



Welcome to Colab!

```
In [15]: from google.colab import files  
uploaded = files.upload()
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Customer Churn.csv to Customer Churn (1).csv

```
In [16]: import pandas as pd  
df = pd.read_csv("Customer Churn.csv")  
df
```

Out[16]:

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

7043 rows × 21 columns

```
In [19]: df["TotalCharges"] = df["TotalCharges"].replace(" ", "0")  
df["TotalCharges"] = df["TotalCharges"].astype("float")
```

```
In [17]: df.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 [20]: df.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   float64 
 20  Churn          7043 non-null   object  
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

```
In [22]: df.isnull().sum().sum()
```

```
Out[22]: np.int64(0)
```

```
In [23]: df.describe()
```

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692	2279.734304
std	0.368612	24.559481	30.090047	2266.794470
min	0.000000	0.000000	18.250000	0.000000
25%	0.000000	9.000000	35.500000	398.550000
50%	0.000000	29.000000	70.350000	1394.550000
75%	0.000000	55.000000	89.850000	3786.600000
max	1.000000	72.000000	118.750000	8684.800000

```
In [24]: df.duplicated().sum()
```

```
Out[24]: np.int64(0)
```

```
In [25]: df["customerID"].duplicated().sum()
```

```
Out[25]: np.int64(0)
```

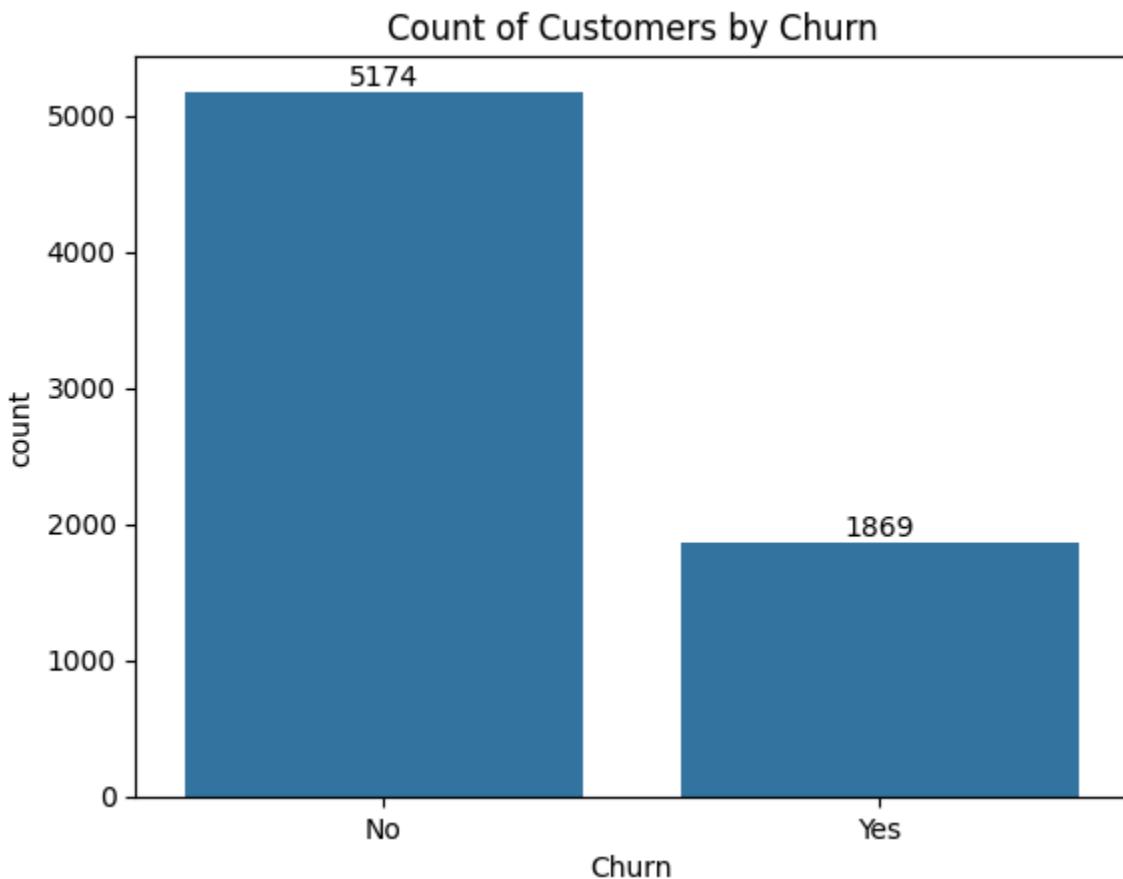
```
In [26]: def conv(value):
    if value == 1:
        return "yes"
    else:
        return "no"
```

```
df['SeniorCitizen'] = df["SeniorCitizen"].apply(conv)
```

Here we have converted 0 and 1 values of senior citizen to yes/no to make it easier to understand.

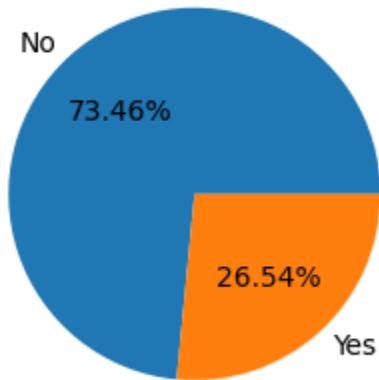
```
In [31]: ax = sns.countplot(x = 'Churn', data = df)
```

```
ax.bar_label(ax.containers[0])
plt.title("Count of Customers by Churn")
plt.show()
```



```
In [32]: plt.figure(figsize = (3,4))
gb = df.groupby("Churn").agg({'Churn':'count"})
plt.pie(gb['Churn'], labels = gb.index, autopct = "%1.2f%%")
plt.title("Percentage of Churned Customeres", fontsize = 10)
plt.show()
```

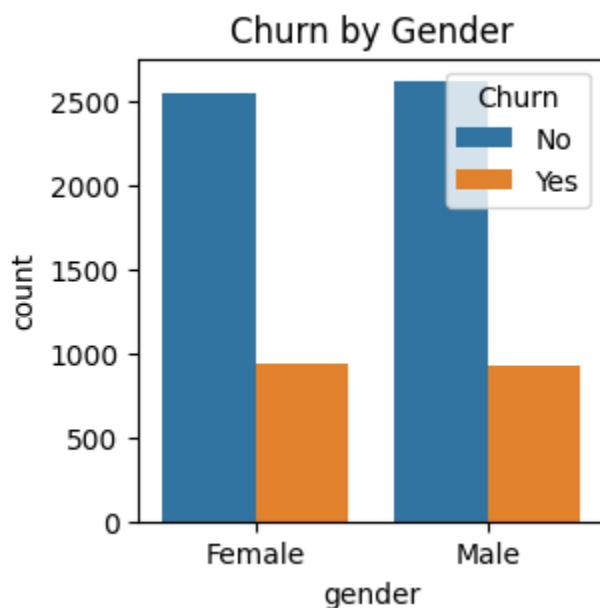
Percentage of Churned Customeres



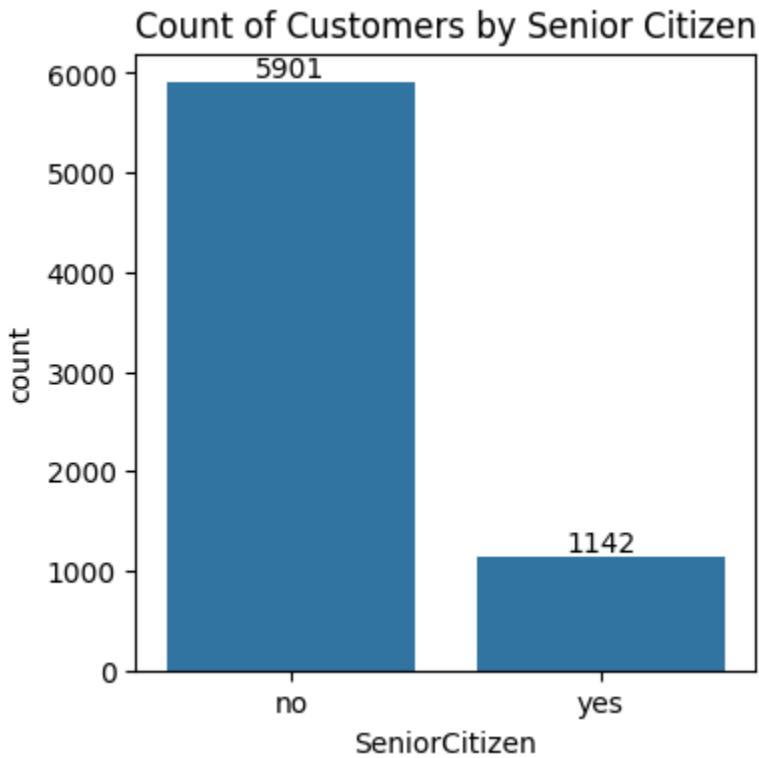
From the given pie chart we can conclude that 26.54% of our customers have churned out.

