Notebook

May 7, 2025

'#e1d3c1', '#c64343' color

[1]:
```
#import csv file
from google.colab import files
uploded= files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving Telco-Customer-Churn.csv to Telco-Customer-Churn.csv
```

[2]:
```
import pandas as pd
#read csv
df= pd.read_csv("Telco-Customer-Churn.csv")
```

[3]:
```
df.head()
```

[3]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	…	DeviceProtection
0	7590-VHVEG	Female	0	Yes	No	1	No	…	No
1	5575-GNVDE	Male	0	No	No	34	Yes	…	Yes
2	3668-QPYBK	Male	0	No	No	2	Yes	…	No
3	7795-CFOCW	Male	0	No	No	45	No	…	Yes
4	9237-HQITU	Female	0	No	No	2	Yes	…	No

	MultipleLines	InternetService	OnlineSecurity	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	No phone service	DSL	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
1	No	DSL	Yes	No	No	No	One year	No				
2	No	DSL	Yes	No	No	No	Month-to-month	Yes				
3	No phone service	DSL	Yes	Yes	No	No	One year	No				
4	No	Fiber optic	No	No	No	No	Month-to-month	Yes				

```
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]
```

[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   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

[5]:
```
#first replace blank speace with zeros
df["TotalCharges"]= df["TotalCharges"].replace(" ","0")

# converts it into float beacuse payment will be float values
df["TotalCharges"]= df["TotalCharges"].astype("float")
```

[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   float64
 20  Churn             7043 non-null   object
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

[7]:
```
df.head()
```

[7]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	…	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling
0	7590-VHVEG	Female	0	Yes	No	1	No	…	No	No	No	No	Month-to-month	Yes
1	5575-GNVDE	Male	0	No	No	34	Yes	…	Yes	No	No	No	One year	No
2	3668-QPYBK	Male	0	No	No	2	Yes	…	No	No	No	No	Month-to-month	Yes
3	7795-CFOCW	Male	0	No	No	45	No	…	Yes	No	No	No	One year	No
4	9237-HQITU	Female	0	No	No	2	Yes	…	No	No	No	No	Month-to-month	Yes

```
3         Yes          No               No        One year               No
4          No          No               No  Month-to-month              Yes

               PaymentMethod MonthlyCharges  TotalCharges  Churn
0           Electronic check          29.85         29.85     No
1               Mailed check          56.95       1889.50     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]
```

[8]:
```
#check null value
df.isnull().sum().sum()
```

[8]:
```
np.int64(0)
```

[9]:
```
#see some descriptive analysis
df.describe()
```

[9]:
```
       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
```

[10]:
```
# cheack duplicated
df.duplicated().sum()
```

[10]:
```
np.int64(0)
```

[11]:
```
#create a function for converting numerical data as "1= yes"or "0= no" for␣
 ↪"SeniorCitizen"
def senior(x):
  if x==1:
    return "Yes"
  else:
    return "No"
```

[12]:
```
df["SeniorCitizen"]= df["SeniorCitizen"].apply(senior)
```

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

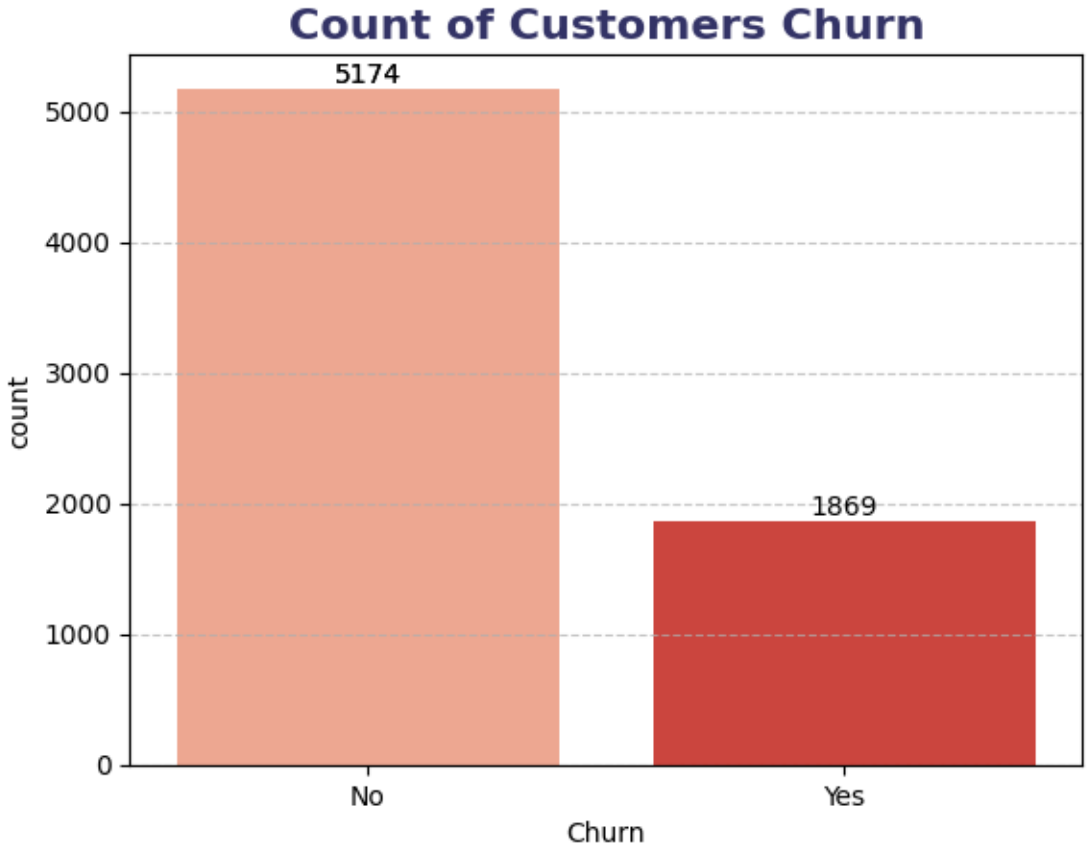
[13]:
```
   customerID  gender SeniorCitizen Partner Dependents  tenure PhoneService  \
0  7590-VHVEG  Female             No     Yes         No       1           No
1  5575-GNVDE    Male             No      No         No      34          Yes
2  3668-QPYBK    Male             No      No         No       2          Yes
3  7795-CFOCW    Male             No      No         No      45           No
4  9237-HQITU  Female             No      No         No       2          Yes

      MultipleLines InternetService OnlineSecurity  … DeviceProtection  \
0  No phone service             DSL             No  …               No
1                No             DSL            Yes  …              Yes
2                No             DSL            Yes  …               No
3  No phone service             DSL            Yes  …              Yes
4                No     Fiber optic             No  …               No

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

               PaymentMethod MonthlyCharges  TotalCharges Churn
0           Electronic check          29.85         29.85    No
1               Mailed check          56.95       1889.50    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]
```

[14]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[15]:
```
#  ammount of churn "yes" and "no"
#  count plot
count_plot = sns.countplot( x="Churn", data=df, palette="Reds")

for container in count_plot.containers:
    count_plot.bar_label(container)

count_plot.bar_label(count_plot.containers[0])
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.title("Count of Customers Churn", color="#333366", fontsize=16,␣
 ↪fontweight="bold")
```

