In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv(r"C:\Users\HOME\Desktop\WA_Fn-UseC_-Telco-Customer-Churn.csv")
```

In [3]:
```
df.head()
```

Out[3]:

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

5 rows × 21 columns

In [4]:
```
df=df.drop("customerID",axis=1)
```

In [5]:
```
df.head()
```

Out[5]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechS
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No	
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Yes	
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No	

In [6]:
```
df.info()
```

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

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

Out[7]:

	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 [8]:
```
df["TotalCharges"]=pd.to_numeric(df["TotalCharges"])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
File ~\anaconda3\lib\site-packages\pandas\_libs\lib.pyx:2369, in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string " "

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
Cell In[8], line 1
----> 1 df["TotalCharges"]=pd.to_numeric(df["TotalCharges"])

File ~\anaconda3\lib\site-packages\pandas\core\tools\numeric.py:185, in to_numeric(arg, errors, downcast)
    183 coerce_numeric = errors not in ("ignore", "raise")
    184 try:
--> 185     values, _ = lib.maybe_convert_numeric(
    186         values, set(), coerce_numeric=coerce_numeric
    187     )
    188 except (ValueError, TypeError):
    189     if errors == "raise":

File ~\anaconda3\lib\site-packages\pandas\_libs\lib.pyx:2411, in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string " " at position 488
```

In [9]:
```
df["TotalCharges"].nunique()
```

Out[9]: 6531

In [10]:
```
df[df["TotalCharges"]==" "].shape
```

Out[10]: (11, 20)

In [11]:
```
df.shape
```

Out[11]: (7043, 20)

In [12]:
```
df_final=df[df["TotalCharges"]!=" "]
```

In [13]:
```
df_final.shape
```

Out[13]: (7032, 20)

In [14]:
```
df_final.isnull().sum()
```

Out[14]:
```
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 [15]:
```
df_final["TotalCharges"]=pd.to_numeric(df_final["TotalCharges"])
```

```
C:\Users\HOME\AppData\Local\Temp\ipykernel_10456\1716929626.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_final["TotalCharges"]=pd.to_numeric(df_final["TotalCharges"])
```

In [16]:
```
df_final["TotalCharges"].dtype
```

Out[16]:
```
dtype('float64')
```

In [17]:
```
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7032 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   gender            7032 non-null   object 
 1   SeniorCitizen     7032 non-null   int64  
 2   Partner           7032 non-null   object 
 3   Dependents        7032 non-null   object 
 4   tenure            7032 non-null   int64  
 5   PhoneService      7032 non-null   object 
 6   MultipleLines     7032 non-null   object 
 7   InternetService   7032 non-null   object 
 8   OnlineSecurity    7032 non-null   object 
 9   OnlineBackup      7032 non-null   object 
 10  DeviceProtection  7032 non-null   object 
 11  TechSupport       7032 non-null   object 
 12  StreamingTV       7032 non-null   object 
 13  StreamingMovies   7032 non-null   object 
 14  Contract          7032 non-null   object 
 15  PaperlessBilling  7032 non-null   object 
 16  PaymentMethod     7032 non-null   object 
 17  MonthlyCharges    7032 non-null   float64
 18  TotalCharges      7032 non-null   float64
 19  Churn             7032 non-null   object 
dtypes: float64(2), int64(2), object(16)
memory usage: 1.1+ MB
```

