```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv(r'D:\python\protfilo\WA_Fn-UseC_-Telco-Customer-
Churn.csv')
df.head()
```

```
   customerID  gender  SeniorCitizen Partner Dependents  tenure
PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1
No
1  5575-GNVDE    Male              0      No         No      34
Yes
2  3668-QPYBK    Male              0      No         No       2
Yes
3  7795-CFOCW    Male              0      No         No      45
No
4  9237-HQITU  Female              0      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        1889.5    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]
```

Assistant

The error occurs because you're trying to use `pd.read.csv()` which is incorrect. In pandas, the correct method to read CSV files is `pd.read_csv()` (with an underscore, not a dot).

Would you like me to provide the corrected code?

```
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
```

#repling the blank with O , on the totacharges

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

#df.isnull() is used to find any null values presents on the table

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

```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

```
df.describe()
```

```
       SeniorCitizen       tenure  MonthlyCharges  TotalCharges
count    7043.000000  7043.000000     7043.000000   7043.000000
mean        0.162147    32.371149       64.761692   2279.734304
std         0.368612    24.559481       30.090047   2266.794470
min         0.000000     0.000000       18.250000      0.000000
25%         0.000000     9.000000       35.500000    398.550000
50%         0.000000    29.000000       70.350000   1394.550000
75%         0.000000    55.000000       89.850000   3786.600000
max         1.000000    72.000000      118.750000   8684.800000
```

#cheching the duplication values on the table

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

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

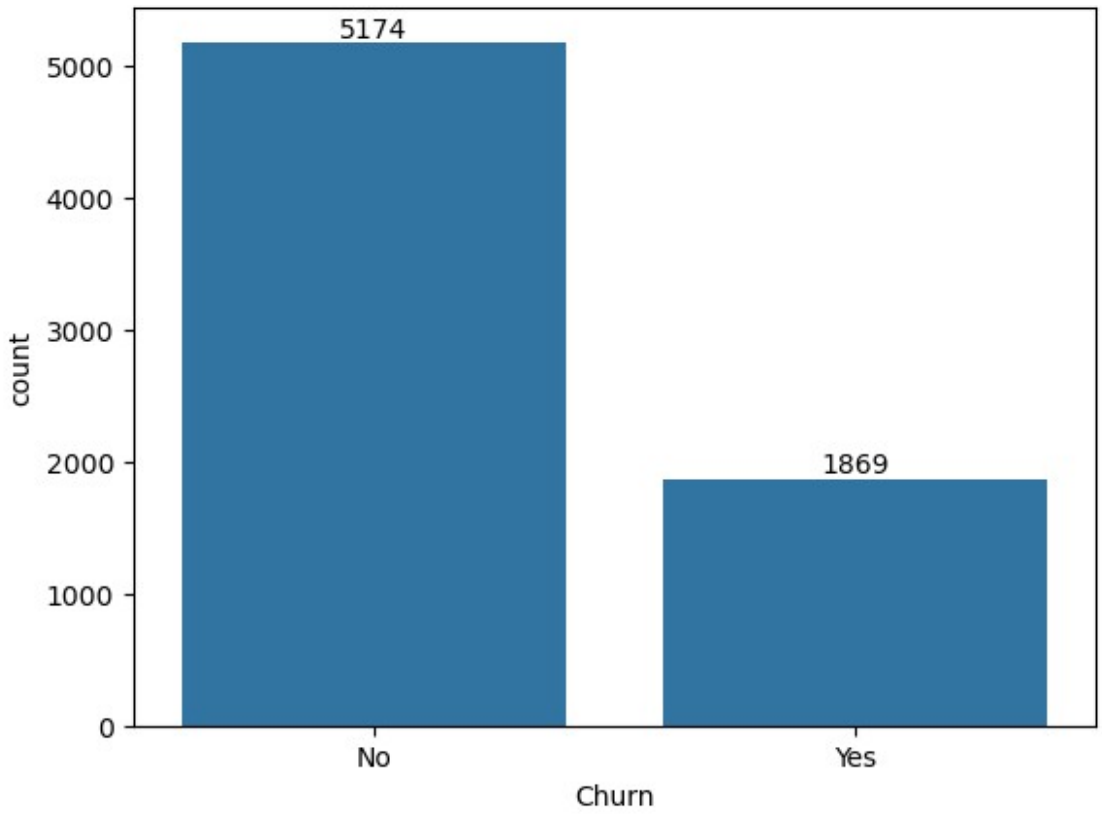
#converted 0 and 1 value of senoircitizon to yes/no , to make it easiler to understand

