```
!PIP INSTALL PANDAS
```

```
/bin/bash: line 1: PIP: command not found
```

```
PIP INSTALL MATPLOTLIB
```

```
  File "/tmp/ipython-input-14-871634849.py", line 1
    PIP INSTALL MATPLOTLIB
        ^
SyntaxError: invalid syntax
```

```
PIP INSTALL SEABORN;
```

```
PIP INSTALL NUMPY
```

```
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
```

```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving Customer Churn.csv to Customer Churn (2).csv
```

```
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
```

```
#replacing blank with 0 as tenure is 0 and total charges are recorded

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

#replacing blank with 0 as tenure is 0 and total charges are recorded

```
df.isnull()
```

{"type":"dataframe"}

```
df.isnull().sum()
```

```
customerID          0
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
```

```
df.isnull().sum().sum()
```

```
np.int64(0)
```

df.isnull is showing, the null value in dataframes and it makes true, whichever shows false.

df.isnull().sum()- sum check shows 0 in dataframe

df.innull().sum().sum() - it shows overall 0 in dataframe

```
df.describe()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 8,\n \"fields\": [\n{\n   \"column\": \"SeniorCitizen\",\n   \"properties\": {\n\"dtype\": \"number\",\n   \"std\": 2489.9992387084,\n\"min\": 0.0,\n   \"max\": 7043.0,\n\"num_unique_values\": 5,\n   \"samples\": [\n0.1621468124378816,\n   1.0,\n   0.36861160561002687\n],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n}\n   },\n   {\n   \"column\": \"tenure\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 2478.9752758409018,\n   \"min\": 0.0,\n   \"max\": 7043.0,\n

```
\"num_unique_values\": 8,\n        \"samples\": [\n
32.37114865824223,\n          29.0,\n          7043.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"MonthlyCharges\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2468.7047672837775,\n        \"min\": 18.25,\n        \"max\":
7043.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
64.76169246059918,\n          70.35,\n          7043.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"TotalCharges\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3122.5732655623974,\n        \"min\": 0.0,\n        \"max\": 8684.8,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
2279.7343035638223,\n          1394.55,\n          7043.0\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```

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

```
np.int64(0)
```

```
df.duplicated().sum()
```

```
np.int64(0)
```

#converted 0 and 1 value of Senior Citizen to yes/no to make it easier to understand

```
def conv(value):
  if value == 1:
    return "Yes"
  else:
    return "No"
df['SeniotrCitizen'] = df['SeniorCitizen'].apply(conv)
df.head()
```

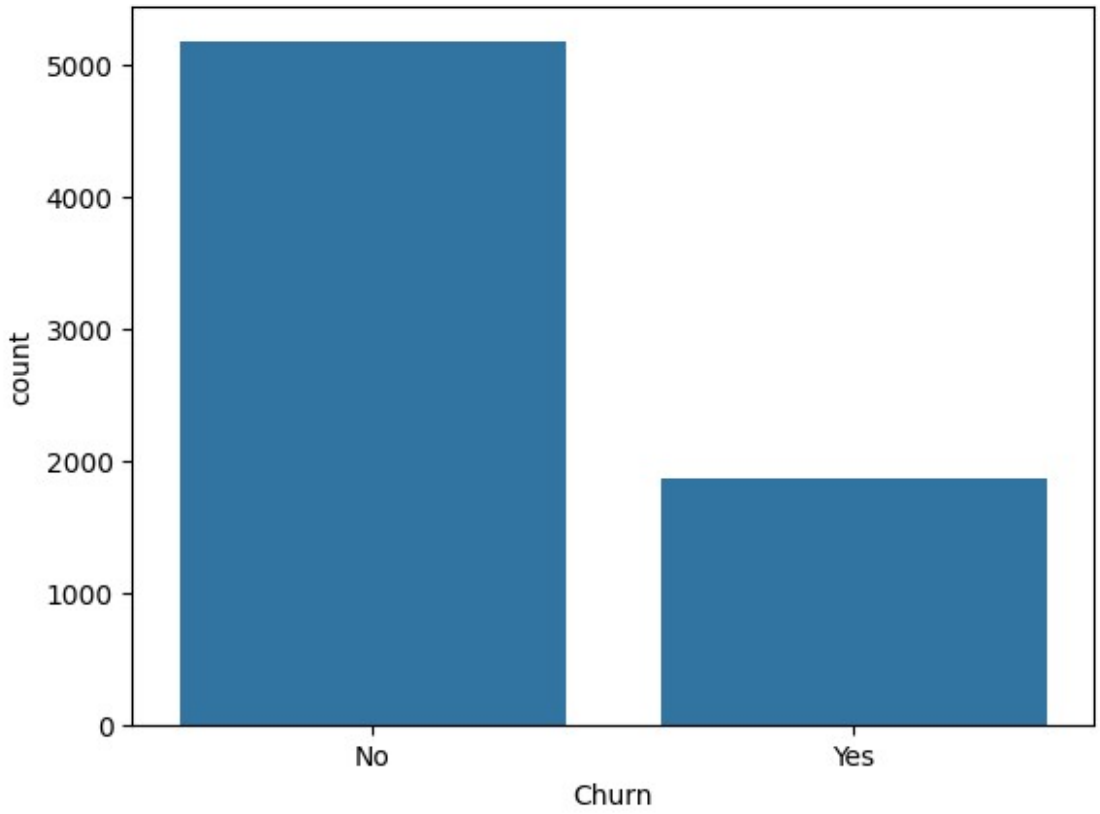
```
{"type":"dataframe","variable_name":"df"}
```

