In [7]:
```
pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\hrith\appdata\local\programs\python\python312\lib\site-package
s (2.3.0)
Requirement already satisfied: numpy>=1.26.0 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-
packages (from pandas) (2.3.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\hrith\appdata\local\programs\python\python312\
lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-p
ackages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\hrith\appdata\local\programs\python\python312\lib\site
-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packa
ges (from python-dateutil>=2.8.2->pandas) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
```

In [8]:
```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\hrith\appdata\local\programs\python\python312\lib\site-pac
kages (3.10.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\hrith\appdata\local\programs\python\python312\lib\si
te-packages (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-p
ackages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\hrith\appdata\local\programs\python\python312\lib\s
ite-packages (from matplotlib) (4.58.4)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\hrith\appdata\local\programs\python\python312\lib\s
ite-packages (from matplotlib) (1.4.8)
Requirement already satisfied: numpy>=1.23 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-pa
ckages (from matplotlib) (2.3.1)
Requirement already satisfied: packaging>=20.0 in c:\users\hrith\appdata\local\programs\python\python312\lib\sit
e-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-pack
ages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\hrith\appdata\local\programs\python\python312\lib\si
te-packages (from matplotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\hrith\appdata\local\programs\python\python312\li
b\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packa
ges (from python-dateutil>=2.7->matplotlib) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
```

In [9]:
```
pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packages
(2.3.1)
Note: you may need to restart the kernel to use updated packages.
```

In [10]:
```
pip install seaborn
```

```
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packages (from seaborn) (2.3.1)
Requirement already satisfied: pandas>=1.2 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packages (from seaborn) (2.3.0)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packages (from seaborn) (3.10.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.2)
Requirement already satisfied: cycler>=0.10 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.58.4)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.8)
Requirement already satisfied: packaging>=20.0 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (25.0)
Requirement already satisfied: pillow>=8 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\hrith\appdata\local\programs\python\python312\lib\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.17.0)
Downloading seaborn-0.13.2-py3-none-any.whl (294 kB)
Installing collected packages: seaborn
Successfully installed seaborn-0.13.2
Note: you may need to restart the kernel to use updated packages.
```

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

Out[5]:

	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 [6]:
```
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 chargees are recorded

In [7]:
```
df["TotalCharges"] = df["TotalCharges"].replace(" ","0")
df["TotalCharges"] = df["TotalCharges"].astype("float")
```

In [8]:
```
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 [9]:
```
df.isnull().sum()
```

Out[9]:
```
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 [10]:
```
df.describe()
```

Out[10]:

	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 [11]:
```
df["customerID"].duplicated().sum()
```

Out[11]:
```
np.int64(0)
```

In [12]:
```
def conv(value):
    if value == 1:
        return "yes"
    else:
        return"no"

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

converted 0 or 1 to ye and no for better understanding

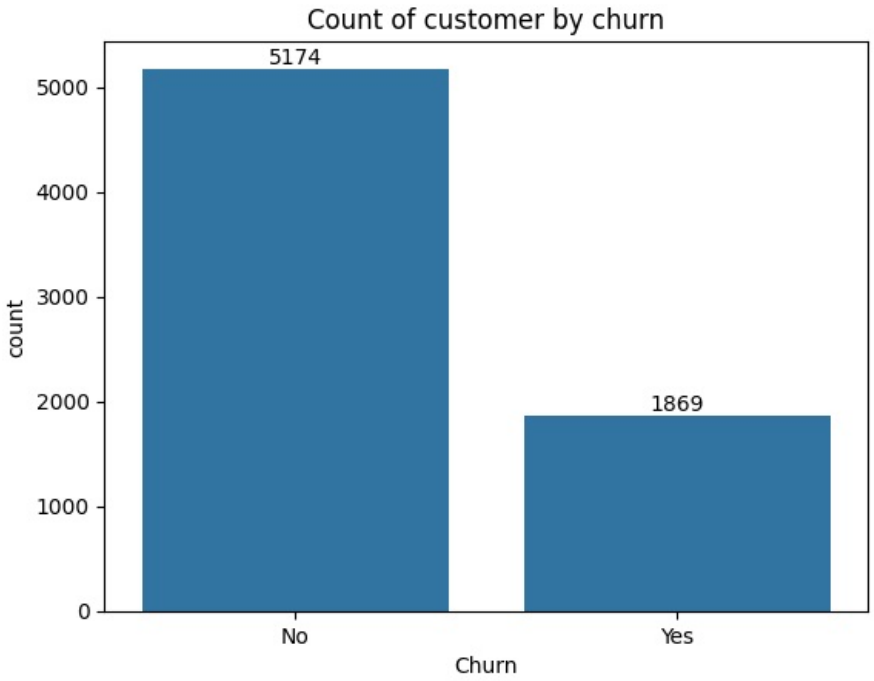
In [13]:
```
df.head()
```

Out[13]:

	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	..

