```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data = pd.read_csv("/content/Customer Churn.csv")
display(data)
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure
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

```
df=pd.DataFrame(data)
df
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure
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

```
df.isna().sum()
```

```
                  0
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.describe()
```

	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

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

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

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

Replacing TotalCharges to 0 (becauese They have Blank) and Converting it into Float Replacing Blanks with 0 as tenure is 0 and no total charges are recorded

```
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
```

```
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

Converted 0 and 1 values of SeniorCitizen to YES/NO to make it easier to understand

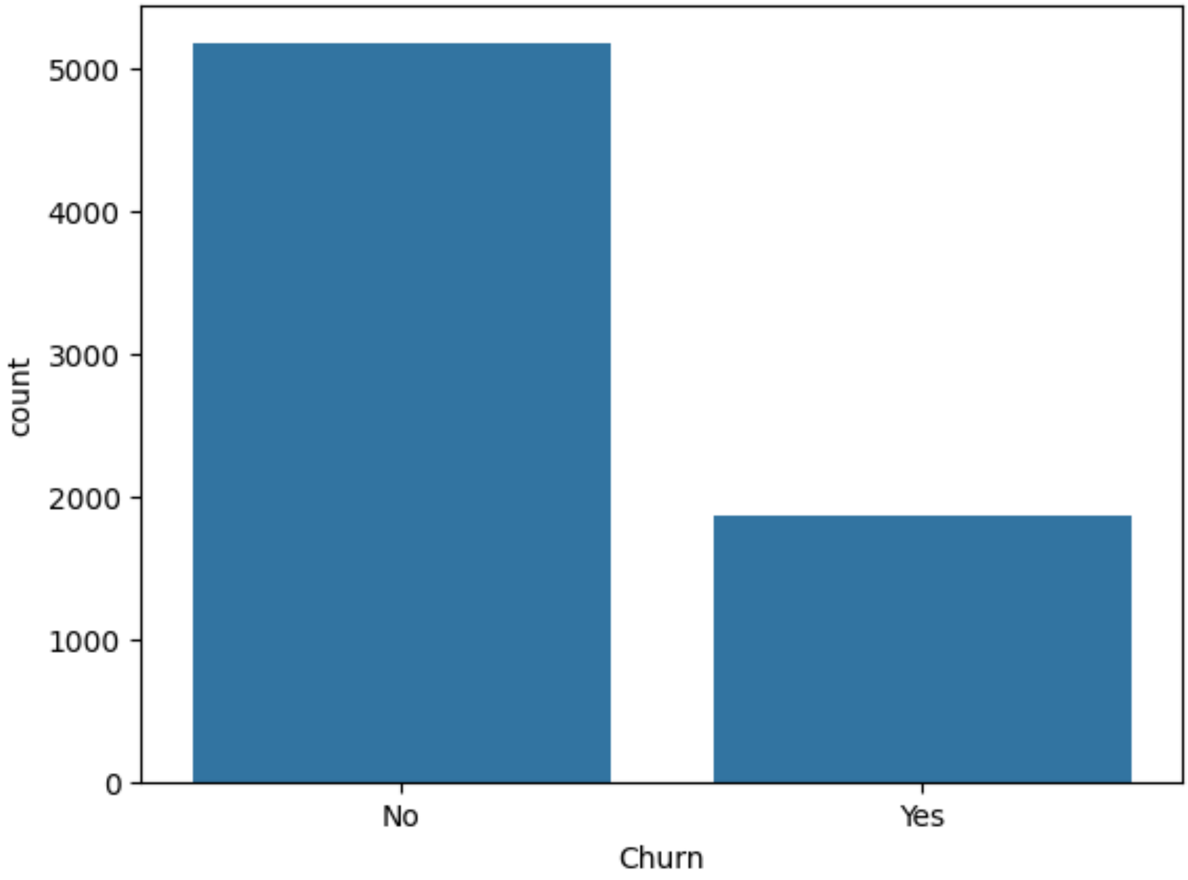
```
def convert(value):
  if value == 1:
    return "Yes"
  else:
    return "No"
df["SeniorCitizen"] = df["SeniorCitizen"].apply(convert)
df.head(25)
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	Ph
0	7590-VHVEG	Female	No	Yes	No	1	
1	5575-GNVDE	Male	No	No	No	34	
2	3668-QPYBK	Male	No	No	No	2	
3	7795-CFOCW	Male	No	No	No	45	
4	9237-HQITU	Female	No	No	No	2	
5	9305-CDSKC	Female	No	No	No	8	
6	1452-KIOVK	Male	No	No	Yes	22	
7	6713-OKOMC	Female	No	No	No	10	
8	7892-POOKP	Female	No	Yes	No	28	
9	6388-TABGU	Male	No	No	Yes	62	
10	9763-GRSKD	Male	No	Yes	Yes	13	
11	7469-LKBCI	Male	No	No	No	16	
12	8091-TTVAX	Male	No	Yes	No	58	
13	0280-XJGEX	Male	No	No	No	49	