```
def conv(value):
    if value == 1:
        return "yess"
    else:
        return "Noo"

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

df.head(5
       )
```

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

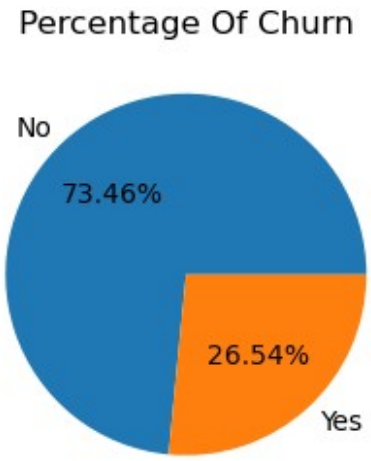
```
               PaymentMethod MonthlyCharges  TotalCharges  Churn
0           Electronic check          29.85         29.85     No
1               Mailed check          56.95       1889.50     No
2               Mailed check          53.85        108.15    Yes
3  Bank transfer (automatic)          42.30       1840.75     No
4           Electronic check          70.70        151.65    Yes

[5 rows x 21 columns]
```

```
ax =sns.countplot(x= 'Churn',data =df)

ax.bar_label(ax.containers[0])
plt.show()
```

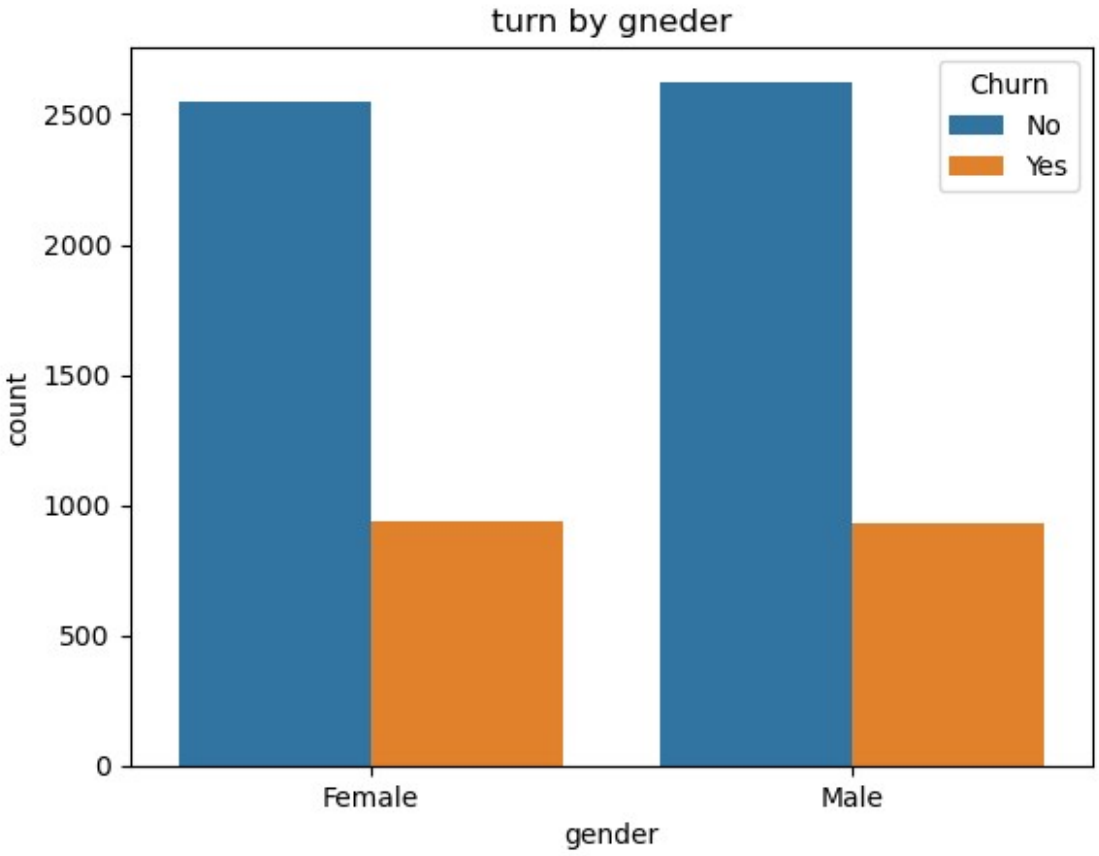


#from give pie chart we can conclude that26.54% of our customer have churned out.

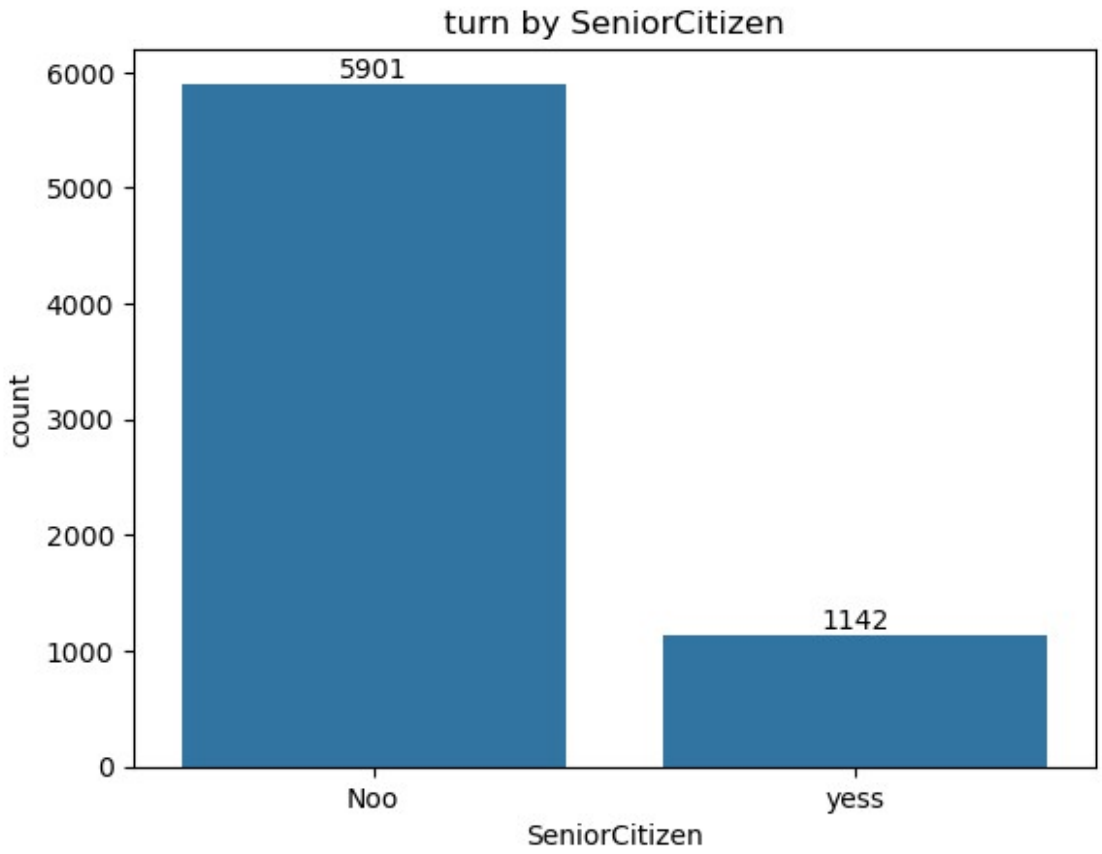
```
gb = df.groupby("Churn").agg({'Churn':"count"})
plt.figure(figsize =(3,4))
plt.pie(gb['Churn'], labels = gb.index ,autopct = "%1.2f%%")
plt.title("Percentage Of Churn")
plt.show()
```




