In [4]:

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

In [5]:

```
df = pd.read_csv(r"C:\Users\Om Satyawan Pathak\OneDrive\Desktop\churn analysis\Customer Churn.csv")
```

In [6]:

```
df
```

Out[6]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSe
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	
...	...	...	...	...	...	...	...	...	...	...
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	

7043 rows × 21 columns

In [7]:

```
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

In [8]:

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

In [9]:

```
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
```

In [10]:

```
df.isnull()
```

Out[10]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSe
0	False	False	False	False	False	False	False	False	False	

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSe
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	
7038	False	False	False	False	False	False	False	False	False	
7039	False	False	False	False	False	False	False	False	False	
7040	False	False	False	False	False	False	False	False	False	
7041	False	False	False	False	False	False	False	False	False	
7042	False	False	False	False	False	False	False	False	False	

7043 rows × 21 columns

In [11]:

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

Out[11]:

```
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
```

In [12]:

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

Out[12]:

```
0
```

In [13]:

```
df.describe()
```

Out[13]:

	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

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
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

In [14]:

```
df.duplicated()
```

Out[14]:

```
0       False
1       False
2       False
3       False
4       False
        ...
7038    False
7039    False
7040    False
7041    False
7042    False
Length: 7043, dtype: bool
```

In [15]:

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

Out[15]:

```
0
```

We will ensure that no duplicate customer-id is there:

In [16]:

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

Out[16]:

```
0
```

In [ ]:

```

```

In [ ]:

```

```

now we will apply a function to make senior-citizen COLUMN value as yes or no instead of any numerical value:

In [17]:

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

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

In [18]:

```
df.head(5) #check for first 5 values
```

Out[18]:

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

5 rows × 21 columns

In [ ]:

```
```

In [ ]:

```
```

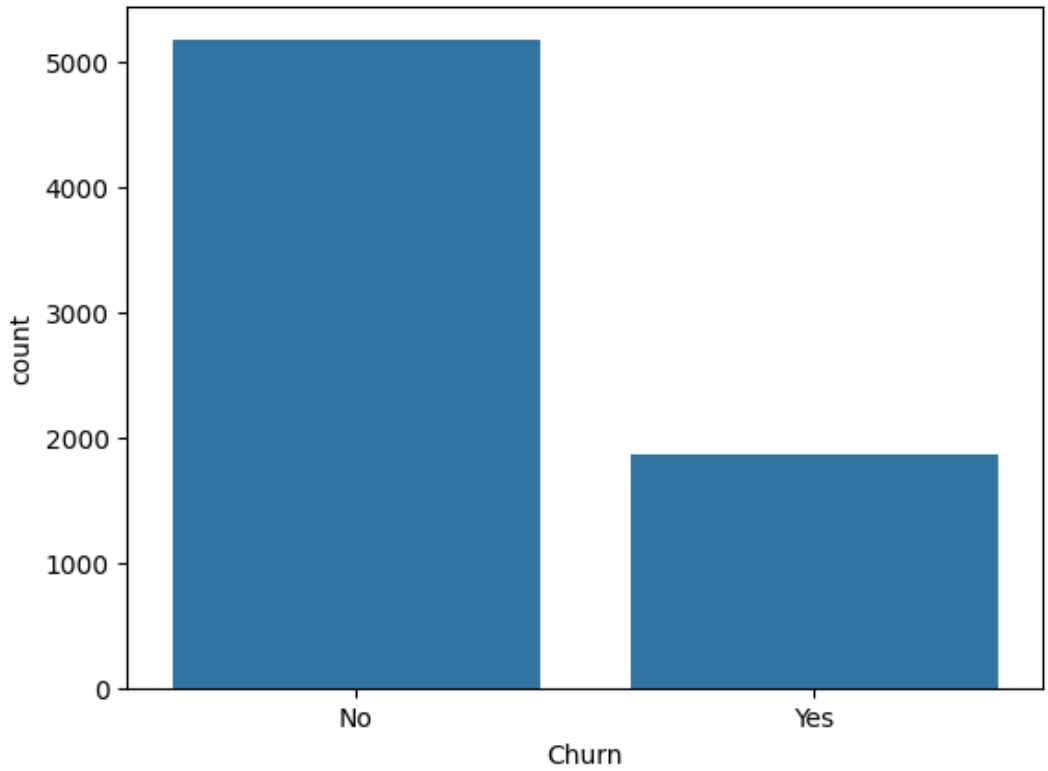
check how many customers have churned-out or not:

In [19]:

```
sns.countplot(x=df["Churn"],data=df)
```

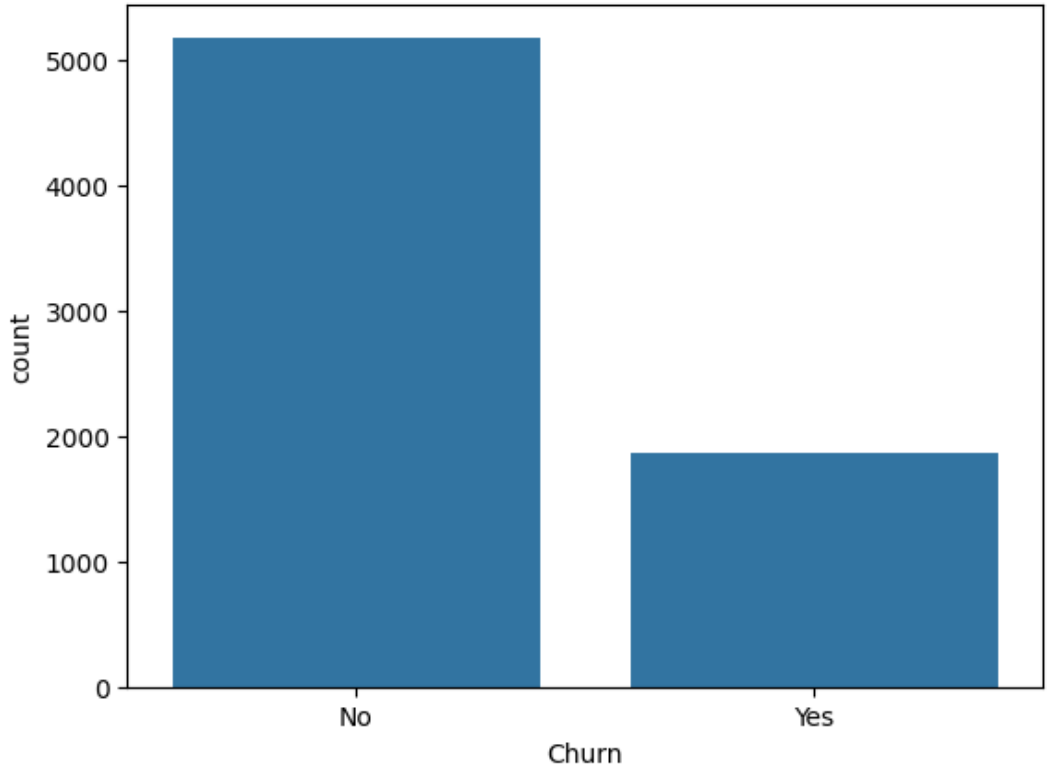
Out[19]:

```
<Axes: xlabel='Churn', ylabel='count'>
```



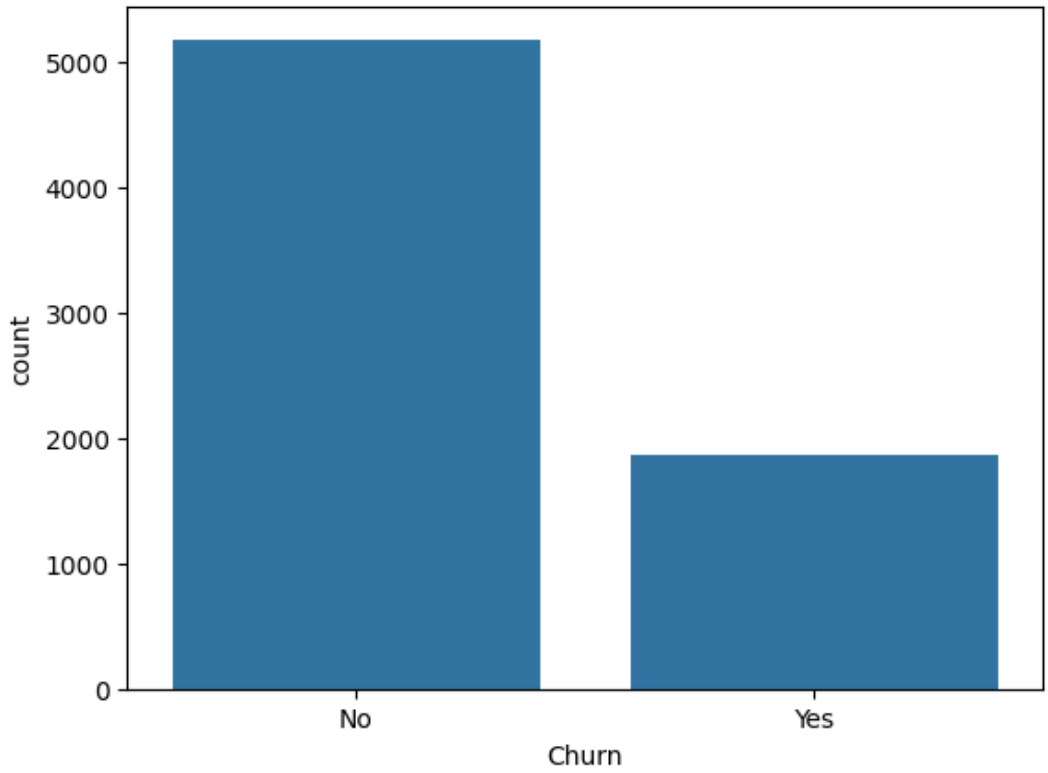
In [20]:

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



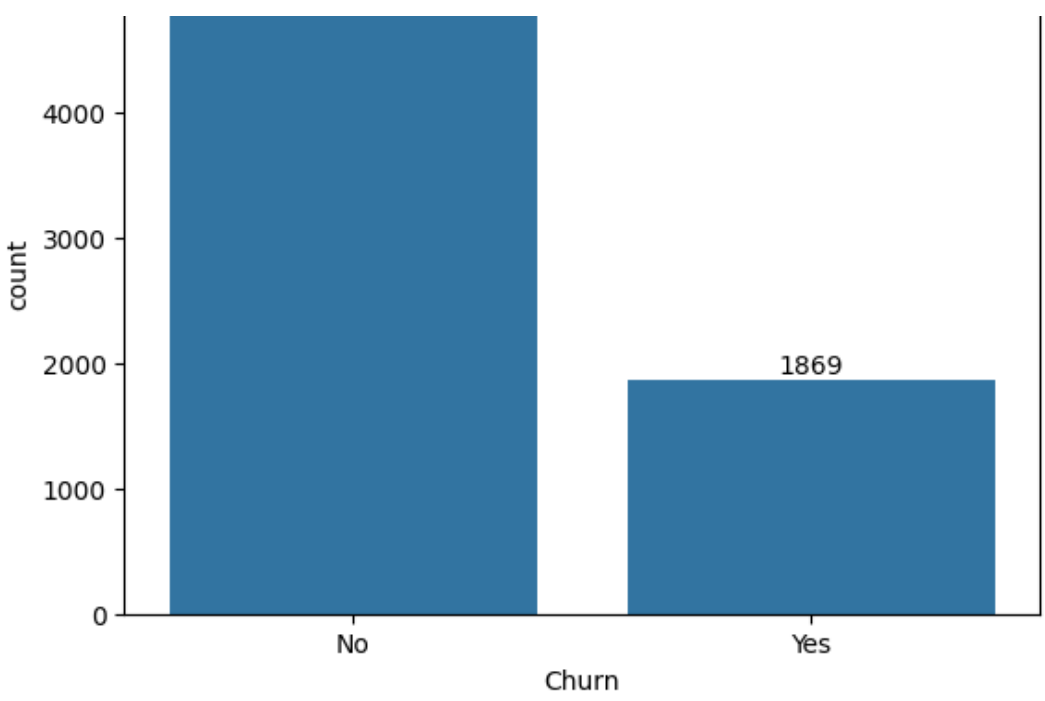
In [21]:

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



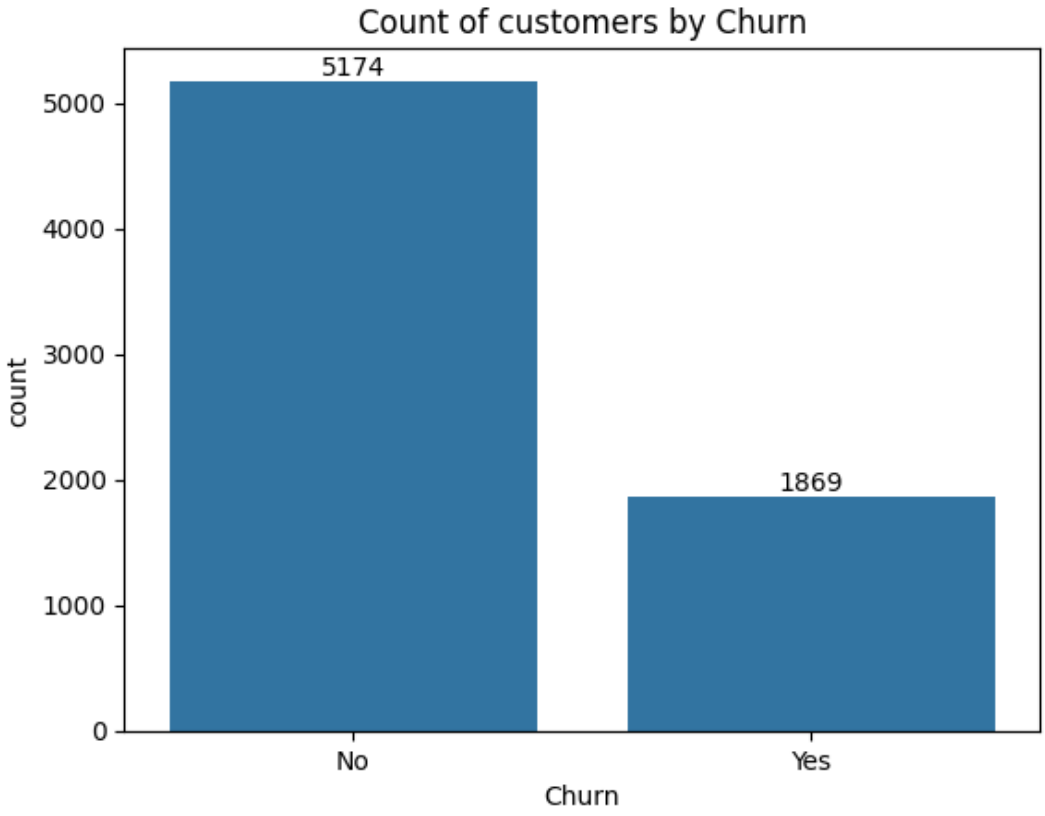
In [22]:

```
ax = sns.countplot(x="Churn",data=df)
ax.bar_label(ax.containers[0]) #check how many customers have churned out or not on top o
f the bars.
m.show()
```



In [23]:

```
ax = sns.countplot(x=df["Churn"],data=df)
ax.bar_label(ax.containers[0]) #check how many customers have churned out or not on top o
f the bars.
m.title("Count of customers by Churn")
m.show()
```



In [24]:

```
gb = df.groupby("Churn").agg({"Churn":"count"})
gb
```

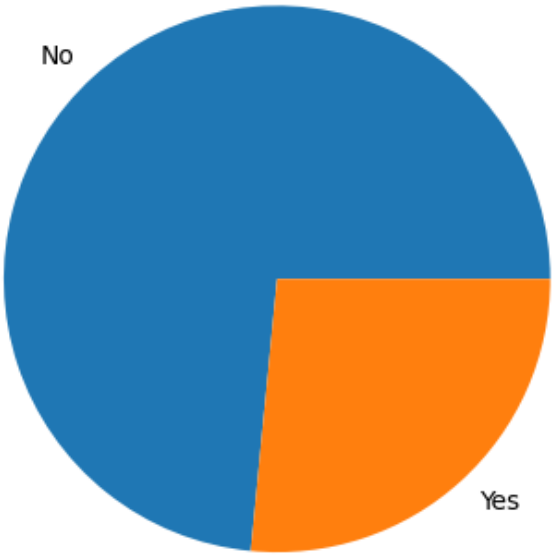
Out[24]:

	Churn
Churn	
No	**5174**

Yes 1869
Churn

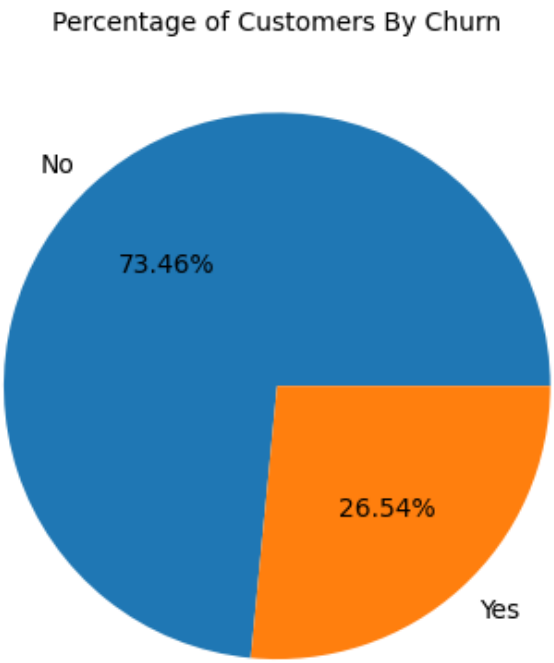
In [25]:

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



In [26]:

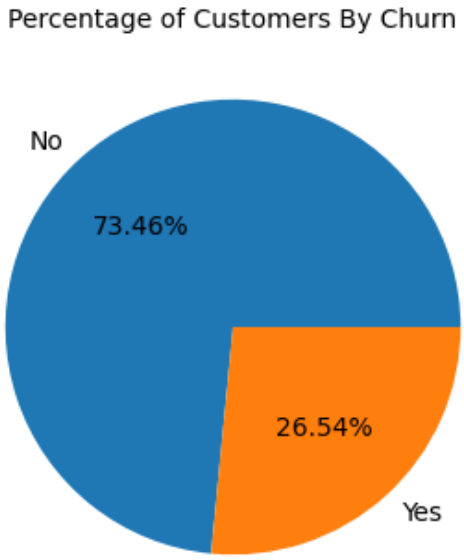
```
gb = df.groupby("Churn").agg({"Churn":"count"})
m.pie(gb["Churn"] , labels = gb.index, autopct = "%1.2f%%")
m.title("Percentage of Customers By Churn",fontsize=10)
m.show()
```



In [27]:

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

```
m.show()
```

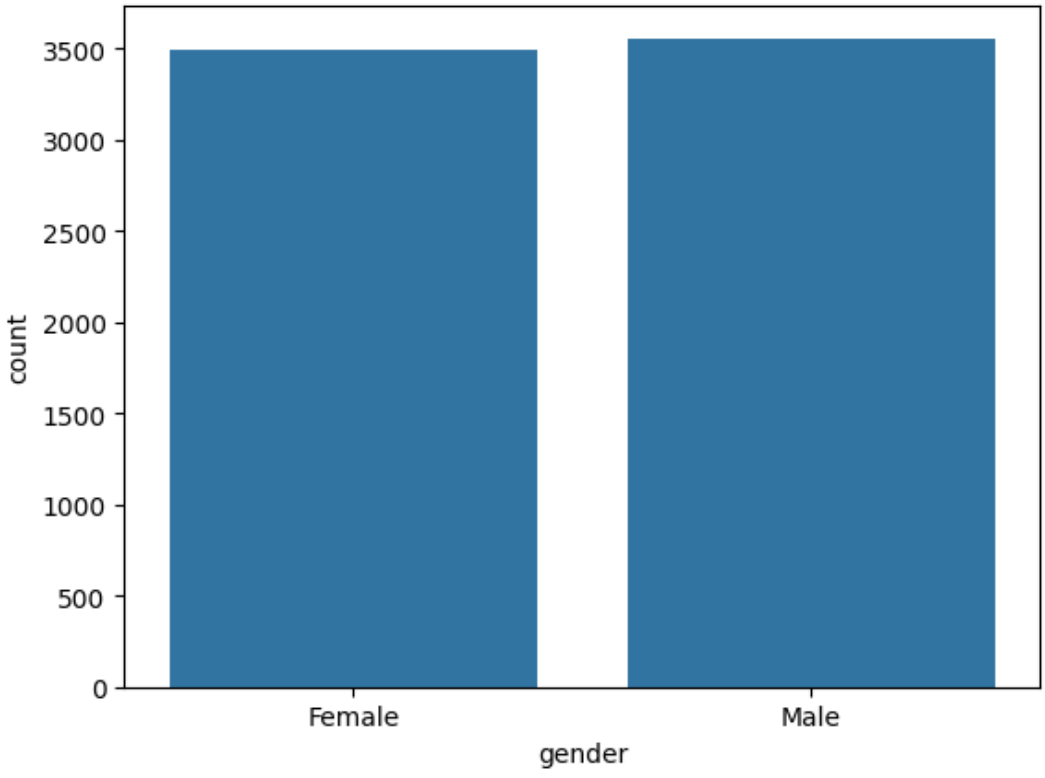


From the given pie chart we can conclude that 26.54% of our customers have churned out.

Now lets explore the reason behind it.

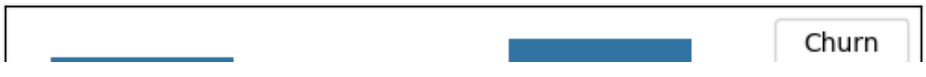
In [28]:

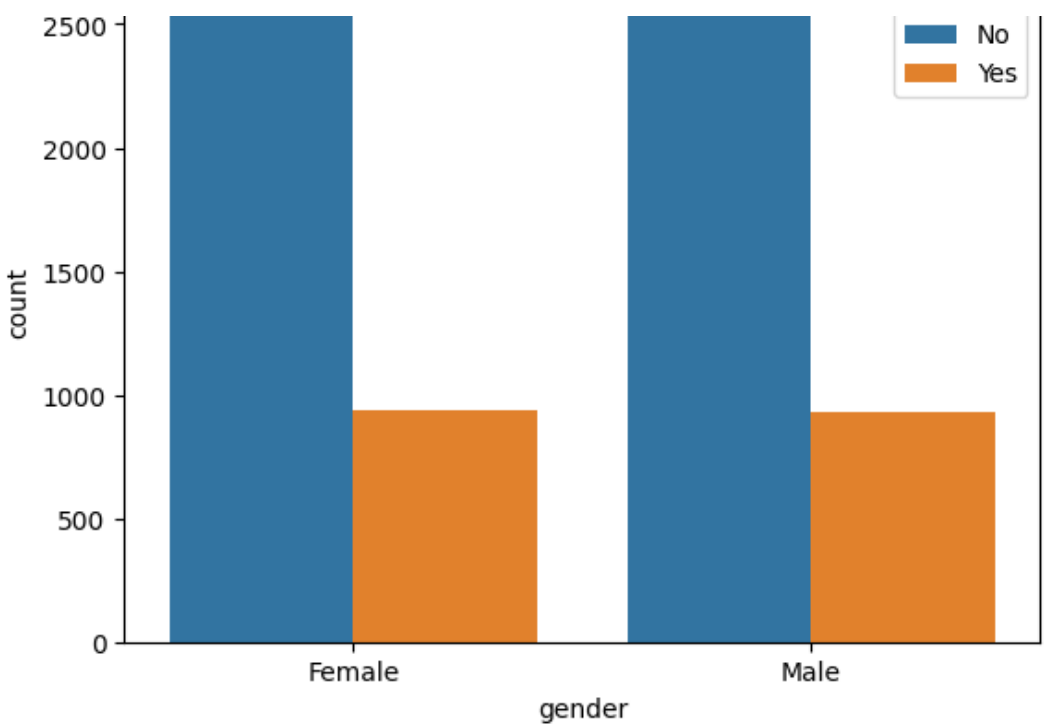
```
sns.countplot(x="gender",data=df)
m.show()
```



In [29]:

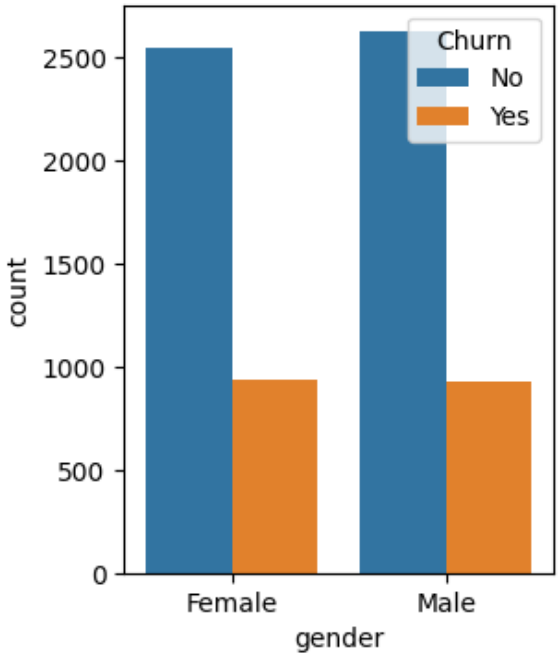
```
sns.countplot(x="gender",data=df,hue="Churn") #check by gender that how many people from each gender have churned out
m.show()
```





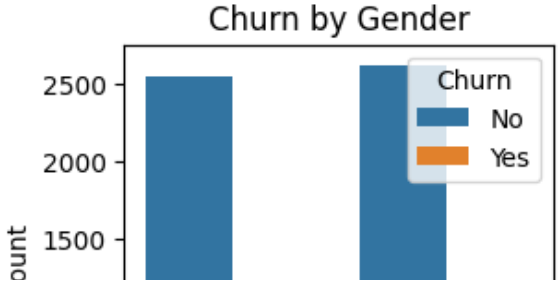
In [30]:

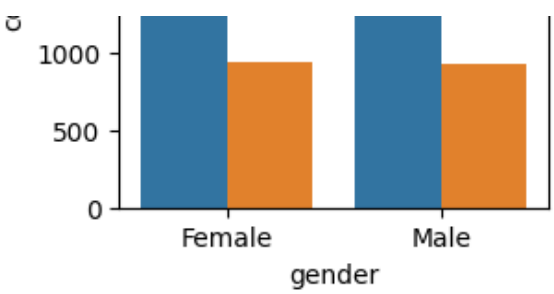
```
m.figure(figsize=(3,4))
sns.countplot(x="gender",data=df,hue="Churn") #check by gender that how many people from each gender have churned out
m.show()
```



In [31]:

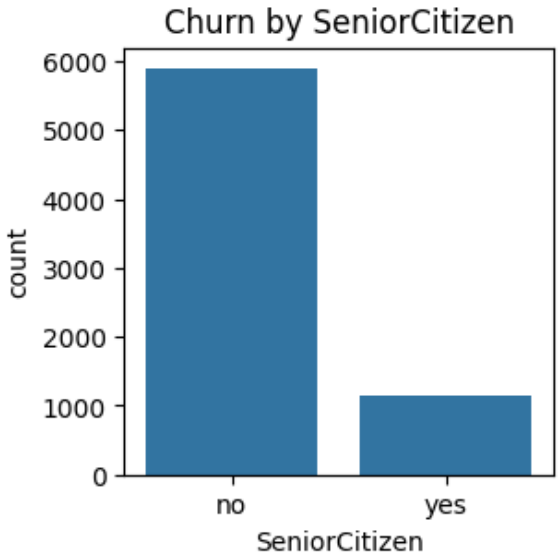
```
m.figure(figsize=(3,3))
sns.countplot(x="gender",data=df,hue="Churn") #check by gender that how many people from each gender have churned out
m.title("Churn by Gender")
m.show()
```





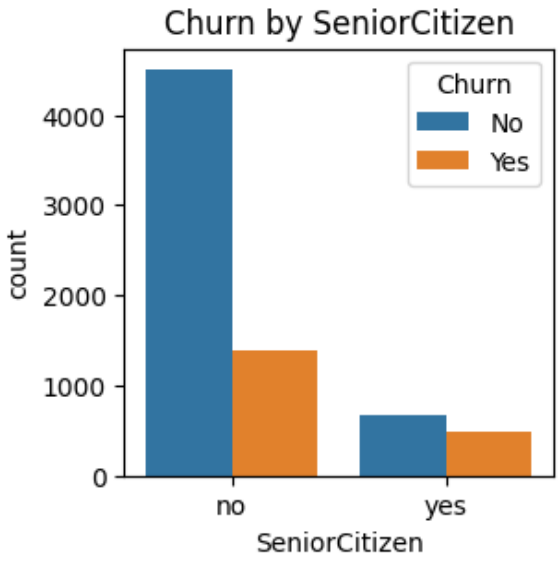
In [32]:

```
m.figure(figsize=(3,3))
sns.countplot(x="SeniorCitizen",data=df) #check by SeniorCitizen that how many people fro
m have churned out and how many have not
m.title("Churn by SeniorCitizen")
m.show()
```



In [33]:

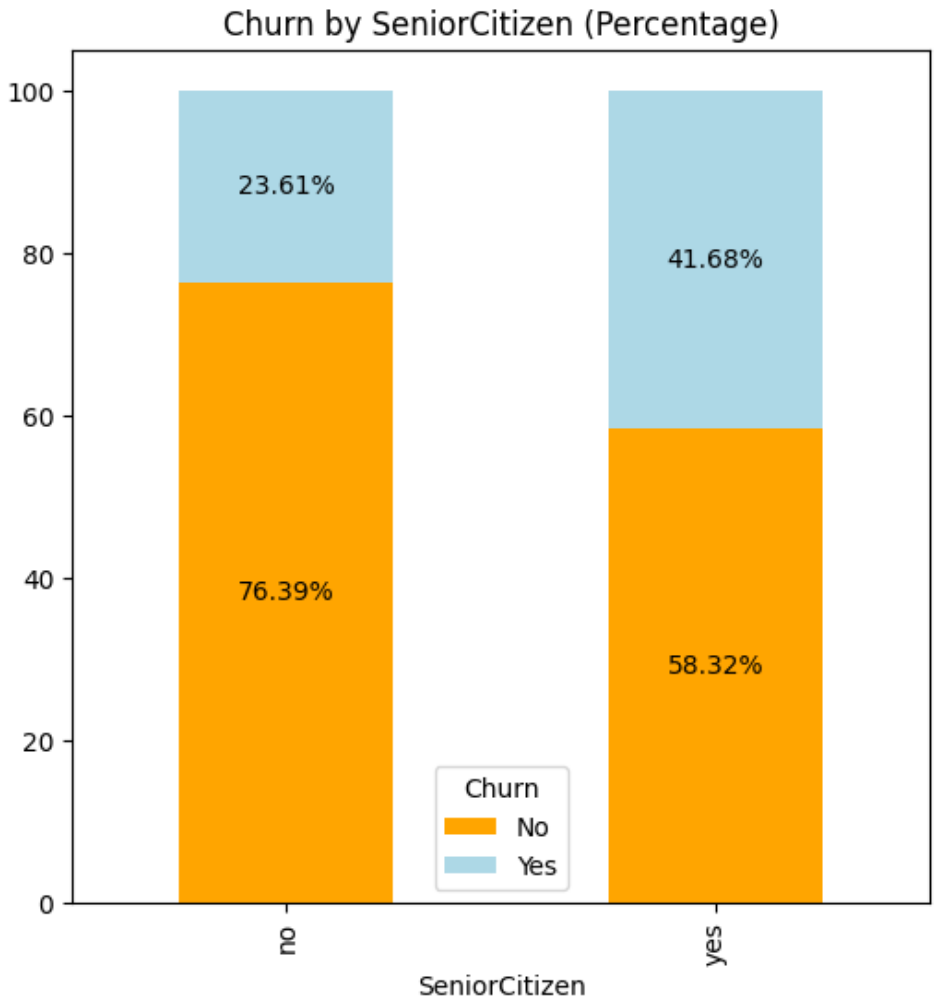
```
#how many senior citizens and non-senior citizens have churned (left the service) and how
many stayed:
m.figure(figsize=(3,3))
sns.countplot(x="SeniorCitizen",data=df,hue="Churn")
m.title("Churn by SeniorCitizen")
m.show()
```



In [34]:

```
#THE ABOVE GRAPH IN STACK BAR-CHART FORM:
count_data = df.groupby(['SeniorCitizen', 'Churn']).size().unstack(fill_value=0)
```

```
percentage_data = count_data.div(count_data.sum(axis=1), axis=0) * 100

# Plotting the stacked bar chart
m.figure(figsize=(6,6))
percentage_data.plot(kind='bar', stacked=True, color=['orange', 'lightblue'], ax=m.gca()
)

# Adding the percentage labels
for p in m.gca().patches:
    height = p.get_height()
    width = p.get_width()
    x, y = p.get_xy()  # Get the x and y position of the rectangle
    percentage = round(height, 2)  # Get the height (which is the percentage here)
    m.gca().text(x + width / 2, y + height / 2, f'{percentage}%', ha='center', va='cente
r', color='black')

m.title("Churn by SeniorCitizen (Percentage)")
m.show()
```

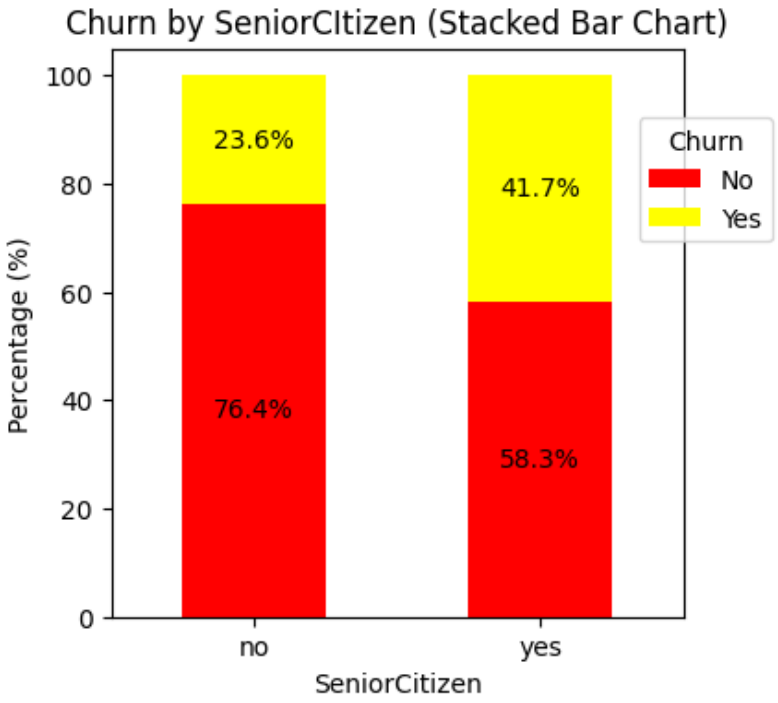


comparative a gretaed percentage of people in senior-citizen catgeory have churned

In [37]:

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

fig,ax=m.subplots(figsize=(4,4))
total_counts.plot(kind='bar',stacked=True,ax=ax,color=['red','yellow'])

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')
```

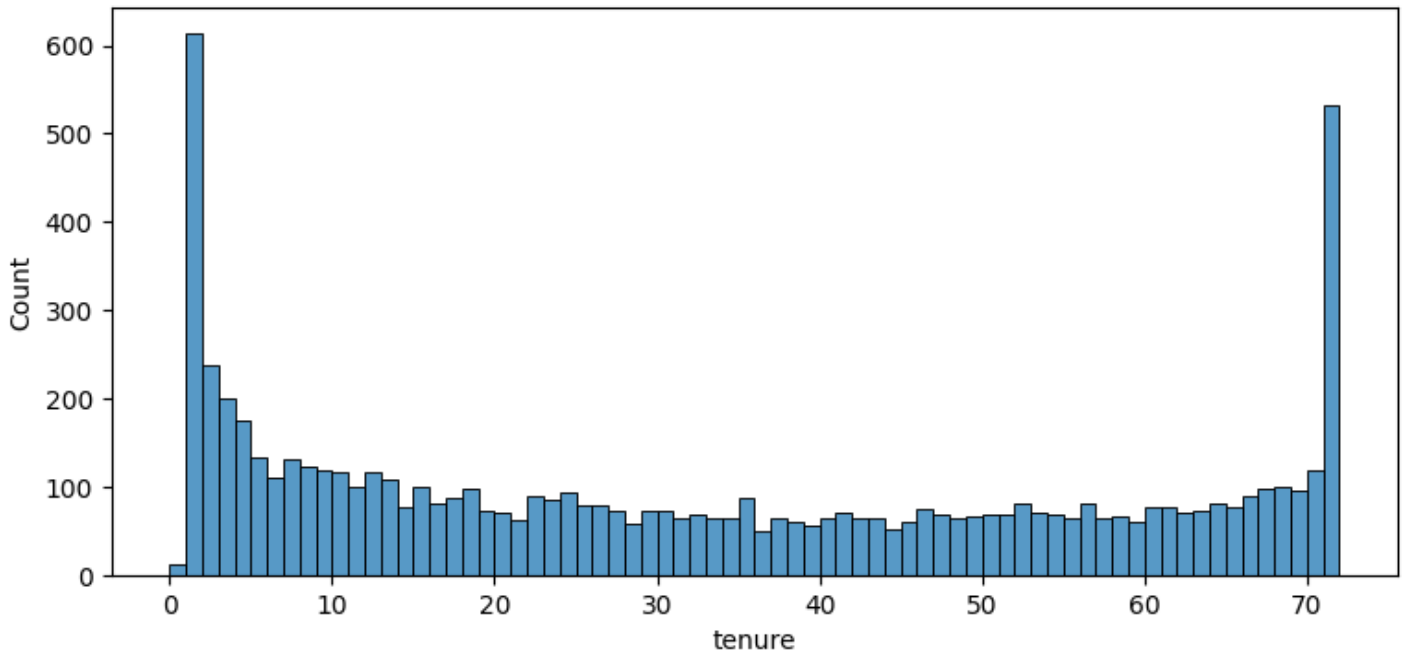
```
m.title('Churn by SeniorCItizen (Stacked Bar Chart)')
m.xlabel('SeniorCitizen')
m.ylabel('Percentage (%)')
m.xticks(rotation=0)
m.legend(title='Churn',bbox_to_anchor=(0.9,0.9))

m.show()
```



how many customers have churned out based on tenure:

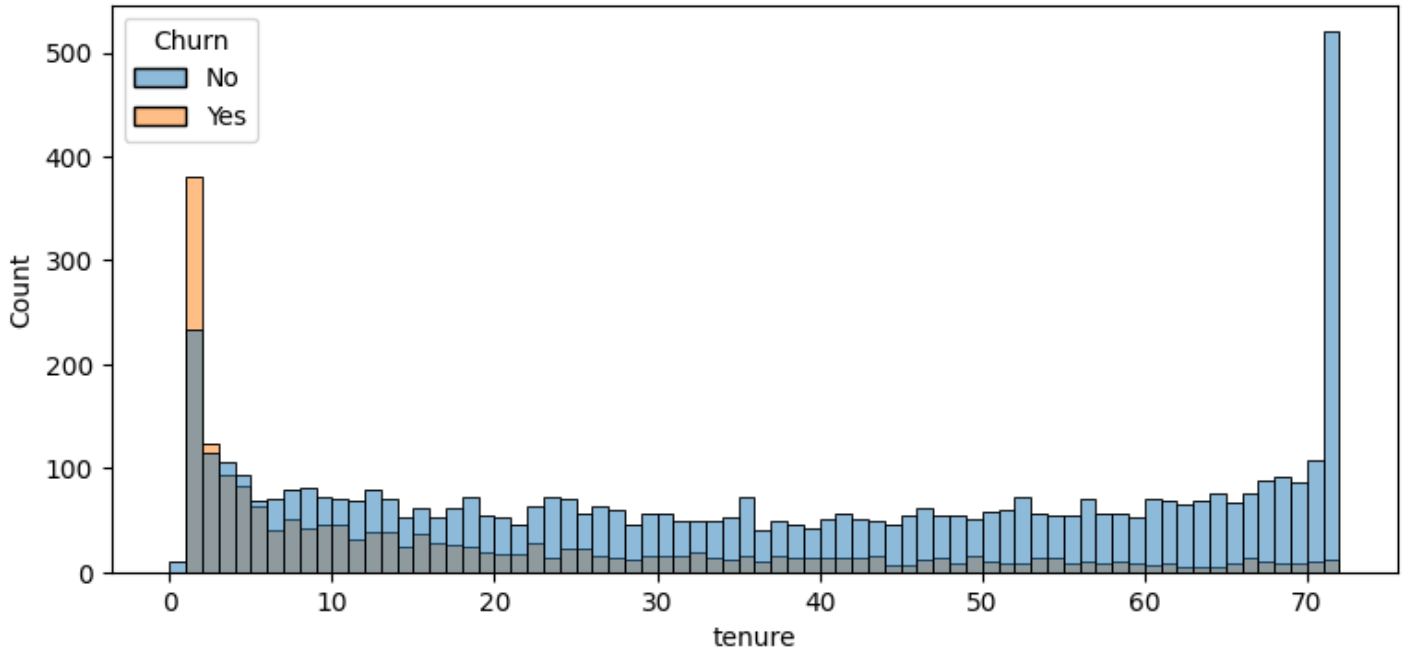
In [47]:

```
m.figure(figsize=(9,4)) #width,height
sns.histplot(x="tenure",data=df,bins=72) #72 bars hence 72 bins
m.show()
```



In [46]:

```
m.figure(figsize=(9,4)) #width,height
sns.histplot(x="tenure",data=df,bins=72, hue="Churn") #72 bars hence 72 bins
m.show()
```

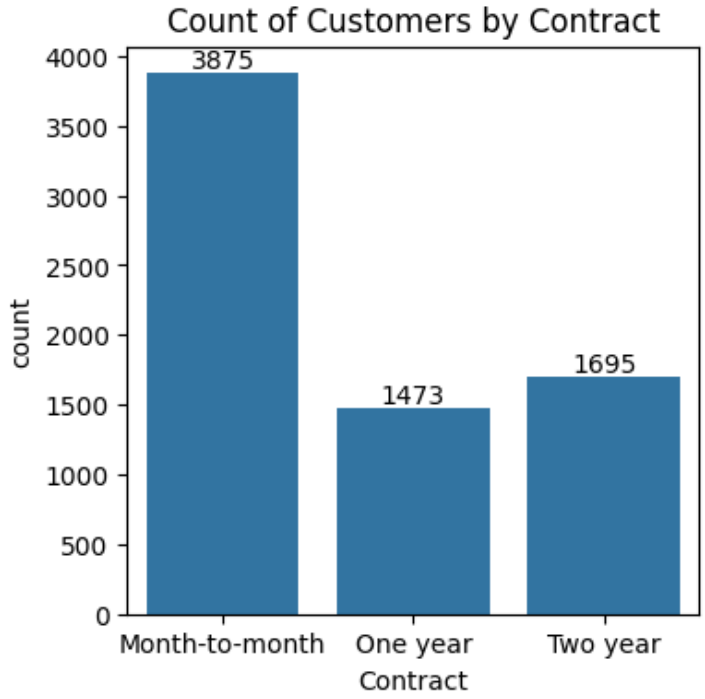


people who have used our services for a longer time have stayed and people who have used our services

for 1 or 2 months have churned

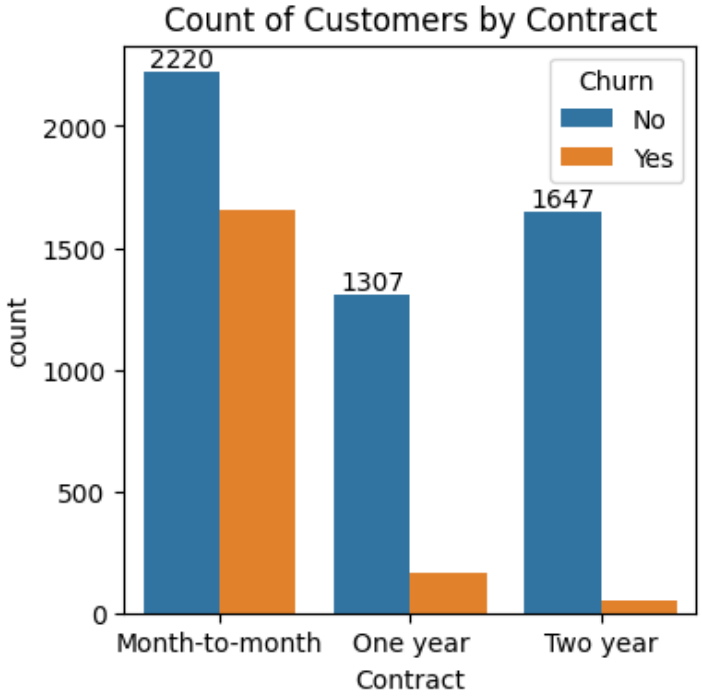
In [51]:

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



In [52]:

```
m.figure(figsize=(4,4))
ax=sns.countplot(x="Contract",data=df,hue="Churn")
ax.bar_label(ax.containers[0])
m.title("Count of Customers by Contract")
m.show()
```



people who have month-to-month contract are likely

to churn out then from those who have 1 or 2 years of contract.

In [53]:

```
df.columns.values
```

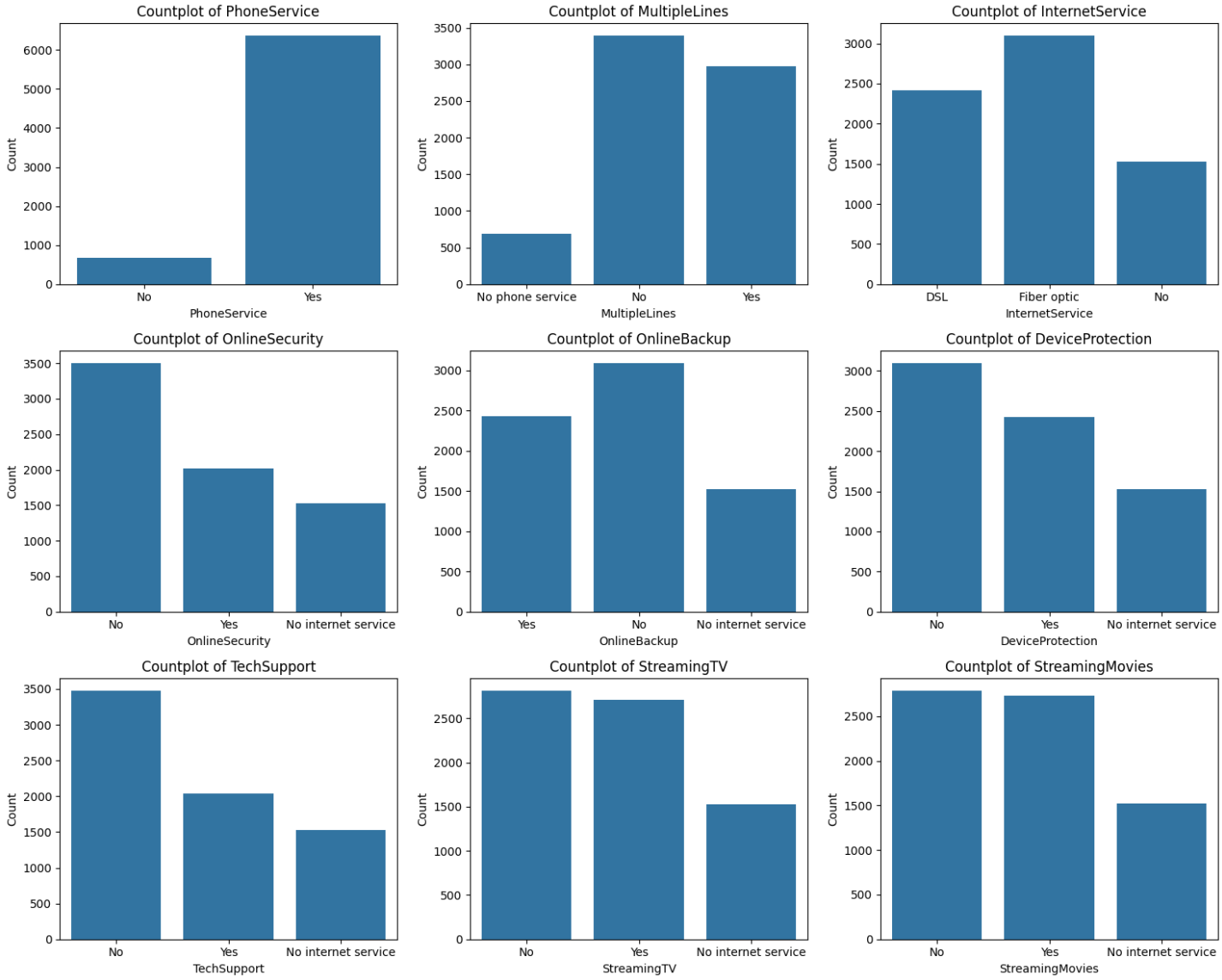
Out[53]:

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

In [58]:

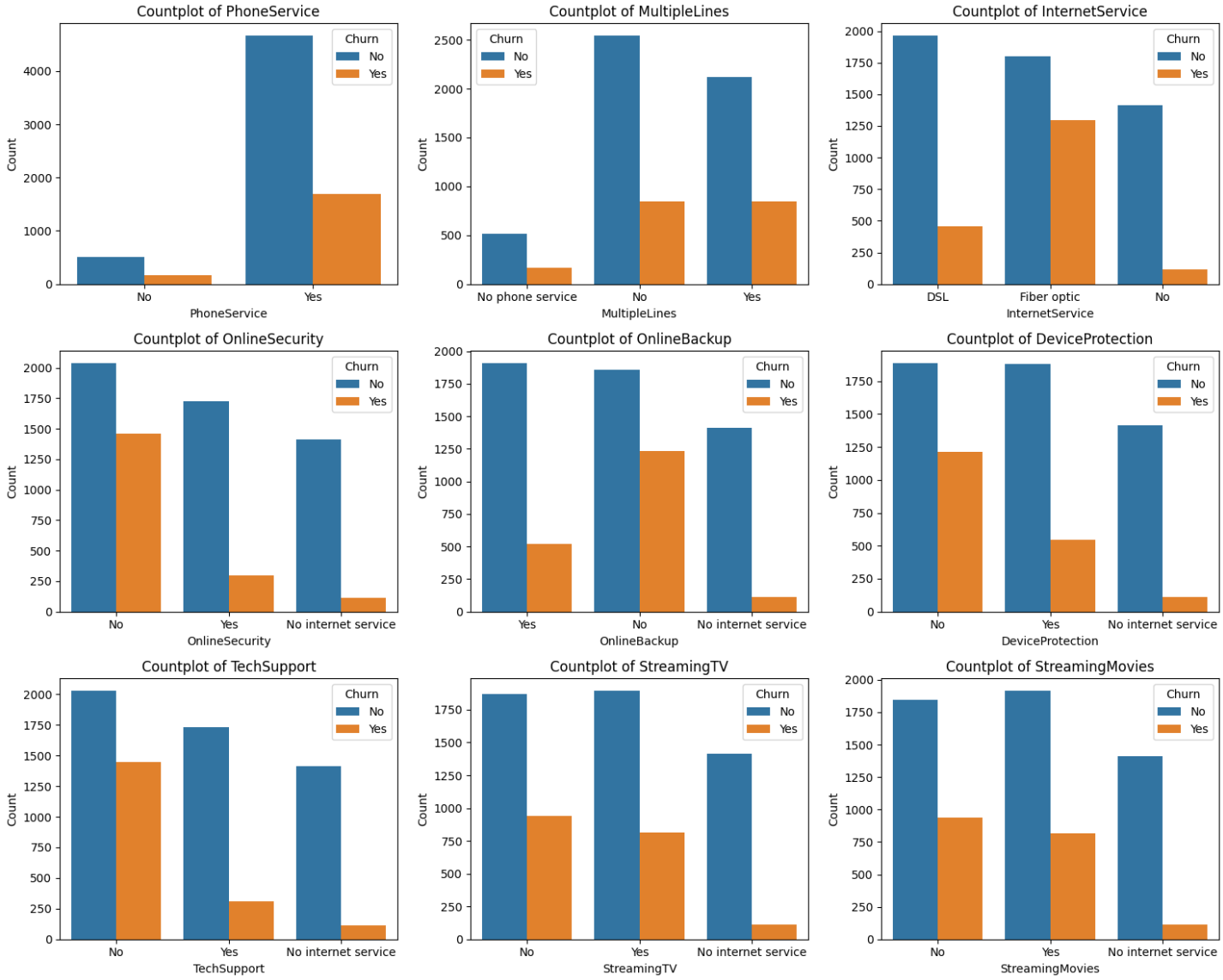
```
# List of columns you want to plot
cols = ['PhoneService', 'MultipleLines', 'InternetService',
        'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
        'TechSupport', 'StreamingTV', 'StreamingMovies']

# Define number of rows and columns for subplots
n_cols = 3
n_rows =  (len(cols) + n_cols - 1) // n_cols


# Set figure size dynamically
fig, axes = m.subplots(n_rows, n_cols, figsize=(15,n_rows*4))
axes = axes.flatten()  # Flatten to make indexing easier

# Create countplots
for i, col in enumerate(cols):
    sns.countplot(x=col, data=df, ax=axes[i])
    axes[i].set_title(f'Countplot of {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Count')

# Remove any extra empty subplots (in case of mismatch)
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])
```

```
m.tight_layout()
m.show()
```



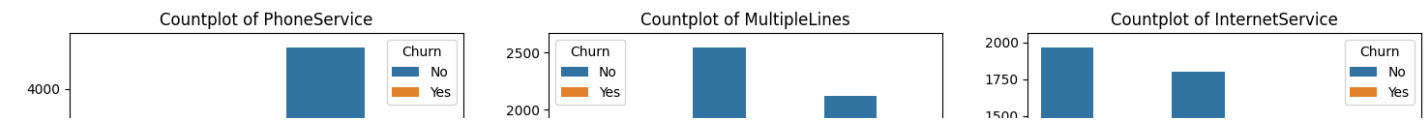
In [59]:

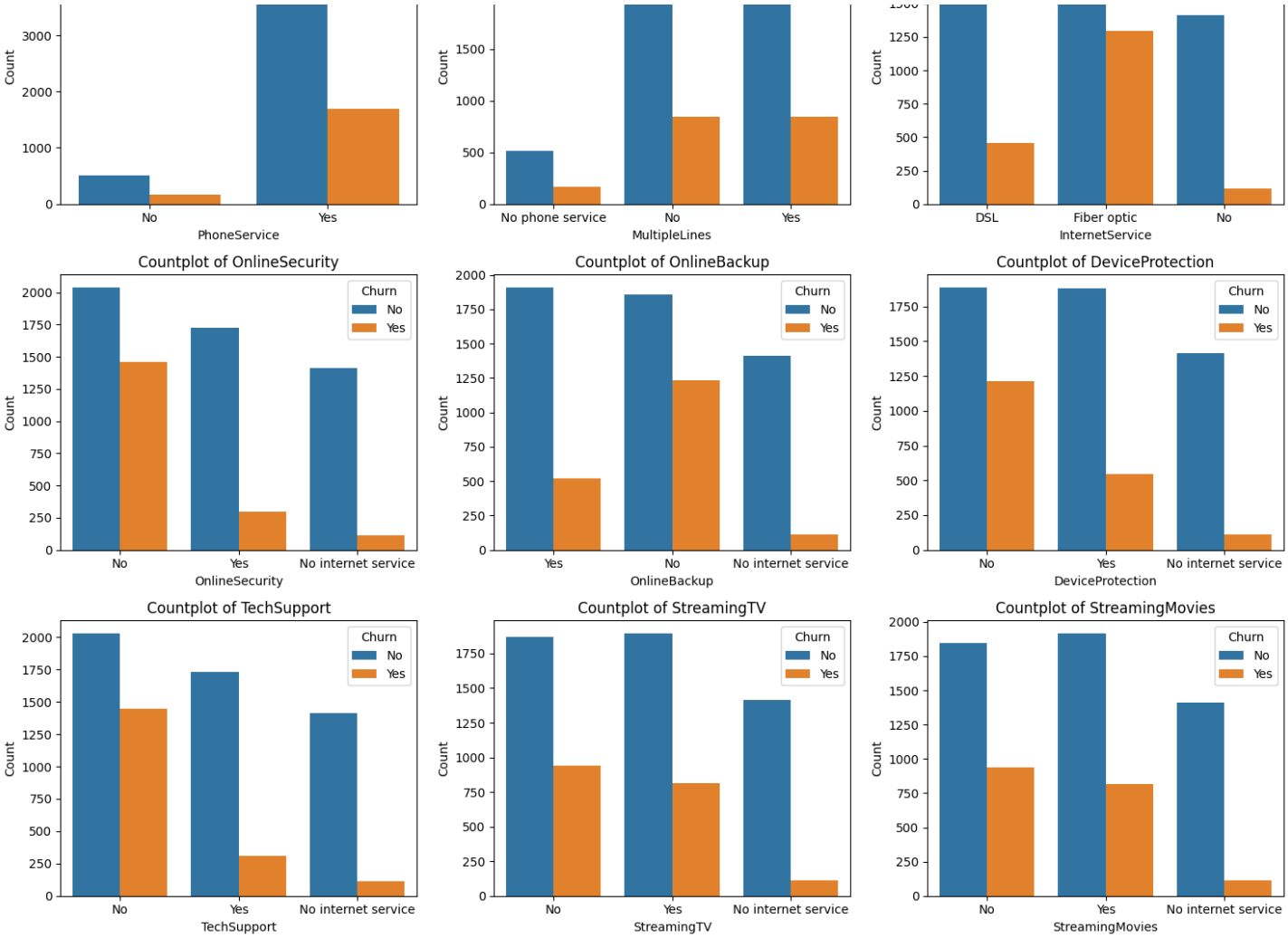
```
# List of columns you want to plot
cols = ['PhoneService', 'MultipleLines', 'InternetService',
        'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
        'TechSupport', 'StreamingTV', 'StreamingMovies']

# Define number of rows and columns for subplots
n_cols = 3
n_rows =  (len(cols) + n_cols - 1) // n_cols


# Set figure size dynamically
fig, axes = m.subplots(n_rows, n_cols, figsize=(15,n_rows*4))
axes = axes.flatten()  # Flatten to make indexing easier

# Create countplots
for i, col in enumerate(cols):
    sns.countplot(x=col, data=df, ax=axes[i],hue=df["Churn"])
    axes[i].set_title(f'Countplot of {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Count')

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

m.tight_layout()
m.show()
```



In [60]:

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

# Define number of rows and columns for subplots
n_cols = 3
n_rows =  (len(cols) + n_cols - 1) // n_cols


# Set figure size dynamically
fig, axes = m.subplots(n_rows, n_cols, figsize=(15,n_rows*4))
axes = axes.flatten()  # Flatten to make indexing easier

# Create countplots
for i, col in enumerate(cols):
    sns.countplot(x=col, data=df, ax=axes[i],hue="Churn")
    axes[i].set_title(f'Countplot of {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Count')

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

m.tight_layout()
m.show()
```

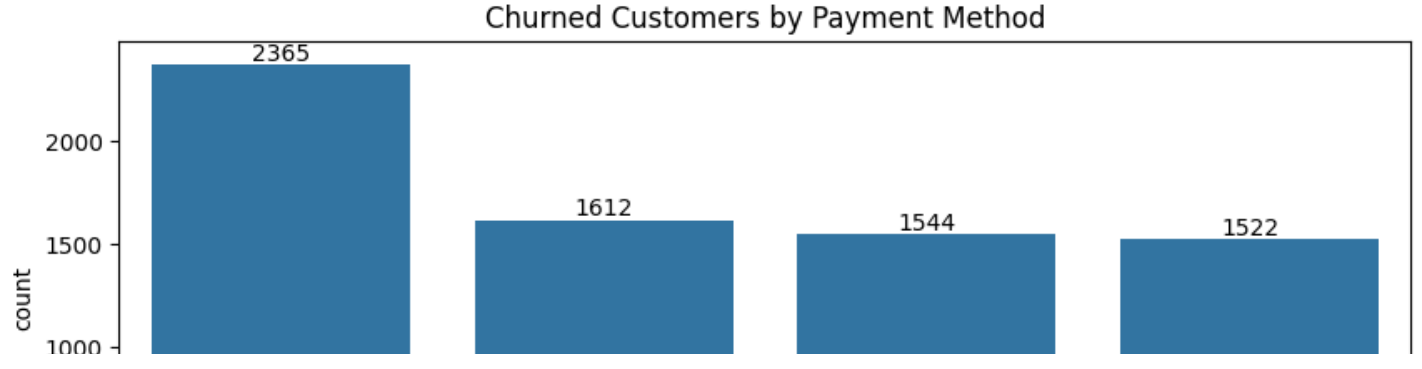


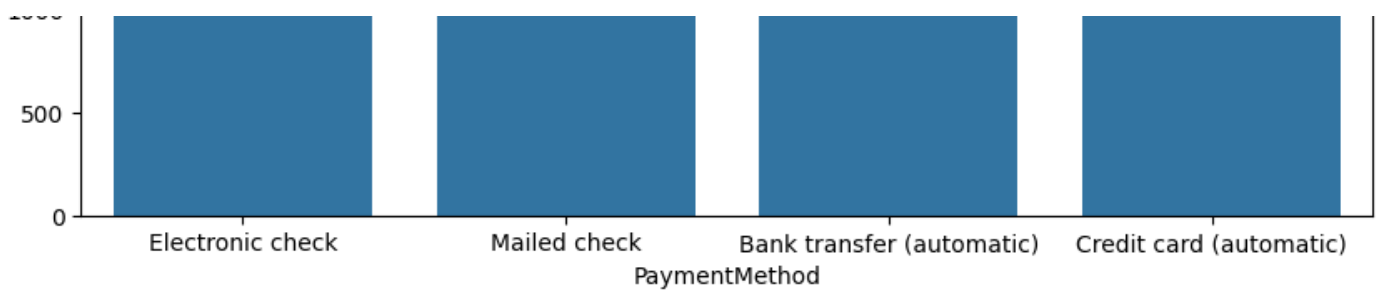


Customers without internet services or related add-ons like online security, backup, and tech support tend to churn less, indicating that minimal service users are more stable. Those using fiber optic internet have significantly higher churn compared to DSL users. Streaming services such as StreamingTV and StreamingMovies show a more balanced churn rate, suggesting they have less impact on customer retention. Overall, customers subscribed to multiple technical services appear more likely to churn, possibly due to higher costs or dissatisfaction.

In [66]:

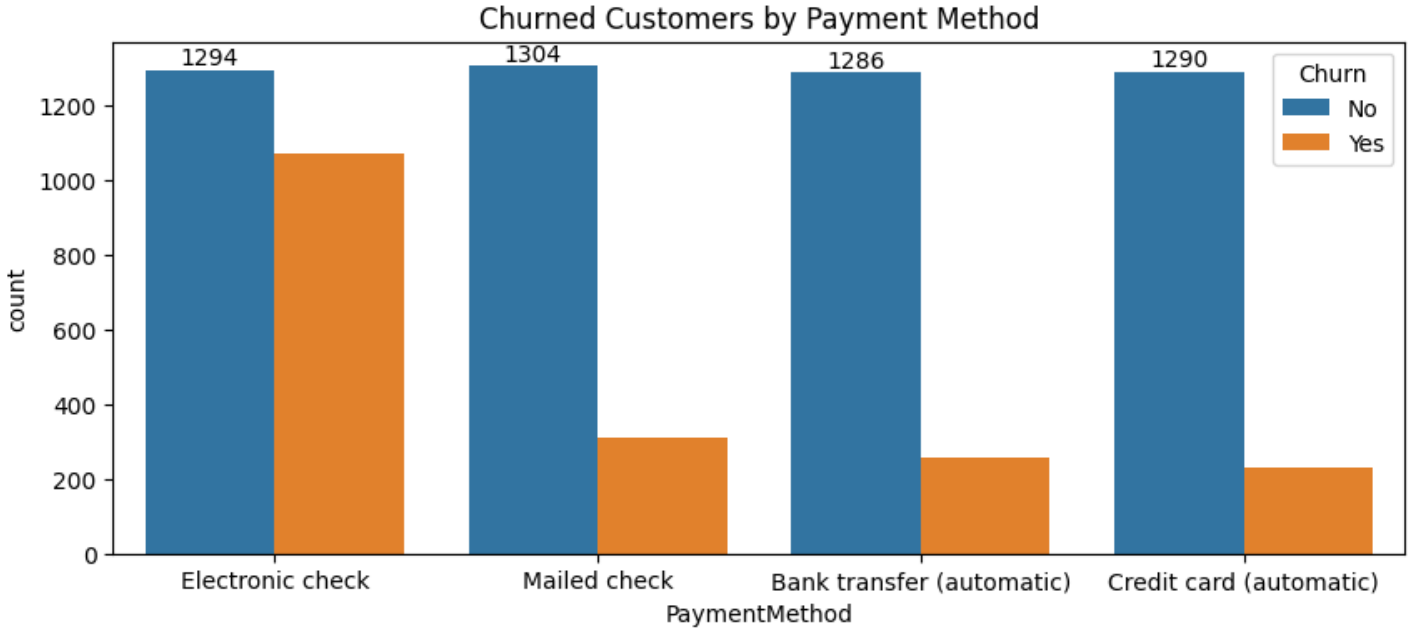
```
m.figure(figsize=(10,4)) #width,height
ax=sns.countplot(x="PaymentMethod",data=df)
ax.bar_label(ax.containers[0])
m.title("Churned Customers by Payment Method")
m.show()
```





In [67]:

```
m.figure(figsize=(10,4)) #width,height
ax=sns.countplot(x="PaymentMethod",data=df,hue="Churn")
ax.bar_label(ax.containers[0])
m.title("Churned Customers by Payment Method")
m.show()
```



Customer is likely to churn when he is using Electronic-Check as a payment method

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```

```

In [ ]:

```

```