```
df.head(30)
```

```
{"type":"dataframe","variable_name":"df"}
```

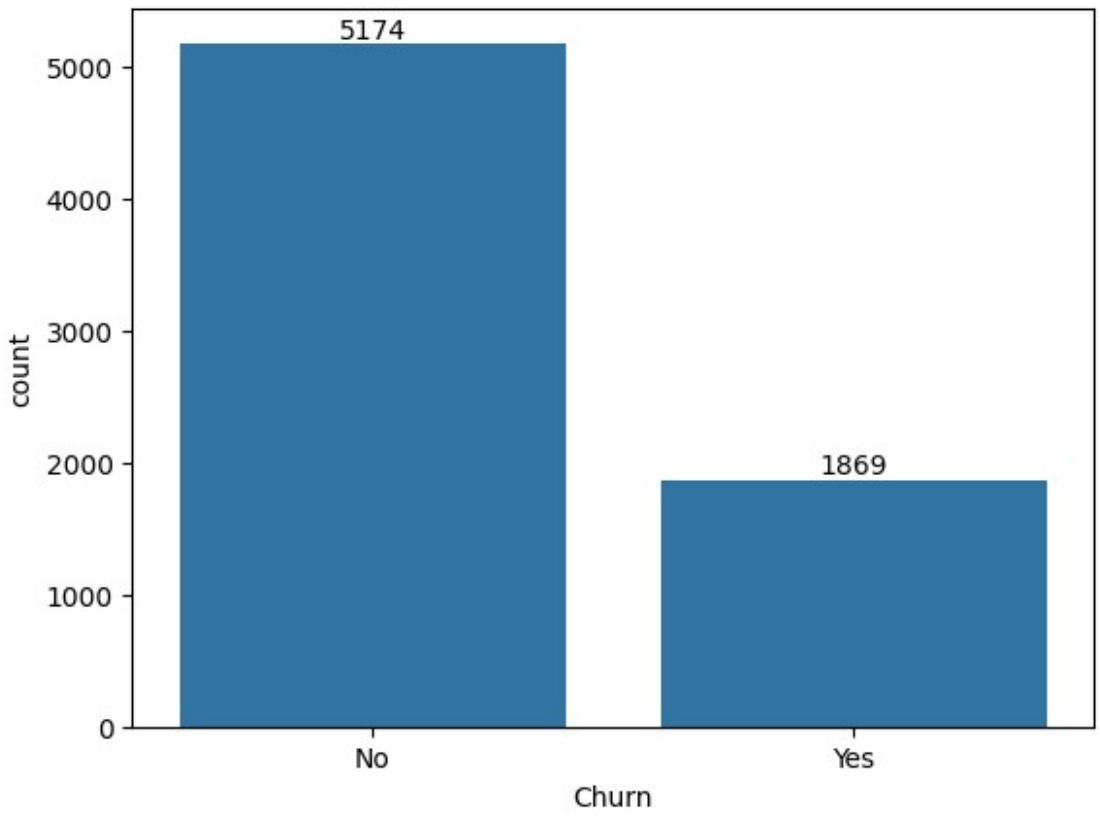
#df.head() - it is used for rows 30, 20 whatever, in senior citizen it shows yes in row for checking the rows.

```
sns.countplot(x = df['Churn'], data = df)
plt.show()
```



in counplot - it shows yes/no value

```
import seaborn as sns
import matplotlib.pyplot as plt

ax = sns.countplot(x = 'Churn', data = df)

ax.bar_label(ax.containers[0])
plt.show()
```

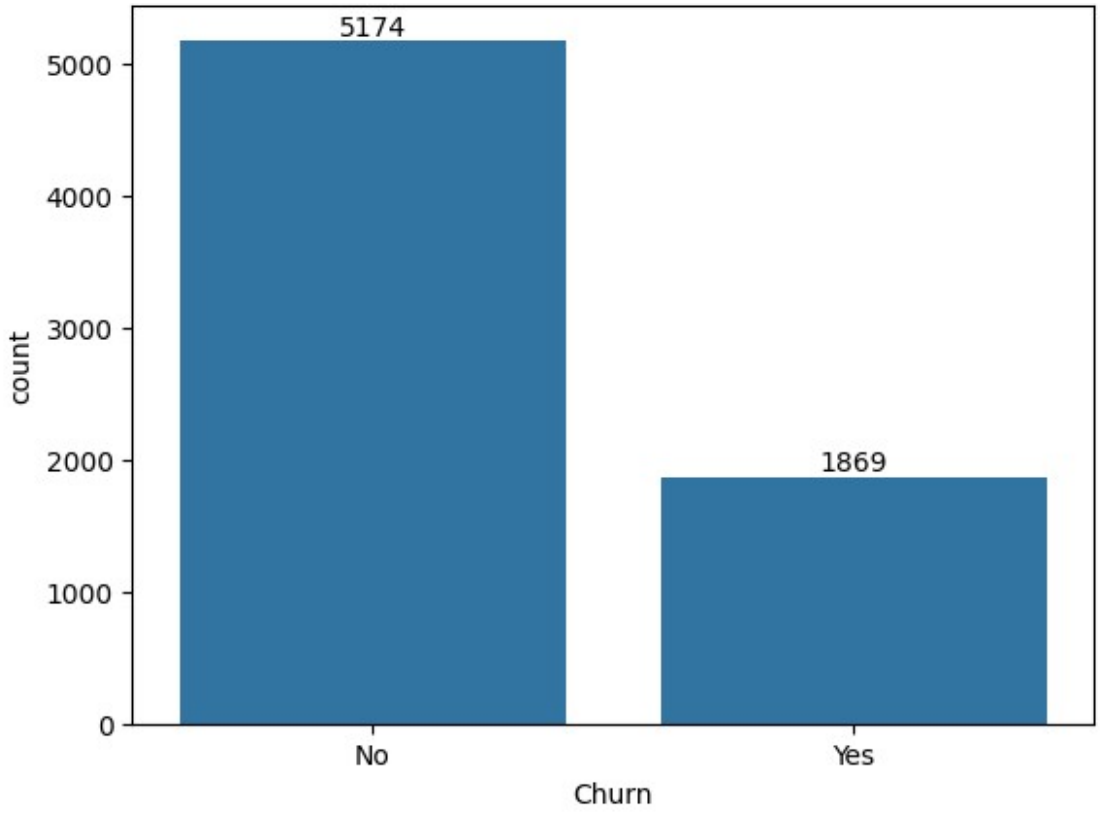