```
sns.countplot(x = "gender", data = df, hue = "Churn"  )
plt.title("turn by gneder")
plt.show()
```



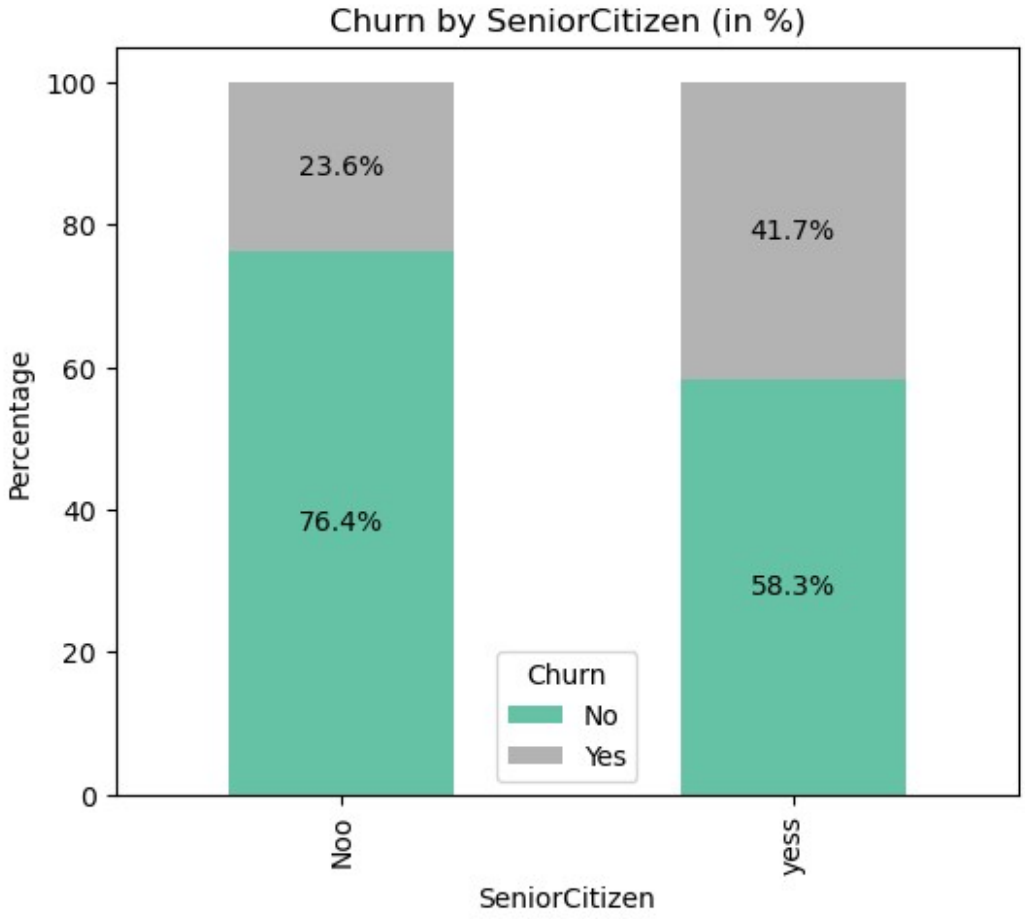
```
ax = sns.countplot(x = "SeniorCitizen", data = df  )
ax.bar_label(ax.containers[0])
