churn-analysis

November 25, 2024

[6]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('customer churn.csv')
df.head()
```

[6]:
```
   customerID  gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1           No
1  5575-GNVDE    Male              0      No         No      34          Yes
2  3668-QPYBK    Male              0      No         No       2          Yes
3  7795-CFOCW    Male              0      No         No      45           No
4  9237-HQITU  Female              0      No         No       2          Yes

      MultipleLines InternetService OnlineSecurity  … DeviceProtection  \
0  No phone service             DSL             No  …               No
1                No             DSL            Yes  …              Yes
2                No             DSL            Yes  …               No
3  No phone service             DSL            Yes  …              Yes
4                No     Fiber optic             No  …               No

  TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No          No              No  Month-to-month              Yes
1          No          No              No        One year               No
2          No          No              No  Month-to-month              Yes
3         Yes          No              No        One year               No
4          No          No              No  Month-to-month              Yes

               PaymentMethod MonthlyCharges  TotalCharges Churn
0           Electronic check          29.85         29.85    No
1               Mailed check          56.95        1889.5    No
2               Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4           Electronic check          70.70        151.65   Yes

[5 rows x 21 columns]
```

[12]:
```
df["TotalCharges"] = df["TotalCharges"].replace(" " , "0 ")
df["TotalCharges"] = df["TotalCharges"].astype("float")
```

1 Replacing the blanks with 0 as tenure is 0 no total charges are recorded

[13]:
```
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
```

[16]:
```
df.isnull().sum().sum()
```

[16]:
```
np.int64(0)
```

[17]:
```
df.describe()
```

[17]:
```
       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
```

[21]:
```
df["customerID"].duplicated().sum()
```

[21]:
```
np.int64(0)
```

2 we have converted 0 and 1 value of senior citizen into yes/no

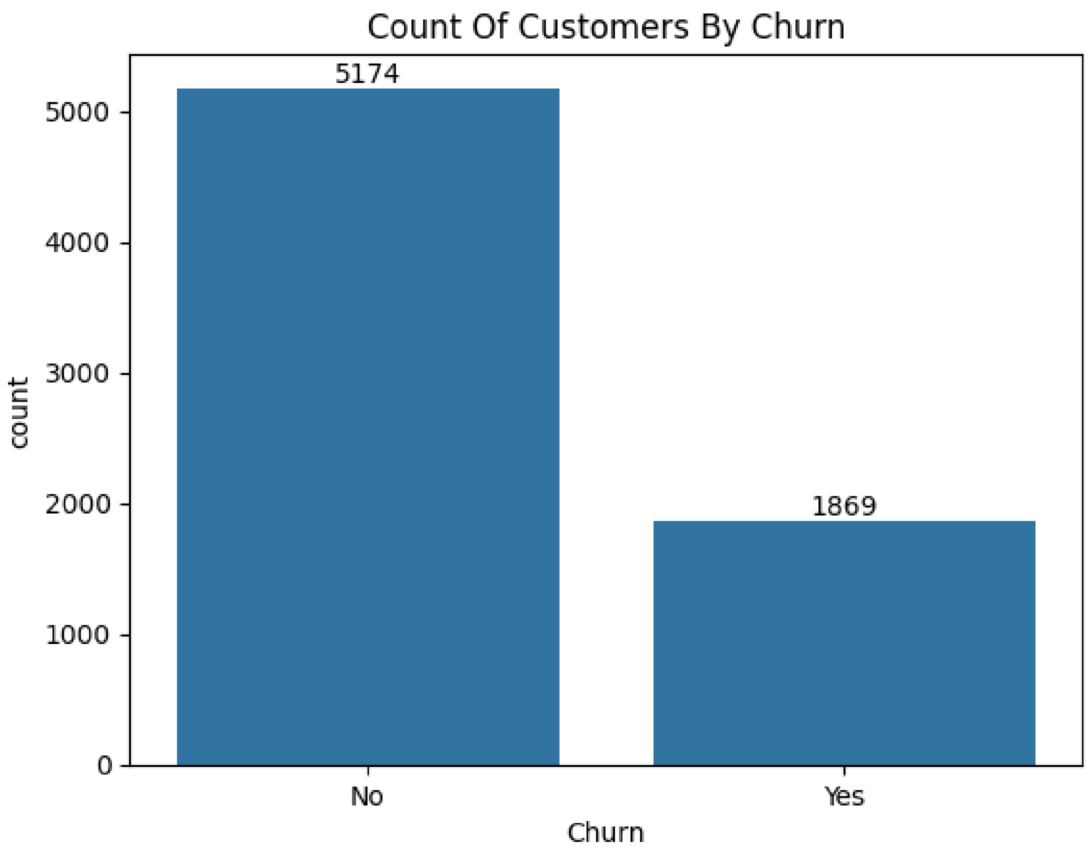
[23]:
```
def conv(value):
    if value == 1:
        return "yes"
    else:
        return "no"

