

```
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()
```

	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-CF0CW	Male	0	No	No	45
4	9237-HQITU	Female	0	No	No	2

	MultipleLines	InternetService	OnlineSecurity	...
0	No phone service	DSL	No	...
1	No	DSL	Yes	...
2	No	DSL	Yes	...
3	No phone service	DSL	Yes	...
4	No	Fiber optic	No	...

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

	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]

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 blanks with 0 as tenure is 0 and no 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

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

0

```
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

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

0

```

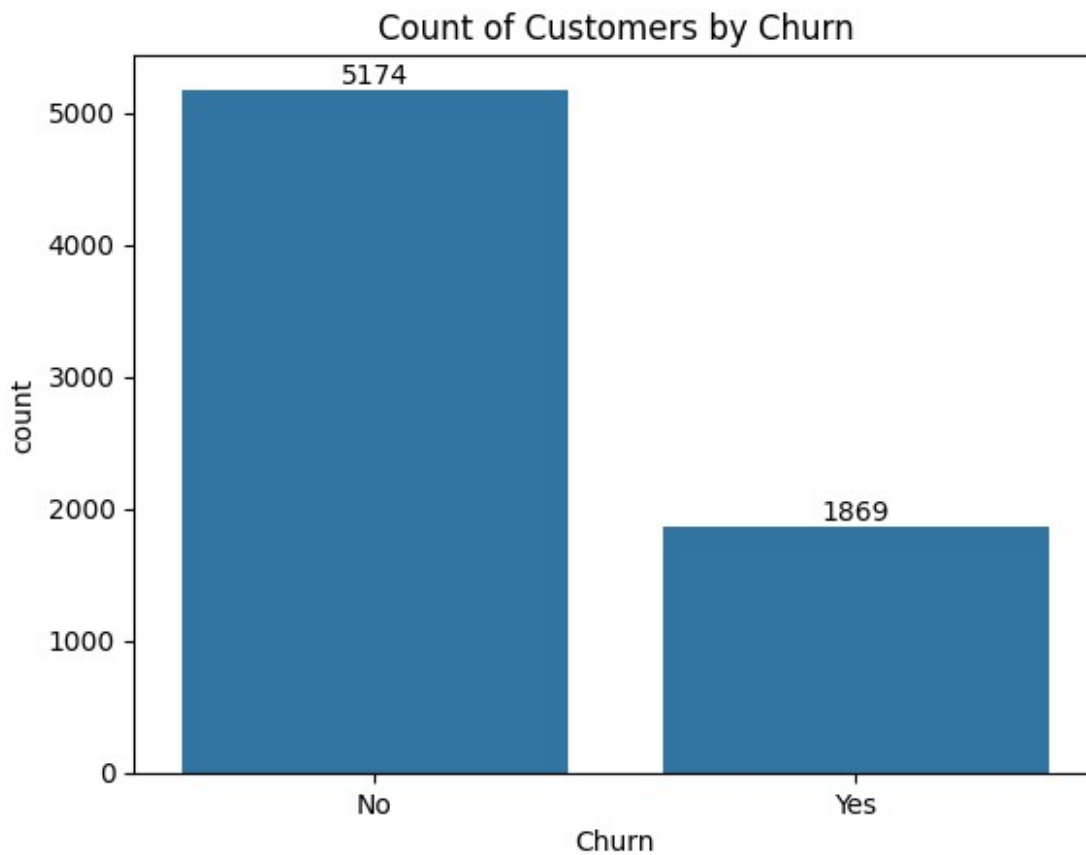
def conv(value):
    if value == 1:
        return "yes"
    else:
        return "no"

```

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

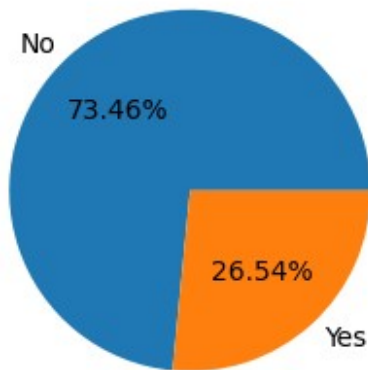
#converted 0 and 1 values of senior citizen to yes/no to make it easier to understand

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



