TELECOM-CUSTOMER-CHURN-EDA-PROJECT_VIVEK_CHAUHAN

In [1]:

```
# import necessary libraries to work with dataset

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

In [2]:

```
data = pd.read_csv("C:/Users/VIVEK CHAUHAN/Desktop/eda-projects (1)/7-eda-project/Telco-Customer-Churn.csv")
data
```

Out[2]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...
...	...	...	...	...	...	...	...	...	...	...	...
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	...

7043 rows × 21 columns

In [3]:

```
# to check all the column names

data.columns
```

Out[3]:

```
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 [4]:

```
# top 10 data from the dataset

data.head(10)
```

Out[4]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	Dev
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	
5	9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic	No	...	
6	1452-KIOVK	Male	0	No	Yes	22	Yes	Yes	Fiber optic	No	...	
7	6713-OKOMC	Female	0	No	No	10	No	No phone service	DSL	Yes	...	
8	7892-POOKP	Female	0	Yes	No	28	Yes	Yes	Fiber optic	No	...	
9	6388-TABGU	Male	0	No	Yes	62	Yes	No	DSL	Yes	...	

10 rows × 21 columns

In [5]:

```
# last 10 data from the dataset

data.tail(10)
```

Out[5]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...
7033	9767-FFLEM	Male	0	No	No	38	Yes	No	Fiber optic	No	...
7034	0639-TSIQW	Female	0	No	No	67	Yes	Yes	Fiber optic	Yes	...
7035	8456-QDAVC	Male	0	No	No	19	Yes	No	Fiber optic	No	...
7036	7750-EYXWZ	Female	0	No	No	12	No	No phone service	DSL	No	...
7037	2569-WGERO	Female	0	No	No	72	Yes	No	No	No internet service	...
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	...

10 rows × 21 columns

In [6]:

```
# checking the data types of all the columns
```

```
data.dtypes
```

Out[6]:

```
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 [7]:

```
# info about our dataset

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

In [8]:

```
# get the satistics about our dataser

data.describe()
```

Out[8]:

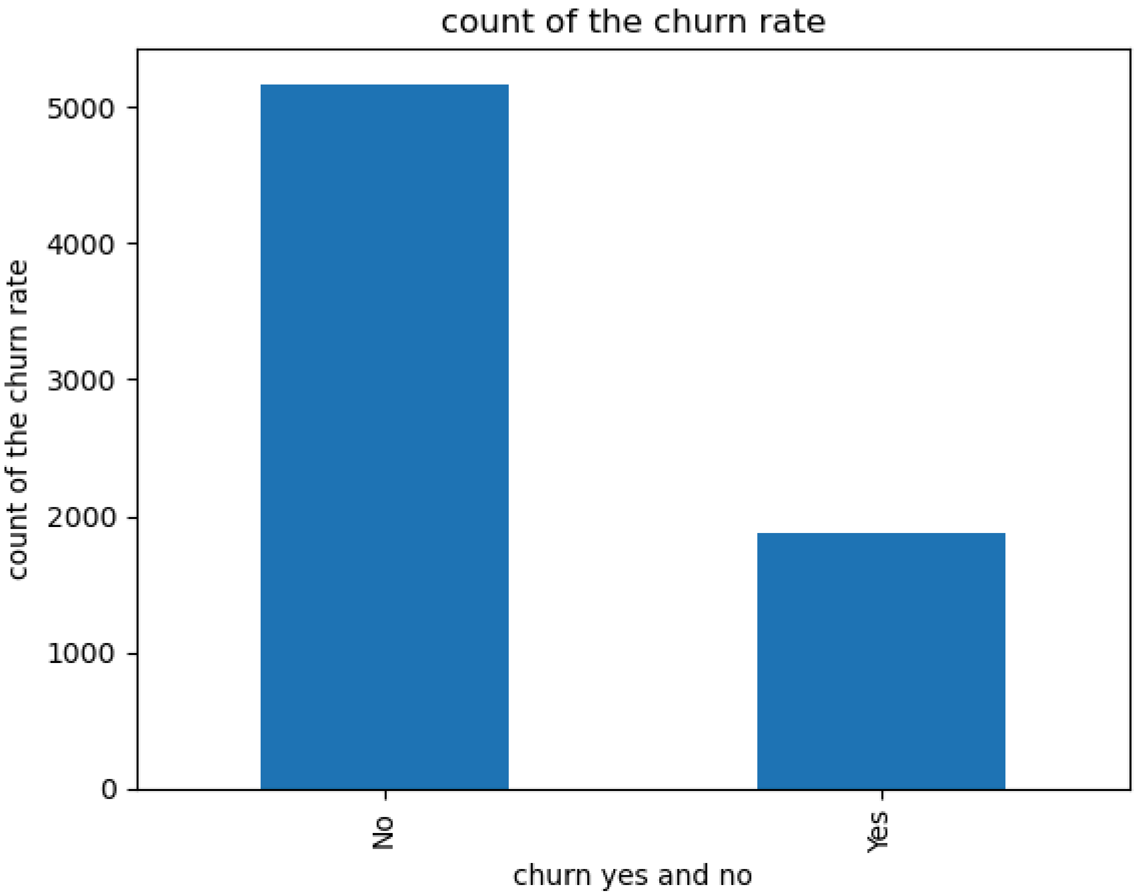
	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

In [9]:

```
# churn rate

data["Churn"].value_counts().plot(kind="bar")
plt.xlabel("churn yes and no")
plt.ylabel("count of the churn rate")
plt.title("count of the churn rate")
```

Out[9]:

```
Text(0.5, 1.0, 'count of the churn rate')
```



In [10]:

```
# churn counts

data["Churn"].value_counts()
```

Out[10]:

```
Churn
No     5174
Yes    1869
Name: count, dtype: int64
```

In [11]:

```
# percentage of the churn rate

100 * data["Churn"].value_counts() / len(data["Churn"])
```

Out[11]:

```
Churn
No     73.463013
Yes    26.536987
Name: count, dtype: float64
```

In [12]:

```
# check our dataset and take random column has any nan or empty values or not

data["Churn"].hasnans
```

Out[12]:

```
False
```

In [13]:

```
# minimum tenures

data["tenure"].min()
```

Out[13]:

```
0
```

In [14]:

```
# max tenures

data["tenure"].max()
```

Out[14]:

```
72
```

In [15]:

```
# average tenures

data["tenure"].mean()
```

Out[15]:

```
32.37114865824223
```

In [16]:

```
# most common tenures

data["tenure"].mode()
```

Out[16]:

```
0    1
Name: tenure, dtype: int64
```

In [17]:

```
# most common internet services so we can identify which service is public prefer

data["InternetService"].mode()
```

Out[17]:

```
0    Fiber optic
Name: InternetService, dtype: object
```

In [18]:

```
# most of the customer which type of contracts have

data["Contract"].mode()
```

Out[18]:

```
0    Month-to-month
Name: Contract, dtype: object
```

In [19]:

```
# which type of payment method that customer used

data["PaymentMethod"].mode()
```

Out[19]:

```
0    Electronic check
Name: PaymentMethod, dtype: object
```

In [20]:

```
# let's count the customers genders

data["gender"].value_counts()
```

Out[20]:

```
gender
Male      3555
Female    3488
Name: count, dtype: int64
```

In [21]:

```
# let's count the partner

data["Partner"].value_counts()
```

Out[21]:

```
Partner
No     3641
Yes    3402
Name: count, dtype: int64
```

In [22]:

```
# let's count the Dependents

data["Dependents"].value_counts()
```

Out[22]:

```
Dependents
No     4933
Yes    2110
Name: count, dtype: int64
```

In [23]:

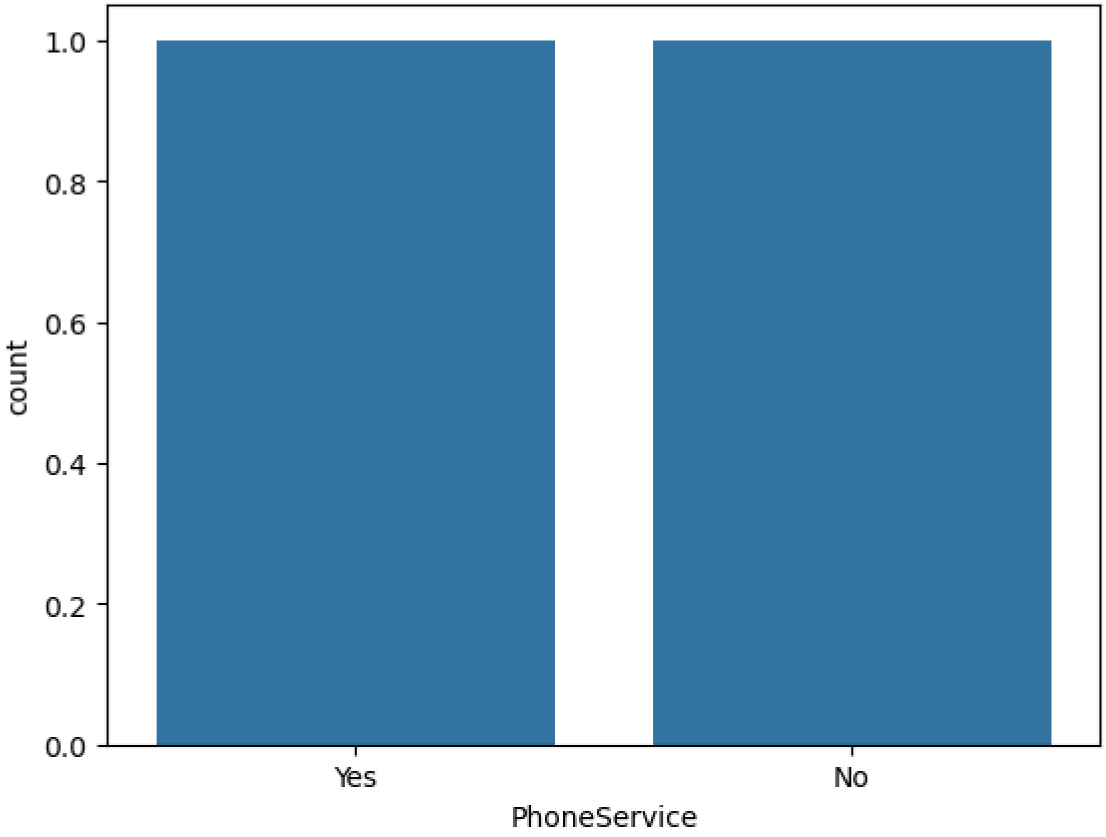
```
# let's count the total tenure

data["tenure"].sum()
```