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

[24]:
```
df.head(10)
```

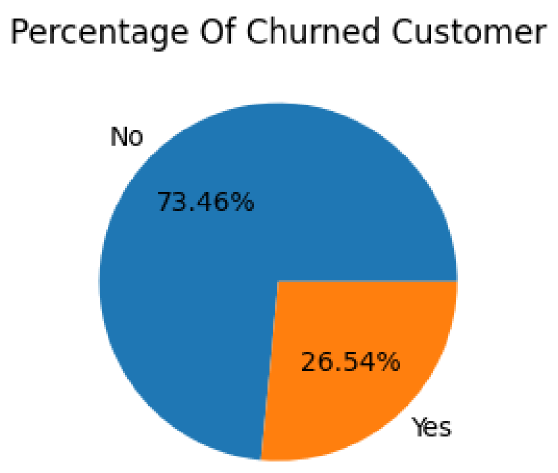
[24]:
```
   customerID  gender SeniorCitizen Partner Dependents  tenure PhoneService  \
0  7590-VHVEG  Female            no     Yes         No       1           No
1  5575-GNVDE    Male            no      No         No      34          Yes
2  3668-QPYBK    Male            no      No         No       2          Yes
3  7795-CFOCW    Male            no      No         No      45           No
4  9237-HQITU  Female            no      No         No       2          Yes
5  9305-CDSKC  Female            no      No         No       8          Yes
6  1452-KIOVK    Male            no      No        Yes      22          Yes
7  6713-OKOMC  Female            no      No         No      10           No
8  7892-POOKP  Female            no     Yes         No      28          Yes
9  6388-TABGU    Male            no      No        Yes      62          Yes

      MultipleLines InternetService OnlineSecurity  … DeviceProtection  \
0  No phone service             DSL             No  …               No
1                No             DSL            Yes  …              Yes
2                No             DSL            Yes  …               No
3  No phone service             DSL            Yes  …              Yes
4                No     Fiber optic             No  …               No
5               Yes     Fiber optic             No  …              Yes
6               Yes     Fiber optic             No  …               No
7  No phone service             DSL            Yes  …               No
8               Yes     Fiber optic             No  …              Yes
9                No             DSL            Yes  …               No
```

```
  TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No          No              No  Month-to-month              Yes
1          No          No              No        One year               No
2          No          No              No  Month-to-month              Yes
3         Yes          No              No        One year               No
4          No          No              No  Month-to-month              Yes
5          No         Yes             Yes  Month-to-month              Yes
6          No         Yes              No  Month-to-month              Yes
7          No          No              No  Month-to-month               No
8         Yes         Yes             Yes  Month-to-month              Yes
9          No          No              No        One year               No

              PaymentMethod MonthlyCharges  TotalCharges  Churn
0          Electronic check          29.85         29.85     No
1              Mailed check          56.95       1889.50     No
2              Mailed check          53.85        108.15    Yes
3 Bank transfer (automatic)          42.30       1840.75     No
4          Electronic check          70.70        151.65    Yes
5          Electronic check          99.65        820.50    Yes
6   Credit card (automatic)          89.10       1949.40     No
7              Mailed check          29.75        301.90     No
8          Electronic check         104.80       3046.05    Yes
9 Bank transfer (automatic)          56.15       3487.95     No

[10 rows x 21 columns]
```

[28]:
```
ax = sns.countplot(x= 'Churn' , data = df )

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

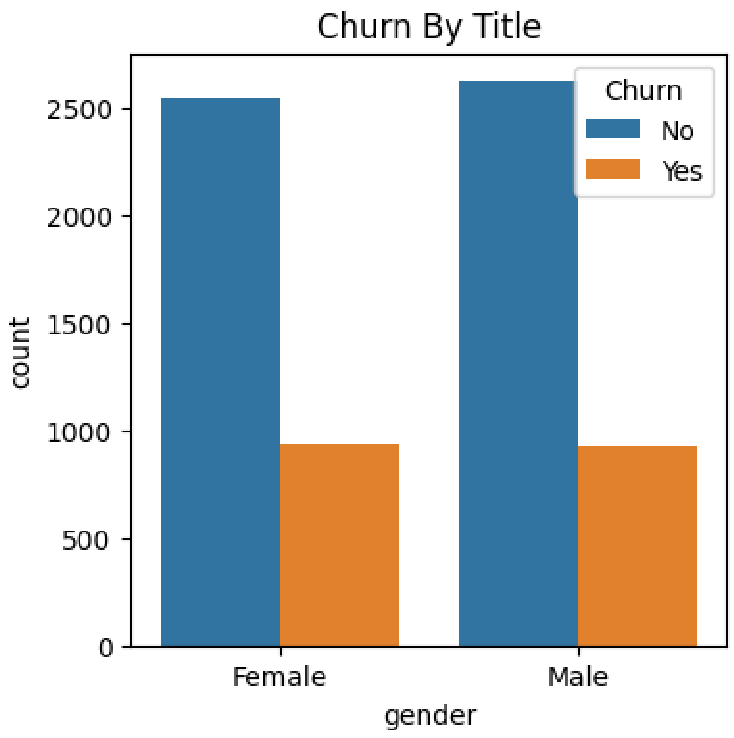


[30]:
```
plt.figure(figsize = (3,4))
gb = df.groupby("Churn").agg({'Churn': "count"})
plt.pie(gb["Churn"] , labels = gb.index , autopct = "%1.2f%%")
plt.title("Percentage Of Churned Customer")
plt.show()
```

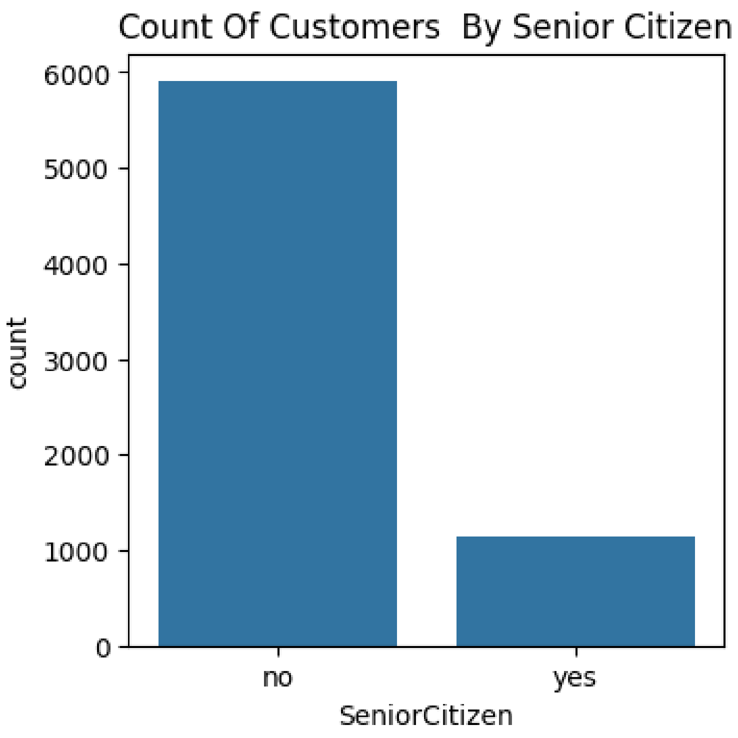


3 from the given pie chart we can conclude that 26.54 % customers have churned out . Now lets figure out the reason behind it

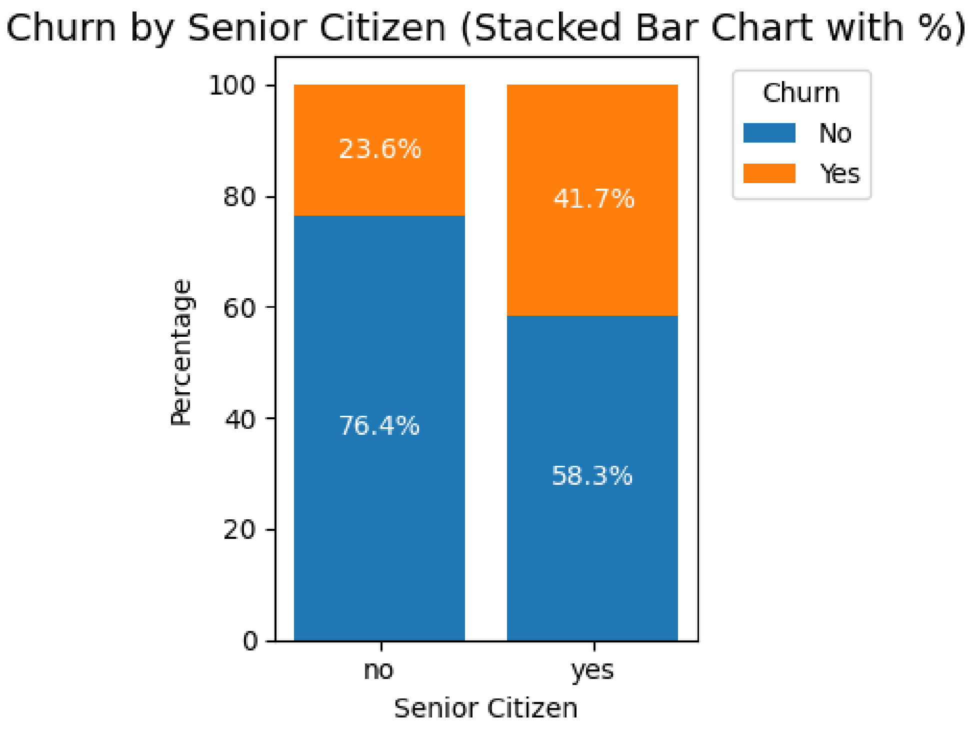
[36]:
```
plt.figure(figsize = (4,4))
sns.countplot(x = "gender" , data = df , hue = "Churn")
plt.title("Churn By Title")
plt.show()
```



[52]:
```
plt.figure(figsize = (4,4))
sns.countplot(x = "SeniorCitizen" , data = df )
ax.bar_label(ax.containers[0])
plt.title("Count Of Customers  By Senior Citizen")
plt.show()
```

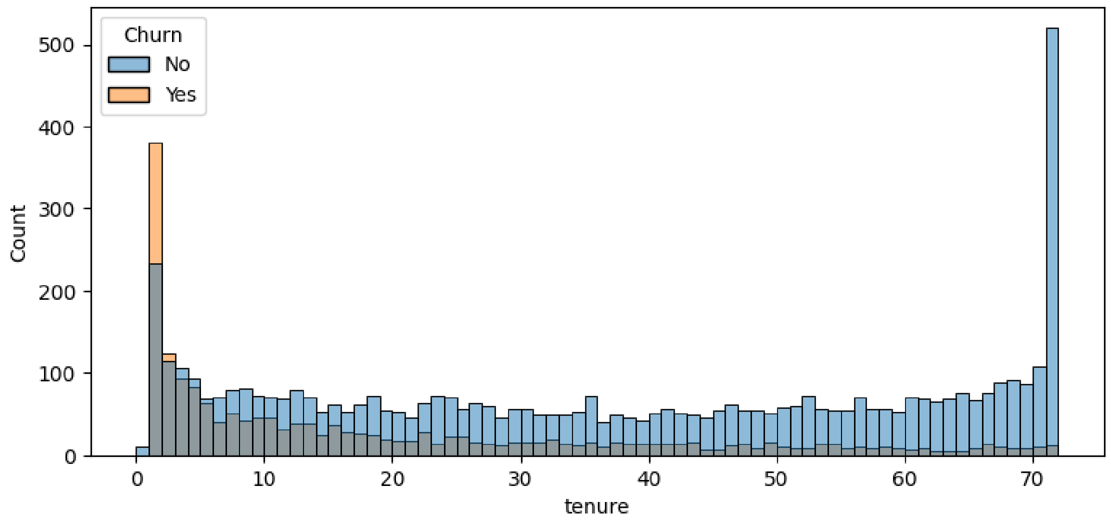


[43]:
```
grouped = df.groupby(['SeniorCitizen', 'Churn']).size().
↪reset_index(name='Count')

# Calculate the percentage contribution within each SeniorCitizen group
grouped['Percentage'] = grouped.groupby('SeniorCitizen')['Count'].
↪transform(lambda x: x / x.sum() * 100)

# Pivot the data to prepare for stacked bar plotting
pivot_data = grouped.pivot(index='SeniorCitizen', columns='Churn',␣
↪values='Percentage').fillna(0)

# Step 2: Plot the stacked bar chart
fig, ax = plt.subplots(figsize=(4, 4))
bottoms = [0] * len(pivot_data)

for churn_value in pivot_data.columns:
    ax.bar(
        pivot_data.index,  # x-axis labels (SeniorCitizen categories)
        pivot_data[churn_value],  # bar heights (percentages)
        bottom=bottoms,  # bottom of the bar for stacking
        label=churn_value
    )
    # Update the bottom positions for stacking
    bottoms = [i + j for i, j in zip(bottoms, pivot_data[churn_value])]

# Step 3: Add percentage labels
for i, senior in enumerate(pivot_data.index):
    cumulative_bottom = 0
    for churn_value in pivot_data.columns:
        value = pivot_data.loc[senior, churn_value]
        if value > 0:  # Only annotate non-zero segments
            ax.text(
                i,
                cumulative_bottom + value / 2,
                f'{value:.1f}%',
                ha='center',
                va='center',
                color='white' if value > 10 else 'black',  # Adjust text color␣
 ↪for readability
                fontsize=10
            )
        cumulative_bottom += value

# Step 4: Customize the plot
ax.set_title("Churn by Senior Citizen (Stacked Bar Chart with %)", fontsize=14)
ax.set_xlabel("Senior Citizen")
ax.set_ylabel("Percentage")
ax.legend(title='Churn', bbox_to_anchor=(1.05, 1), loc='upper left')

plt.tight_layout()
plt.show()
```



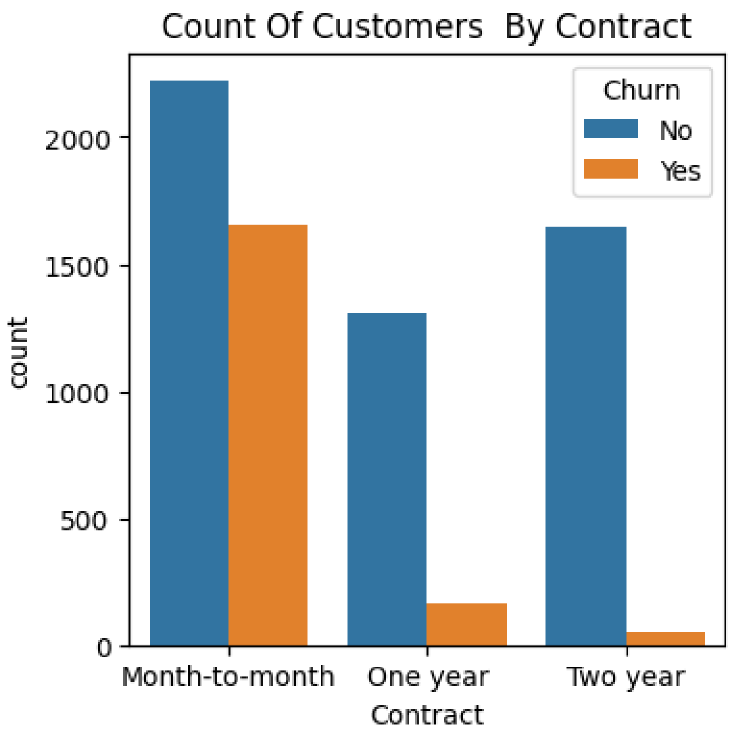
#Comparative a greater percentage of people in senior citizen category have churned

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



#People who have used our services for a long time have stayed and people who have used our services for 1 -2 months have churned out

[53]:
```
plt.figure(figsize = (4,4))
sns.countplot(x = "Contract" , data = df, hue = "Churn")
ax.bar_label(ax.containers[0])
plt.title("Count Of Customers  By Contract")