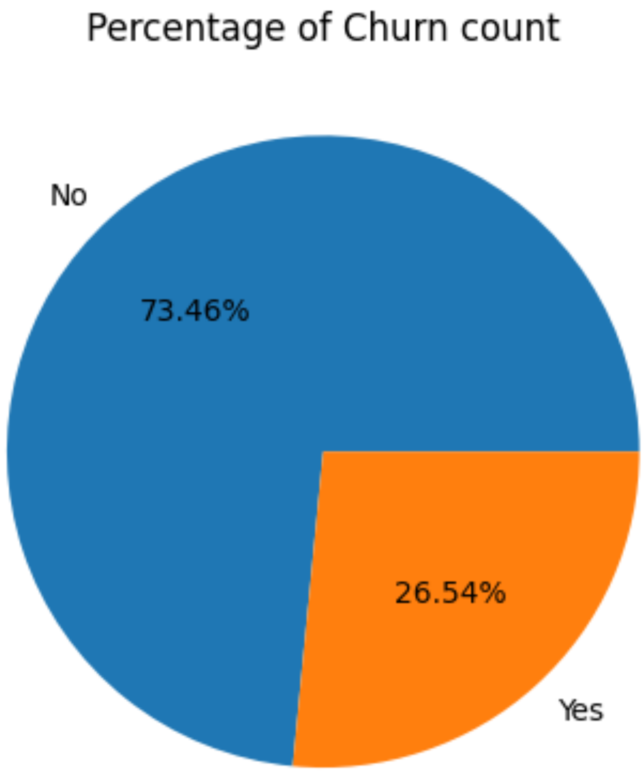
13	0280-XJGEX	Male	No	No	No	49
14	5129-JLPIS	Male	No	No	No	25
15	3655-SNQYZ	Female	No	Yes	Yes	69
16	8191-XWSZG	Female	No	No	No	52
17	9959-WOFKT	Male	No	No	Yes	71
18	4190-MFLUW	Female	No	Yes	Yes	10
19	4183-MYFRB	Female	No	No	No	21
20	8779-QRDMV	Male	No	No	No	1
21	1680-VDCWW	Male	No	Yes	No	12
22	1066-JKSGK	Male	No	No	No	1
23	3638-WEABW	Female	No	Yes	No	58
24	6322-HRPFA	Male	No	Yes	Yes	49

25 rows × 22 columns

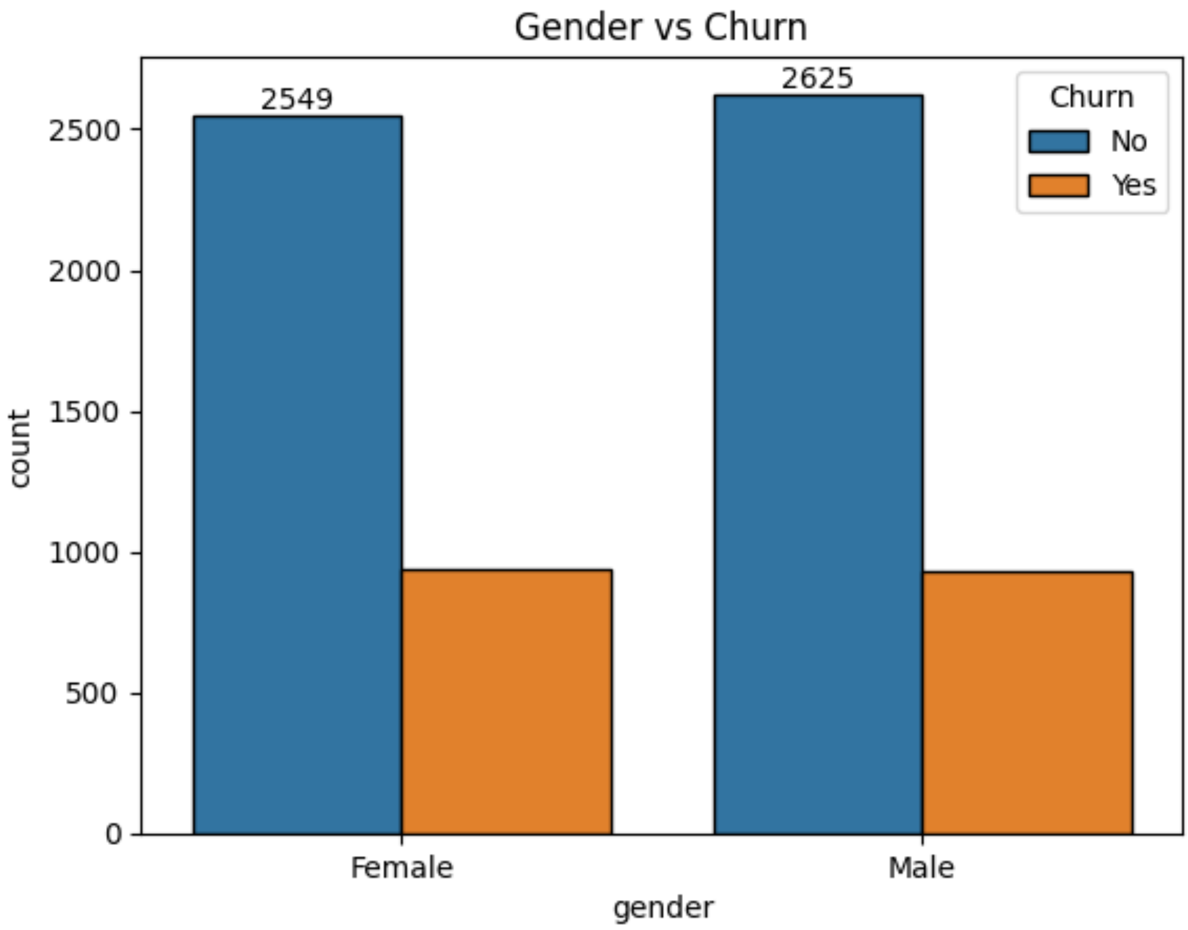
```
sns.countplot(x="Churn",data=df)
plt.show()
```



