IT Customer Churn Prediction with Imbalanced Data

Overview of Problem Statement:

Predicting customer churn is a critical business objective for IT companies as it is significantly more cost-effective to retain existing customers than to acquire new ones. This project aims to develop a predictive model to identify customers who are likely to churn, enabling the company to implement targeted retention strategies. A key challenge in this task is the imbalanced nature of churn data, where the number of churning customers is typically much smaller than non-churning customers. Addressing this data imbalance is crucial for building an accurate and reliable churn prediction model.

Key Project Points:

- **Objective:** Predict customer churn to inform retention strategies.
- **Challenge:** Handling imbalanced dataset (fewer churn examples).
- **Data:** Utilize provided customer data including services, account info, and demographics.
- **Methodology:** Explore techniques for handling imbalanced data (e.g., resampling, different evaluation metrics) and build a predictive model.
- **Impact:** Enable targeted interventions to reduce customer churn and increase retention.

Objective:

To develop an accurate and reliable predictive model for IT customer churn, specifically addressing the challenges posed by imbalanced data, in order to support targeted customer retention strategies.

Data Description:

- **Source:** [Specify the source of data, e.g., Kaggle, internal database]
- **Features:**
    - Churn (Target Variable)
    - Services (phone, multiple lines, internet, online security, online backup, device protection, tech support, streaming TV and movies)
    - Account Information (tenure, contract, payment method, paperless billing, monthly charges, total charges)
    - Demographics (gender, age range, partners, dependents)

Data Collection

In [2]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```
df = pd.read_csv('IT_Customer_Churn.csv')
df.head()
```

Out[3]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	On
0	Female	0	Yes	No	1	No	No phone service	DSL	No
1	Male	0	No	No	34	Yes	No	DSL	Ye
2	Male	0	No	No	2	Yes	No	DSL	Ye
3	Male	0	No	No	45	No	No phone service	DSL	Ye
4	Female	0	No	No	2	Yes	No	Fiber optic	No

In [4]:

```
df.tail()
```

Out[4]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService
7038	Male	0	Yes	Yes	24	Yes	Yes	DSL
7039	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic
7040	Female	0	Yes	Yes	11	No	No phone service	DSL
7041	Male	1	Yes	No	4	Yes	Yes	Fiber optic
7042	Male	0	No	No	66	Yes	No	Fiber optic

In [5]:

```
df.shape
```

Out[5]:

```
(7043, 20)
```

In [6]:

```
print(df.isnull().sum())
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 [7]:

```
print(df.info())
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   gender            7043 non-null   object
 1   SeniorCitizen     7043 non-null   int64
 2   Partner           7043 non-null   object
 3   Dependents        7043 non-null   object
 4   tenure            7043 non-null   int64
 5   PhoneService      7043 non-null   object
 6   MultipleLines     7043 non-null   object
 7   InternetService   7043 non-null   object
 8   OnlineSecurity    7043 non-null   object
 9   OnlineBackup      7043 non-null   object
 10  DeviceProtection  7043 non-null   object
 11  TechSupport       7043 non-null   object
 12  StreamingTV       7043 non-null   object
 13  StreamingMovies   7043 non-null   object
 14  Contract          7043 non-null   object
 15  PaperlessBilling  7043 non-null   object
 16  PaymentMethod     7043 non-null   object
 17  MonthlyCharges    7043 non-null   float64
 18  TotalCharges      7043 non-null   object
 19  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(17)
memory usage: 1.1+ MB
None
```

In [8]:

```
print(df.describe())
       SeniorCitizen       tenure  MonthlyCharges
count    7043.000000  7043.000000     7043.000000
mean        0.162147    32.371149       64.761692
```

```
std         0.368612    24.559481      30.090047
min         0.000000     0.000000      18.250000
25%         0.000000     9.000000      35.500000
50%         0.000000    29.000000      70.350000
75%         0.000000    55.000000      89.850000
max         1.000000    72.000000     118.750000
```