plt.show()
```

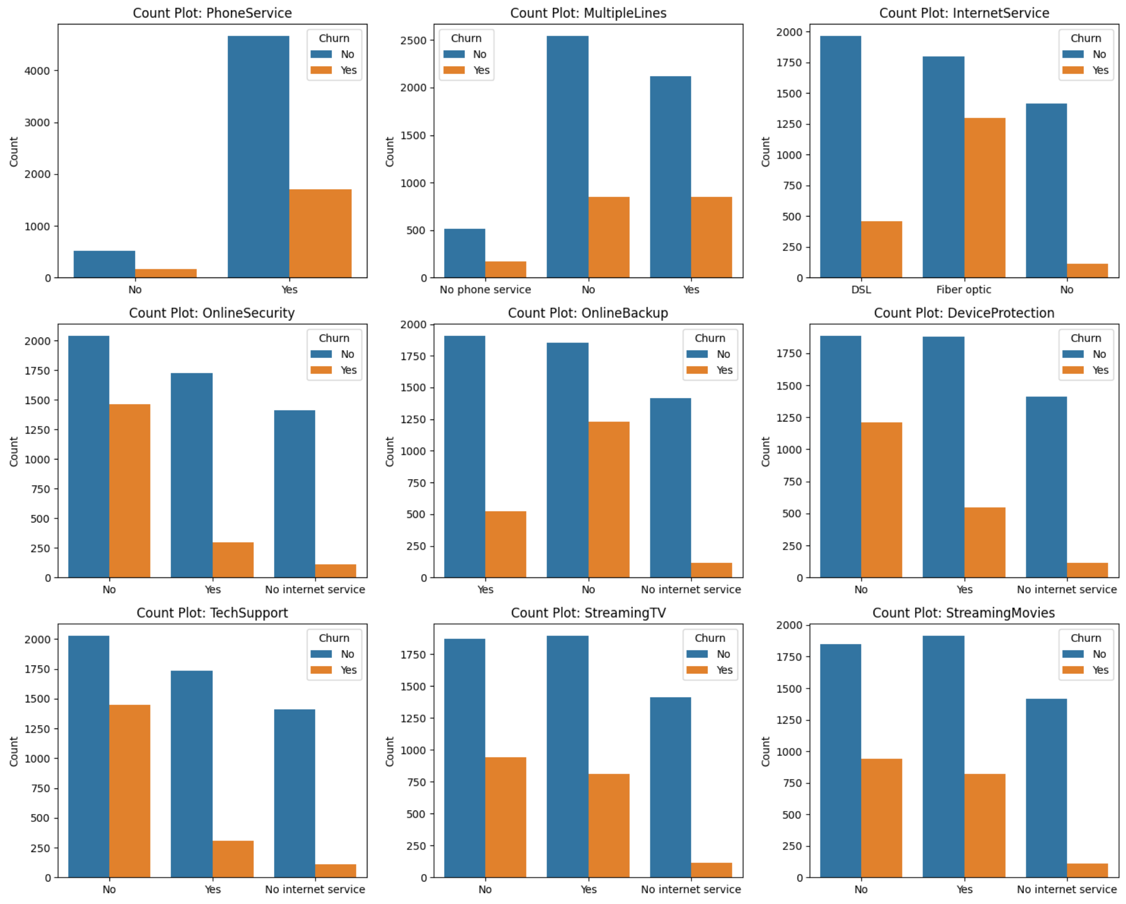


#people who have month to month contract are likrly to churn then from those who have 1-2 years of contract

[54]:
```
df.columns.values
```

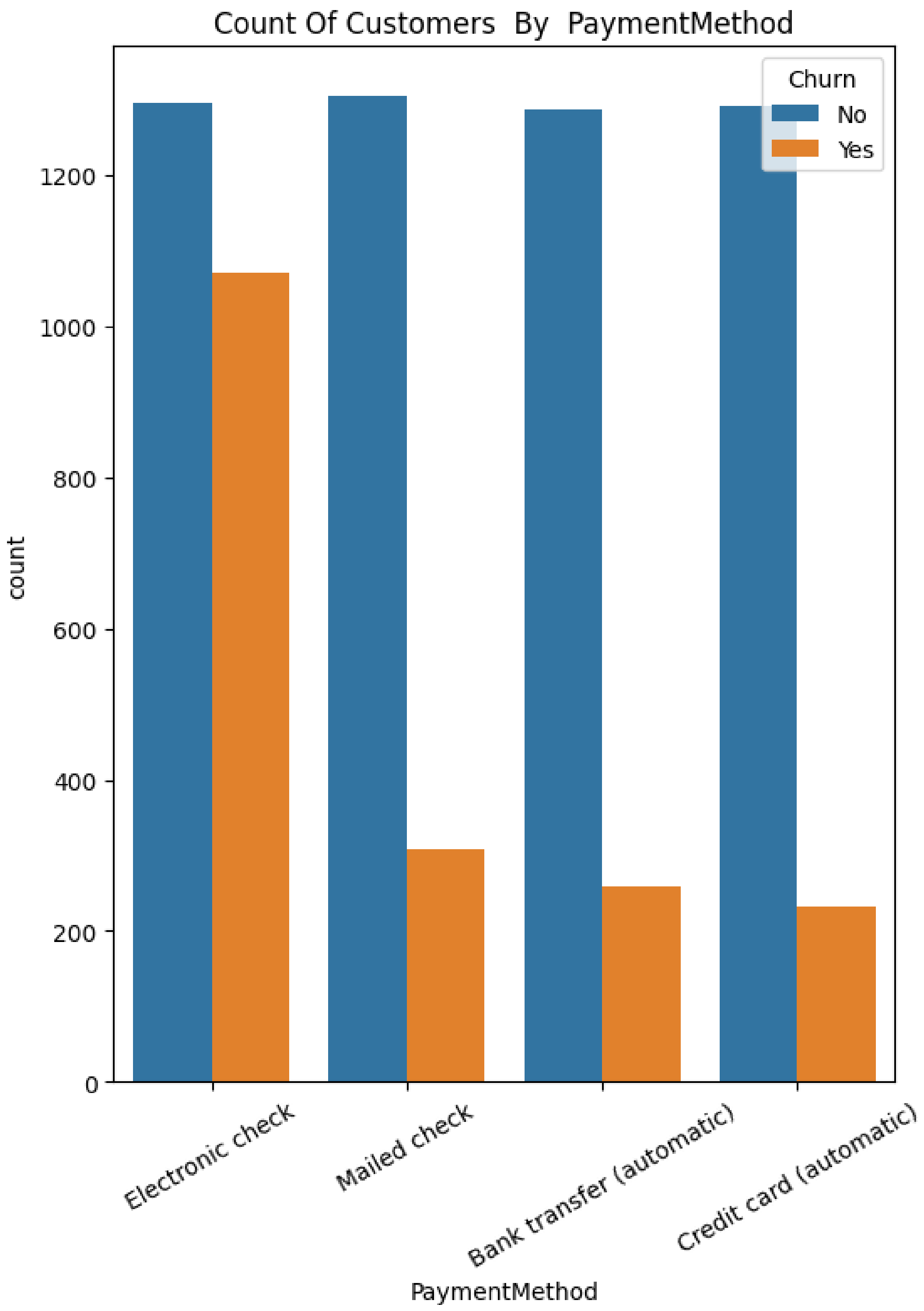
[54]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

[63]:
```
# Columns to plot (categorical columns)
columns_to_plot = ['PhoneService',
    'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',
    'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']

# Define the number of rows and columns for subplots
n_cols = 3  # Number of columns in the subplot grid
n_rows = -(-len(columns_to_plot) // n_cols)  # Calculate rows needed (ceil␣
 ↪division)

# Create subplots
fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, n_rows * 4))
axes = axes.flatten()  # Flatten to iterate easily

# Loop through columns and create count plots
for i, col in enumerate(columns_to_plot):
    ax = axes[i]
    sns.countplot(data=df, x=col, ax=ax, hue = df["Churn"])
    ax.set_title(f"Count Plot: {col}", fontsize=12)
    ax.set_xlabel("")
    ax.set_ylabel("Count")
    ax.tick_params(axis='x', rotation=0)

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

# Adjust layout
plt.tight_layout()
plt.show()
```



#Services such as “OnlineSecurity,” “OnlineBackup,” and “TechSupport” show higher churn rates when customers do not have these features. “Fiber optic” internet service appears to have a higher churn rate compared to “DSL” or no internet service. The availability of features like “StreamingTV” and “StreamingMovies” does not clearly favor retention, as churn is relatively similar between “Yes” and “No” groups.

[68]:
```
plt.figure(figsize = (6,8))
sns.countplot(x = "PaymentMethod" , data = df, hue = "Churn")
ax.bar_label(ax.containers[0])
ax.bar_label(ax.containers[1])
plt.xticks(rotation = 30 )
plt.title("Count Of Customers  By  PaymentMethod")
plt.show()
```



#Customer is likely to churn when he is using electronic check as a payment method

```
[ ]:
```