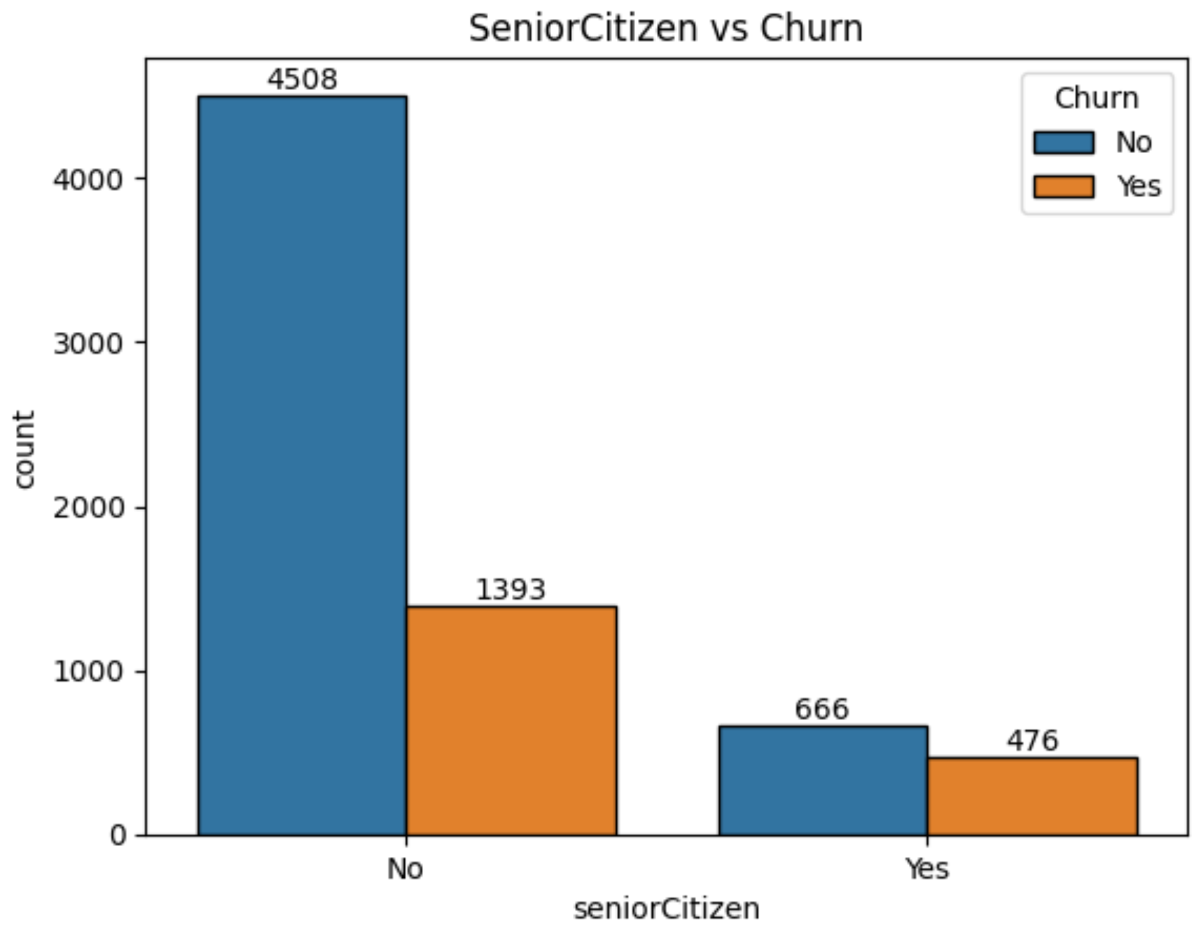
```
groupby=df.groupby('Churn').agg({"Churn":"count"})
plt.pie(groupby['Churn'],labels=groupby.index,autopct="%1.2f%%")
plt.title("Churn Percentage")
plt.show()
```



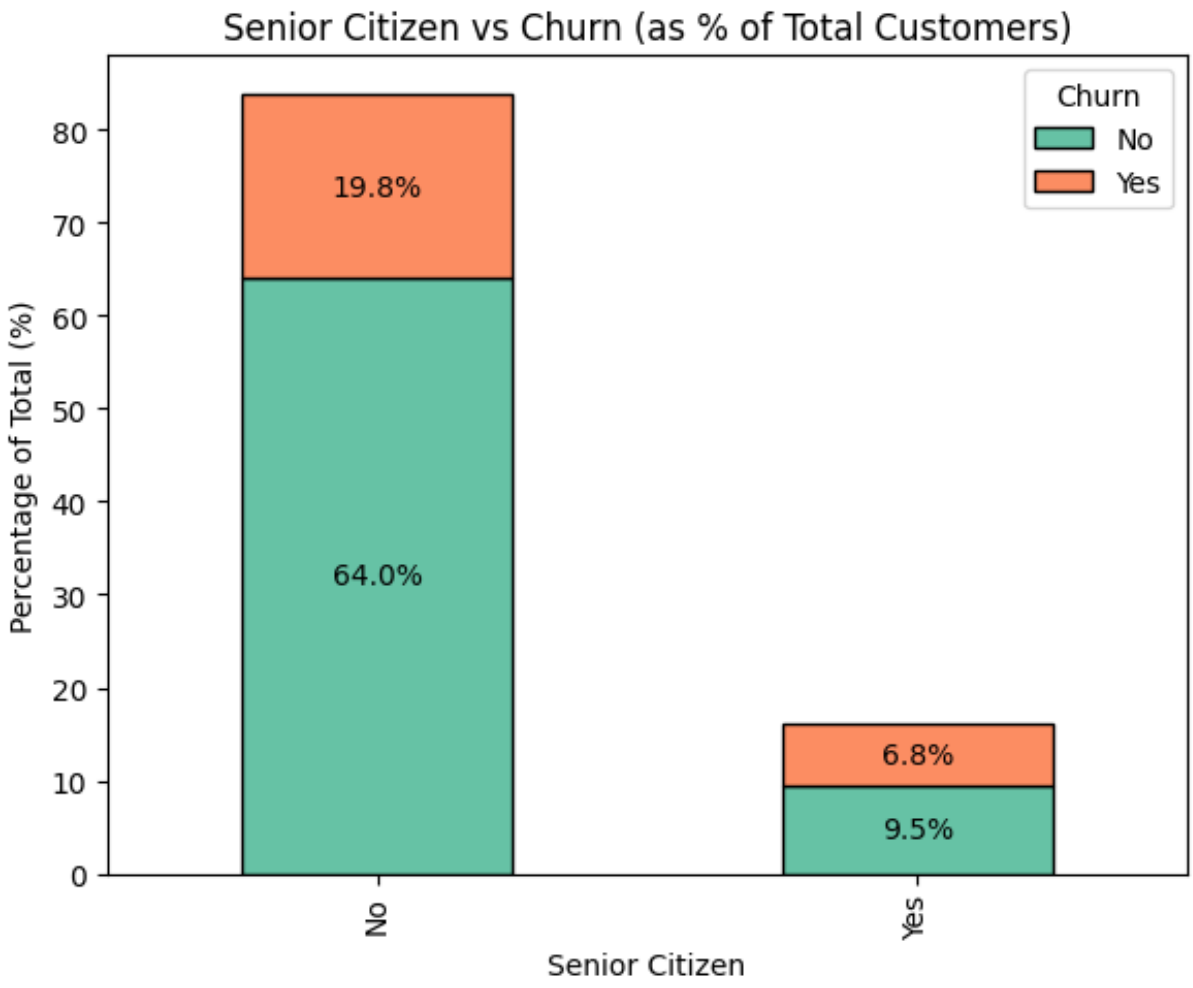
```
ax=sns.countplot(x='gender',data=df,hue='Churn',edgecolor='black')
ax.bar_label(ax.containers[0])
plt.title("Gender vs Churn")
plt.show()
```