```
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()
```

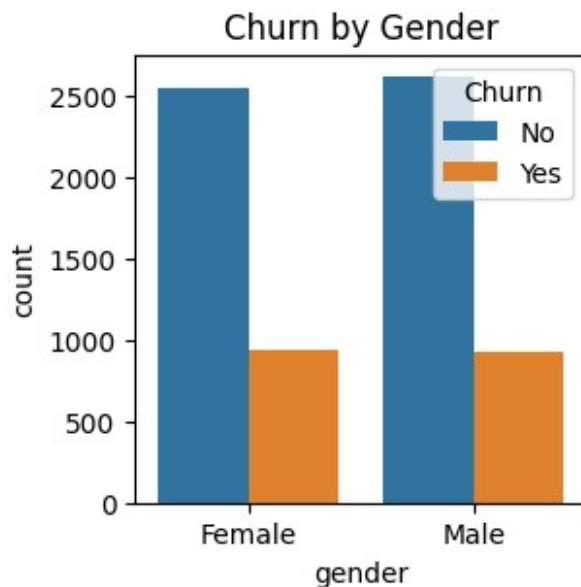
Percentage of Churned Customeres



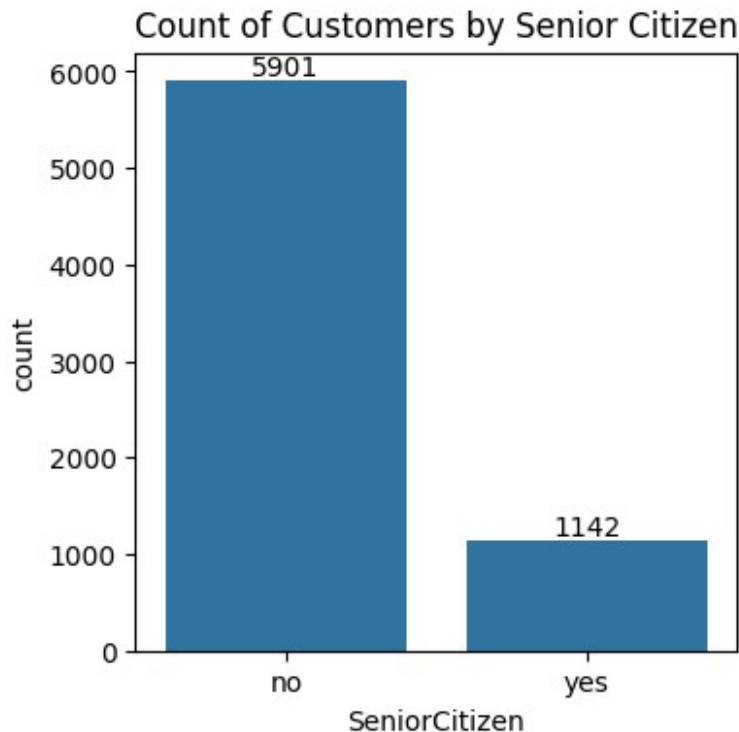
#from the given pie chart we can conclude that 26.54% of our customers have churned out.

#not let's explore the reason behind it

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



```
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()
```



```
total_counts = df.groupby('SeniorCitizen')
['Churn'].value_counts(normalize=True).unstack() * 100

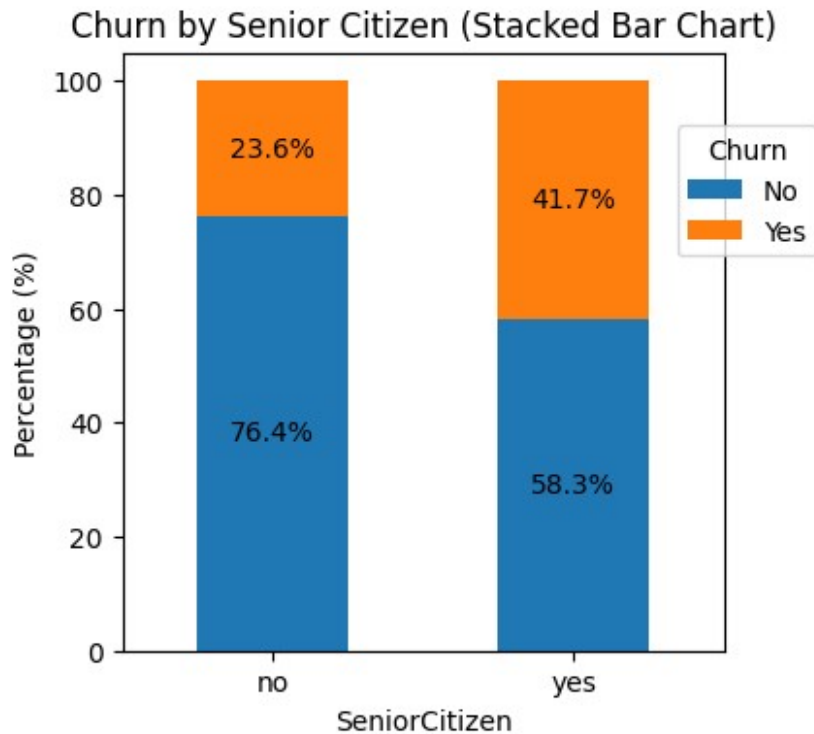
# Plot
fig, ax = plt.subplots(figsize=(4, 4)) # Adjust figsize for better
visualization

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

# 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='center')

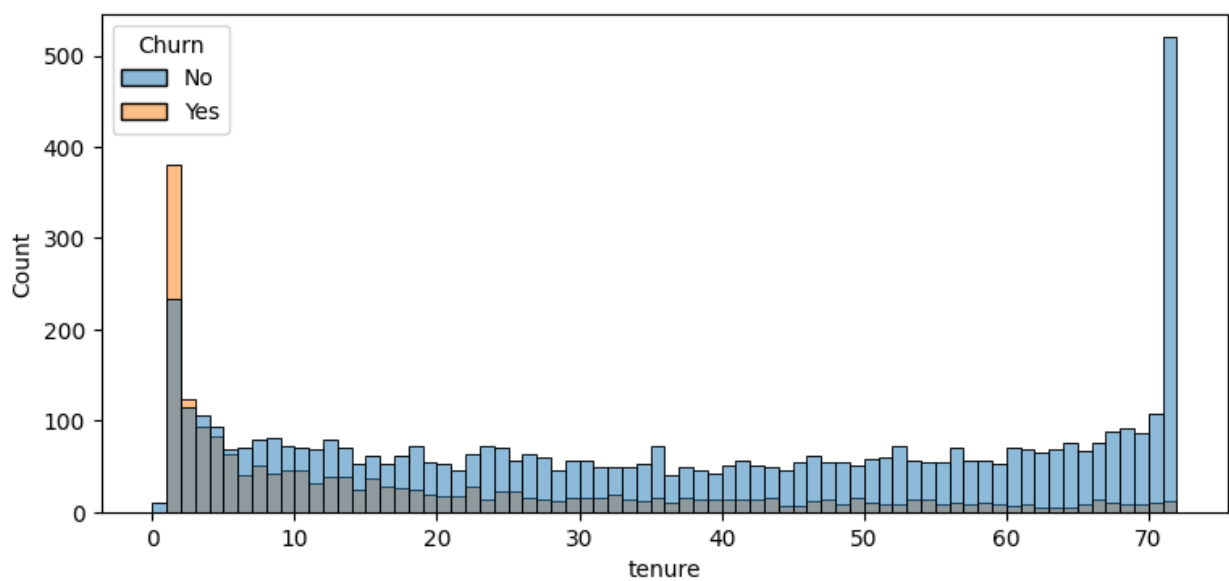
plt.title('Churn by Senior Citizen (Stacked Bar Chart)')
plt.xlabel('SeniorCitizen')
plt.ylabel('Percentage (%)')
plt.xticks(rotation=0)
plt.legend(title='Churn', bbox_to_anchor = (0.9,0.9)) # Customize
legend location

plt.show()
```



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

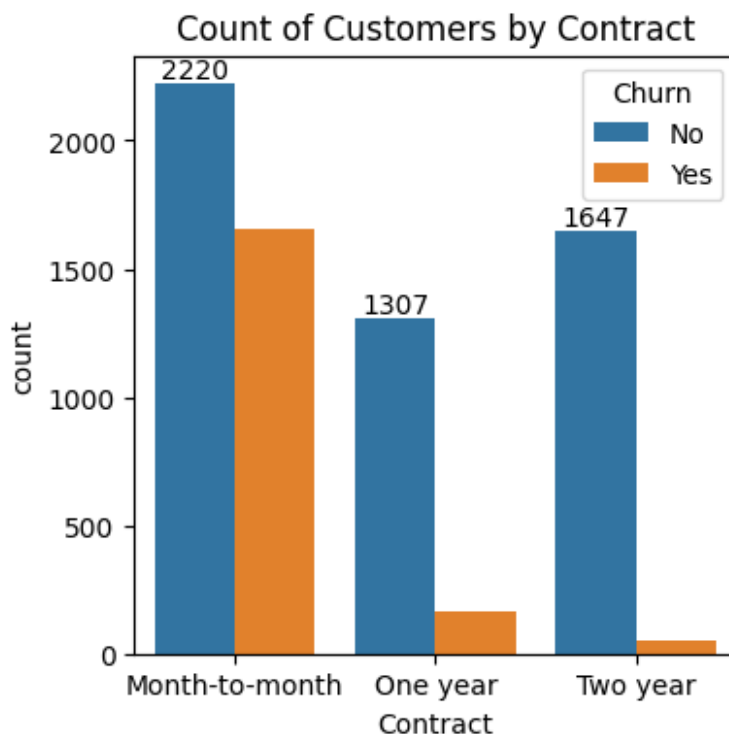
```
plt.figure(figsize = (9,4))
sns.histplot(x = "tenure", data = df, bins = 72, hue = "Churn")
plt.show()
```



#people who have used our services for a long time have stayed and people who have used our sevices

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

```
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'], dtype=object)
```

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

*# 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 needed

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

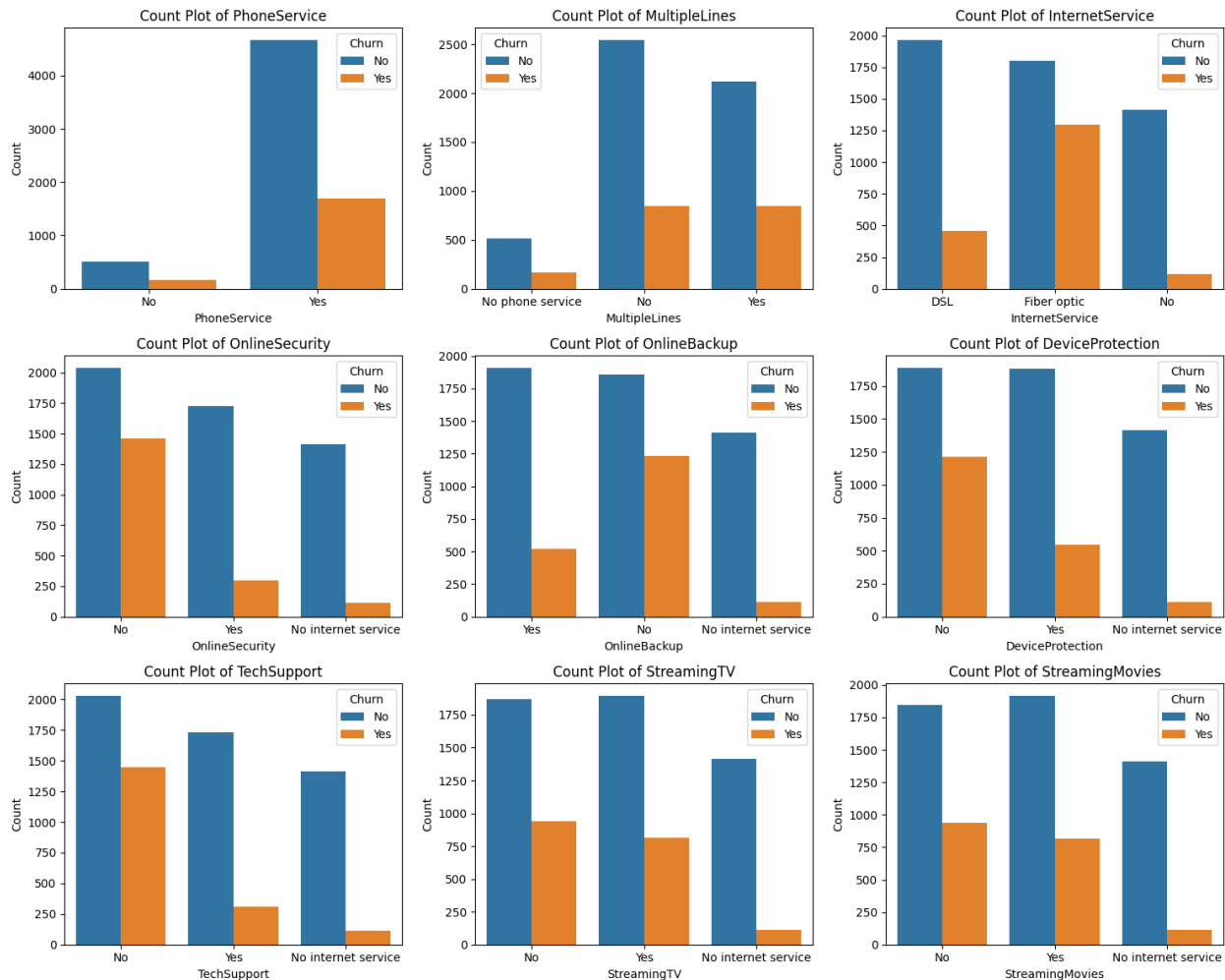
# Flatten the axes array for easy iteration (handles both 1D and 2D
arrays)
axes = axes.flatten()

# 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])

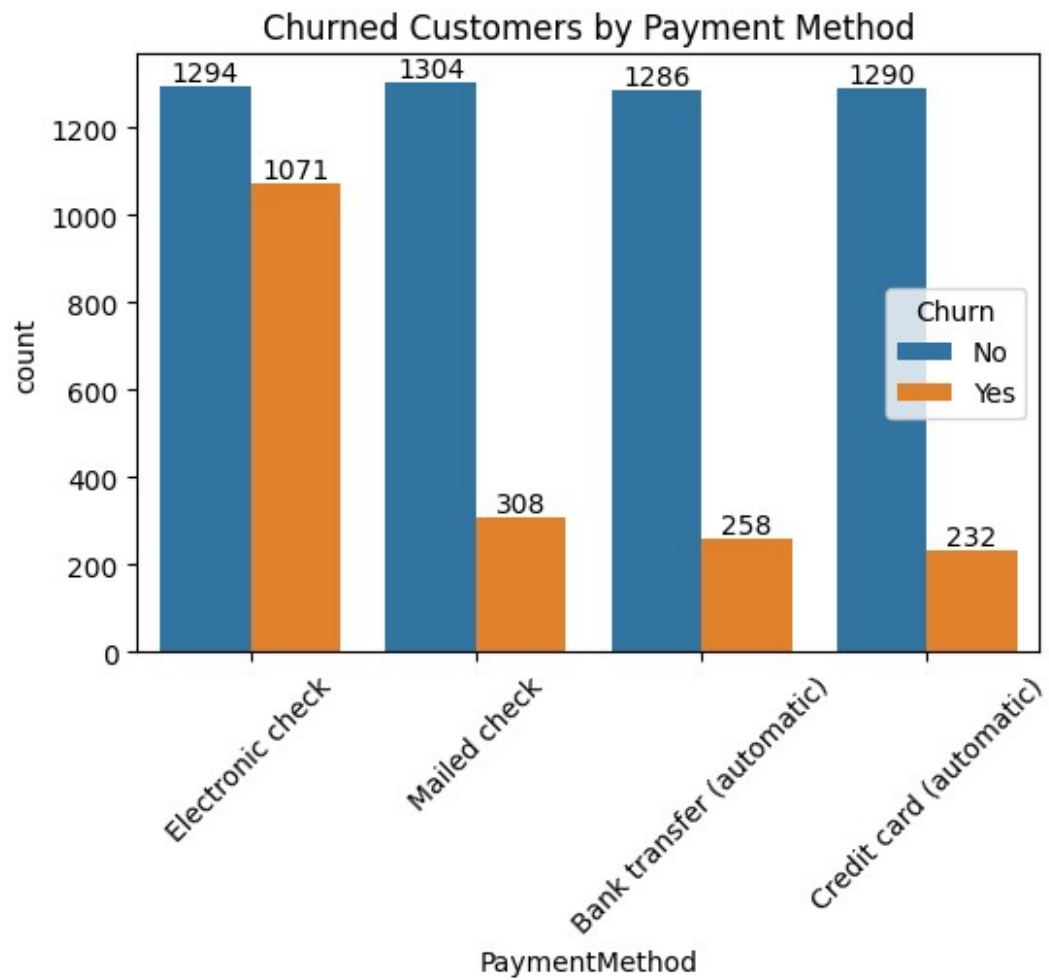
plt.tight_layout()
plt.show()

```



#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.

```
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.