```
!pip install numpy
!pip install pandas
!pip install matplotlib
!pip install seaborn
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.0.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: numpy>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.59.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.12/dist-packages (from seaborn) (2.0.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.12/dist-packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.12/dist-packages (from seaborn) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.5
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.2.
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->
```

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

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

Choose Files Customer Churn.csv

- **Customer Churn.csv**(text/csv) - 977501 bytes, last modified: 8/18/2025 - 100% done

```
Saving Customer Churn.csv to Customer Churn.csv
```

```
df = pd.read_csv('Customer Churn.csv')
```

```
df.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	Dev
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	

5 rows × 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
```

```
df['TotalCharges'] = df['TotalCharges'].replace(' ','0').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() # null values in column wise#
```

	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.isnull().sum().sum() ##total data null values ##
```

```
np.int64(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.duplicated().sum() ## whole data duplicate values ##
```

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

```
df['customerID'].duplicated().sum() ## based on unique data ##
```

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

```
def conv(value):
  if value == 1:
    return 'yes'
  else:
```

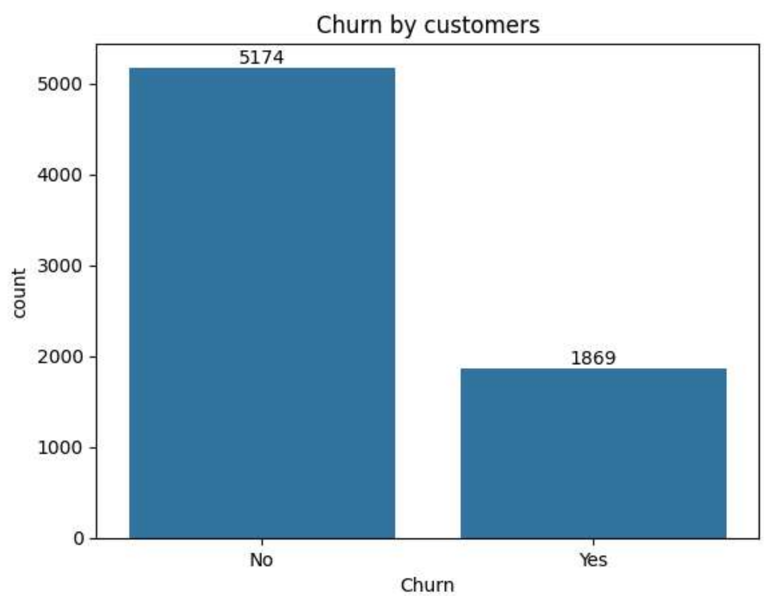
```
    return 'no'
df['SeniorCitizen'] = df['SeniorCitizen'].apply(conv)
```

```
df.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	Dev
0	7590-VHVEG	Female	no	Yes	No	1	No	No phone service	DSL	No	...	
1	5575-GNVDE	Male	no	No	No	34	Yes	No	DSL	Yes	...	
2	3668-QPYBK	Male	no	No	No	2	Yes	No	DSL	Yes	...	
3	7795-CFOCW	Male	no	No	No	45	No	No phone service	DSL	Yes	...	
4	9237-HQITU	Female	no	No	No	2	Yes	No	Fiber optic	No	...	

5 rows × 21 columns

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



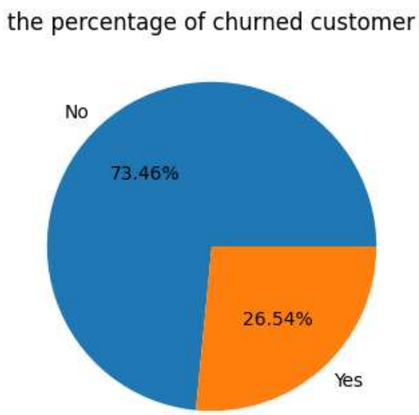

```
gb = df.groupby('Churn').agg({'Churn':'count'})
gb
##plt.pie(df['Churn'])
##plt.show()
```

	Churn
Churn	
No	5174
Yes	1869

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

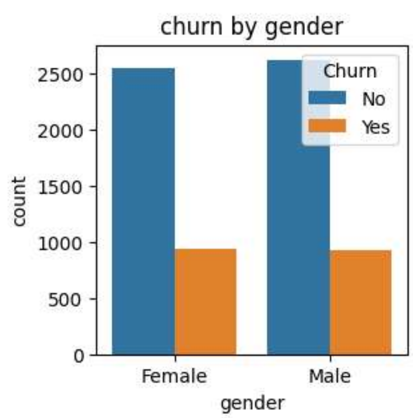
	SeniorCitizen
SeniorCitizen	
no	5901
yes	1142

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



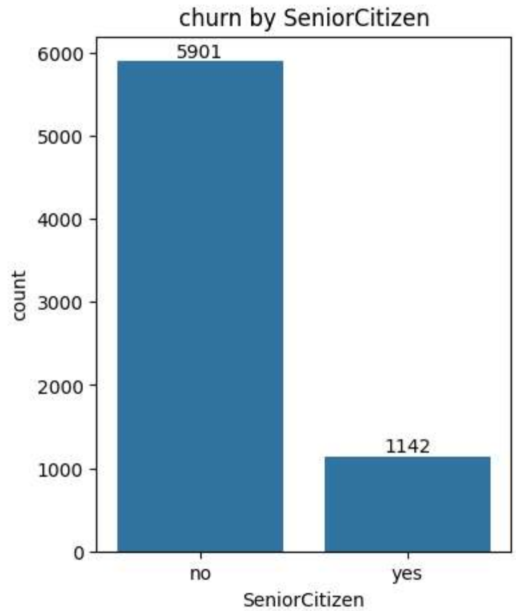
Here we have seen that 26.5% of the customers churn out their services.

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



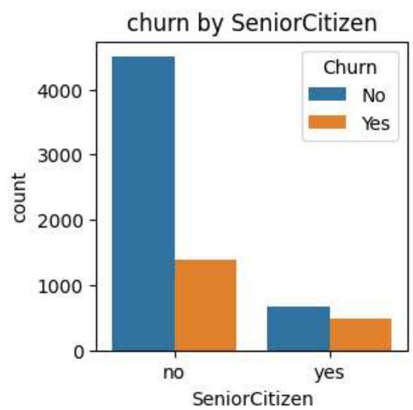
We have observed that the percentage of male and female customers here is almost equal.

```
plt.figure(figsize=(4,5))
ax= sns.countplot(x='SeniorCitizen',data=df)
ax.bar_label(ax.containers[0])
plt.title('churn by SeniorCitizen')
plt.show()
```



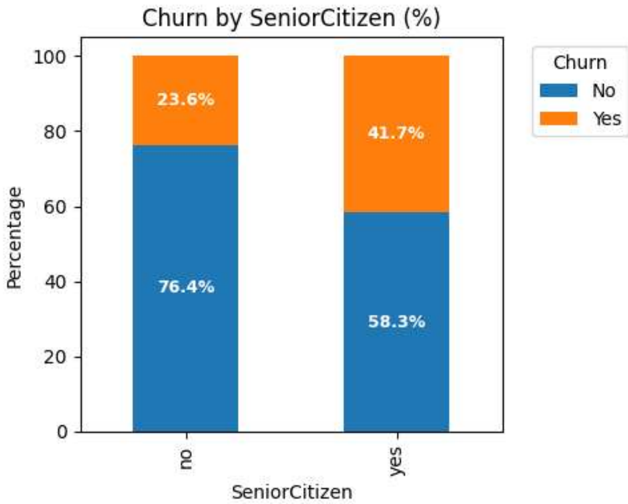
We found that senior citizen customers are less in number among total customers in this analysis.

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

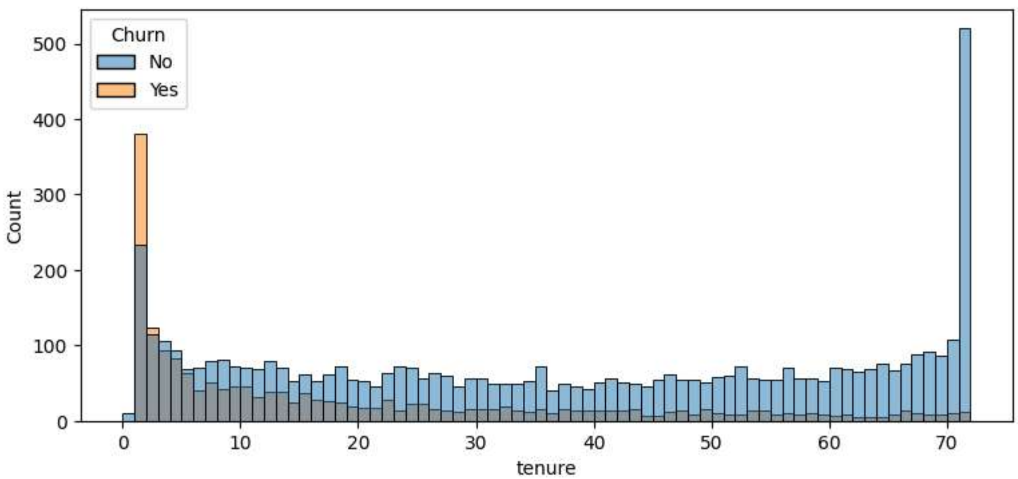


Here we have seen that senior citizens have got more churn out than other customers.

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

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

plt.title('Churn by SeniorCitizen (%)')
plt.xlabel('SeniorCitizen')
plt.ylabel('Percentage')
plt.legend(title='Churn', bbox_to_anchor=(1.05, 1), loc='upper left')

# Add percentage labels
for c in ax.containers:
    ax.bar_label(c, fmt='%.1f%%', label_type='center', fontsize=9, color="white", weight="bold")

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

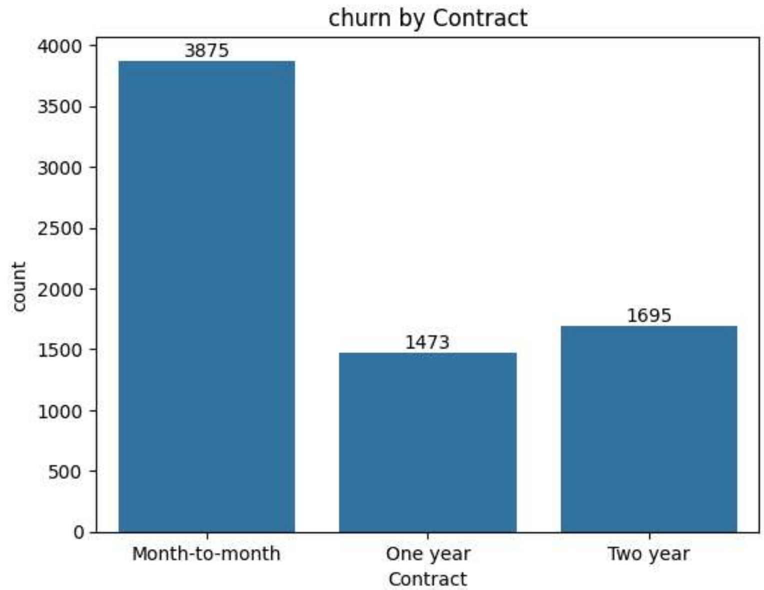



```
plt.figure(figsize =(9,4))
sns.histplot(x = 'tenure', data = df, bins =72, hue = 'Churn')
plt.show()
```

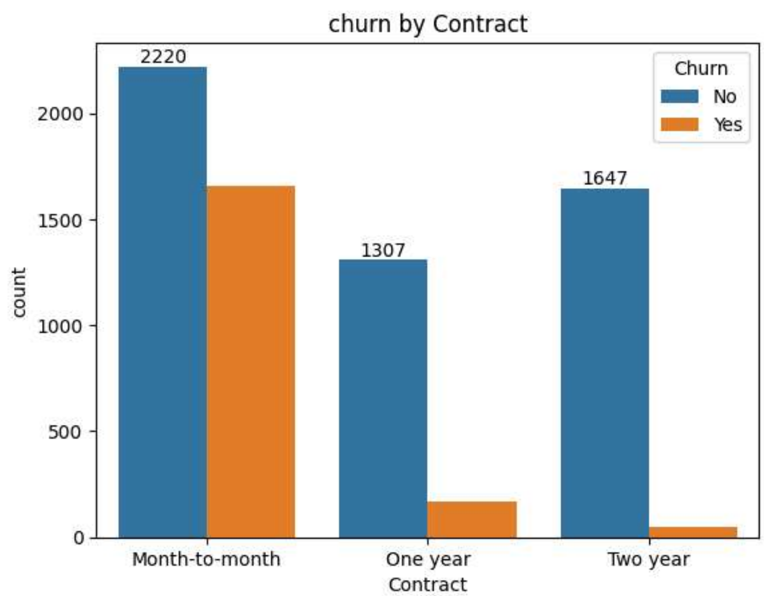


People who have used our services for a long time have stayed and people who have used our services for 2 to 3 month have churned.

```
ay= sns.countplot(x='Contract',data=df)
ay.bar_label(ay.containers[0])
plt.title('churn by Contract')
plt.show()
```




```
ay= sns.countplot(x='Contract',data=df, hue='Churn')
ay.bar_label(ay.containers[0])
plt.title('churn by Contract')
plt.show()
```



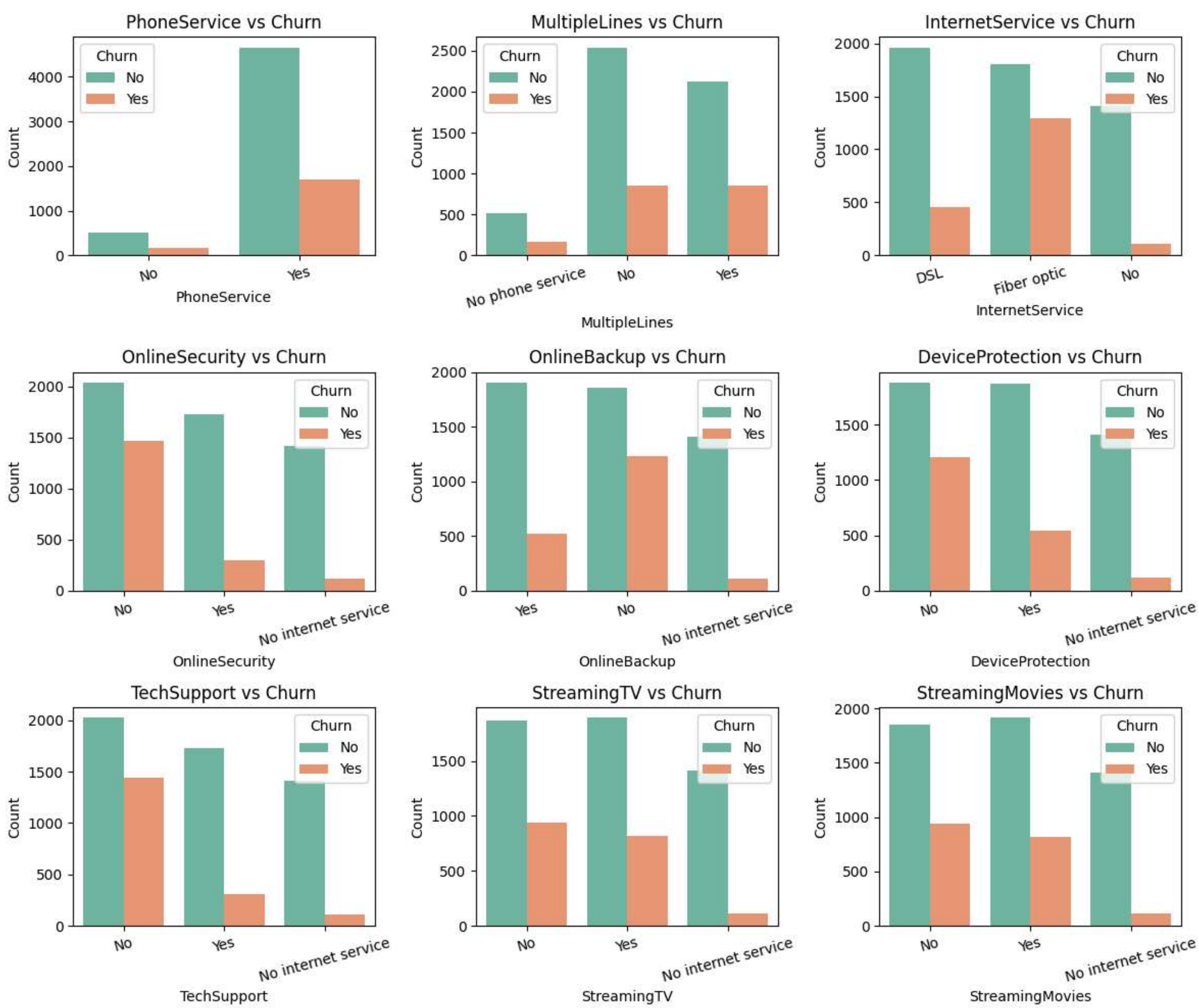
People who have month to month contract are likely to Churn then from those who have 1 or 2 year contract.

```
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)
```

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

# Set up subplot grid
n_cols = 3  # number of plots in each row
n_rows = (len(cols) + n_cols - 1) // n_cols  # auto calculate rows
```

```
plt.figure(figsize=(12, 10))

for i, col in enumerate(cols, 1):
    plt.subplot(n_rows, n_cols, i)
    sns.countplot(x=df[col], hue=df['Churn'], palette="Set2")
    plt.title(f'{col} vs Churn')
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.xticks(rotation=15)

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



📊 Summary of Service Features vs Churn

PhoneService -

Almost all customers have phone service.

Churn rate does not differ much based on phone service.

MultipleLines -

Customers with no phone service have relatively lower churn.

Customers with multiple lines do not show a big difference in churn compared to single line users.

InternetService -

Fiber optic users churn more compared to DSL users.

Customers with no internet service churn much less (logical, since they subscribe to fewer services).

OnlineSecurity -

Customers with no online security have higher churn.

Having online security appears to reduce churn risk.

OnlineBackup -

Customers without backup service churn more.

Online backup slightly helps retention, but not as strongly as security/tech support.

DeviceProtection -

Similar to backup – no device protection → higher churn.

TechSupport -

Very strong indicator.

Customers with no tech support have a much higher churn rate.

Customers with tech support churn significantly less.

StreamingTV & StreamingMovies -

Streaming services (TV, movies) do not have a strong relationship with churn.

Churn rates look almost equal regardless of streaming add-ons.

