In [1]:
```python
import numpy as np
import pandas as pd
data=pd.read_csv('/Users/bbkpa/Downloads/churn_data.csv')
data
```

Out[1]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	Nc
1	5575-	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	Nc
...	...	...	...	...	...	...	...	...	...	..
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes
7039	2234-	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	Nc
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes
7041	8361-	Male	1	Yes	No	4	Yes	Yes	Fiber optic	Nc
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes

7043 rows × 21 columns

In [2]:
```python
data=data.drop(columns=['customerID'])
data
```

Out[2]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBack
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Y
1	Male	0	No	No	34	Yes	No	DSL	Yes	
2	Male	0	No	No	2	Yes	No	DSL	Yes	Y
3	Male	0	No	No	45	No	No phone service	DSL	Yes	
4	Female	0	No	No	2	Yes	No	Fiber optic	No	
...	...	...	...	...	...	...	...	...	...	
7038	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	
7039	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	Y
7040	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	
7041	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	
7042	Male	0	No	No	66	Yes	No	Fiber optic	Yes	

7043 rows × 20 columns

In [3]:
```python
data.shape
```

Out[3]: (7043, 20)

In [4]:
```python
data['SeniorCitizen'].value_counts()
```

Out[4]: SeniorCitizen
```
0    5901
1    1142
Name: count, dtype: int64
```

In [5]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

To check data type and null values in each column

In [6]:
```
data.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]:
```
data.columns
```

Out[7]:
```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod',
       'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

In [8]:
```
data['gender'].unique()
```

Out[8]:
```
array(['Female', 'Male'], dtype=object)
```

To check number of categories in each categorical column

In [9]:
```
numerical_col=['SeniorCitizen','tenure','MonthlyCharges']
for col in data.columns:
    if col not in numerical_col:
        print(col, data[col].unique())
        print("-"*100)
```

```
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)']
--------------------------------------------------------------------------------
TotalCharges ['29.85' '1889.5' '108.15' ... '346.45' '306.6' '6844.5']
--------------------------------------------------------------------------------
Churn ['No' 'Yes']
--------------------------------------------------------------------------------
```

In [10]:
```python
data[data['TotalCharges']==" "]
```

Out[10]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBack
488	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	
753	Male	0	No	Yes	0	Yes	No	No	No internet service	No inter serv
936	Female	0	Yes	Yes	0	Yes	No	DSL	Yes	Y
1082	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No inter serv
1340	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	Y
3331	Male	0	Yes	Yes	0	Yes	No	No	No internet service	No inter serv
3826	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No inter serv
4380	Female	0	Yes	Yes	0	Yes	No	No	No internet service	No inter serv
5218	Male	0	Yes	Yes	0	Yes	No	No	No internet service	No inter serv
6670	Female	0	Yes	Yes	0	Yes	Yes	DSL	No	Y
6754	Male	0	No	Yes	0	Yes	Yes	DSL	Yes	Y

In [11]:
```python
len(data[data['TotalCharges']==" "])
```

Out[11]: 11

We are replacing empty value in total charges column with 0

In [12]:
```python
data['TotalCharges']=data['TotalCharges'].replace(" ",0.0)
```

In [13]:
```python
data.isnull().sum()
```

```
Out[13]: 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 [14]: data['TotalCharges']=data['TotalCharges'].astype('float64')
```

```
In [15]: data.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   float64
 19  Churn             7043 non-null   object 

dtypes: float64(2), int64(2), object(16)
memory usage: 1.1+ MB
```

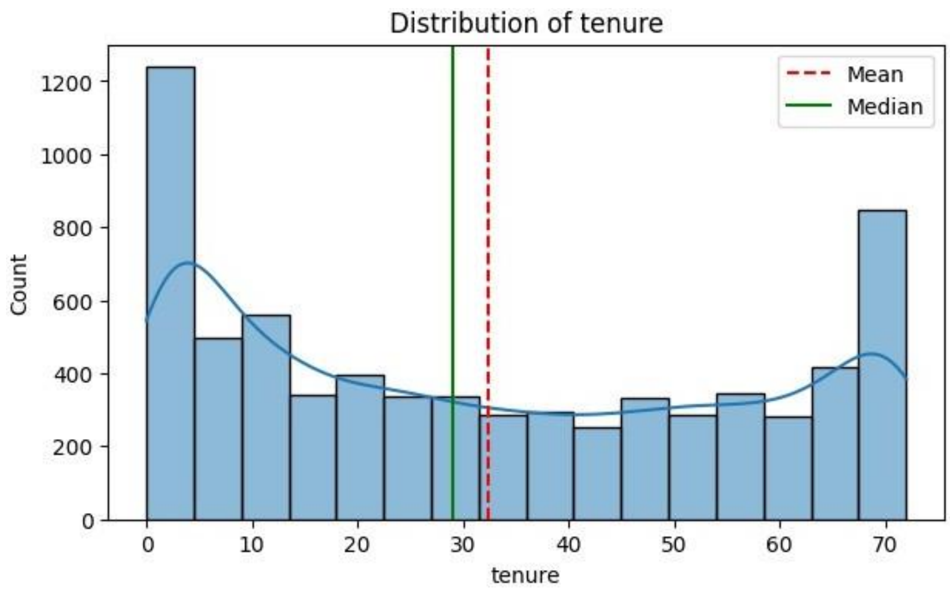
Exploratory Data Analysis

Histogram of all the numerical columns in the dataset

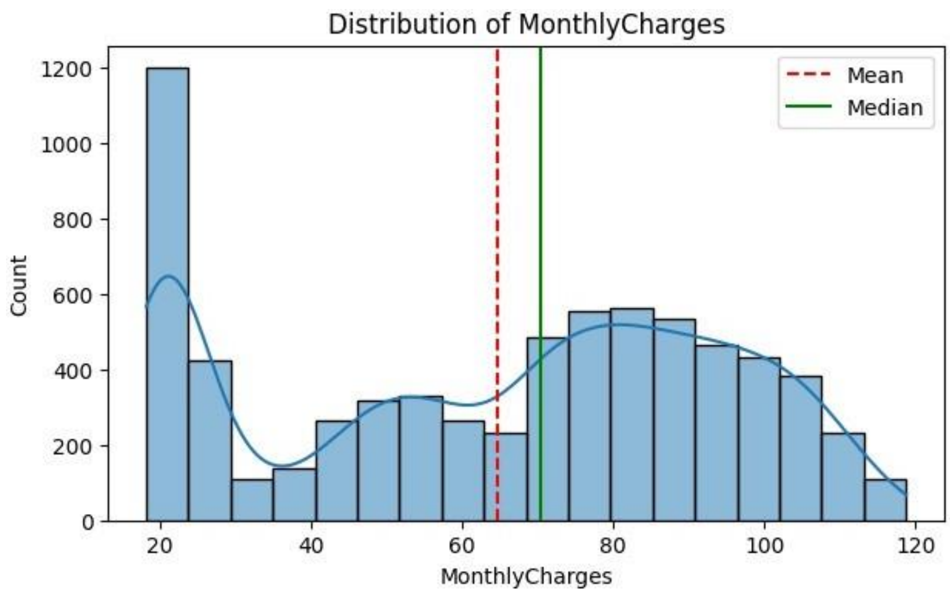
```
In [16]: import warnings
         warnings.filterwarnings('ignore')
```

```
In [17]: def plot_histogram(data,column):
             plt.figure(figsize=(7,4))
             sns.histplot(x=data[column],kde=True)
             plt.title(f"Distribution of {column}")
             # calculating each column mean and column median
             col_mean=data[column].mean()
             col_med=data[column].median()
             # drawing a horizontal line of col_mean and median
             plt.axvline(col_mean,color='red',linestyle='--',label='Mean')
             plt.axvline(col_med,color='green',linestyle='-',label='Median')
             plt.legend()
             plt.show()
```

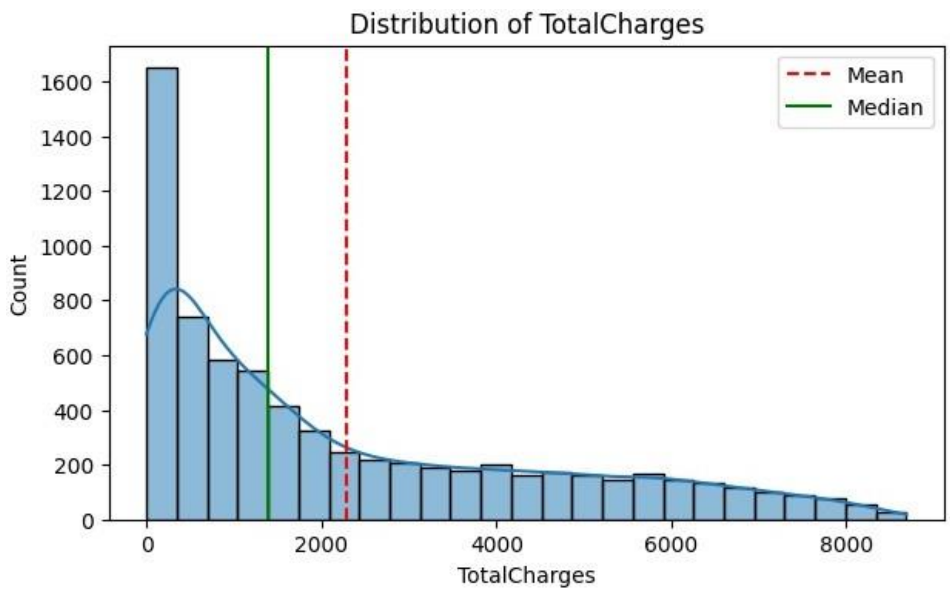
```
In [18]: plot_histogram(data,'tenure')
```



In [20]:
```
plot_histogram(data,'MonthlyCharges')
```



In [21]:
```
plot_histogram(data,'TotalCharges')
```

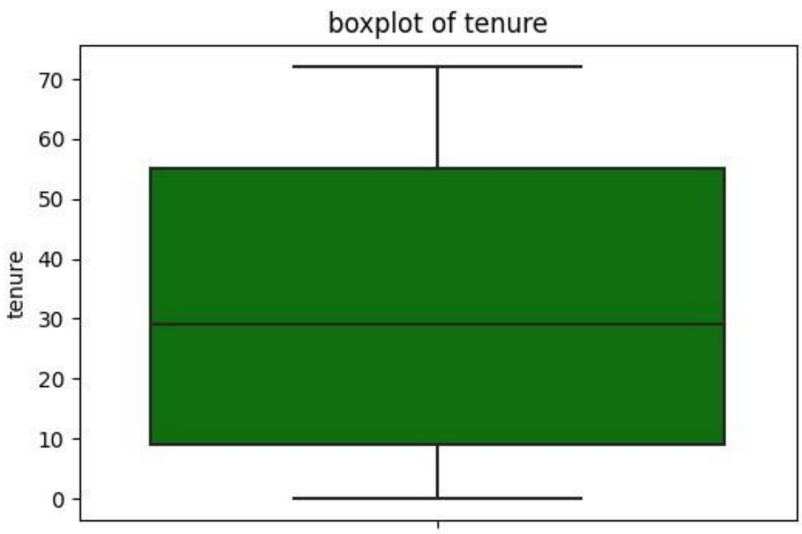


Boxplot of all the numerical columns in the dataset (checking for outliers)

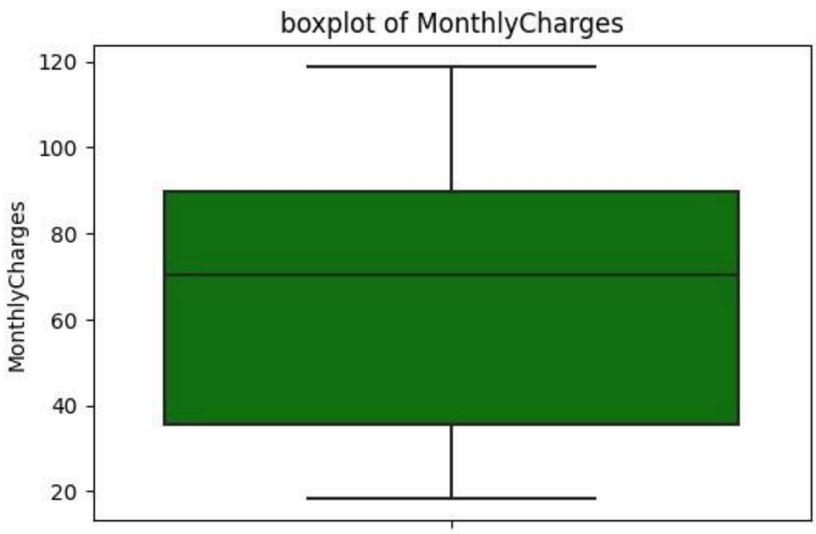
In [22]:
```
def plot_boxplot(data,column):
    plt.figure(figsize=(6,4))
    sns.boxplot(data=data,y=data[column],color='green')
    plt.title(f"boxplot of {column}")
```

```
    plt.show()
```

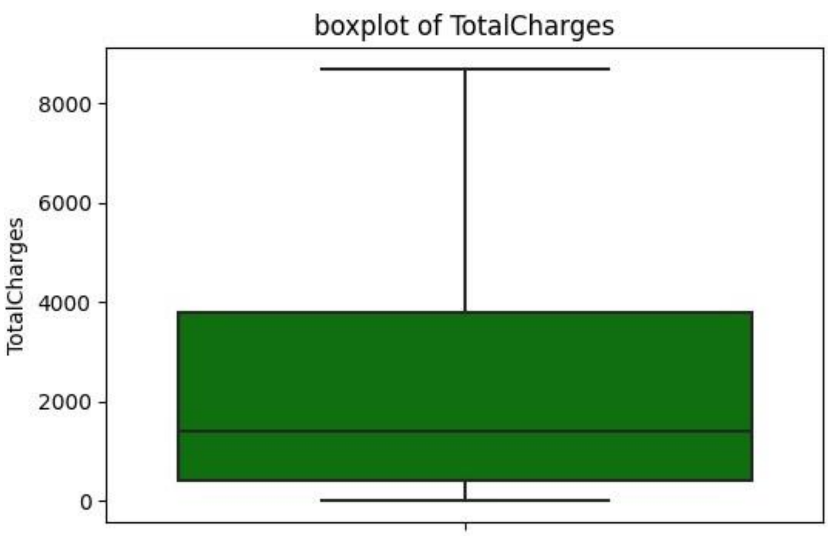
In [23]:
```
plot_boxplot(data,'tenure')
```