```
df.describe()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8,\n  \"fields\": [\n
{\n      \"column\": \"SeniorCitizen\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 2489.9992387084,\n
\"min\": 0.0,\n        \"max\": 7043.0,\n
\"num_unique_values\": 5,\n        \"samples\": [\n
0.1621468124378816,\n          1.0,\n          0.36861160561002687\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"tenure\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
2478.9752758409018,\n        \"min\": 0.0,\n        \"max\": 7043.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
32.37114865824223,\n          29.0,\n          7043.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"MonthlyCharges\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2468.7047672837775,\n        \"min\": 18.25,\n        \"max\":
7043.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
64.76169246059918,\n          70.35,\n          7043.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"TotalCharges\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3122.5732655623974,\n        \"min\": 0.0,\n        \"max\": 8684.8,\n
```

\"num_unique_values\": 8,\n   \"samples\": [\n
2279.7343035638223,\n   1394.55,\n   7043.0\n   ],\
n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe"}

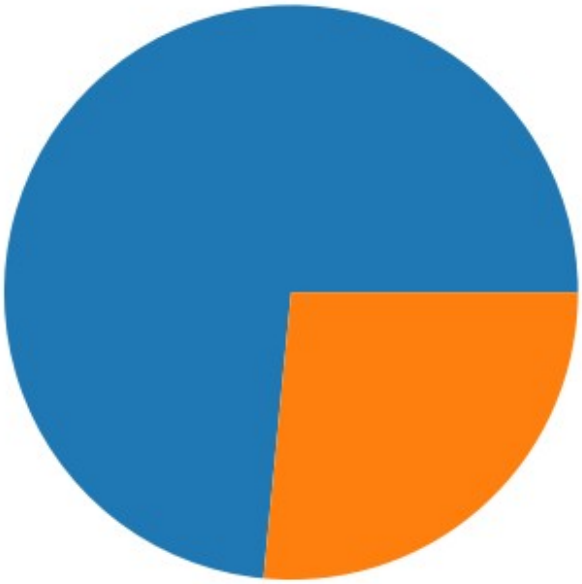
```
ax = sns.countplot(x = 'Churn', data = df)

ax.bar_label(ax.containers[0])
plt.show()
```



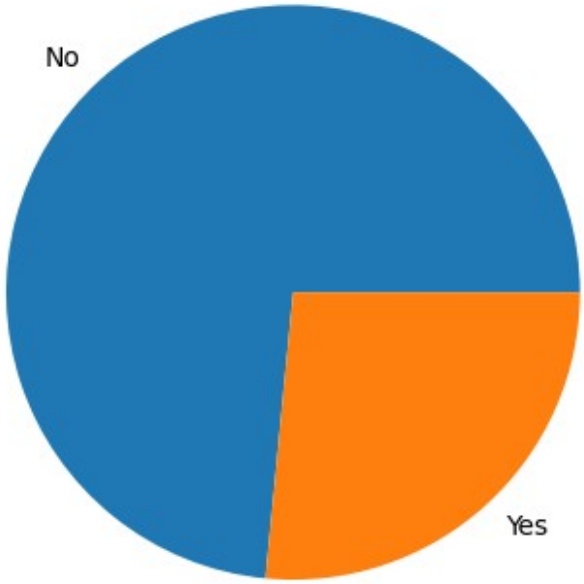

```
gb = df.groupby("Churn").agg({'Churn':"count"})
gb
#plt.pie(df['Churn'])
#plt.show()
```

{"repr_error":"cannot insert Churn, already
exists","type":"dataframe","variable_name":"gb"}

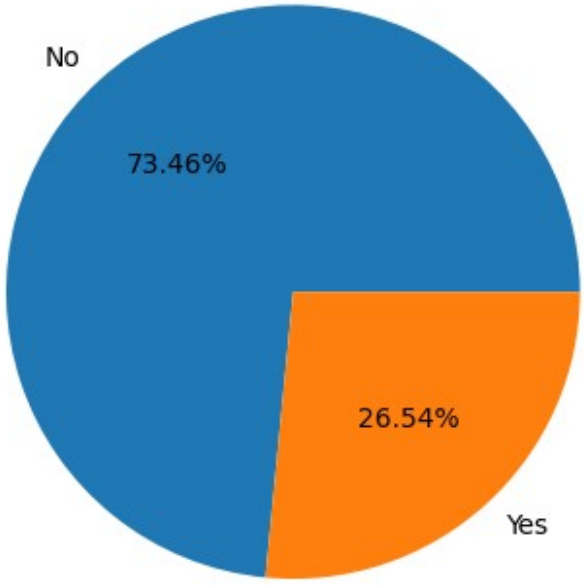
```
gb = df.groupby("Churn").agg({'Churn':"count"})
plt.pie(gb['Churn'])
plt.show()
gb
```



```
{"repr_error":"cannot insert Churn, already
exists","type":"dataframe","variable_name":"gb"}

gb = df.groupby("Churn").agg({'Churn':"count"})
plt.pie(gb['Churn'], labels = gb.index)
plt.show()
```

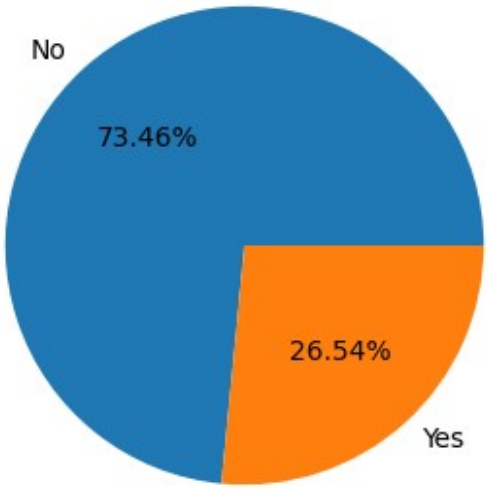



```
gb = df.groupby("Churn").agg({'Churn':"count"})
plt.pie(gb['Churn'], labels = gb.index, autopct = "%1.2f%%")
plt.show()
```

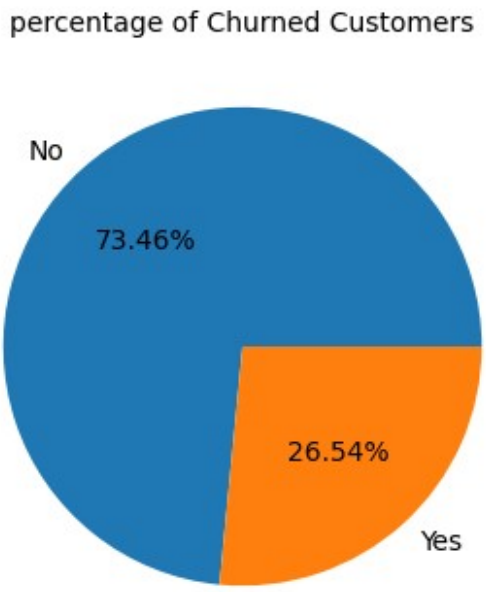


##26.5% customer are churn out from the total customer, i.e.(converted into another sim).

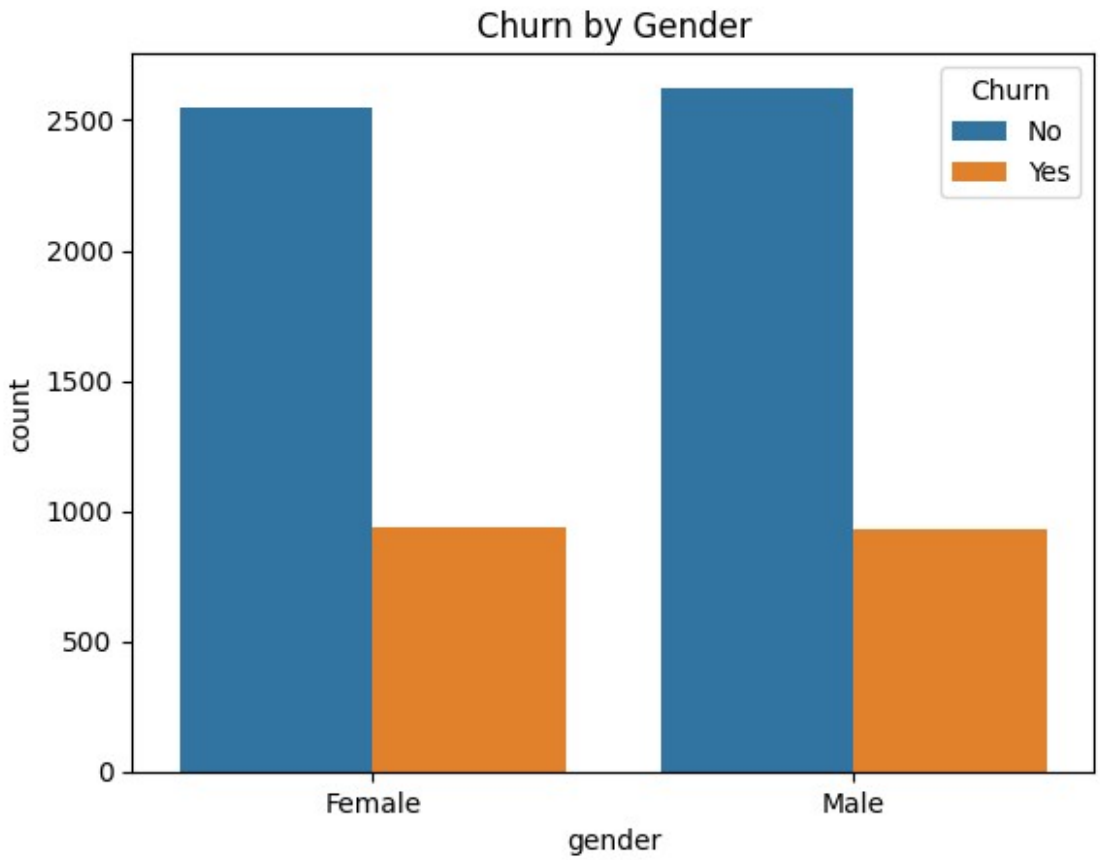
```
plt.figure(figsize = (4,4))
gb = df.groupby("Churn").agg({'Churn':"count"})
plt.pie(gb['Churn'], labels = gb.index, autopct = "%1.2f%%")
plt.show()
```




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

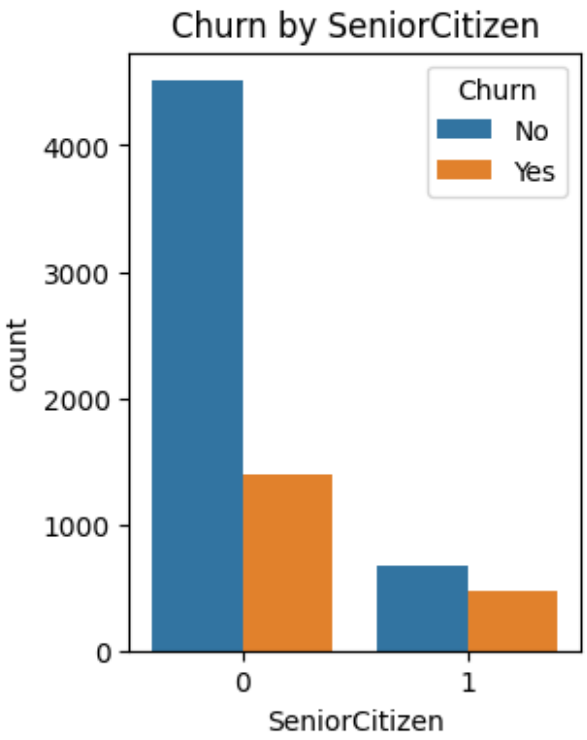



```
#from the gien pie chart we can couclude that 26.54% of our customers
have churned out
#now lt\ets's explore the reason behind it.

sns.countplot(x = 'gender', data = df, hue ='Churn')
plt.title ("Churn by Gender")
plt.show()
```



#hue -its denoted, coulmn basis of churn

```
plt.figure (figsize = (3,4))
sns.countplot(x = 'SeniorCitizen', data = df, hue ='Churn')
plt.title ("Churn by SeniorCitizen")
plt.show()
```



#copied the code and ask to chgatgpt

#plt.figure (figsize = (3,4))

#sns.countplot(x = 'SeniorCitizen', data = df, hue ='Churn')

#plt.title ("Churn by SeniorCitizen")

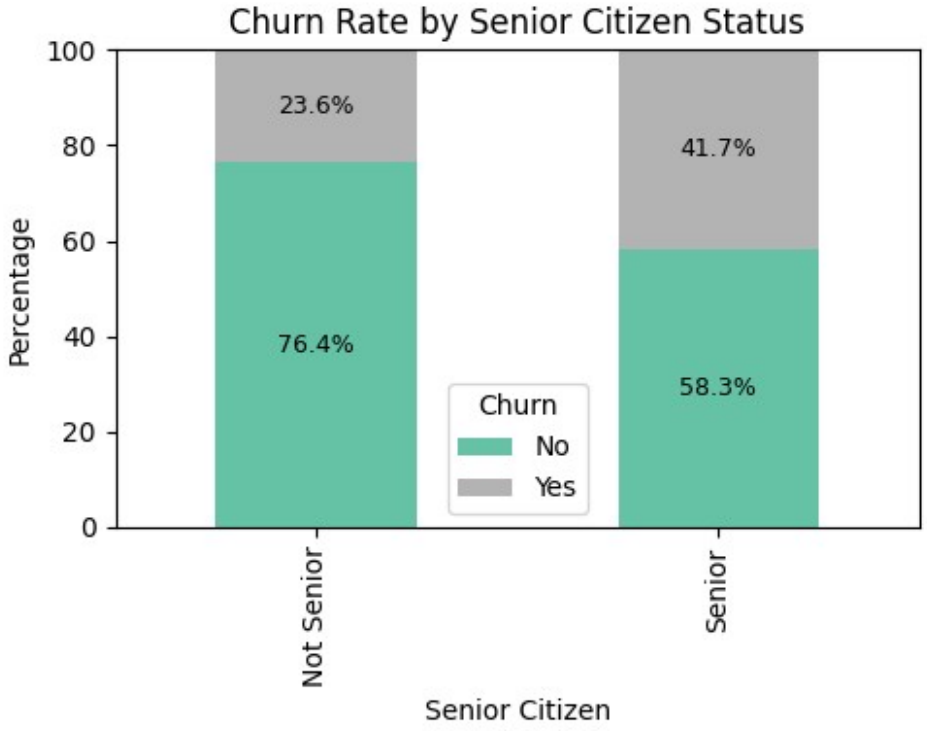
#plt.show()

#i want to create a stack bar chart which gives me labels as % to total

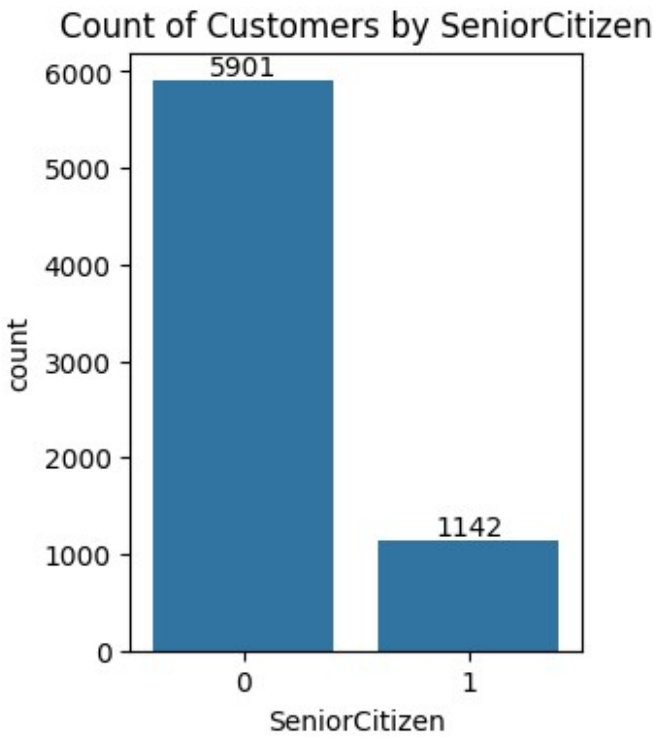
#chatgpt coding given below

```
ct = pd.crosstab(df['SeniorCitizen'], df['Churn'], normalize='index')
* 100

# 2. Plot stacked bar
ax = ct.plot(kind='bar', stacked=True, figsize=(5,4), colormap='Set2')

# 3. Add % labels to each segment
for i, row in ct.iterrows():
    cumulative = 0
    for churn_value in ct.columns:
        percent = row[churn_value]
        if percent > 0:
            ax.text(i, cumulative + percent / 2, f'{percent:.1f}%',
ha='center', va='center', fontsize=9)
```

```
            cumulative += percent

# 4. Styling
plt.title("Churn Rate by Senior Citizen Status")
plt.xlabel("Senior Citizen")
plt.ylabel("Percentage")
plt.xticks([0,1], ["Not Senior", "Senior"])  # Optional: label 0/1
plt.legend(title='Churn')
plt.ylim(0, 100)
plt.tight_layout()
plt.show()
```

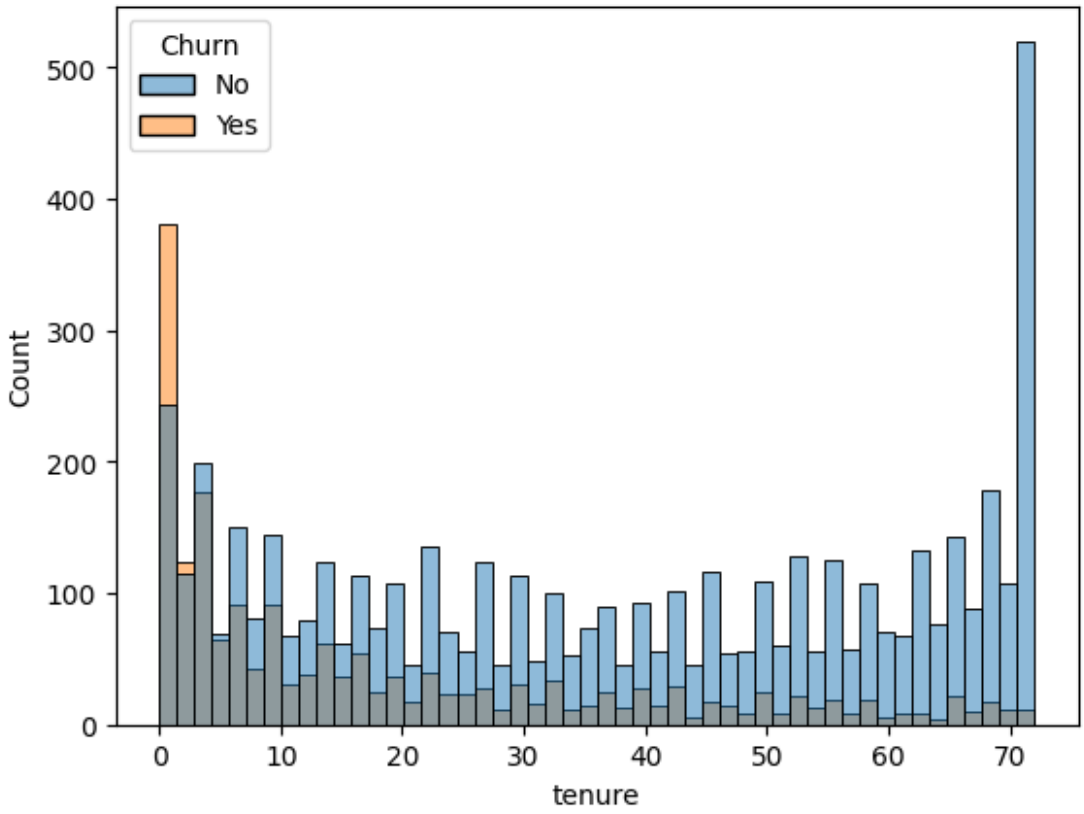



```
plt.figure (figsize = (3,4))
ax = sns.countplot(x = 'SeniorCitizen', data = df)
ax.bar_label(ax.containers[0])
plt.title ("Count of Customers by SeniorCitizen")
plt.show()
```

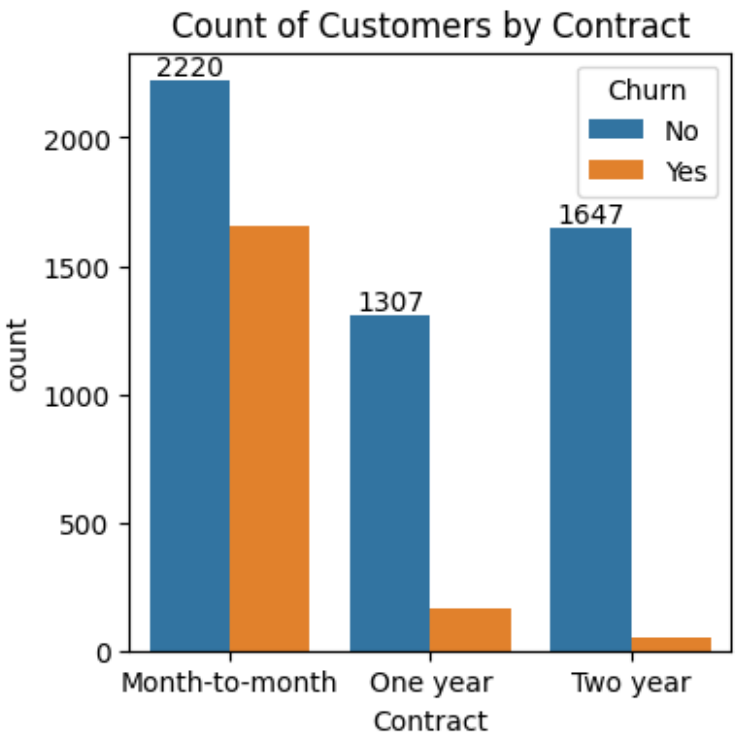


#COMPARATIVELY a greater percentage of people in seniorcitizen category have churched out.

```
sns.histplot (x = 'tenure', data = df, bins = 50, hue ="Churn")
plt.show()
```



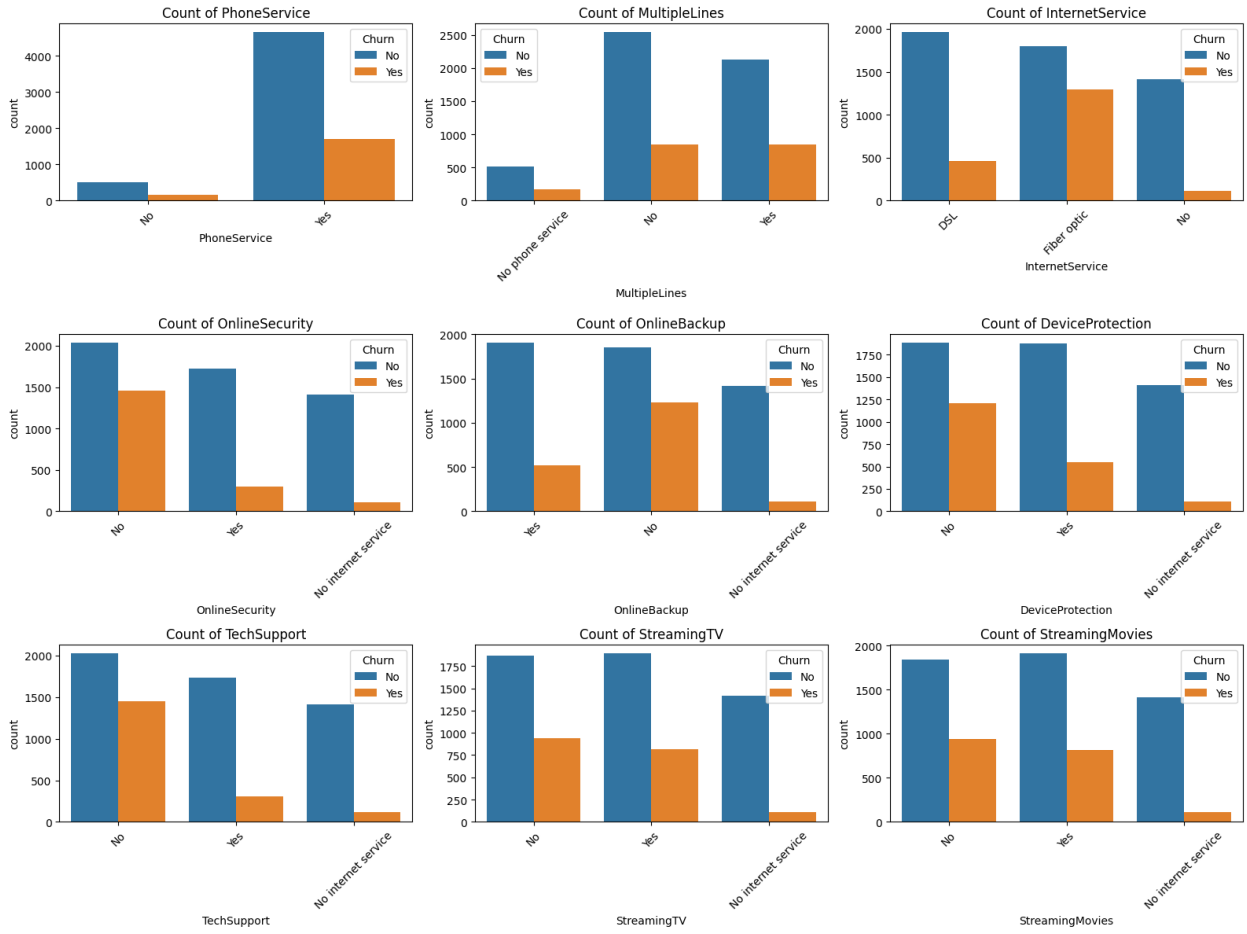

```
#people who have used our services for a long time have stayed and
peole who have used our srvices for 1 or 2 month have churned

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 wo have a contract for month to month likely to churn then to the people who are staying for a year/ 2 year.

