```
# R.CHARAN SATHVIK
# 21BCE7365
# VITAP MORNING SLOT
# ASSIGNMENT-3
# Data Preprocessing on TITANIC dataset.
```

```
#   Data Preprocessing.
# Import the Libraries.
# Import the dataset
# Checking for Null Values.
```

```
# Data Visualization.
# Outlier Detection
#   Splitting Dependent and Independent variables
#   Encoding
#   Feature Scaling.
#   Splitting Data into Train and Test.
```

Import the Libraries

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

Import the Dataset

```
df = pd.read_csv("/content/drive/MyDrive/DATASETS/Titanic-Dataset.csv")
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
     Mounted at /content/drive
```

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embark
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
				Heikkinen								

```
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          714 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        204 non-null    object
      11  Embarked     889 non-null    object
     dtypes: float64(2), int64(5), object(5)
     memory usage: 83.7+ KB
```

```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000

```
df.corr()
```

```
<ipython-input-8-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is
  df.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

```
df.corr().Survived.sort_values(ascending = False)
```

```
<ipython-input-9-936bc0a2ea37>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
  df.corr().Survived.sort_values(ascending = False)
Survived       1.000000
Fare           0.257307
Parch          0.081629
PassengerId   -0.005007
SibSp         -0.035322
Age           -0.077221
Pclass        -0.338481
Name: Survived, dtype: float64
```

Handling Missing/Null Values

```
df.isnull().any()
```

```
PassengerId    False
Survived       False
Pclass         False
Name           False
Sex            False
Age             True
SibSp          False
Parch          False
Ticket         False
Fare           False
Cabin           True
Embarked        True
dtype: bool
```

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

```
687
```

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

```
177
```

```
df["Age"].fillna(df["Age"].mean(),inplace=True)
```

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

```
2
```

```
df["Embarked"].fillna(df["Embarked"].mode()[0],inplace=True)
```

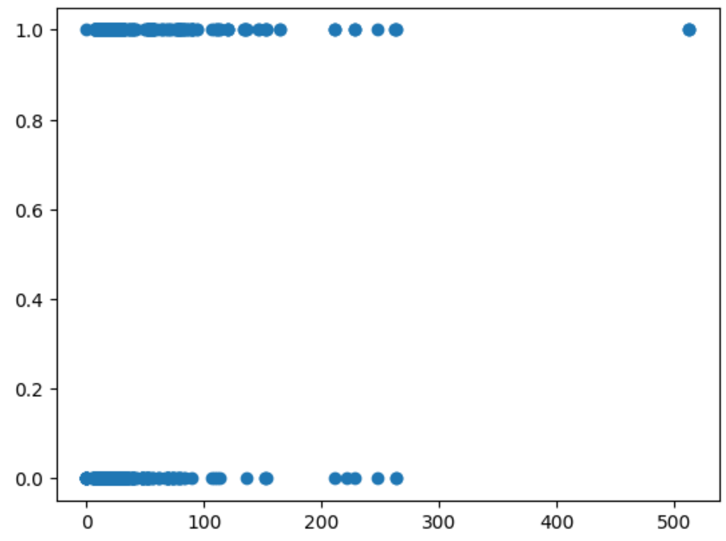
```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	13.002015	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Data Visualization

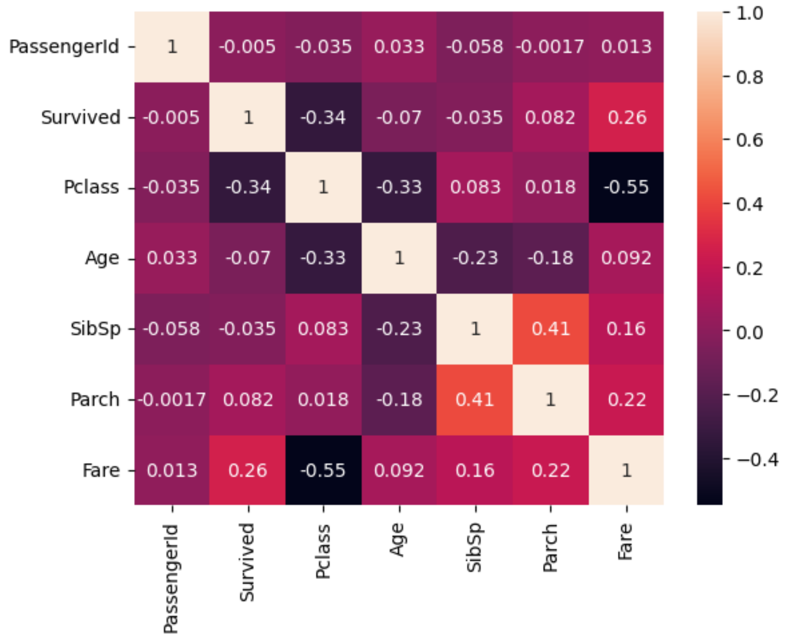
```
plt.scatter(df["Fare"],df["Survived"])
```

```
<matplotlib.collections.PathCollection at 0x799cabdb2950>
```



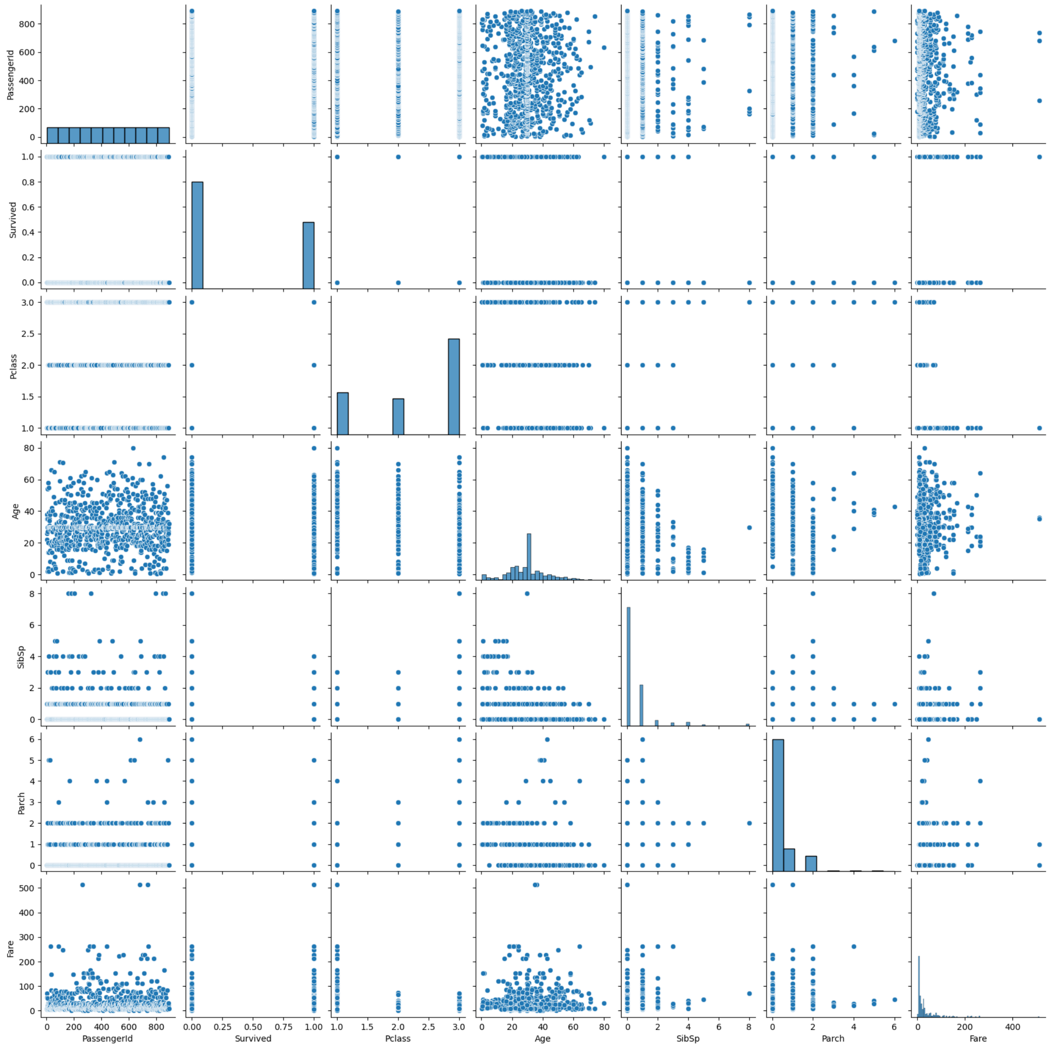
```
sns.heatmap(df.corr(),annot=True)
```

```
<ipython-input-18-8df7bcac526d>:1: FutureWarning: The default value of numeric_only in DataFrame.corr :
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```



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

```
<seaborn.axisgrid.PairGrid at 0x799ca9c15840>
```

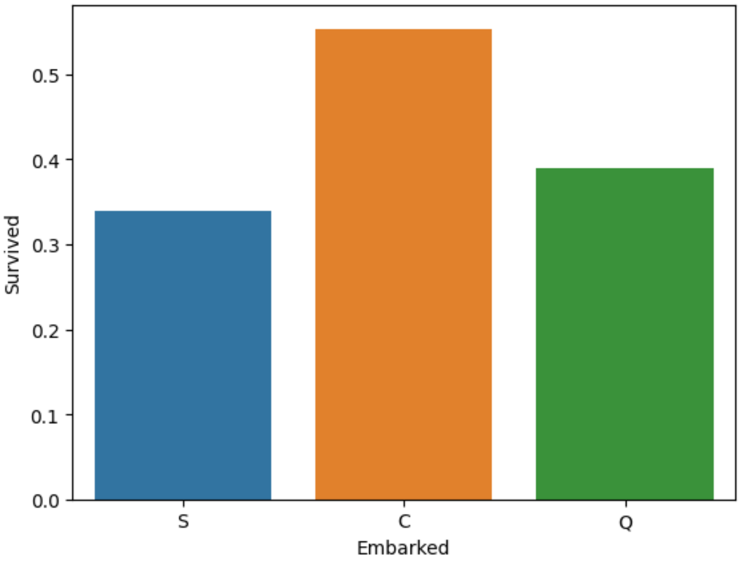


```
sns.barplot(x=df["Sex"],y=df["Survived"],ci=0)
```

```
<ipython-input-20-8ae461271d98>:1: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=('ci', 0)` for the same effect.

  sns.barplot(x=df["Sex"],y=df["Survived"],ci=0)
<Axes: xlabel='Sex', ylabel='Survived'>
```

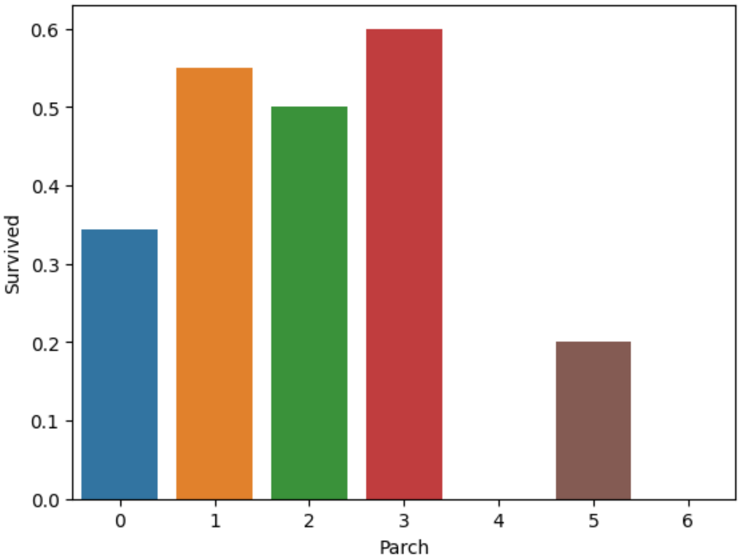
```
sns.barplot(x=df["Embarked"],y=df["Survived"],ci=0)
```

```
<ipython-input-21-d5b0276940a6>:1: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=('ci', 0)` for the same effect.

  sns.barplot(x=df["Embarked"],y=df["Survived"],ci=0)
<Axes: xlabel='Embarked', ylabel='Survived'>
```



```
sns.barplot(x=df["Parch"],y=df["Survived"],ci=0)
```

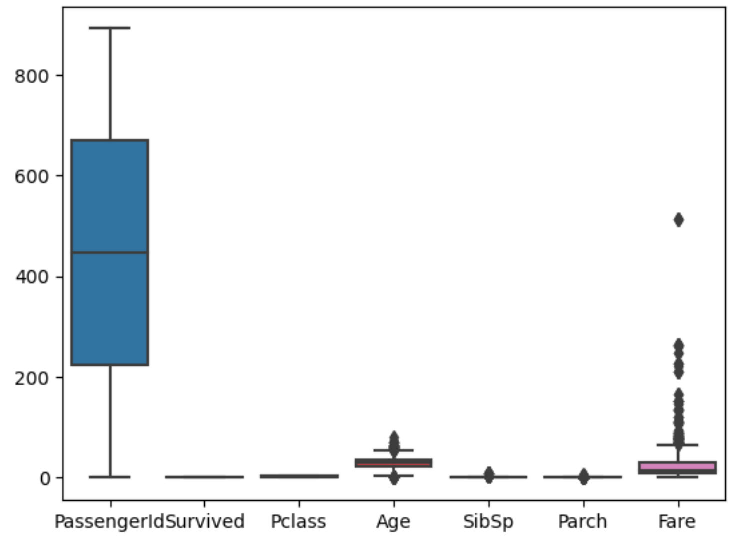
```
<ipython-input-22-a1496fefeaf8>:1: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=('ci', 0)` for the same effect.

  sns.barplot(x=df["Parch"],y=df["Survived"],ci=0)
<Axes: xlabel='Parch', ylabel='Survived'>
```



Outlier Detection

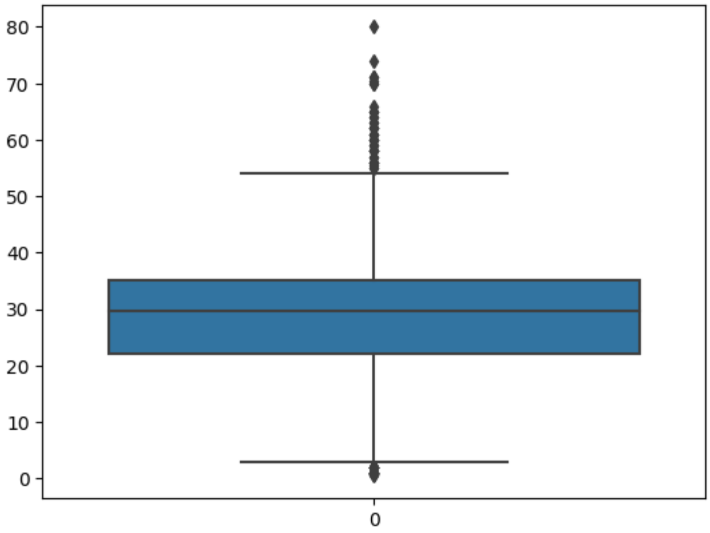
```
sns.boxplot(df)
```

```
<Axes: >
```



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

```
<Axes: >
```



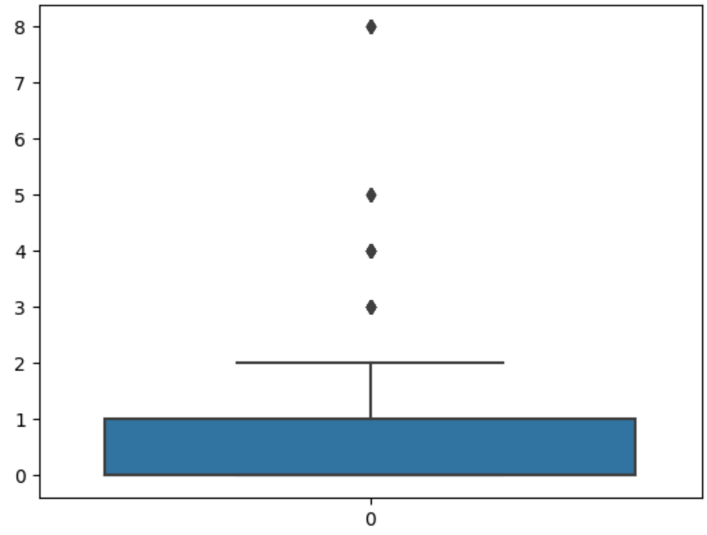
```
Q1 = df['Age'].quantile(0.25)
Q3 = df['Age'].quantile(0.75)

IQR = Q3 - Q1

threshold = 1.5 * IQR

df = df[(df['Age'] >= Q1 - threshold) & (df['Age'] <= Q3 + threshold)]
```

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

```
<Axes: >
```

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

```
<Axes: >
```

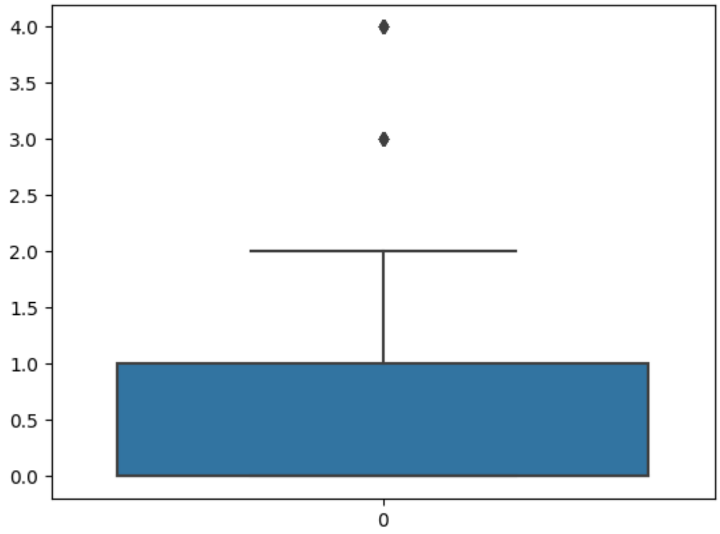


```
p99 = df.SibSp.quantile(0.99)
```

```
df = df[df.SibSp < p99]
```

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

```
<Axes: >
```



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

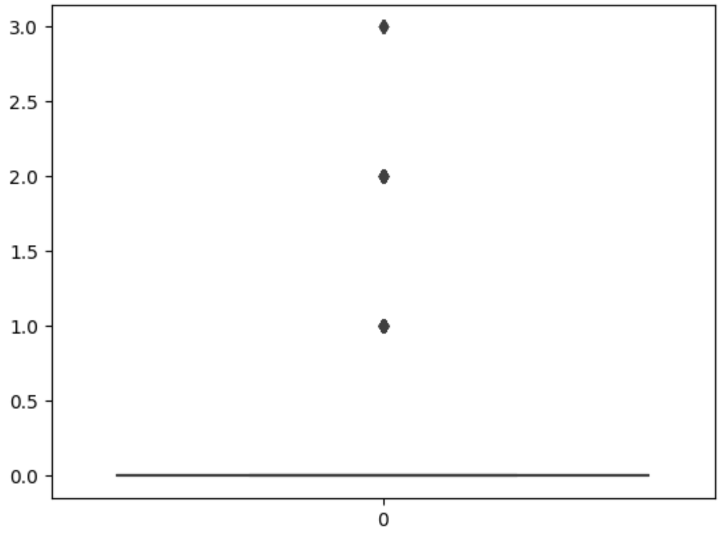
```
<Axes: >
```

```
p99 = df.Parch.quantile(0.99)
```

```
df = df[df.Parch < p99]
```

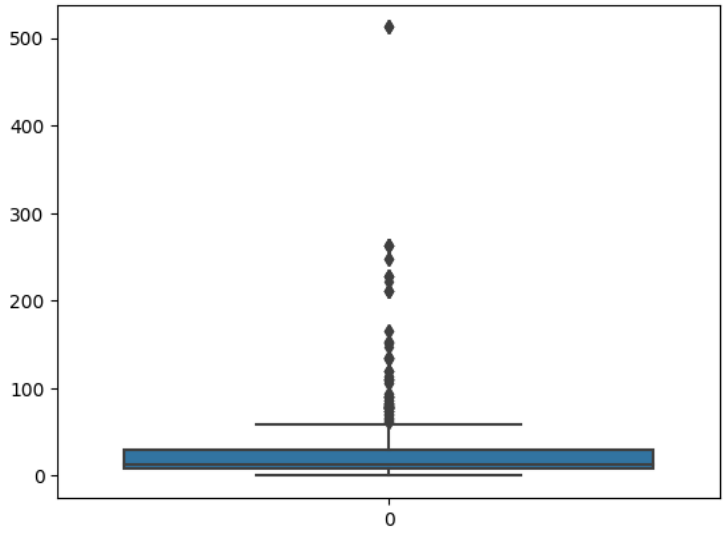
```
sns.boxplot(df["Parch"])
```

```
<Axes: >
```



```
sns.boxplot(df["Fare"])
```

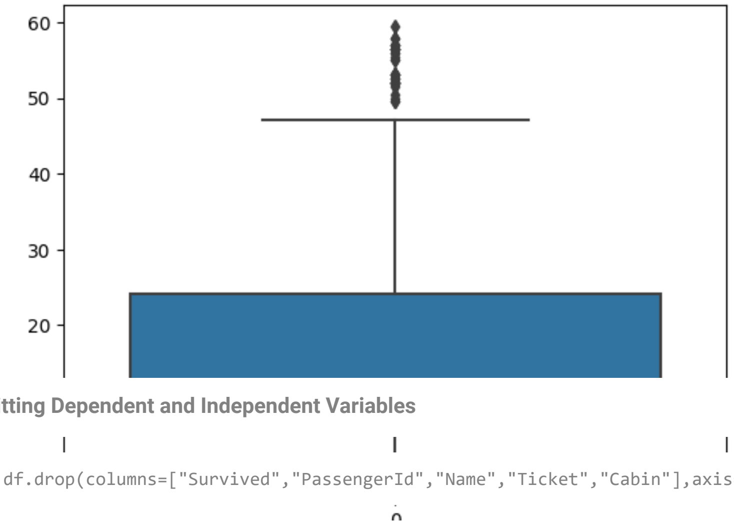
```
<Axes: >
```



```
Q1 = df['Fare'].quantile(0.25)
Q3 = df['Fare'].quantile(0.75)

IQR = Q3 - Q1

threshold = 1.5 * IQR

df = df[(df['Fare'] >= Q1 - threshold) & (df['Fare'] <= Q3 + threshold)]
```

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

```
<Axes: >
```



Splitting Dependent and Independent Variables

```
x = df.drop(columns=["Survived","PassengerId","Name","Ticket","Cabin"],axis=1) # Independent variables should be in df or 2d array
```

```
x.head()
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	22.000000	1	0	7.2500	S
2	3	female	26.000000	0	0	7.9250	S
3	1	female	35.000000	1	0	53.1000	S
4	3	male	35.000000	0	0	8.0500	S
5	3	male	29.699118	0	0	8.4583	Q

```
y = pd.Series(df["Survived"])
```

```
y.head()
```

```
0    0
2    1
3    1
4    0
5    0
Name: Survived, dtype: int64
```

Encoding

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
x["Sex"] = le.fit_transform(x["Sex"])
```

```
x.head()
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	22.000000	1	0	7.2500	S
2	3	0	26.000000	0	0	7.9250	S
3	1	0	35.000000	1	0	53.1000	S
4	3	1	35.000000	0	0	8.0500	S
5	3	1	29.699118	0	0	8.4583	Q

```
print(le.classes_)
```

```
['female' 'male']
```

```
mapping=dict(zip(le.classes_,range(len(le.classes_))))
```

```
mapping
```

```
{'female': 0, 'male': 1}
```

```
le1 = LabelEncoder()
```

```
x["Embarked"] = le1.fit_transform(x["Embarked"])
```

```
x.head()
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	22.000000	1	0	7.2500	2
2	3	0	26.000000	0	0	7.9250	2
3	1	0	35.000000	1	0	53.1000	2
4	3	1	35.000000	0	0	8.0500	2
5	3	1	29.699118	0	0	8.4583	1

```
print(le1.classes_)
```

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

```
mapping1=dict(zip(le1.classes_,range(len(le1.classes_))))
```

```
mapping1
```

```
{'C': 0, 'Q': 1, 'S': 2}
```

Feature Scaling

```
from sklearn.preprocessing import MinMaxScaler
ms = MinMaxScaler()
```

```
x_Scaled = pd.DataFrame(ms.fit_transform(x),columns = x.columns)
```

```
x_Scaled.head()
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1.0	1.0	0.372549	0.25	0.0	0.122054	1.0
1	1.0	0.0	0.450980	0.00	0.0	0.133418	1.0
2	0.0	0.0	0.627451	0.25	0.0	0.893939	1.0
3	1.0	1.0	0.627451	0.00	0.0	0.135522	1.0
4	1.0	1.0	0.523512	0.00	0.0	0.142396	0.5

Splitting Training and Testing Data

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(x_Scaled,y,test_size = 0.2,random_state =0)
```

```
print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)
```

```
(562, 7) (141, 7) (562,) (141,)
```

Executing (1m 23s)