Out[23]:

```
227990
```

In [24]:

```
# let's count the phoneservices

a = data.PhoneService.value_counts()

sns.countplot(data=a)
plt.show()
```



In [25]:
```
# lets count the multiplelines

data.MultipleLines.value_counts()
```

Out[25]:
```
MultipleLines
No                  3390
Yes                 2971
No phone service     682
Name: count, dtype: int64
```

In [26]:
```
# let's count the internet services

data.InternetService.value_counts()
```

Out[26]:

```
InternetService
Fiber optic    3096
DSL            2421
No             1526
Name: count, dtype: int64
```

In [27]:

```
# let's count the onlinesecurity services

data.OnlineSecurity.value_counts()
```

Out[27]:

```
OnlineSecurity
No                     3498
Yes                    2019
No internet service    1526
Name: count, dtype: int64
```

In [28]:

```
# let's count the onlinebackup services

data.OnlineBackup.value_counts()
```

Out[28]:

```
OnlineBackup
No                     3088
Yes                    2429
No internet service    1526
Name: count, dtype: int64
```

In [29]:

```
# let's count the deviceprotection services

data.DeviceProtection.value_counts()
```

Out[29]:

```
DeviceProtection
No                     3095
Yes                    2422
No internet service    1526
Name: count, dtype: int64
```

In [30]:

```
# let's count the TechSupport services

data.TechSupport.value_counts()
```

Out[30]:

```
TechSupport
No                     3473
Yes                    2044
No internet service    1526
Name: count, dtype: int64
```

In [31]:

```
# let's count the StreamingTV services

data.StreamingTV.value_counts()
```

Out[31]:

```
StreamingTV
No                     2810
Yes                    2707
No internet service    1526
Name: count, dtype: int64
```

In [32]:

```
# let's count the StreamingMovies services

data.StreamingMovies.value_counts()
```

Out[32]:

```
StreamingMovies
No                     2785
Yes                    2732
No internet service    1526
Name: count, dtype: int64
```

In [33]:

```
# let's count the Contract services

data.Contract.value_counts()
```

Out[33]:

```
Contract
Month-to-month    3875
Two year          1695
One year          1473
Name: count, dtype: int64
```

In [34]:

```
# let's count the PaperlessBilling services

data.PaperlessBilling.value_counts()
```

Out[34]:

```
PaperlessBilling
Yes    4171
No     2872
Name: count, dtype: int64
```

In [35]:

```
# let's count the PaymentMethod services

data.PaymentMethod.value_counts()
```

Out[35]:

```
PaymentMethod
Electronic check             2365
Mailed check                 1612
Bank transfer (automatic)    1544
Credit card (automatic)      1522
Name: count, dtype: int64
```

In [36]:

```
# let's count the MonthlyCharges services

data.MonthlyCharges.value_counts()
```

Out[36]:

```
MonthlyCharges
20.05     61
19.85     45
19.95     44
19.90     44
20.00     43
          ..
23.65      1
114.70     1
43.65      1
87.80      1
78.70      1
Name: count, Length: 1585, dtype: int64
```

In [37]:

```
# minimum MonthlyCharges

data.MonthlyCharges.min()
```

Out[37]:

```
18.25
```

In [38]:

```
# maximum MonthlyCharges

data.MonthlyCharges.max()
```

Out[38]:

```
118.75
```

In [39]:

```
# average mothlycharge()

data.MonthlyCharges.mean()
```

Out[39]:

```
64.76169246059918
```

In [40]:

```
# minimum TotalCharges

data.TotalCharges.min()
```

Out[40]:

```
' '
```

In [41]:

```
# maximum TotalCharges

data.TotalCharges.max()
```

Out[41]:

```
'999.9'
```

In [42]:

```
# check the type of totalcharges column data

data.TotalCharges.dtype
```

Out[42]:

```
dtype('O')
```

In [43]:

```
# let's count the churn customer

data.Churn.value_counts()
```

Out[43]:

```
Churn
No     5174
Yes    1869
Name: count, dtype: int64
```

In [44]:

```
# to check if the dataset has any blank or empty cell or values still present

data.isna()
```

Out[44]:

```
      customerID  gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService  MultipleLines  InternetService  OnlineSecurity  ...
   0       False   False          False    False       False   False         False          False            False           False  ...
   1       False   False          False    False       False   False         False          False            False           False  ...
   2       False   False          False    False       False   False         False          False            False           False  ...
   3       False   False          False    False       False   False         False          False            False           False  ...
   4       False   False          False    False       False   False         False          False            False           False  ...
 ...         ...     ...            ...      ...         ...     ...           ...            ...              ...             ...  ...
7038       False   False          False    False       False   False         False          False            False           False  ...
7039       False   False          False    False       False   False         False          False            False           False  ...
7040       False   False          False    False       False   False         False          False            False           False  ...
7041       False   False          False    False       False   False         False          False            False           False  ...
7042       False   False          False    False       False   False         False          False            False           False  ...

7043 rows × 21 columns
```

In [45]:

```
# let's find out the customer id whose minimum tenures

a = data.tenure.min()
b = data["customerID"][data["tenure"]==a]
b
```

```
Out[45]: 488     4472-LVYGI
         753     3115-CZMZD
         936     5709-LVOEQ
         1082    4367-NUYAO
         1340    1371-DWPAZ
         3331    7644-OMVMY
         3826    3213-VVOLG
         4380    2520-SGTTA
         5218    2923-ARZLG
         6670    4075-WKNIU
         6754    2775-SEFEE
         Name: customerID, dtype: object
```

```
In [46]: # let's find out the customer id,gender whose maximum tenures

         a = data.tenure.max()
         b = data[["customerID","gender"]][data["tenure"]==a]
         b
```

Out[46]:

	customerID	gender
28	5248-YGIJN	Male
35	6234-RAAPL	Female
59	5954-BDFSG	Female
62	0526-SXDJP	Male
94	9848-JQJTX	Male
...	...	...
6982	8468-FZTOE	Female
7007	2274-XUATA	Male
7022	7203-OYKCT	Male
7037	2569-WGERO	Female
7039	2234-XADUH	Female

362 rows × 2 columns

In [47]:

```
# let's find out the min monthlycharges customer id,gender

a = data.MonthlyCharges.min()
b = data[["customerID","gender"]][data["MonthlyCharges"]==a]
b
```

Out[47]:

	customerID	gender
3719	6823-SIDFQ	Male

In [48]:

```
# let's find out the max monthlycharges customer id,gender

a = data.MonthlyCharges.max()
```

```
b = data[["customerID","gender"]][data["MonthlyCharges"]==a]
b
```

Out[48]:

	customerID	gender
4586	7569-NMZYQ	Female

In [49]:

```
# let's find out the min TotalCharges customer id,gender

a = data.TotalCharges.min()
b = data[["customerID","gender"]][data["TotalCharges"]==a]
b
```

Out[49]:

	customerID	gender
488	4472-LVYGI	Female
753	3115-CZMZD	Male
936	5709-LVOEQ	Female
1082	4367-NUYAO	Male
1340	1371-DWPAZ	Female
3331	7644-OMVMY	Male
3826	3213-VVOLG	Male
4380	2520-SGTTA	Female
5218	2923-ARZLG	Male
6670	4075-WKNIU	Female
6754	2775-SEFEE	Male

In [50]:

```
# let's find out the max TotalCharges customer id,gender

a = data.TotalCharges.max()
```

```
b = data[["customerID","gender"]][data["TotalCharges"]==a]
b
```

Out[50]:

	customerID	gender
2845	9093-FPDLG	Female

In [51]:

```
# let's count the gender and customer-id who already churned means yes
b = data["gender"] [data.Churn == "Yes"]
b.value_counts()
```