Data Preprocessing - Data Cleaning

In [9]:

```
df.rename(columns={
    'gender':'Gender',
    'SeniorCitizen':'SeniorCitizen',
    'Partner':'Partner',
    'Dependents':'Dependents',
    'tenure':'Tenure',
    'PhoneService':'PhoneService',
    'MultipleLines':'MultipleLines',
    'InternetService':'InternetService',
    'OnlineSecurity':'OnlineSecurity',
    'OnlineBackup':'OnlineBackup',
    'DeviceProtection':'DeviceProtection',
    'TechSupport':'TechSupport',
    'StreamingTV':'StreamingTV',
    'StreamingMovies':'StreamingMovies',
    'Contract':'Contract',
    'PaperlessBilling':'PaperlessBilling',
    'PaymentMethod':'PaymentMethod',
    'MonthlyCharges':'MonthlyCharges',
    'TotalCharges':'TotalCharges',
    'Churn':'Churn'}, inplace=True)
df.head()
```

Out[9]:

	Gender	SeniorCitizen	Partner	Dependents	Tenure	PhoneService	MultipleLines	InternetService	O
0	Female	0	Yes	No	1	No	No phone service	DSL	N
1	Male	0	No	No	34	Yes	No	DSL	Y
2	Male	0	No	No	2	Yes	No	DSL	Y
3	Male	0	No	No	45	No	No phone service	DSL	Y
4	Female	0	No	No	2	Yes	No	Fiber optic	N

In [10]:

```
df.tail()
```

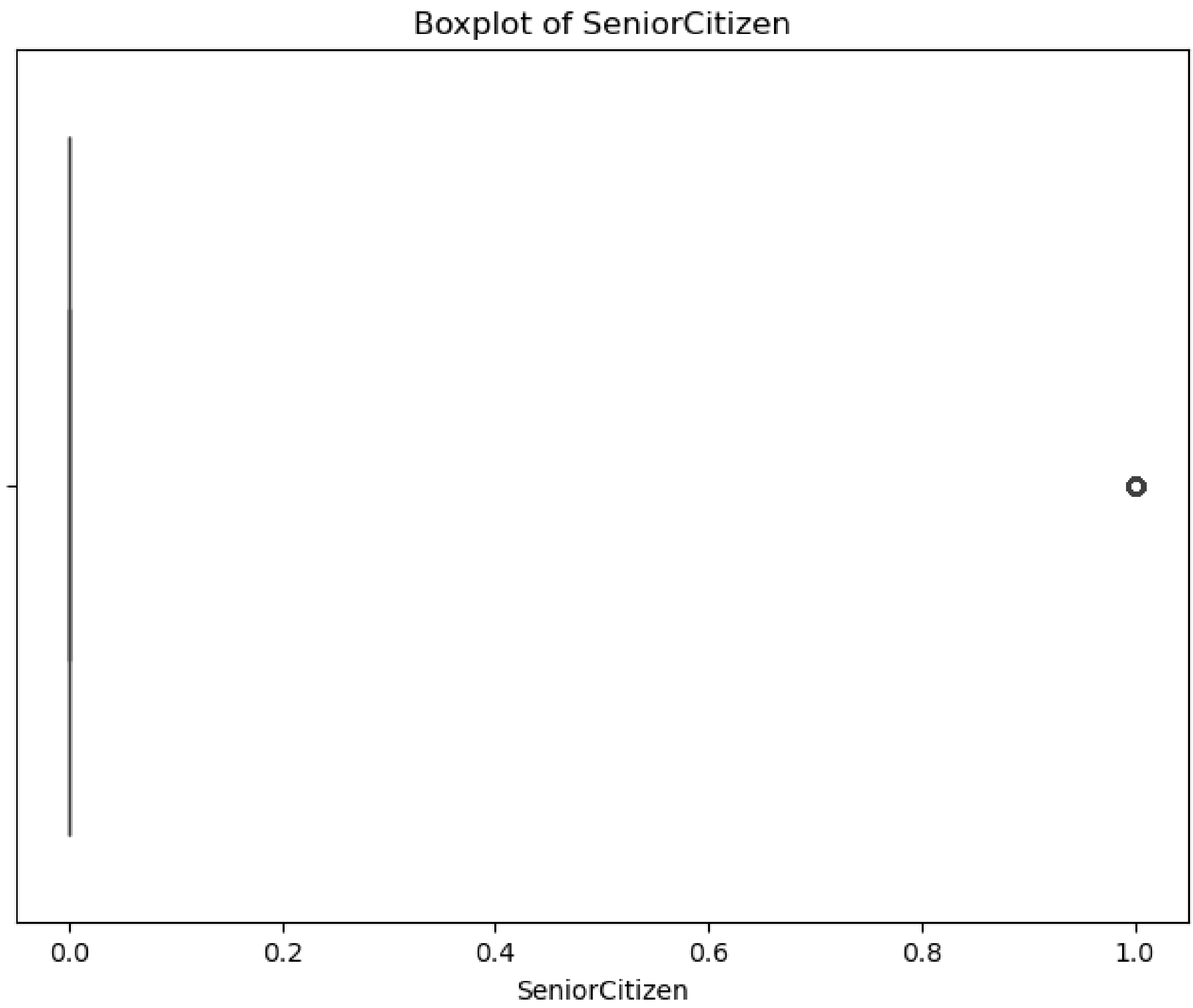
Out[10]:

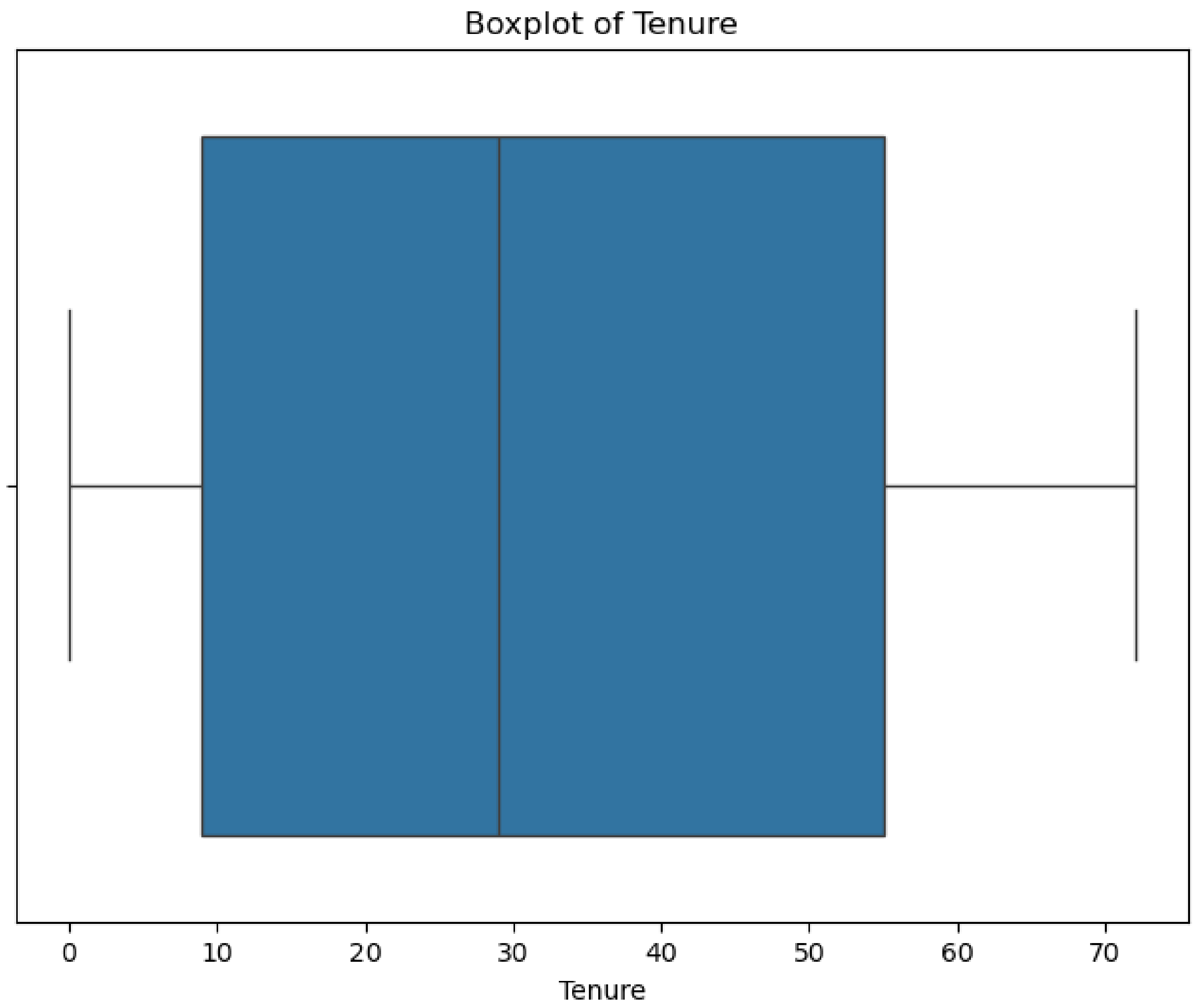
	Gender	SeniorCitizen	Partner	Dependents	Tenure	PhoneService	MultipleLines	InternetServic
7038	Male	0	Yes	Yes	24	Yes	Yes	DSL
7039	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic
7040	Female	0	Yes	Yes	11	No	No phone service	DSL
7041	Male	1	Yes	No	4	Yes	Yes	Fiber optic
7042	Male	0	No	No	66	Yes	No	Fiber optic