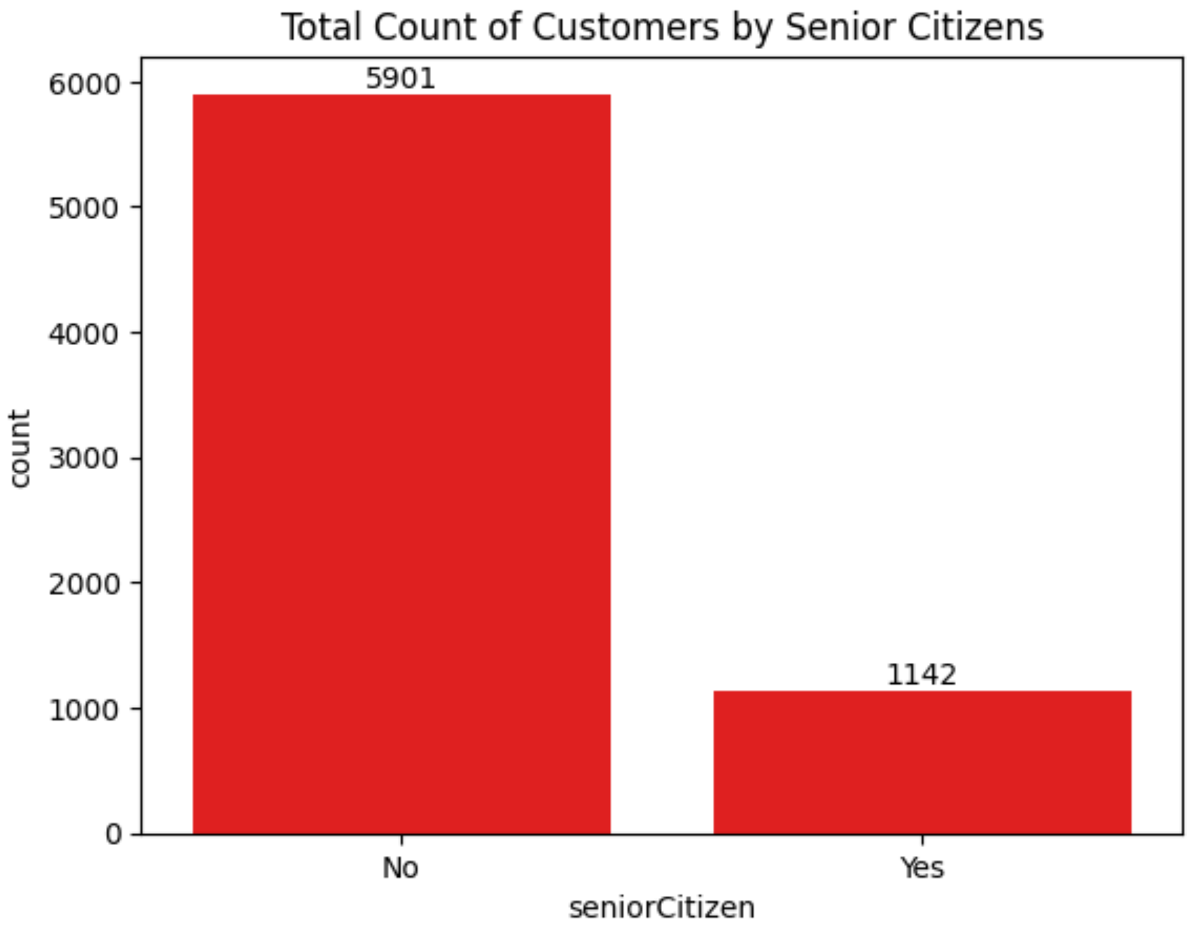
```
ax=sns.countplot(x='seniorCitizen',data=df,hue="Churn",edgecolor='b
ax.bar_label(ax.containers[0])
ax.bar_label(ax.containers[1])
plt.title('SeniorCitizen vs Churn')
plt.show()
```



