```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\programdata\anaconda3\
lib\site-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\programdata\
anaconda3\lib\site-packages (from seaborn) (2.1.3)
Requirement already satisfied: pandas>=1.2 in c:\programdata\
anaconda3\lib\site-packages (from seaborn) (2.2.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\
programdata\anaconda3\lib\site-packages (from seaborn) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(4.55.3)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(1.4.8)
Requirement already satisfied: packaging>=20.0 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(24.2)
Requirement already satisfied: pillow>=8 in c:\programdata\anaconda3\
lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\
anaconda3\lib\site-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\programdata\
anaconda3\lib\site-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\
lib\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4-
>seaborn) (1.17.0)
```

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

```
df = pd.read_excel(r"C:\Users\Lenovo\Downloads\train.csv.xlsx")
df.head()
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
```

```
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  
SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0  
1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0  
1
2                             Heikkinen, Miss. Laina  female  26.0  
0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0  
1
4                           Allen, Mr. William Henry    male  35.0  
0

   Parch            Ticket     Fare Cabin Embarked  
0      0         A/5 21171   7.2500     0        S  
1      0          PC 17599  71.2833   C85        C  
2      0  STON/O2. 3101282   7.9250     0        S  
3      0            113803  53.1000  C123        S  
4      0            373450   8.0500     0        S  
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        891 non-null    object 
 11  Embarked     891 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

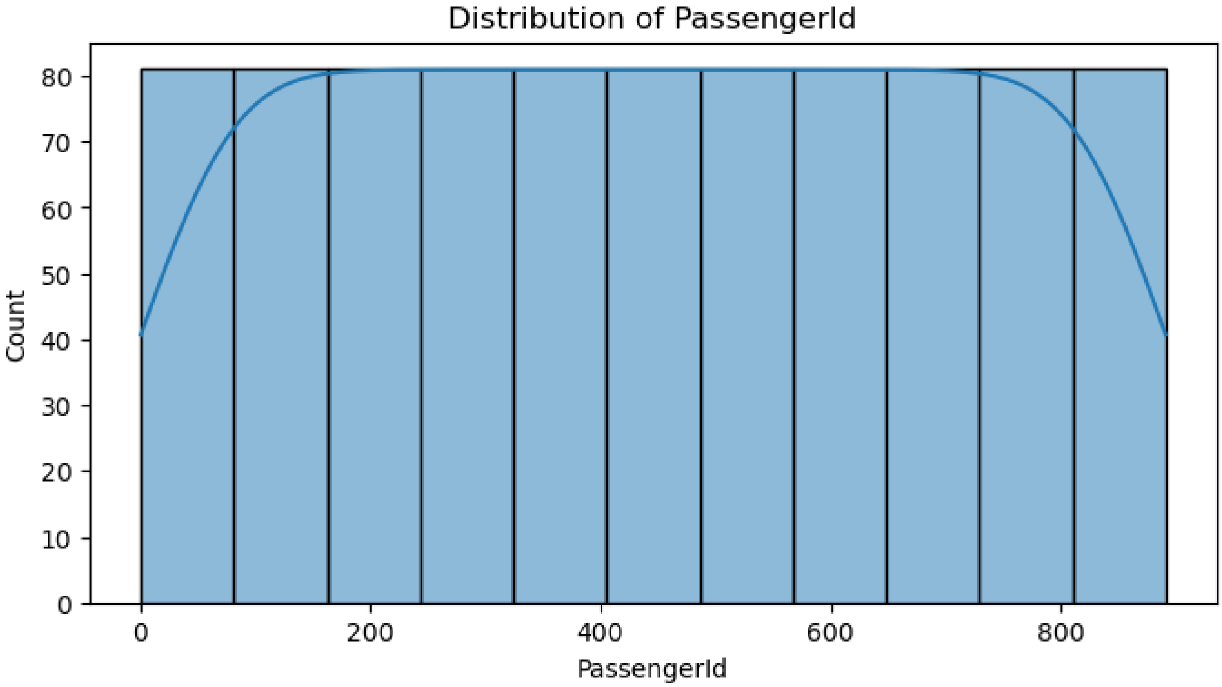
```
df.describe()
```

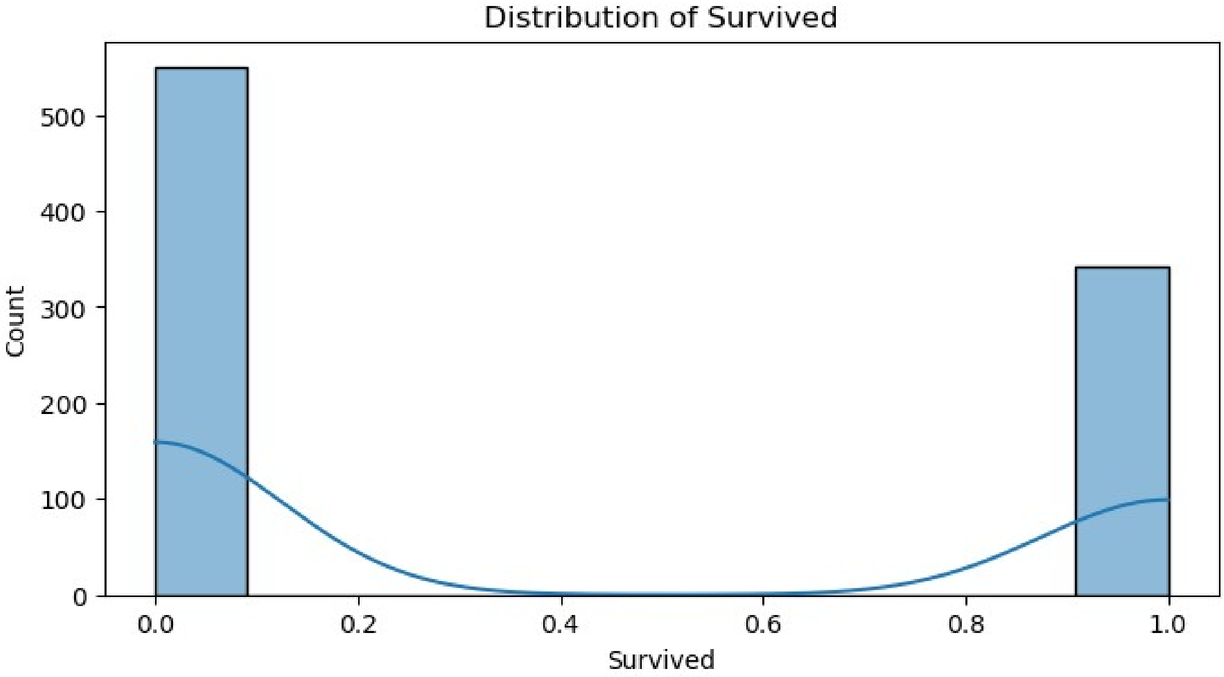
```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  891.000000  891.000000   
mean    446.000000    0.383838    2.308642   28.566970    0.523008   
std     257.353842    0.486592    0.836071   13.199572    1.102743   
min       1.000000    0.000000    1.000000    0.420000    0.000000   
25%     223.500000    0.000000    2.000000   22.000000    0.000000   
50%     446.000000    0.000000    3.000000   24.000000    0.000000   
75%     668.500000    1.000000    3.000000   35.000000    1.000000   
max     891.000000    1.000000    3.000000   80.000000    8.000000   

            Parch        Fare  
count  891.000000  891.000000  
mean     0.381594   32.204208  
std      0.806057   49.693429  
min      0.000000    0.000000  
25%      0.000000    7.910400  
50%      0.000000   14.454200  
75%      0.000000   31.000000  
max      6.000000  512.329200  
```

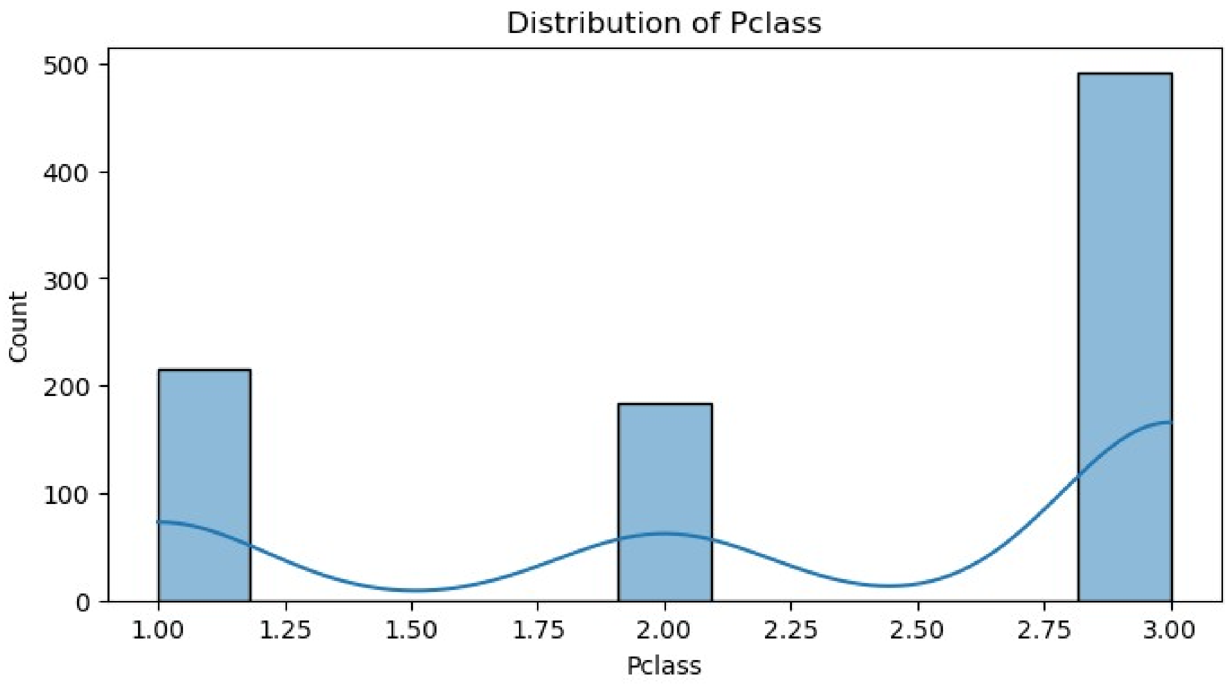
```
df['Age'].value_counts()
```

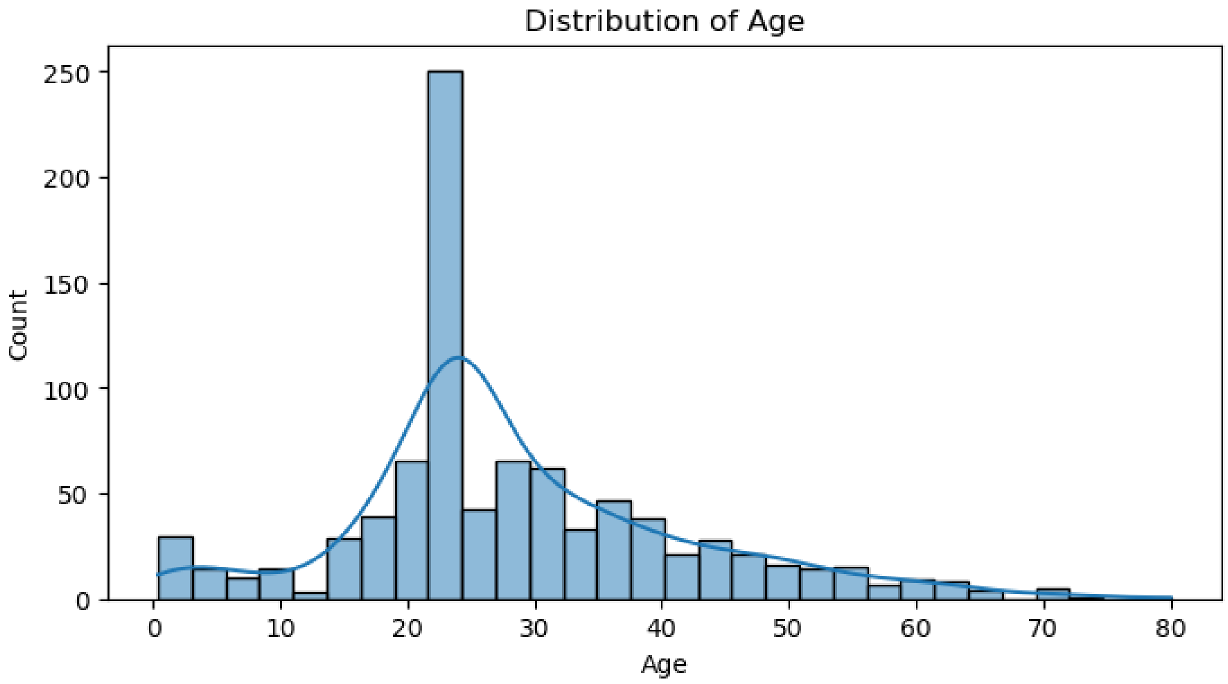
```
Age
24.00    207
22.00     27
18.00     26
28.00     25
30.00     25
        ... 