In [18]:
```
df_final.head()
```

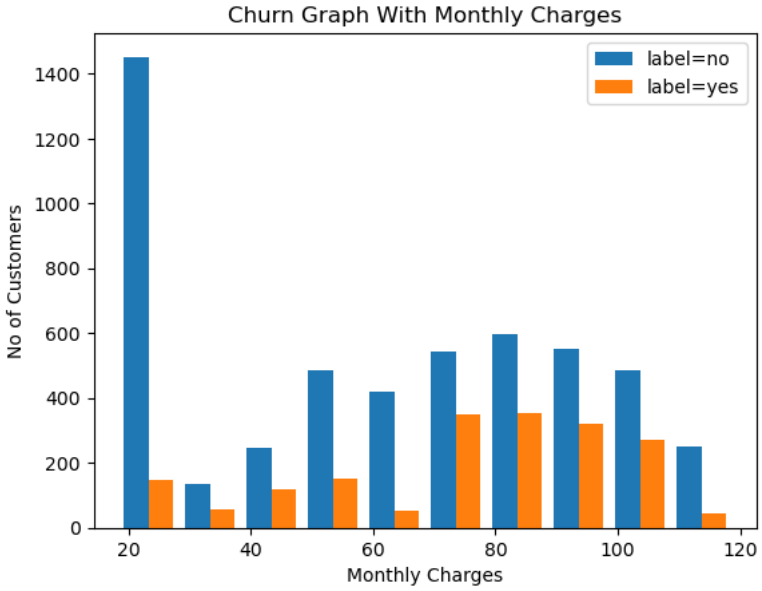
Out[18]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechS
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No	
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Yes	
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No	

In [19]:
```
MonthlyCharges_Churn_no=df_final[df_final["Churn"]=="No"].MonthlyCharges
MonthlyCharges_Churn_yes=df_final[df_final["Churn"]=="Yes"].MonthlyCharges
```

In [20]:
```
plt.hist([MonthlyCharges_Churn_no,MonthlyCharges_Churn_yes],label=["label=no","label=yes"])
plt.xlabel("Monthly Charges")
plt.ylabel("No of Customers")
plt.legend()
plt.title("Churn Graph With Monthly Charges")
```

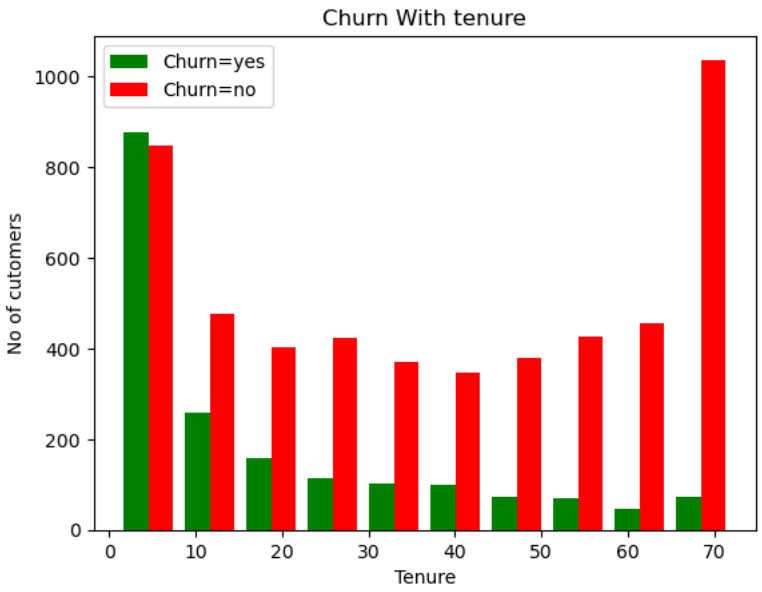
Out[20]: Text(0.5, 1.0, 'Churn Graph With Monthly Charges')



In [21]:
```
Tenure_CHurn_no=df_final[df_final["Churn"]=="No"].tenure
Tenure_CHurn_yes=df_final[df_final["Churn"]=="Yes"].tenure
```

In [22]:
```
plt.hist([Tenure_CHurn_yes,Tenure_CHurn_no],color=["green","red"],label=["Churn=yes","Churn=no"])
plt.xlabel("Tenure")
plt.ylabel("No of cutomers")
plt.legend()
plt.title("Churn With tenure")
```