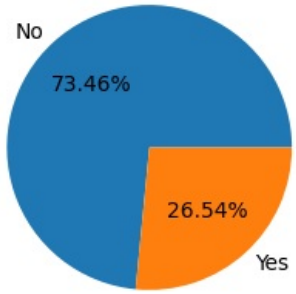
5 rows × 21 columns

In [14]:
```
ax = sns.countplot(x = 'Churn' , data = df)

ax.bar_label(ax.containers[0])
plt.title("Count of customer by churn")
plt.show()
```



In [15]:
```
plt.figure(figsize = (3,3))
gb = df.groupby("Churn").agg({'Churn':"count"})
plt.pie(gb['Churn'] , labels = gb.index, autopct = "%1.2f%%")
plt.title("percentage of customer by churn")
plt.show()
```

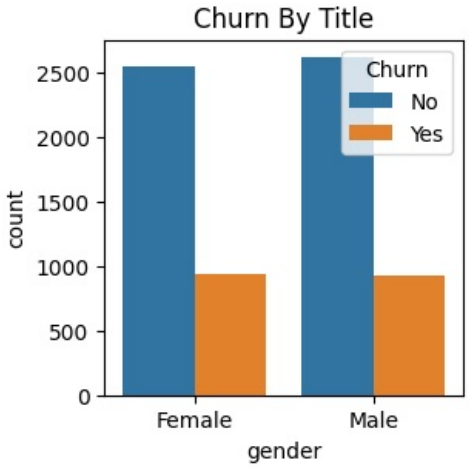
percentage of customer by churn



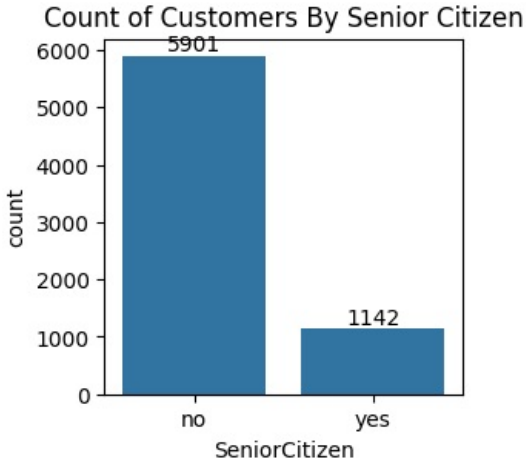
from the given pie chart we can conclude that 26.54% of our customer have churned out

now lets explore the reason behind it

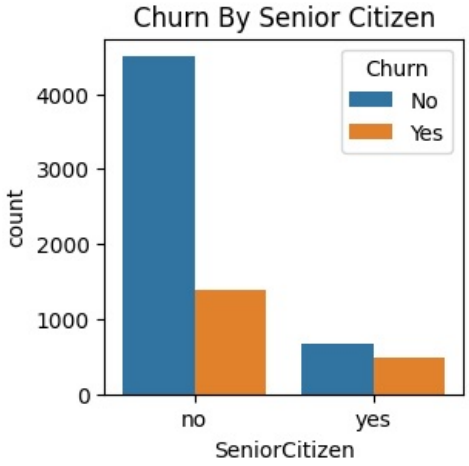
In [16]:
```
plt.figure(figsize = (3,3))
sns.countplot(x = "gender", data = df , hue = 'Churn')
plt.title("Churn By Title")
plt.show()
```



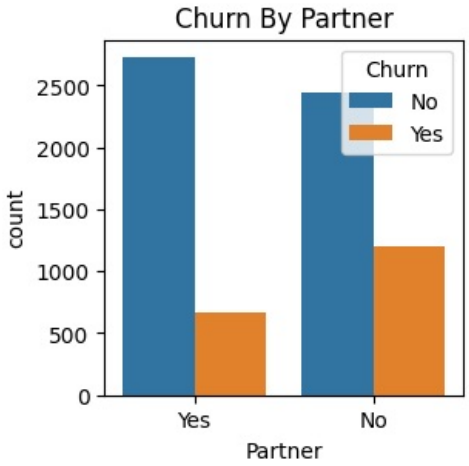
In [17]:
```
plt.figure(figsize = (3,3))
ax = sns.countplot(x = "SeniorCitizen", data = df)
ax.bar_label(ax.containers[0])
plt.title("Count of Customers By Senior Citizen")
plt.show()
```