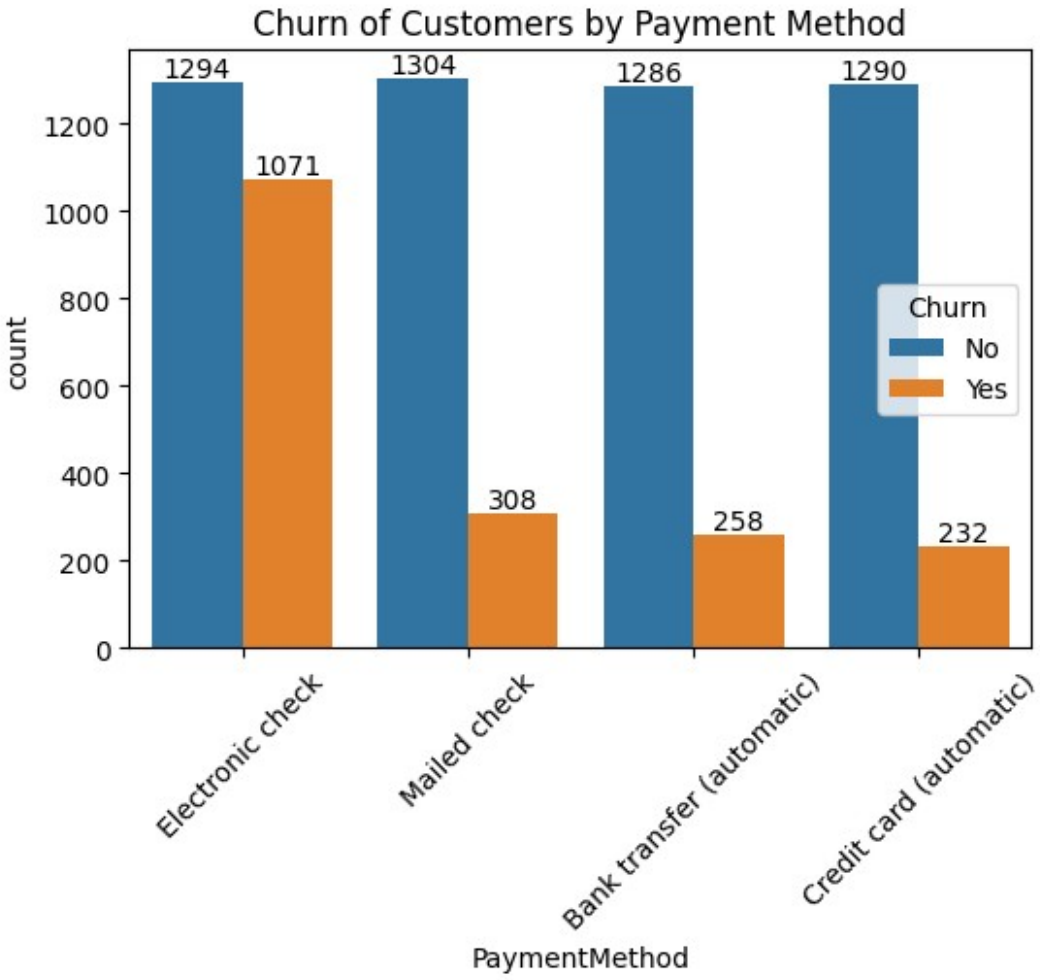
```
df.columns.values

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

# List of columns to plot
cols = ['PhoneService', 'MultipleLines', 'InternetService',
        'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
        'TechSupport', 'StreamingTV', 'StreamingMovies']

# Define subplot layout
n_cols = 3
n_rows = (len(cols) + n_cols - 1) // n_cols

# Create figure and axes
fig, axes = plt.subplots(n_rows, n_cols, figsize=(16, n_rows * 4))
axes = axes.flatten()  # Flatten in case of multiple rows

# Loop through each column and plot on its subplot
```

```
for i, col in enumerate(cols):
    sns.countplot(x=col, data=df, ax=axes[i], hue ="Churn")
    axes[i].set_title(f'Count of {col}')
    axes[i].tick_params(axis='x', rotation=45)

# Hide any unused axes
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

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




```
#here in one plot there are many subplots

#Customers without internet services or who opted "No" for add-on services like Online Security, Backup, and Tech Support showed higher churn rates.
Those using Fiber Optic Internet also showed a noticeably higher churn compared to DSL or no internet users.
Services such as StreamingTV and StreamingMovies showed mixed churn, but churn was generally higher where services were not active.
```

```
This suggests that lack of value-added services is strongly associated
with customer churn.

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 ("Churn of Customers by Payment Method")
plt.xticks(rotation = 45)
plt.show()
```




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