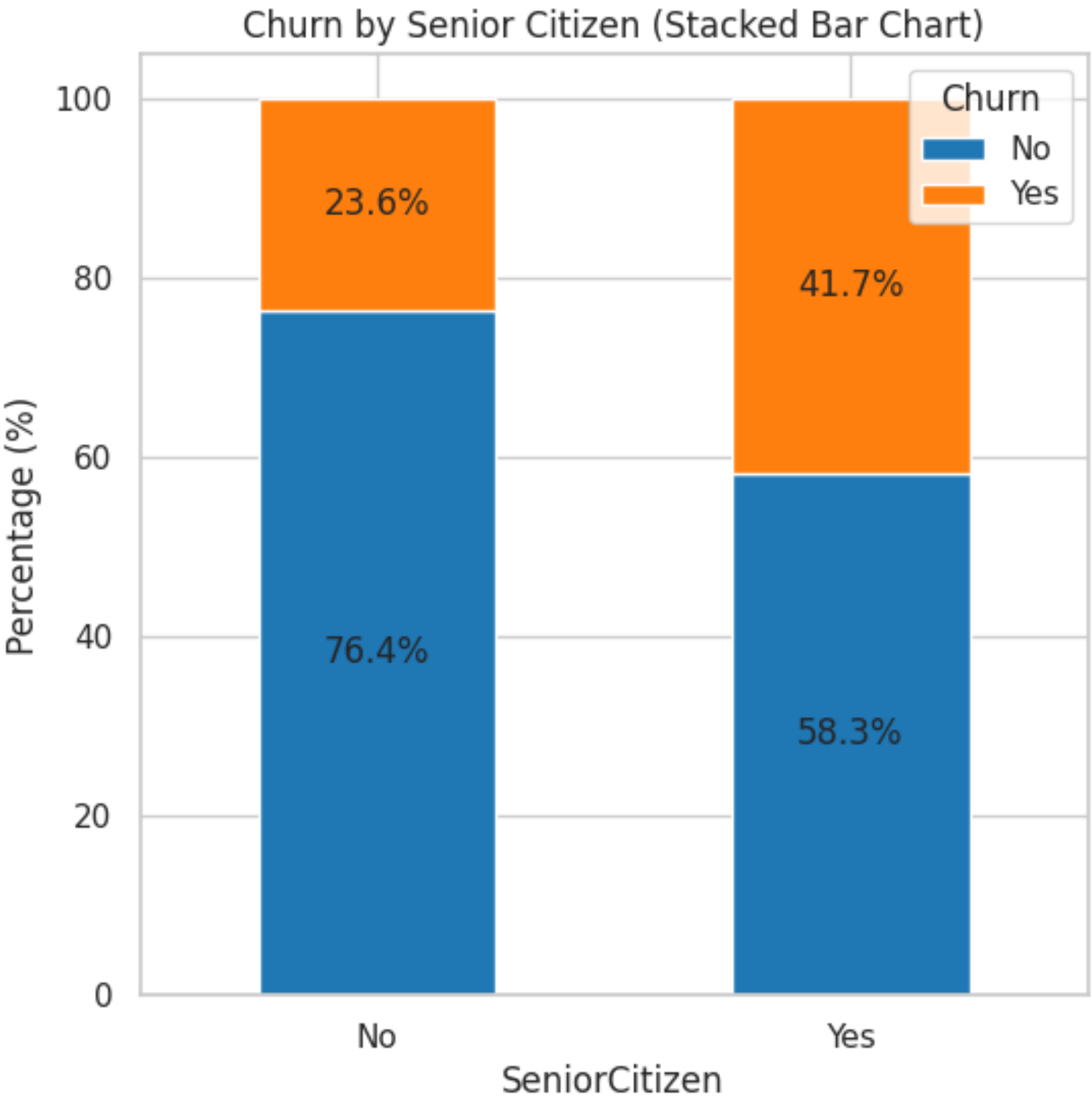
```
# Convert 'SeniorCitizen' back to numeric (0/1) if it was convertec
# df['SeniorCitizen'] = df['SeniorCitizen'].map({'Yes': 1, 'No': 0}

# Create cross-tab of seniorCitizen vs Churn
ct = pd.crosstab(df['seniorCitizen'], df['Churn'])

# Convert to percentages of TOTAL (not per row)
ct_percent_total = ct / ct.values.sum() * 100

# Plot stacked bar chart
ax = ct_percent_total.plot(
    kind='bar',
    stacked=True,
    figsize=(6,5),
    color=sns.color_palette('Set2', n_colors=len(ct.columns)),
    edgecolor='black'
)

# Add percentage labels inside bars
for container in ax.containers:
    ax.bar_label(container, fmt='%.1f%%', label_type='center', font

plt.title('Senior Citizen vs Churn (as % of Total Customers)')
plt.xlabel('Senior Citizen')
plt.ylabel('Percentage of Total (%)')
plt.legend(title='Churn')
plt.tight_layout()
plt.show()
```



```
ax=sns.countplot(x='seniorCitizen',data=df , color='red')
ax.bar_label(ax.containers[0])
plt.title("Total Count of Customers by Senior Citizens")
plt.show()
```