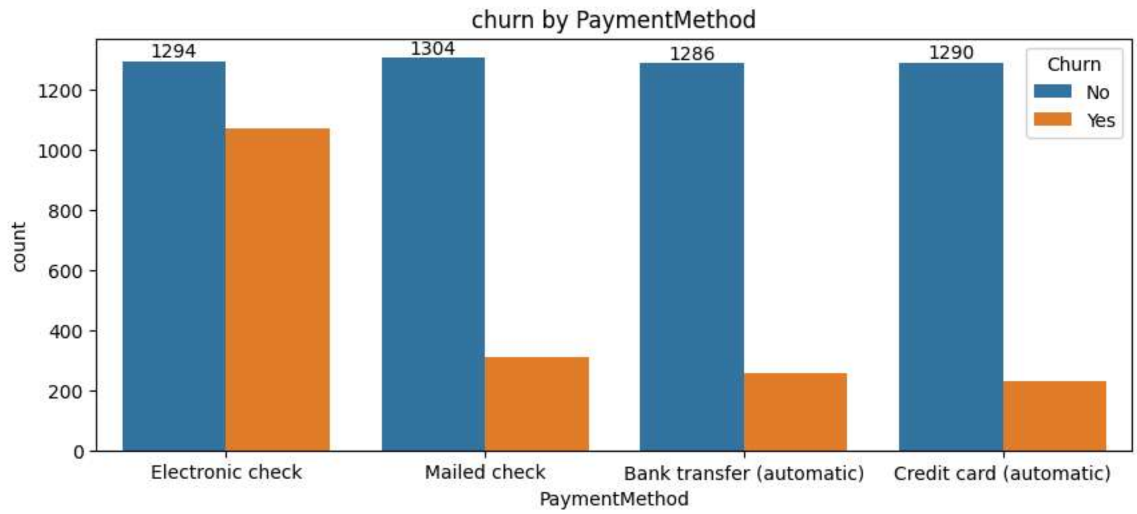
📝 Key Takeaways -

High churn risk groups: Fiber optic users, customers without Online Security, Backup, Device Protection, and Tech Support.

Retention drivers: Tech Support and Online Security are the strongest factors reducing churn.

Low impact features: Phone service, Multiple lines, Streaming TV, and Streaming Movies do not strongly influence churn.

```
plt.figure(figsize =(10,4))
ay= sns.countplot(x='PaymentMethod',data=df, hue='Churn')
ay.bar_label(ay.containers[0])
plt.title('churn by PaymentMethod')
plt.show()
```



Customer are very less likely to churn when they were used automatic payment method

Start coding or generate with AI.

```
#                  **Customer Churn Analysis**

This project delivers a detailed exploratory data analysis of customer
behavior in a subscription-based service, highlighting patterns across
demographics, tenure, and contract types.
• Overall Churn Rate: 26.5% of customers discontinued services, indica
nearly 1 in 4 customers leave, a critical metric for retention strateg

**• Demographic Trends:**

Gender distribution is balanced (50.3% male vs. 49.7% female), showing
churn difference by gender.
o Senior citizens account for a smaller share of the customer base (~1
show higher churn rates (40%+) compared to younger customers.

**• Tenure & Loyalty:**

Customers with less than 3 months of service are at the highest risk o
(~45%), suggesting early-stage dissatisfaction.
o In contrast, customers with >2 years tenure show churn rates below 1
demonstrating strong loyalty.

**• Contract Insights:**

oMonth-to-month subscribers form the largest group (~55% of customers)
churn the most (45% churn rate).
o Customers with 1-year contracts churn at ~12%, while 2-year contract
churn at only ~3%, proving the effectiveness of long-term commitments.

**Conclusion:**
```

Customer Churn Analysis

This project delivers a detailed exploratory data analysis of customer churn behavior in a subscription-based service, highlighting patterns across demographics, tenure, and contract types. • Overall Churn Rate: 26.5% of customers discontinued services, indicating that nearly 1 in 4 customers leave, a critical metric for retention strategy.

• Demographic Trends:

Gender distribution is balanced (50.3% male vs. 49.7% female), showing no major churn difference by gender. o Senior citizens account for a smaller share of the customer base (~16%) but show higher churn rates (40%+) compared to younger customers.

• Tenure & Loyalty:

Customers with less than 3 months of service are at the highest risk of churn (~45%), suggesting early-stage dissatisfaction. o In contrast, customers with >2 years tenure show churn rates below 10%, demonstrating strong loyalty.

• Contract Insights:

oMonth-to-month subscribers form the largest group (~55% of customers) and churn the most (45% churn rate). o Customers with 1-year contracts churn at ~12%, while 2-year contract holders churn at only ~3%,