let's explore the reason behind it

```
In [33]: plt.figure(figsize = (3,3))
sns.countplot(x = "gender", data = df, hue = "Churn")
plt.title("Churn by Gender")
plt.show()
```



```
In [34]: plt.figure(figsize = (4,4))
ax = sns.countplot(x = "SeniorCitizen", data = df)
ax.bar_label(ax.containers[0])
plt.title("Count of Customers by Senior Citizen")
plt.show()
```



```
In [35]: total_counts = df.groupby('SeniorCitizen')['Churn'].value_counts(normalize=True)

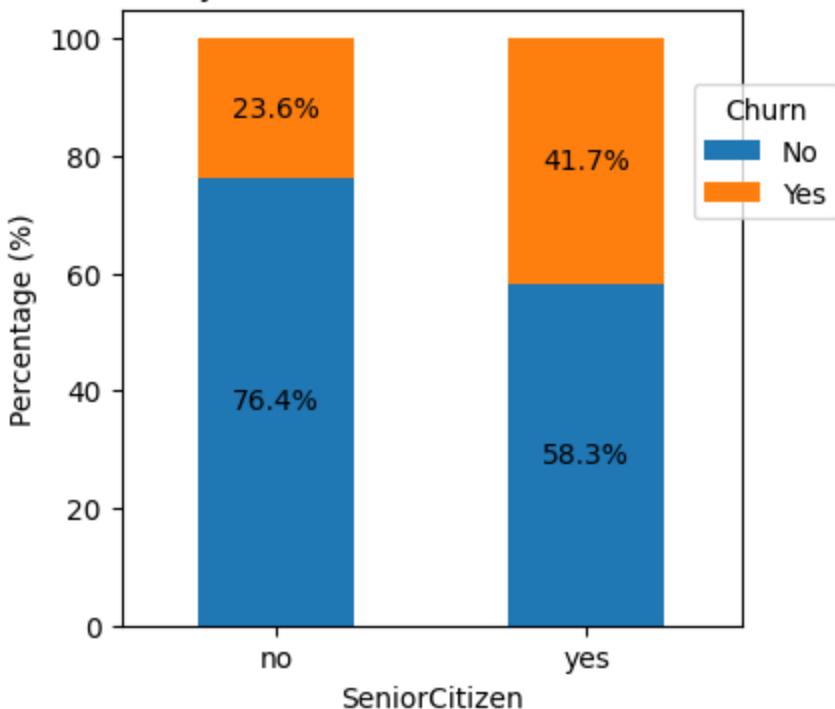
# Plot
fig, ax = plt.subplots(figsize=(4, 4)) # Adjust figsize for better visualization

# Plot the bars
total_counts.plot(kind='bar', stacked=True, ax=ax, color=['#1f77b4', '#ff7f0e'])

# Add percentage labels on the bars
for p in ax.patches:
    width, height = p.get_width(), p.get_height()
    x, y = p.get_xy()
    ax.text(x + width / 2, y + height / 2, f'{height:.1f}%', ha='center', va='bottom')

plt.title('Churn by Senior Citizen (Stacked Bar Chart)')
plt.xlabel('SeniorCitizen')
plt.ylabel('Percentage (%)')
plt.xticks(rotation=0)
plt.legend(title='Churn', bbox_to_anchor = (0.9,0.9)) # Customize legend location
plt.show()
```

Churn by Senior Citizen (Stacked Bar Chart)