In [18]:
```
plt.figure(figsize = (3,3))
sns.countplot(x = "SeniorCitizen", data = df , hue = 'Churn')
plt.title("Churn By Senior Citizen")
plt.show()
```



In [19]:
```
plt.figure(figsize = (3,3))
sns.countplot(x = "Partner", data = df , hue = 'Churn')
plt.title("Churn By Partner")
plt.show()
```

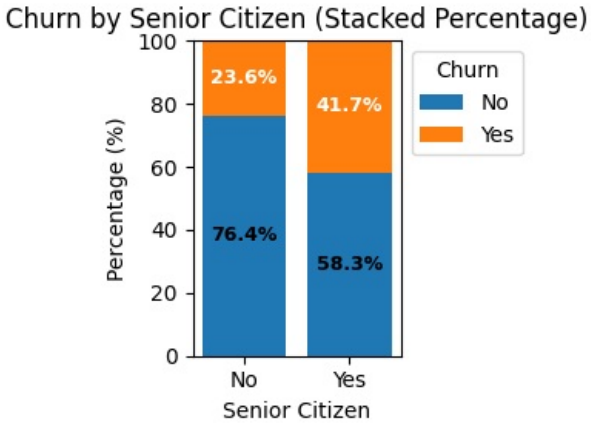


In [20]:
```
print(df['Churn'].value_counts())
print(df['SeniorCitizen'].value_counts())
print(df.groupby(['SeniorCitizen', 'Churn']).size())
```

```
Churn
No     5174
Yes    1869
Name: count, dtype: int64
SeniorCitizen
no     5901
yes    1142
Name: count, dtype: int64
SeniorCitizen  Churn
no             No       4508
               Yes      1393
yes            No        666
               Yes       476
dtype: int64
```

In [21]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Group and calculate percentages
counts = df.groupby(['SeniorCitizen', 'Churn']).size().unstack(fill_value=0)
percentages = counts.div(counts.sum(axis=1), axis=0) * 100

# Step 2: Define bar colors
colors = {
    'No': '#1f77b4',   # Blue
    'Yes': '#ff7f0e'   # Orange
}

# Step 3: Plot
plt.figure(figsize=(3, 3))
bottom = [0] * len(percentages)

for churn_value in percentages.columns:
    plt.bar(
        percentages.index,
        percentages[churn_value],
        bottom=bottom,
        label=churn_value,
        color=colors[churn_value]
    )
    # Update bottom for stacking
    bottom = [i + j for i, j in zip(bottom, percentages[churn_value])]

# Step 4: Add percentage labels
for i, senior in enumerate(percentages.index):
    cumulative = 0
    for churn_value in percentages.columns:
        pct = percentages.loc[senior, churn_value]
        if pct > 3:  # Show label only if >3%
            color = 'white' if cumulative + pct / 2 > 50 else 'black'
            plt.text(
                x=i,
                y=cumulative + pct / 2,
                s=f'{pct:.1f}%',
                ha='center',
                va='center',
                fontsize=9,
                color=color,
                fontweight='bold'
            )
```

