In [1]:
```
import pandas as pd
import statistics as st
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('titanic (1).csv')
df.head(2)
df.tail(2)
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

In [3]:
```
df.shape
```

Out[3]:
```
(891, 12)
```

Exploratory Data Analysis

In [4]:
```
df.dtypes
```

Out[4]:
```
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
```

In [5]:

```
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
```

In [6]:
```
df.describe(include='all')
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
count	891.000000	891.000000	891.000000	891	891	714.000000	891.000000	891.000000	891	891.000000	204	889
unique	NaN	NaN	NaN	891	2	NaN	NaN	NaN	681	NaN	147	3
top	NaN	NaN	NaN	Braund, Mr. Owen Harris	male	NaN	NaN	NaN	347082	NaN	B96 B98	S
freq	NaN	NaN	NaN	1	577	NaN	NaN	NaN	7	NaN	4	644
mean	446.000000	0.383838	2.308642	NaN	NaN	29.699118	0.523008	0.381594	NaN	32.204208	NaN	NaN
std	257.353842	0.486592	0.836071	NaN	NaN	14.526497	1.102743	0.806057	NaN	49.693429	NaN	NaN
min	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.000000	0.000000	NaN	0.000000	NaN	NaN
25%	223.500000	0.000000	2.000000	NaN	NaN	20.125000	0.000000	0.000000	NaN	7.910400	NaN	NaN
50%	446.000000	0.000000	3.000000	NaN	NaN	28.000000	0.000000	0.000000	NaN	14.454200	NaN	NaN
75%	668.500000	1.000000	3.000000	NaN	NaN	38.000000	1.000000	0.000000	NaN	31.000000	NaN	NaN
max	891.000000	1.000000	3.000000	NaN	NaN	80.000000	8.000000	6.000000	NaN	512.329200	NaN	NaN

In [7]:
```
df.corr()
```

Out[7]:

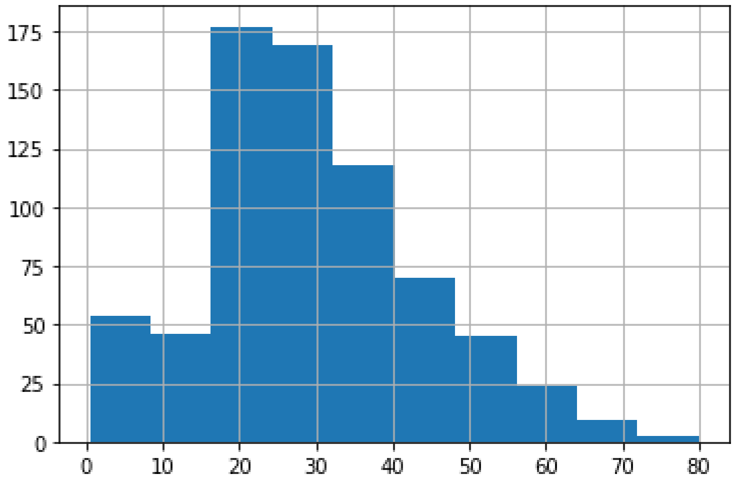
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221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