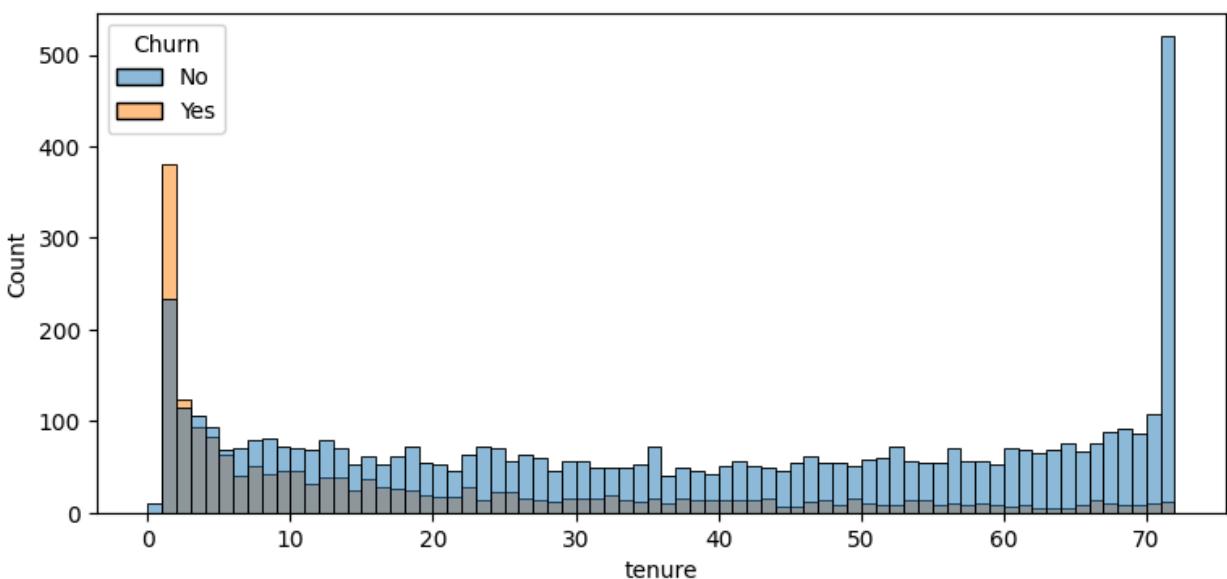


In []:

comparative a greater pecentage of people in senior citizen category have churned

In [36]:

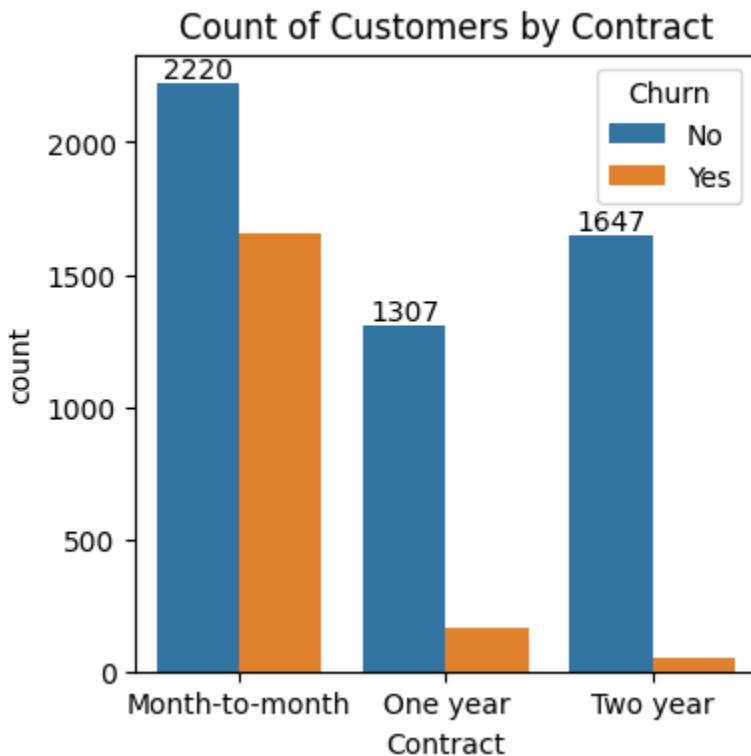
```
plt.figure(figsize = (9,4))
sns.histplot(x = "tenure", data = df, bins = 72, hue = "Churn")
plt.show()
```



In []:

people who have used our services for a long time have stayed and people who have used our sevices 1 or 2 months have churned

```
In [37]: plt.figure(figsize = (4,4))
ax = sns.countplot(x = "Contract", data = df, hue = "Churn")
ax.bar_label(ax.containers[0])
plt.title("Count of Customers by Contract")
plt.show()
```



people who have month to month contract are likely to churn then from those who have 1 or 2 years or contract.

```
In [38]: df.columns.values
```

```
Out[38]: array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

