In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
telco = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")
type(telco)
```

Out[2]:
```
pandas.core.frame.DataFrame
```

In [3]:
```
telco.head()
```

Out[3]:

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

5 rows × 21 columns

In [ ]:
```
1. The target variable in this dataset is "Churn." It is the variable that you are trying to predict. Specifically, you want to determine whether a customer v
```

In [ ]:
```
2. This is a binary classification problem. You are trying to classify customers into two categories: those who will churn and those who will stay.
```

In [4]:
```
column_names = telco.columns
data_types = telco.dtypes

print("Column Names:")
print(column_names)

print("\nData Types:")
print(data_types)
```

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

Data Types:
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

In [5]:
```
missing_values = telco.isnull().sum()
print(missing_values)
```

```
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]:
```
telco = telco.dropna()
```

In [7]:
```
duplicates = telco[telco.duplicated()]
print(duplicates)
```

```
Empty DataFrame
Columns: [customerID, gender, SeniorCitizen, Partner, Dependents, tenure, PhoneService, MultipleLines, InternetService, OnlineSecurity, OnlineBackup, De
viceProtection, TechSupport, StreamingTV, StreamingMovies, Contract, PaperlessBilling, PaymentMethod, MonthlyCharges, TotalCharges, Churn]
Index: []

[0 rows x 21 columns]
```

In [8]:
```
telco = telco.drop_duplicates()
```

In [ ]:
```
Use methods like label encoding or one-hot encoding if there are category qualities that need to be converted into numerical values. The categorical fe
```

In [11]:
```
churn_counts = telco['Churn'].value_counts()
print(churn_counts)
```

```
Churn
No     5174
Yes    1869
Name: count, dtype: int64
```

In [ ]:
```
An outlier is an observation that lies an abnormal distance from other values in a dataset. For example,a very high or extremely low income compared
```

In [ ]:
```
You can use techniques like Z-score or IQR (Interquartile Range) to detect outliers and handle them. Depending on the specific analysis and how outli
```

In [17]:
```
import seaborn as sns
```

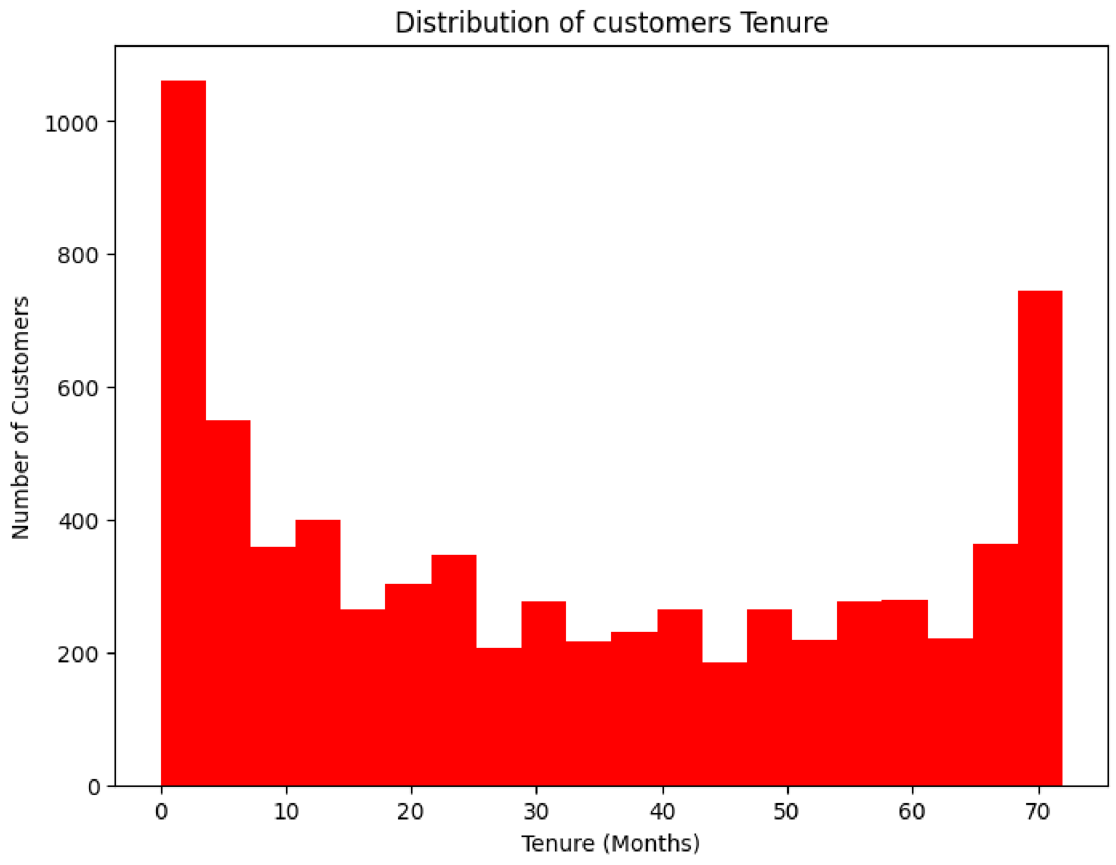
In [18]:
```
correlation = telco['Churn'].replace({'No':0, 'Yes':1}).corr(telco['gender'].replace({'Male':0, 'Female':1}))
print(correlation)
```

```
0.008612095078997867
There is no pattern
```