Out[51]:

```
gender
Female    939
Male      930
Name: count, dtype: int64
```

In [52]:

```
# let's count the gender and customer-id who not churned means no
b = data["gender"] [data.Churn == "No"]
b.value_counts()
```

Out[52]:

```
gender
Male      2625
Female    2549
Name: count, dtype: int64
```

In [53]:

```
# change the datatype of TotalCharges column

data['TotalCharges'] = data['TotalCharges'].replace(r'^\s*$', np.nan, regex=True)
data['TotalCharges'] = data['TotalCharges'].astype(float)
```

In [77]:

```
# count the null values in data['TotalCharges'] column

data['TotalCharges'].isnull().sum()
```

Out[77]:

```
11
```

In [80]:

```
# fill the null cells with means
```

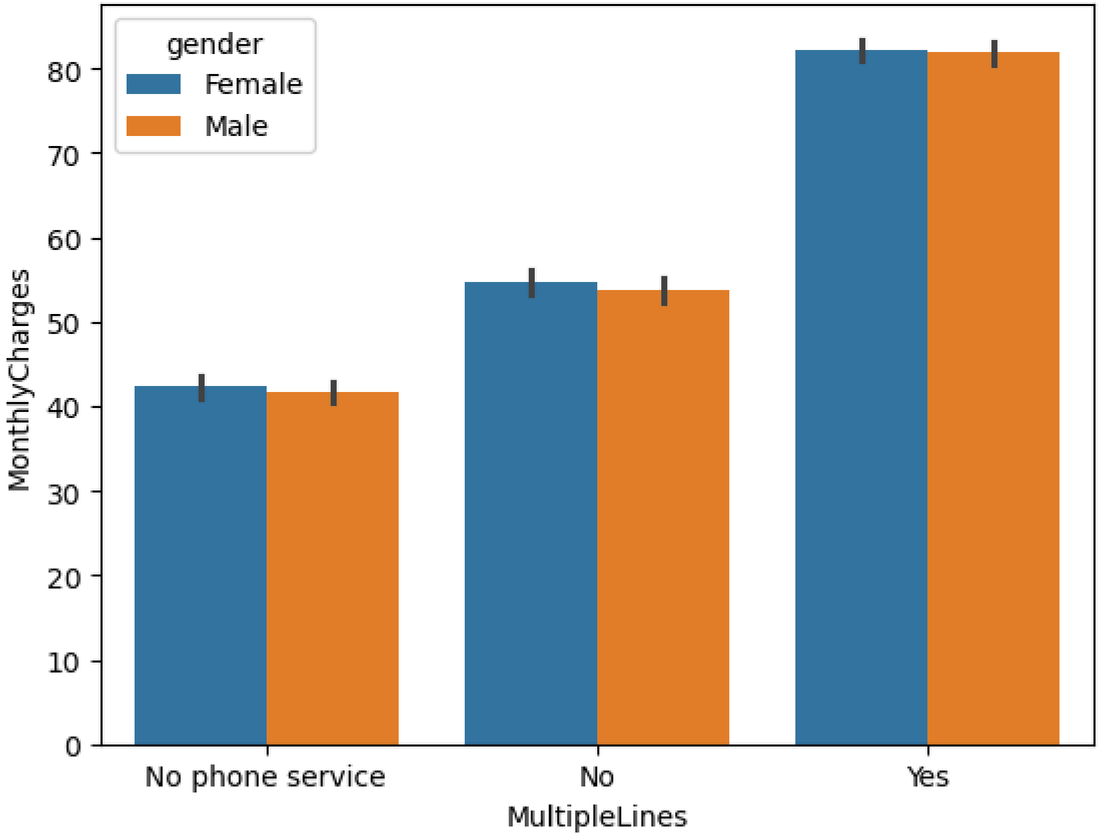
```
avg_totalcharges = data['TotalCharges'].mean()
data['TotalCharges'] = data['TotalCharges'].fillna(avg_totalcharges)
```

In [81]:
```
# count the null values in data['TotalCharges'] column

data['TotalCharges'].isnull().sum()
```

Out[81]:
```
0
```

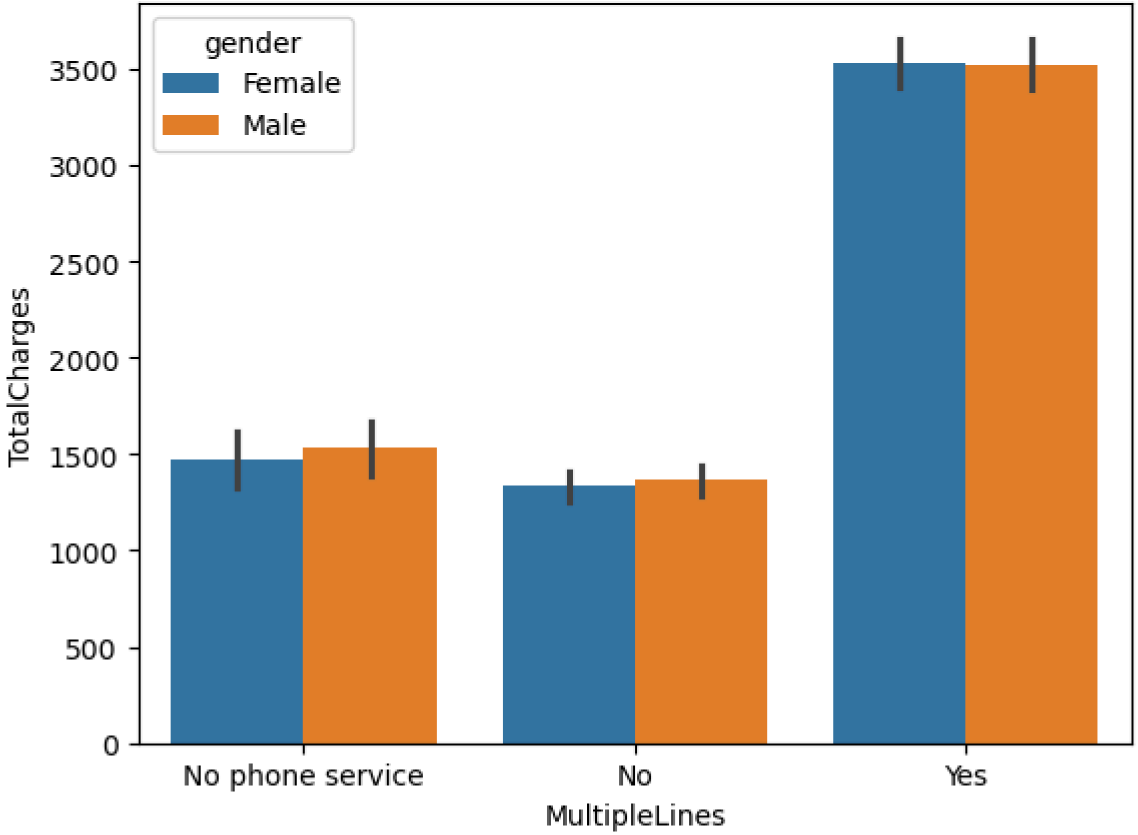
In [54]:
```
# MultipleLines wise monthly charges

sns.barplot(x="MultipleLines",y="MonthlyCharges",data=data,hue="gender")
```

Out[54]:
```
<Axes: xlabel='MultipleLines', ylabel='MonthlyCharges'>
```



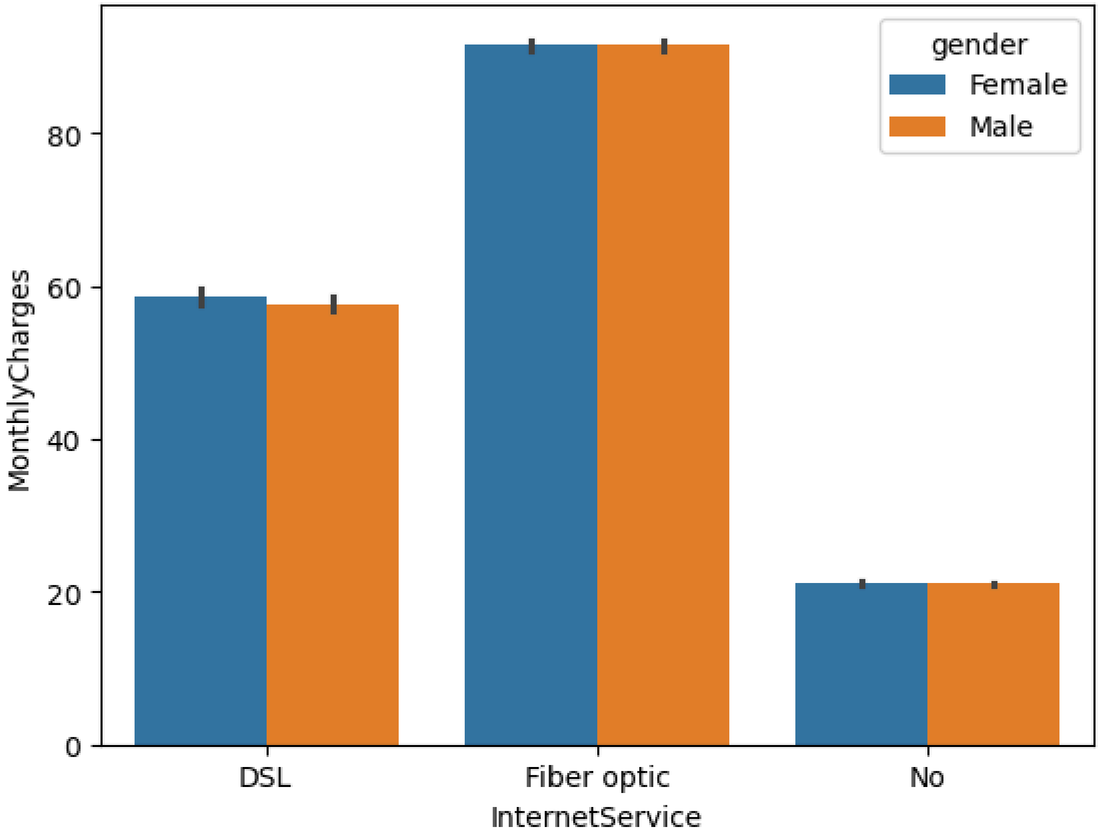
In [55]:
```
# MultipleLines wise total charges

sns.barplot(x="MultipleLines",y="TotalCharges",data=data,hue="gender")
```