```
In [39]: columns = ['PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', '
# Number of columns for the subplot grid (you can change this)
n_cols = 3
n_rows = (len(columns) + n_cols - 1) // n_cols # Calculate number of rows nee
```

```

# Create subplots
fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, n_rows * 4)) # Adjust figsize

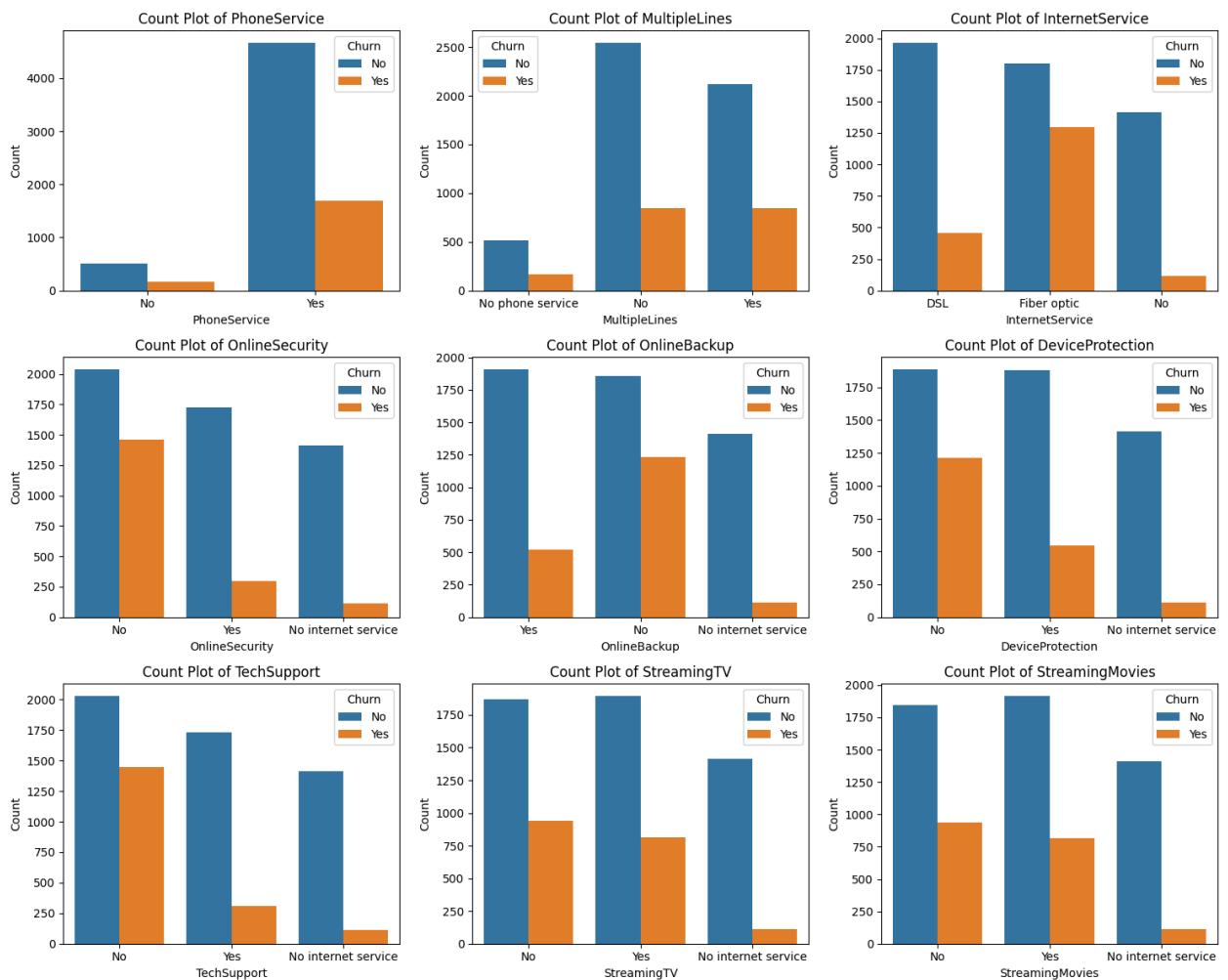
# Flatten the axes array for easy iteration (handles both 1D and 2D arrays)
axes = axes.flatten()

# Iterate over columns and plot count plots
for i, col in enumerate(columns):
    sns.countplot(x=col, data=df, ax=axes[i], hue = df["Churn"])
    axes[i].set_title(f'Count Plot of {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Count')

# Remove empty subplots (if any)
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()

```

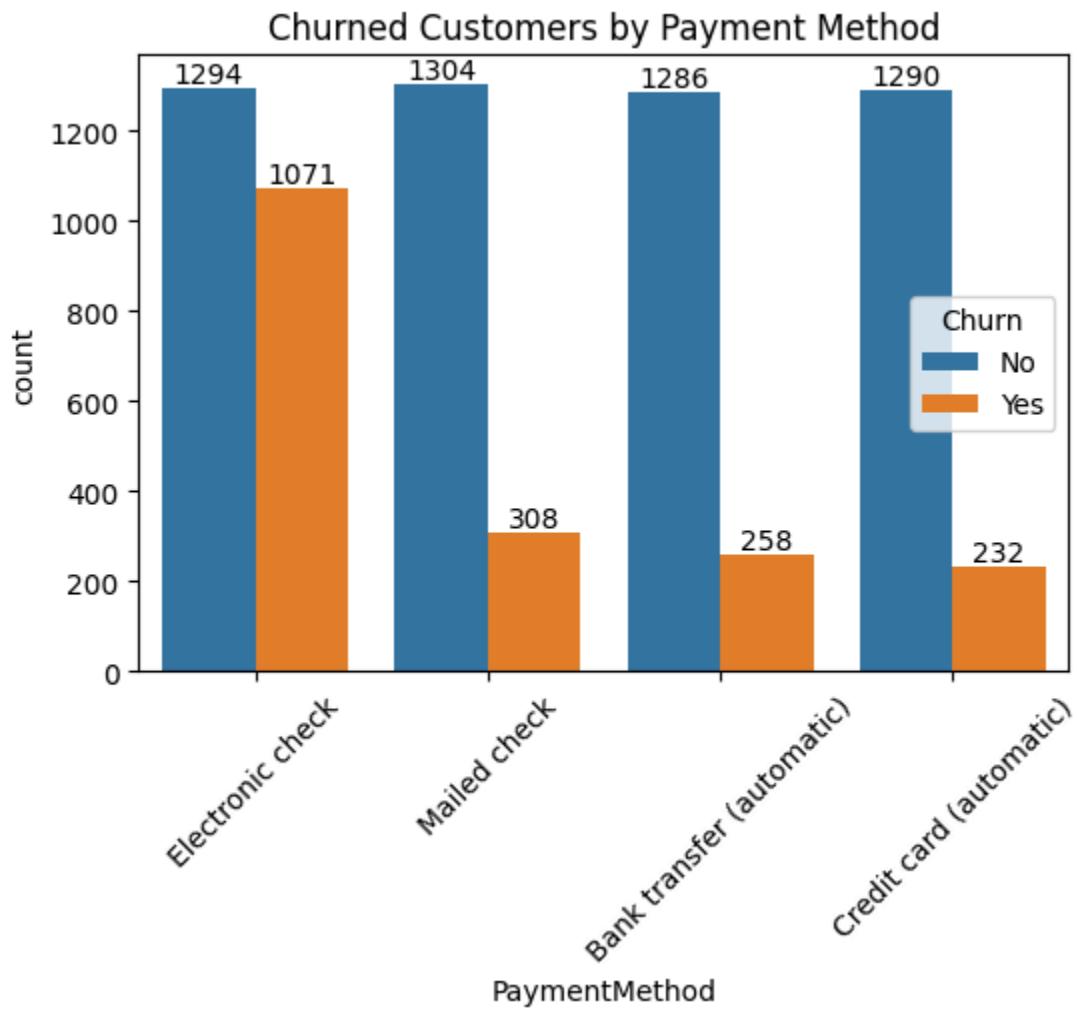


In []:

The majority of customers who do not churn tend to have services like

PhoneService, InternetService (particularly DSL), and OnlineSecurity enabled. For services like OnlineBackup, TechSupport, and StreamingTV, churn rates are noticeably higher when these services are not used or are unavailable.

```
In [40]: plt.figure(figsize = (6,4))
ax = sns.countplot(x = "PaymentMethod", data = df, hue = "Churn")
ax.bar_label(ax.containers[0])
ax.bar_label(ax.containers[1])
plt.title("Churned Customers by Payment Method")
plt.xticks(rotation = 45)
plt.show()
```



customer is likely to churn when he is using electronic check as a payment method.