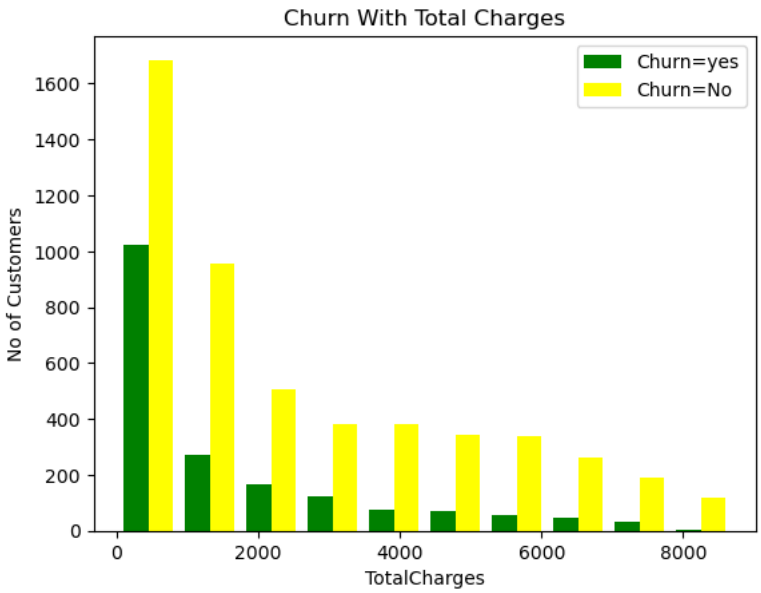
Out[22]: Text(0.5, 1.0, 'Churn With tenure')



In [23]:
```
TotalCharges_Churn_yes=df_final[df_final["Churn"]=="Yes"].TotalCharges
TotalCharges_Churn_no=df_final[df_final["Churn"]=="No"].TotalCharges
```

In [24]:
```
plt.hist([TotalCharges_Churn_yes,TotalCharges_Churn_no],color=["green","yellow"],label=["Churn=yes","Churn=No"])
plt.xlabel("TotalCharges")
plt.ylabel("No of Customers")
plt.legend()
plt.title("Churn With Total Charges")
```

Out[24]:
```
Text(0.5, 1.0, 'Churn With Total Charges')
```



In [25]:
```
df_final.corr()
```

```
C:\Users\HOME\AppData\Local\Temp\ipykernel_10456\2875322423.py:1: FutureWarning: The default value of numeric_only in DataF
rame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of n
umeric_only to silence this warning.
  df_final.corr()
```

Out[25]:

	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

In [26]:
```
df_final.columns
```

Out[26]:
```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod',
       'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

In [27]:
```
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7032 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   gender            7032 non-null   object 
 1   SeniorCitizen     7032 non-null   int64  
 2   Partner           7032 non-null   object 
 3   Dependents        7032 non-null   object 
 4   tenure            7032 non-null   int64  
 5   PhoneService      7032 non-null   object 
 6   MultipleLines     7032 non-null   object 
 7   InternetService   7032 non-null   object 
 8   OnlineSecurity    7032 non-null   object 
 9   OnlineBackup      7032 non-null   object 
 10  DeviceProtection  7032 non-null   object 
 11  TechSupport       7032 non-null   object 
 12  StreamingTV       7032 non-null   object 
 13  StreamingMovies   7032 non-null   object 
 14  Contract          7032 non-null   object 
 15  PaperlessBilling  7032 non-null   object 
 16  PaymentMethod     7032 non-null   object 
 17  MonthlyCharges    7032 non-null   float64 
 18  TotalCharges      7032 non-null   float64 
 19  Churn             7032 non-null   object 
dtypes: float64(2), int64(2), object(16)
memory usage: 1.1+ MB
```

In [28]:
```
for i in df_final.columns:
    if df_final[i].dtypes=="object":
        print(i,":",df_final[i].unique())
```

```
gender : ['Female' 'Male']
Partner : ['Yes' 'No']
Dependents : ['No' 'Yes']
PhoneService : ['No' 'Yes']
MultipleLines : ['No phone service' 'No' 'Yes']
InternetService : ['DSL' 'Fiber optic' 'No']
OnlineSecurity : ['No' 'Yes' 'No internet service']
OnlineBackup : ['Yes' 'No' 'No internet service']
DeviceProtection : ['No' 'Yes' 'No internet service']
TechSupport : ['No' 'Yes' 'No internet service']
StreamingTV : ['No' 'Yes' 'No internet service']
StreamingMovies : ['No' 'Yes' 'No internet service']
Contract : ['Month-to-month' 'One year' 'Two year']
PaperlessBilling : ['Yes' 'No']
PaymentMethod : ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
Churn : ['No' 'Yes']
```

In [29]:
```
Col_to_be_renamed=["StreamingMovies","StreamingTV","TechSupport","DeviceProtection","OnlineBackup","OnlineSecurity"]
```

In [30]:
```
for i in Col_to_be_renamed:
    if 'No internet service' in df_final[i].unique():
        df_final[i].replace('No internet service',"No",inplace=True)
```

```
C:\Users\HOME\AppData\Local\Temp\ipykernel_10456\1990900800.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vie
w-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_final[i].replace('No internet service',"No",inplace=True)
```

In [32]:
```
for j in Col_to_be_renamed:
    print(j,":",df_final[j].unique())
```

```
StreamingMovies : ['No' 'Yes']
StreamingTV : ['No' 'Yes']
TechSupport : ['No' 'Yes']
DeviceProtection : ['No' 'Yes']
OnlineBackup : ['Yes' 'No']
OnlineSecurity : ['No' 'Yes']
```

In [33]:
```
for i in df_final.columns:
    if df_final[i].dtypes=="object":
        print(i,":",df_final[i].unique())
```

```
gender : ['Female' 'Male']
Partner : ['Yes' 'No']
Dependents : ['No' 'Yes']
PhoneService : ['No' 'Yes']
MultipleLines : ['No phone service' 'No' 'Yes']
InternetService : ['DSL' 'Fiber optic' 'No']
OnlineSecurity : ['No' 'Yes']
OnlineBackup : ['Yes' 'No']
DeviceProtection : ['No' 'Yes']
TechSupport : ['No' 'Yes']
StreamingTV : ['No' 'Yes']
StreamingMovies : ['No' 'Yes']
Contract : ['Month-to-month' 'One year' 'Two year']
PaperlessBilling : ['Yes' 'No']
PaymentMethod : ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
Churn : ['No' 'Yes']
```

In [34]:
```
df_final["MultipleLines"]=df_final["MultipleLines"].astype("str")
```

```
C:\Users\HOME\AppData\Local\Temp\ipykernel_10456\3019708431.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vie
w-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_final["MultipleLines"]=df_final["MultipleLines"].astype("str")
```

In [35]:
```
df_final["MultipleLines"].replace('No phone service',"No",inplace=True)
```

```
C:\Users\HOME\AppData\Local\Temp\ipykernel_10456\3124602703.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vie
w-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_final["MultipleLines"].replace('No phone service',"No",inplace=True)
```

In [36]:
```
for i in df_final.columns:
    if df_final[i].dtypes=="object":
        print(i,":",df_final[i].unique())
```

```
gender : ['Female' 'Male']
Partner : ['Yes' 'No']
Dependents : ['No' 'Yes']
PhoneService : ['No' 'Yes']
MultipleLines : ['No' 'Yes']
InternetService : ['DSL' 'Fiber optic' 'No']
OnlineSecurity : ['No' 'Yes']
OnlineBackup : ['Yes' 'No']
DeviceProtection : ['No' 'Yes']
TechSupport : ['No' 'Yes']
StreamingTV : ['No' 'Yes']
StreamingMovies : ['No' 'Yes']
Contract : ['Month-to-month' 'One year' 'Two year']
PaperlessBilling : ['Yes' 'No']
PaymentMethod : ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
Churn : ['No' 'Yes']
```

In [37]:
```
for column in df_final:
    print(column,":",df_final[column].unique())
```

```
gender : ['Female' 'Male']
SeniorCitizen : [0 1]
Partner : ['Yes' 'No']
Dependents : ['No' 'Yes']
tenure : [ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26 39]
PhoneService : ['No' 'Yes']
MultipleLines : ['No' 'Yes']
InternetService : ['DSL' 'Fiber optic' 'No']
OnlineSecurity : ['No' 'Yes']
OnlineBackup : ['Yes' 'No']
DeviceProtection : ['No' 'Yes']
TechSupport : ['No' 'Yes']
StreamingTV : ['No' 'Yes']
StreamingMovies : ['No' 'Yes']
Contract : ['Month-to-month' 'One year' 'Two year']
PaperlessBilling : ['Yes' 'No']
PaymentMethod : ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
MonthlyCharges : [29.85 56.95 53.85 ... 63.1  44.2  78.7 ]
TotalCharges : [  29.85 1889.5   108.15 ...  346.45  306.6  6844.5 ]
Churn : ['No' 'Yes']
```

In [38]:
```
df_final["gender"]=df_final["gender"].apply(lambda x:0 if x=="Female" else 1)
```

```
C:\Users\HOME\AppData\Local\Temp\ipykernel_10456\2053909378.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vie
w-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_final["gender"]=df_final["gender"].apply(lambda x:0 if x=="Female" else 1)
```

In [40]:
```
df_final["gender"].value_counts()
```

Out[40]:
```
1    3549
0    3483
Name: gender, dtype: int64
```

In [41]:
```
for column in df_final:
    print(column,":",df_final[column].unique())
```

```
gender : [0 1]
SeniorCitizen : [0 1]
Partner : ['Yes' 'No']
Dependents : ['No' 'Yes']
tenure : [ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26 39]
PhoneService : ['No' 'Yes']
MultipleLines : ['No' 'Yes']
InternetService : ['DSL' 'Fiber optic' 'No']
OnlineSecurity : ['No' 'Yes']
OnlineBackup : ['Yes' 'No']
DeviceProtection : ['No' 'Yes']
TechSupport : ['No' 'Yes']
StreamingTV : ['No' 'Yes']
StreamingMovies : ['No' 'Yes']
Contract : ['Month-to-month' 'One year' 'Two year']
PaperlessBilling : ['Yes' 'No']
PaymentMethod : ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
MonthlyCharges : [29.85 56.95 53.85 ... 63.1  44.2  78.7 ]
TotalCharges : [  29.85 1889.5   108.15 ...  346.45  306.6  6844.5 ]
Churn : ['No' 'Yes']
```

In [42]:
```
yes_no=["Partner","Dependents","Churn","PaperlessBilling","StreamingMovies","StreamingTV","TechSupport","DeviceProtection","(
```

In [44]:
```
for i in yes_no:
    df_final[i]=df_final[i].apply(lambda x:1 if x=="Yes" else 0)
```

```
C:\Users\HOME\AppData\Local\Temp\ipykernel_10456\4171918526.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vie
w-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_final[i]=df_final[i].apply(lambda x:1 if x=="Yes" else 0)
```

In [45]:
```
for column in df_final:
    print(column,":",df_final[column].unique())
```

```
gender : [0 1]
SeniorCitizen : [0 1]
Partner : [1 0]
Dependents : [0 1]
tenure : [ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26 39]
PhoneService : [0 1]
MultipleLines : [0 1]
InternetService : ['DSL' 'Fiber optic' 'No']
OnlineSecurity : [0 1]
OnlineBackup : [1 0]
DeviceProtection : [0 1]
TechSupport : [0 1]
StreamingTV : [0 1]
StreamingMovies : [0 1]
Contract : ['Month-to-month' 'One year' 'Two year']
PaperlessBilling : [1 0]
PaymentMethod : ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
MonthlyCharges : [29.85 56.95 53.85 ... 63.1  44.2  78.7 ]
TotalCharges : [  29.85 1889.5   108.15 ...  346.45  306.6  6844.5 ]
Churn : [0 1]
```

In [46]:
```
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7032 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   gender            7032 non-null   int64
 1   SeniorCitizen     7032 non-null   int64
 2   Partner           7032 non-null   int64
 3   Dependents        7032 non-null   int64
 4   tenure            7032 non-null   int64
 5   PhoneService      7032 non-null   int64
 6   MultipleLines     7032 non-null   int64
 7   InternetService   7032 non-null   object
 8   OnlineSecurity    7032 non-null   int64
 9   OnlineBackup      7032 non-null   int64
 10  DeviceProtection  7032 non-null   int64
 11  TechSupport       7032 non-null   int64
 12  StreamingTV       7032 non-null   int64
 13  StreamingMovies   7032 non-null   int64
 14  Contract          7032 non-null   object
 15  PaperlessBilling  7032 non-null   int64
 16  PaymentMethod     7032 non-null   object
 17  MonthlyCharges    7032 non-null   float64
 18  TotalCharges      7032 non-null   float64
 19  Churn             7032 non-null   int64
dtypes: float64(2), int64(15), object(3)
memory usage: 1.1+ MB
```

In [47]:
```
onehot=["Contract","PaymentMethod","InternetService"]
```

In [48]:
```
df1=pd.get_dummies(data=df_final,columns=onehot)
```

In [51]:
```
df1.head()
```

Out[51]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	...	Contract_Month-to-month
0	0	0	1	0	1	0	0	0	1	0	...	1
1	1	0	0	0	34	1	0	1	0	1	...	0
2	1	0	0	0	2	1	0	1	1	0	...	1
3	1	0	0	0	45	0	0	1	0	1	...	0
4	0	0	0	0	2	1	0	0	0	0	...	1

5 rows × 27 columns

In [52]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7032 entries, 0 to 7042
Data columns (total 27 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   gender                                 7032 non-null   int64
 1   SeniorCitizen                          7032 non-null   int64
 2   Partner                                7032 non-null   int64
 3   Dependents                             7032 non-null   int64
 4   tenure                                 7032 non-null   int64
 5   PhoneService                           7032 non-null   int64
 6   MultipleLines                          7032 non-null   int64
 7   OnlineSecurity                         7032 non-null   int64
 8   OnlineBackup                           7032 non-null   int64
 9   DeviceProtection                       7032 non-null   int64
 10  TechSupport                            7032 non-null   int64
 11  StreamingTV                            7032 non-null   int64
 12  StreamingMovies                        7032 non-null   int64
 13  PaperlessBilling                       7032 non-null   int64
 14  MonthlyCharges                         7032 non-null   float64
 15  TotalCharges                           7032 non-null   float64
 16  Churn                                  7032 non-null   int64
 17  Contract_Month-to-month                7032 non-null   uint8
 18  Contract_One year                      7032 non-null   uint8
 19  Contract_Two year                      7032 non-null   uint8
 20  PaymentMethod_Bank transfer (automatic)  7032 non-null   uint8
 21  PaymentMethod_Credit card (automatic)    7032 non-null   uint8
 22  PaymentMethod_Electronic check         7032 non-null   uint8
 23  PaymentMethod_Mailed check             7032 non-null   uint8
 24  InternetService_DSL                    7032 non-null   uint8
 25  InternetService_Fiber optic            7032 non-null   uint8
 26  InternetService_No                     7032 non-null   uint8
dtypes: float64(2), int64(15), uint8(10)
memory usage: 1.0 MB
```

In [53]:
```
X=df1.drop("Churn",axis=1)
```

In [63]:
```
X1=df1.drop("Churn",axis=1).values
```

In [57]:
```
X.head()
```

Out[57]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	...	Contract_Month-to-month
0	0	0	1	0	1	0	0	0	1	0	...	1
1	1	0	0	0	34	1	0	1	0	1	...	0
2	1	0	0	0	2	1	0	1	1	0	...	1
3	1	0	0	0	45	0	0	1	0	1	...	0
4	0	0	0	0	2	1	0	0	0	0	...	1

5 rows × 26 columns

In [59]:
```
X.shape
```

Out[59]:
```
(7032, 26)
```

In [60]:
```
y=df1["Churn"]
```

In [61]:
```
y.head()
```

Out[61]:
```
0    0
1    0
2    1
3    0
4    1
Name: Churn, dtype: int64
```

In [62]:
```
y.shape
```

Out[62]:
```
(7032,)
```

In [64]:
```
X1
```

Out[64]:
```
array([[0., 0., 1., ..., 1., 0., 0.],
       [1., 0., 0., ..., 1., 0., 0.],
       [1., 0., 0., ..., 1., 0., 0.],
       ...,
       [0., 0., 1., ..., 1., 0., 0.],
       [1., 1., 1., ..., 0., 1., 0.],
       [1., 0., 0., ..., 0., 1., 0.]])
```

In [65]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score_score
```

In [66]:
```
sc=StandardScaler()
X_=sc.fit_transform(X)
```

In [69]:
```
X_
```

Out[69]:
```
array([[-1.00943013, -0.44032709,  1.03561683, ...,  1.38224311,
        -0.88689648, -0.52513044],
       [ 0.99065797, -0.44032709, -0.9656081 , ...,  1.38224311,
        -0.88689648, -0.52513044],
       [ 0.99065797, -0.44032709, -0.9656081 , ...,  1.38224311,
        -0.88689648, -0.52513044],
       ...,
       [-1.00943013, -0.44032709,  1.03561683, ...,  1.38224311,
        -0.88689648, -0.52513044],
       [ 0.99065797,  2.27103902,  1.03561683, ..., -0.72346173,
         1.12752731, -0.52513044],
       [ 0.99065797, -0.44032709, -0.9656081 , ..., -0.72346173,
         1.12752731, -0.52513044]])
```

In [145]:
```
X_train,X_test,y_train,y_test=train_test_split(X_,y,test_size=0.3,random_state=0)
```

In [146]:
```
train_score=lg.score(X_train,y_train)
train_score
```

Out[146]:
```
0.8073953677366924
```

In [147]:
```
test_score=lg.score(X_test,y_test)
test_score
```

Out[147]:
```
0.8042654028436019
```

In [148]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
```

In [188]:
```
lg=LogisticRegression(penalty="l2",solver='lbfgs')
lg.fit(X_train,y_train)
y_pred=lg.predict(X_test)
accuracy=accuracy_score(y_pred,y_test)
```

In [189]:
```
accuracy
```

Out[189]:
```
0.8004739336492891
```

In [198]:
```
rf=RandomForestClassifier(max_depth= 10, min_samples_leaf= 4, min_samples_split= 5, n_estimators= 100)
rf.fit(X_train,y_train)
y_pred_rf=rf.predict(X_test)
accuracy_rf=accuracy_score(y_pred_rf,y_test)
```

In [199]:
```
accuracy_rf
```

Out[199]:
```
0.7995260663507109
```

In [153]:
```
dt=DecisionTreeClassifier(criterion="entropy",splitter="best",min_samples_split=5)
dt.fit(X_train,y_train)
y_pred_dt=dt.predict(X_test)
accuracy_dt=accuracy_score(y_pred_dt,y_test)
```

In [154]:
```
accuracy_dt
```

Out[154]:
```
0.7298578199052133
```

In [155]:
```
cls=classification_report(y_pred,y_test)
print(cls)
```

```
              precision    recall  f1-score   support

           0       0.89      0.84      0.87      1648
           1       0.54      0.65      0.59       462

    accuracy                           0.80      2110
   macro avg       0.72      0.74      0.73      2110
weighted avg       0.82      0.80      0.81      2110
```

In [156]:
```
cf=confusion_matrix(y_pred,y_test)
```

In [157]:
```
print(cf)
```

```
[[1391  257]
 [ 164  298]]
```

In [158]:
```
import xgboost
```

In [160]:
```
from xgboost import XGBClassifier
```

In [161]:
```
xg=XGBClassifier()
xg.fit(X_train,y_train)
y_pred_xg=xg.predict(X_test)
Accuracy_xg=accuracy_score(y_pred,y_test)
```

In [162]:
```
Accuracy_xg
```

Out[162]:
```
0.8004739336492891
```

In [163]:
```
from sklearn.ensemble import AdaBoostClassifier
```

In [172]:
```
ad=AdaBoostClassifier(algorithm='SAMME')
ad.fit(X_train,y_train)
y_pred_ad=ad.predict(X_test)
Accuracy_ad=accuracy_score(y_pred_ad,y_test)
```

In [173]:
```
Accuracy_ad
```

Out[173]:
```
0.7985781990521327
```

In [174]:
```
from sklearn.model_selection import GridSearchCV
```

In [177]:
```
param_grid={
    "penalty":["l1","l2"],
    "solver":["lbfgs","liblinear","newton_cg","newton_cholesky","sag","saga"]
}
model=LogisticRegression()
```

In [179]:
```
grid_search_lg=GridSearchCV(estimator=model,param_grid=param_grid,cv=5,scoring="accuracy")
```

In [180]:
```
grid_search_lg.fit(X,y)
```

```
--------------------------------------------------------------------------------
5 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\HOME\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\HOME\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1162, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\HOME\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 54, in _check_solver
    raise ValueError(
ValueError: Solver sag supports only 'l2' or 'none' penalties, got l1 penalty.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\HOME\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:952: UserWarning: One or more of the test sc
ores are non-finite: [        nan 0.80346874         nan         nan         nan 0.76336828
 0.80418008 0.80389568         nan         nan 0.7694829  0.76336828]
  warnings.warn(

C:\Users\HOME\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to conv
erge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

In [184]:
```
best=grid_search_lg.best_params_
```

In [185]:
```
best
```

Out[185]:
```
{'penalty': 'l2', 'solver': 'lbfgs'}
```

In [190]:
```
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

In [191]:
```
grid_search_rf=GridSearchCV(param_grid=param_grid,scoring="accuracy",estimator=rf,cv=5)
```

In [192]:
```
grid_search_rf.fit(X,y)
```

Out[192]:
```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
  ▸ RandomForestClassifier
```

In [193]:
```
best=grid_search_rf.best_params_
```

In [194]:
```
print(best)
```

```
{'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 5, 'n_estimators': 100}
```

In [200]:
```
param = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'subsample': [0.8, 0.9, 1.0],
    'colsample_bytree': [0.8, 0.9, 1.0],
}
```

In [201]:
```
grid_xg=GridSearchCV(scoring="accuracy",estimator=xg,param_grid=param)
```

In [202]:
```
grid_xg.fit(X,y)
```

Out[202]:
```
▸ GridSearchCV
▸ estimator: XGBClassifier
  ▸ XGBClassifier
```

In [204]:
```
best=grid_xg.best_params_
```

In [205]:
```
print(best)
```

```
{'colsample_bytree': 0.8, 'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 50, 'subsample': 0.9}
```

In [206]:
```
xg=XGBClassifier(colsample_bytree=0.8,learning_rate=0.1,max_depth=3,n_estimators=50,subsample=0.9)
xg.fit(X_train,y_train)
y_pred_xg=xg.predict(X_test)
Accuracy_xg=accuracy_score(y_pred,y_test)
```

In [207]:
```
Accuracy_xg
```

Out[207]:
```
0.8004739336492891
```

In [ ]:
```

```