24.50      1
0.67       1
0.42       1
34.50      1
74.00      1
Name: count, Length: 88, dtype: int64
```

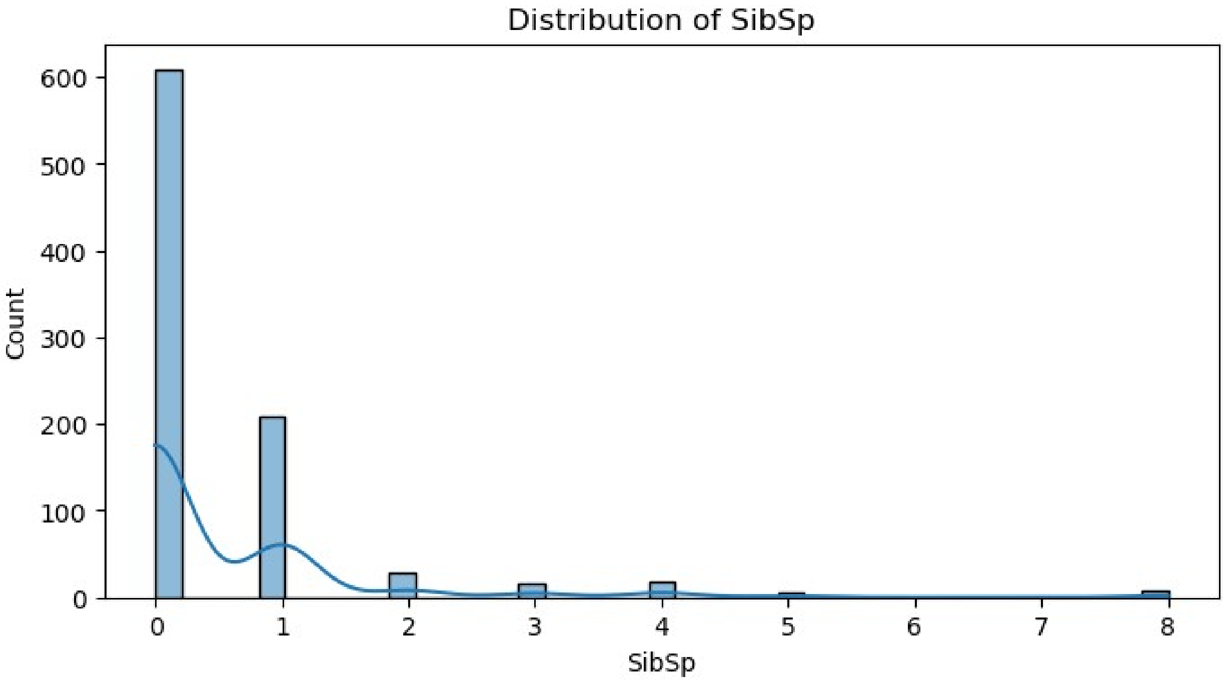
```
num_cols = df.select_dtypes(include=['float64', 'int64']).columns

for col in num_cols:
    plt.figure(figsize=(8, 4))
    sns.histplot(df[col], kde=True)
    plt.title(f'Distribution of {col}')
    plt.show()
```

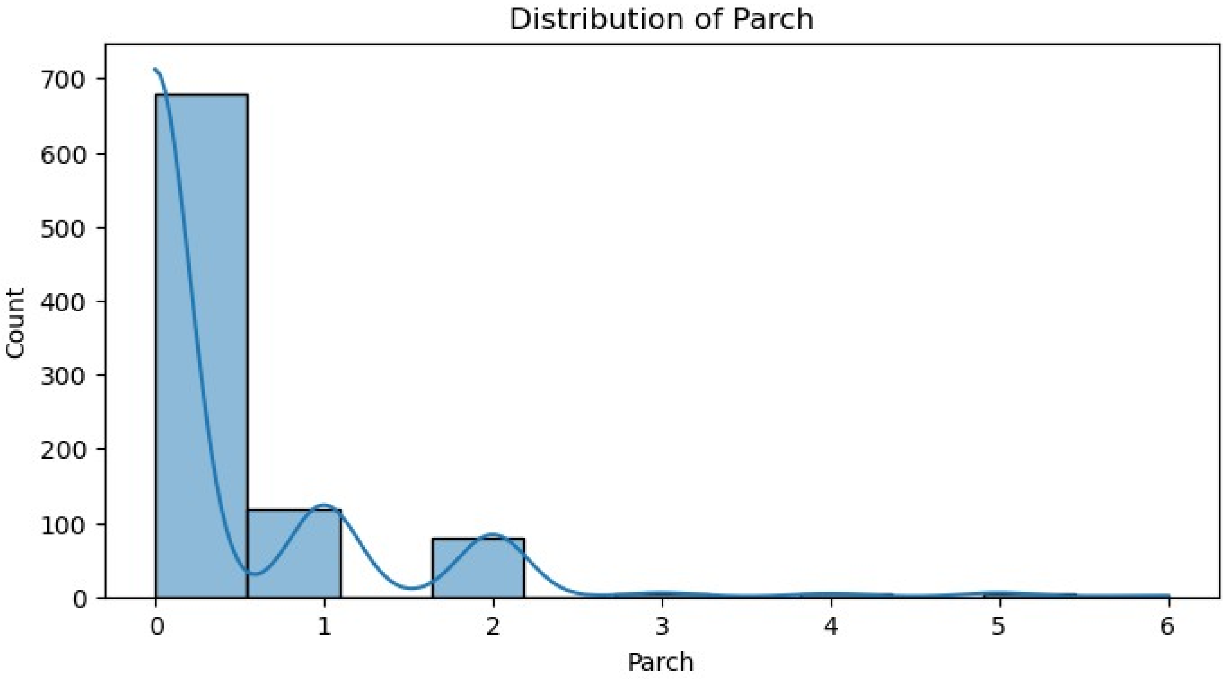

Distribution of PassengerId
Count
0
10
20
30
40
50
60
70
80
0
200
400
600
800
PassengerId

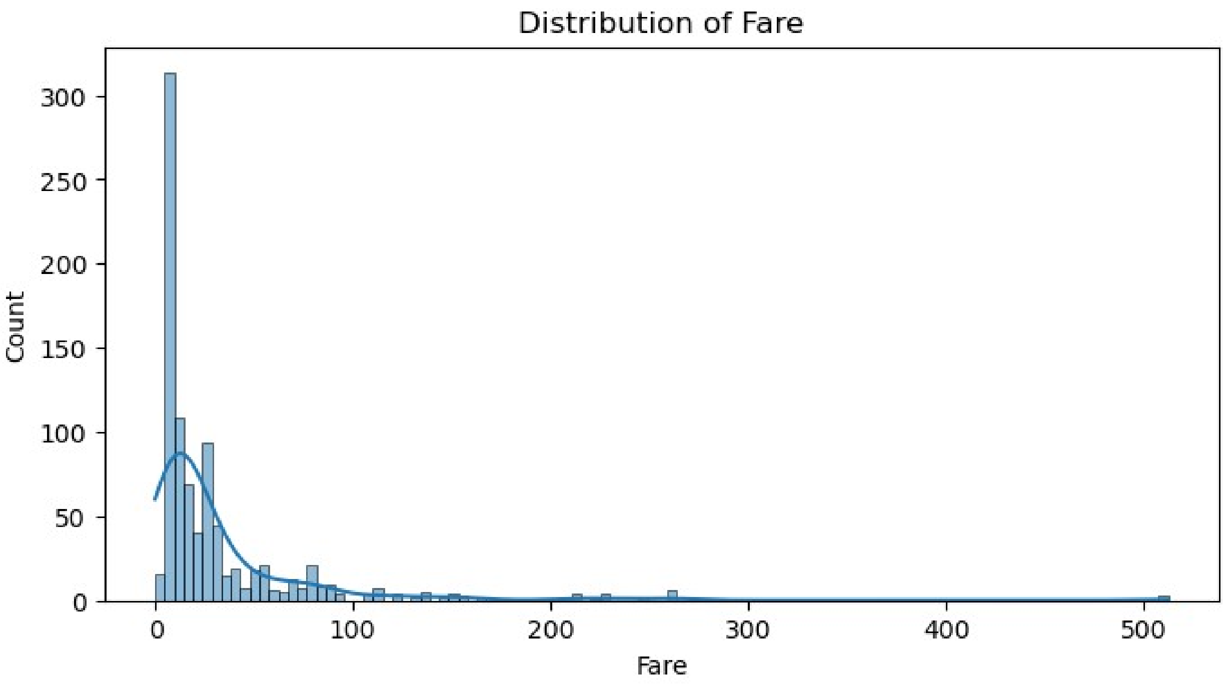

Distribution of Survived
Count
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