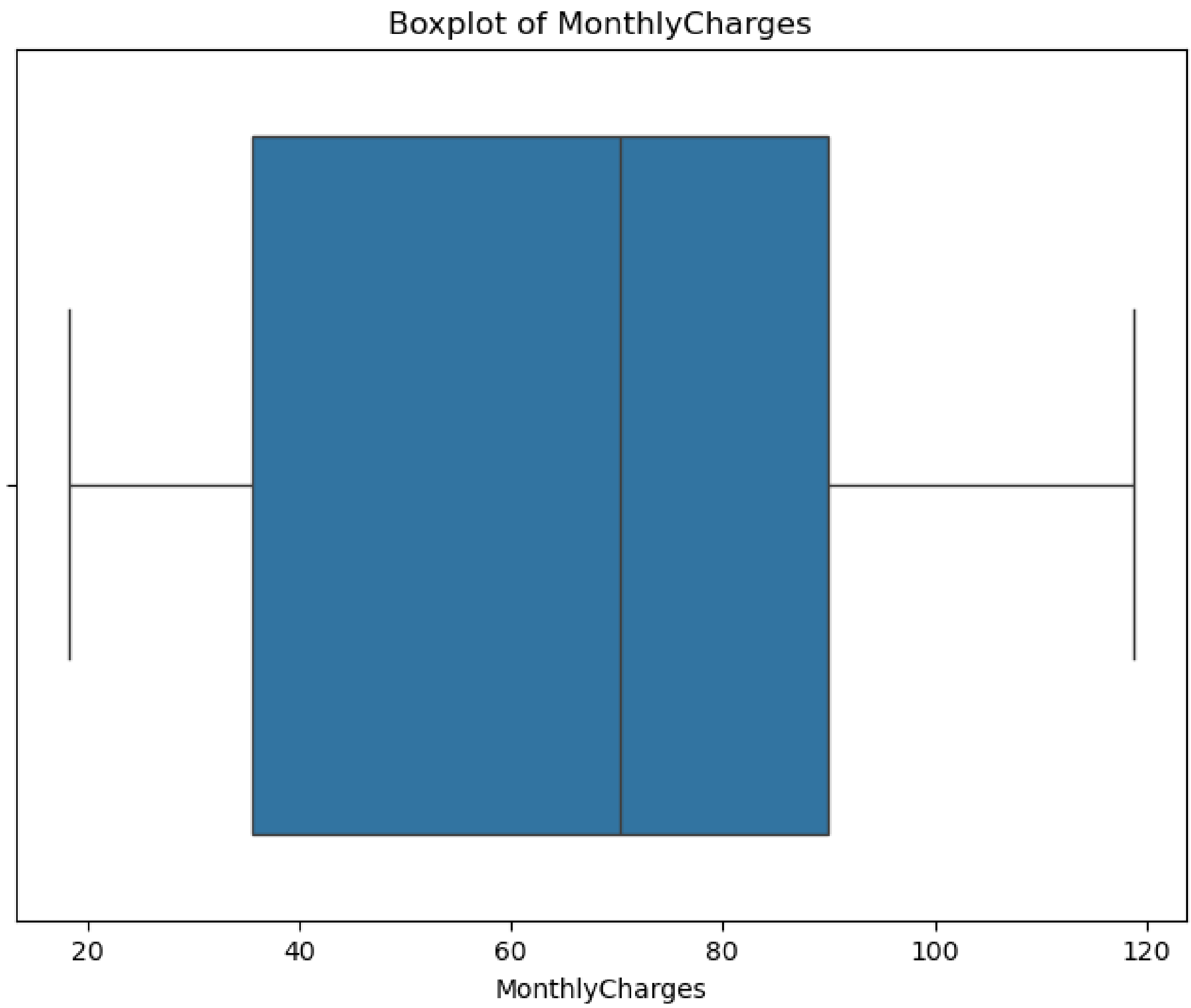
Outlier Detection

In [11]:

```
# outliers using boxplots for numerical features
numerical_features = df.select_dtypes(include=['number']).columns
for col in numerical_features:
  plt.figure(figsize=(8, 6))
  sns.boxplot(x=df[col])
  plt.title(f'Boxplot of {col}')
  plt.show()
```


