```

Out[15]:
```
customerID           0
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
Churn                0
dtype: int64
```

In [16]:
```
data = data.dropna()
```

In [17]:
```
data.isnull().sum()
```

Out[17]:

```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

In [18]:

```
# Select categorical columns
categorical_columns = data.select_dtypes(include=['object']).columns.tolist()

# Select numeric columns
numeric_columns = data.select_dtypes(include=['number']).columns.tolist()
```

In [19]:

```
categorical_columns
```

Out[19]:

```
['customerID',
 'gender',
 'Partner',
 'Dependents',
 'PhoneService',
 'MultipleLines',
 'InternetService',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'Contract',
 'PaperlessBilling',
 'PaymentMethod',
 'Churn']
```

In [20]:

```
numeric_columns
```

Out[20]:

```
['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges']
```

In [21]:

```
categorical_count = data['customerID'].nunique()
```

In [22]:

```
data['Contract'].value_counts()
```

Out[22]:

```
Contract
Month-to-month    3875
Two year          1685
One year          1472
Name: count, dtype: int64
```

In [23]:

```
numeric_columns = data.select_dtypes(exclude=['object'])
```

In [24]:

```
numeric_columns.corr()
```

Out[24]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
SeniorCitizen	1.000000	0.015683	0.219874	0.102411
tenure	0.015683	1.000000	0.246862	0.825880
MonthlyCharges	0.219874	0.246862	1.000000	0.651065
TotalCharges	0.102411	0.825880	0.651065	1.000000