In [24]:
```
plot_boxplot(data,'MonthlyCharges')
```



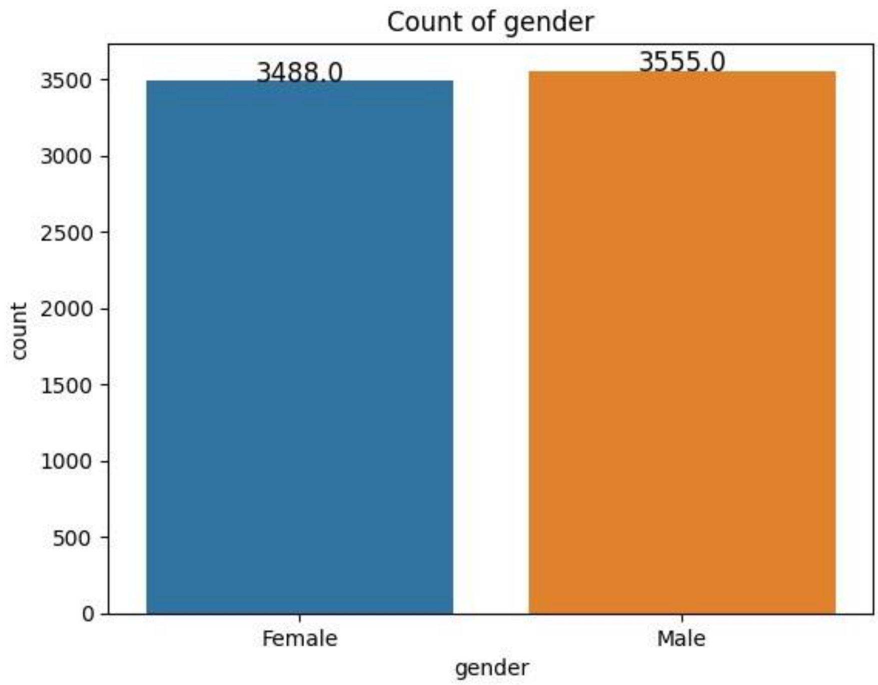
In [25]:
```
plot_boxplot(data,'TotalCharges')
```

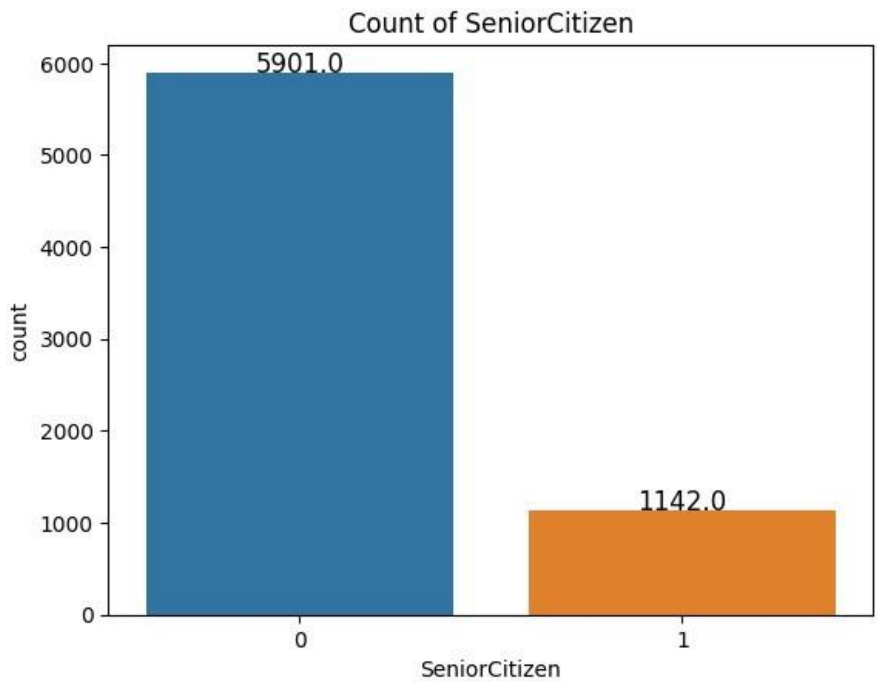


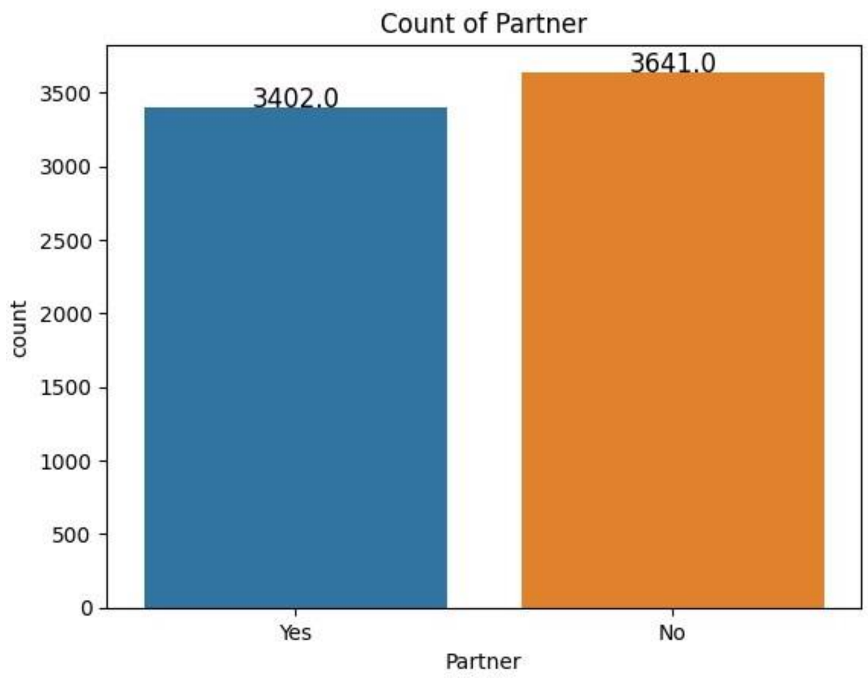
Countplot of all the categorical columns in the dataset

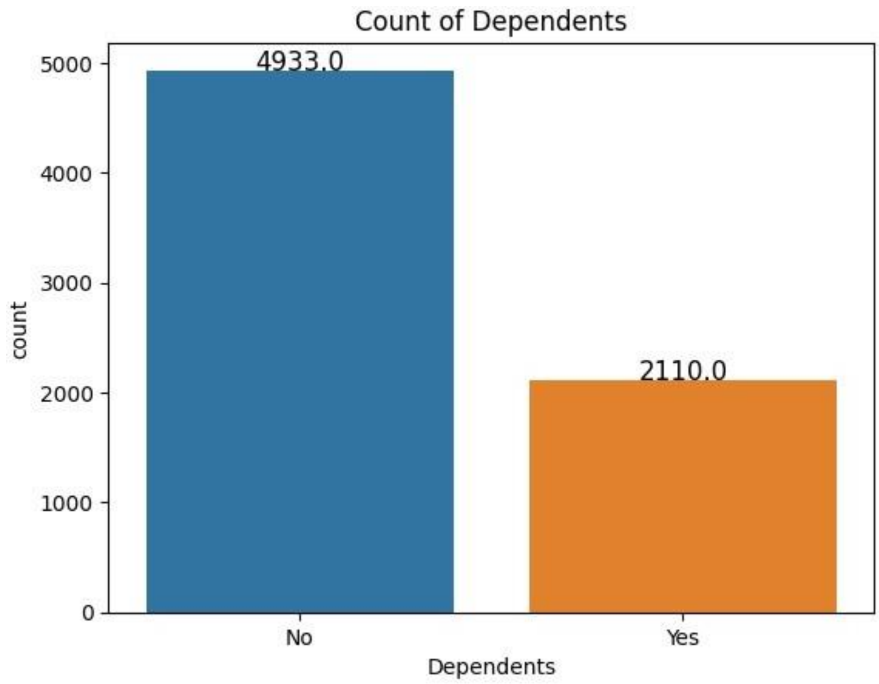
In [26]:
```
numerical_col1=["tenure","MonthlyCharges","TotalCharges"]
for col in data.columns:
    if col not in numerical_col1:
        ax=sns.countplot(data=data,x=data[col])
        for p in ax.patches:
          ax.text(p.get_x() + p.get_width()/2,p.get_height(),str(p.get_height()),fontsize=12,color='black',ha='
```

```
    plt.title(f"Count of {col}")
    plt.show()
```

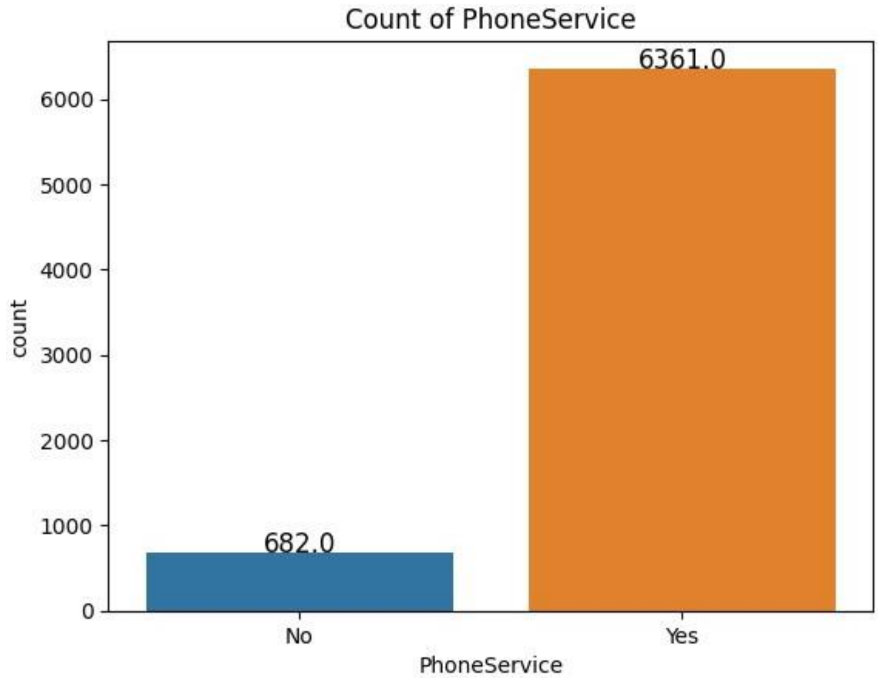




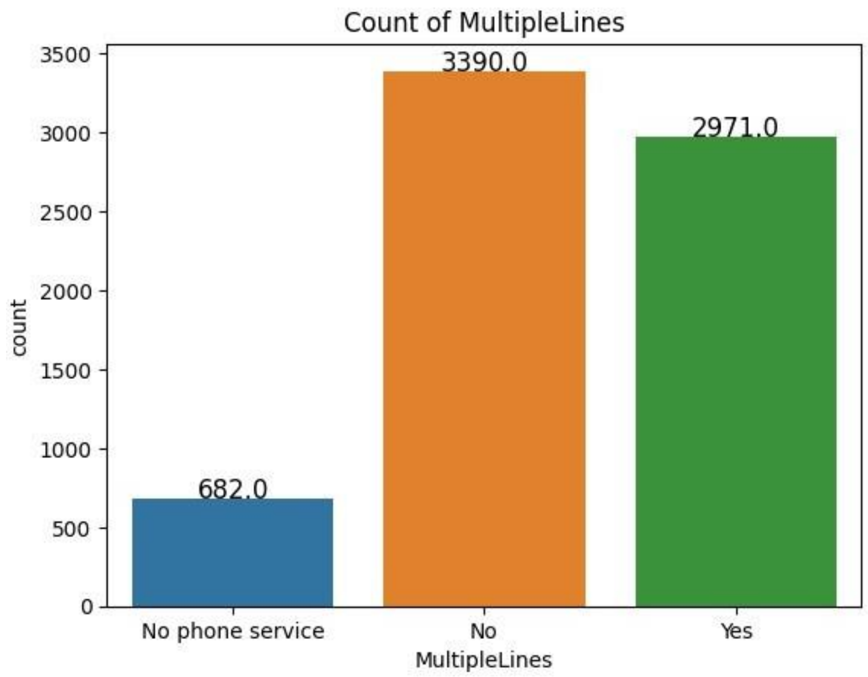




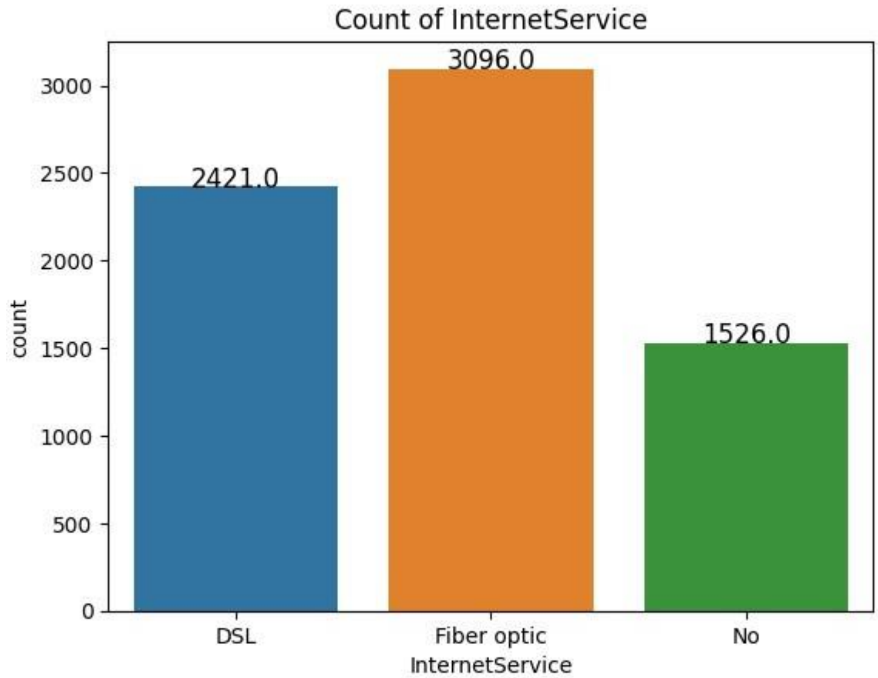
Count of Dependents
4933.0
2110.0
count
0
1000
2000
3000
4000
5000
No
Yes
Dependents