Boxplot of SeniorCitizen
0.0
0.2
0.4
0.6
0.8
1.0
SeniorCitizen

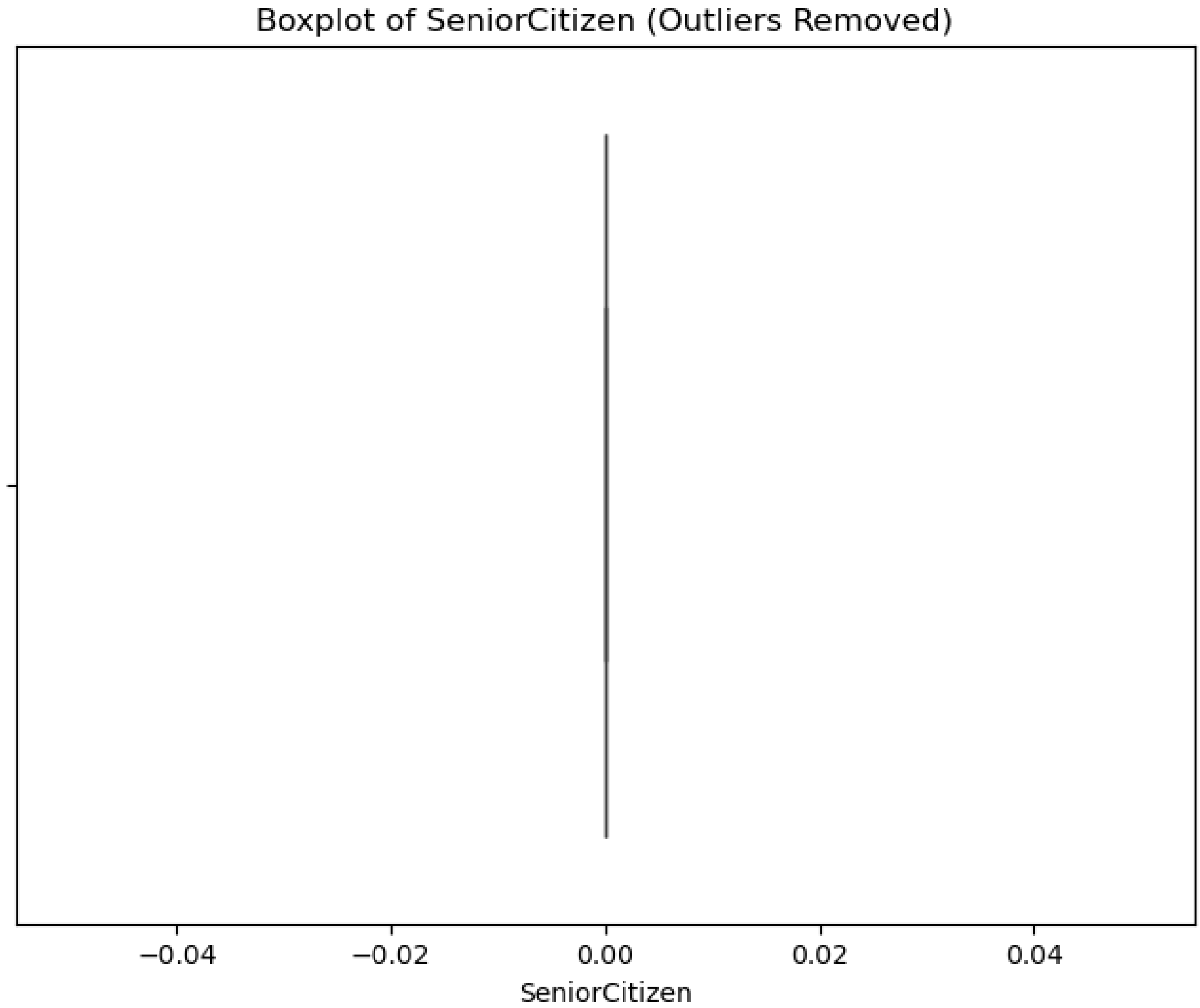

Boxplot of Tenure
0
10
20
30
40
50
60
70
Tenure

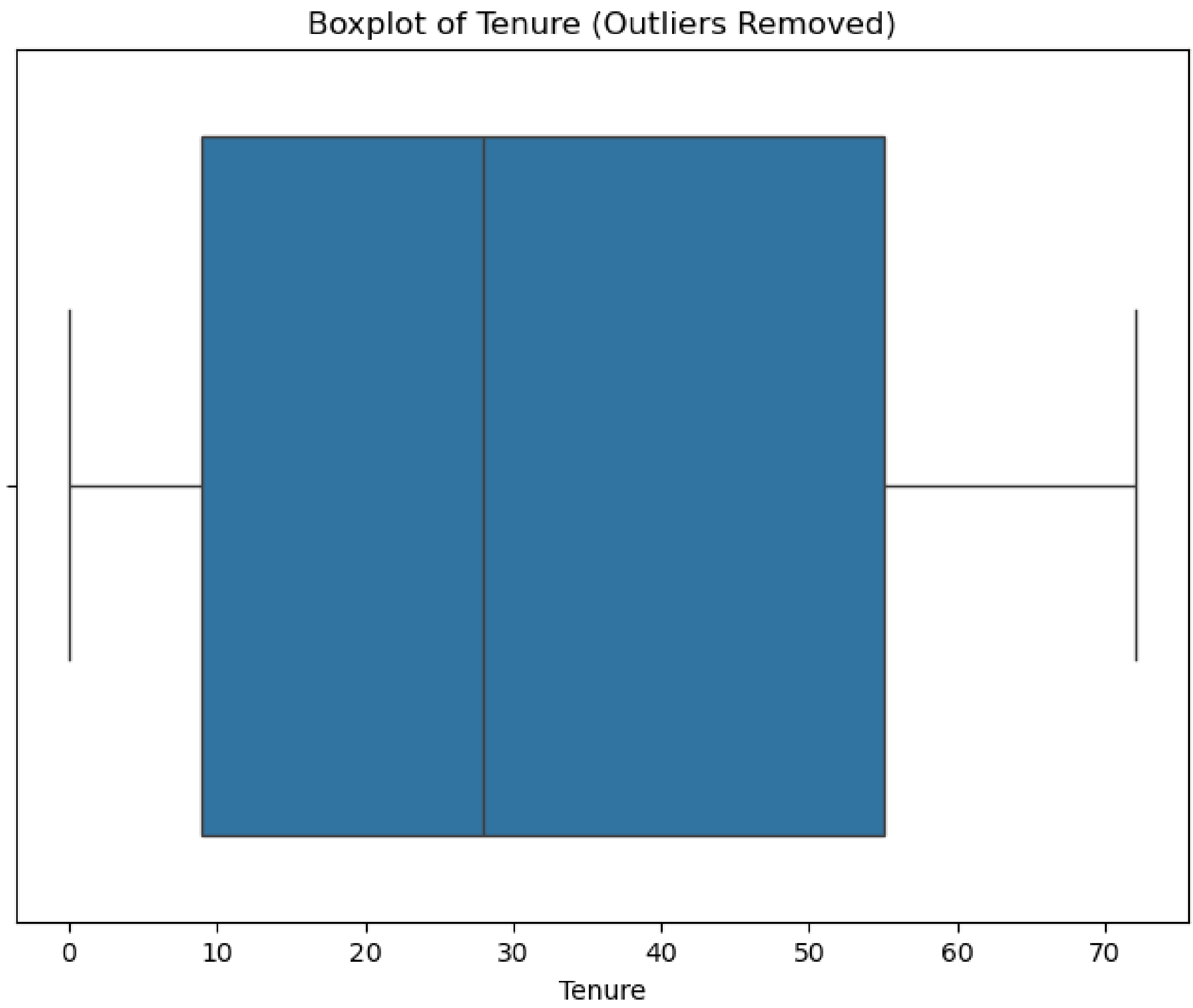


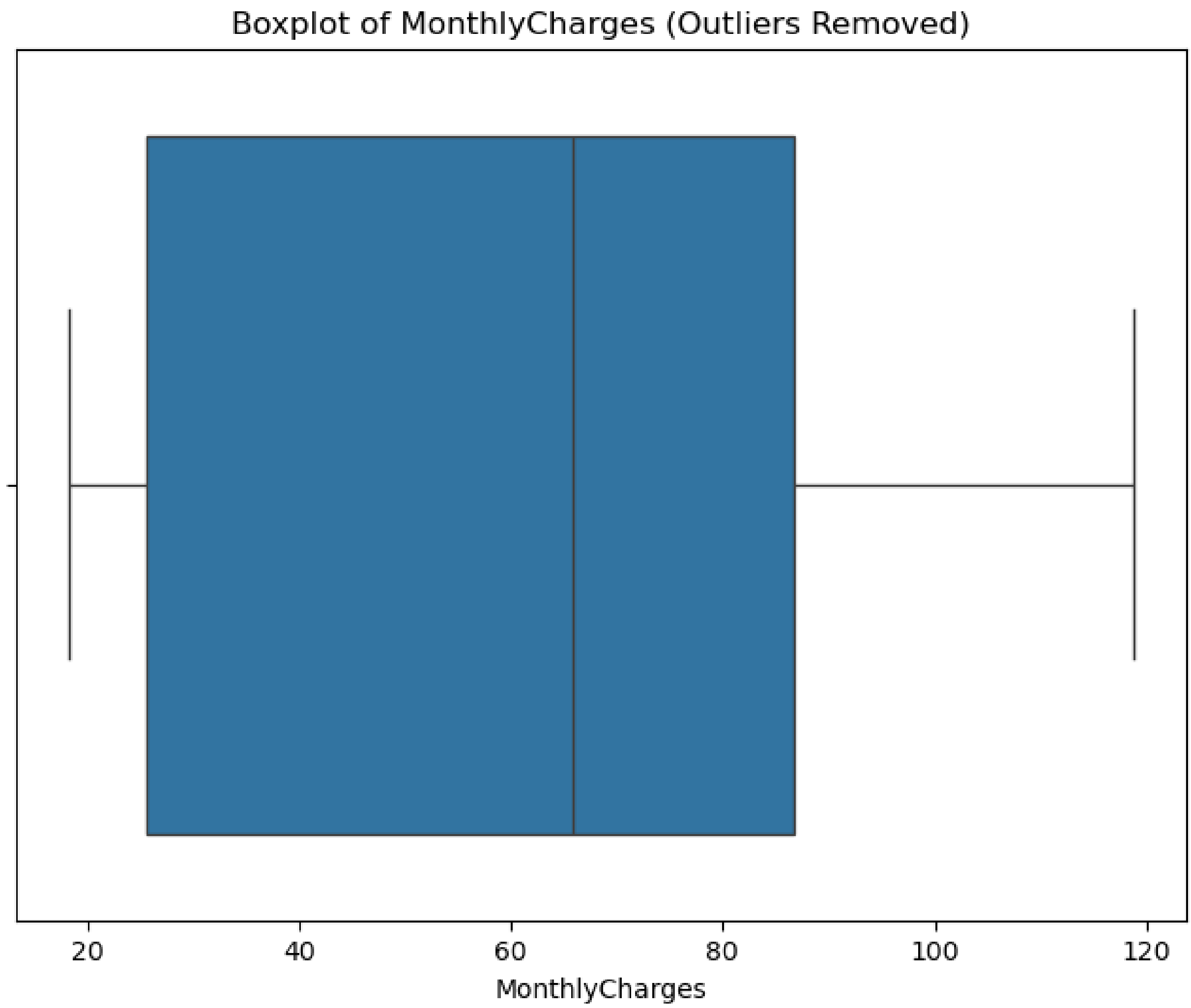
In [12]:

```
# IQR for outlier removal
def remove_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df_filtered = df[(df[column] >= lower_bound) & (df[column] <=
upper_bound)]
    return df_filtered

# Iterate through numerical features and remove outliers
numerical_features = df.select_dtypes(include=['number']).columns
df_filtered = df.copy() # Create a copy to avoid modifying the original df
in place within the loop

for col in numerical_features:
    df_filtered = remove_outliers_iqr(df_filtered, col) # Apply outlier
removal to the copied dataframe
```

```
# Verify outlier removal by plotting boxplots again
for col in numerical_features:
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=df_filtered[col])  # Use the filtered DataFrame
    plt.title(f'Boxplot of {col} (Outliers Removed)')
    plt.show()

df = df_filtered # Update the original dataframe with the filtered version
```




Boxplot of Tenure (Outliers Removed)
0
10
20
30
40
50
60
70
Tenure


Boxplot of MonthlyCharges (Outliers Removed)
20
40
60
80
100
120
MonthlyCharges

In [ ]:

In [ ]: