

B.VENKATA SAI KUMAR

21BCE9341

▼ Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
```

▼ Importing Dataset

```
df = pd.read_csv('Titanic_Dataset.csv')
```

```
print(df.shape)
```

```
(891, 12)
```

```
print(df.dtypes)
```

```
PassengerId    int64
Survived       int64
Pclass         int64
Name           object
Sex            object
Age           float64
SibSp          int64
Parch          int64
Ticket         object
Fare           float64
Cabin          object
Embarked       object
dtype: object
```

▼ Checking for null values

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

```

PassengerId    0
Survived        0
Pclass         0
Name           0
Sex            0
Age           177
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin         687
Embarked       2
dtype: int64

```

```
df.head()
```

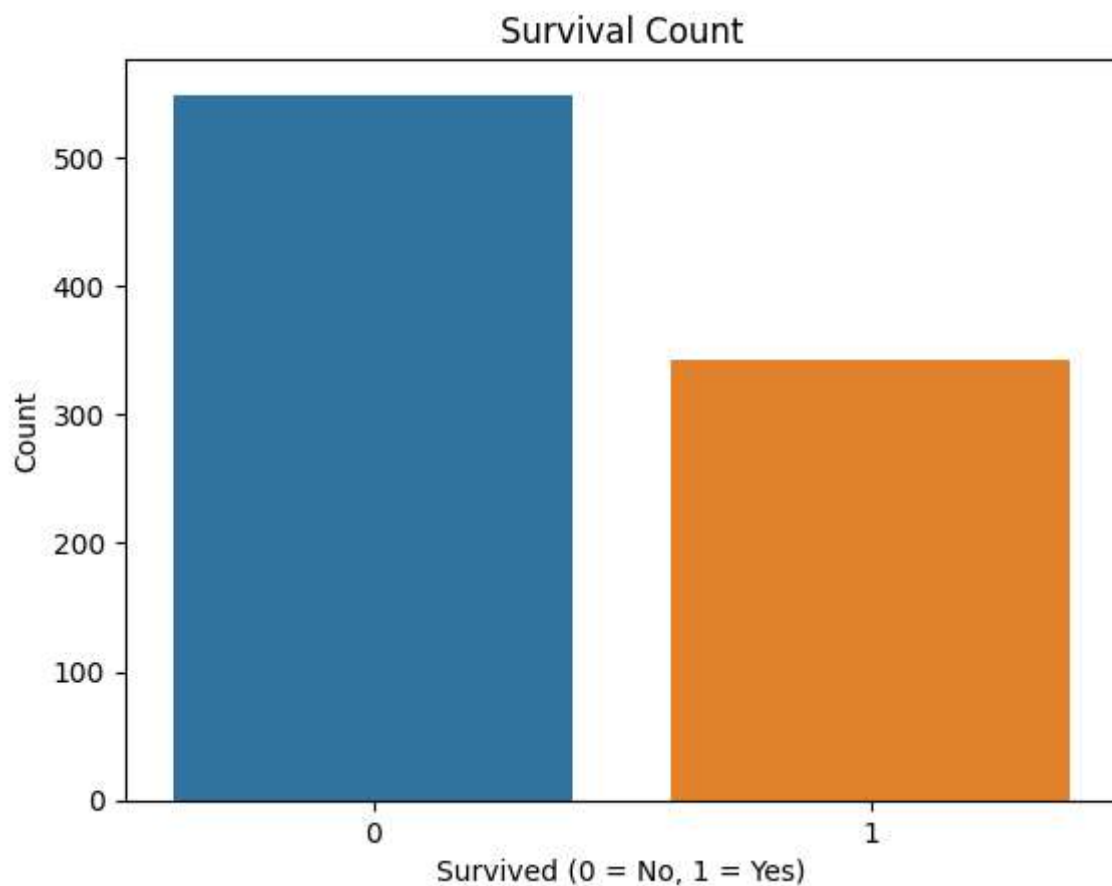
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0.0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0.0	PC 17599	14.4542
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0.0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0.0	113803	14.4542
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0.0	373450	8.0500

```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.0	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.0	14.200184
std	257.353842	0.486592	0.836071	13.002015	1.102743	0.0	7.332994
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.0	0.000000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.0	7.910400
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.0	14.454200
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.0	15.500000

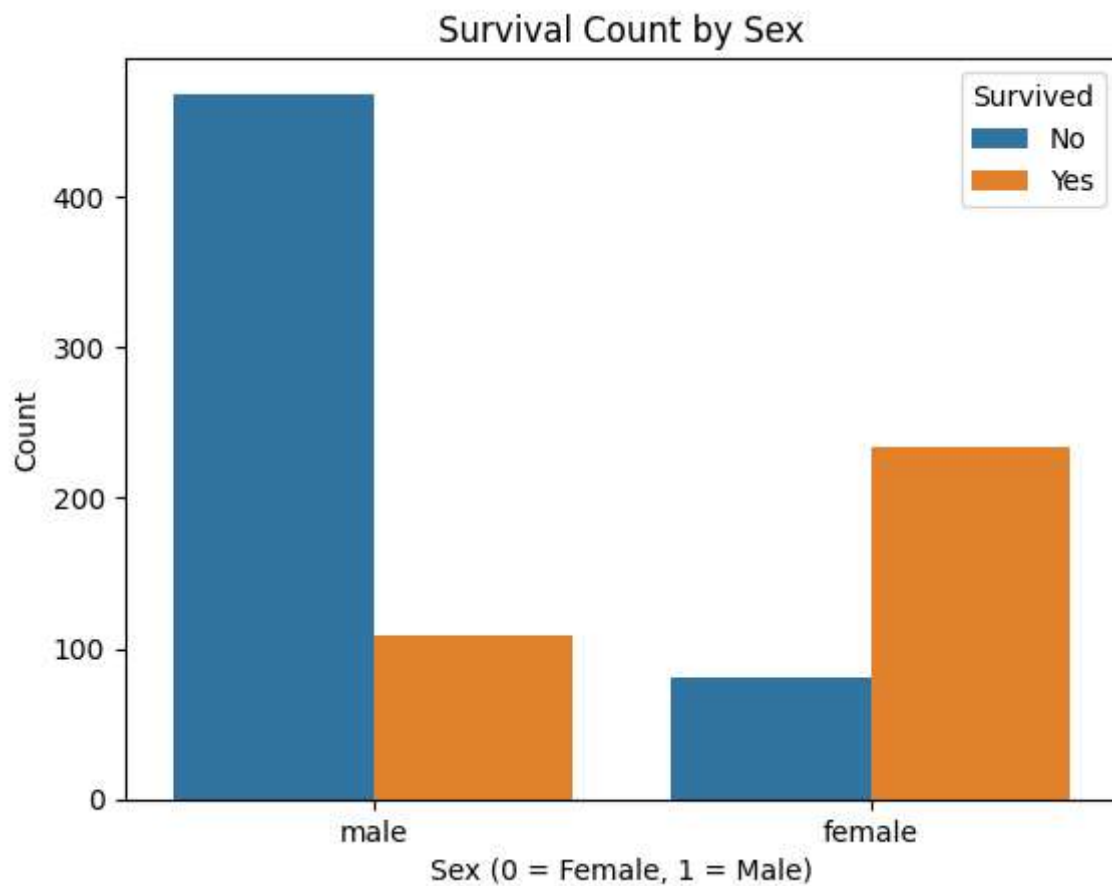
▼ Data Visualization

```
sns.countplot(x='Survived', data=df)
plt.title('Survival Count')
plt.xlabel('Survived (0 = No, 1 = Yes)')
plt.ylabel('Count')
plt.show()
```

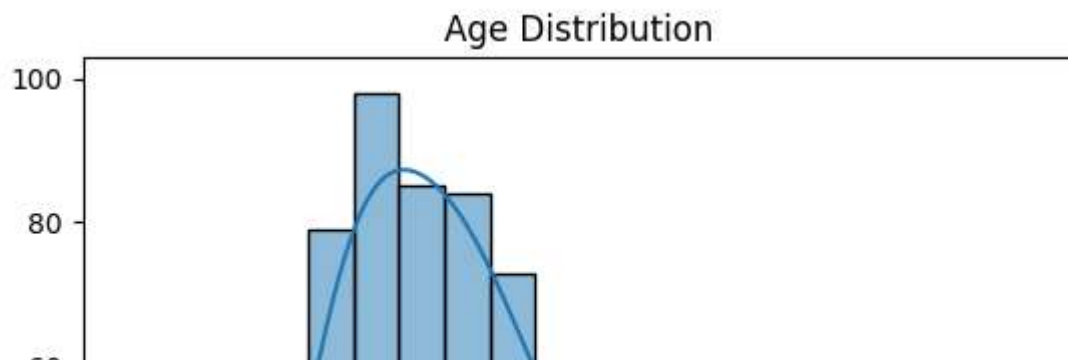


```
sns.countplot(x='Sex', hue='Survived', data=df)
plt.title('Survival Count by Sex')
```

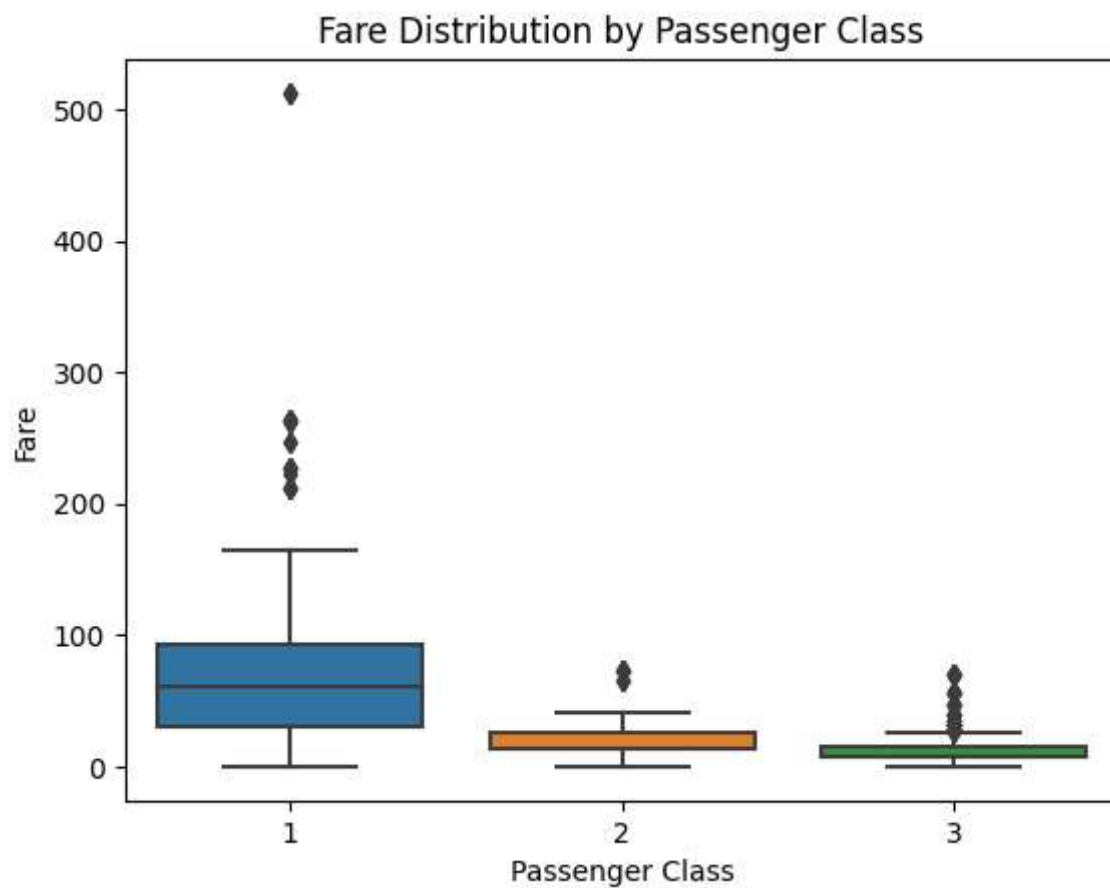
```
plt.xlabel('Sex (0 = Female, 1 = Male)')  
plt.ylabel('Count')  
plt.legend(title='Survived', labels=['No', 'Yes'])  
plt.show()
```



```
sns.histplot(df['Age'], bins=20, kde=True)  
plt.title('Age Distribution')  
plt.xlabel('Age')  
plt.ylabel('Frequency')  
plt.show()
```

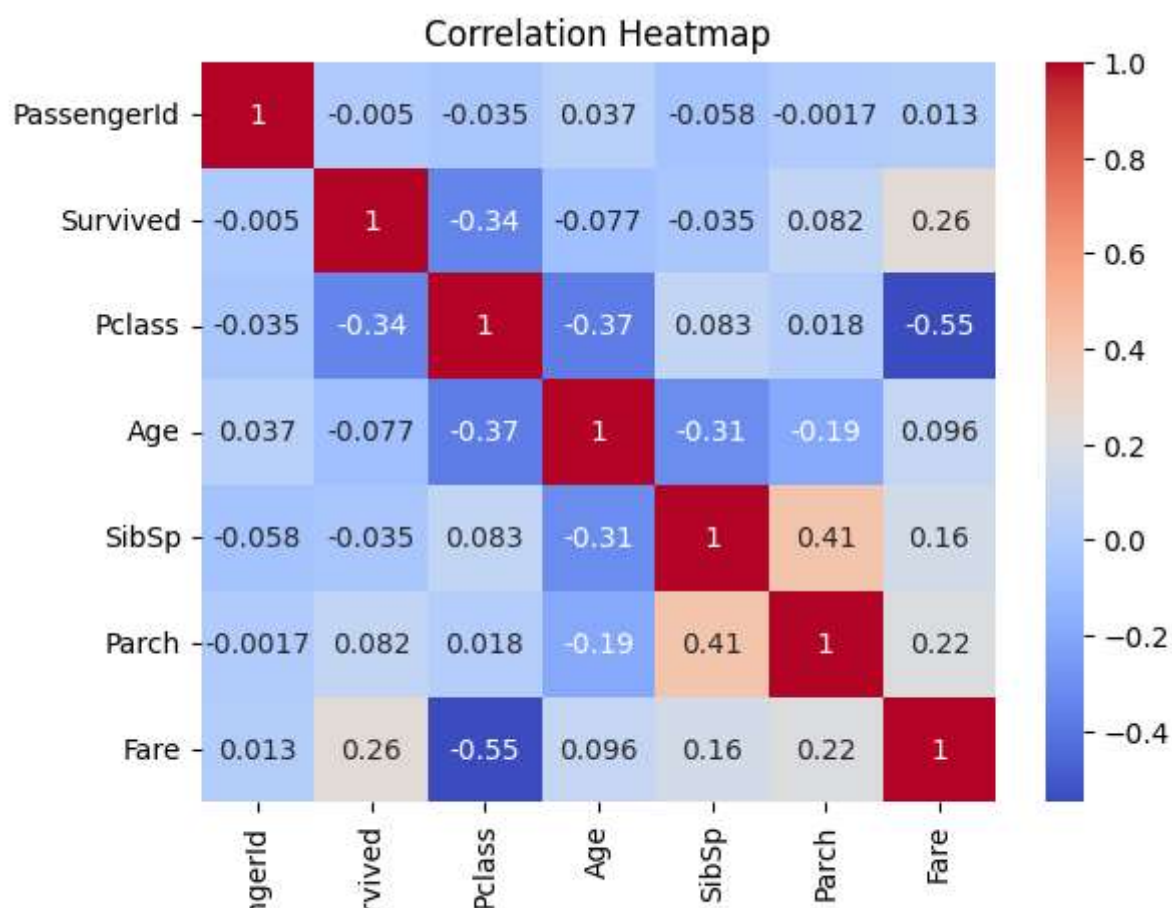


```
sns.boxplot(x='Pclass', y='Fare', data=df)
plt.title('Fare Distribution by Passenger Class')
plt.xlabel('Passenger Class')
plt.ylabel('Fare')
plt.show()
```



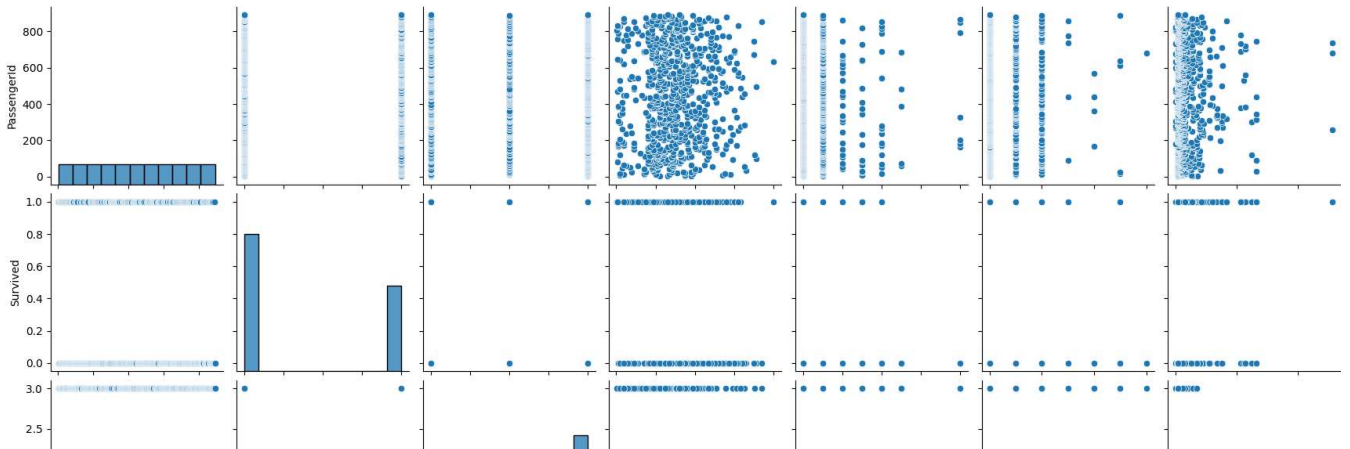
```
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

```
<ipython-input-68-182fd031f822>:1: FutureWarning: The default value of numeric_only in [
correlation_matrix = df.corr()
```



```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7be07f7cf430>
```

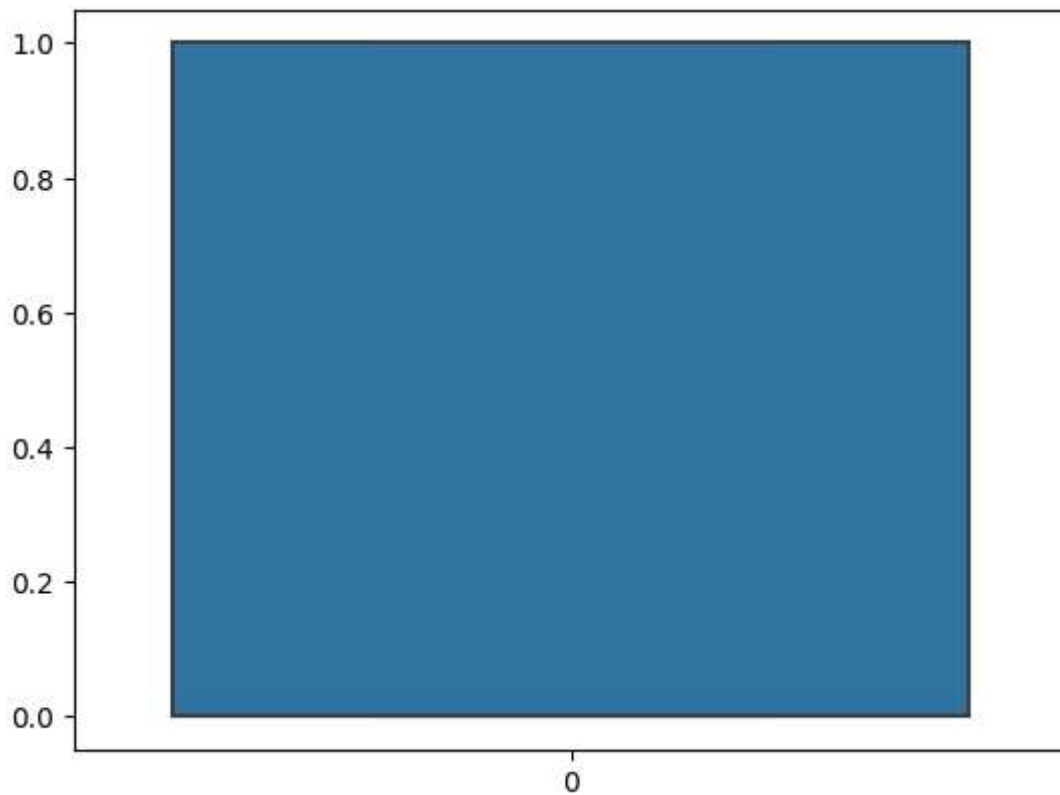


▼ Outlier Detection



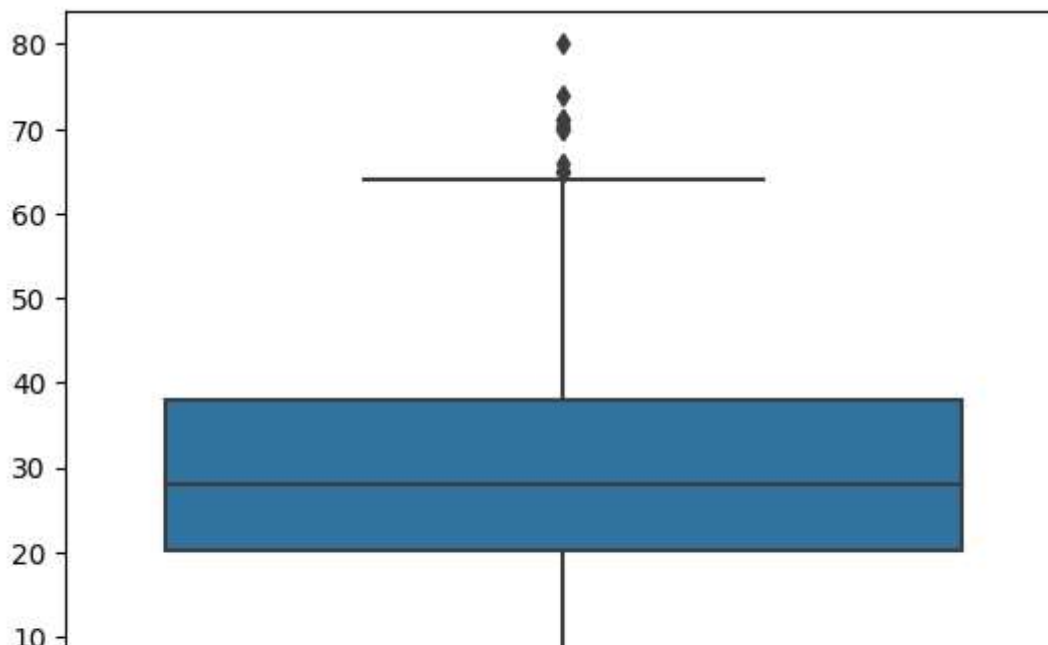
```
sns.boxplot(df.Survived)
```

```
<Axes: >
```



```
sns.boxplot(df.Age)
```

<Axes: >



```
q1 = df.Age.quantile(0.25)
q3 = df.Age.quantile(0.75)
print(q1)
print(q3)
```

```
20.125
38.0
```

```
IQR = q3-q1
print(IQR)
```

```
17.875
```

```
ul = q3+1.5*IQR
print(ul)
```

```
64.8125
```

```
ll = q1-1.5*IQR
print(ll)
```

```
-6.6875
```

```
df.median()
```

```
<ipython-input-77-6d467abf240d>:1: FutureWarning: The default value of numeric_only in I
df.median()
PassengerId    446.0000
Survived        0.0000
Pclass         3.0000
Age            28.0000
```

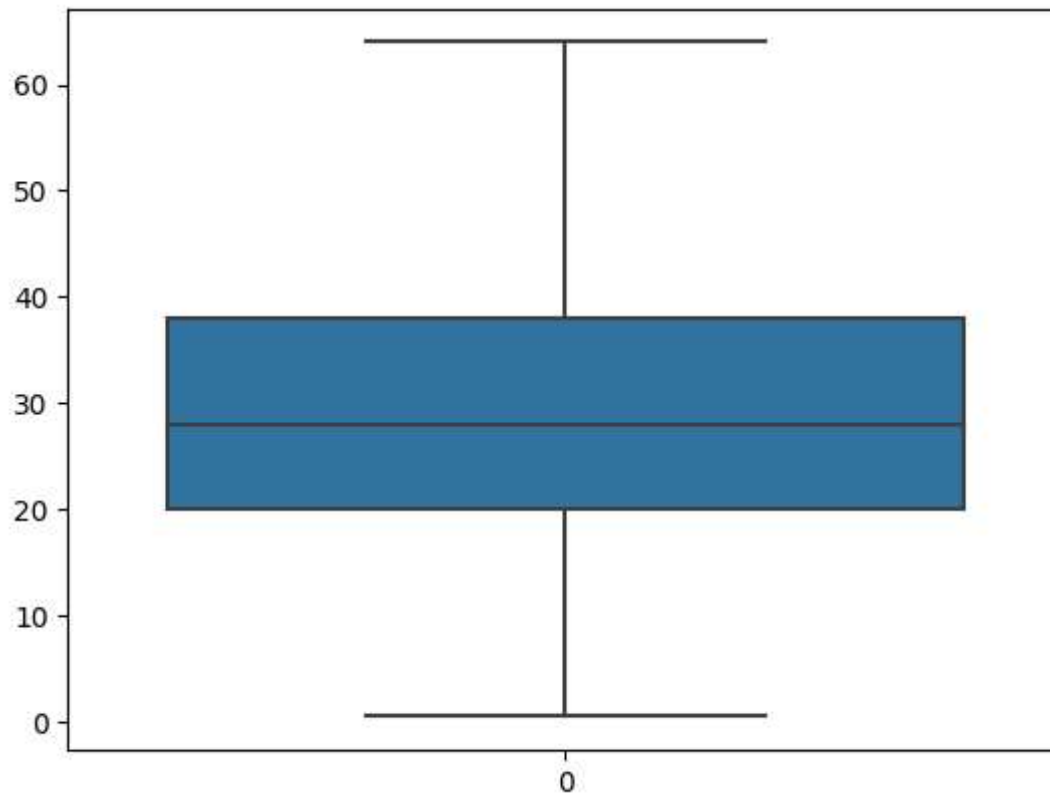


```
SibSp      0.0000  
Parch      0.0000  
Fare      14.4542  
dtype: float64
```

```
df = df[df.Age<ul]
```

```
sns.boxplot(df.Age)
```

<Axes: >



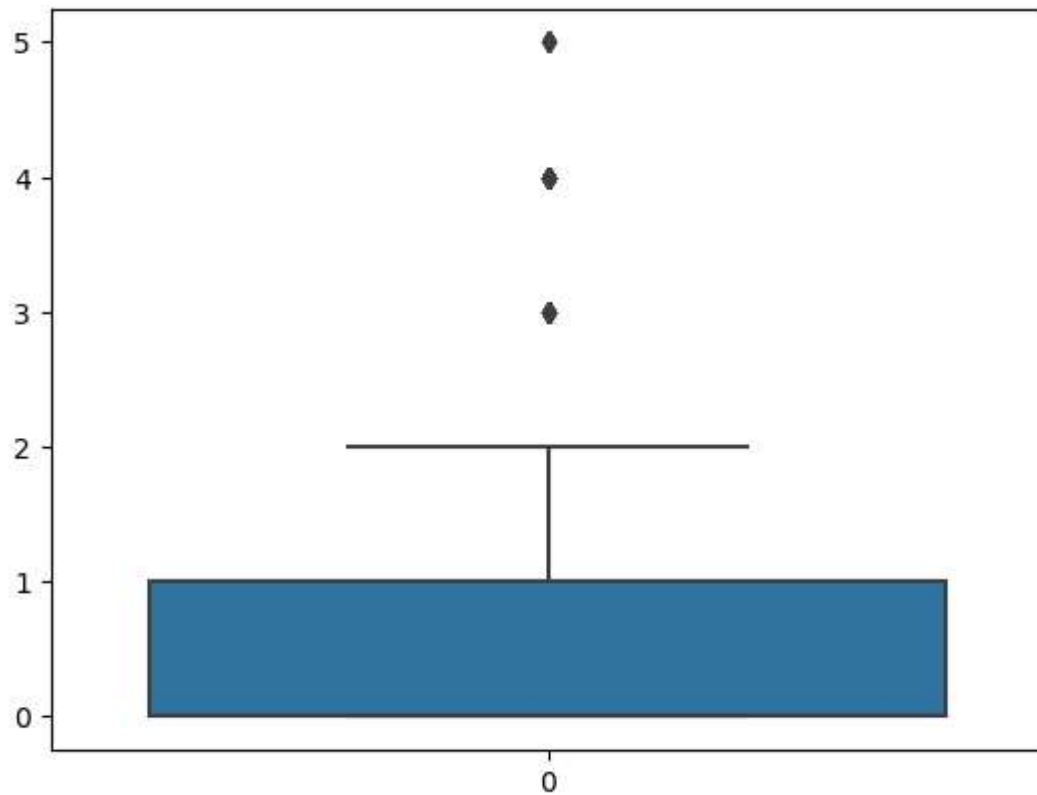
```
sns.boxplot(df.Pclass)
```

<Axes: >



```
sns.boxplot(df.SibSp)
```

<Axes: >



```
q1 = df.SibSp.quantile(0.25)
q3 = df.SibSp.quantile(0.75)
print(q1)
print(q3)
```

```
IQR = q3-q1
print(IQR)
```

```
ul = q3+1.5*IQR
print(ul)
```

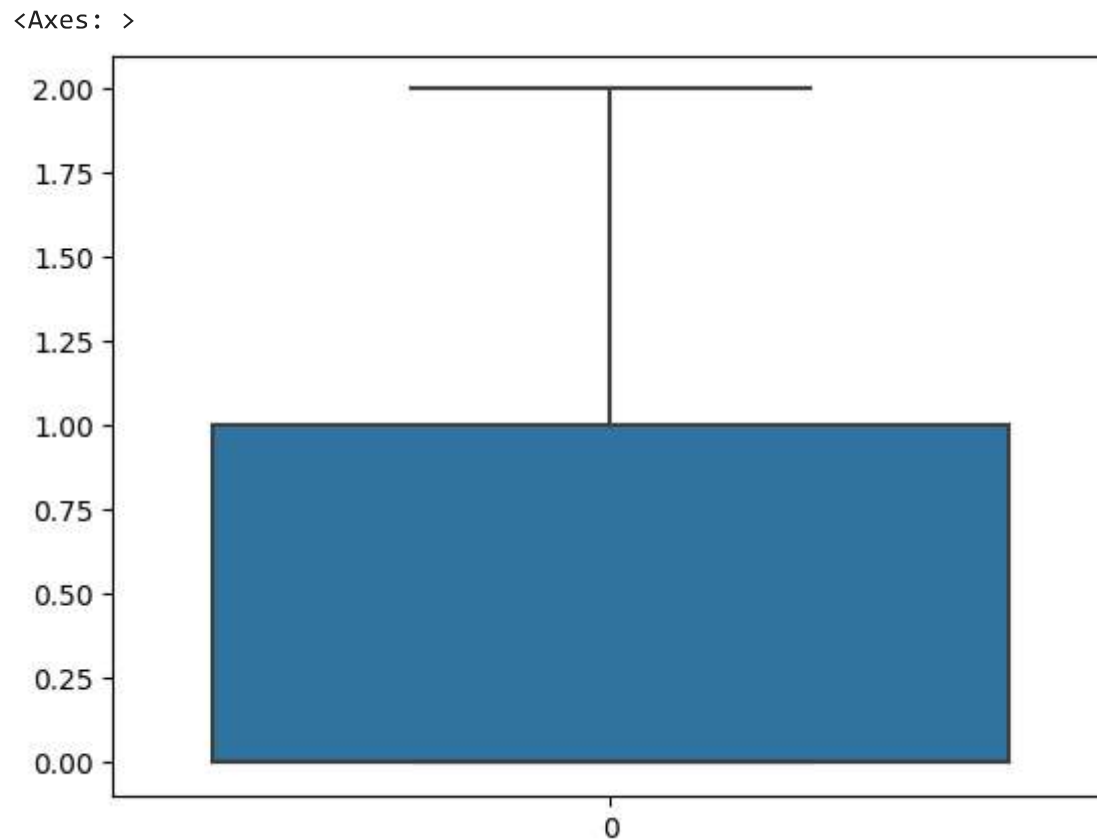
```
ll = q1-1.5*IQR  
print(ll)
```

```
df.median()
```

```
<ipython-input-4-6d467abf240d>:1: FutureWarning: The default value of numeric_only in D  
df.median()  
PassengerId      446.0000  
Survived          0.0000  
Pclass           3.0000  
Age              28.0000  
SibSp            0.0000  
Parch            0.0000  
Fare             14.4542  
dtype: float64
```

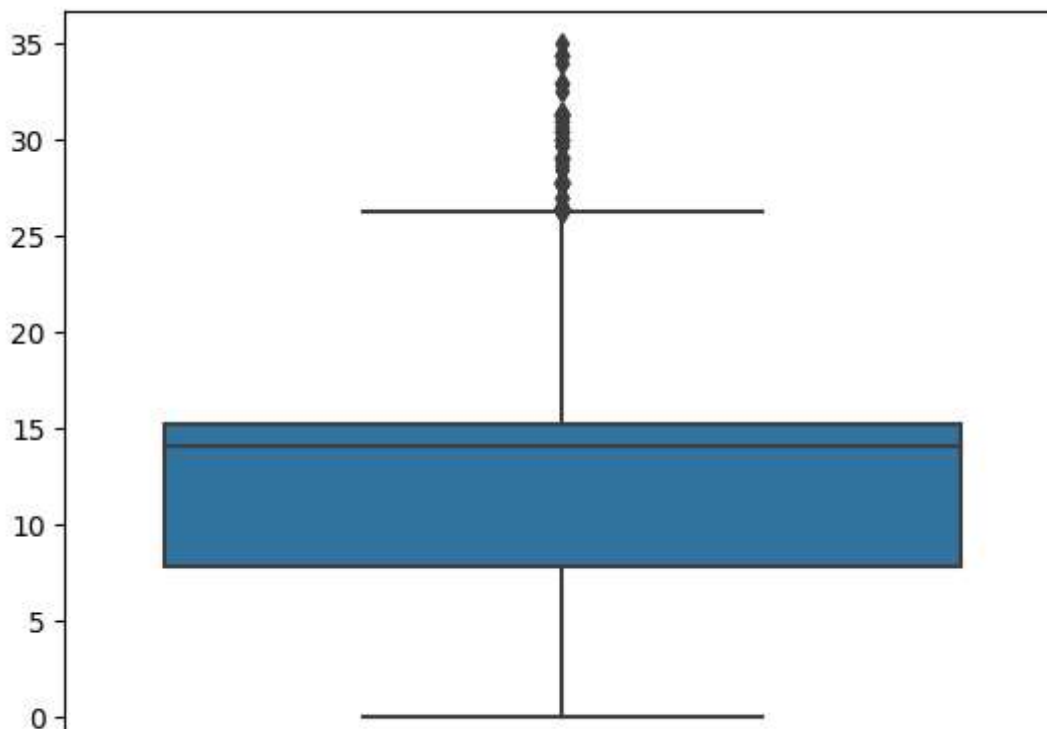
```
df = df[df.SibSp<u1]
```

```
sns.boxplot(df.SibSp)
```



```
sns.boxplot(df.Fare)
```

<Axes: >



```
q1 = df.Fare.quantile(0.25)
q3 = df.Fare.quantile(0.75)
print(q1)
print(q3)
```

```
7.9104
18.75
```

```
IQR = q3-q1
print(IQR)
ul = q3+1.5*IQR
print(ul)
ll = q1-1.5*IQR
print(ll)
```

```
10.8396
35.0094
-8.349
```

```
df.median()
```

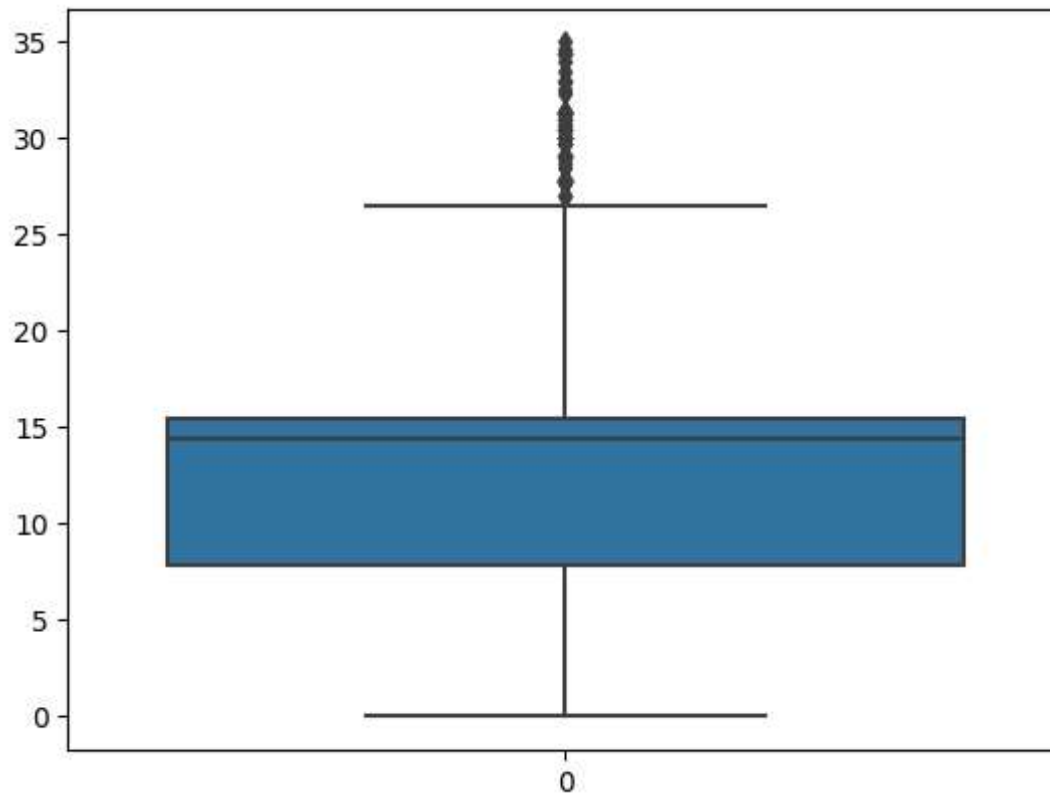
```
<ipython-input-29-6d467abf240d>:1: FutureWarning: The default value of numeric_only in I
df.median()
PassengerId    446.0000
Survived        0.0000
Pclass         3.0000
Age           28.0000
SibSp          0.0000
Parch          0.0000
```

```
Fare          14.4542  
dtype: float64
```

```
df['Fare'] = np.where(df['Fare']>u1,14.4542,df['Fare'])
```

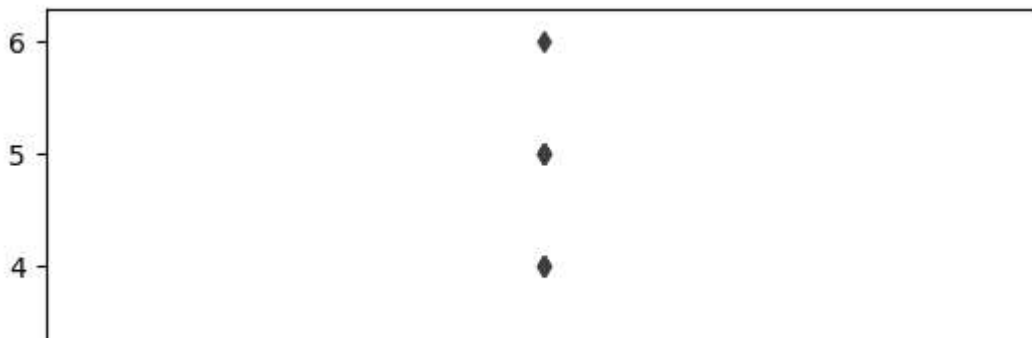
```
sns.boxplot(df.Fare)
```

<Axes: >



```
sns.boxplot(df.Parch)
```

<Axes: >



```
q1 = df.Parch.quantile(0.25)
q3 = df.Parch.quantile(0.75)
print(q1)
print(q3)
```

```
0.0
0.0
```

```
IQR = q3-q1
print(IQR)
ul = q3 + 1.5*IQR
print(ul)
ll = q1-1.5*IQR
print(ll)
```

```
0.0
0.0
0.0
```

```
df.median()
```

```
<ipython-input-17-6d467abf240d>:1: FutureWarning: The default value of numeric_only in [
df.median()
PassengerId      446.0000
Survived          0.0000
Pclass           3.0000
Age              28.0000
SibSp            0.0000
Parch            0.0000
Fare             14.4542
dtype: float64
```

```
df['Parch'] = np.where(df['Parch']>ul,0.0,df['Parch'])
```

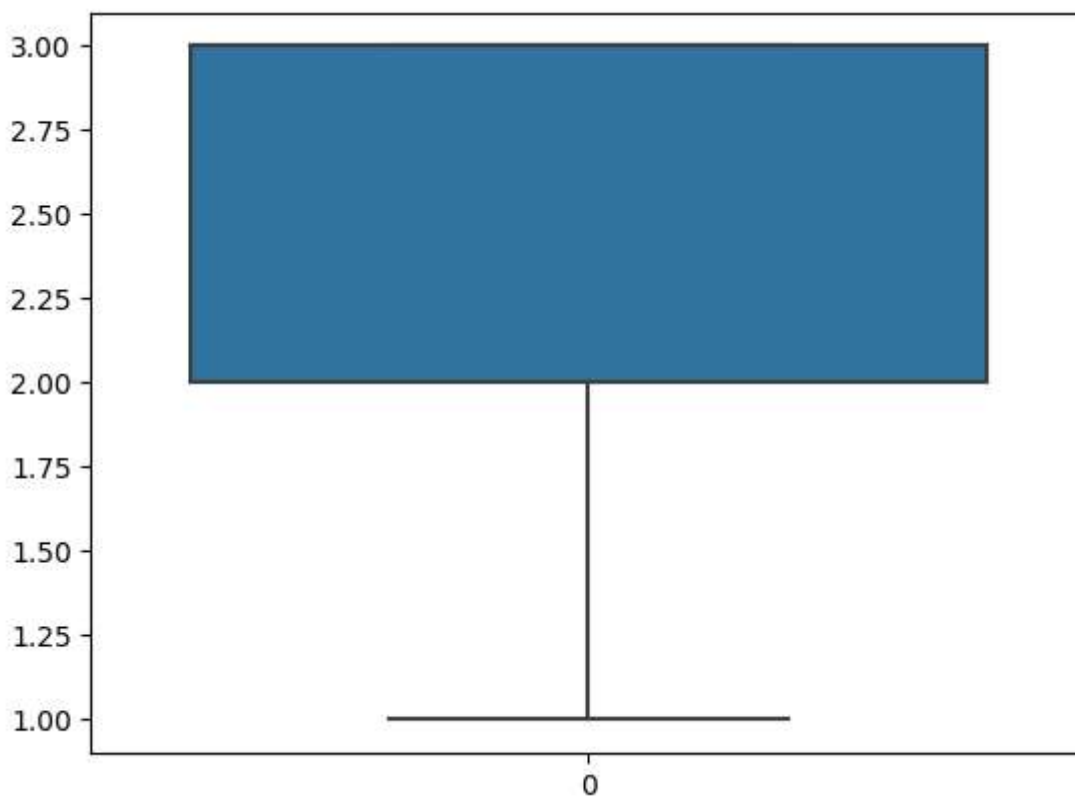
```
sns.boxplot(df.Parch)
```

<Axes: >



```
sns.boxplot(df.Pclass)
```

<Axes: >



▼ Splitting into Independent and Dependent Variables

Independent Variables

```
X = df.drop('Survived', axis=1)
print(X)
```

	PassengerId	Pclass	Name \
0	1	3	Braund, Mr. Owen Harris
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...
2	3	3	Heikkinen, Miss. Laina
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)
4	5	3	Allen, Mr. William Henry
..
886	887	2	Montvila, Rev. Juozas
887	888	1	Graham, Miss. Margaret Edith
888	889	3	Johnston, Miss. Catherine Helen "Carrie"
889	890	1	Behr, Mr. Karl Howell
890	891	3	Dooley, Mr. Patrick

	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	male	22.0	1	0.0	A/5 21171	7.2500	NaN	S
1	female	38.0	1	0.0	PC 17599	14.4542	C85	C
2	female	26.0	0	0.0	STON/O2. 3101282	7.9250	NaN	S
3	female	35.0	1	0.0	113803	14.4542	C123	S
4	male	35.0	0	0.0	373450	8.0500	NaN	S
..
886	male	27.0	0	0.0	211536	13.0000	NaN	S
887	female	19.0	0	0.0	112053	30.0000	B42	S
888	female	NaN	1	0.0	W./C. 6607	23.4500	NaN	S
889	male	26.0	0	0.0	111369	30.0000	C148	C
890	male	32.0	0	0.0	370376	7.7500	NaN	Q

[891 rows x 11 columns]

Dependent Variables

```
y = df['Survived']
print(Y)
```

```
[ 'S' 'C' 'S' 'S' 'S' 'Q' 'S' 'S' 'S' 'C' 'S' 'S' 'S' 'S' 'S' 'S' 'Q' 'S'
  'S' 'C' 'S' 'S' 'Q' 'S' 'S' 'S' 'C' 'S' 'Q' 'S' 'C' 'C' 'Q' 'S' 'C' 'S'
  'C' 'S' 'S' 'C' 'S' 'S' 'C' 'C' 'Q' 'S' 'Q' 'Q' 'C' 'S' 'S' 'S' 'C' 'S'
  'C' 'S' 'S' 'C' 'S' 'S' 'C' nan 'S' 'S' 'C' 'C' 'S' 'S' 'S' 'S' 'S' 'S'
  'S' 'C' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'Q' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S'
  'S' 'S' 'S' 'S' 'S' 'S' 'C' 'C' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S'
  'S' 'Q' 'S' 'C' 'S' 'S' 'C' 'S' 'Q' 'S' 'C' 'S' 'S' 'S' 'C' 'S' 'S' 'C'
  'Q' 'S' 'C' 'S' 'C' 'S' 'S' 'S' 'S' 'C' 'S' 'S' 'S' 'C' 'C' 'S' 'S' 'Q'
  'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'C' 'Q' 'S' 'S' 'S' 'S' 'S'
  'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'Q' 'S' 'S' 'C' 'S' 'S' 'C' 'S' 'S'
  'S' 'C' 'S' 'S' 'S' 'S' 'Q' 'S' 'Q' 'S' 'S' 'S' 'S' 'S' 'C' 'C' 'Q' 'S'
  'Q' 'S' 'S' 'S' 'S' 'C' 'S' 'S' 'S' 'C' 'Q' 'C' 'S' 'S' 'S' 'S' 'Q' 'C'
  'S' 'S' 'C' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' ]
```



```

'S' 'S' 'S' 'S' 'S' 'S' 'C' 'Q' 'S' 'S' 'C' 'Q' 'S' 'S' 'S' 'S' 'S' 'S'
'S' 'S' 'S' 'C' 'C' 'S' 'C' 'S' 'Q' 'S' 'S' 'S' 'Q' 'S' 'S' 'S' 'S' 'S'
'S' 'S' 'S' 'C' 'Q' 'S' 'S' 'S' 'Q' 'S' 'Q' 'S' 'S' 'S' 'S' 'C' 'S' 'S'
'S' 'Q' 'S' 'C' 'C' 'S' 'S' 'C' 'C' 'S' 'S' 'C' 'Q' 'Q' 'S' 'Q' 'S' 'S'
'C' 'C' 'C' 'C' 'C' 'C' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'C' 'S' 'S' 'Q' 'S'
'S' 'C' 'S' 'S' 'S' 'S' 'C' 'Q' 'S' 'S' 'S' 'S' 'S' 'S' 'C' 'S' 'S' 'S' 'S'
'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'C' 'S' 'C' 'S' 'S' 'S' 'Q' 'Q'
'S' 'C' 'C' 'S' 'Q' 'S' 'C' 'C' 'Q' 'C' 'C' 'S' 'S' 'C' 'S' 'C' 'S' 'C'
'C' 'S' 'C' 'C' 'S' 'S' 'S' 'S' 'S' 'S' 'Q' 'C' 'S' 'S' 'S' 'C' 'S' 'S'
'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'Q' 'Q' 'S'
'S' 'S' 'S' 'S' 'S' 'S' 'C' 'Q' 'S' 'S' 'S' 'S' 'S' 'S' 'Q' 'S' 'S' 'S'
'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'C' 'S'
'S' 'S' 'C' 'C' 'S' 'C' 'S' 'S' 'S' 'Q' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S'
'Q' 'C' 'S' 'S' 'S' 'C' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'C' 'S'
'S' 'C' 'S' 'S' 'S' 'S' 'S' 'S' 'C' 'S' 'C' 'C' 'S' 'S' 'S' 'S' 'Q' 'Q' 'S'
'S' 'C' 'S' 'S' 'S' 'S' 'S' 'Q' 'S' 'S' 'C' 'S' 'S' 'S' 'Q' 'S' 'S' 'S'
'C' 'C' 'C' 'Q' 'S' 'S' 'S' 'S' 'S' 'S' 'C' 'C' 'C' 'S' 'S' 'S' 'C' 'S' 'C'
'S' 'S' 'S' 'S' 'C' 'S' 'S' 'C' 'S' 'S' 'C' 'S' 'Q' 'C' 'S' 'S' 'C' 'C'
'S' 'S' 'Q' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'C' 'S' 'S' 'S' 'S' 'Q' 'S' 'S'
'S' 'S' 'C' 'S' 'S' 'C' 'S' 'C' 'C' 'S' 'S' 'C' 'S' 'S' 'S' 'C' 'S' 'Q'
'S' 'S' 'S' 'S' 'C' 'C' 'S' 'S' 'S' 'S' 'C' 'S' 'S' 'S' 'C' 'S' 'S' 'S'
'Q' 'Q' 'S' 'S' 'S' 'S' 'S' 'S' 'C' 'S' 'C' 'S' 'S' 'S' 'Q' 'S' 'S' 'Q'
'S' 'S' 'C' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'C' 'S' 'S' 'C' 'C' 'S' 'C'
'S' 'S' 'S' 'S' 'S' 'Q' 'Q' 'S' 'S' 'Q' 'S' 'C' 'S' 'C' 'S' 'S' 'S' 'S'
'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'C' 'Q' 'C' 'S' 'S'
'S' 'C' 'S' 'S' 'S' 'S' 'S' 'C' 'S' 'C' 'S' 'S' 'S' 'Q' 'C' 'S' 'C' 'S'
'C' 'Q' 'S' 'S' 'S' 'S' 'S' 'C' 'C' 'S' 'S' 'S' 'S' 'S' 'C' 'S' 'Q' 'S'
'S' 'S' 'S' 'S' 'S' 'S' 'S' 'Q' 'S' 'S' 'S' 'C' 'S' 'S' 'S' 'S' 'S' 'C'
'S' 'S' 'S' 'S' 'C' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'Q' 'S' 'S' 'S' 'S' 'S'
'S' 'S' 'S' 'S' 'S' 'S' 'C' 'S' 'S' 'S' 'C' 'Q' 'Q' 'S' 'S' 'S' 'S' 'C'
'S' 'S' 'Q' 'S' 'Q' 'S' 'C' 'S' 'S' 'S' 'S' 'S' 'S' 'Q' 'S' 'C' 'Q' 'S'
'S' 'C' 'S' 'S' 'S' 'S' 'C' 'S' 'S' 'S' 'S' 'C' 'S' 'S' 'S' 'S' 'S' 'S'
'S' 'S' 'S' 'S' 'S' 'S' 'S' 'C' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'Q' 'S' 'C'
'Q' nan 'C' 'S' 'C' 'S' 'S' 'C' 'S' 'S' 'S' 'C' 'S' 'S' 'C' 'C' 'S' 'S'
'S' 'C' 'S' 'C' 'S' 'S' 'S' 'C' 'S' 'S' 'S' 'S' 'S' 'C' 'C' 'S' 'S' 'S'
'S' 'S' 'C' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'C' 'C' 'S' 'S' 'S' 'C' 'S' 'S'
'S' 'S' 'S' 'Q' 'S' 'S' 'S' 'C' 'Q']

```

▼ Encoding

```

label_encoder = preprocessing.LabelEncoder()
df['Embarked'] = label_encoder.fit_transform(df['Embarked'])
df['Embarked'].unique()
label_encoder2 = preprocessing.LabelEncoder()
df['PassengerId'] = label_encoder2.fit_transform(df['PassengerId'])
df['PassengerId'].unique()
label_encoder3 = preprocessing.LabelEncoder()
df['Survived'] = label_encoder3.fit_transform(df['Survived'])
df['Survived'].unique()
label_encoder4 = preprocessing.LabelEncoder()
df['Pclass'] = label_encoder4.fit_transform(df['Pclass'])

```

```

df['Pclass'].unique()
label_encoder5 = preprocessing.LabelEncoder()
df['Name']= label_encoder5.fit_transform(df['Name'])
df['Name'].unique()
label_encoder6 = preprocessing.LabelEncoder()
df['Sex']= label_encoder6.fit_transform(df['Sex'])
df['Sex'].unique()
label_encoder7 = preprocessing.LabelEncoder()
df['Age']= label_encoder7.fit_transform(df['Age'])
df['Age'].unique()
label_encoder8 = preprocessing.LabelEncoder()
df['SibSp']= label_encoder8.fit_transform(df['SibSp'])
df['SibSp'].unique()
label_encoder9 = preprocessing.LabelEncoder()
df['Parch']= label_encoder9.fit_transform(df['Parch'])
df['Parch'].unique()
label_encoder10 = preprocessing.LabelEncoder()
df['Ticket']= label_encoder10.fit_transform(df['Ticket'])
df['Ticket'].unique()
label_encoder11 = preprocessing.LabelEncoder()
df['Fare']= label_encoder11.fit_transform(df['Fare'])
df['Fare'].unique()
label_encoder12 = preprocessing.LabelEncoder()
df['Cabin']= label_encoder12.fit_transform(df['Cabin'])
df['Cabin'].unique()

```

```

array([[147, 81, 55, 129, 145, 49, 111, 13, 63, 41, 101, 23, 71,
        21, 80, 142, 140, 122, 12, 91, 98, 52, 36, 116, 138, 107,
        45, 141, 61, 123, 18, 14, 69, 144, 9, 28, 43, 8, 103,
        93, 87, 78, 102, 83, 40, 134, 46, 57, 89, 54, 113, 3,
        31, 90, 62, 51, 74, 125, 72, 35, 76, 124, 65, 17, 56,
        85, 127, 146, 59, 104, 24, 131, 79, 47, 115, 128, 10, 50,
        53, 86, 126, 97, 117, 133, 1, 25, 64, 96, 42, 121, 106,
        39, 88, 26, 27, 20, 82, 77, 2, 48, 75, 0, 135, 29,
        4, 95, 110, 114, 5, 33, 7, 108, 132, 58, 38, 34, 109,
        32, 19, 139, 73, 120, 84, 66, 137, 15, 105, 67, 100, 118,
        92, 136, 143, 22, 112, 44, 94, 11, 16, 37, 130, 68, 99,
        119, 6, 70, 30, 60])

```

▼ Feature Scaling

```

numerical_columns = df.select_dtypes(include=['int64', 'float64']).columns
scaler = StandardScaler()
df[numerical_columns] = scaler.fit_transform(df[numerical_columns])
print(df.head())

```

	Survived	Pclass	Name \
0	-0.789272	0.827377	Braund, Mr. Owen Harris
1	1.266990	-1.566107	Cumings, Mrs. John Bradley (Florence Briggs Th...

2	1.266990	0.827377						Heikkinen, Miss. Laina
3	1.266990	-1.566107						Futrelle, Mrs. Jacques Heath (Lily May Peel)
4	-0.789272	0.827377						Allen, Mr. William Henry

	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	\
0	male	-0.530377	0.432793	-0.473674	A/5 21171	-0.502445	NaN	
1	female	0.571831	0.432793	-0.473674	PC 17599	0.786845	C85	
2	female	-0.254825	-0.474545	-0.473674	STON/O2. 3101282	-0.488854	NaN	
3	female	0.365167	0.432793	-0.473674	113803	0.420730	C123	
4	male	0.365167	-0.474545	-0.473674	373450	-0.486337	NaN	

	... PassengerId_882	PassengerId_883	PassengerId_884	PassengerId_885	\
0	...	0	0	0	0
1	...	0	0	0	0
2	...	0	0	0	0
3	...	0	0	0	0
4	...	0	0	0	0

	PassengerId_886	PassengerId_887	PassengerId_888	PassengerId_889	\
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

	PassengerId_890	PassengerId_891
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0

[5 rows x 901 columns]

▼ Splitting into Train and Test

```
X = df.drop('Survived', axis=1)
y = df['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (623, 11)
X_test shape: (268, 11)
y_train shape: (623,)
y_test shape: (268,)
```

✓ 0s completed at 12:05 PM

● ✕