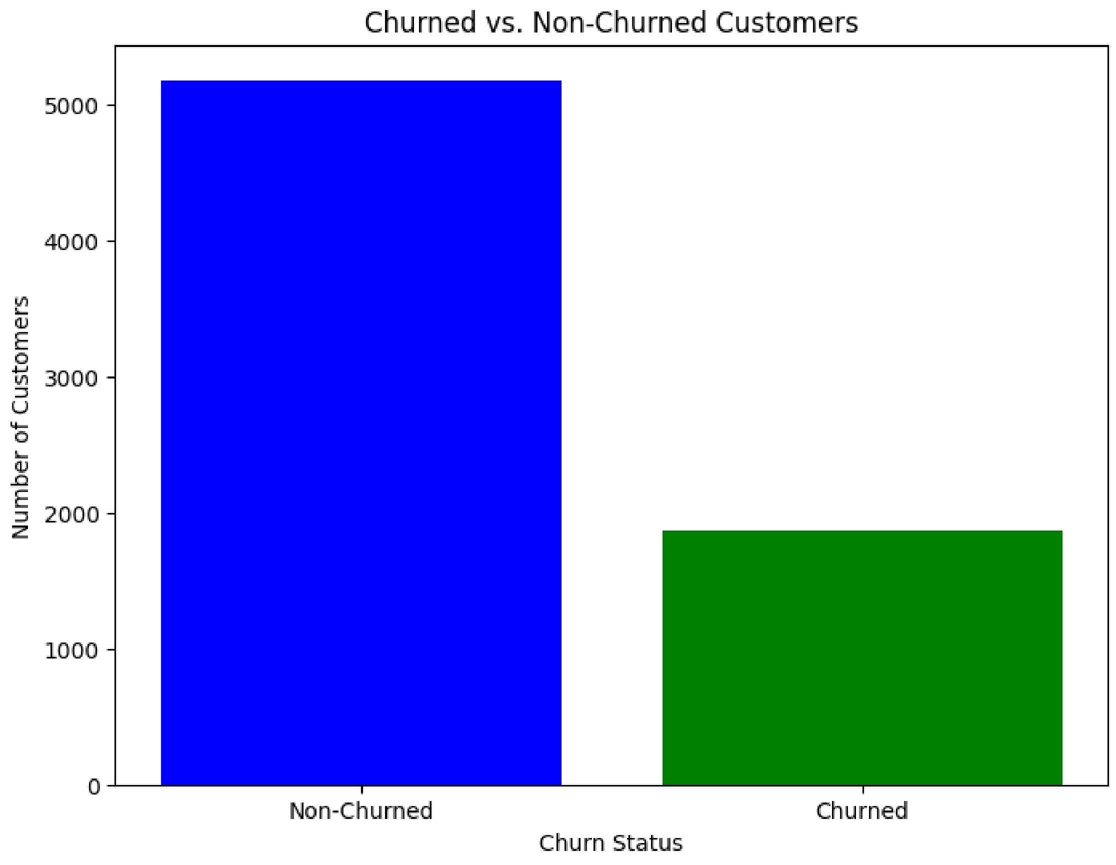
In [19]:
```
churn_percent = (telco['Churn'].value_counts() / len(telco)) * 100
print(churn_percent)
```

```
Churn
No     73.463013
Yes    26.536987
Name: count, dtype: float64
```

In [20]:
```
import matplotlib.pyplot as plt
plt. figure(figsize=(8, 6))
plt.hist (telco['tenure'], bins=20, color='red')
plt.xlabel('Tenure (Months)')
plt.ylabel('Number of Customers')
plt.title('Distribution of customers Tenure')
plt.show()
```

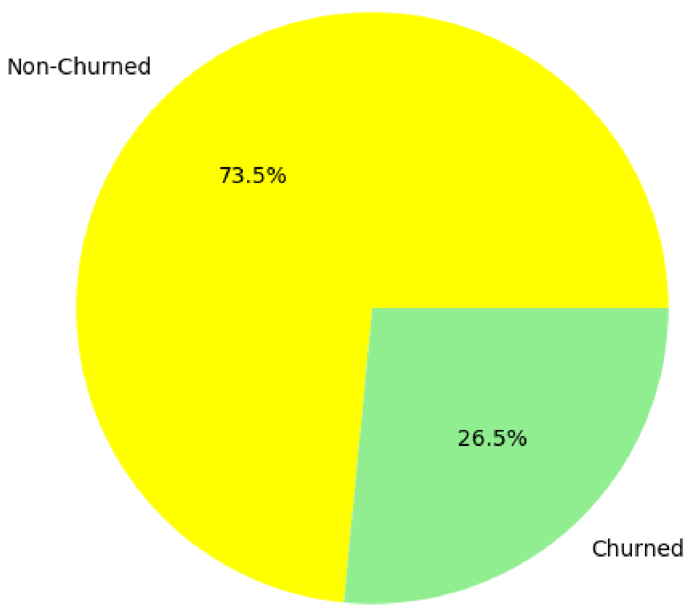


In [21]:
```
plt.figure(figsize=(8, 6))
plt.bar(churn_counts. index, churn_counts.values, color=['blue', 'green'])
plt.xlabel ('Churn Status')
plt.ylabel ('Number of Customers')
plt.title('Churned vs. Non-Churned Customers')
plt.xticks([0,1], ['Non-Churned', 'Churned'])
plt.show()
```



In [22]:
```
churn_percent = (telco['Churn'].value_counts() / len(telco)) * 100
plt.figure(figsize=(8, 6))
plt.pie(churn_percent, labels=['Non-Churned', 'Churned'], autopct='%1.1f%%', colors=['yellow', 'lightgreen'])
plt.title('Percentage of Churned vs. Active Customers')
plt.show()
```

Percentage of Churned vs. Active Customers


Non-Churned
Churned