In [1]:
```
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(5)
```

Out[1]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	..
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

In [2]:
```
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 because as tenure is 0 and no total charges are recorded.

In [3]:
```
df["TotalCharges"]=df["TotalCharges"].replace(" ","0")
df["TotalCharges"]=df["TotalCharges"].astype("float")
```

In [4]:
```
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 [5]:
```
df.isnull().sum()          # to check if there is a null value in the data set
```

Out[5]:
```
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 [6]:
```
df.describe()
```

Out[6]:

	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

In [7]:
```
df.duplicated().sum()                  # to check if there are duplicates in the data set
df["customerID"].duplicated().sum()#checking duplicates in "customerID".its a unique value
```

Out[7]: 0

In [8]:
```
def conv(value):
    if value==1:            #converted 0 and 1 values of "senior citizen" to yes/No to make
        return "yes"        #      it easier to understand.
    else:
        return "No"
```

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

In [9]:
```
df.head(30)
```

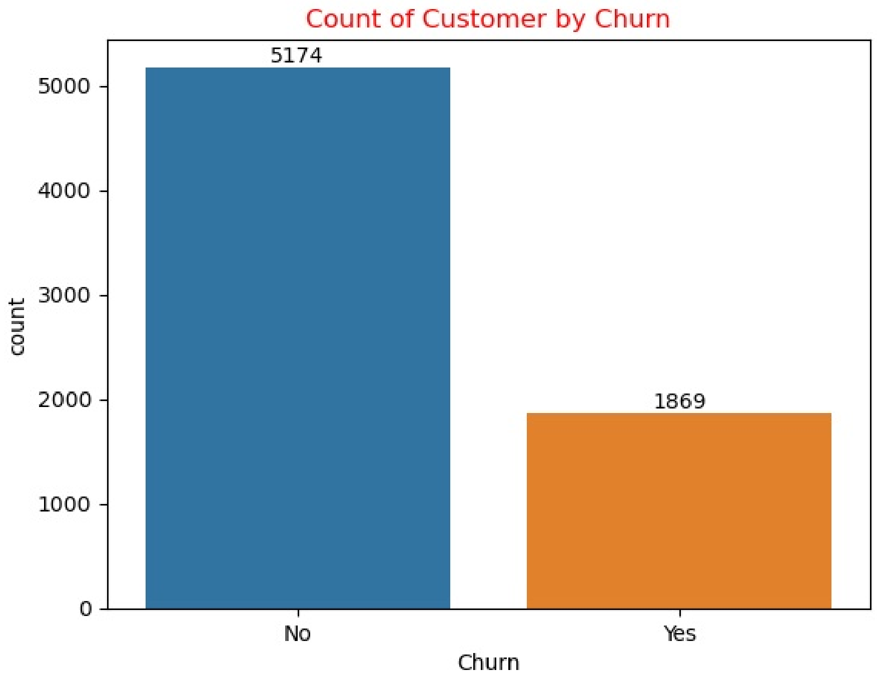
Out[9]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity
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	9305-CDSKC	Female	No	No	No	8	Yes	Yes	Fiber optic	No
6	1452-KIOVK	Male	No	No	Yes	22	Yes	Yes	Fiber optic	No
7	6713-OKOMC	Female	No	No	No	10	No	No phone service	DSL	Yes
8	7892-POOKP	Female	No	Yes	No	28	Yes	Yes	Fiber optic	No
9	6388-TABGU	Male	No	No	Yes	62	Yes	No	DSL	Yes
10	9763-GRSKD	Male	No	Yes	Yes	13	Yes	No	DSL	Yes
11	7469-LKBCI	Male	No	No	No	16	Yes	No	No	No internet service
12	8091-TTVAX	Male	No	Yes	No	58	Yes	Yes	Fiber optic	No
13	0280-XJGEX	Male	No	No	No	49	Yes	Yes	Fiber optic	No
14	5129-JLPIS	Male	No	No	No	25	Yes	No	Fiber optic	Yes
15	3655-SNQYZ	Female	No	Yes	Yes	69	Yes	Yes	Fiber optic	Yes
16	8191-XWSZG	Female	No	No	No	52	Yes	No	No	No internet service
17	9959-WOFKT	Male	No	No	Yes	71	Yes	Yes	Fiber optic	Yes
18	4190-MFLUW	Female	No	Yes	Yes	10	Yes	No	DSL	No
19	4183-MYFRB	Female	No	No	No	21	Yes	No	Fiber optic	No
20	8779-QRDMV	Male	yes	No	No	1	No	No phone service	DSL	No
21	1680-VDCWW	Male	No	Yes	No	12	Yes	No	No	No internet service
22	1066-JKSGK	Male	No	No	No	1	Yes	No	No	No internet service
23	3638-WEABW	Female	No	Yes	No	58	Yes	Yes	DSL	No
24	6322-HRPFA	Male	No	Yes	Yes	49	Yes	No	DSL	Yes
25	6865-JZNKO	Female	No	No	No	30	Yes	No	DSL	Yes
26	6467-CHFZW	Male	No	Yes	Yes	47	Yes	Yes	Fiber optic	No
27	8665-UTDHZ	Male	No	Yes	Yes	1	No	No phone service	DSL	No
28	5248-YGIJN	Male	No	Yes	No	72	Yes	Yes	DSL	Yes
29	8773-HHUOZ	Female	No	No	Yes	17	Yes	No	DSL	No

30 rows × 21 columns

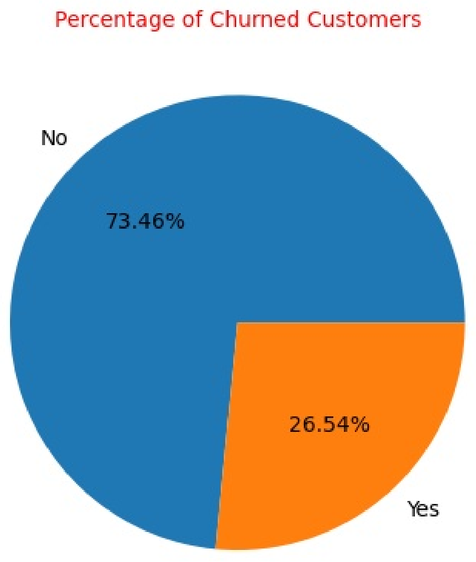
In [10]:
```
ax=sns.countplot(x="Churn",data=df)
```