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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
```

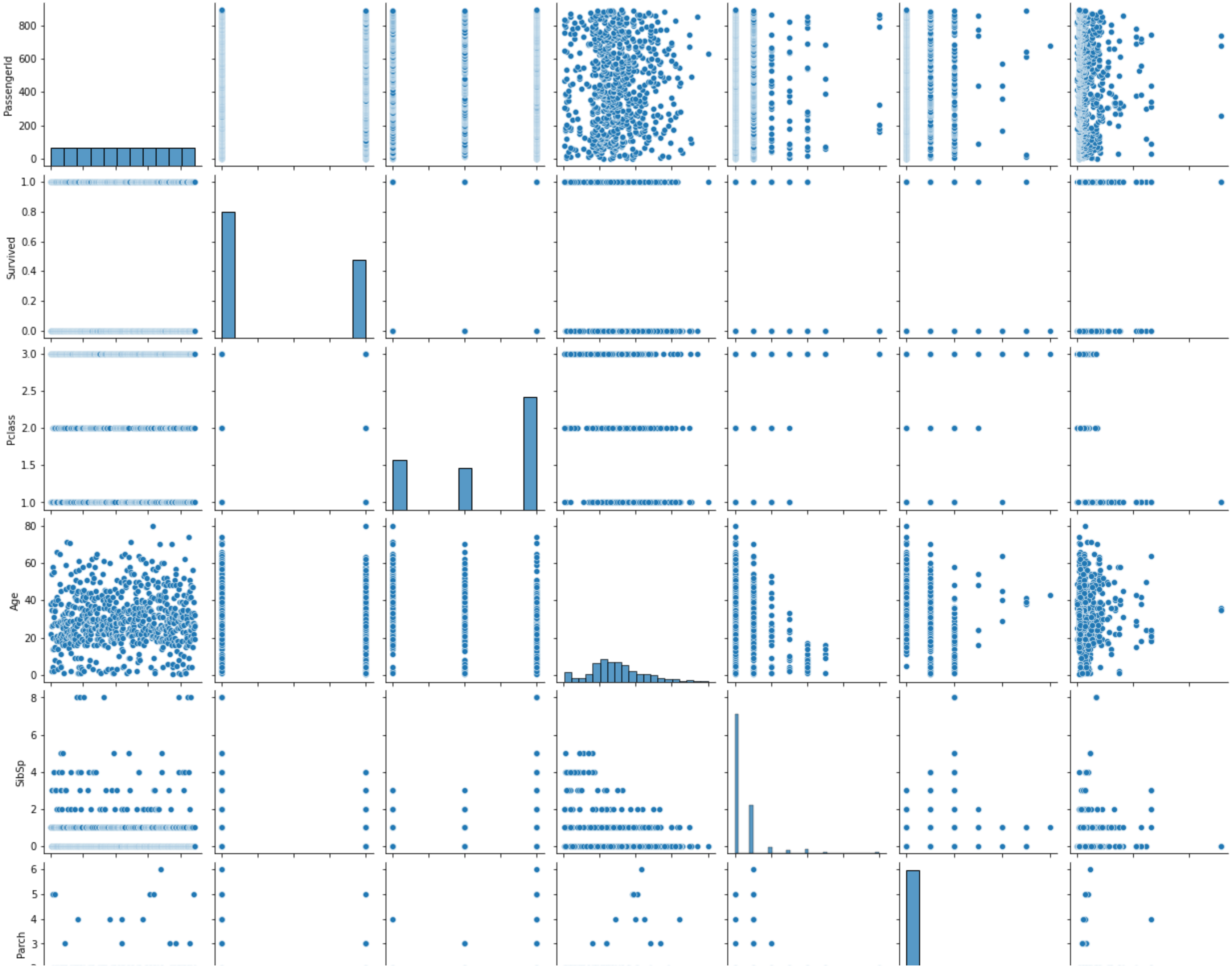
In [9]:
```
df['Age'].hist()
```

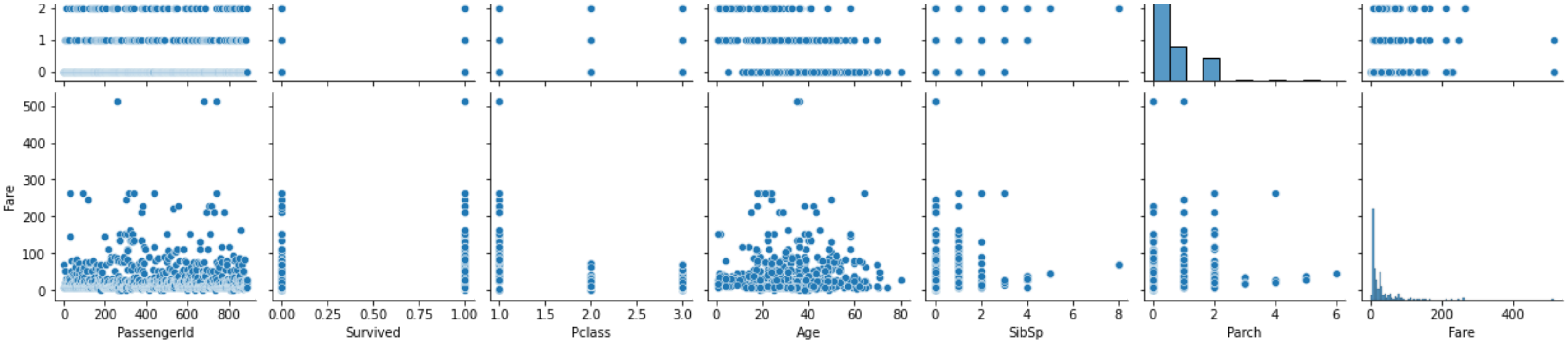
Out[9]:
```
<AxesSubplot:>
```



In [10]:
```
sns.pairplot(df)
```

Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x1e7e0506f50>
```