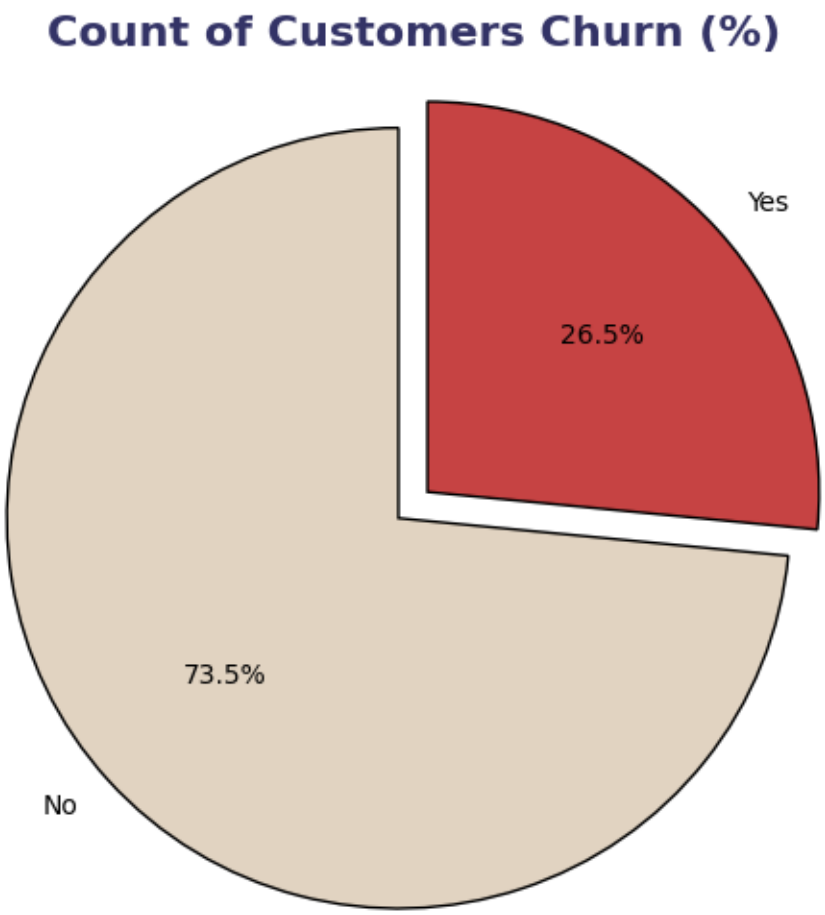
```
plt.show()
```

```
<ipython-input-15-a9a52f4eb407>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  count_plot = sns.countplot( x="Churn", data=df, palette="Reds")
```

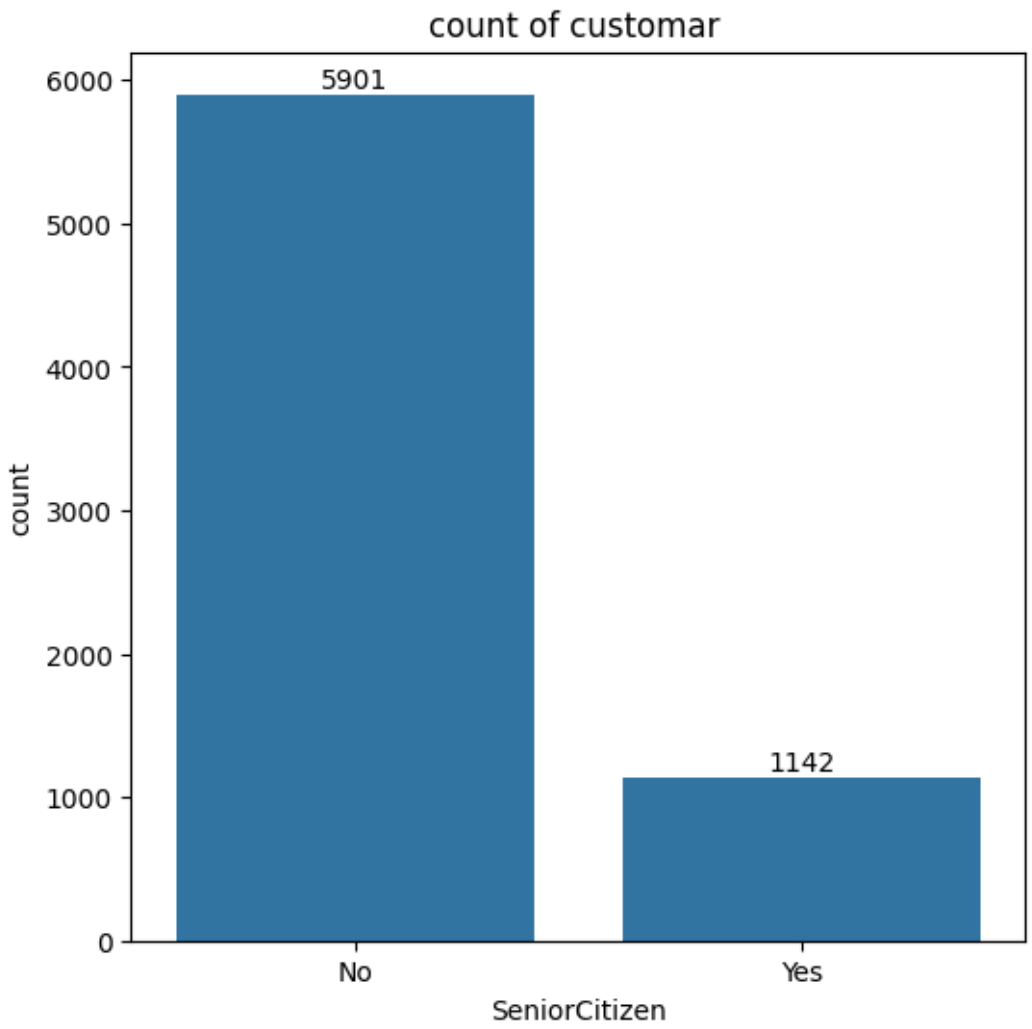


[16]:
```
group_by = df.groupby("Churn").agg({"Churn": "count"})

explode = [0, 0.1]

# make pie chart
plt.figure(figsize=(6, 6))
plt.pie( group_by["Churn"],labels=group_by.index,autopct="%1.
 ↪1f%%",startangle=90,colors=["#e1d3c1", "#c64343"],wedgeprops={'edgecolor':␣
 ↪'black'},explode=explode) # color= "#e1d3c1", "#c64343"
```

```
plt.title("Count of Customers Churn (%)",color="#333366", fontsize=16,␣
  ↪fontweight="bold")
plt.axis("equal")
plt.show()
```

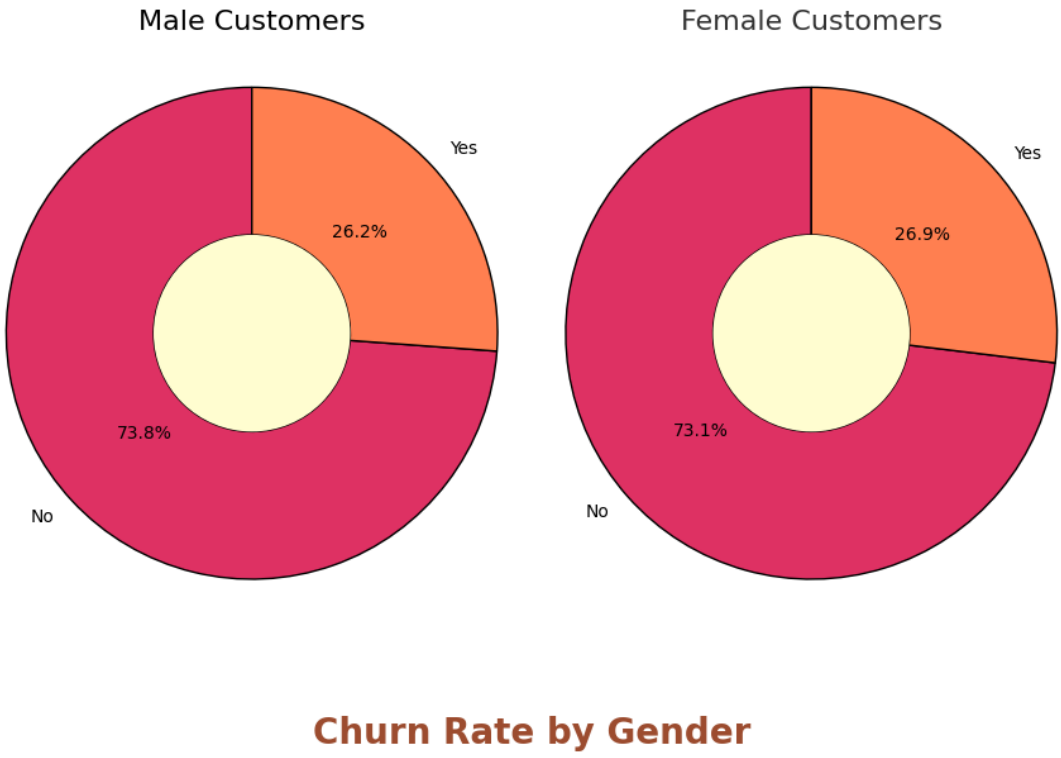


[17]:
```
plt.figure(figsize=(6,6))
ax=sns.countplot(x= "SeniorCitizen", data= df)
ax.bar_label(ax.containers[0])
plt.title("count of customar ")
plt.show()
```



[18]:
```
# Separate by gender
male_churn = df[df['gender'] == 'Male']['Churn'].value_counts()
female_churn = df[df['gender'] == 'Female']['Churn'].value_counts()

#  Color Combination
colors = ["#DE3163", "#FF7F50"]

# Create Donut Charts
fig, axes = plt.subplots(1, 2, figsize=(9, 7))

# Donut chart for Male
axes[0].pie(male_churn, labels=male_churn.index, autopct='%1.
 ↪1f%%',startangle=90, colors=colors, wedgeprops={'edgecolor': 'black',␣
 ↪'width': 0.6},radius=1)
```

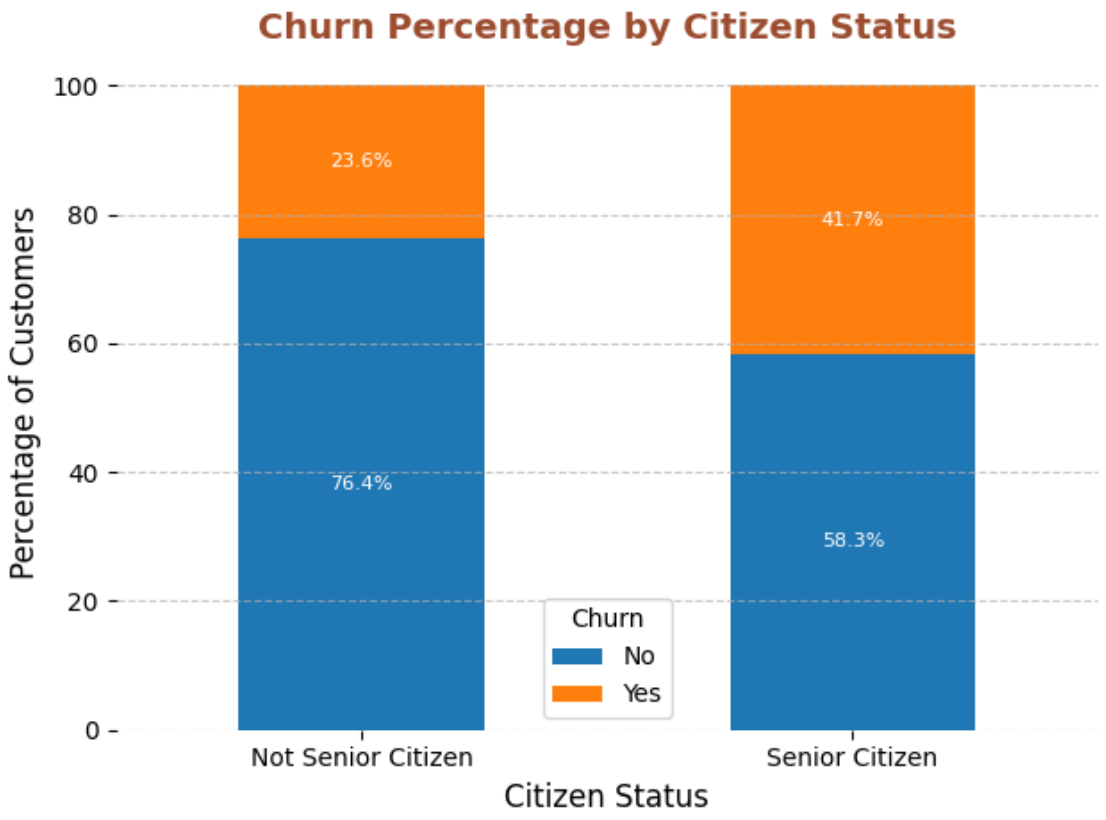
```
centre_circle_male = plt.Circle((0, 0), 0.4, fc='#FFFDD0')
axes[0].add_artist(centre_circle_male)
axes[0].set_title('Male Customers', fontsize=16)
axes[0].axis('equal')

# Donut chart for Female
axes[1].pie(female_churn, labels=female_churn.index, autopct='%1.1f%%',
            startangle=90, colors=colors, wedgeprops={'edgecolor': 'black',␣
 ↪'width': 0.6},
            radius=1)
centre_circle_female = plt.Circle((0, 0), 0.4, fc='#FFFDD0')
axes[1].add_artist(centre_circle_female)
axes[1].set_title('Female Customers', fontsize=16,color="#333333" )
axes[1].axis('equal')

fig.suptitle('Churn Rate by Gender', fontsize=20, y=0.02, ha='center',␣
 ↪color="#9c4c2f", fontweight="bold")
plt.tight_layout(rect=[0, 0.1, 1, 0.9])
plt.show()
```



[19]:
```
#  SeniorCitizen and Churn
senior_churn_ct = pd.crosstab(df['SeniorCitizen'], df['Churn'])
```

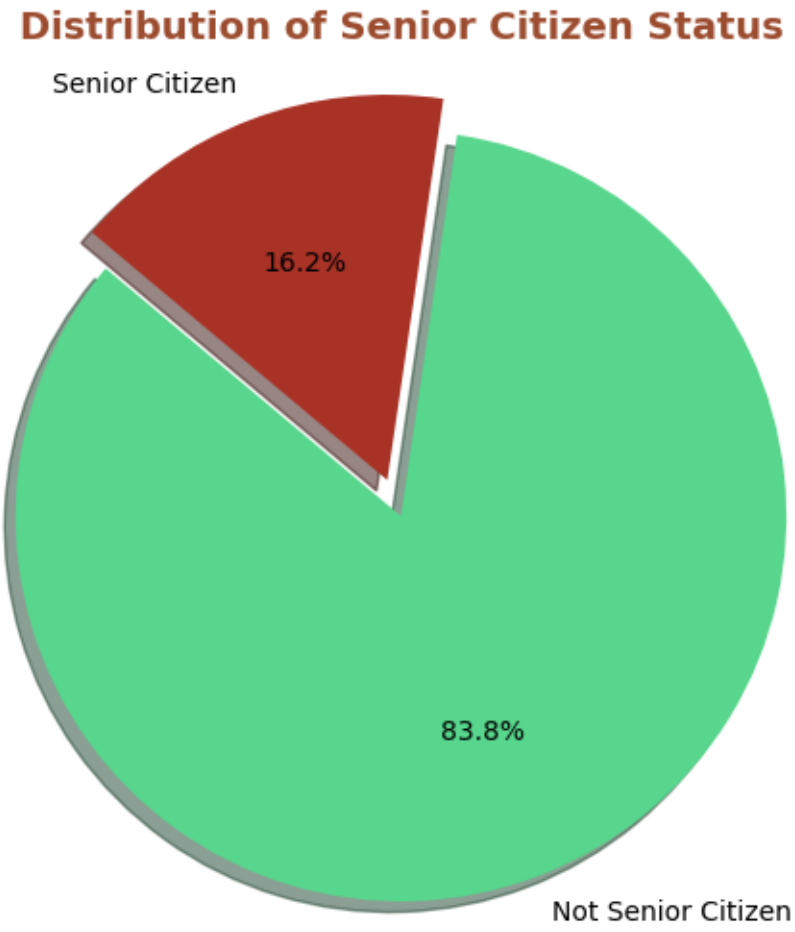
```
# the percentage of  Churn status within each SeniorCitizen group
senior_churn_pct = senior_churn_ct.apply(lambda x: x*100/sum(x), axis=1)

# figure size
plt.figure(figsize=(10, 10))

#  stacked bar chart
senior_churn_pct.plot(kind='bar', stacked=True, color=['#1f77b4', '#ff7f0e'])


plt.title("Churn Percentage by Citizen Status", fontsize=14, color="#9c4c2f",␣
 ↪fontweight="bold")
plt.xlabel("Citizen Status", fontsize=12)
plt.ylabel("Percentage of Customers", fontsize=12)
plt.xticks([0, 1], ['Not Senior Citizen', 'Senior Citizen'], rotation=0)
plt.legend(title="Churn", labels=["No", "Yes"], fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
sns.despine(left=True, bottom=True)

# percentage labels
for i in range(senior_churn_pct.shape[0]):
    total = senior_churn_pct.iloc[i].sum()
    for j in range(senior_churn_pct.shape[1]):
        percentage = senior_churn_pct.iloc[i, j]
        plt.text(i, senior_churn_pct.iloc[i, :j].sum() + percentage/2,␣
 ↪f'{percentage:.1f}%',
                 ha='center', va='center', color='white', fontsize=8)

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

```
<Figure size 1000x1000 with 0 Axes>
```



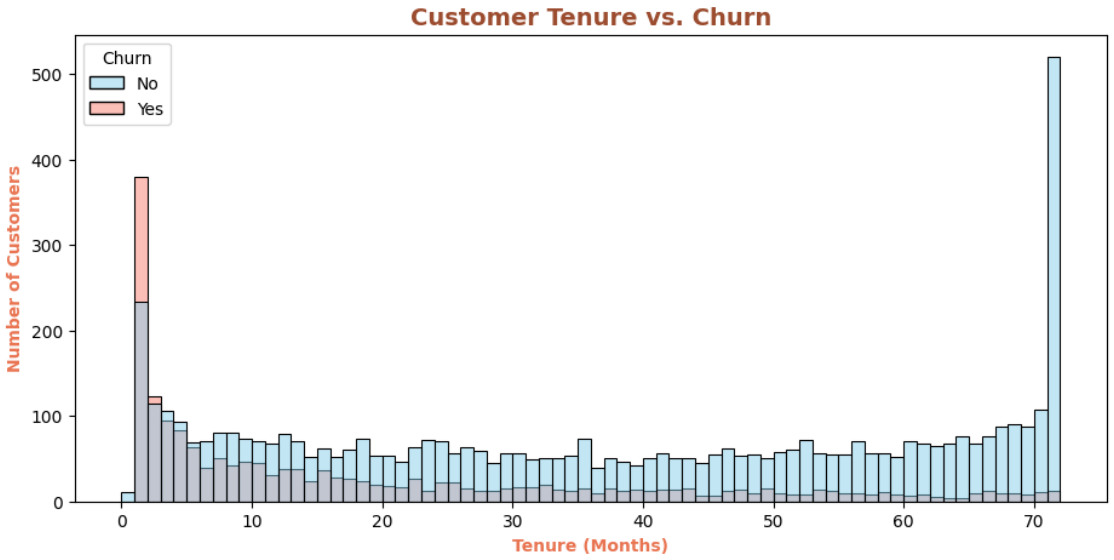
[20]:
```
import matplotlib.pyplot as plt

senior_citizen_counts = df['SeniorCitizen'].value_counts()

# Create labels
labels = ['Not Senior Citizen', 'Senior Citizen']

# Get the sizes
sizes = senior_citizen_counts.values

# colors  codes
colors = ['#58d68d', '#a93226']


# Explode
explode = (0, 0.1)

# pie chart
plt.figure(figsize=(6, 6))
```

```
plt.pie(sizes,explode=explode,labels=labels,colors=colors,autopct='%1.
  ↪1f%%',shadow=True,startangle=140)