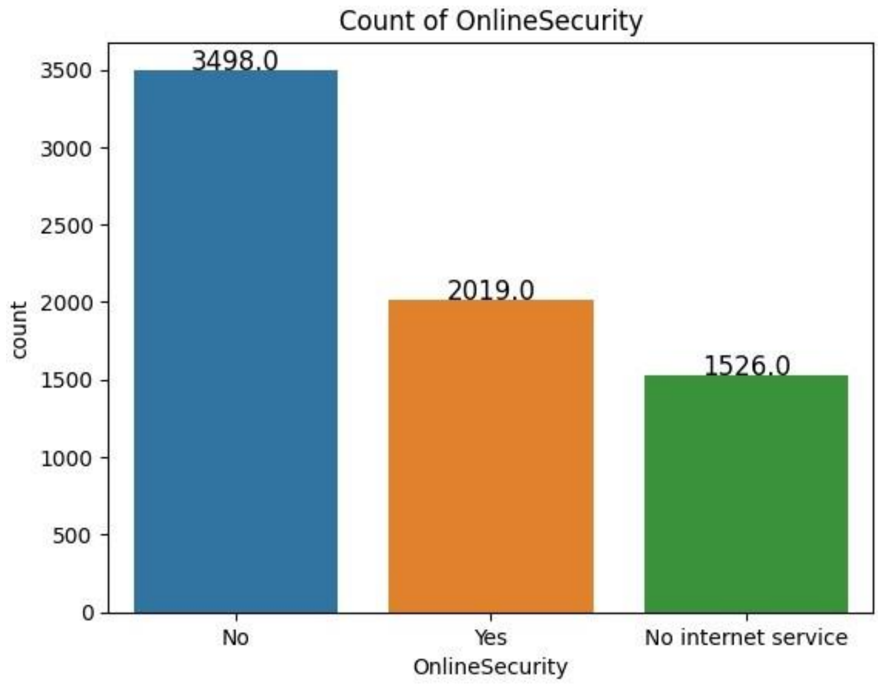
Count of PhoneService
682.0
6361.0
count
0
1000
2000
3000
4000
5000
6000
No
Yes
PhoneService



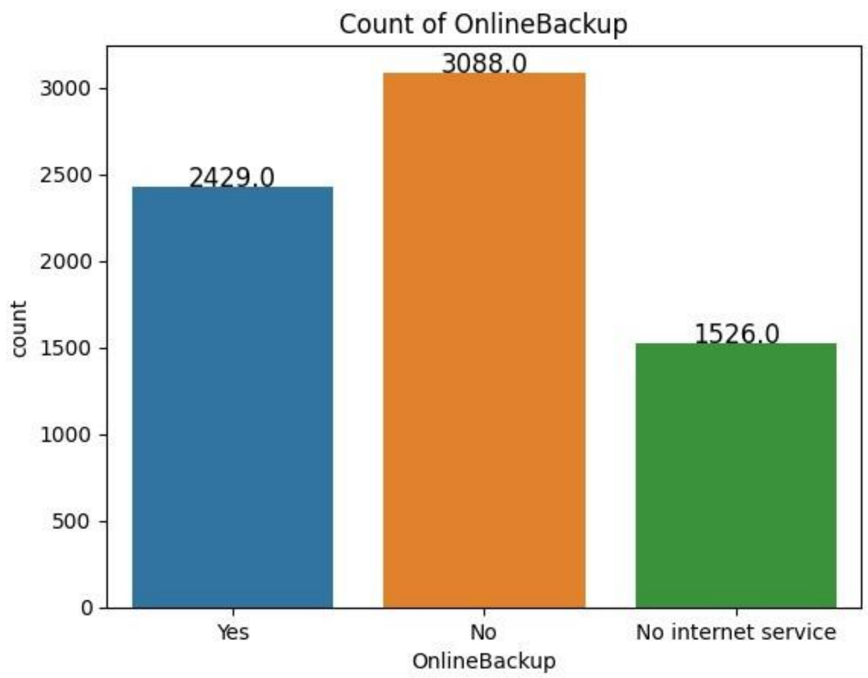
Count of MultipleLines
682.0
3390.0
2971.0
count
0
500
1000
1500
2000
2500
3000
3500
No phone service
No
Yes
MultipleLines



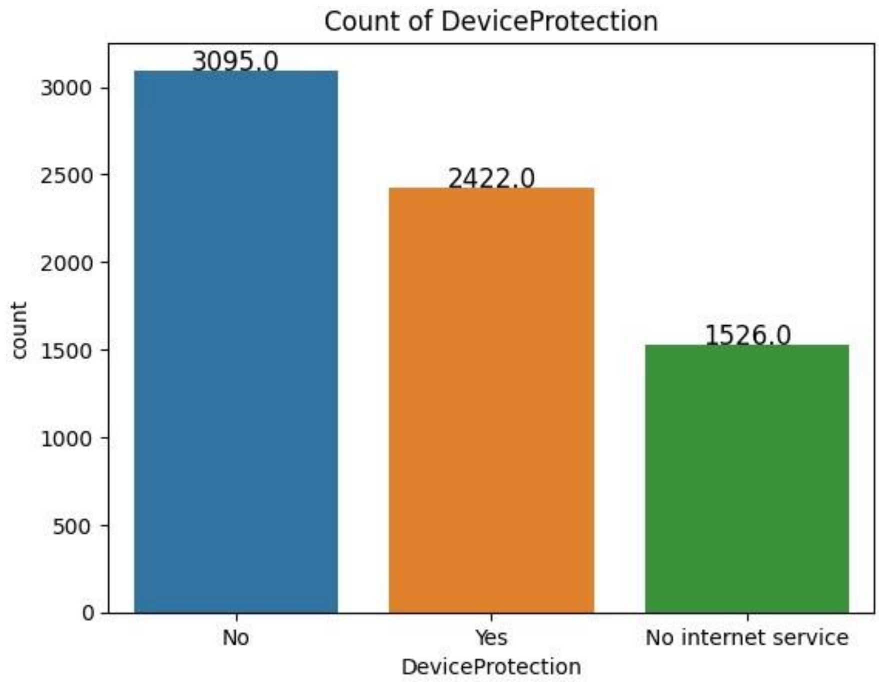
Count of InternetService
3096.0
2421.0
1526.0
count
0
500
1000
1500
2000
2500
3000
DSL
Fiber optic
No
InternetService

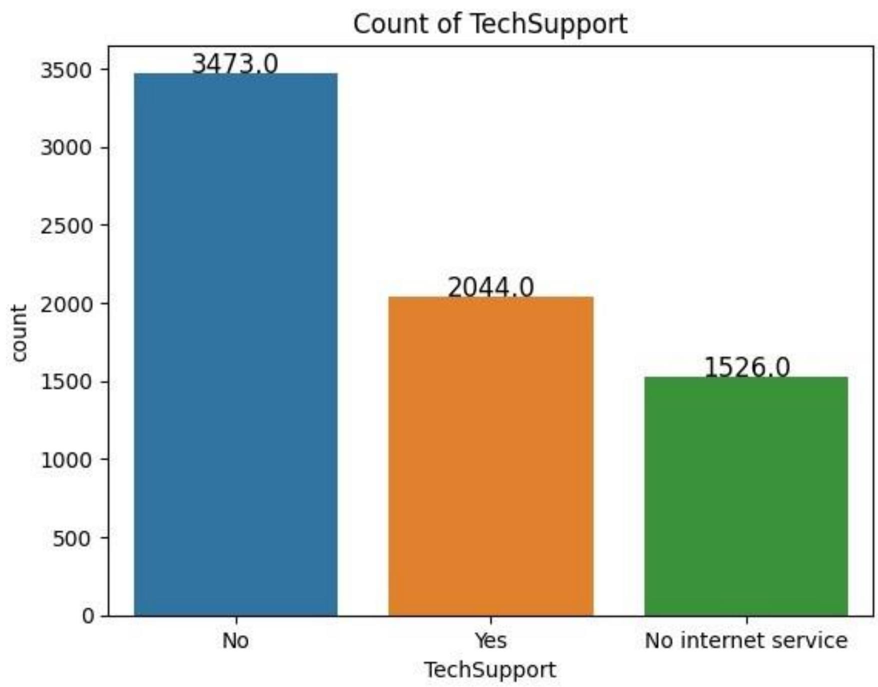


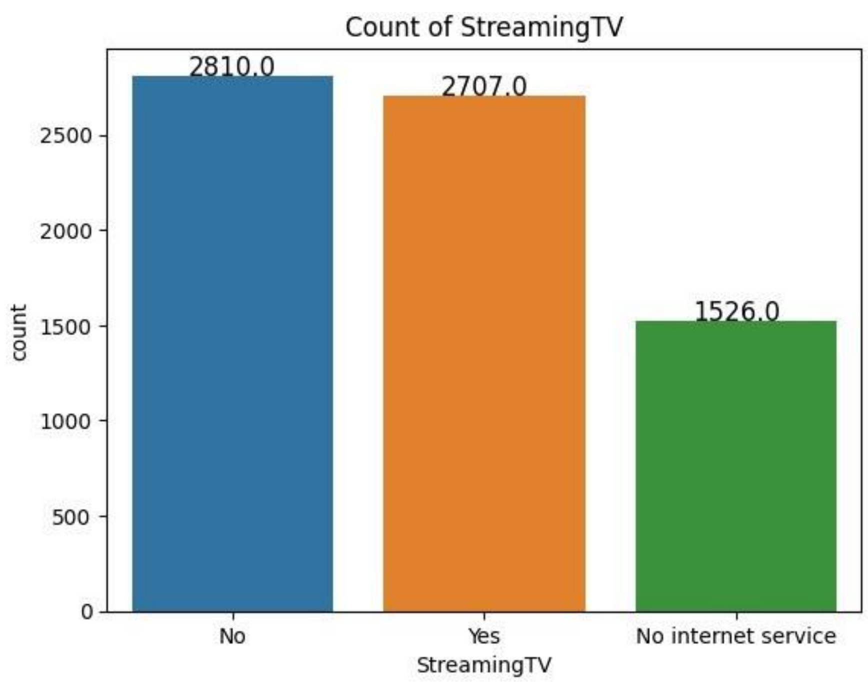
Count of OnlineSecurity
3498.0
2019.0
1526.0
count
0
500
1000
1500
2000
2500
3000
3500
No
Yes
No internet service
OnlineSecurity

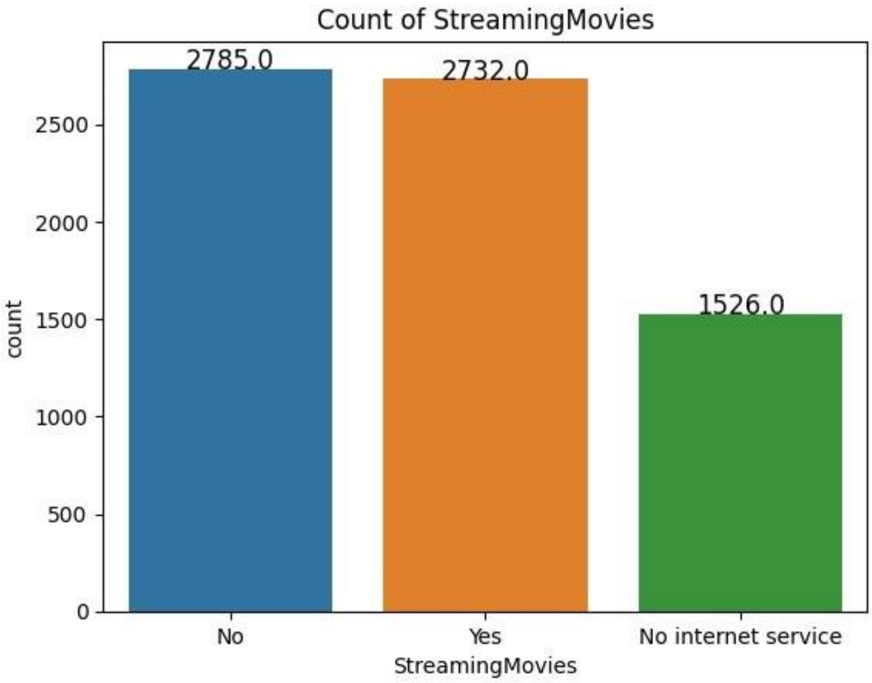


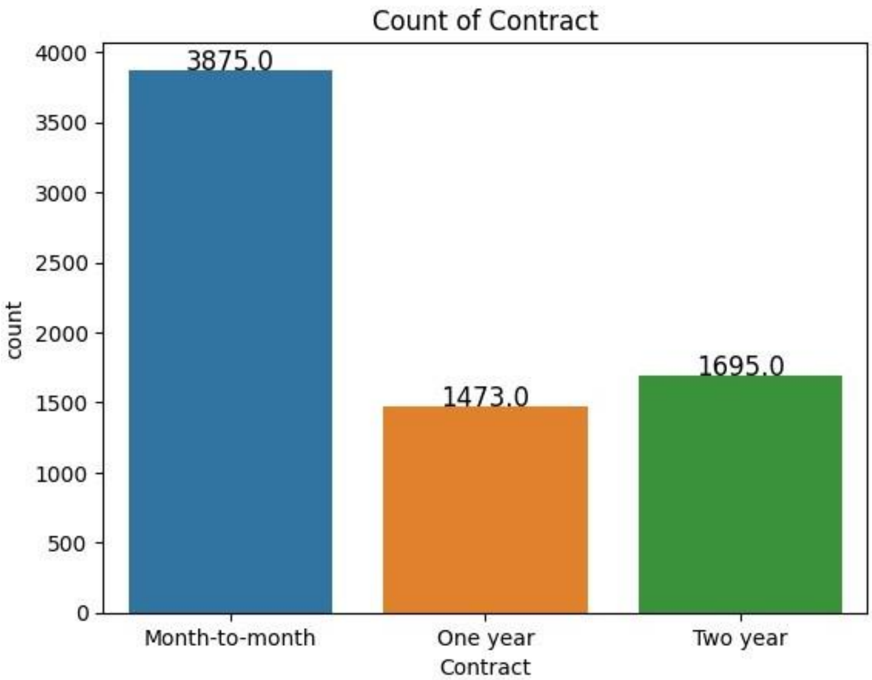
Count of OnlineBackup
3088.0
2429.0
1526.0
count
0
500
1000
1500
2000
2500
3000
Yes
No
No internet service
OnlineBackup

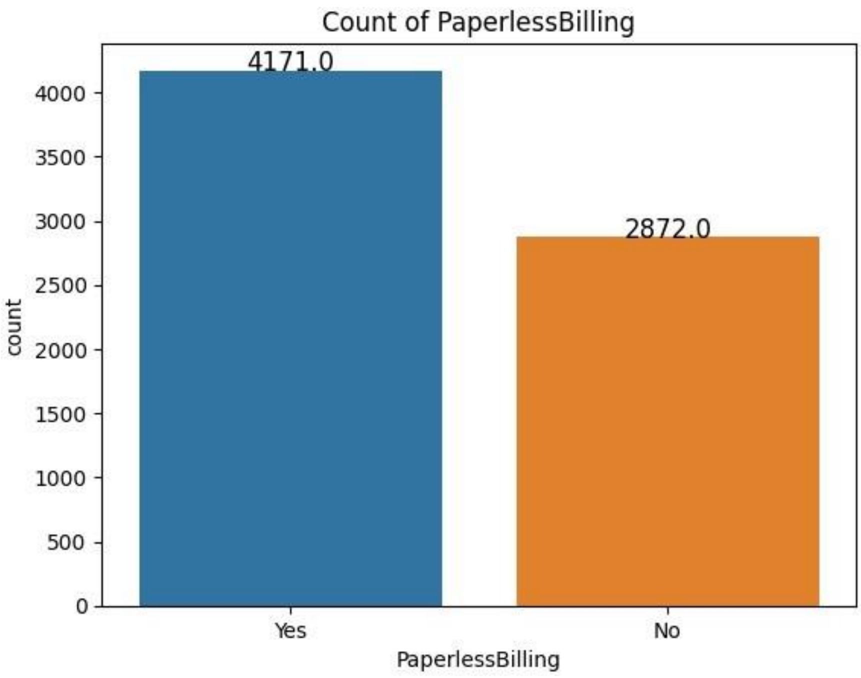

Count of DeviceProtection
3095.0
2422.0
1526.0
count
0
500
1000
1500
2000
2500
3000
No
Yes
No internet service
DeviceProtection

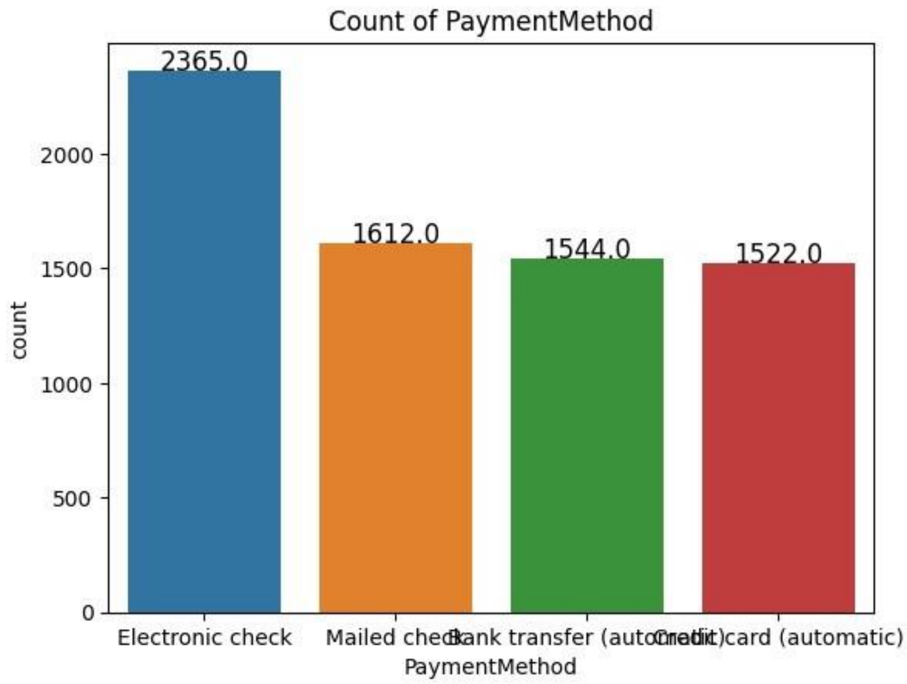

Count of TechSupport
3473.0
2044.0
1526.0
count
0
500
1000
1500
2000
2500
3000
3500
No
Yes
No internet service
TechSupport

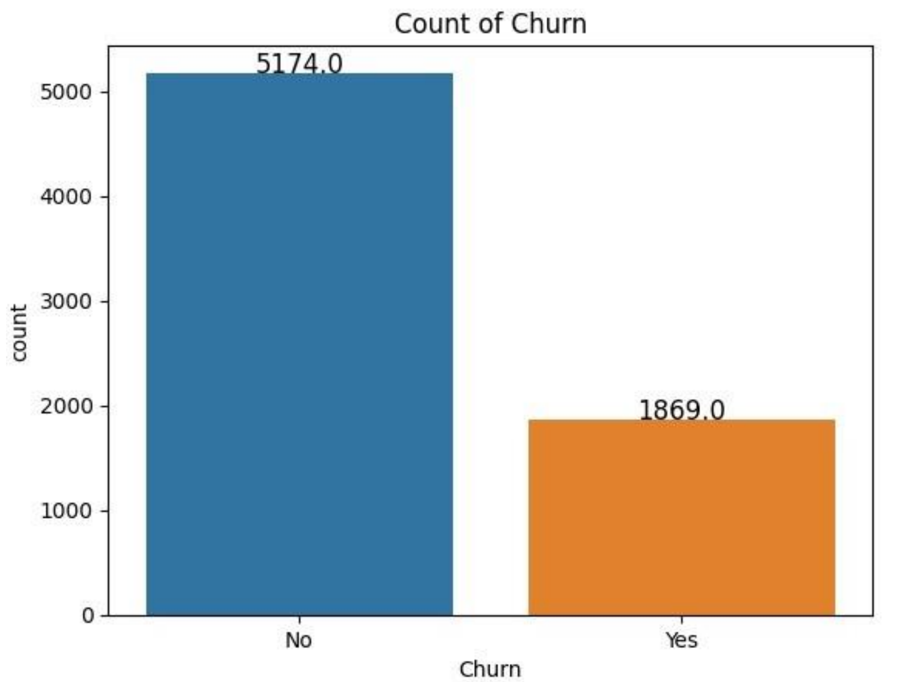

Count of StreamingTV
2810.0
2707.0
1526.0
count
0
500
1000
1500
2000
2500
No
Yes
No internet service
StreamingTV


Count of StreamingMovies
2785.0
2732.0
1526.0
count
0
500
1000
1500
2000
2500
No
Yes
No internet service
StreamingMovies


Count of Contract
3875.0
1473.0
1695.0
count
0
500
1000
1500
2000
2500
3000
3500
4000
Month-to-month
One year
Two year
Contract


Count of PaperlessBilling
4171.0
2872.0
count
0
500
1000
1500
2000
2500
3000
3500
4000
Yes
No
PaperlessBilling





Insights

1. by above countplot we can see the number of customers are more having no parther like-wife ,spouse

2. the number of customers are more having no dependents like-family,children

3. the number of customers are more having phone service with telecom company

4. the number of customers are more compare to others having no multiple lines (home phone,mobile phone)

5. fibre optic type of internet service are most compare to dsl and no

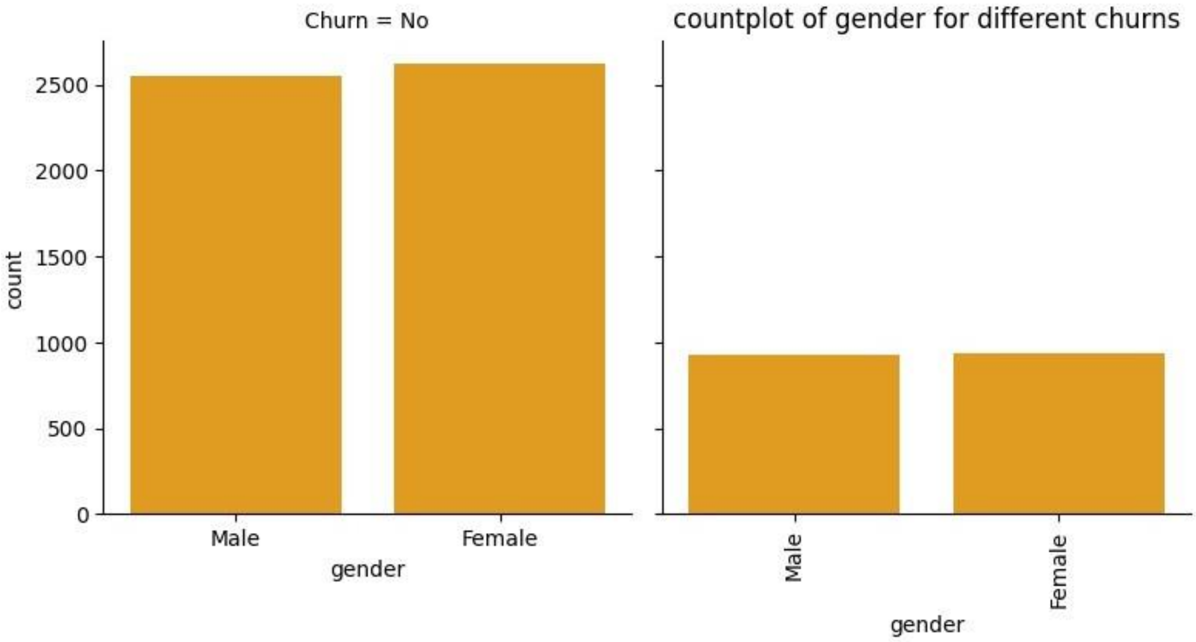
6. there are not much difference between customers who subscribe and unsubscribe streaming movies/streaming tv services provided by telecom company

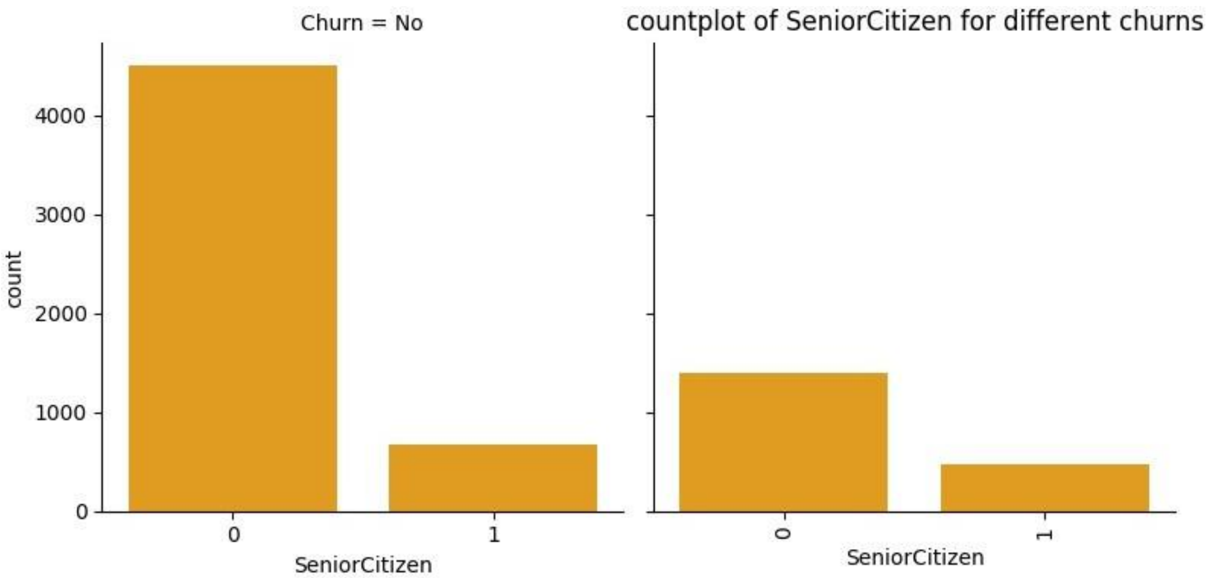
7. mostly customers wants to month_to_month contract with telecom company(like they can cancel their plans from company within a month)

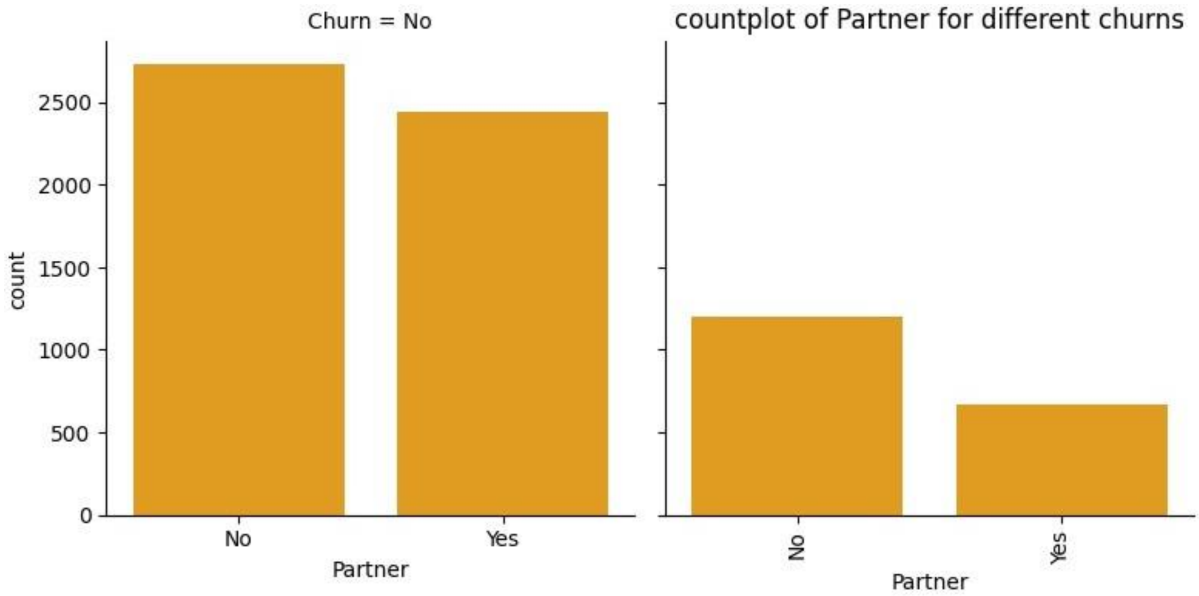
8. payment done by customers using electronic check are most compare to mailed _check,credit_card and bank transfer

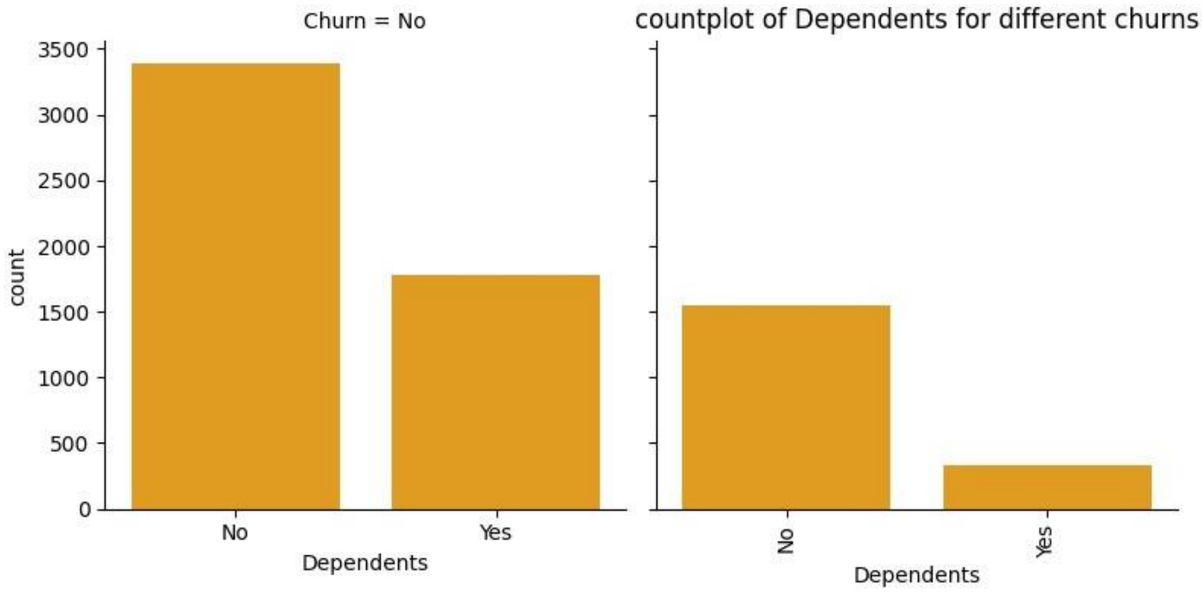
Countplot of each categorical column (having customer's churn and not churn)

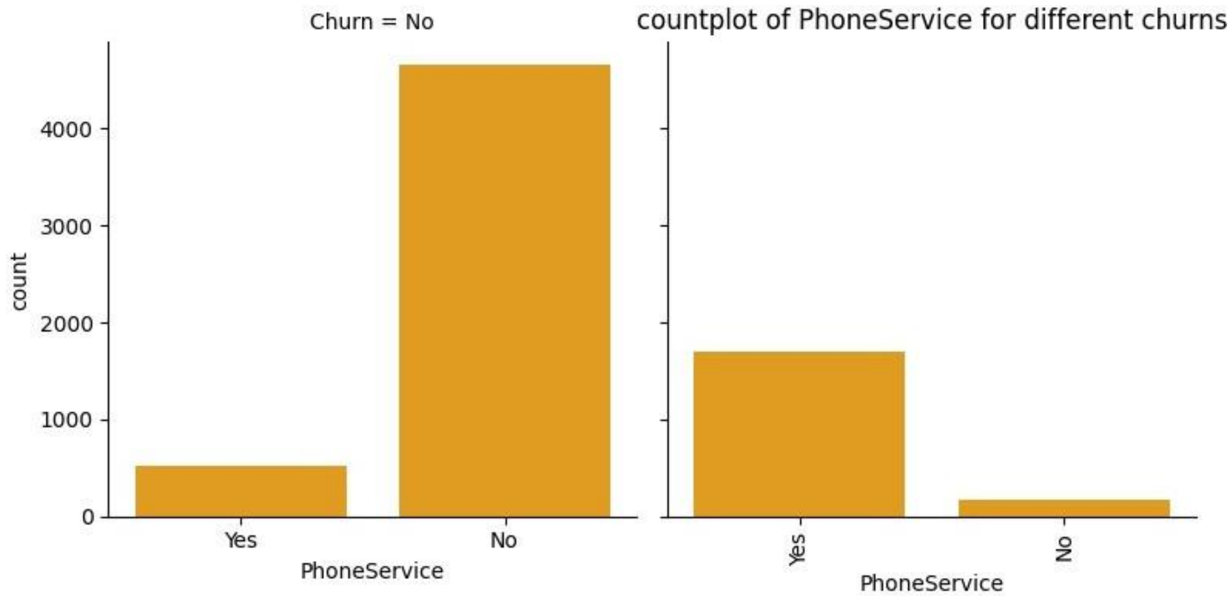
In [33]:
```
for col in data.columns:
    if col not in numerical_col1:
        sns.set_palette(['orange'])

        g=sns.FacetGrid(data,col='Churn',height=4,aspect=1)

        g.map(sns.countplot,col)
        plt.xticks(rotation='vertical')
        plt.title(f"countplot of {col} for different churns")

        plt.show()
```

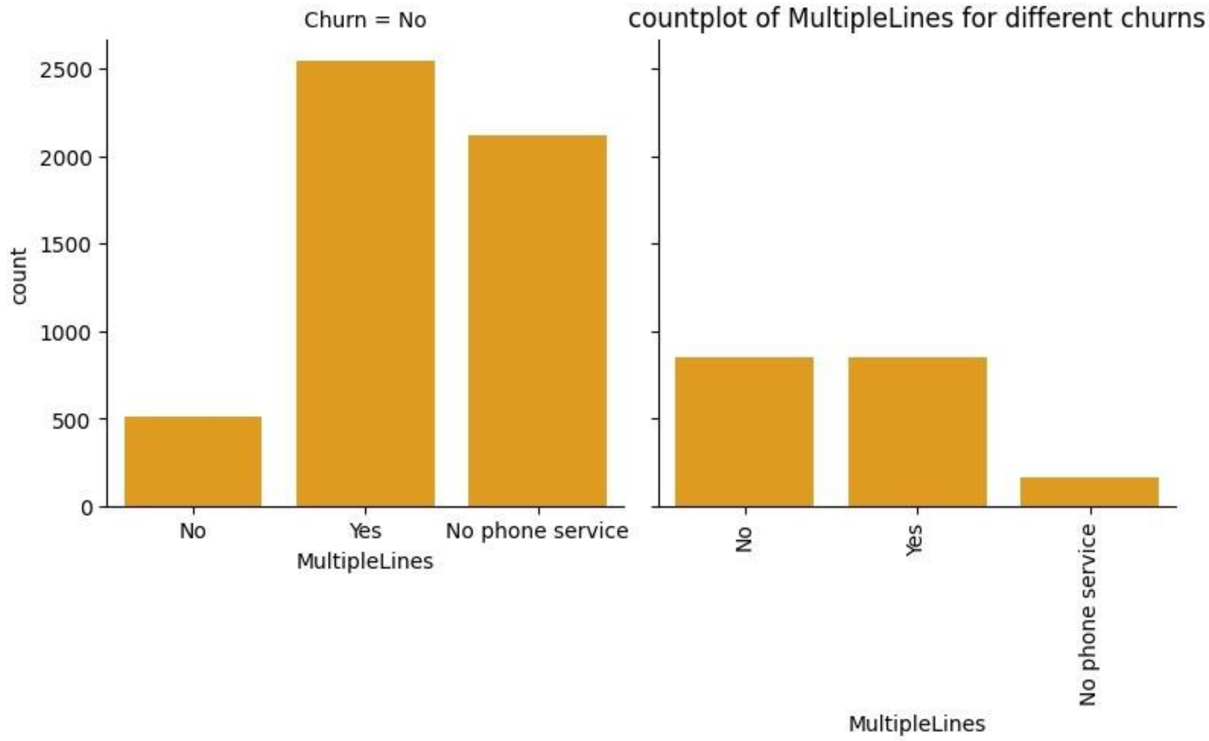


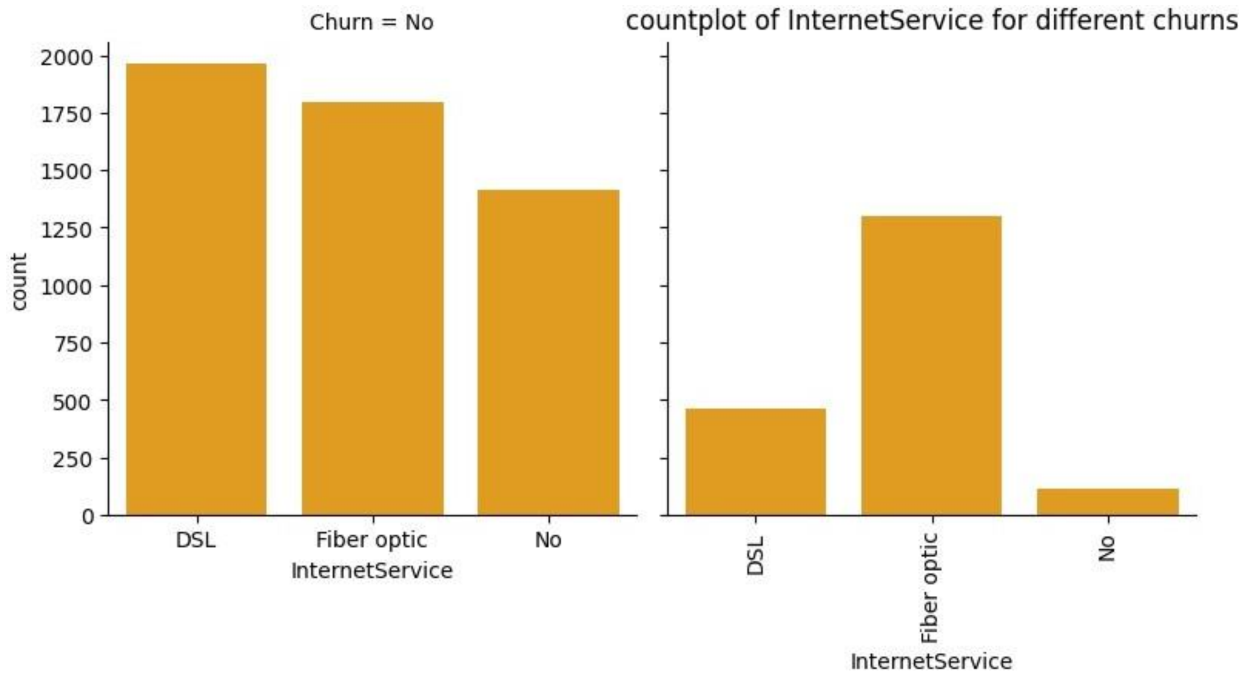



Churn = No
countplot of Partner for different churns
count
2500
2000
1500
1000
500
0
No
Yes
Partner
No
Yes
Partner

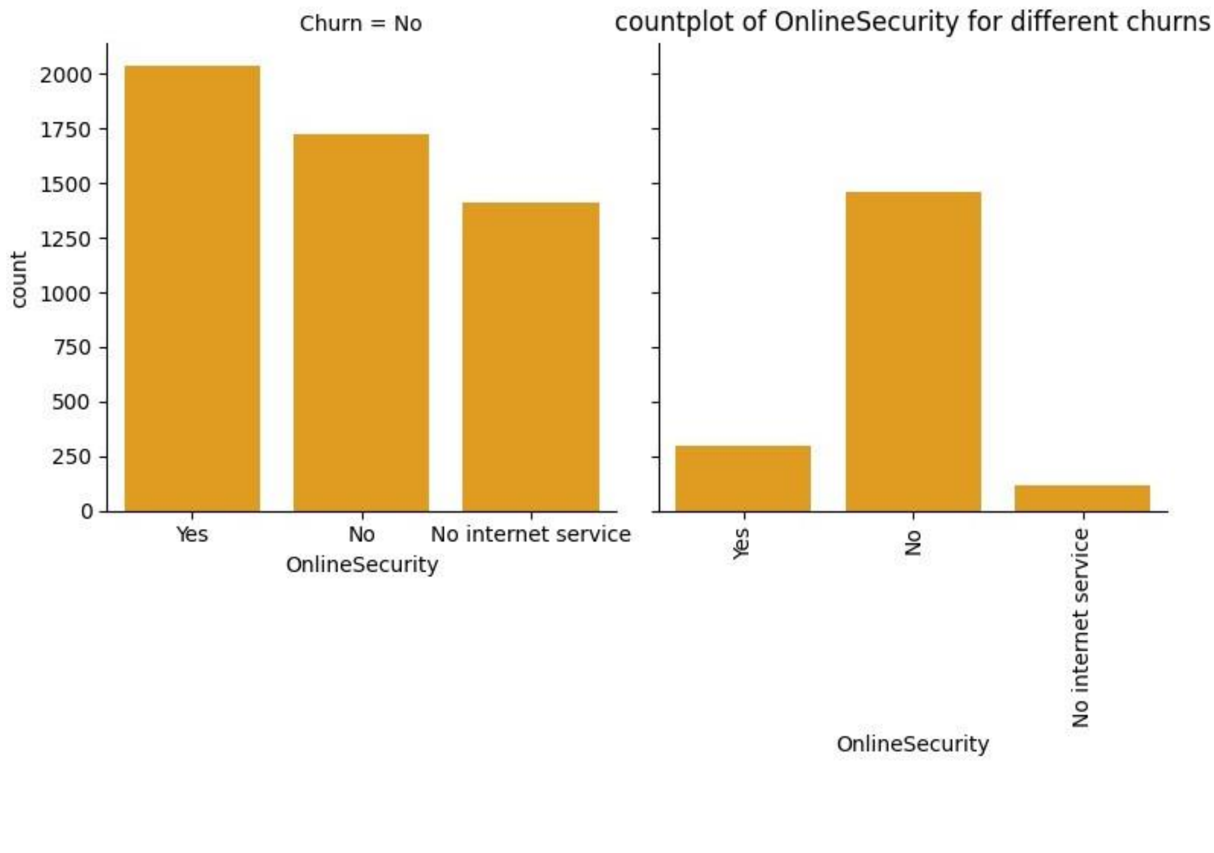

Churn = No
countplot of Dependents for different churns
count
3500
3000
2500
2000
1500
1000
500
0
No
Yes
Dependents
No
Yes
Dependents


Churn = No
countplot of PhoneService for different churns
count
4000
3000
2000
1000
0
Yes
No
PhoneService
Yes
No
PhoneService

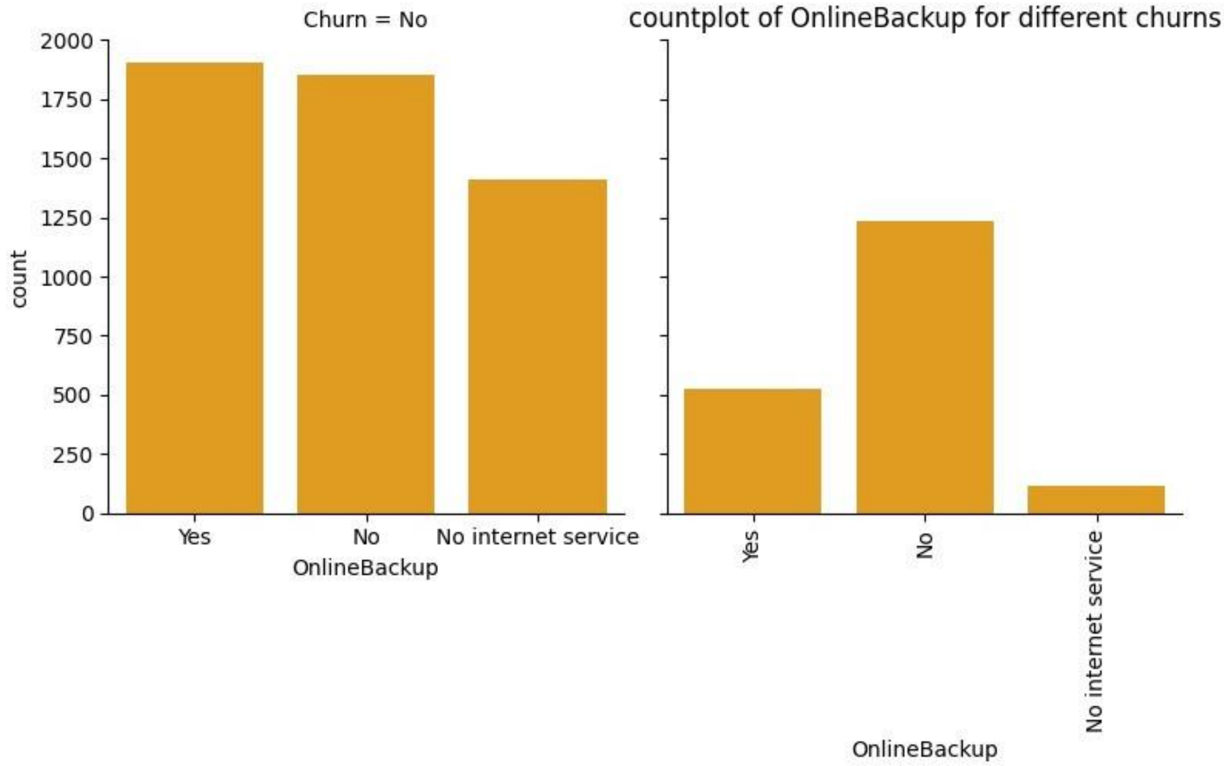

Churn = No
countplot of MultipleLines for different churns
count
2500
2000
1500
1000
500
0
No
Yes
No phone service
MultipleLines
No
Yes
No phone service
MultipleLines

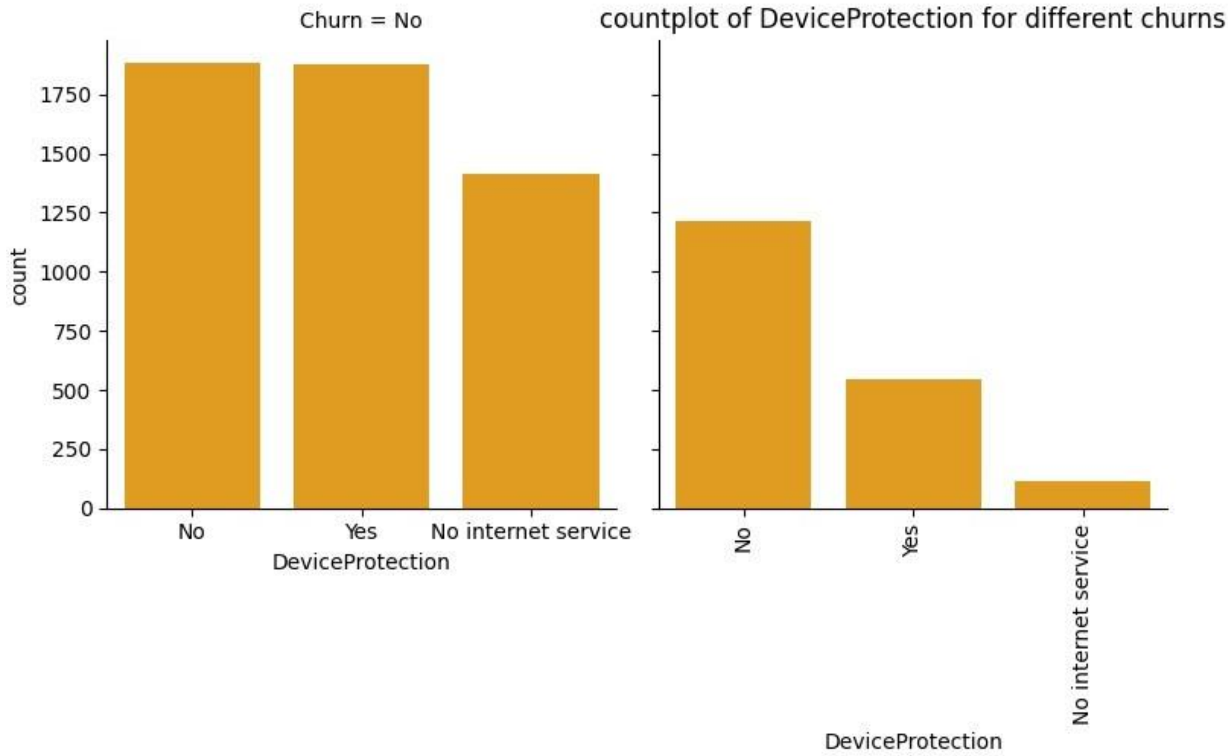


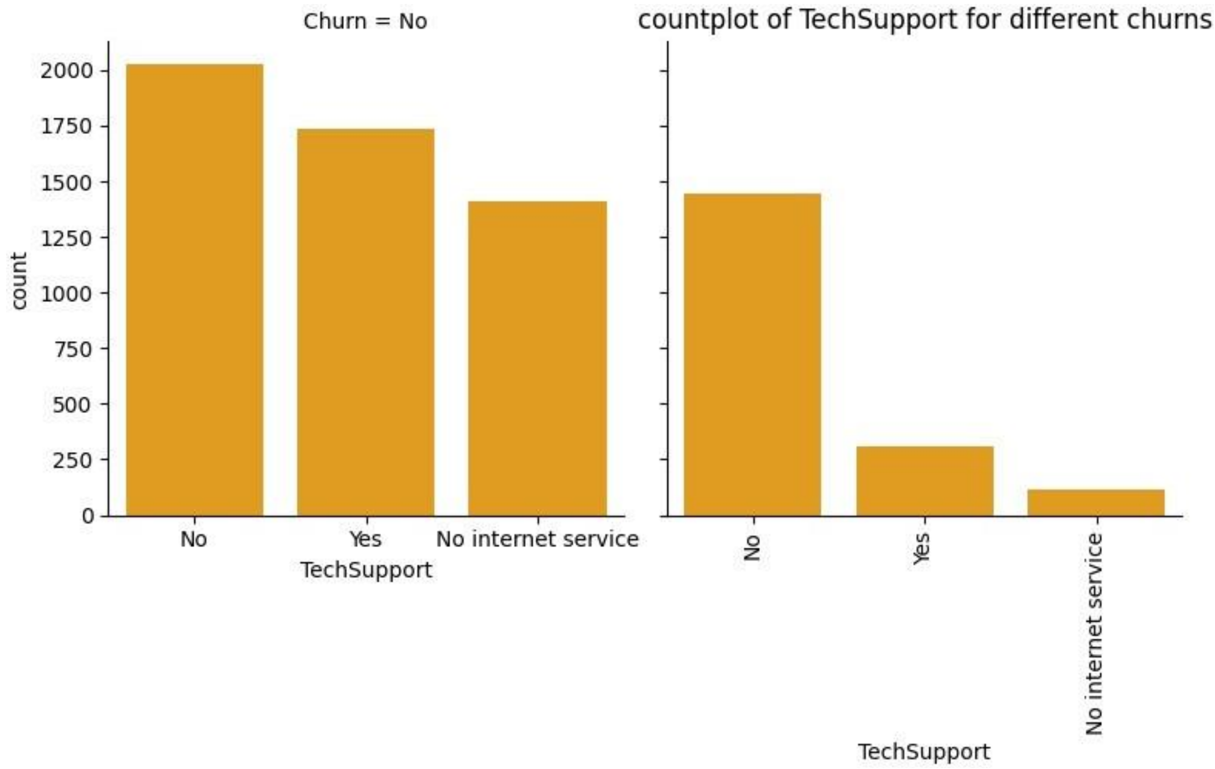
Churn = No
countplot of InternetService for different churns
count
2000
1750
1500
1250
1000
750
500
250
0
DSL
Fiber optic
No
InternetService
DSL
Fiber optic
No
InternetService

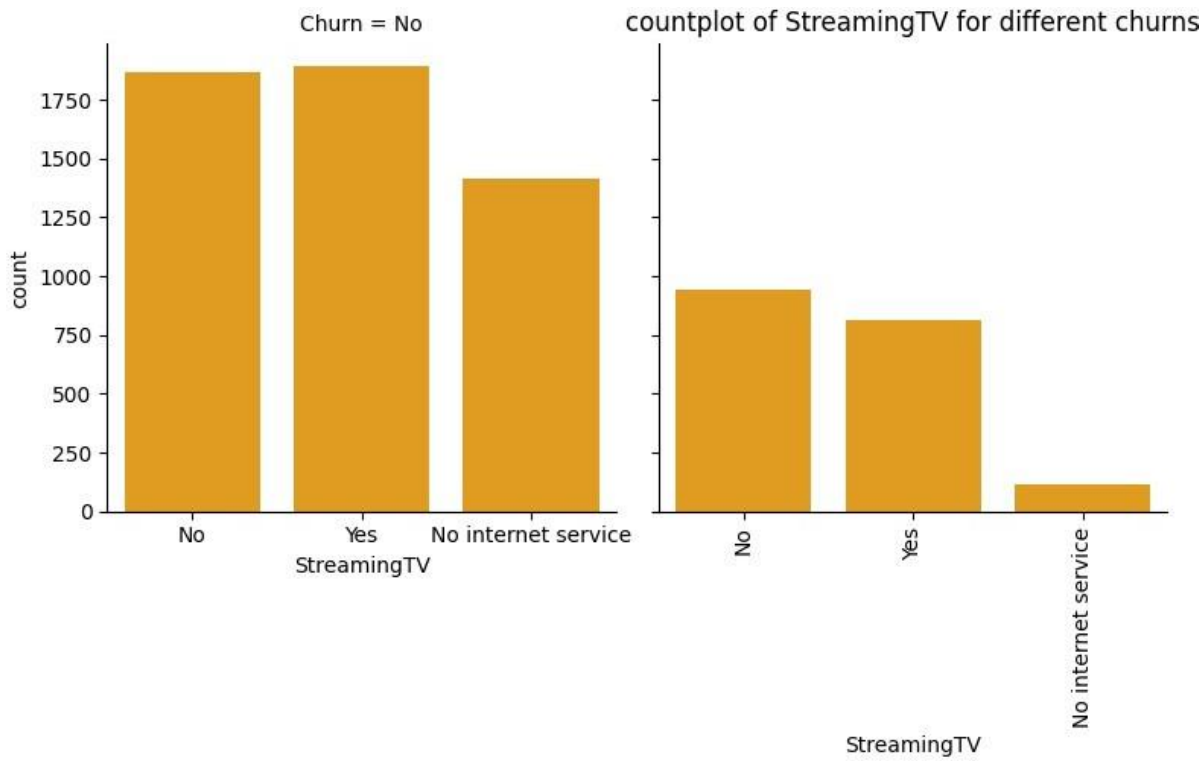


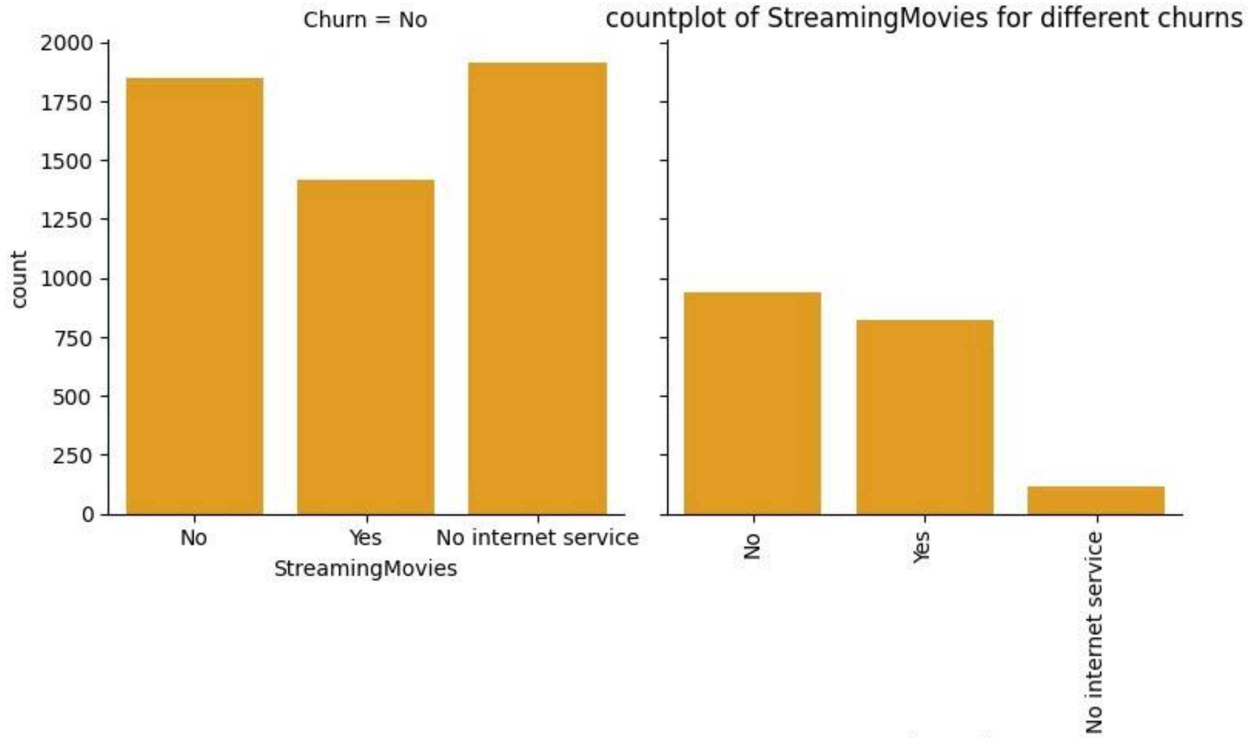
Churn = No
countplot of OnlineSecurity for different churns
count
2000
1750
1500
1250
1000
750
500
250
0
Yes
No
No internet service
OnlineSecurity
Yes
No
No internet service
OnlineSecurity


Churn = No
countplot of OnlineBackup for different churns
count
2000
1750
1500
1250
1000
750
500
250
0
Yes
No
No internet service
OnlineBackup
Yes
No
No internet service
OnlineBackup

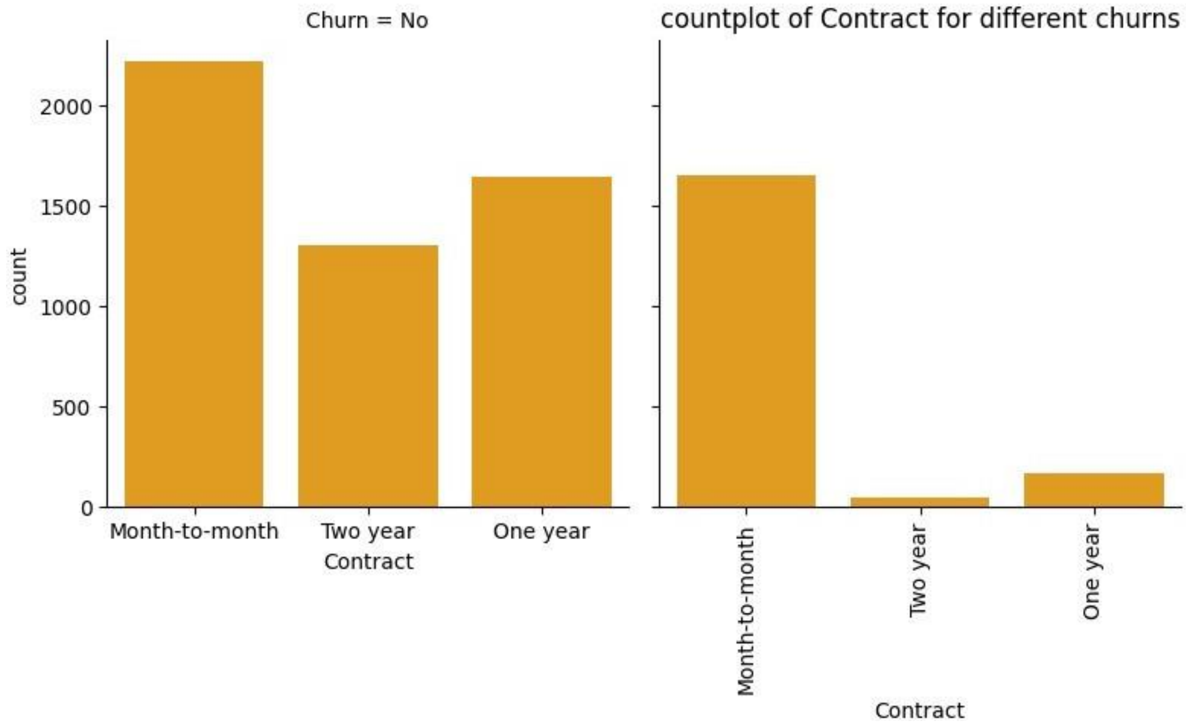

Churn = No
countplot of DeviceProtection for different churns
count
1750
1500
1250
1000
750
500
250
0
No
Yes
No internet service
DeviceProtection
No
Yes
No internet service
DeviceProtection


Churn = No
countplot of TechSupport for different churns
count
2000
1750
1500
1250
1000
750
500
250
0
No
Yes
No internet service
TechSupport
No
Yes
No internet service
TechSupport

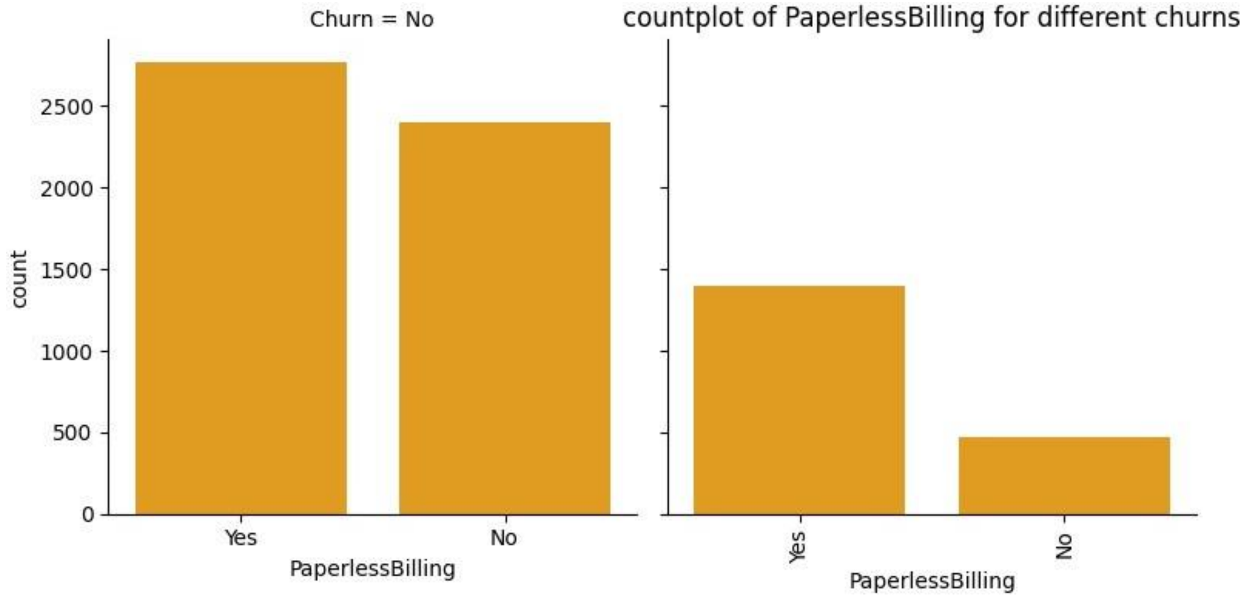

Churn = No
countplot of StreamingTV for different churns
count
1750
1500
1250
1000
750
500
250
0
No
Yes
No internet service
StreamingTV
No
Yes
No internet service
StreamingTV

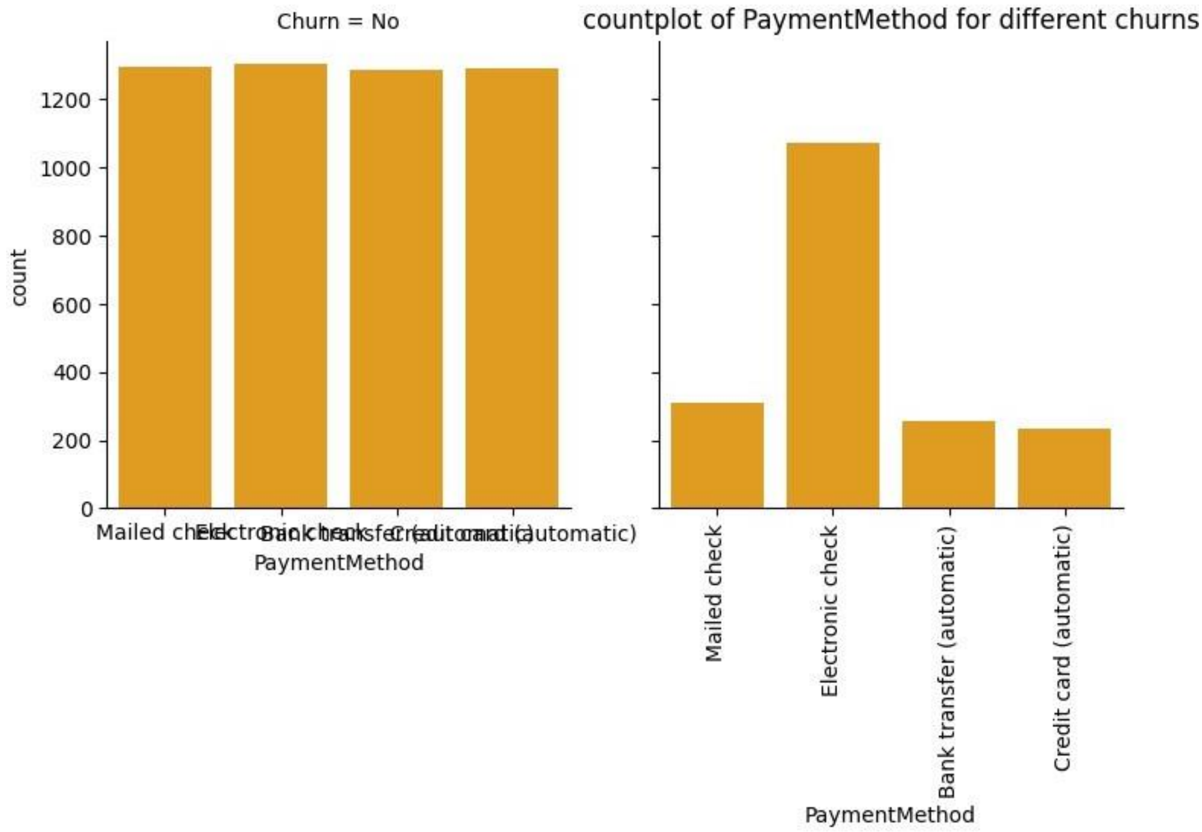


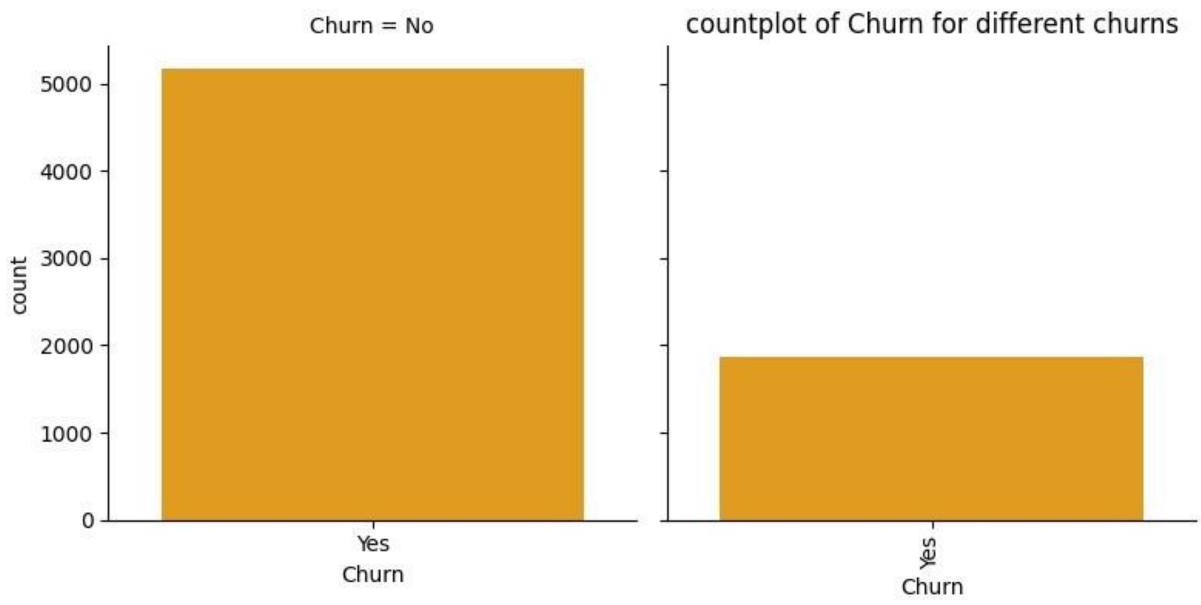
Churn = No
countplot of StreamingMovies for different churns
count
2000
1750
1500
1250
1000
750
500
250
0
No
Yes
No internet service
StreamingMovies
No
Yes
No internet service
StreamingMovies



Churn = No
countplot of Contract for different churns
count
2000
1500
1000
500
0
Month-to-month
Two year
One year
Contract
Month-to-month
Two year
One year
Contract


Churn = No
countplot of PaperlessBilling for different churns
count
0
500
1000
1500
2000
2500
Yes
No
PaperlessBilling
Yes
No
PaperlessBilling


Churn = No
countplot of PaymentMethod for different churns
count
0
200
400
600
800
1000
1200
Mailed check
Electronic check
Bank transfer (automatic)
Credit card (automatic)
PaymentMethod
Mailed check
Electronic check
Bank transfer (automatic)
Credit card (automatic)
PaymentMethod



Insights

1.customers who left the company generaly having phone service

2.the churned customer is in higher in number who doesn't have tech support

3.Most of the customers left the company having Fibre optic type of internet service

4.Mostly customers left the company due to havinhg no online security such as ,antivirus software,firewall protection

5.cutomers who left the telecom company mostly of them having no device protection(if their mobile phone damage,lost then they have to buy phone)

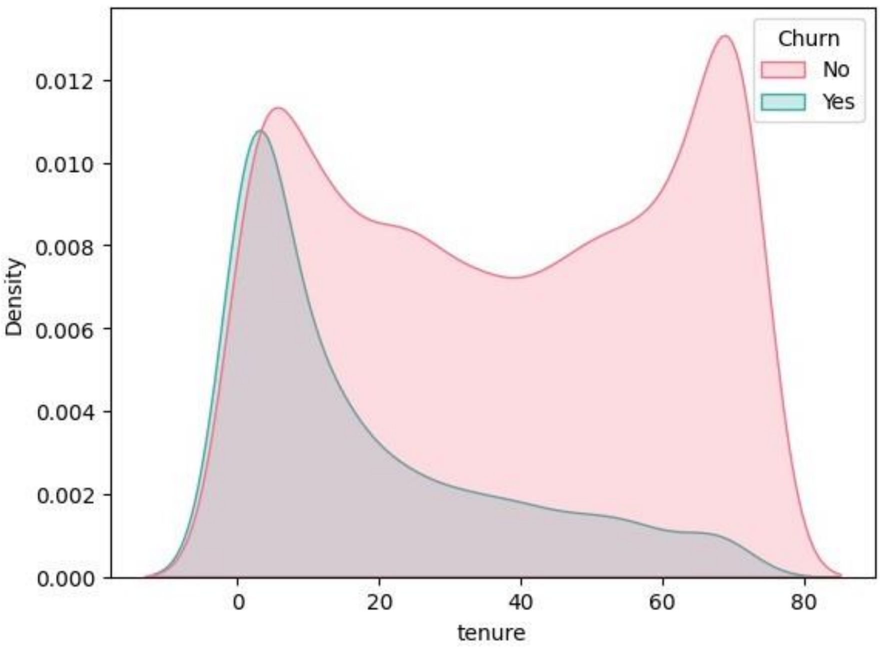
6.the customers who churned from the company mainly done payment via electronic check

7.the customers who left the company generally was of month_to_month contract

Kdeplot of tenure numerical column (having customer's churn and not churn)

In [47]:
```
sns.kdeplot(data=data,x='tenure',hue='Churn',fill=True,color='green')
```

Out[47]: <Axes: xlabel='tenure', ylabel='Density'>



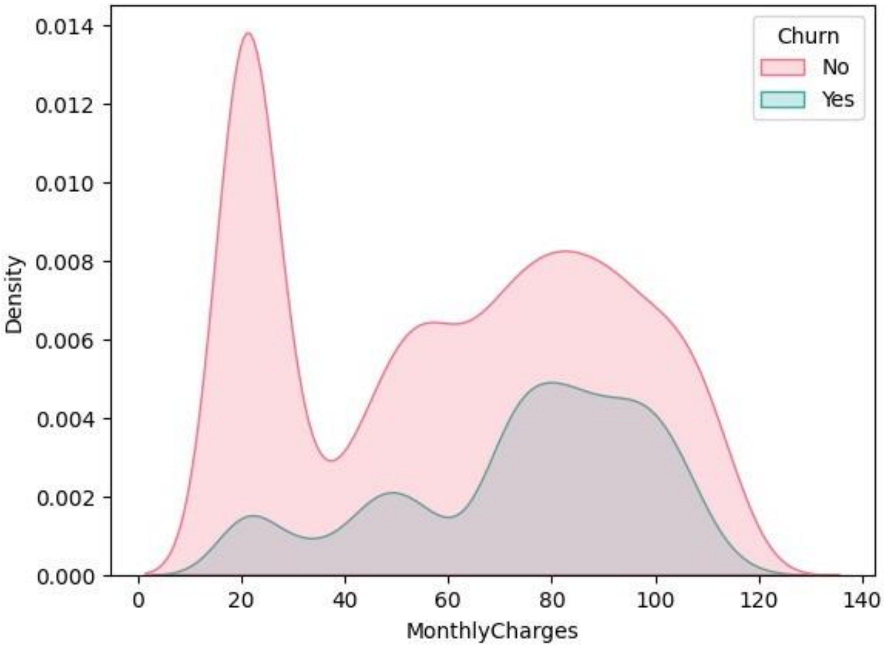
Most of the churned customers having tenure is in between (0-20) unit

In [ ]:

Kdeplot of Monthly Charges numerical column (having customer's churn and not churn)

In [43]:
```
sns.kdeplot(data=data,x='MonthlyCharges',hue='Churn',fill=True)
```

Out[43]: <Axes: xlabel='MonthlyCharges', ylabel='Density'>

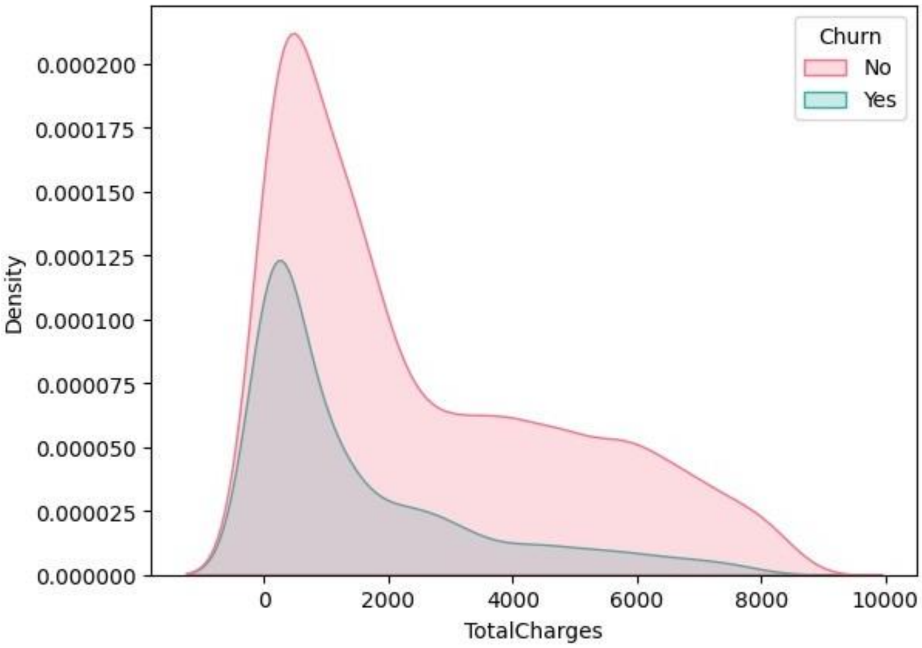


Most of the customers left the company due to high monthly charges

Kdeplot of Monthly Charges numerical column (having customer's churn and not churn)

In [44]:
```
sns.kdeplot(data=data,x='TotalCharges',hue='Churn',fill=True)3
```

Out[44]: <Axes: xlabel='TotalCharges', ylabel='Density'>



Customers who left the company ,most of them having total charges in between (0-2000)

In [35]:
```
data.sample(5)
```

Out[35]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBack
1210	Male	0	Yes	Yes	17	Yes	No	Fiber optic	No	
6038	Female	0	Yes	Yes	70	Yes	Yes	Fiber optic	Yes	`
3394	Male	0	No	No	26	Yes	Yes	DSL	Yes	`
2678	Male	0	No	No	30	Yes	No	No	No internet service	No inter serv
137	Female	0	Yes	Yes	64	Yes	No	No	No internet service	No inter serv

To find all object columns in the dataset

In [36]:
```
object_col=data.select_dtypes(include='object').columns
object_col
```

Out[36]:
```
Index(['gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines',
       'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'Churn'],
      dtype='object')
```

Transforming all categories into numerical

In [37]:
```
from sklearn.preprocessing import LabelEncoder
for col in object_col:
    le=LabelEncoder()
    data[col]=le.fit_transform(data[col])
```

In [38]:
```
data.sample(10)
```

Out[38]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBack
5974	1	0	1	1	10	1	0	2	1	
3444	1	0	1	0	36	0	1	0	2	
911	0	1	0	0	15	1	2	1	0	
6227	1	0	0	0	2	1	0	0	0	
6717	1	1	1	0	57	1	2	1	0	
5577	1	0	0	0	51	1	0	2	1	
308	1	1	1	1	38	1	2	1	0	
3307	0	0	1	1	48	1	2	0	0	
1783	1	0	0	1	1	1	0	1	0	
1603	0	0	0	0	15	1	2	0	2	

In [39]:
```
data['OnlineSecurity'].value_counts()
```

Out[39]:
```
OnlineSecurity
0    3498
2    2019
1    1526
Name: count, dtype: int64
```

In [40]:
```
data['TechSupport'].value_counts()
```

Out[40]:
```
TechSupport
0    3473
2    2044
1    1526
Name: count, dtype: int64
```

In [41]:
```
x=data.drop(columns=['Churn'])
y=data['Churn']
y.sample(5)
```

Out[41]:
```
2697    0
6694    0
3613    0
2037    0
445     1
Name: Churn, dtype: int32
```

training and test data

from sklearn.model_selection import train_test_split x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42) x_train.shape

In [43]:
```
x_test.shape
```

Out[43]:
```
(1409, 19)
```

In [44]:
```
y_train.value_counts()
```

Out[44]:
```
Churn
0    4138
1    1496
Name: count, dtype: int64
```

In [45]:
```
from imblearn.over_sampling import SMOTE
```

In [46]:
```
smote=SMOTE(random_state=42)
x_train_smote,y_train_smote=smote.fit_resample(x_train,y_train)
y_train_smote.value_counts()
```

Out[46]:
```
Churn
0    4138
1    4138
Name: count, dtype: int64
```

Model Training

In [47]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

```
from xgboost import XGBClassifier
```

In [48]:
```
models={"DecisionTree":DecisionTreeClassifier(random_state=42),
        "RandomForest":RandomForestClassifier(random_state=42),
        "xgboost":XGBClassifier(random_state=42)}
```

In [49]:
```
cv_scores=[]
for model_name,model in models.items():
    print(model_name)
    print(model)
    print('-'*50)
```

```
DecisionTree
DecisionTreeClassifier(random_state=42)
--------------------------------------------------
RandomForest
RandomForestClassifier(random_state=42)
--------------------------------------------------
xgboost
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=42, ...)
--------------------------------------------------
```

In [50]:
```
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
```

In [51]:
```
cv_scores={}
for model_name,model in models.items():
    score=cross_val_score(model,x_train_smote,y_train_smote,cv=5,scoring="accuracy")
    cv_scores[model_name]=score
    print(f"{model} croos_validation_accuracy:{np.mean(score):.2f}")
    print('-'*50)
```

```
DecisionTreeClassifier(random_state=42) croos_validation_accuracy:0.78
--------------------------------------------------
RandomForestClassifier(random_state=42) croos_validation_accuracy:0.84
--------------------------------------------------
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=42, ...) croos_validation_accuracy:0.83
--------------------------------------------------
```

In [52]:
```
cv_scores
```

Out[52]:
```
{'DecisionTree': array([0.68115942, 0.71903323, 0.81752266, 0.84350453, 0.84350453]),
 'RandomForest': array([0.72705314, 0.76676737, 0.90453172, 0.89244713, 0.89848943]),
 'xgboost': array([0.71074879, 0.75226586, 0.90271903, 0.89123867, 0.89909366])}
```

In [53]:
```
cv_scores={}
for model_name,model in models.items():
    score=cross_val_score(model,x_train,y_train,cv=5,scoring="accuracy")
    cv_scores[model_name]=score
    print(f"{model} croos_validation_accuracy:{np.mean(score):.2f}")
    print('-'*50)
```

```
DecisionTreeClassifier(random_state=42)  croos_validation_accuracy:0.73
-------------------------------------------------------
RandomForestClassifier(random_state=42)  croos_validation_accuracy:0.79
-------------------------------------------------------
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=42, ...) croos_validation_accuracy:0.78
-------------------------------------------------------
```

In [54]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [55]:
```
y_test.value_counts()
```

Out[55]:
```
Churn
0    1036
1     373
Name: count, dtype: int64
```

In [56]:
```
rfc=RandomForestClassifier(n_estimators=350,max_features=0.6,max_samples=0.6)
rfc.fit(x_train_smote,y_train_smote)
```

Out[56]:
```
RandomForestClassifier
RandomForestClassifier(max_features=0.6, max_samples=0.6, n_estimators=350)
```

Prediction,Confusion Matrix and Classification Report of the data

In [57]:
```
y_pred=rfc.predict(x_test)
print(accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
0.7735982966643009
[[861 175]
 [144 229]]
              precision    recall  f1-score   support

           0       0.86      0.83      0.84      1036
           1       0.57      0.61      0.59       373

    accuracy                           0.77      1409
   macro avg       0.71      0.72      0.72      1409
weighted avg       0.78      0.77      0.78      1409
```

In [58]:
```
from sklearn.ensemble import GradientBoostingClassifier
gb=GradientBoostingClassifier(n_estimators=20,learning_rate=0.5)
gb.fit(x_train_smote,y_train_smote)
```

Out[58]:
```
GradientBoostingClassifier
GradientBoostingClassifier(learning_rate=0.5, n_estimators=20)
```

In [59]:
```
y_pred1=gb.predict(x_test) accuracy_score(y_pred,y_test)
```

Out[59]: 0.7735982966643009

Conclusion of the Churn Rate Analysis Project:

The exploratory data analysis (EDA) of telecom customer churn provided valuable insights into customer behavior and factors influencing churn. The key takeaways from the project include:

1. **Data Preprocessing and Cleaning:**
    - The dataset contained 7043 entries with 20 columns.
    - Null values in the TotalCharges column were identified and replaced with 0.
    - Data types were corrected to ensure proper analysis, including converting categorical variables into numerical form using label encoding.
2. **Key Insights from EDA:**
    - Most customers preferred **month-to-month contracts**, indicating a desire for flexibility.
    - Customers who **churned** were more likely to have **Fiber Optic** internet service, suggesting potential issues with service satisfaction or pricing.
    - **Electronic checks** were the most common payment method used by customers who left the company, hinting at a possible correlation between payment method and churn.
    - Customers without **tech support** and **online security** services were more likely to leave.
    - **Tenure and monthly charges** played a significant role in churn, with many customers leaving within the first 0-20 months.
3. **Model Training and Evaluation:**
    - Several machine learning models were trained, including:
        - **Decision Tree** (cross-validation accuracy: ~73%)
        - **Random Forest** (cross-validation accuracy: ~79%)
        - **XGBoost** (cross-validation accuracy: ~78%)
    - SMOTE (Synthetic Minority Over-sampling Technique) was used to balance the dataset, improving model performance.
    - The Random Forest model achieved an accuracy of **77.36%**, with a precision of **86%** for non-churn and **57%** for churn cases.
    - The Gradient Boosting model provided comparable results, showing that boosting techniques can further enhance predictive performance.
4. **Key Factors Influencing Churn:**
    - Customers with **low tenure** and **high monthly charges** were more likely to churn.
    - Lack of additional services such as **online security, tech support, and device protection** contributed to higher churn rates.
    - Customers with **dependents and partners** were less likely to leave, indicating customer stability.
5. **Recommendations:**
    - Offer incentives or discounts to customers with short tenure to increase retention.
    - Improve service offerings for Fiber Optic users to address potential concerns.
    - Educate customers about the benefits of additional services like online security and device protection.
    - Encourage alternative payment methods to reduce churn among electronic check users.