```
        cumulative += pct

# Step 5: Final chart settings
plt.title('Churn by Senior Citizen (Stacked Percentage)', fontsize=12)
plt.xlabel('Senior Citizen', fontsize=10)
plt.ylabel('Percentage (%)', fontsize=10)
plt.ylim(0, 100)
plt.xticks(ticks=range(len(percentages.index)), labels=[str(x).capitalize() for x in percentages.index])
plt.legend(title='Churn' , bbox_to_anchor = (1,1))
plt.tight_layout()
plt.show()
```



✅ Step 1: Group data and calculate percentages

python Copy Edit counts = df.groupby(['SeniorCitizen', 'Churn']).size().unstack(fill_value=0) percentages = counts.div(counts.sum(axis=1), axis=0) * 100 groupby karta hai combination of SeniorCitizen and Churn.

size() counts the occurrences.

unstack() se hum Churn values ko columns bana lete hain.

div(...)*100: Row-wise division to convert count into percent of total in each SeniorCitizen group.

✅ Step 2: Define custom colors python Copy Edit colors = { 'No': '#1f77b4', # Blue 'Yes': '#ff7f0e' # Orange } Is dictionary mein bataya gaya hai ki 'No' churn ka bar blue hoga, aur 'Yes' churn ka bar orange.

✅ Step 3: Plot the bars python Copy Edit plt.figure(figsize=(5, 5)) bottom = [0] * len(percentages) # For stacking each bar

for churn_value in percentages.columns: plt.bar(percentages.index, # x-axis values → ['no', 'yes'] percentages[churn_value], # height of each segment bottom=bottom, # start from this height (for stacking) label=churn_value, # for legend color=colors[churn_value] # bar color) bottom = [i + j for i, j in zip(bottom, percentages[churn_value])] bottom keeps track of where the next bar segment should start (for stacking).

zip helps us update the bottom for each bar as we stack them.

plt.bar() draws each segment of the stacked bar.

✅ Step 4: Add % labels to bars python Copy Edit for i, senior in enumerate(percentages.index): cumulative = 0 for churn_value in percentages.columns: pct = percentages.loc[senior, churn_value] if pct > 3: # Skip small percentages color = 'white' if cumulative + pct / 2 > 50 else 'black' plt.text(x=i, y=cumulative + pct / 2, # Middle of current bar segment s=f'{pct:.1f}%', ha='center', va='center', fontsize=9, color=color, fontweight='bold') cumulative += pct Yeh loop har stacked part ke beech mein exact center par % label lagata hai.

cumulative helps us figure out where to place the label vertically.

Color is chosen based on brightness — dark bar → white text, light bar → black text.

✅ Step 5: Final chart settings python Copy Edit plt.title('Churn by Senior Citizen (Stacked Percentage)', fontsize=12) plt.xlabel('Senior Citizen', fontsize=10) plt.ylabel('Percentage (%)', fontsize=10) plt.ylim(0, 100) plt.xticks(ticks=range(len(percentages.index)), labels=[str(x).capitalize() for x in percentages.index]) plt.legend(title='Churn') plt.tight_layout() plt.show() Set title, labels, y-limit = 100%, and format x-tick labels ('no' → 'No').

plt.tight_layout() ensures spacing is clean.

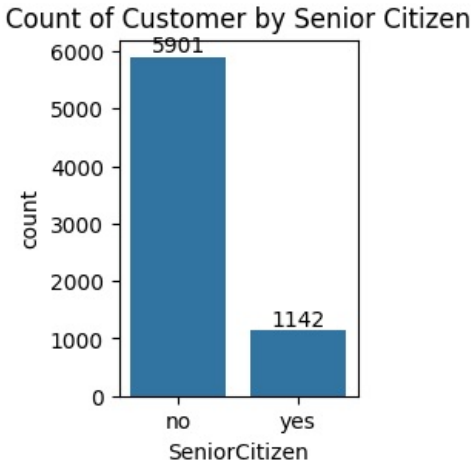
plt.show() displays the graph.

Summary Yeh chart tumhe kitne % log churn kiye ya nahi kiye batata hai per SeniorCitizen group.

Har bar ke andar likha % directly bolta hai kitne % churn huye ya nahi huye.

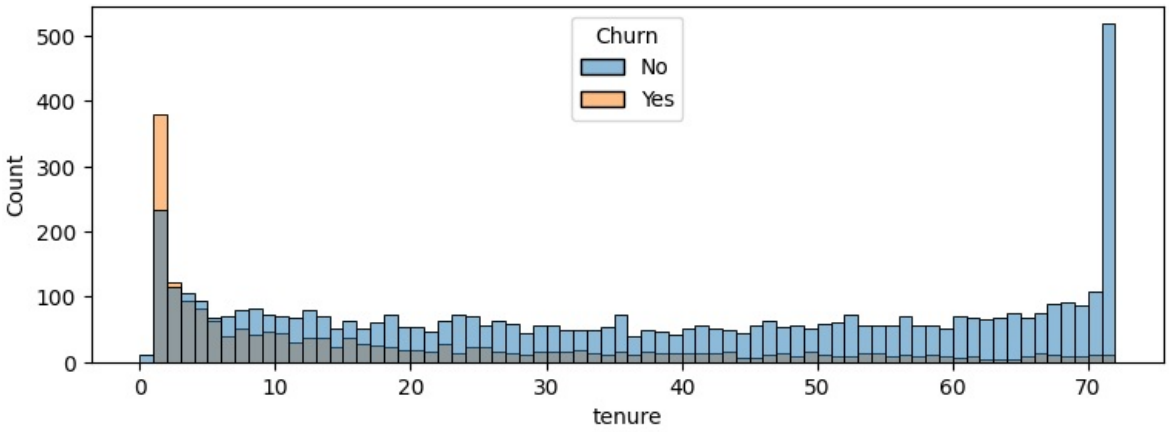
Visually compare kar sakte ho ki seniors ka churn rate zyada hai ya non-seniors ka.

In [22]:
```
plt.figure(figsize = (2,3))
ax = sns.countplot(x = "SeniorCitizen", data = df)
ax.bar_label(ax.containers[0])
plt.title("Count of Customer by Senior Citizen")
plt.show()
```



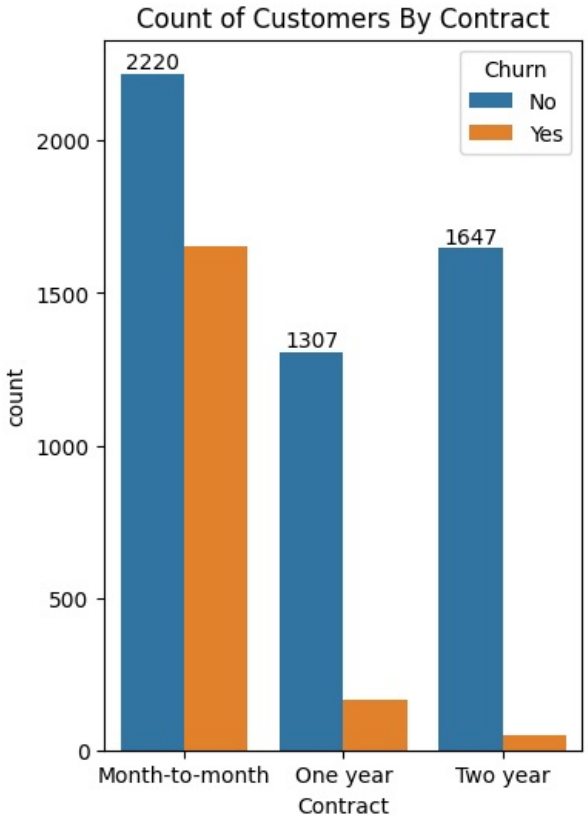
compartive a greated no. of people in senior citizen have churned out

In [23]:
```
plt.figure(figsize = (9,3))
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 who have used our services have churned