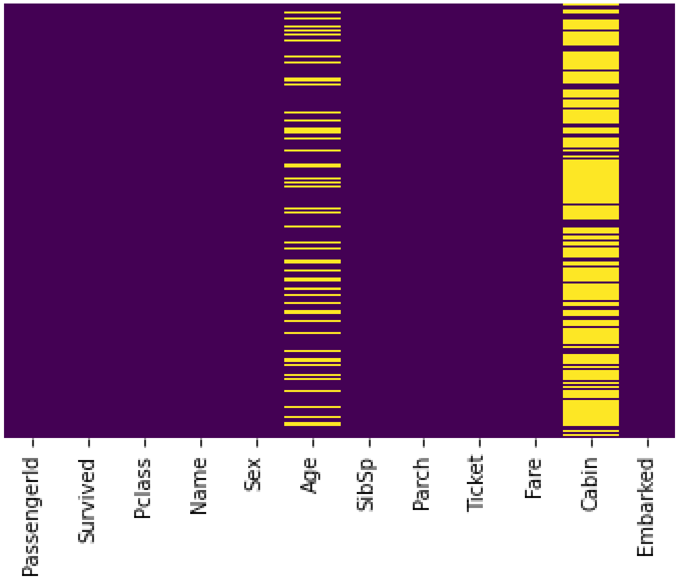


In [11]:

```
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

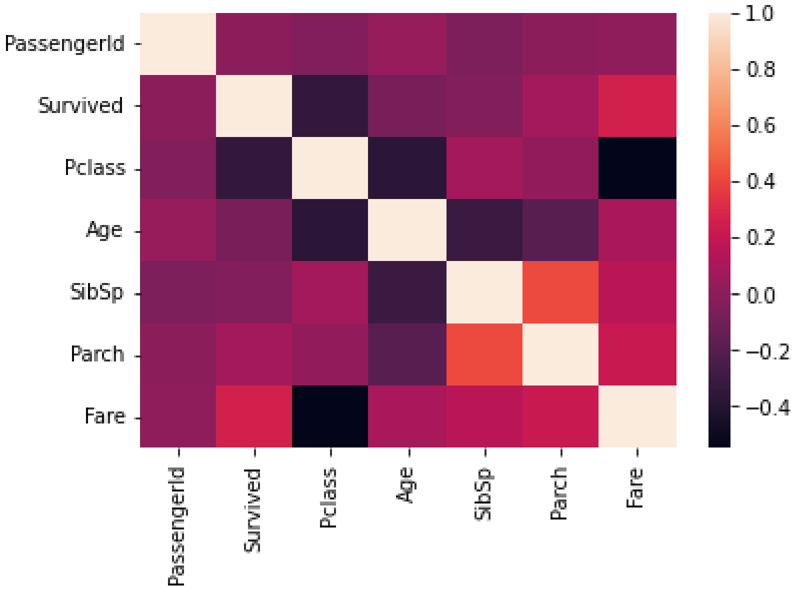
Out[11]:

```
<AxesSubplot:>
```



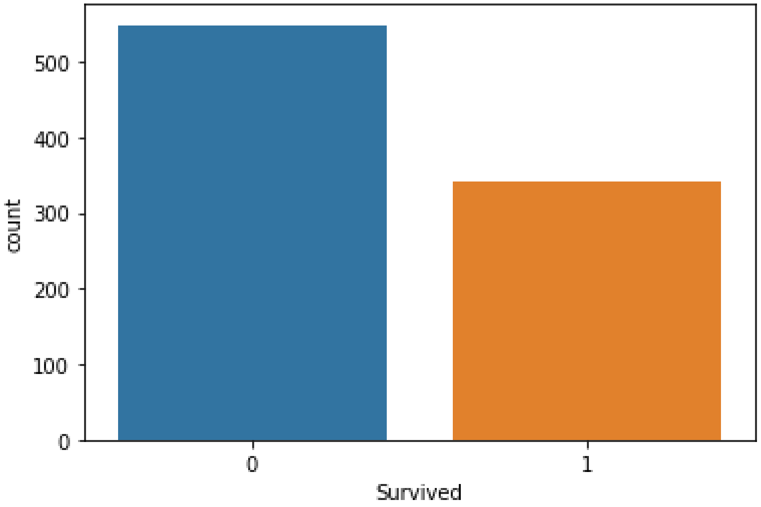
In [12]:
```
sns.heatmap(df.corr())
```

Out[12]:
```
<AxesSubplot:>
```



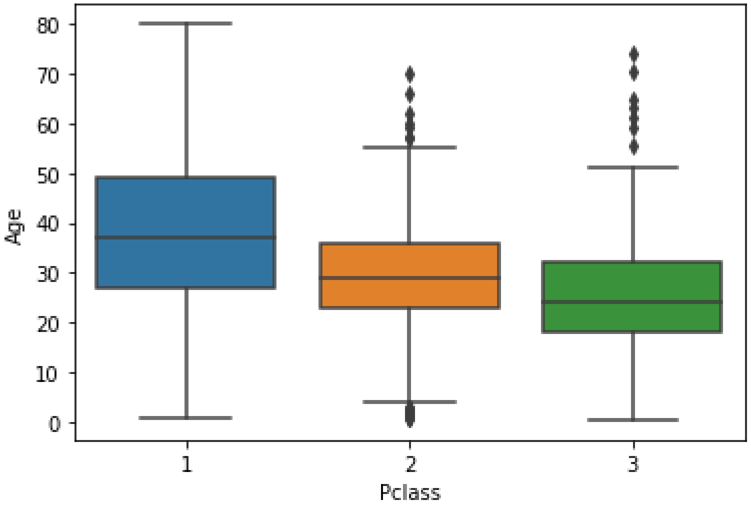
In [13]:
```
sns.countplot(x='Survived',data=df)
```

Out[13]:
```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```



```
In [14]: sns.boxplot(x='Pclass', y='Age', data =df)
```

```
Out[14]: <AxesSubplot:xlabel='Pclass', ylabel='Age'>
```



data wrengling:- cleaning data

In [15]:

```
# plt.scatter(df['Age'],df['Pclass'])
# plt.bar(df['Age'],df['Pclass'])
df.head()
```

Out[15]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [16]:

```
df.dropna(inplace=True)
df.isnull().sum()
```

Out[16]:

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

In [17]:
```
df.dtypes
```

Out[17]:
```
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
```

In [18]:
```
df.corr()
```

Out[18]:

	PassengerId	**Survived**	**Pclass**	**Age**	**SibSp**	**Parch**	**Fare**
PassengerId	1.000000	0.148495	-0.089136	0.030933	-0.083488	-0.051454	0.029740
Survived	0.148495	1.000000	-0.034542	-0.254085	0.106346	0.023582	0.134241
Pclass	-0.089136	-0.034542	1.000000	-0.306514	-0.103592	0.047496	-0.315235
Age	0.030933	-0.254085	-0.306514	1.000000	-0.156162	-0.271271	-0.092424
SibSp	-0.083488	0.106346	-0.103592	-0.156162	1.000000	0.255346	0.286433
Parch	-0.051454	0.023582	0.047496	-0.271271	0.255346	1.000000	0.389740
Fare	0.029740	0.134241	-0.315235	-0.092424	0.286433	0.389740	1.000000

In [19]:
```
df.drop(['PassengerId','Ticket','Cabin','Name'],axis=1,inplace=True)
```

In [20]:
```
df
```

Out[20]:
```
     Survived  Pclass     Sex   Age  SibSp  Parch     Fare  Embarked
  1         1       1  female  38.0      1      0  71.2833         C
  3         1       1  female  35.0      1      0  53.1000         S
  6         0       1    male  54.0      0      0  51.8625         S
 10         1       3  female   4.0      1      1  16.7000         S
 11         1       1  female  58.0      0      0  26.5500         S
...       ...     ...     ...   ...    ...    ...      ...       ...
871         1       1  female  47.0      1      1  52.5542         S
872         0       1    male  33.0      0      0   5.0000         S
879         1       1  female  56.0      0      1  83.1583         C
887         1       1  female  19.0      0      0  30.0000         S
889         1       1    male  26.0      0      0  30.0000         C

183 rows × 8 columns
```

```
In [21]: df
```

```
Out[21]:
        Survived  Pclass     Sex   Age  SibSp  Parch     Fare  Embarked
    1          1       1  female  38.0      1      0  71.2833         C
    3          1       1  female  35.0      1      0  53.1000         S
    6          0       1    male  54.0      0      0  51.8625         S
   10          1       3  female   4.0      1      1  16.7000         S
   11          1       1  female  58.0      0      0  26.5500         S
  ...        ...     ...     ...   ...    ...    ...      ...       ...
  871          1       1  female  47.0      1      1  52.5542         S
  872          0       1    male  33.0      0      0   5.0000         S
  879          1       1  female  56.0      0      1  83.1583         C
  887          1       1  female  19.0      0      0  30.0000         S
  889          1       1    male  26.0      0      0  30.0000         C

183 rows × 8 columns
```

In [22]:

```
import numpy as np
outlier = []
def detect_z(data):
    thres = 3
    mean = np.mean(data)
    std = np.std(data)

    for i in data:
        z = (i-mean)/std
        if (np.abs(z) > thres):
            outlier.append(i)
    print(outlier)

detect_z(df['Fare'])
```

```
[512.3292, 512.3292]
```

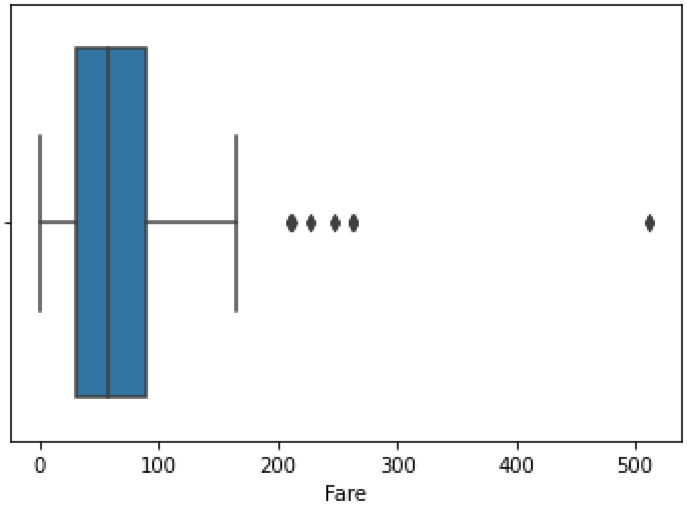
In [23]:

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

```
C:\python314\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[23]:

```
<AxesSubplot:xlabel='Fare'>
```



In [24]:

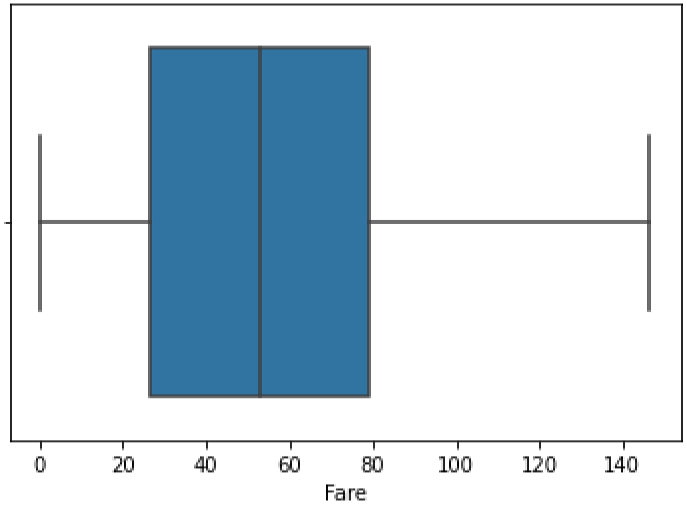
```
for i in df.index:
    if df.loc[i,'Fare']>150:
        df.drop(i, inplace=True)
```

In [25]:

```
sns.boxplot(df['Fare'])
print(df.shape)
```

```
(160, 8)
```

```
C:\python314\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyw
ord arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



change datatype

In [26]:

```
# df['Embarked'] = df['Embarked'].astype(int)
```

replace str with null

In [27]:
```
# df['Embarked'] = df['Embarked'].replace('C', np.nan)
# df.dropna(inplace=True)
```

algorithm apply

In [28]:
```
from sklearn.preprocessing import LabelEncoder
df1 = df.copy()
e1 = LabelEncoder()
e2 = LabelEncoder()

df1.Sex = e1.fit_transform(df1.Sex)
df1.Embarked = e2.fit_transform(df1.Embarked)


df1
```

Out[28]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
1	1	1	0	38.0	1	0	71.2833	0
3	1	1	0	35.0	1	0	53.1000	2
6	0	1	1	54.0	0	0	51.8625	2
10	1	3	0	4.0	1	1	16.7000	2
11	1	1	0	58.0	0	0	26.5500	2
...	...	...	...	...	...	...	...	...
871	1	1	0	47.0	1	1	52.5542	2
872	0	1	1	33.0	0	0	5.0000	2
879	1	1	0	56.0	0	1	83.1583	0
887	1	1	0	19.0	0	0	30.0000	2
889	1	1	1	26.0	0	0	30.0000	0

160 rows × 8 columns

In [29]:
```
df = df1
df.dtypes
```

Out[29]:
```
Survived      int64
Pclass        int64
Sex           int32
Age         float64
SibSp         int64
Parch         int64
Fare        float64
Embarked      int32
dtype: object
```

In [30]:
```
# X = df[['Pclass','Sex','Age','SibSp','Parch','Fare','Embarked']]
X = df.drop('Survived', axis=1)
y = df['Survived']
```

In [31]:
```
from sklearn.model_selection import train_test_split
```

In [32]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=.30,random_state=5)
```

In [33]:
```
# from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
```

In [34]:
```
T = LogisticRegression()
```

```
In [35]: T.fit(X_train,y_train)        #train dataset
```

```
C:\python314\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to conv
erge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/prepro
cessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/st
able/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

```
Out[35]: LogisticRegression()
```

```
In [36]: pred = T.predict(X_test)        #predict by      x test
```

```
In [37]: pred
```

```
Out[37]: array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1,
                1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
                1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
                1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
                0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1,
                1, 1], dtype=int64)
```

```
In [38]: from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
```

```
In [39]: print(classification_report(y_test,pred))       #y test use
```

```
              precision    recall  f1-score   support

           0       0.65      0.50      0.56        40
           1       0.75      0.85      0.80        72

    accuracy                           0.72       112
   macro avg       0.70      0.67      0.68       112
weighted avg       0.71      0.72      0.71       112
```

```
In [40]: print(accuracy_score(y_test,pred))

         0.7232142857142857
```

```
In [41]: confusion_matrix(y_test,pred)

Out[41]: array([[20, 20],
                [11, 61]], dtype=int64)
```

```
In [ ]:
```