```
ax.bar_label(ax.containers[0])          # to see values of yes/no in numbers
plt.title("Count of Customer by Churn",color="r")
plt.show()
```



we can also view it in pie chart as below

In [11]:
```
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,color="r")
plt.figure(figsize=(5,5))
plt.show()
```

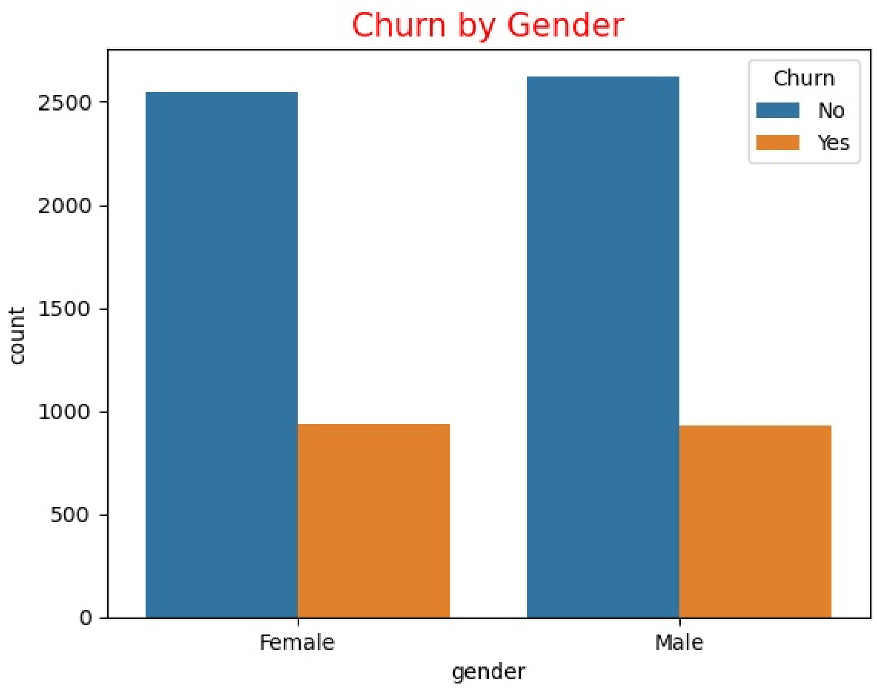