Survived


Distribution of Pclass
Count
500
400
300
200
100
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Pclass


Distribution of Age
Count
250
200
150
100
50
0
0
10
20
30
40
50
60
70
80
Age


Distribution of SibSp
Count
0
100
200
300
400
500
600
SibSp
0
1
2
3
4
5
6
7
8

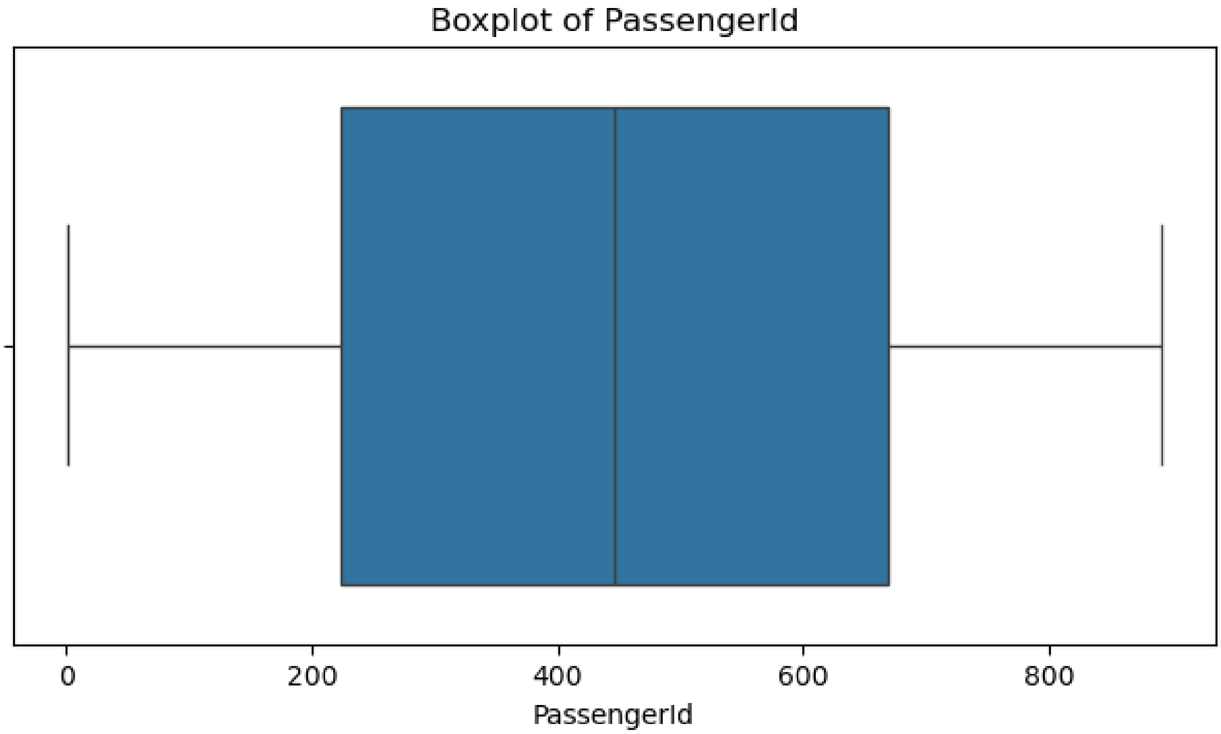

Distribution of Parch
Count
0
100
200
300
400
500
600
700
Parch
0
1
2
3
4
5
6

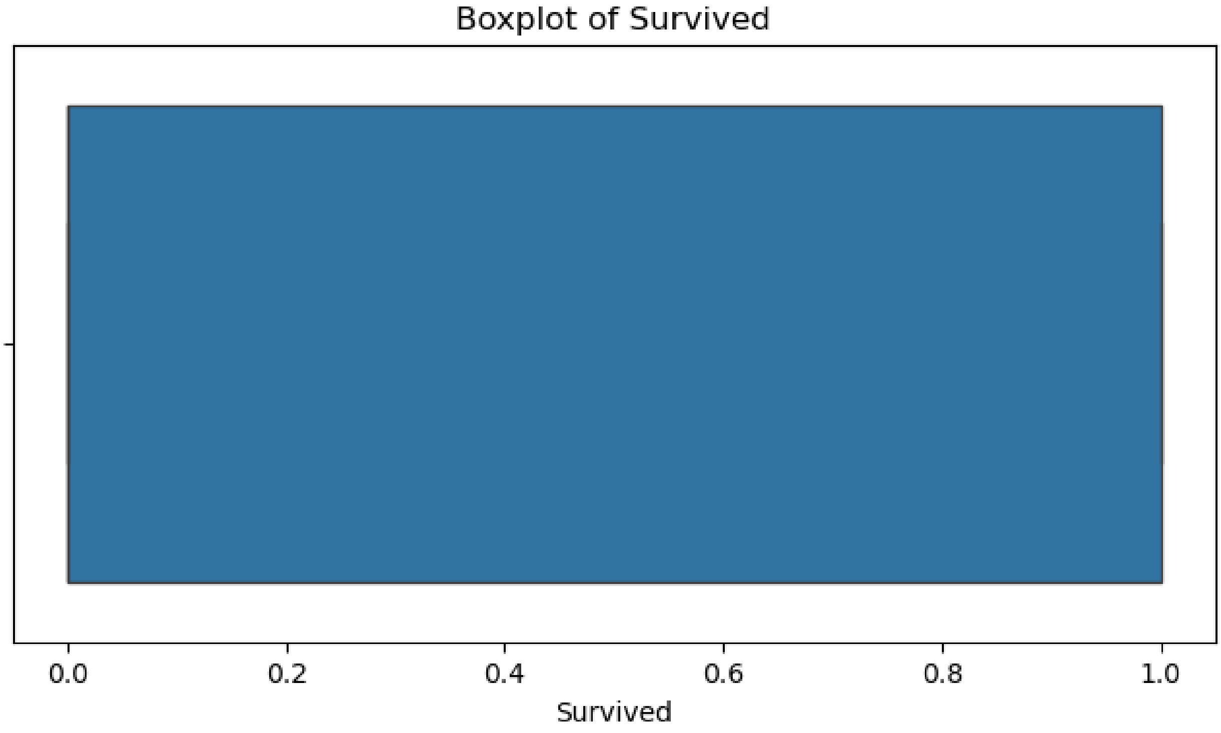


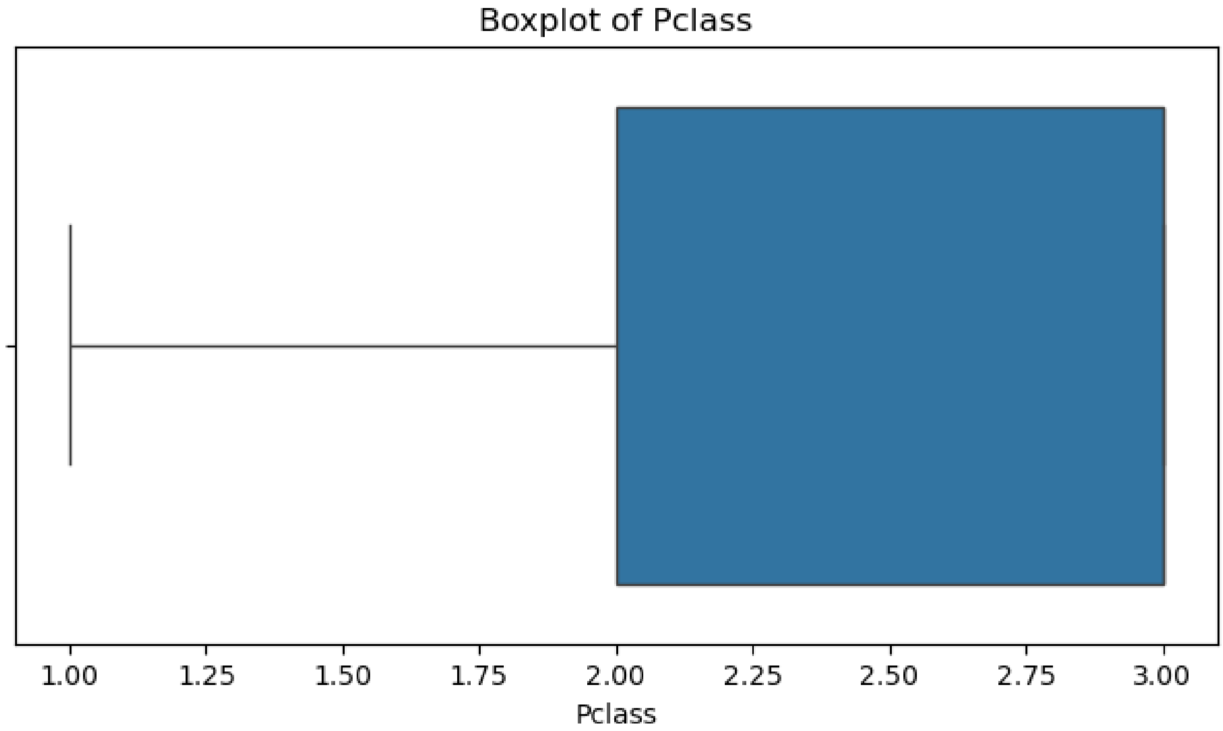
#Each histogram shows the distribution of a numerical column. Note the shape — check if it’s normal, skewed, or has multiple peaks.

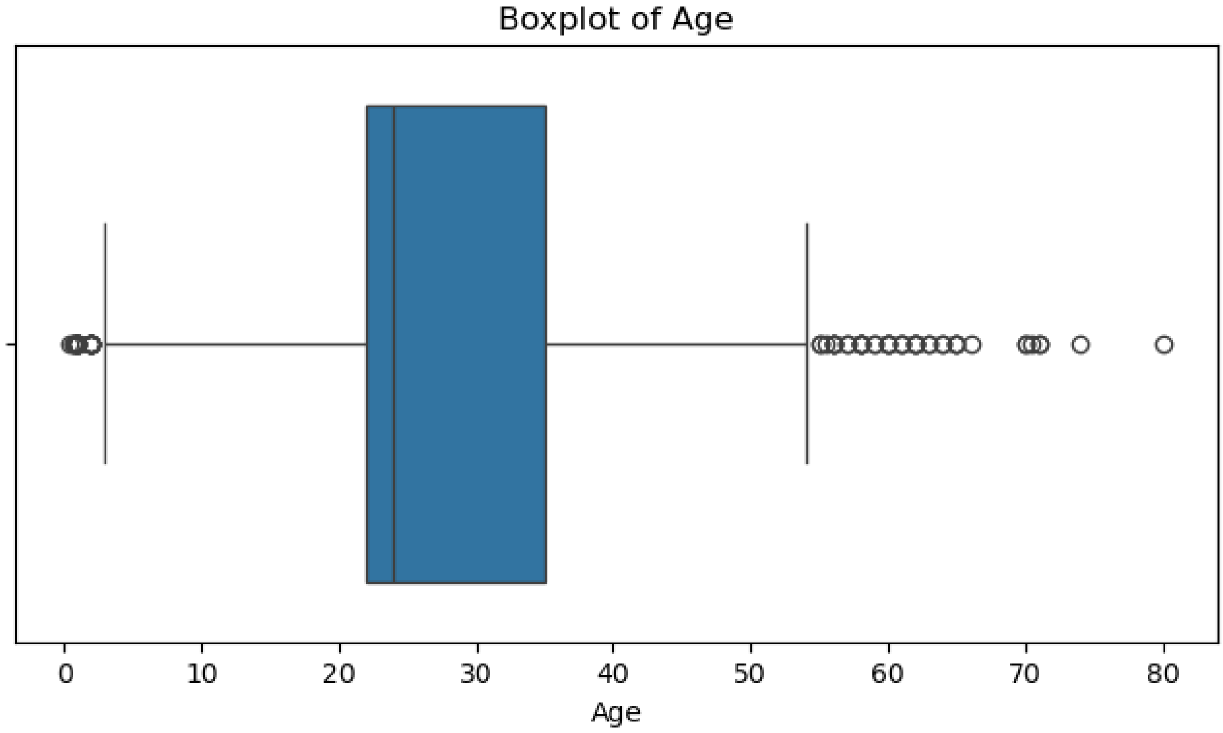
#This helps understand how values are spread and if there are unusual patterns.

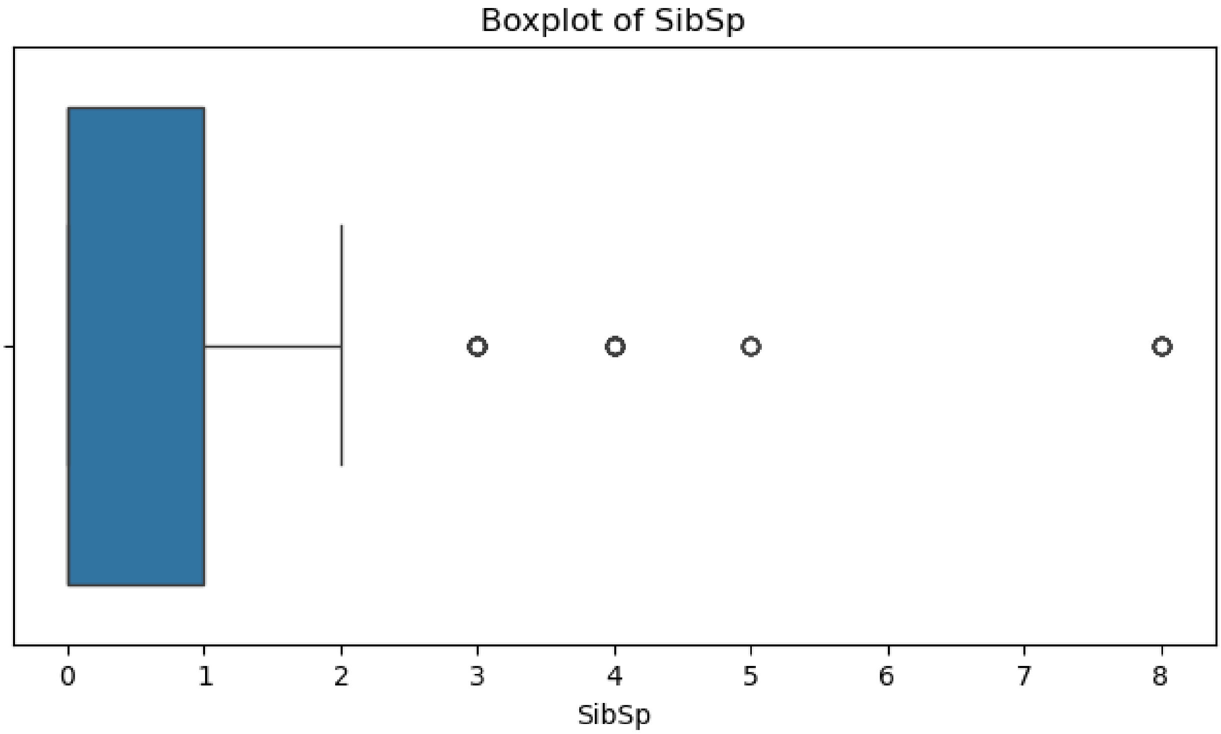
```
for col in num_cols:
    plt.figure(figsize=(8, 4))
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot of {col}')
    plt.show()
```


Boxplot of PassengerId
0
200
400
600
800
PassengerId

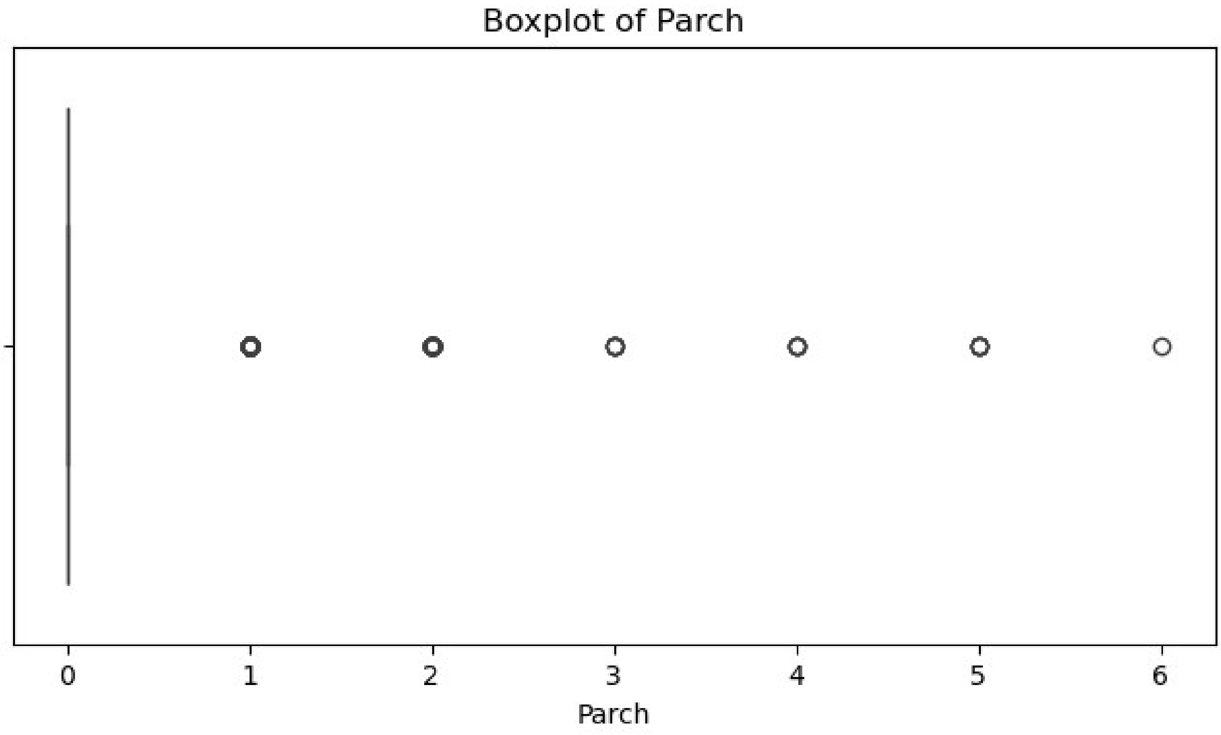

Boxplot of Survived
0.0
0.2
0.4
0.6
0.8
1.0
Survived


Boxplot of Pclass
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Pclass

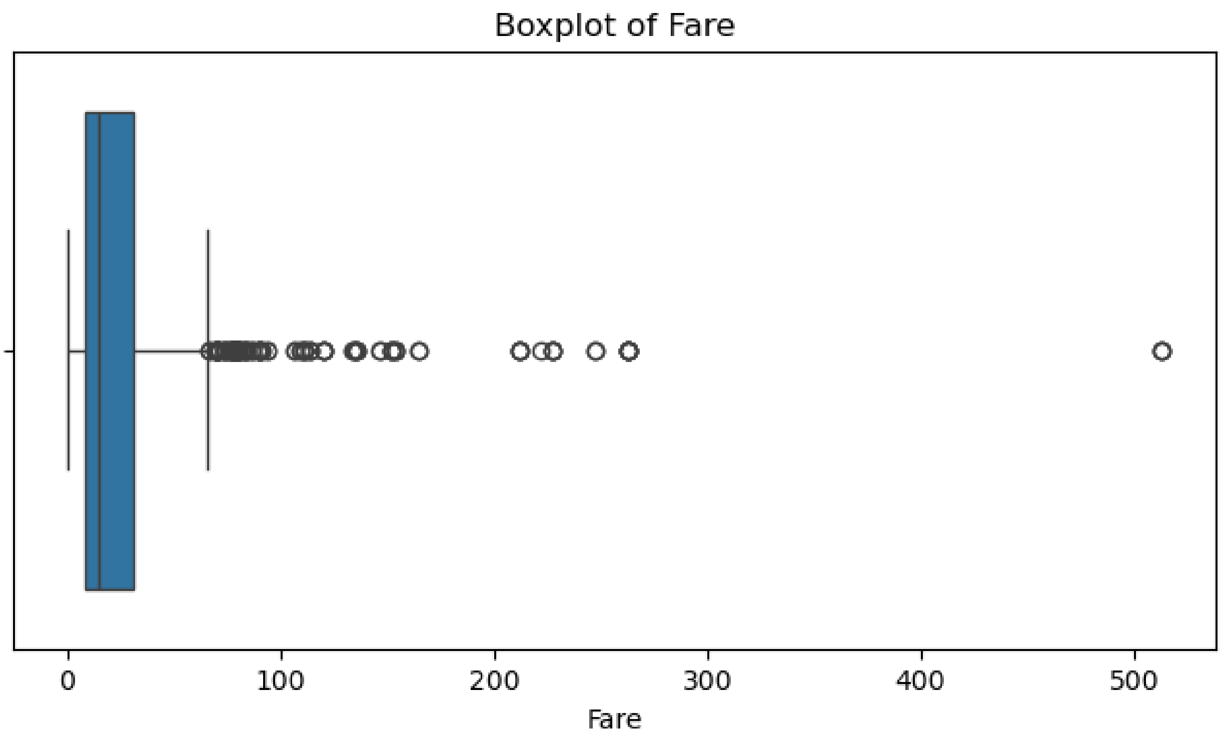

Boxplot of Age
0
10
20
30
40
50
60
70
80
Age



Boxplot of SibSp



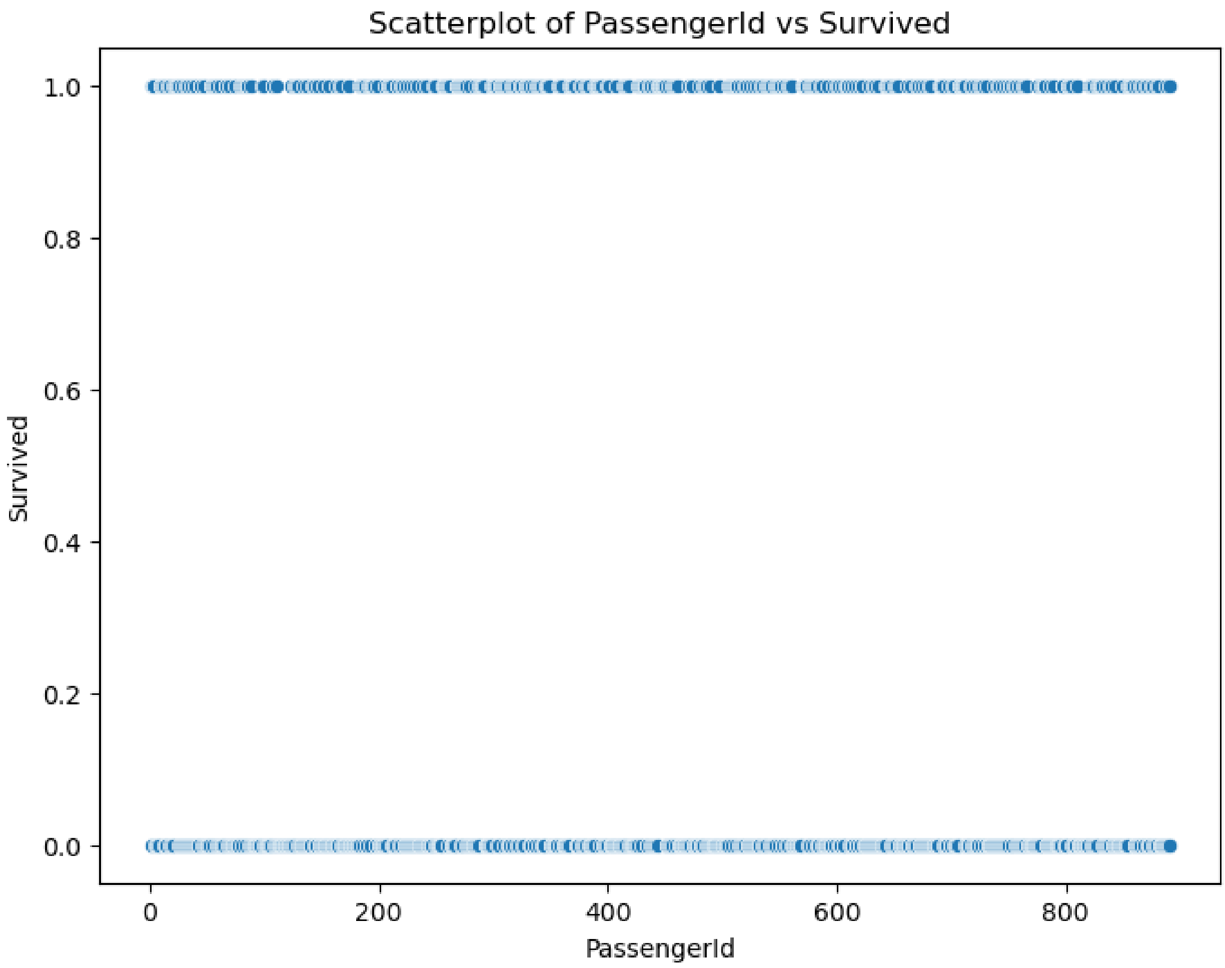
Boxplot of Parch



Each boxplot shows the spread and outliers for each numerical column.

Look for whiskers and points outside them — these are possible outliers that might affect your analysis.

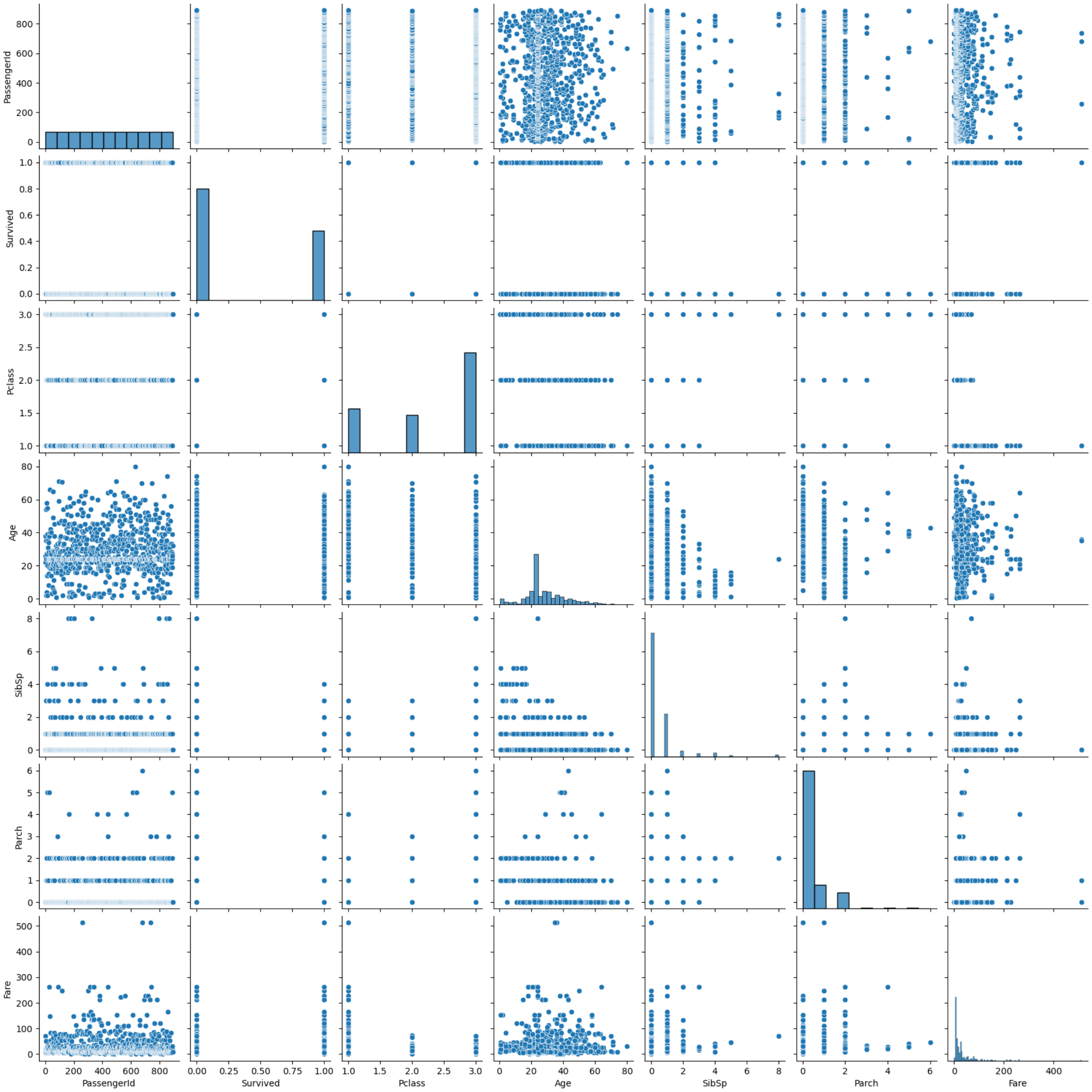
```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x=num_cols[0], y=num_cols[1])
plt.title(f'Scatterplot of {num_cols[0]} vs {num_cols[1]}')
plt.show()
```



The scatterplot compares two numerical columns. Look for trends —

e.g.,a positive or negative slope suggests correlation. Check for clusters or unusual patterns.

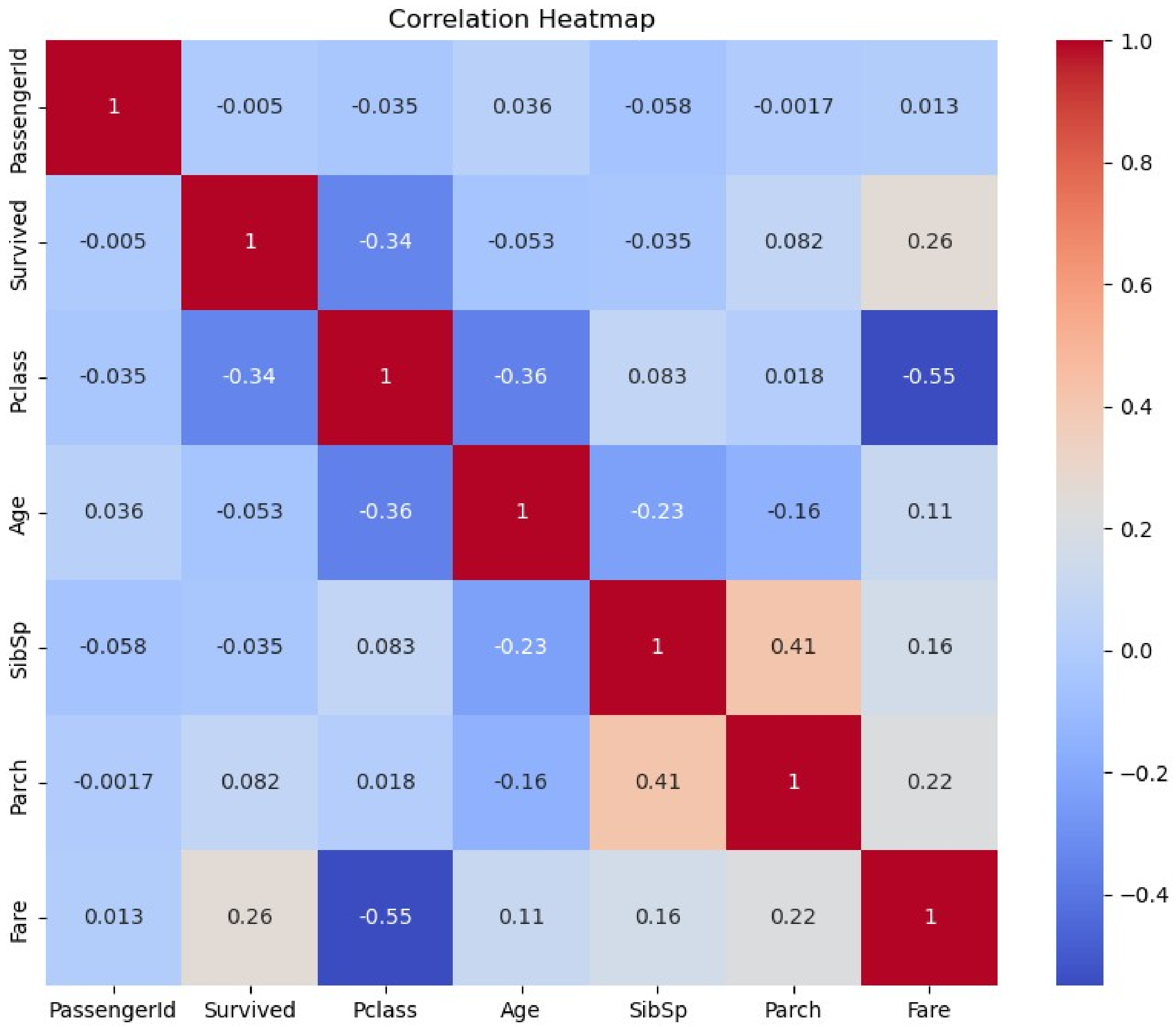
```
sns.pairplot(df[num_cols])
plt.show()
```


PassengerId
0
200
400
600
800
Survived
0.0
0.2
0.4
0.6
0.8
1.0
Pclass
1.0
1.5
2.0
2.5
3.0
Age
0
20
40
60
80
SibSp
0
2
4
6
8
Parch
0
1
2
3
4
5
6
Fare
0
100
200
300
400
500
0
200
400
600
800
PassengerId
0.00
0.25
0.50
0.75
1.00
Survived
1.0
1.5
2.0
2.5
3.0
Pclass
0
20
40
60
80
Age
0
2
4
6
8
SibSp
0
2
4
6
Parch
0
200
400
Fare

The pairplot shows scatterplots for all pairs of numerical columns, plus histograms on the diagonal.

It helps spot relationships, correlations, and clusters across multiple variable pairs at once.

```
plt.figure(figsize=(10, 8))
corr = df[num_cols].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



Correlation Heatmap

The heatmap shows correlation coefficients between numerical columns.

Strong positive (near +1) or negative (near -1) values mean strong relationships.

This helps choose important variables for deeper analysis.

```
numeric_df = df.select_dtypes(include=['number'])

correlation_matrix = numeric_df.corr()
print("Correlation Matrix:")
print(correlation_matrix)

Correlation Matrix:
             PassengerId  Survived    Pclass       Age     SibSp 
Parch  \
PassengerId     1.000000 -0.005007 -0.035144  0.036186 -0.057527 -
0.001652
Survived       -0.005007  1.000000 -0.338481 -0.052872 -0.035322 
0.081629
Pclass         -0.035144 -0.338481  1.000000 -0.356187  0.083081 
0.018443
Age             0.036186 -0.052872 -0.356187  1.000000 -0.232411 -
0.155118
SibSp          -0.057527 -0.035322  0.083081 -0.232411  1.000000 
0.414838
Parch          -0.001652  0.081629  0.018443 -0.155118  0.414838 
1.000000
Fare            0.012658  0.257307 -0.549500  0.107554  0.159651 
0.216225

                 Fare
PassengerId  0.012658
Survived     0.257307
Pclass      -0.549500
Age          0.107554
SibSp        0.159651
Parch        0.216225
Fare         1.000000
```

```
means = numeric_df.mean()
print("\nMeans of Numerical Columns:")
print(means)
```

```
Means of Numerical Columns:
PassengerId    446.000000
Survived         0.383838
Pclass           2.308642
Age             28.566970
SibSp            0.523008
Parch            0.381594
Fare            32.204208
dtype: float64
```

```
medians = numeric_df.median()
print("\nMedians of Numerical Columns:")
print(medians)
```

```
Medians of Numerical Columns:
PassengerId    446.0000
Survived         0.0000
Pclass           3.0000
Age             24.0000
SibSp            0.0000
Parch            0.0000
Fare            14.4542
dtype: float64
```