plt.title('Distribution of Senior Citizen Status', fontsize=14,␣
  ↪color="#9c4c2f", fontweight="bold")
plt.axis('equal')
plt.show()
```

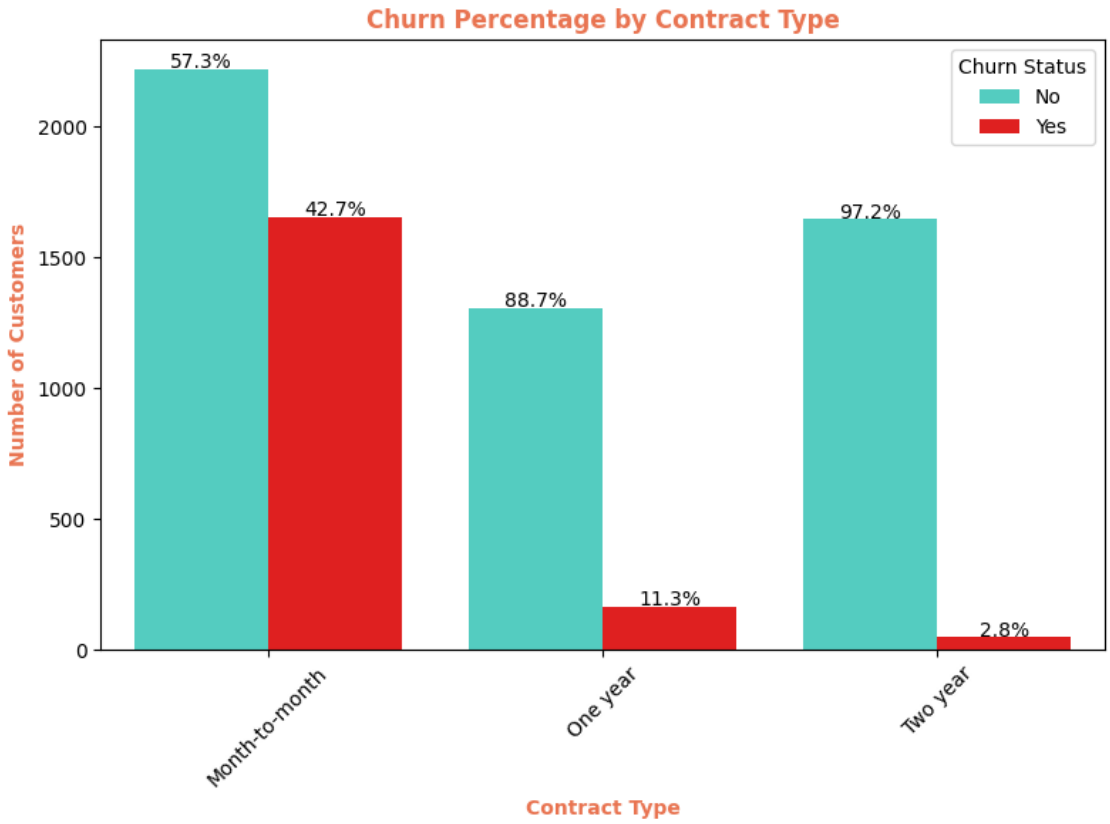


[21]:
```
# according to tenure
plt.figure(figsize=(11, 5))
sns.histplot(x="tenure", data=df, bins=72, hue="Churn", palette=["skyblue",␣
  ↪"salmon"])
plt.title("Customer Tenure vs. Churn", fontsize=14, color="#9c4c2f",␣
  ↪fontweight="bold")
plt.xlabel("Tenure (Months)", color="#E97451",fontweight="bold")
plt.ylabel("Number of Customers", color="#E97451",fontweight="bold")
```

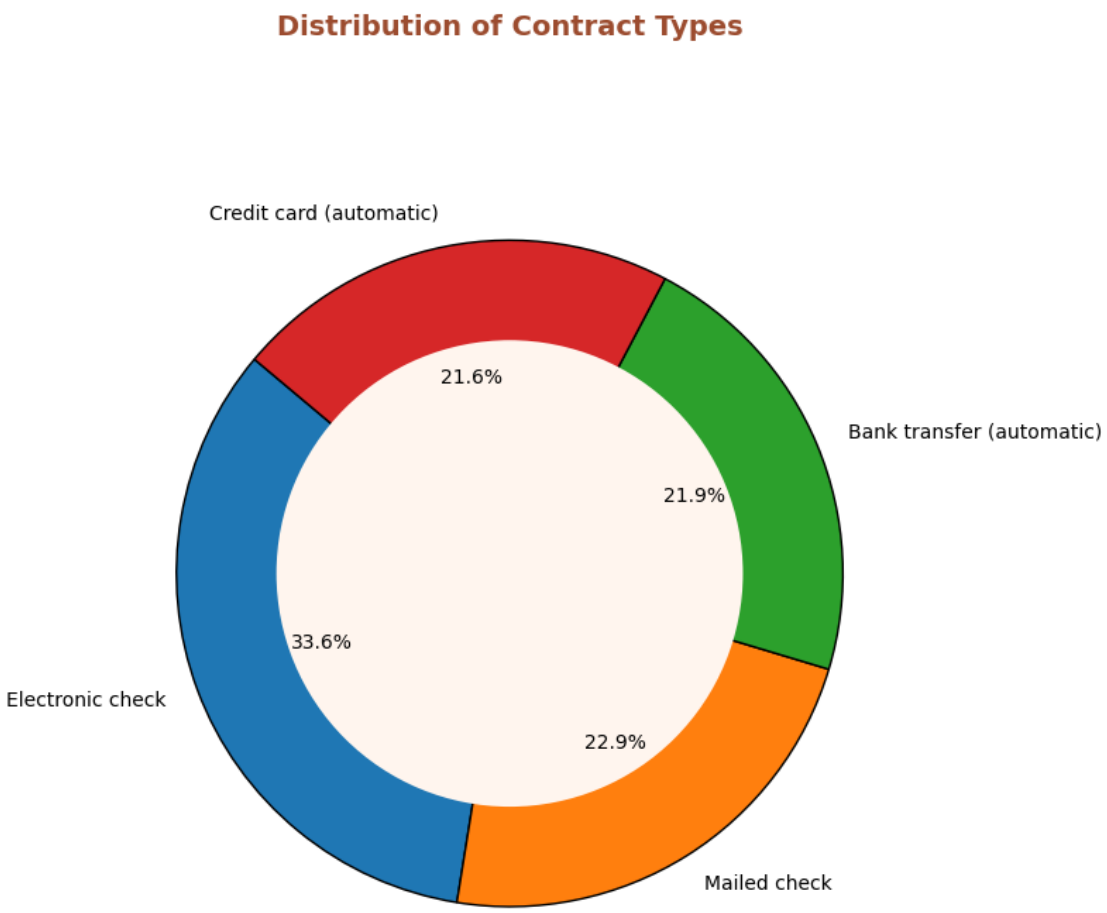
```
plt.show()
```



[22]:
```
#for contract
plt.figure(figsize=(8, 6))
ax = sns.countplot(x='Contract', hue='Churn', data=df, palette=['#40E0D0',␣
↪'red'])
plt.title('Churn Percentage by Contract Type',color="#E97451",fontweight="bold")
plt.xlabel('Contract Type',color="#E97451",fontweight="bold")
plt.ylabel('Number of Customers',color="#E97451",fontweight="bold")
plt.xticks(rotation=45)
plt.legend(title='Churn Status')

total_per_contract = df['Contract'].value_counts()

for p in ax.patches:
    height = p.get_height()
    if height > 0:
        x = p.get_x() + p.get_width() / 2
        x_index = int(round(x))
        if 0 <= x_index < len(ax.get_xticklabels()):
            contract_group = ax.get_xticklabels()[x_index].get_text()
            total = total_per_contract[contract_group]
            percentage = f'{100 * height / total:.1f}%'
            y = height + 3
            ax.text(x, y, percentage, ha='center', fontsize=10)

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

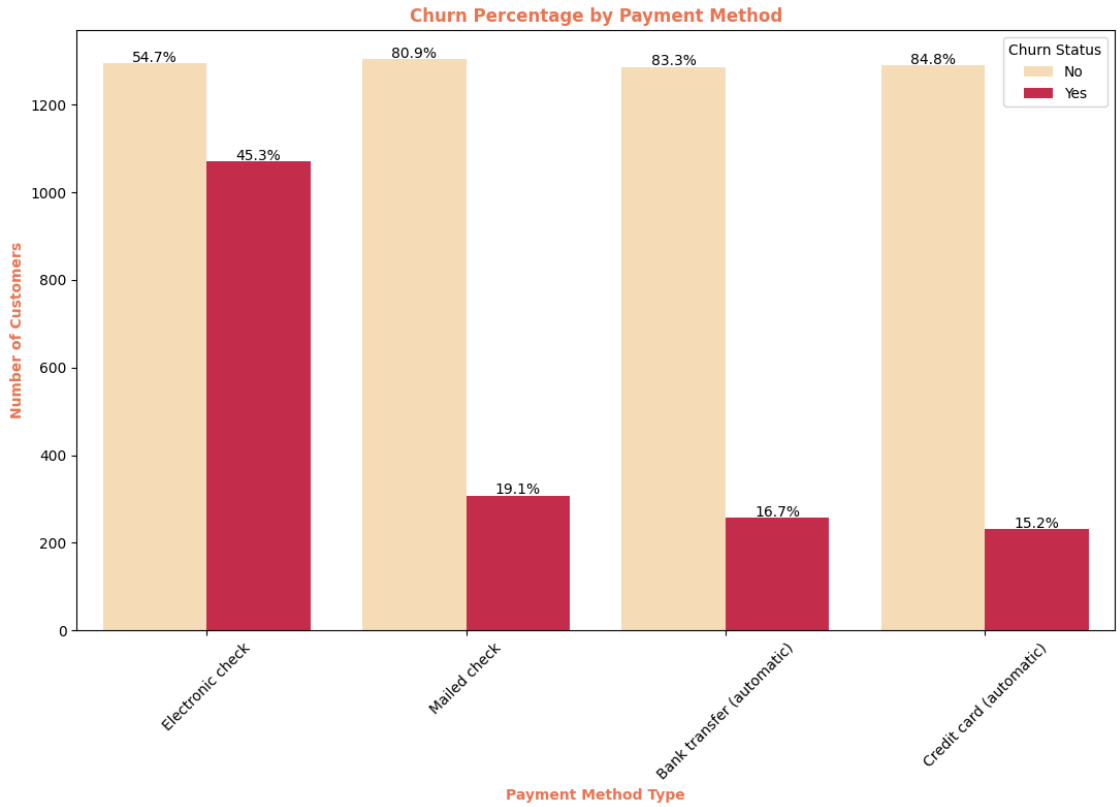


[24]:
```
PaymentMethod = df['PaymentMethod'].value_counts()
labels = PaymentMethod.index
sizes = PaymentMethod.values

# donut chart
plt.figure(figsize=(8, 8))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140,␣
 ↪wedgeprops={'edgecolor': 'black'})

#  hole in the center to make it donut chart
center_circle = plt.Circle((0, 0), 0.70, fc='#FFF5EE')
fig = plt.gcf()
fig.gca().add_artist(center_circle)

plt.title('Distribution of Contract Types', fontsize=14, color="#9c4c2f",␣
 ↪fontweight="bold")
plt.axis('equal')
plt.tight_layout()
plt.show()
```



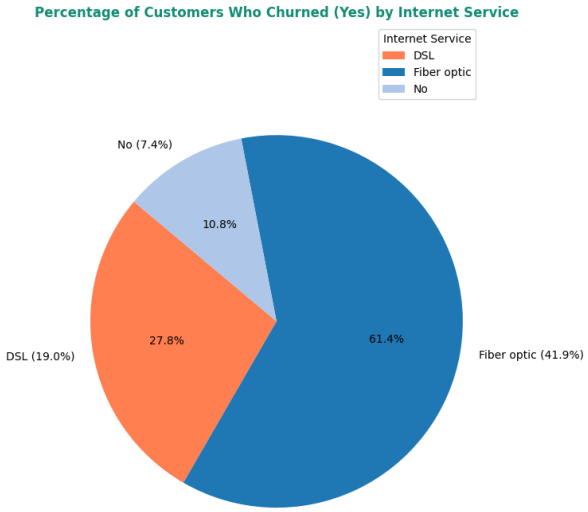
[25]:
```
#for PaymentMethod
plt.figure(figsize=(11, 8))
ax = sns.countplot(x='PaymentMethod', hue='Churn', data=df, palette=['#FFDEAD',␣
↪'#DC143C'])
plt.title('Churn Percentage by Payment␣
↪Method',color="#E97451",fontweight="bold")
plt.xlabel('Payment Method Type',color="#E97451",fontweight="bold")
plt.ylabel('Number of Customers',color="#E97451",fontweight="bold")
plt.xticks(rotation=45)
plt.legend(title='Churn Status')

total_per_contract = df['PaymentMethod'].value_counts()
```

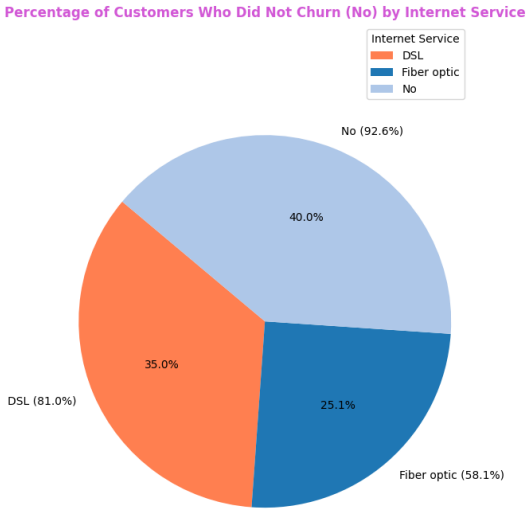
```
for p in ax.patches:
    height = p.get_height()
    if height > 0:
        x = p.get_x() + p.get_width() / 2
        x_index = int(round(x))
        if 0 <= x_index < len(ax.get_xticklabels()):
            contract_group = ax.get_xticklabels()[x_index].get_text()
            total = total_per_contract[contract_group]
            percentage = f'{100 * height / total:.1f}%'
            y = height + 3
            ax.text(x, y, percentage, ha='center', fontsize=10)

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



[26]:
```
# churn percentages for each Internet Service

churn_percentages = df.groupby('InternetService')['Churn'].
  ↪value_counts(normalize=True) * 100

# Extract 'Yes' percentages
```

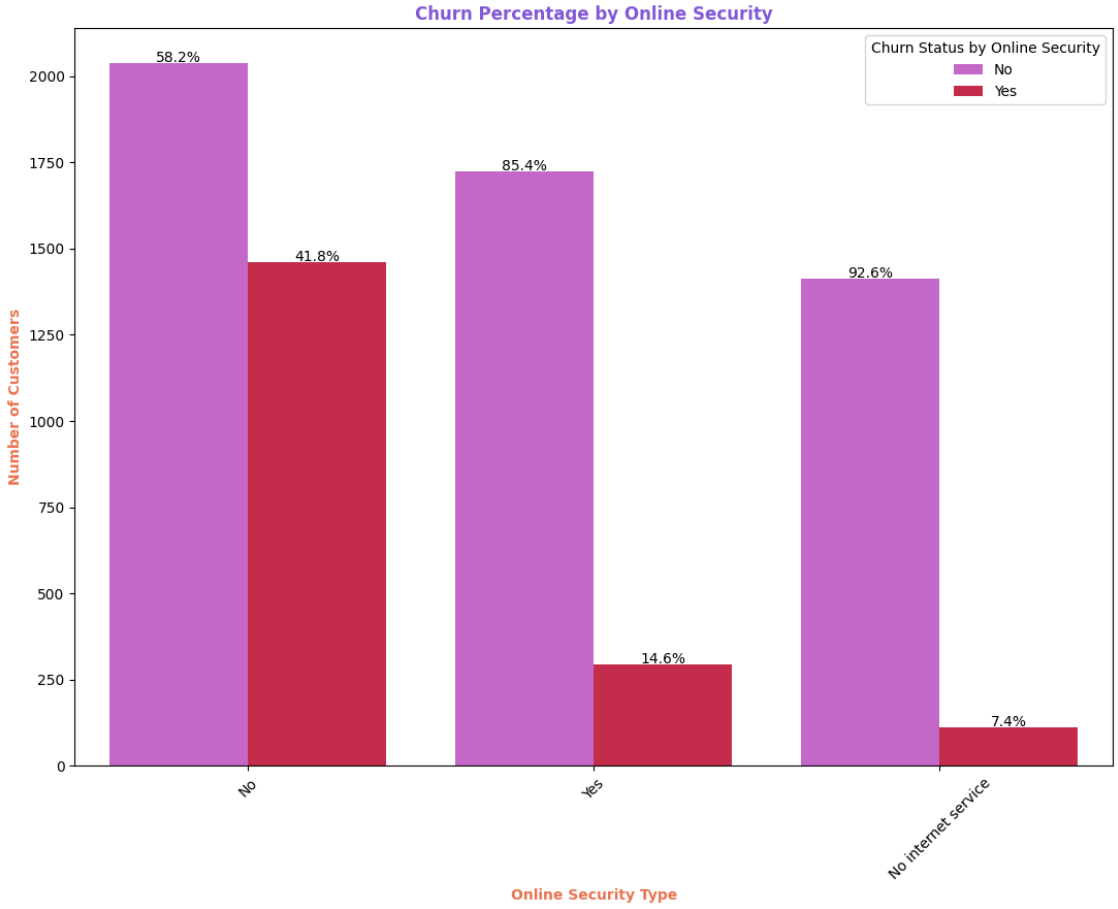
```
yes_churn_pct = {service: churn_percentages.loc[(service, 'Yes')] for service␣
 ↪in df['InternetService'].unique()}

# Extract 'No' percentages
no_churn_pct = {service: churn_percentages.loc[(service, 'No')] for service in␣
 ↪df['InternetService'].unique()}

colors = ["#FF7F50", "#1f77b4", "#aec7e8"]
labels = list(yes_churn_pct.keys())

#  figure and subplots side by side
 # 1 =row, 2 =columns

fig, axes = plt.subplots(1, 2, figsize=(14, 8))

# pie chart for 'Yes' percentage
axes[0].pie(yes_churn_pct.values(), labels=[f'{label} ({pct:.1f}%)' for label,␣
 ↪pct in yes_churn_pct.items()],
           colors=colors, autopct='%1.1f%%', startangle=140)
axes[0].set_title('Percentage of Customers Who Churned (Yes) by Internet␣
 ↪Service', fontsize=12, color="#138d75", fontweight="bold")
axes[0].axis('equal')
axes[0].legend(labels, title='Internet Service', loc='upper right')

# pie chart for 'No' percentages
axes[1].pie(no_churn_pct.values(), labels=[f'{label} ({pct:.1f}%)' for label,␣
 ↪pct in no_churn_pct.items()],
           colors=colors, autopct='%1.1f%%', startangle=140)
axes[1].set_title('Percentage of Customers Who Did Not Churn (No) by Internet␣
 ↪Service', fontsize=12, color="#d258d6", fontweight="bold")
axes[1].axis('equal')
axes[1].legend(labels, title='Internet Service', loc='upper right')

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





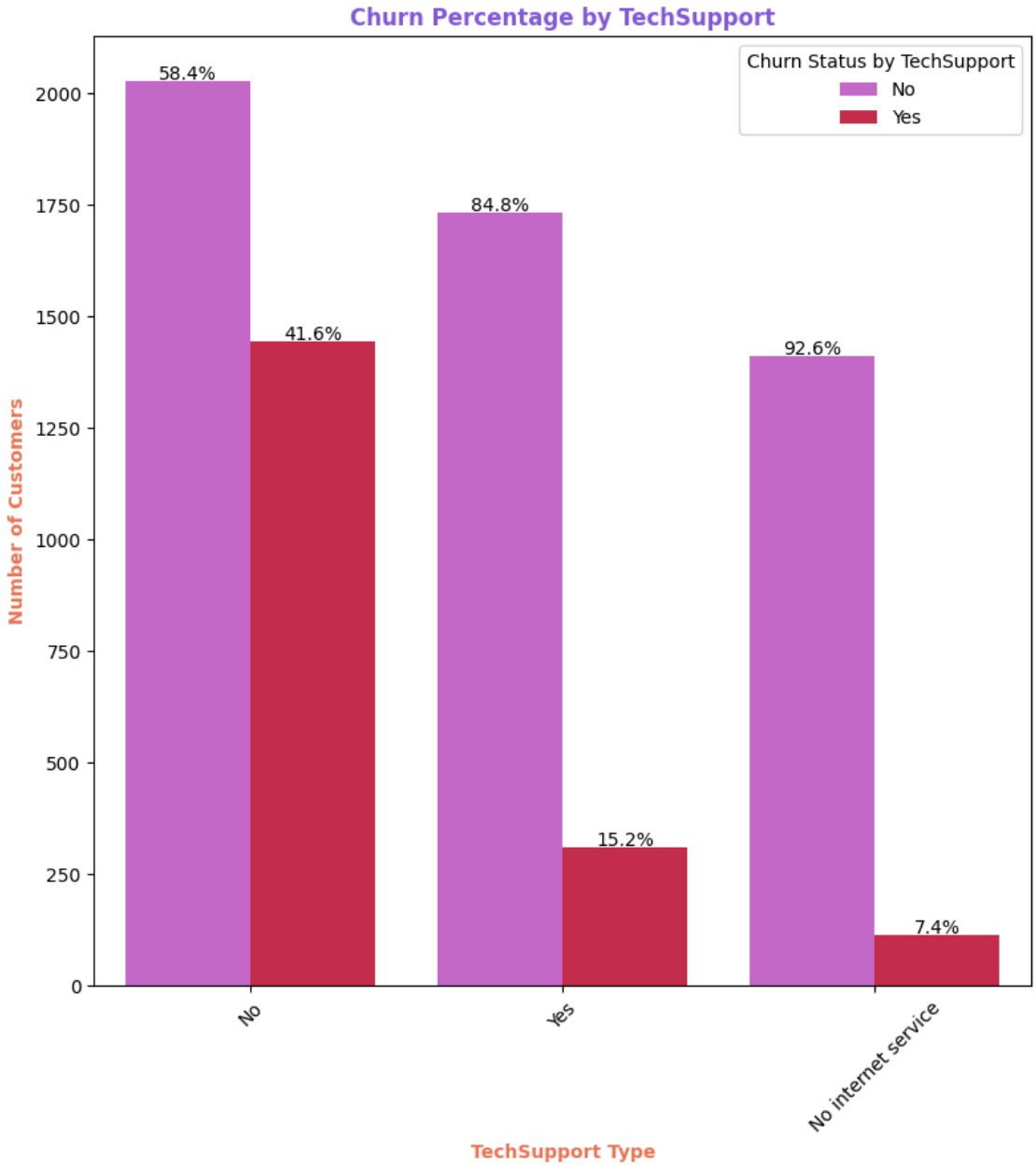
[27]:
```
#for Online Security
plt.figure(figsize=(11, 9))
ax = sns.countplot(x='OnlineSecurity', hue='Churn', data=df,␣
 ↪palette=['#d258d6', '#DC143C'])
plt.title('Churn Percentage by Online␣
 ↪Security',color="#8258d6",fontweight="bold")
plt.xlabel('Online Security Type',color="#E97451",fontweight="bold")
plt.ylabel('Number of Customers',color="#E97451",fontweight="bold")
plt.xticks(rotation=45)
plt.legend(title="Churn Status by Online Security",)

total_per_contract = df['OnlineSecurity'].value_counts()

for p in ax.patches:
    height = p.get_height()
    if height > 0:
        x = p.get_x() + p.get_width() / 2
        x_index = int(round(x))
        if 0 <= x_index < len(ax.get_xticklabels()):
            contract_group = ax.get_xticklabels()[x_index].get_text()
            total = total_per_contract[contract_group]
            percentage = f'{100 * height / total:.1f}%'
            y = height + 3
            ax.text(x, y, percentage, ha='center', fontsize=10)

plt.tight_layout()
```

```
plt.show()
```



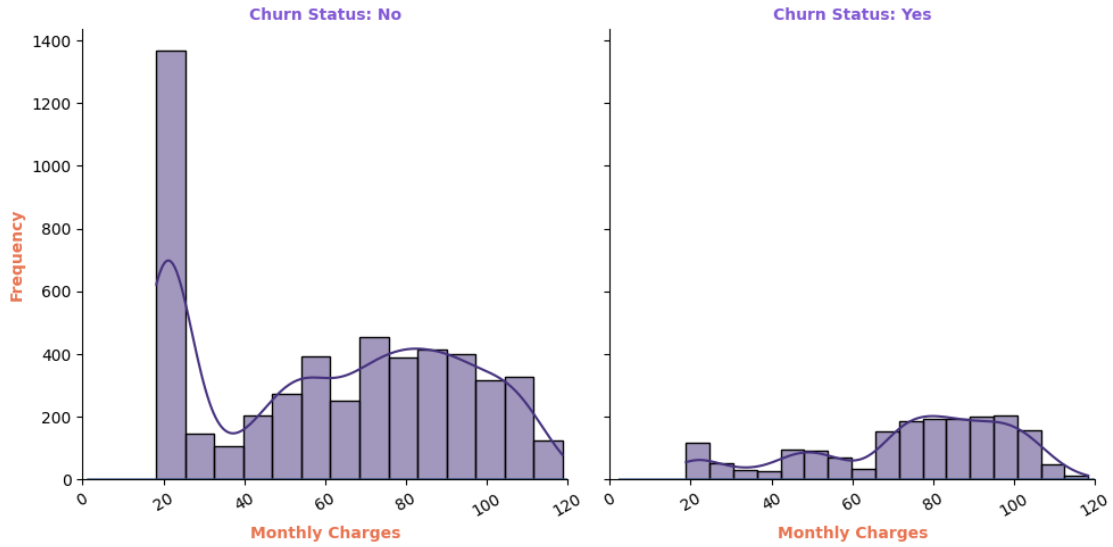
[33]:
```
#for TechSupport
plt.figure(figsize=(8, 9))
ax = sns.countplot(x='TechSupport', hue='Churn', data=df, palette=['#d258d6',␣
↪'#DC143C'])
plt.title('Churn Percentage by TechSupport',color="#8258d6",fontweight="bold")
plt.xlabel('TechSupport Type',color="#E97451",fontweight="bold")
plt.ylabel('Number of Customers',color="#E97451",fontweight="bold")
plt.xticks(rotation=45)
plt.legend(title="Churn Status by TechSupport",)

total_per_contract = df['TechSupport'].value_counts()

for p in ax.patches:
    height = p.get_height()
    if height > 0:
        x = p.get_x() + p.get_width() / 2
```

```
        x_index = int(round(x))
        if 0 <= x_index < len(ax.get_xticklabels()):
            contract_group = ax.get_xticklabels()[x_index].get_text()
            total = total_per_contract[contract_group]
            percentage = f'{100 * height / total:.1f}%'
            y = height + 3
            ax.text(x, y, percentage, ha='center', fontsize=10)

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



MonthlyCharges

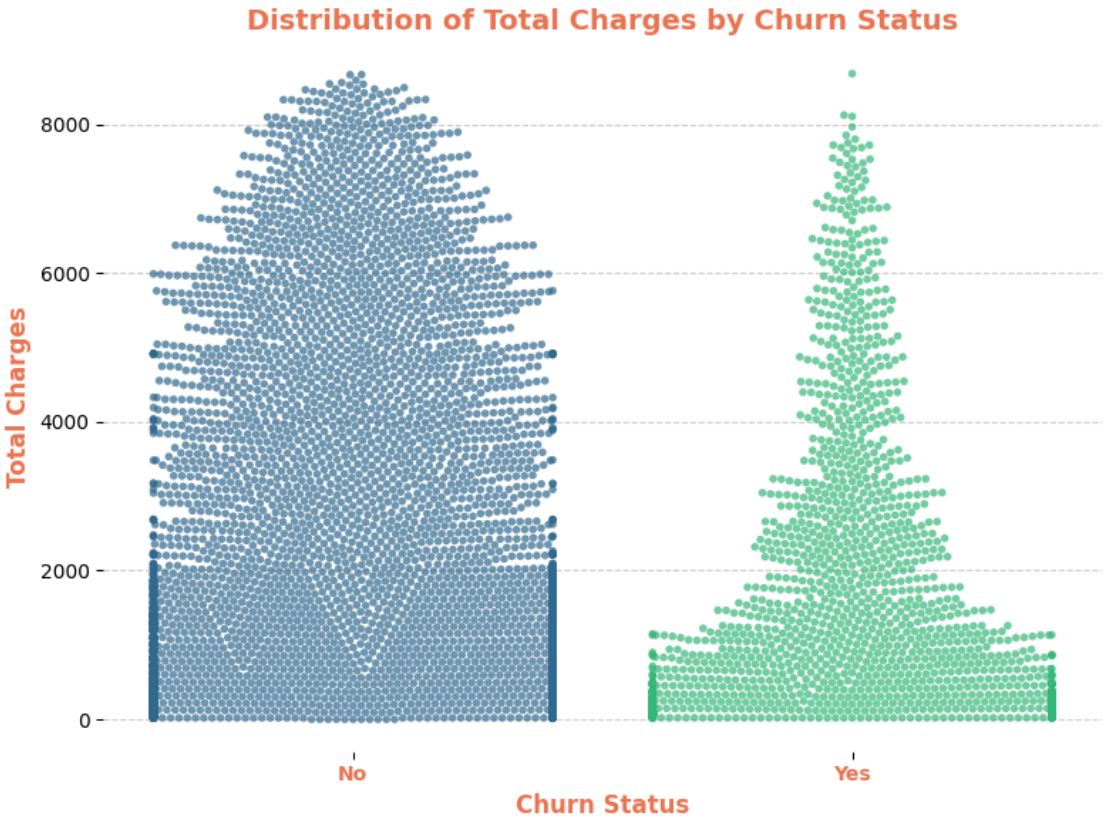
[29]:
```
# color palette
palette = "viridis"

g = sns.FacetGrid(df, col="Churn", height=5, aspect=1, palette=palette,␣
 ↪col_order=['No', 'Yes'])
g.map(sns.histplot, "MonthlyCharges", kde=True, color=sns.
 ↪color_palette(palette)[0])
g.map(sns.kdeplot, "MonthlyCharges", color=sns.color_palette(palette)[1])
g.set_axis_labels("Monthly Charges", "Frequency", color="#E97451",␣
 ↪fontweight="bold")
g.set_titles("Churn Status: {col_name}", color="#8258d6", fontweight="bold")
g.set(xlim=(df['MonthlyCharges'].min(), df['MonthlyCharges'].max()))
g.set_xticklabels(rotation=30)
plt.tight_layout()
plt.show()
```



[30]:
```
plt.figure(figsize=(8, 6))
sns.swarmplot(x='Churn',y='TotalCharges', data=df,size=4,alpha=0.
 ↪7,palette="viridis")

plt.xlabel("Churn Status",color="#E97451",fontweight="bold", fontsize=12)
plt.ylabel("Total Charges",color="#E97451",fontweight="bold", fontsize=12)
plt.title("Distribution of Total Charges by Churn␣
 ↪Status",color="#E97451",fontweight="bold", fontsize=14)
plt.xticks(ticks=[0, 1], labels=['No', 'Yes'],color="#E97451",fontweight="bold")
```

```
plt.grid(axis='y', linestyle='--', alpha=0.6)
sns.despine(left=True, bottom=True)
plt.tight_layout()
plt.show()
```

```
<ipython-input-30-63f919217b5f>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.swarmplot(x='Churn',y='TotalCharges',
data=df,size=4,alpha=0.7,palette="viridis")
/usr/local/lib/python3.11/dist-packages/seaborn/categorical.py:3399:
UserWarning: 25.9% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.11/dist-packages/seaborn/categorical.py:3399:
UserWarning: 25.4% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.11/dist-packages/seaborn/categorical.py:3399:
UserWarning: 36.8% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.11/dist-packages/seaborn/categorical.py:3399:
UserWarning: 32.0% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



This notebook was converted with convert.ploomber.io