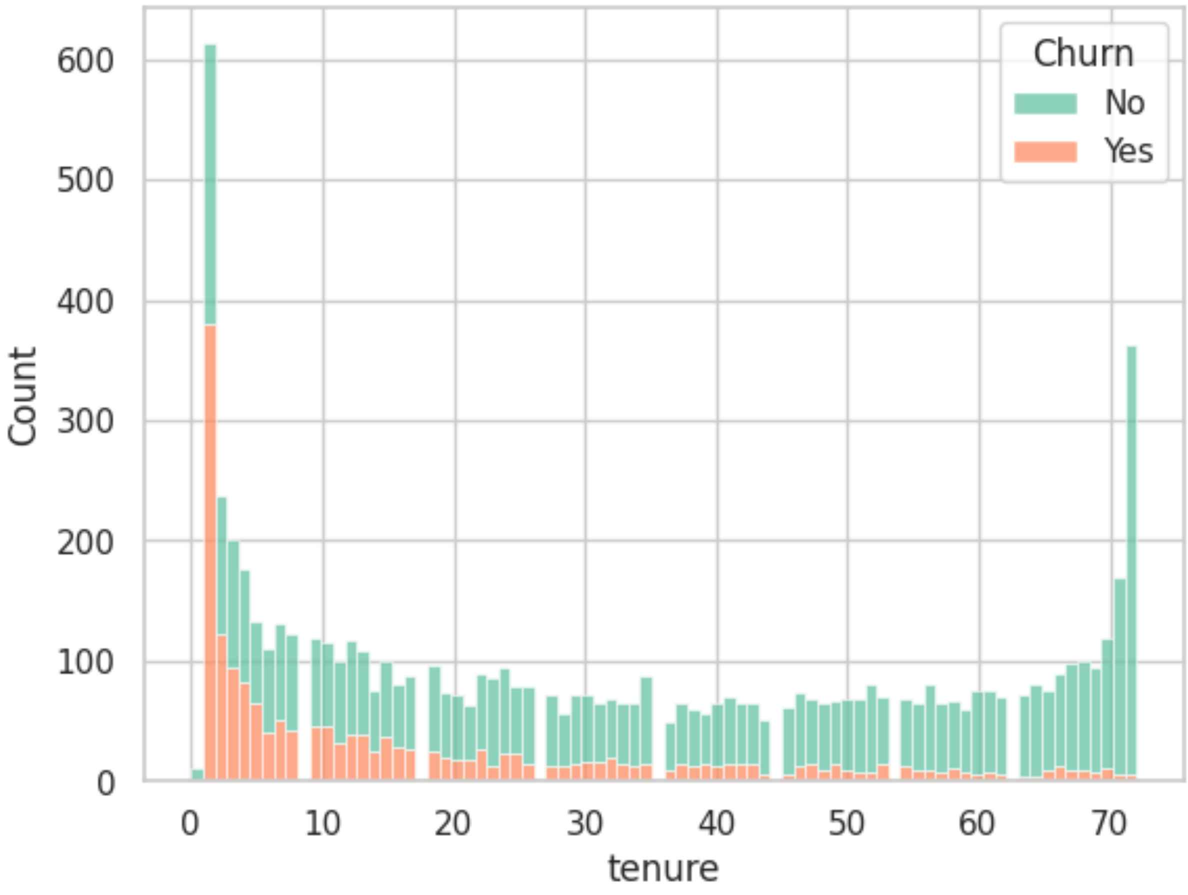


```
total_counts = df.groupby('seniorCitizen')['Churn'].value_counts(nc

# Plot
fig, ax = plt.subplots(figsize=(6, 6))  # Adjust figsize for better

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

# Add percentage labels on the bars
for container in ax.containers:
    ax.bar_label(container, fmt='%.1f%%', label_type='center')

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

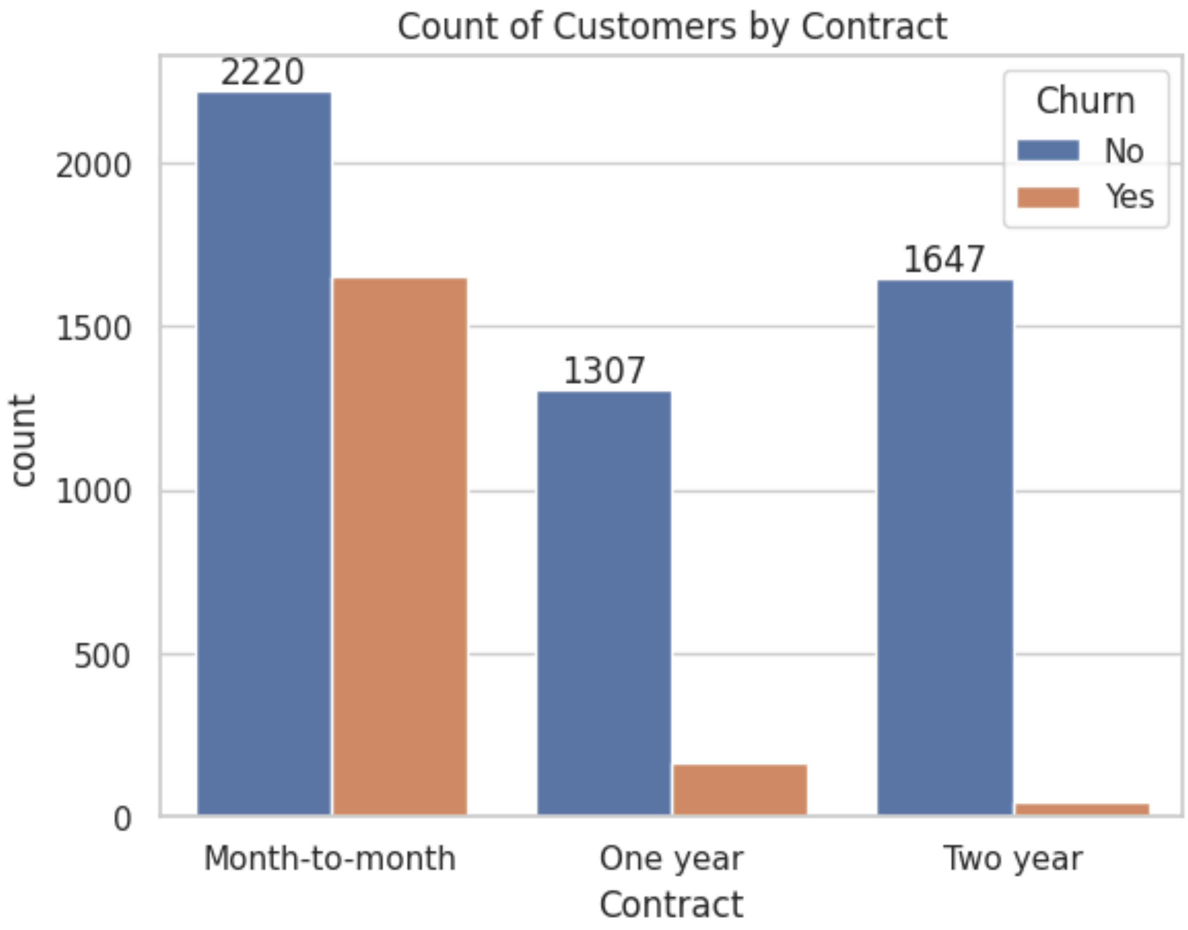


Comparative a grated percentage of people in senior citizen category have curned

```
sns.histplot(x='tenure',data=df, bins = 80 ,hue='Churn',multiple='s
plt.show()
```



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



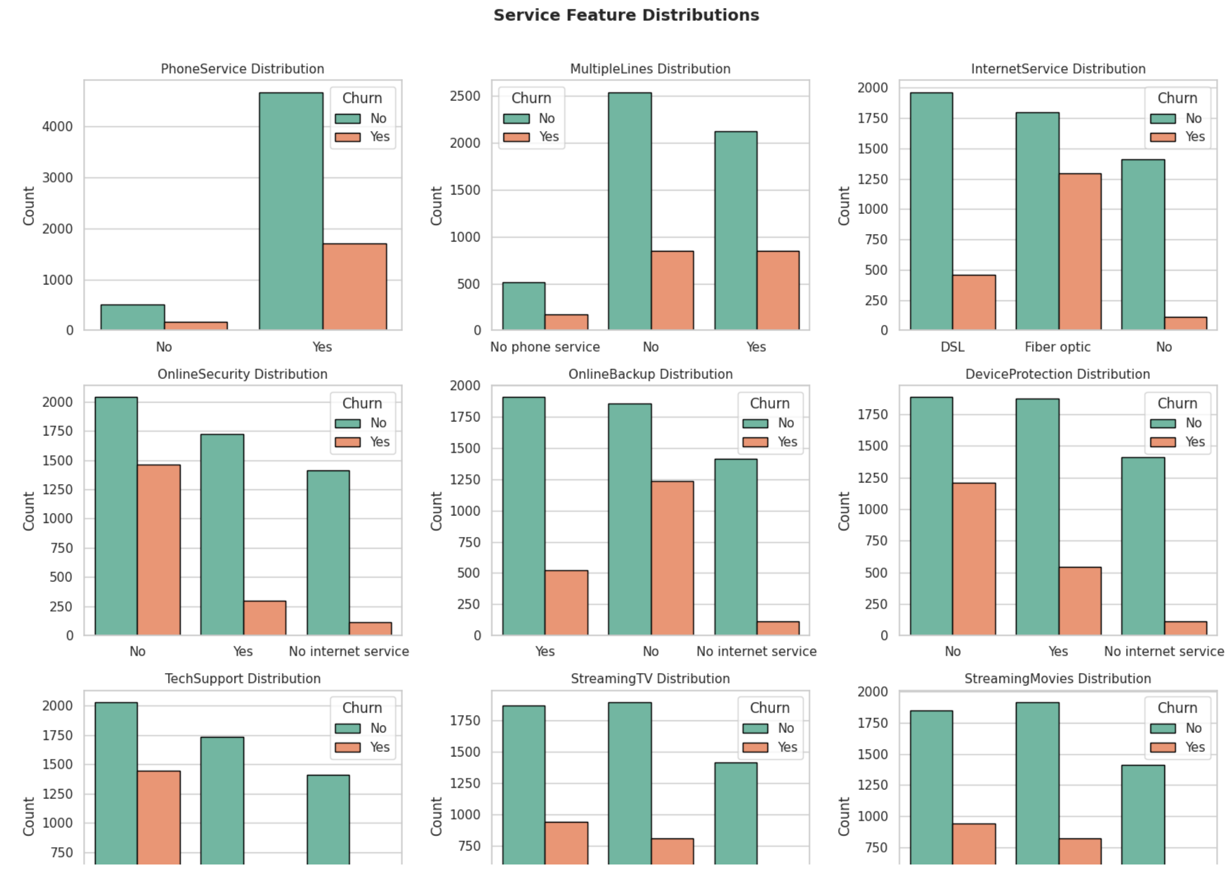
People who have used our services for a long time have stayed and people who have used our services 1 or 2 months have Churned

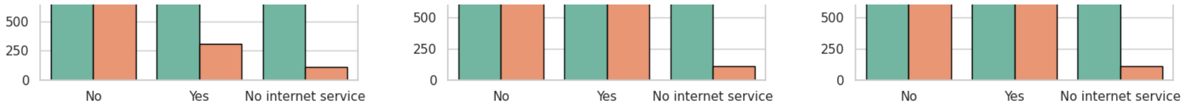
```
df.columns.values
```

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

```
# List of columns you want to plot
```

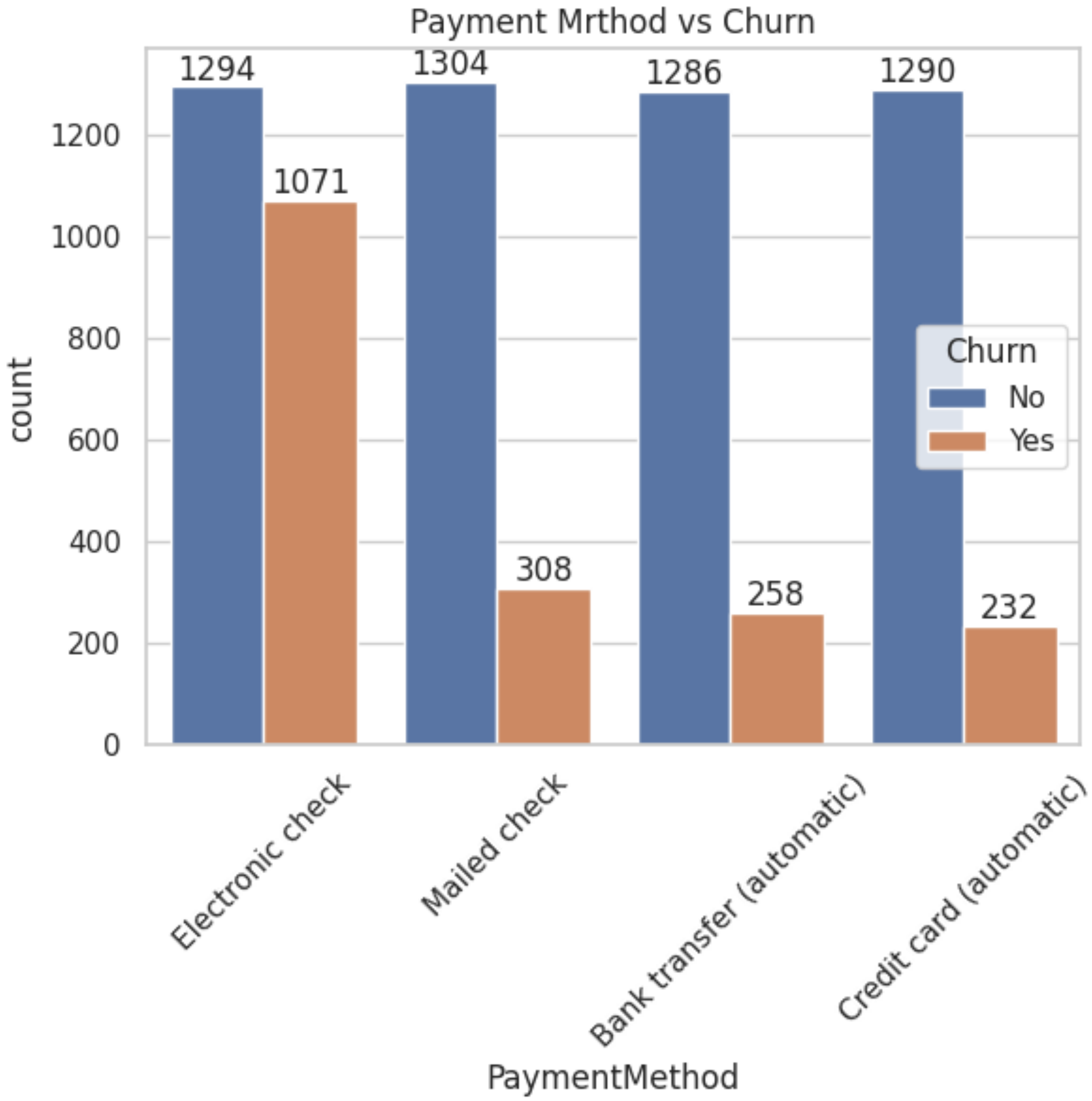
```
cols = [
    'PhoneService', 'MultipleLines', 'InternetService',
    'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
    'TechSupport', 'StreamingTV', 'StreamingMovies'
]

# Set up subplot grid (3 rows × 3 columns)
fig, axes = plt.subplots(3, 3, figsize=(15, 12))
axes = axes.flatten()  # Flatten 2D axes array for easy looping

# Loop through columns and plot each countplot
for i, col in enumerate(cols):
    sns.countplot(x=col, data=df, ax=axes[i],hue="Churn", palette='
    axes[i].set_title(f'{col} Distribution', fontsize=11)
    axes[i].set_xlabel('')
    axes[i].set_ylabel('Count')
    axes[i].tick_params(axis='x')  # Rotate x labels for readabilit

# Remove empty subplot if number of plots < grid size
for j in range(len(cols), len(axes)):
    fig.delaxes(axes[j])

plt.suptitle('Service Feature Distributions', fontsize=14, weight='
plt.tight_layout(rect=[0, 0, 1, 0.97])
plt.show()
```





The plots show that customers with basic or no additional services (like no OnlineSecurity, TechSupport, or DeviceProtection) tend to churn more frequently than those who use these services. Customers with Fiber optic Internet also exhibit a higher churn rate compared to DSL users. Overall, the presence of value-added services appears to reduce churn, indicating that service bundling helps retain customers.

```
ax=sns.countplot(x='PaymentMethod',data=df,hue='Churn')
ax.bar_label(ax.containers[0])
ax.bar_label(ax.containers[1])
plt.title("Payment Mrthod vs Churn")
plt.xticks(rotation=45)
plt.show()
```



Customer is likely to Churn when he is using electronic check as payment method