```
<Figure size 500x500 with 0 Axes>
```

from the above pie chart we can conclude that 26.54% of customers have churned out

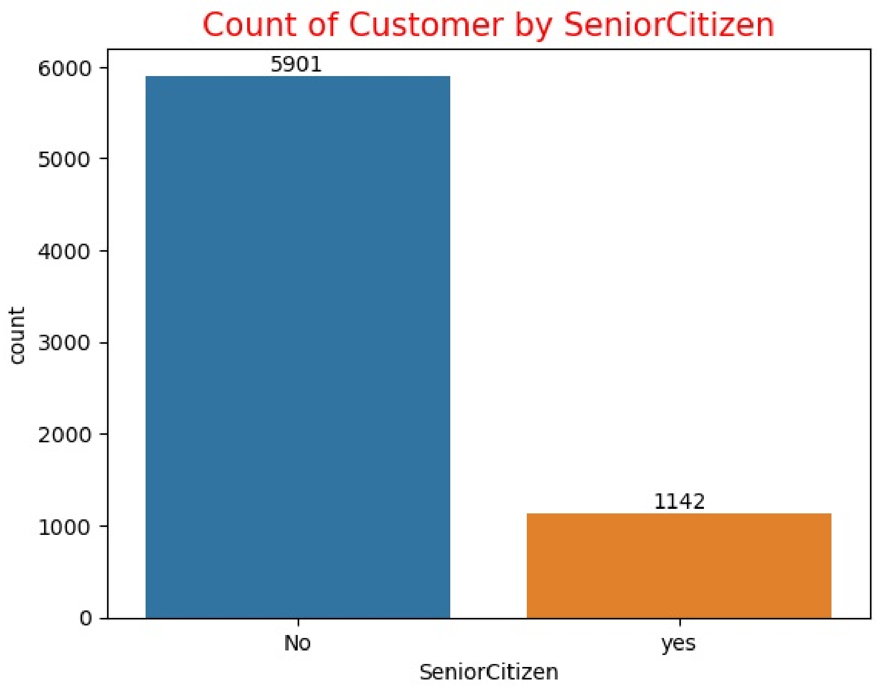
so lets explore the reason behind it

In [12]:
```
sns.countplot(x="gender",data=df,hue="Churn")
plt.title("Churn by Gender",fontsize=15,color="red")
plt.figure(figsize=(4,4))
plt.show()
```



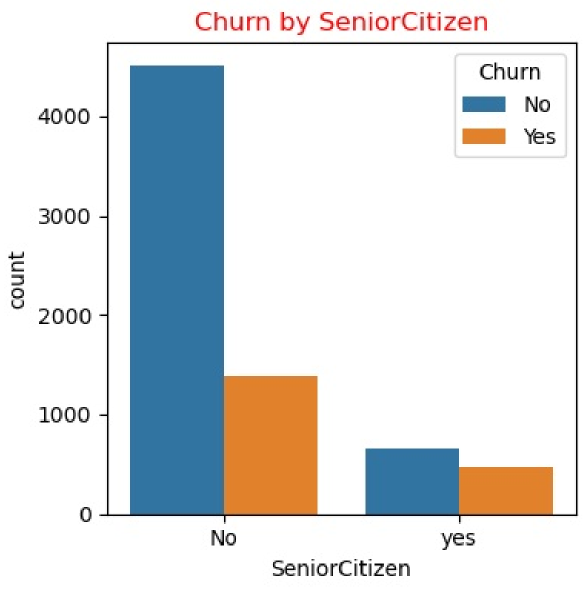
```
<Figure size 400x400 with 0 Axes>
```

In [13]:
```python
ax=sns.countplot(x="SeniorCitizen",data=df)
ax.bar_label(ax.containers[0])
plt.title("Count of Customer by SeniorCitizen",fontsize=15,color="red")
plt.figure(figsize=(4,4))
plt.show()
```

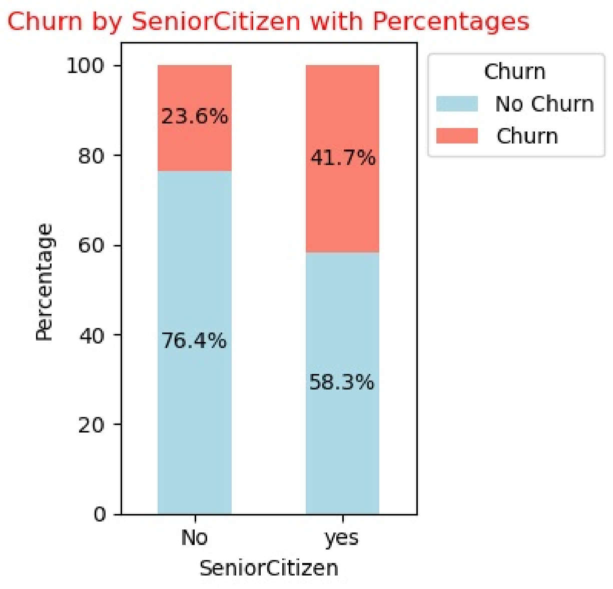


```
<Figure size 400x400 with 0 Axes>
```

In [14]:
```python
plt.figure(figsize=(4,4))
sns.countplot(x="SeniorCitizen",data=df,hue="Churn")
plt.title("Churn by SeniorCitizen",color="red")
plt.tight_layout()                      # to make sure that the map doesnt over lap
plt.show()
```



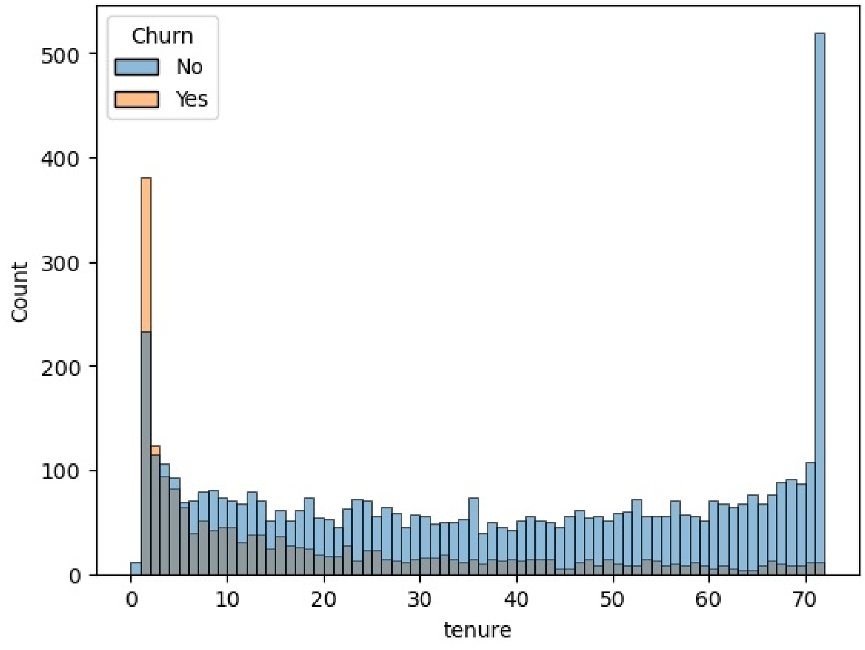
In [15]:
```python
# Assuming df is your DataFrame
# First calculate the percentage of churn for each 'SeniorCitizen' group
churn_percentages = df.groupby(['SeniorCitizen', 'Churn']).size().unstack().apply(lambda x: x / x.sum(), axis=1

# Now plot a stacked bar chart with the percentages
ax = churn_percentages.plot(kind='bar', stacked=True, color=['lightblue', 'salmon'], figsize=(4, 4))

# Add title and labels
plt.title("Churn by SeniorCitizen with Percentages", color="red")
plt.xlabel("SeniorCitizen")
plt.ylabel("Percentage")
plt.legend(title="Churn", labels=["No Churn", "Churn"], loc="upper left", bbox_to_anchor=(1, 1))
plt.xticks(rotation=0)

# Overlay percentage values on the bar chart
for p in ax.patches:
    height = p.get_height()
    width = p.get_width()
    x = p.get_x()
    y = p.get_y()

    # Display the percentage on top of the bar
    ax.text(x + width / 2, y + height / 2, f'{height:.1f}%', ha="center", va="center", color="black")

plt.tight_layout()

# Show the plot
plt.show()
```



comparative a greater percentage of people in senior citizen have churned from the above analysis

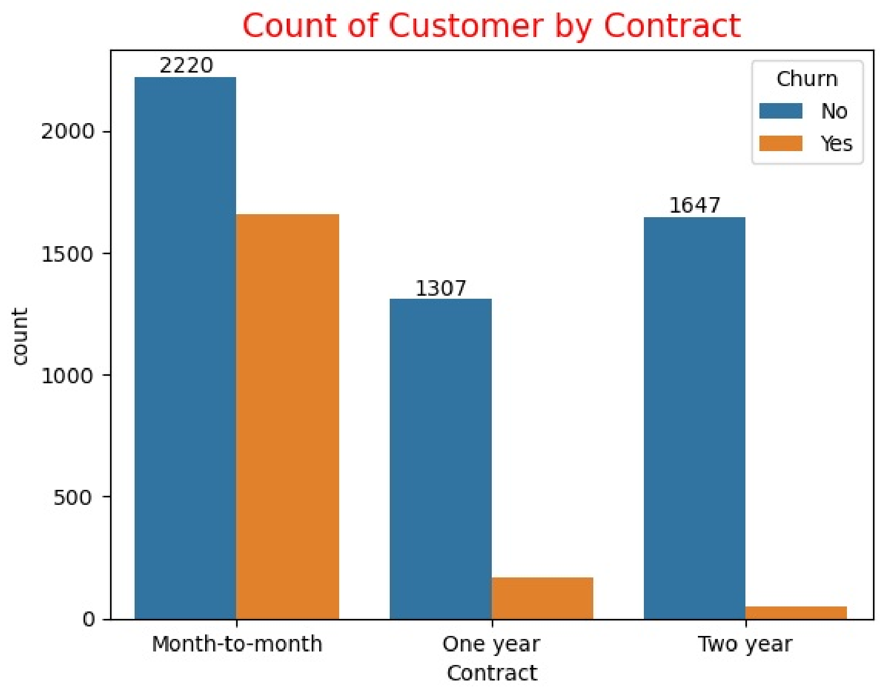
```
In [16]: sns.histplot(x="tenure",data=df,bins=72,hue="Churn")
         plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



from the above analysis 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

```
In [17]: ax=sns.countplot(x="Contract",data=df,hue="Churn")
         ax.bar_label(ax.containers[0])
         plt.title("Count of Customer by Contract",fontsize=15,color="red")
         plt.figure(figsize=(4,4))
         plt.show()
```



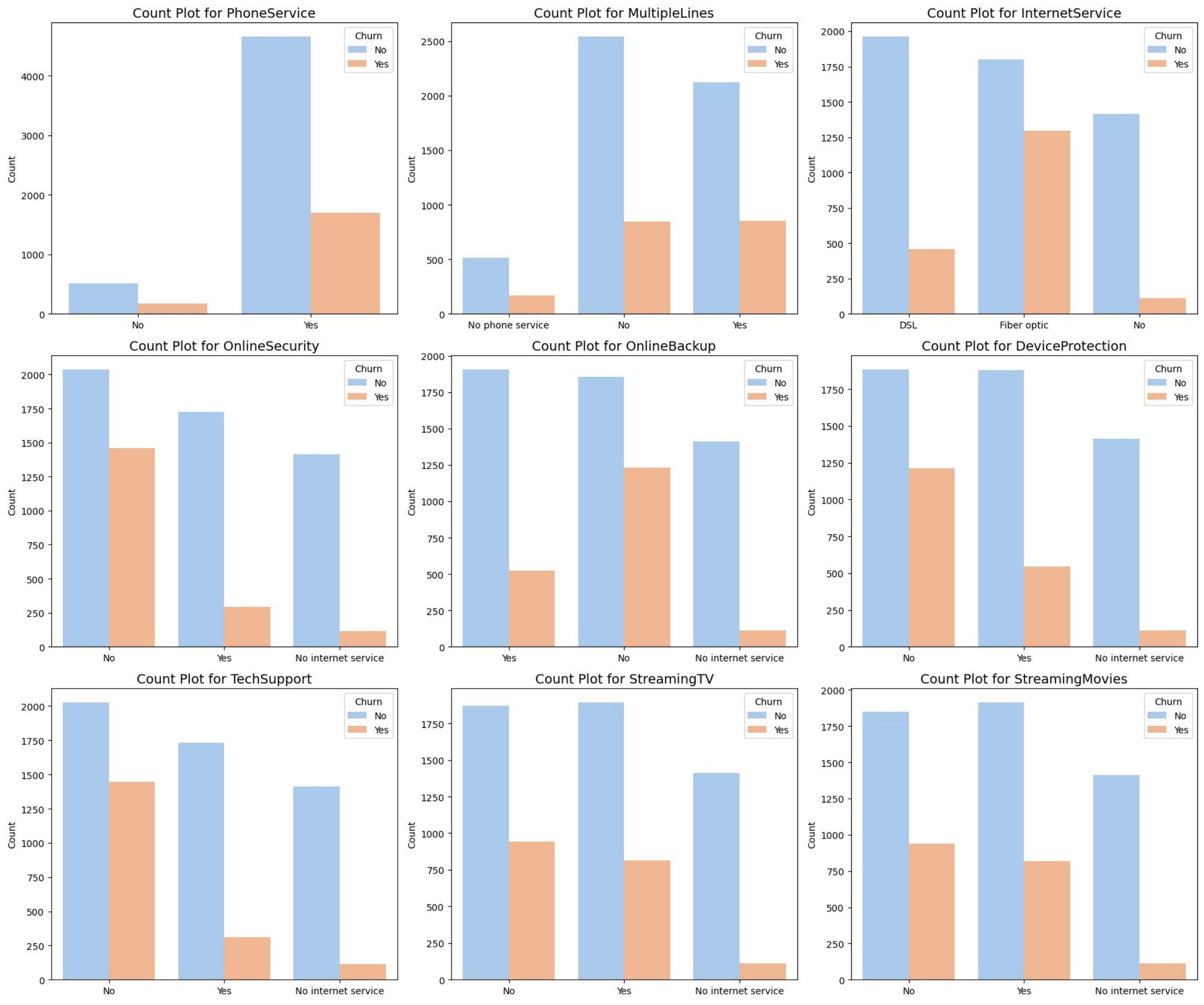
```
<Figure size 400x400 with 0 Axes>
```

from the above analysis people who have month to month are likely to churn from those who have 1 or 2 years of contract

In [18]:
```
df.columns.values
```

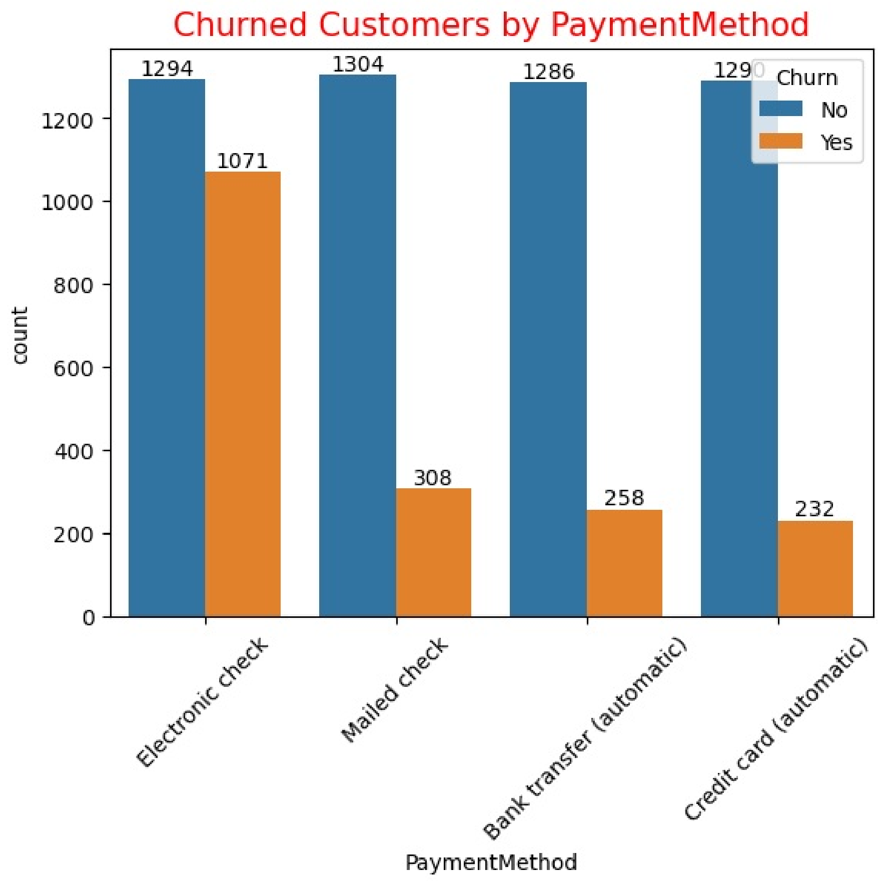
Out[18]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

In [19]:
```
# Define the columns of interest
columns_of_interest = [
    'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
    'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies'
]

# Create subplots (3 rows x 3 columns)
fig, axes = plt.subplots(3, 3, figsize=(18, 15))
axes = axes.flatten()  # Flatten axes array for easy iteration

# Generate count plots for each column
for i, column in enumerate(columns_of_interest):
    sns.countplot(data=df, x=column, ax=axes[i], palette="pastel",hue="Churn")
    axes[i].set_title(f'Count Plot for {column}', fontsize=14)
    axes[i].set_xlabel('')
    axes[i].set_ylabel('Count')

# Remove any empty subplots if columns < total grid slots
for j in range(len(columns_of_interest), len(axes)):
    fig.delaxes(axes[j])

# Adjust layout
plt.tight_layout()
plt.show()
```



the majority of the customers who do not churn tend to have services like PhoneService (particularly DSL), and OnlineSecurity enabled for services like OnlineBackup, TechSupport and StreamingTV, churn rates noticeably higher when these services are not used or available.

```
In [20]: 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 PaymentMethod",fontsize=15,color="red")
         plt.xticks(rotation=45)
         plt.figure(figsize=(4,4))
         plt.show()
```



```
<Figure size 400x400 with 0 Axes>
```

from the above analysis customer is likely to churn when he is using electronic check as a payment method

```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js