plt.title("turn by SeniorCitizen")
plt.show()
```



#comperation agrated percentage of people in senior citizen category have churned

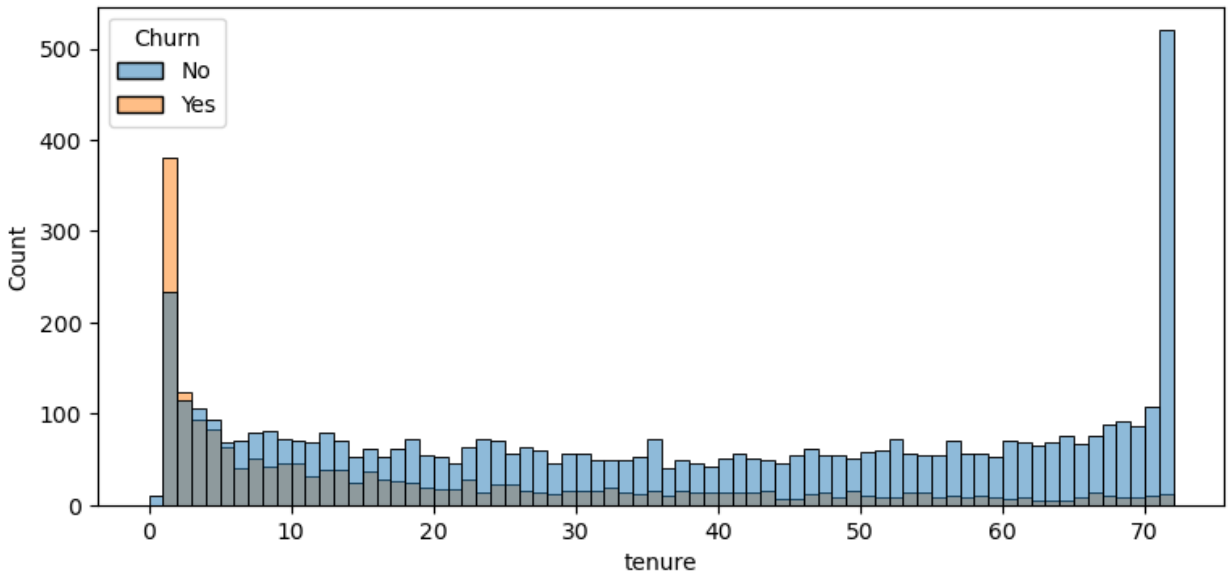
```
ct = pd.crosstab(df['SeniorCitizen'], df['Churn'])