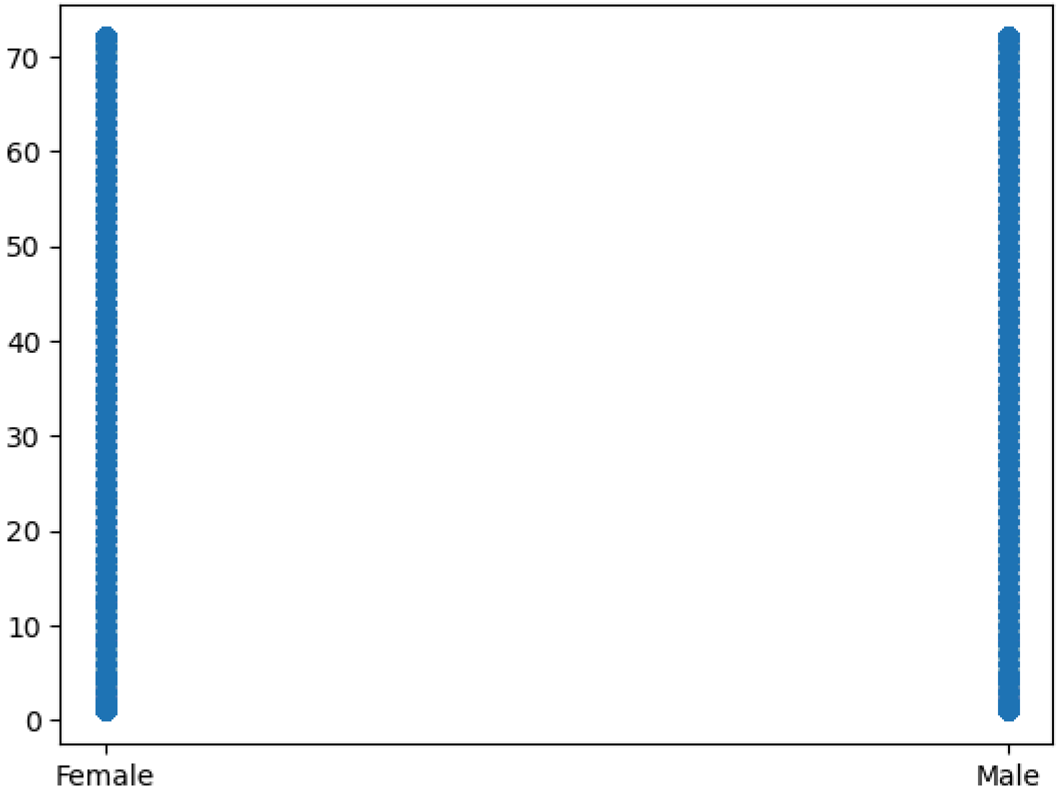
Visualization

In [25]:

```
plt.scatter(data['gender'], data['tenure'])
```

Out[25]:

```
<matplotlib.collections.PathCollection at 0x2999a866ed0>
```

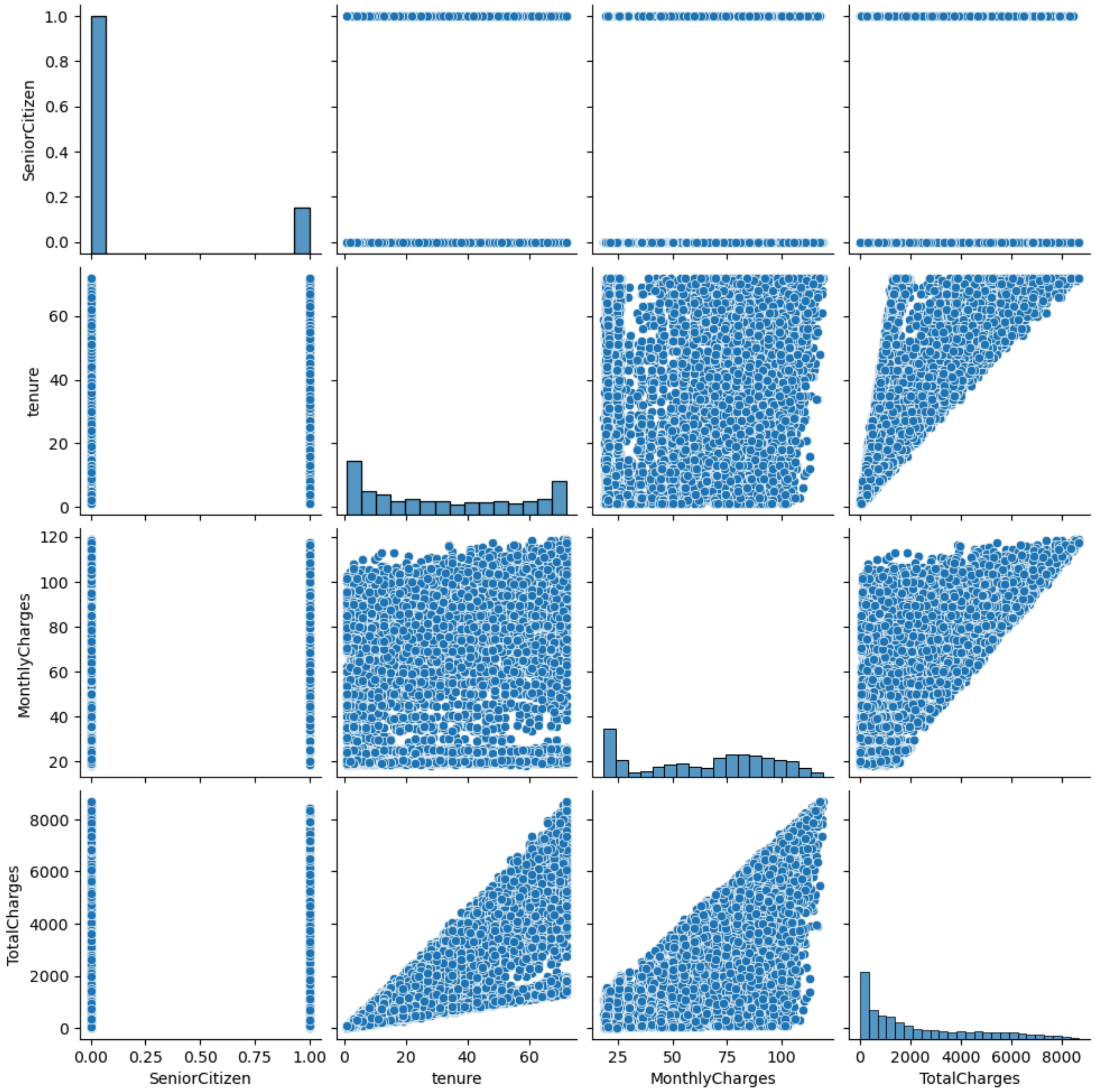


In [26]:

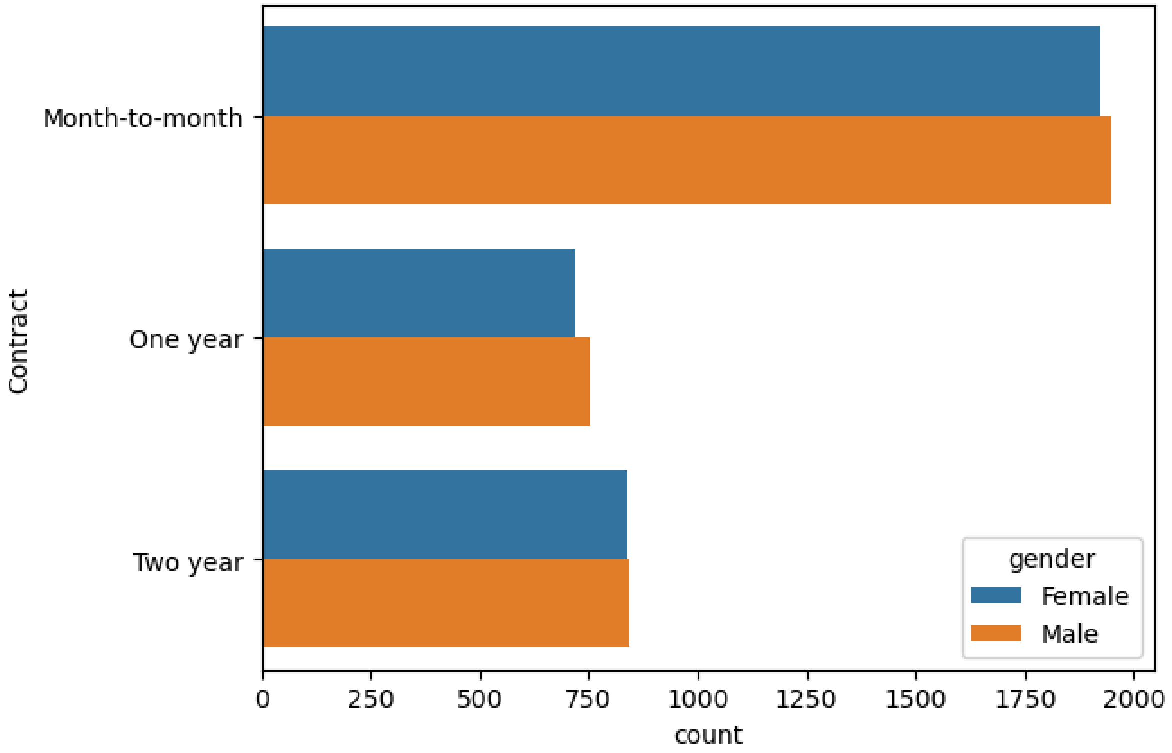
```
sns.pairplot(data)
```

Out[26]:

```
<seaborn.axisgrid.PairGrid at 0x299a0381c10>
```



In [27]:
```
sns.countplot(data=data, y ='Contract', hue='gender');
```



Model Evaluation

In [28]:

```
data.head()
```

Out[28]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service
1	5575-GNVDE	Male	0	No	No	34	Yes	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service
4	9237-HQITU	Female	0	No	No	2	Yes	No

5 rows × 21 columns

In [29]:

```
from sklearn.preprocessing import LabelEncoder




# Select categorical columns
categorical_columns = data.select_dtypes(include=['object'])

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Iterate through each categorical column and transform it
for col in categorical_columns:
    data[col] = label_encoder.fit_transform(data[col])
```

In [30]:

```
data.head()
```

Out[30]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
0	5365	0	0	1	0	1	0	1
1	3953	1	0	0	0	34	1	0
2	2558	1	0	0	0	2	1	0
3	5524	1	0	0	0	45	0	1
4	6500	0	0	0	0	2	1	0

5 rows × 21 columns

In [31]:

```
X= data.drop(['Churn','customerID'],axis=1)
y= data['Churn']
```

In [32]:

```
y
```

Out[32]:

```
0       0
1       0
2       1
3       0
4       1
       ..
7038    0
7039    0
7040    0
7041    1
7042    0
Name: Churn, Length: 7032, dtype: int32
```

In [33]:

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
```

In [34]:

```
# Splitting the dataset into the Training set and Test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [35]:

```
# Creating the decision tree classifier
clf = DecisionTreeClassifier()
# Training the decision tree classifier
clf.fit(X_train, y_train)
```

Out[35]:

```
▾ DecisionTreeClassifier

DecisionTreeClassifier()
```

In [36]:

```
y_test
```

Out[36]:

```
2481    0
6784    0
6125    1
3052    0
4099    0
       ..
1733    0
5250    0
5465    0
5851    0
3984    0
Name: Churn, Length: 1407, dtype: int32
```

In [37]:

```
y_pred = clf.predict(X_test)
```

In [38]:

```
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, classification_report

# Evaluating the model
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7242359630419332
```

In [39]:

```
y_train_pred = clf.predict(X_train)
# Displaying the predicted values
```

```
print("Predictions on Training Data:")
print(y_train_pred)
```

```
Predictions on Training Data:
[1 1 1 ... 0 0 1]
```

In [40]:

```
# Predicting on the test data
y_test_pred = clf.predict(X_test)

# Displaying the predicted values
print("Predictions on Test Data:")
print(y_test_pred)
```

```
Predictions on Test Data:
[0 0 1 ... 0 0 0]
```

In [41]:

```
cm = confusion_matrix(y_test, y_pred)

# Display the confusion matrix
print("Confusion Matrix:")
print(cm)
```

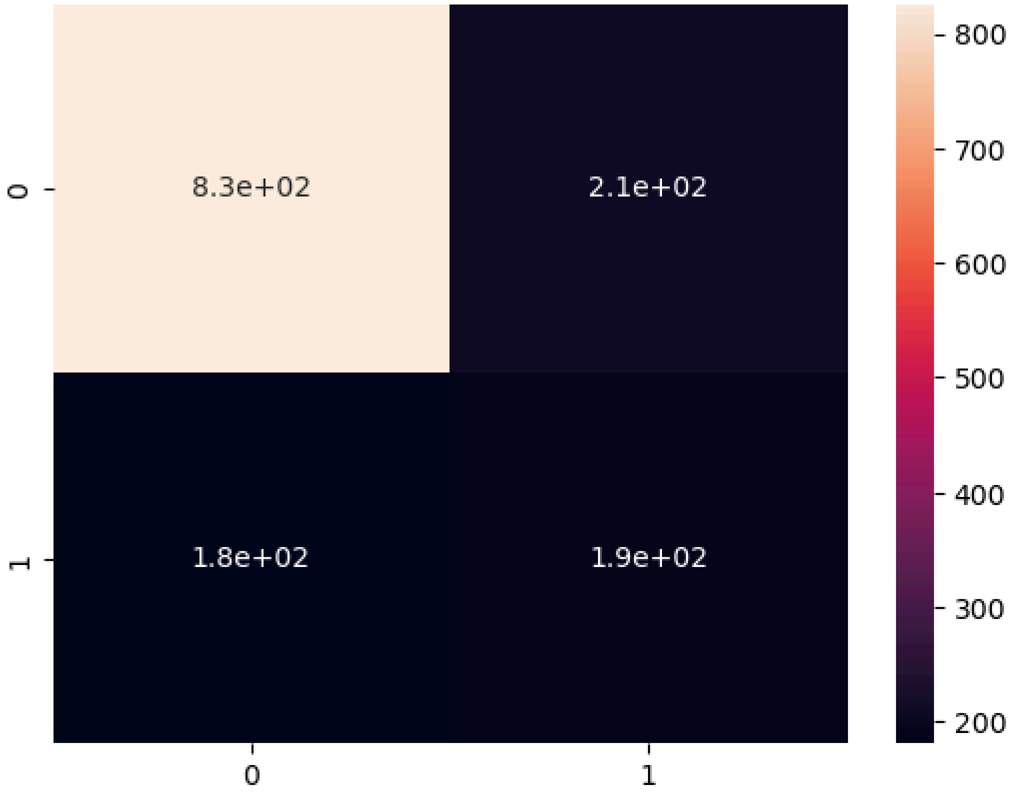
```
Confusion Matrix:
[[826 207]
 [181 193]]
```

In [42]:

```
a = confusion_matrix(y_test, y_pred)
sns.heatmap(a, annot=True)
```

Out[42]:

```
<Axes: >
```



In [43]:

```
s = classification_report(y_test, y_pred)
print(s)
```

```
              precision    recall  f1-score   support

           0       0.82      0.80      0.81      1033
           1       0.48      0.52      0.50       374

    accuracy                           0.72      1407
   macro avg       0.65      0.66      0.65      1407
weighted avg       0.73      0.72      0.73      1407
```

```
In [ ]:
```

```
In [ ]:
```