Out[55]:
```
<Axes: xlabel='MultipleLines', ylabel='TotalCharges'>
```



In [56]:
```
# InternetService wise MonthlyCharges

sns.barplot(x="InternetService",y="MonthlyCharges",data=data,hue="gender")
```

Out[56]:
```
<Axes: xlabel='InternetService', ylabel='MonthlyCharges'>
```

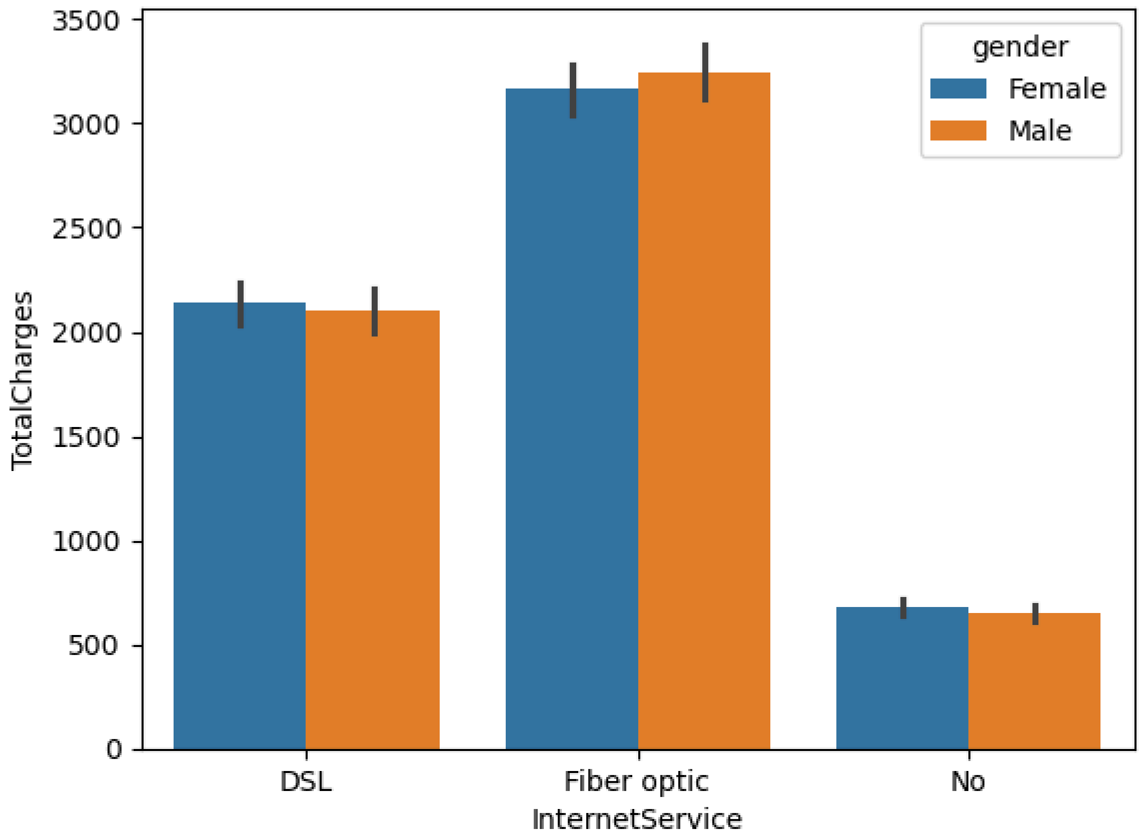


In [57]:

```
# InternetService wise total charges

sns.barplot(x="InternetService",y="TotalCharges",data=data,hue="gender")
```

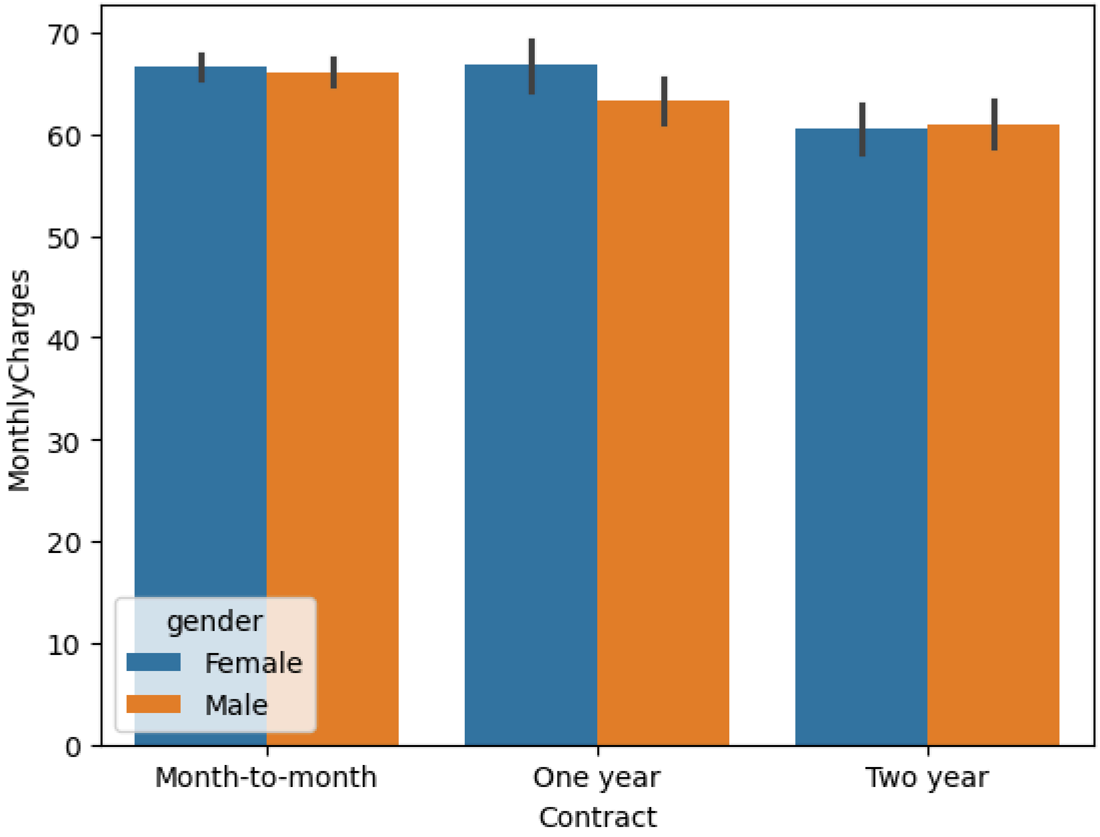
Out[57]:

```
<Axes: xlabel='InternetService', ylabel='TotalCharges'>
```



In [58]:
```
# Contract wise MonthlyCharges

sns.barplot(x="Contract",y="MonthlyCharges",data=data,hue="gender")
```

Out[58]:
```
<Axes: xlabel='Contract', ylabel='MonthlyCharges'>
```

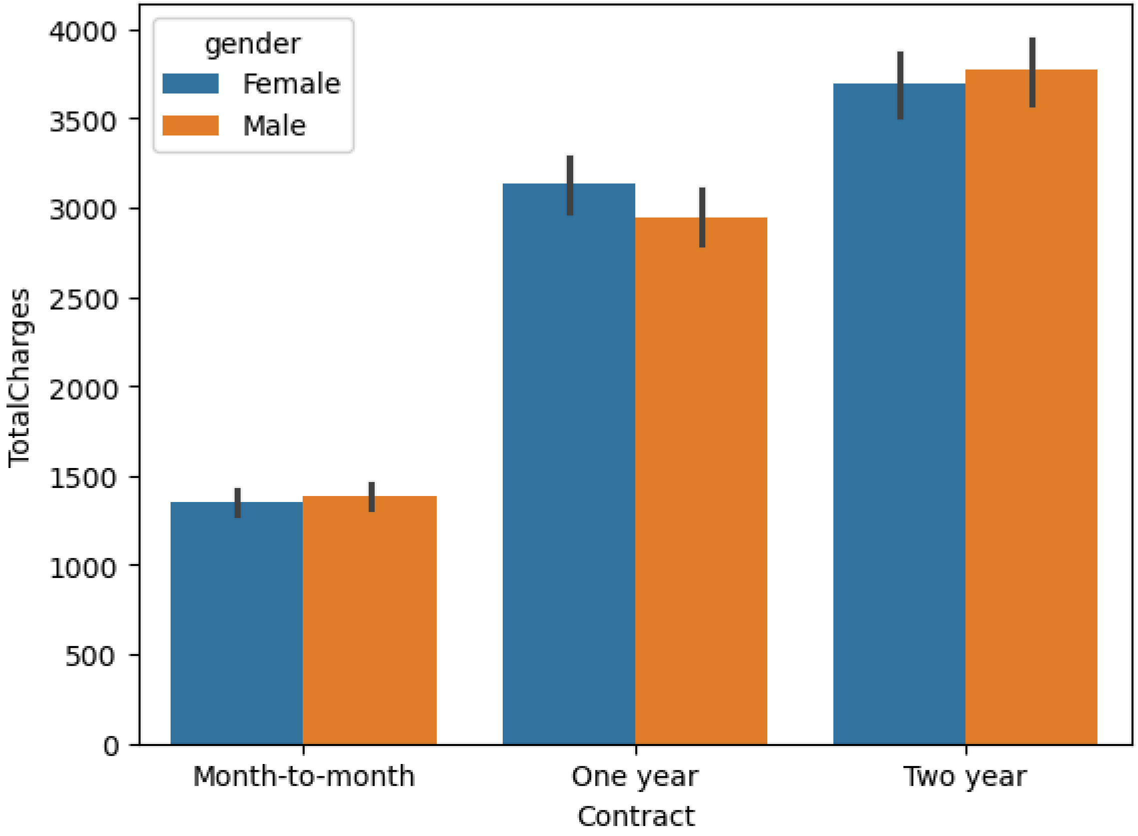


In [59]:

```
# Contract wise TotalCharges

sns.barplot(x="Contract",y="TotalCharges",data=data,hue="gender")
```

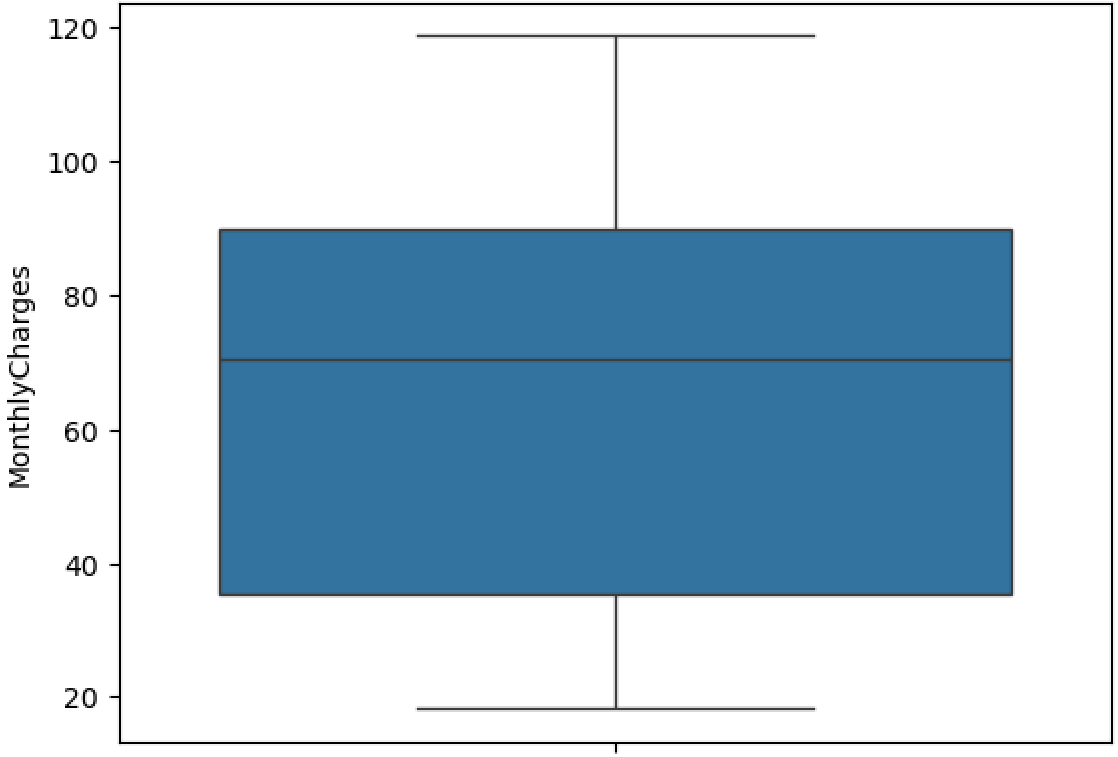
Out[59]:

```
<Axes: xlabel='Contract', ylabel='TotalCharges'>
```

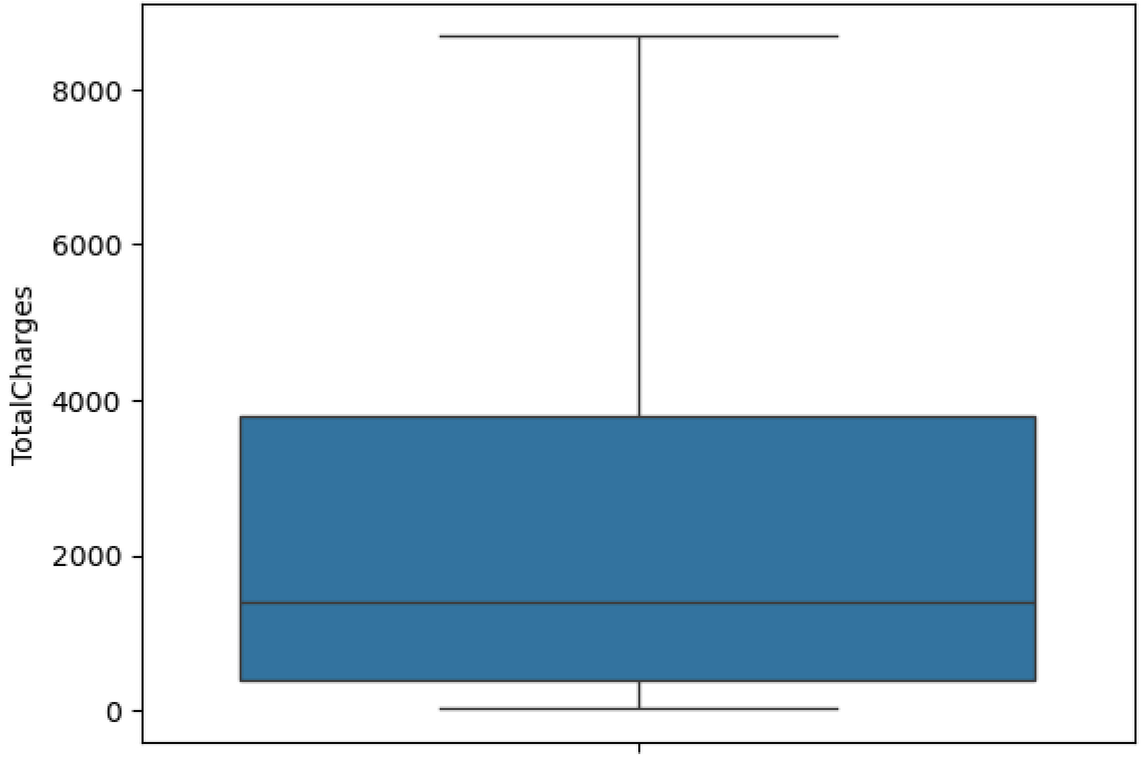


In [60]:

```
# let's check is there any outliers in our dataset

sns.boxplot(y="MonthlyCharges",data=data)
plt.show()
```

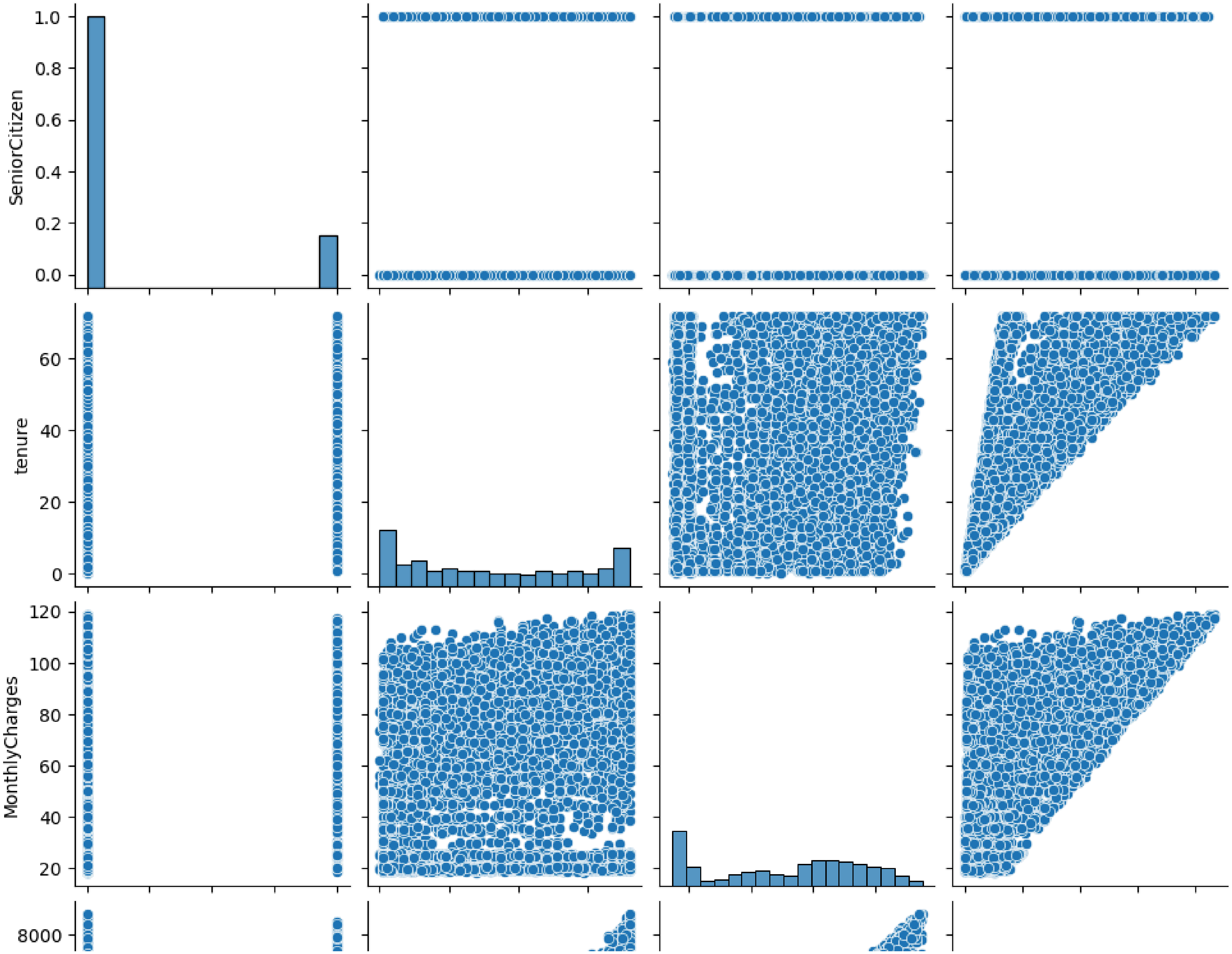


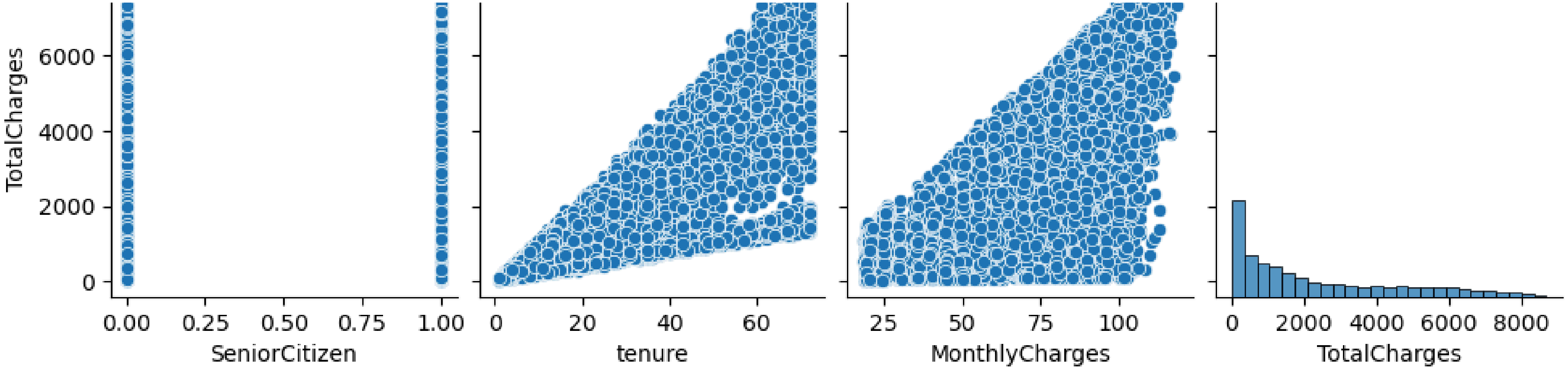
In [61]:
```
# let's check is there any outliers in our dataset

sns.boxplot(y="TotalCharges",data=data)
plt.show()
```



```
In [62]: # let's check the distribution in our dataset

         sns.pairplot(data=data)
```

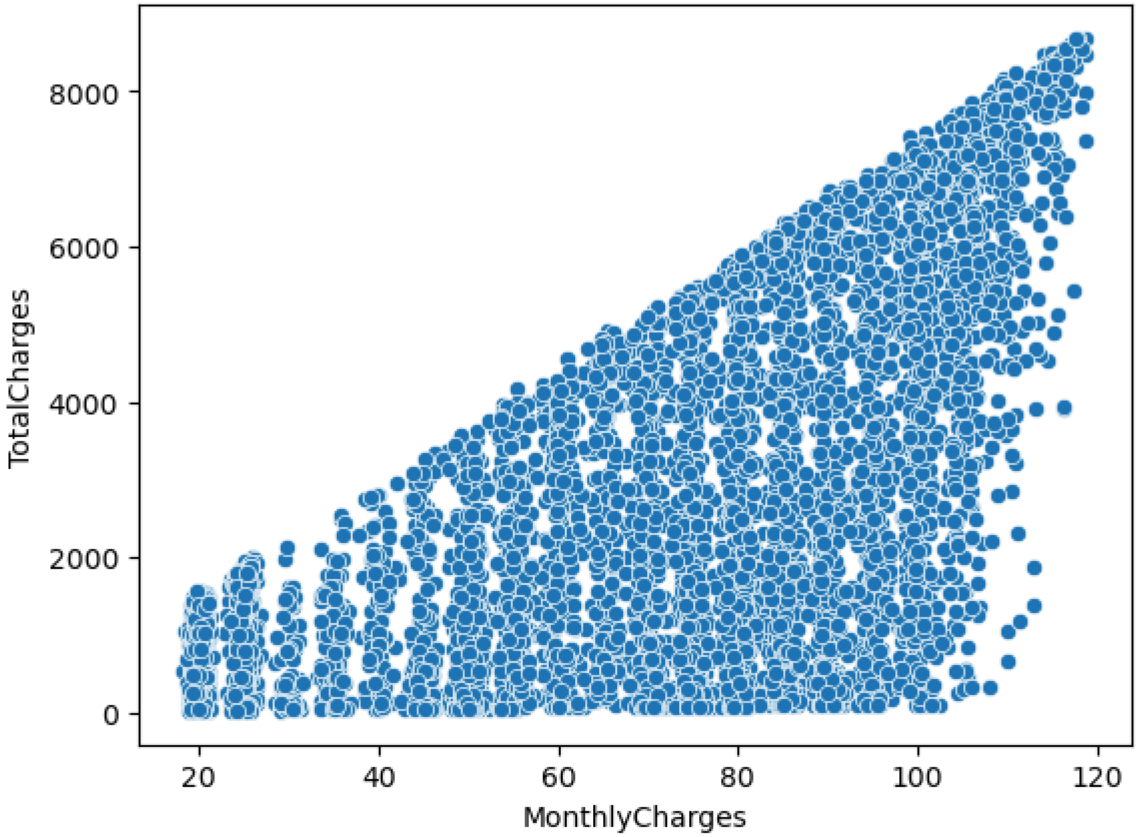
```
Out[62]: <seaborn.axisgrid.PairGrid at 0x220c2205850>
```


SeniorCitizen
1.0
0.8
0.6
0.4
0.2
0.0
tenure
60
40
20
0
MonthlyCharges
120
100
80
60
40
20
8000

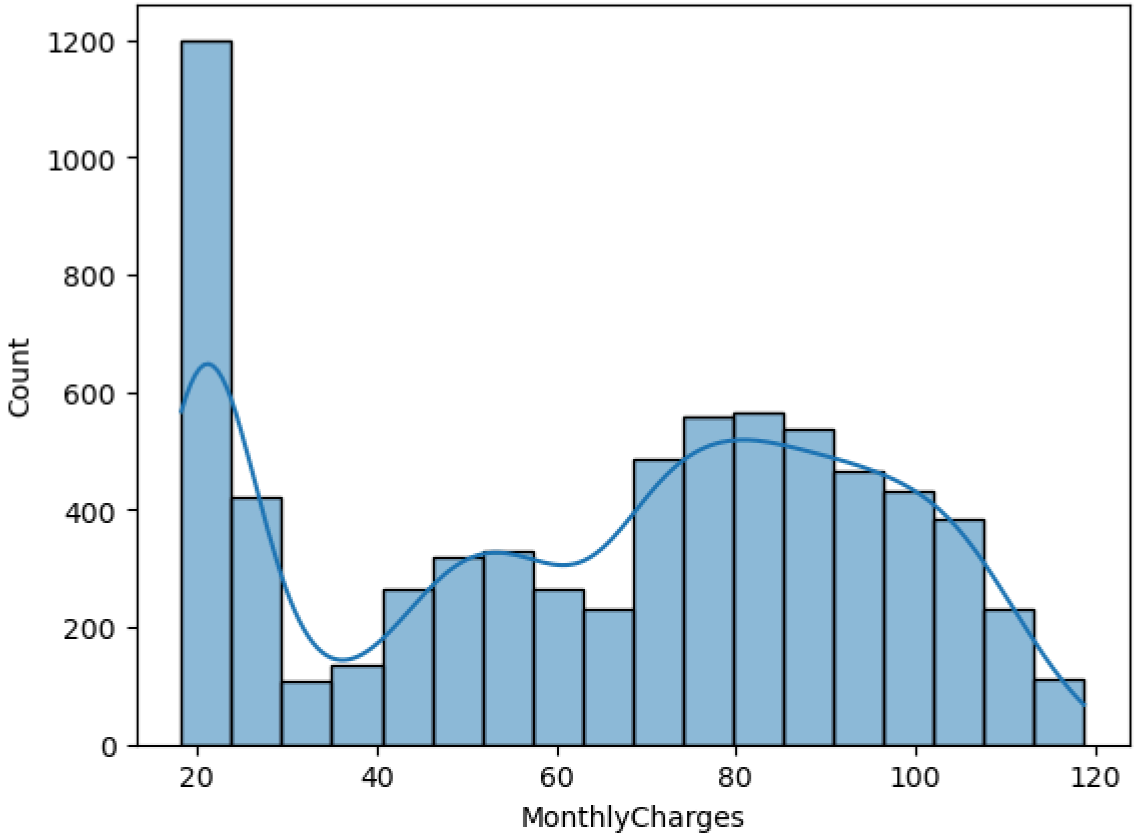


In [63]:

```
# let's check the relation ship betweeen two variables

sns.scatterplot(x="MonthlyCharges",y="TotalCharges",data=data)
plt.show()
```

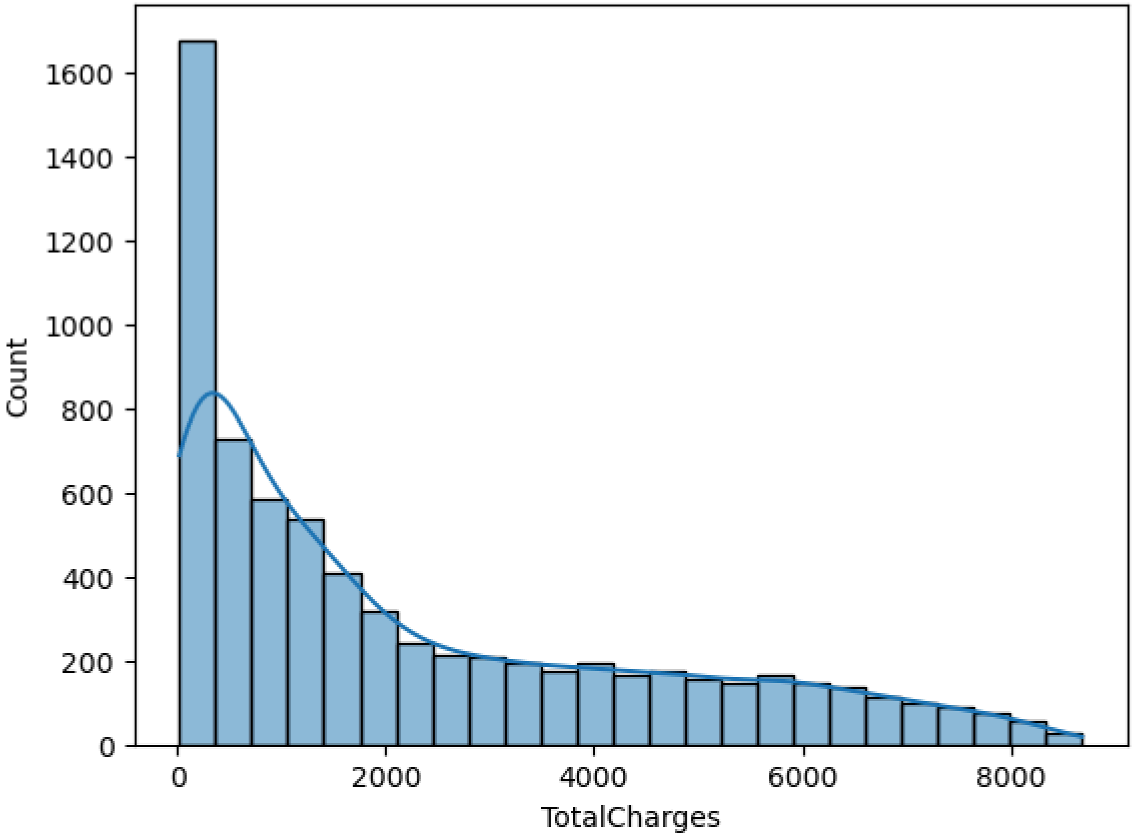


In [64]:
```
# to check the distribution

sns.histplot(x="MonthlyCharges",data=data,kde=True)
plt.show()
```



In [65]:

```
# to check the distribution

sns.histplot(x="TotalCharges",data=data,kde=True)
plt.show()
```



In [66]:

```
# column wise standard deviations

data.std(axis=1,skipna=True,numeric_only=True)
```

Out[66]:

```
0         16.950147
1        929.886015
2         51.238804
3        906.059353
4         71.686777
            ...
7038     977.768236
7039    3652.505658
7040     166.905998
7041     144.114445
7042    3393.921427
Length: 7043, dtype: float64
```

In [67]:

```
# row wise standard deviations

data.std(axis=0,skipna=True,numeric_only=True)
```

Out[67]:

```
SeniorCitizen        0.368612
tenure              24.559481
MonthlyCharges      30.090047
TotalCharges      2266.771362
dtype: float64
```

In [68]:

```
# column wise variance

data.var(axis=1,skipna=True,numeric_only=True)
```

Out[68]:

```
0       2.873075e+02
1       8.646880e+05
2       2.625415e+03
3       8.209436e+05
4       5.138994e+03
            ...
7038    9.560307e+05
7039    1.334080e+07
7040    2.785761e+04
7041    2.076897e+04
7042    1.151870e+07
Length: 7043, dtype: float64
```

In [69]:

```
# row wise variance

data.var(axis=0,skipna=True,numeric_only=True)
```

Out[69]:

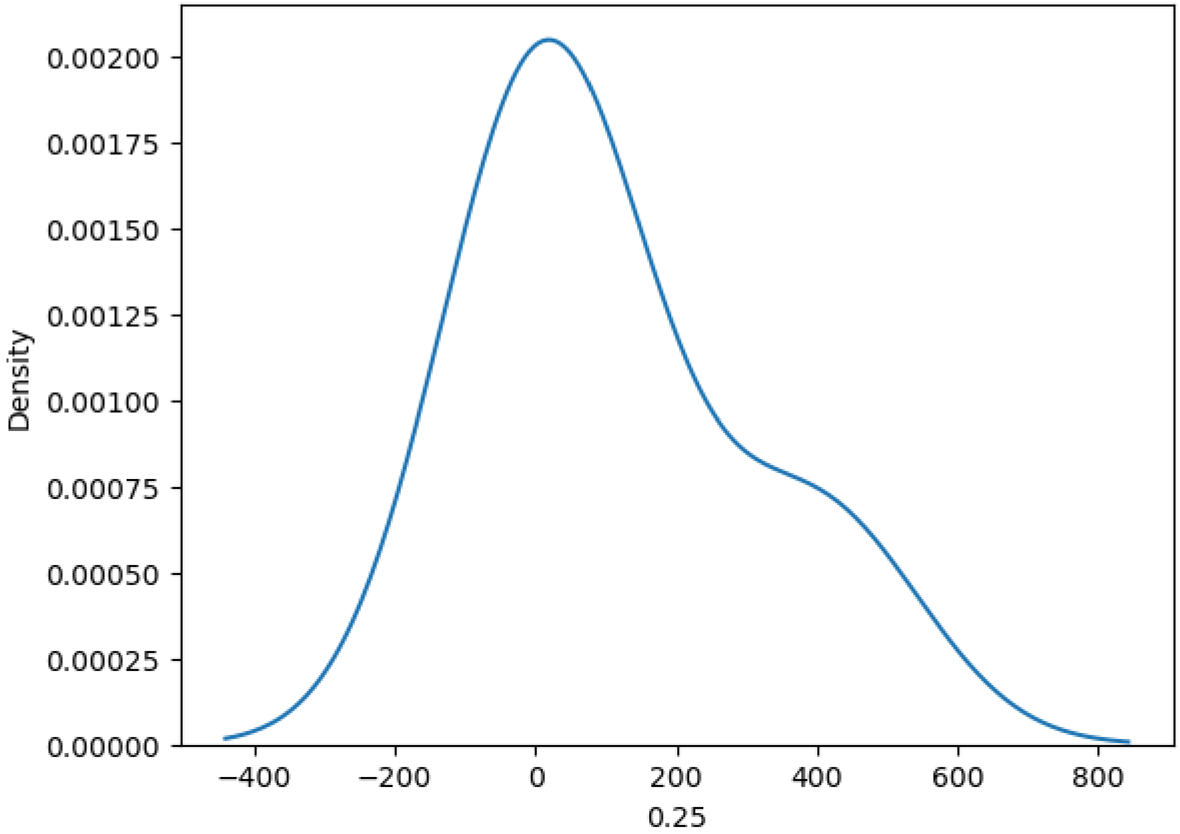
```
SeniorCitizen     1.358745e-01
tenure            6.031681e+02
MonthlyCharges    9.054109e+02
TotalCharges      5.138252e+06
dtype: float64
```

In [70]:

```
# let's check the bell shape curve means proper distribution curve is made by our dataset or not? row wise

a = data.quantile(q=0.25,axis=0,numeric_only=True)

sns.kdeplot(a)
```

Out[70]:

```
<Axes: xlabel='0.25', ylabel='Density'>
```

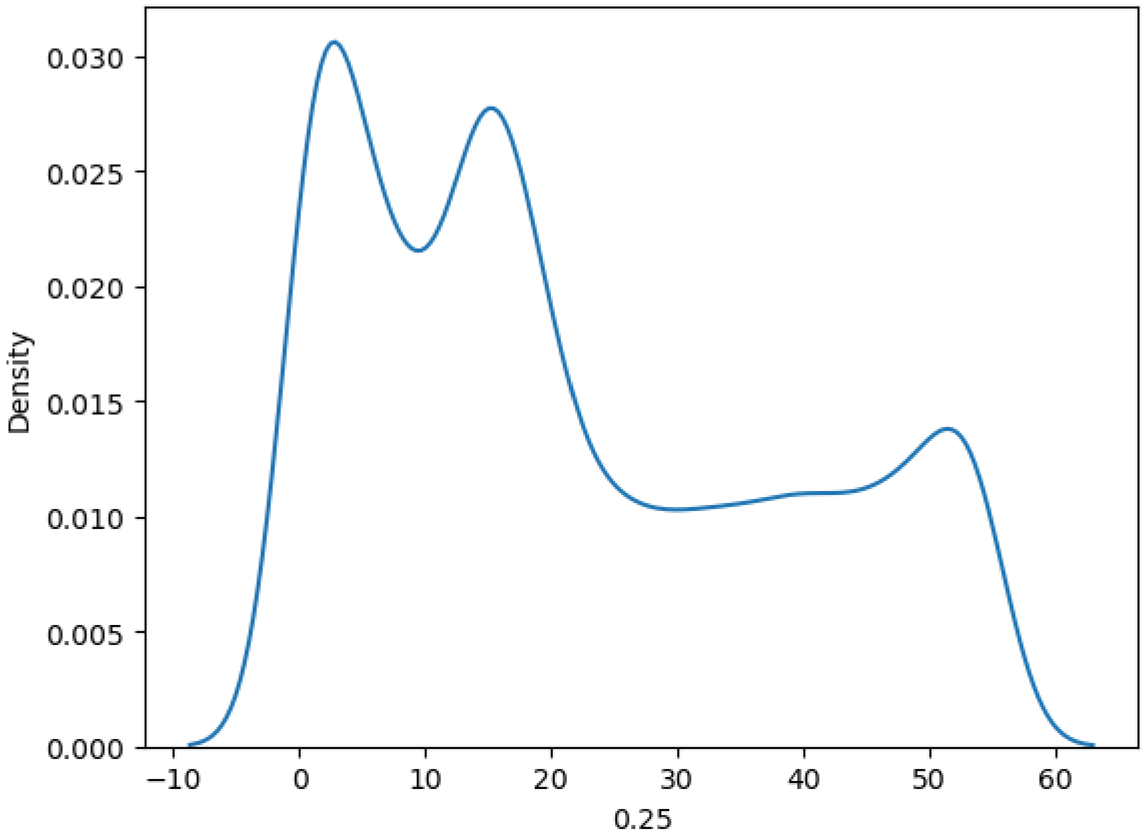


In [71]:

```
# let's check the bell shape curve means proper distribution curve is made by our dataset or not? column wise

a = data.quantile(q=0.25,axis=1,numeric_only=True)

sns.kdeplot(a)
```

Out[71]:

```
<Axes: xlabel='0.25', ylabel='Density'>
```



In [72]:

```
data.columns
```

Out[72]:

```
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 [73]:

```
# customerid has object datatype so we need to change the datatype first

data['customerID'].astype("category")
```

Out[73]:

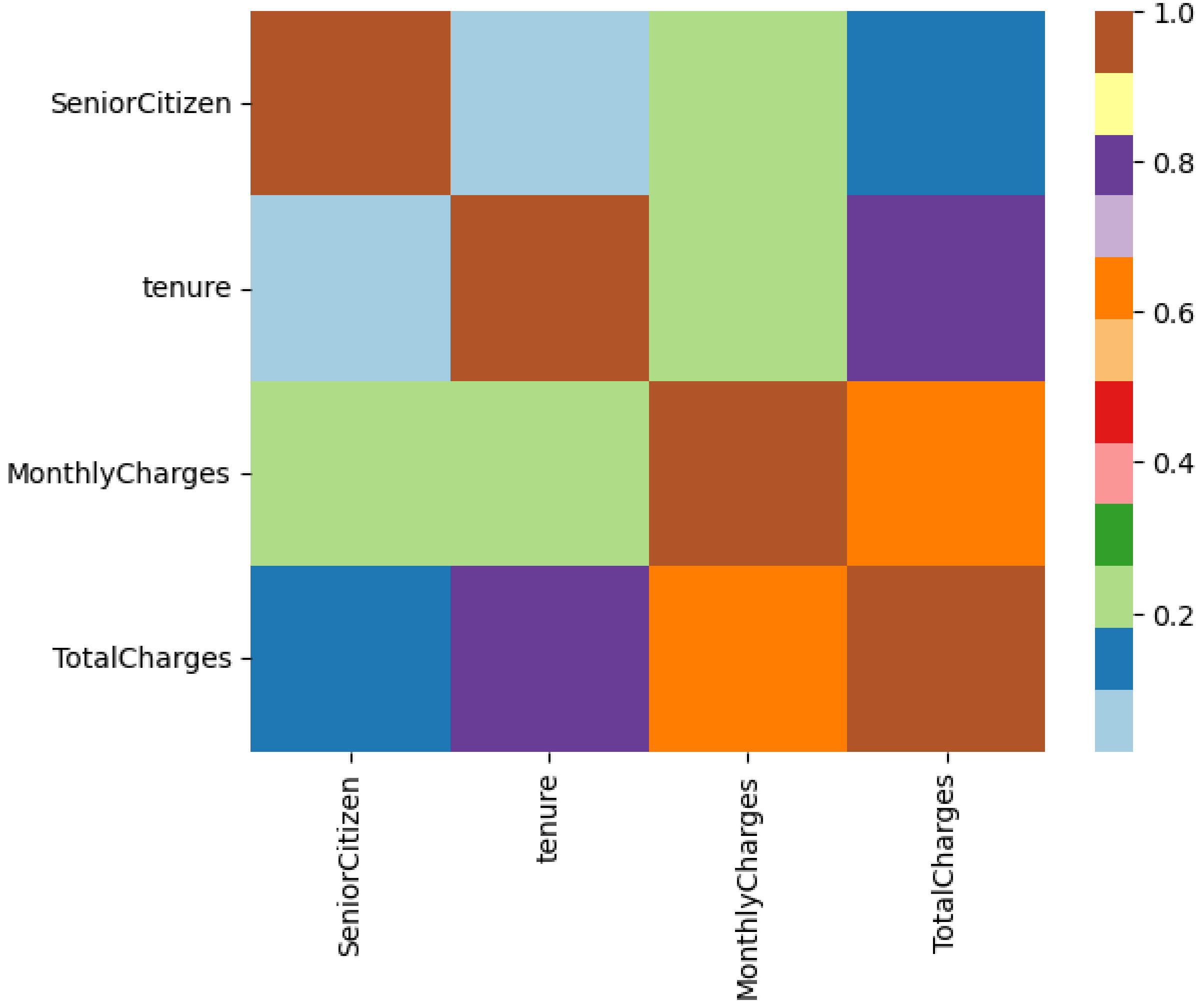
```
0       7590-VHVEG
1       5575-GNVDE
2       3668-QPYBK
3       7795-CFOCW
4       9237-HQITU
           ...
7038    6840-RESVB
7039    2234-XADUH
7040    4801-JZAZL
7041    8361-LTMKD
7042    3186-AJIEK
Name: customerID, Length: 7043, dtype: category
Categories (7043, object): ['0002-ORFBO', '0003-MKNFE', '0004-TLHLJ', '0011-IGKFF', ..., '9992-RRAMN', '9992-UJOEL', '9993-LH
IEB', '9995-HOTOH']
```

In [74]:

```python
# heatmap for the correlation of the data

sns.heatmap(data.corr(numeric_only = True),cmap="Paired")
```

Out[74]:

```
<Axes: >
```


SeniorCitizen
tenure
MonthlyCharges
TotalCharges
SeniorCitizen
tenure
MonthlyCharges
TotalCharges
1.0
0.8
0.6
0.4
0.2