# Convert to percentages row-wise
ct_percent = ct.div(ct.sum(axis=1), axis=0) * 100

# Plot stacked bar chart
ax = ct_percent.plot(kind='bar', stacked=True, figsize=(6,5),
colormap="Set2")

# Add percentage labels
for c in ax.containers:
    ax.bar_label(c, fmt='%.1f%%', label_type='center')

plt.title("Churn by SeniorCitizen (in %)")
plt.ylabel("Percentage")
plt.xlabel("SeniorCitizen")
```

```
plt.legend(title="Churn")
plt.show()
```

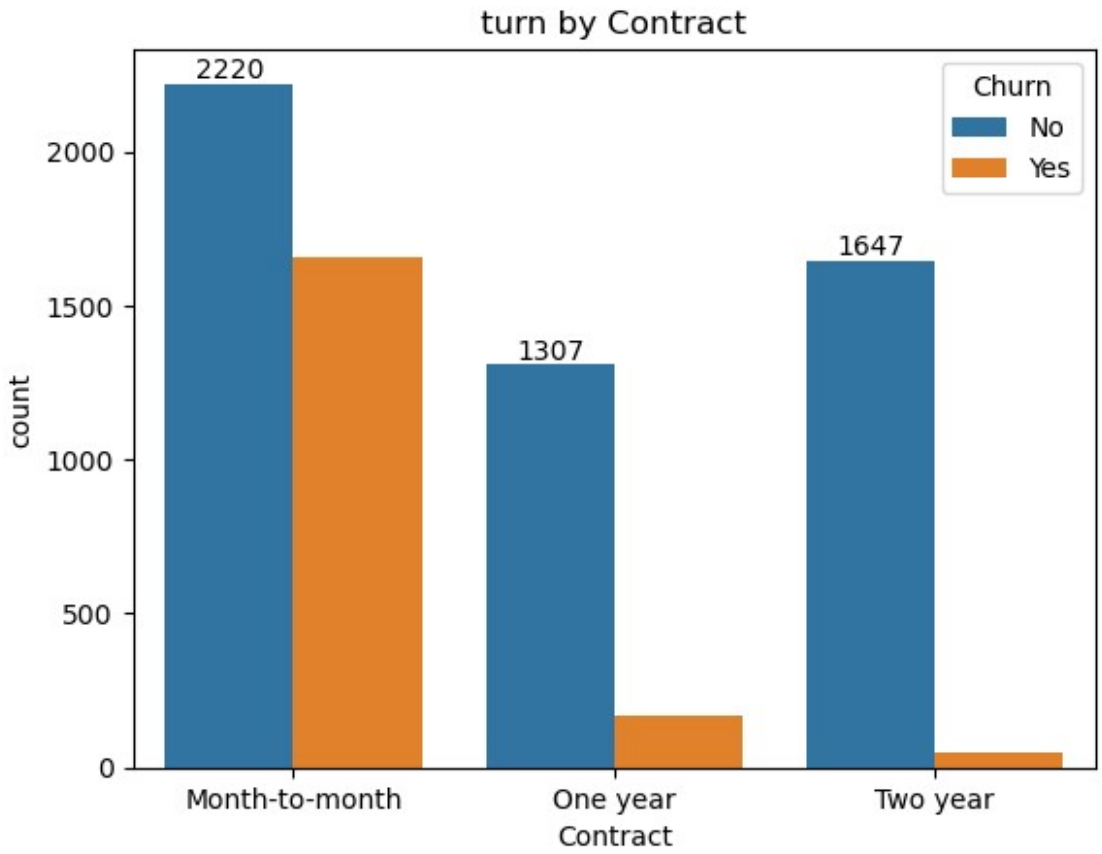


#people whoo have used the service for a long time stayed and people who have used our seviecs on these mouth have churned

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

plt.show()
```




```
#people who have mouth to mouth contact are likely chrun

ax = sns.countplot(x = "Contract", data = df , hue = "Churn" )
ax.bar_label(ax.containers[0])
plt.title("turn by Contract")
plt.show()
```

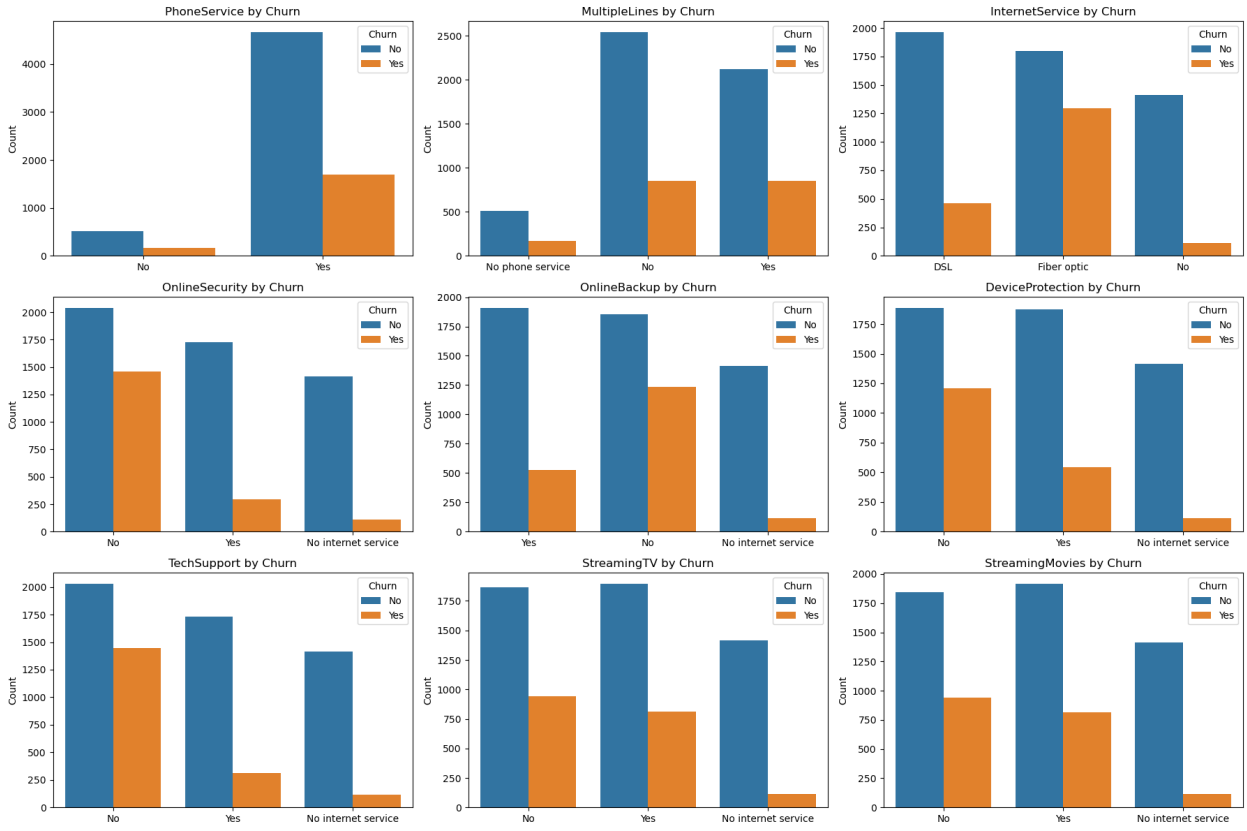



```
df.columns.values

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

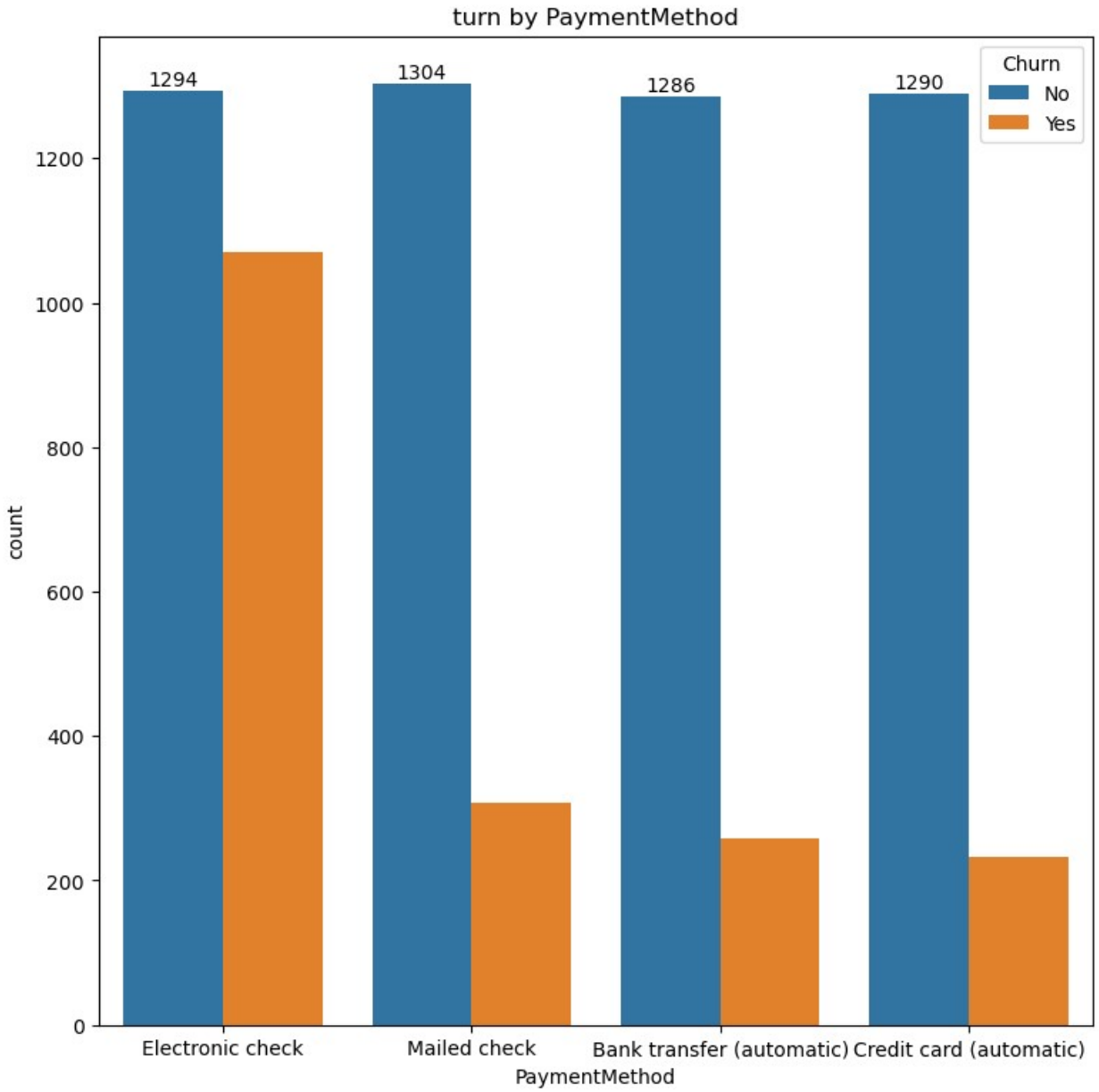
#Most customers have phone and internet services, with fiber optic users showing the highest churn. Lack of value-added services like online security, backup, device protection, and tech support is strongly linked to higher churn. Customers without internet service churn very little. Overall, churn is driven more by service quality and support features than by entertainment options like streaming TV or movies.

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

# Create subplots (3 rows x 3 cols for 9 plots)
fig, axes = plt.subplots(3, 3, figsize=(18, 12))
```

```
axes = axes.flatten()

# Loop through columns and create countplots
for i, col in enumerate(cols):
    sns.countplot(x=col, data=df, hue="Churn", ax=axes[i])
    axes[i].set_title(f"{col} by Churn", fontsize=12)
    axes[i].set_xlabel("")  # remove x-axis label (clean look)
    axes[i].set_ylabel("Count")

# Remove empty subplot if cols < grid size
for j in range(len(cols), len(axes)):
    fig.delaxes(axes[j])

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




```
#the customer who using the electroic checks to make payment , is most churning out customers

plt.figure (figsize = (9,9))
ax = sns.countplot(x = "PaymentMethod", data = df , hue = "Churn" )
ax.bar_label(ax.containers[0])
plt.title("turn by PaymentMethod")
plt.show()
```


turn by PaymentMethod
Churn
No
Yes
1294
1304
1286
1290
1200
1000
800
600
400
200
0
count
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
PaymentMethod