In [24]:
```
plt.figure(figsize = (4,6))
ax = sns.countplot(x = "Contract", data = df , hue = "Churn")
ax.bar_label(ax.containers[0])
plt.title("Count of Customers By Contract")
plt.show()
```

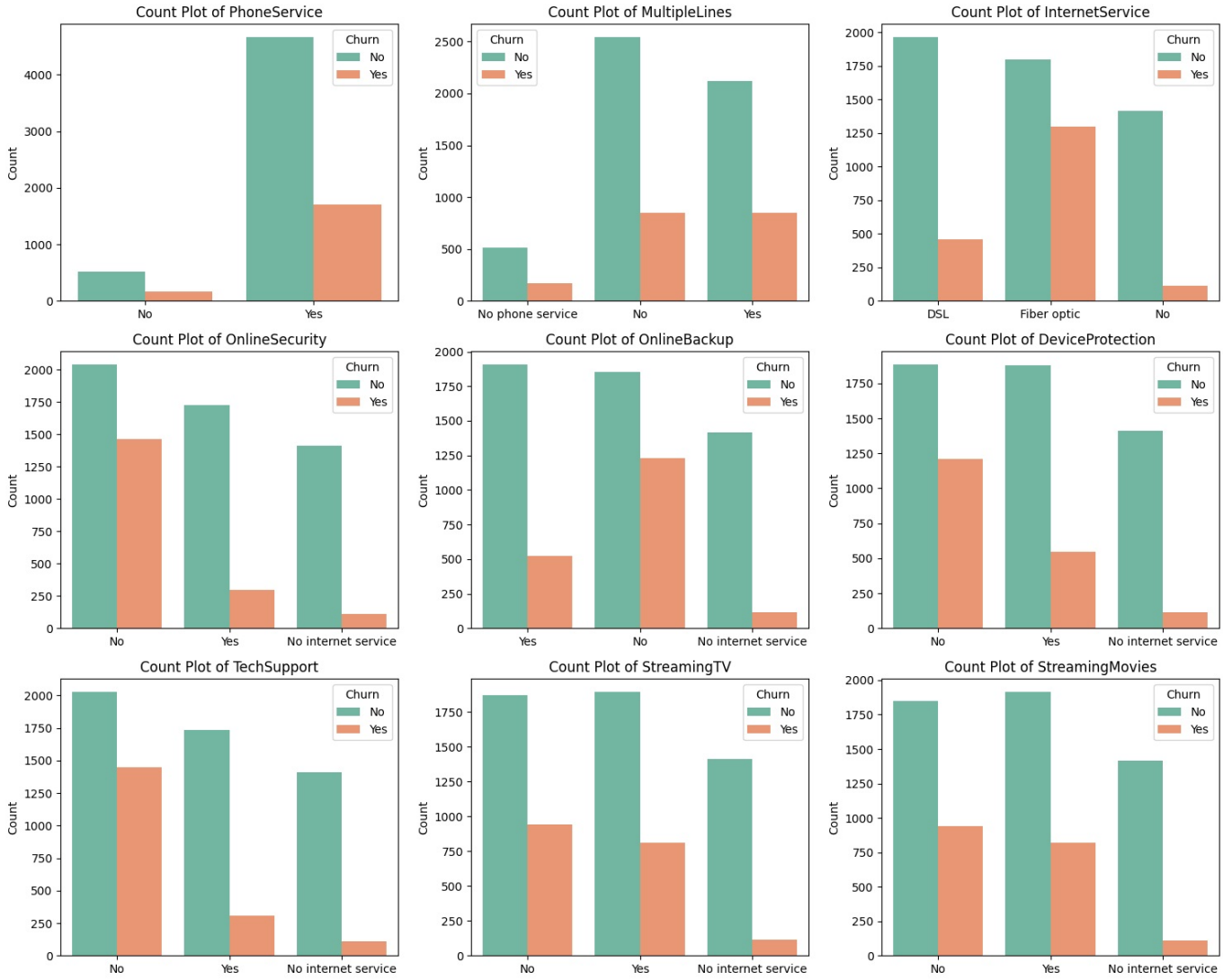


people from the above chart we can conclude that people who have 1 to 1 plan is likely to be churned out then the people who have 1 to many plans

In [25]:
```
df.columns.values
```

Out[25]:
```
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 [26]:
```
import matplotlib.pyplot as plt
import seaborn as sns

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

n_cols = 3
n_rows = (len(cols) + n_cols - 1) // n_cols

fig, axes = plt.subplots(nrows=n_rows, ncols=n_cols, figsize=(5 * n_cols, 4 * n_rows))
axes = axes.flatten()

for i, col in enumerate(cols):
    sns.countplot(data=df, x=col, ax=axes[i], hue=df["Churn"], palette="Set2")
    axes[i].set_title(f'Count Plot of {col}')
    axes[i].set_xlabel('')
    axes[i].set_ylabel('Count')

# Hide any empty subplots
for j in range(len(cols), len(axes)):
    fig.delaxes(axes[j])

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



Customers with Fiber optic internet and those without security or backup services tend to churn more. Features like OnlineSecurity, OnlineBackup, and DeviceProtection are linked to better retention. Customers without internet or phone services show the lowest churn. Having multiple lines slightly increases the chance of churn. Overall, add-on services seem to reduce customer churn.

In [29]:

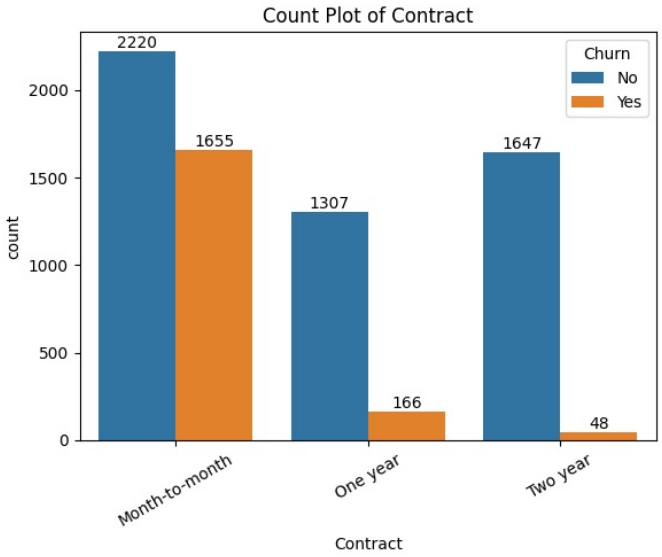
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load your dataset
df = pd.read_csv("your_dataset.csv")  # replace with the actual filename or path

# Now your plotting code
cols = ['Contract', 'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges', 'TotalCharges']
n_cols = 2
n_rows = (len(cols) + n_cols - 1) // n_cols

fig, axes = plt.subplots(nrows=n_rows, ncols=n_cols, figsize=(6 * n_cols, 5 * n_rows))
axes = axes.flatten()

for i, col in enumerate(cols):
    ax = sns.countplot(data=df, x=col, hue='Churn', ax=axes[i])
    for container in ax.containers:
        ax.bar_label(container)
    ax.set_title(f'Count Plot of {col}')
    ax.tick_params(axis='x', rotation=30)

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

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

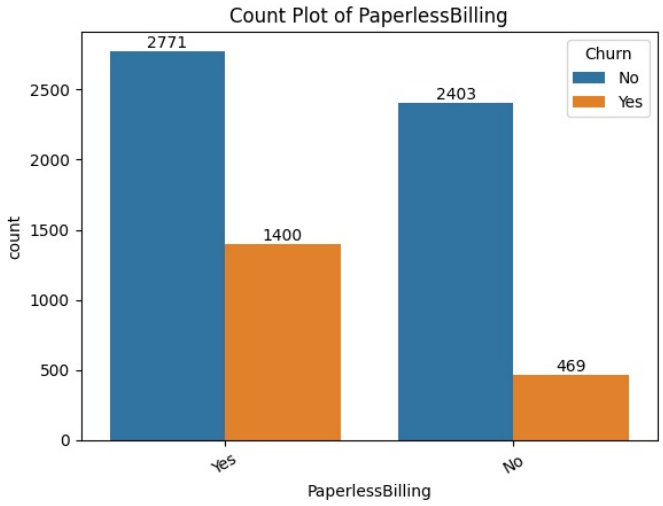
```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[29], line 6
      3 import seaborn as sns
      5 # Load your dataset
----> 6 df = pd.read_csv(                  )  # replace with the actual filename or path
      8 # Now your plotting code
      9 cols = ['Contract', 'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges', 'TotalCharges']

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\io\parsers\readers.py:1026, in read_csv(
filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, dtype, engine, converters, true_values, f
alse_values, skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na, na_filter, verbose, skip
_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser, date_format, dayfirst, cache_dates
, iterator, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, doublequote, escapec
har, comment, encoding, encoding_errors, dialect, on_bad_lines, delim_whitespace, low_memory, memory_map, float_
precision, storage_options, dtype_backend)
   1013 kwds_defaults = _refine_defaults_read(
   1014     dialect,
   1015     delimiter,
   (...)   1022     dtype_backend=dtype_backend,
   1023 )
   1024 kwds.update(kwds_defaults)
-> 1026 return _read(filepath_or_buffer, kwds)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\io\parsers\readers.py:620, in _read(file
path_or_buffer, kwds)
    617 _validate_names(kwds.get("names", None))
    619 # Create the parser.
--> 620 parser = TextFileReader(filepath_or_buffer, **kwds)
    622 if chunksize or iterator:
    623     return parser

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\io\parsers\readers.py:1620, in TextFileR
eader.__init__(self, f, engine, **kwds)
   1617     self.options["has_index_names"] = kwds["has_index_names"]
   1619 self.handles: IOHandles | None = None
-> 1620 self._engine = self._make_engine(f, self.engine)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\io\parsers\readers.py:1880, in TextFileR
eader._make_engine(self, f, engine)
   1878     if "b" not in mode:
   1879         mode += "b"
-> 1880 self.handles = get_handle(
   1881     f,
   1882     mode,
   1883     encoding=self.options.get(          , None),
   1884     compression=self.options.get(             , None),
   1885     memory_map=self.options.get(            , False),
   1886     is_text=is_text,
   1887     errors=self.options.get(                ,         ),
   1888     storage_options=self.options.get(                 , None),
   1889 )
   1890 assert self.handles is not None
   1891 f = self.handles.handle

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\io\common.py:873, in get_handle(path_or_
buf, mode, encoding, compression, memory_map, is_text, errors, storage_options)
    868 elif isinstance(handle, str):
    869     # Check whether the filename is to be opened in binary mode.
    870     # Binary mode does not support 'encoding' and 'newline'.
    871     if ioargs.encoding and "b" not in ioargs.mode:
    872         # Encoding
--> 873         handle = open(
    874             handle,
    875             ioargs.mode,
    876             encoding=ioargs.encoding,
    877             errors=errors,
    878             newline=  ,
    879         )
    880     else:
    881         # Binary mode
    882         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory: 'your_dataset.csv'
```

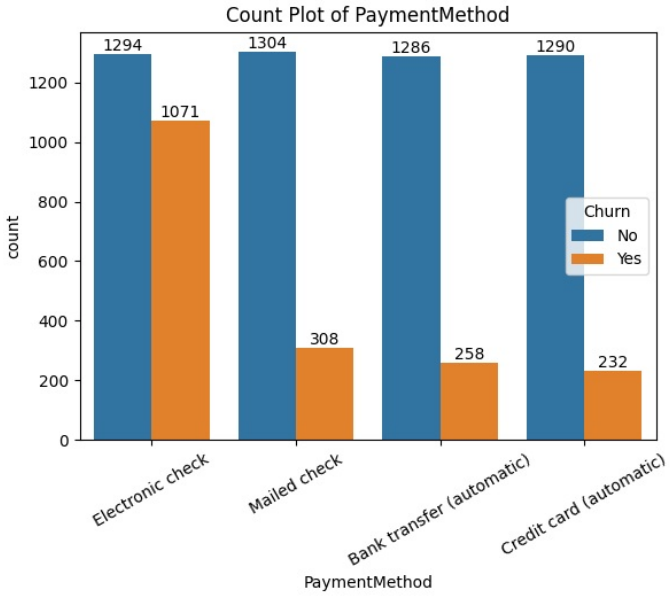
In [28]:

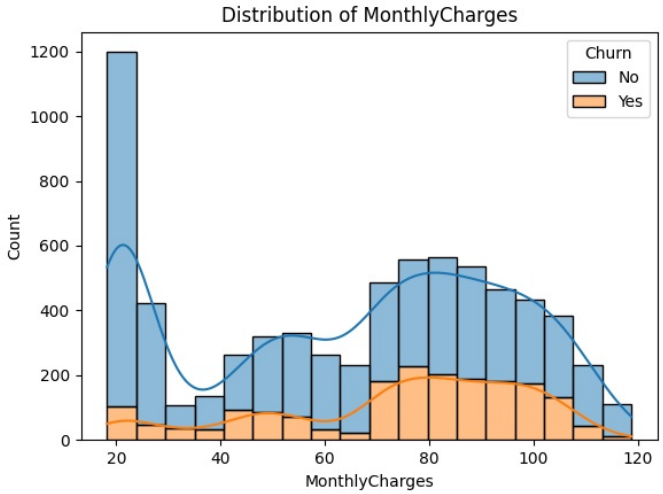
```
import matplotlib.pyplot as plt
import seaborn as sns

# Separate categorical and numerical columns
cat_cols = ['Contract', 'PaperlessBilling', 'PaymentMethod']
num_cols = ['MonthlyCharges', 'TotalCharges']

# Total plots
```

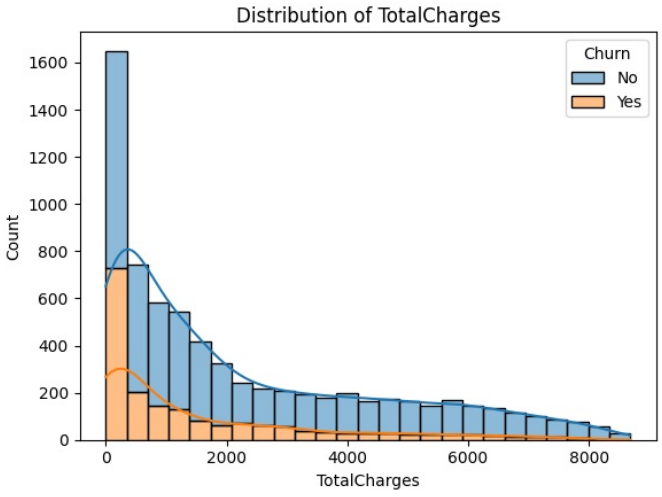
```
cols = cat_cols + num_cols
n_cols = 2
n_rows = (len(cols) + n_cols - 1) // n_cols
fig, axes = plt.subplots(nrows=n_rows, ncols=n_cols, figsize=(6 * n_cols, 5 * n_rows))
axes = axes.flatten()

# Plotting
for i, col in enumerate(cols):
    ax = axes[i]

    if col in cat_cols:
        plot = sns.countplot(data=df, x=col, hue='Churn', ax=ax)
        for container in plot.containers:
            plot.bar_label(container)
        ax.set_title(f'Count Plot of {col}')
        ax.tick_params(axis='x', rotation=30)

    else:  # Numerical columns
        sns.histplot(data=df, x=col, hue='Churn', kde=True, multiple='stack', ax=ax)
        ax.set_title(f'Distribution of {col}')

# Delete extra axes if any
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

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


Count Plot of Contract
Churn
No
Yes
2220
1655
1307
166
1647
48
count
Month-to-month
One year
Two year
Contract


Count Plot of PaperlessBilling
Churn
No
Yes
2771
1400
2403
469
count
Yes
No
PaperlessBilling


Count Plot of PaymentMethod
Churn
No
Yes
1294
1071
1304
308
1286
258
1290
232
count
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
PaymentMethod


Distribution of MonthlyCharges
Churn
No
Yes
Count
MonthlyCharges


Distribution of TotalCharges
Churn
No
Yes
Count
TotalCharges

In [ ]: