```
STEPS

1. Import the Libraries.
2. Importing the dataset.
3. Checking for Null Values.
4. Data Visualization.
5. Outlier Detection
6. Splitting Dependent and Independent variables
7. Perform Encoding
8. Splitting Data into Train and Test
9. Feature Scaling.
```

Step 1 - Import the Libraries.

In [1]:

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

Step 2 - Import the dataset

In [2]:

```
dataset = pd.read_csv('Titanic-Dataset.csv')
```

In [3]:

```
dataset.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050

In [4]:

```
dataset.shape
```

Out[4]:

```
(891, 12)
```

In [5]:

```
dataset.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]:

```
dataset.describe()
```

Out[6]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Far
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.00000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.20420
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.69342
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.00000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.91040
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.45420
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.00000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.32920

In [7]:

```
corr = dataset.corr()
corr
```

```
C:\Users\ABILASH\AppData\Local\Temp\ipykernel_18204\897440734.py:1: Futur
eWarning: The default value of numeric_only in DataFrame.corr is deprecat
ed. In a future version, it will default to False. Select only valid colu
mns or specify the value of numeric_only to silence this warning.
  corr = dataset.corr()
```

Out[7]:

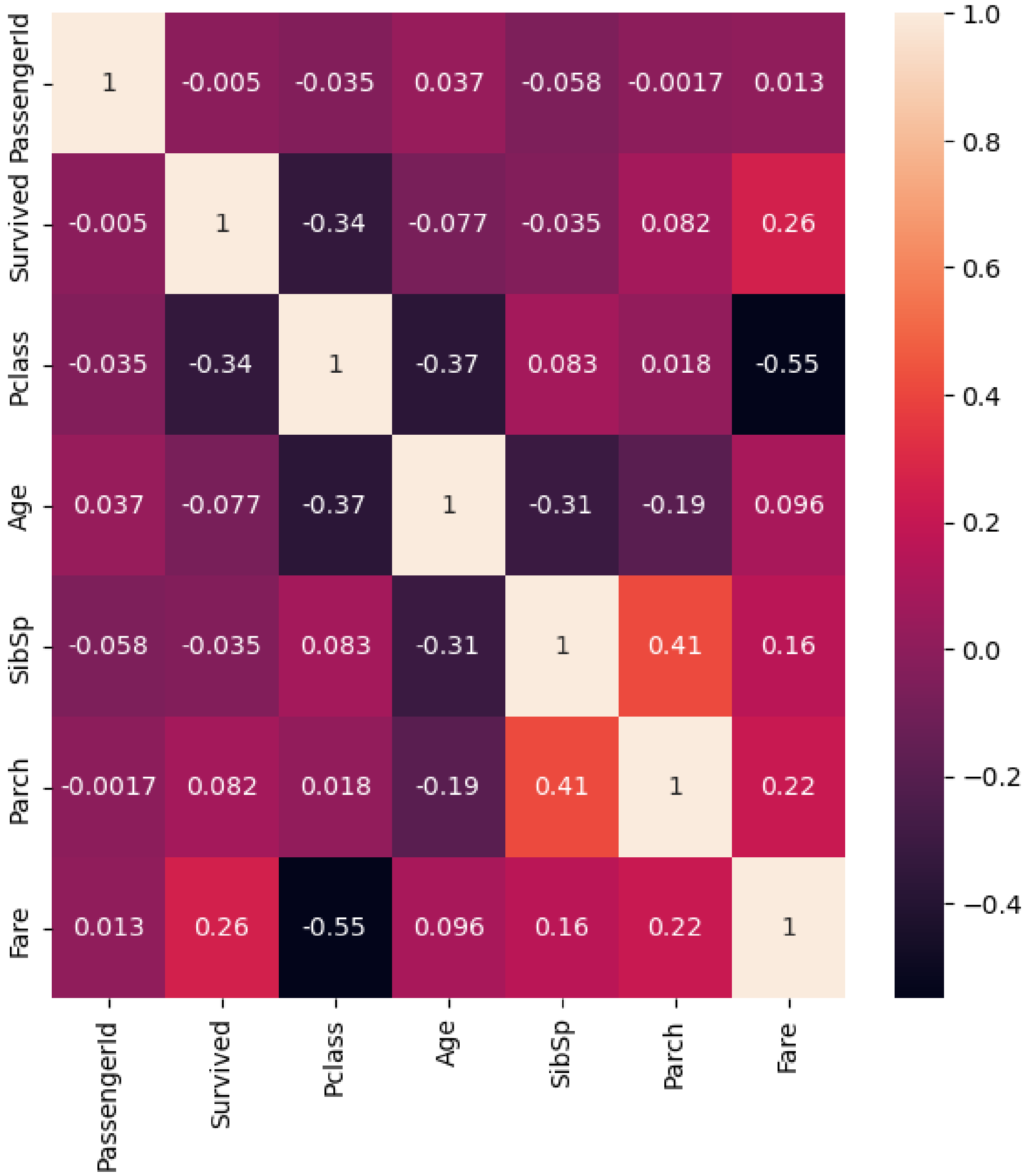
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221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

```
plt.subplots(figsize=(7,7))
sns.heatmap(corr,annot=True)
```

Out[8]:

```
<Axes: >
```



Step 3 - Checking for Null Values.

In [9]:

```
dataset.isnull().any()
```

Out[9]:

```
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
```

In [10]:

```
dataset.isnull().sum()
```

Out[10]:

```
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 [11]:

```
#'Age','Cabin' and 'Embarked' has null values
```

In [12]:

```
#Age is a numerical data, we handle it by using mean

age_mean = dataset['Age'].mean()
age_mean
```

Out[12]:

```
29.69911764705882
```

In [13]:

```
dataset['Age'].fillna(age_mean,inplace=True)
```

In [14]:

```
# 'Cabin' and 'Embarked' are cattegorical data, we handle it by using mode

#Since 'Cabin' has 687 Null values, it is better to drop the column

dataset.drop(columns='Cabin',inplace=True)
```

In [15]:

```
embarked_mode = dataset['Embarked'].mode()[0]
embarked_mode
```

Out[15]:

```
'S'
```

In [16]:

```
dataset['Embarked'].fillna(embarked_mode,inplace=True)
```

In [17]:

```
dataset.isnull().any()
```

Out[17]:

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

In [18]:

```
dataset.isnull().sum()
```

Out[18]:

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

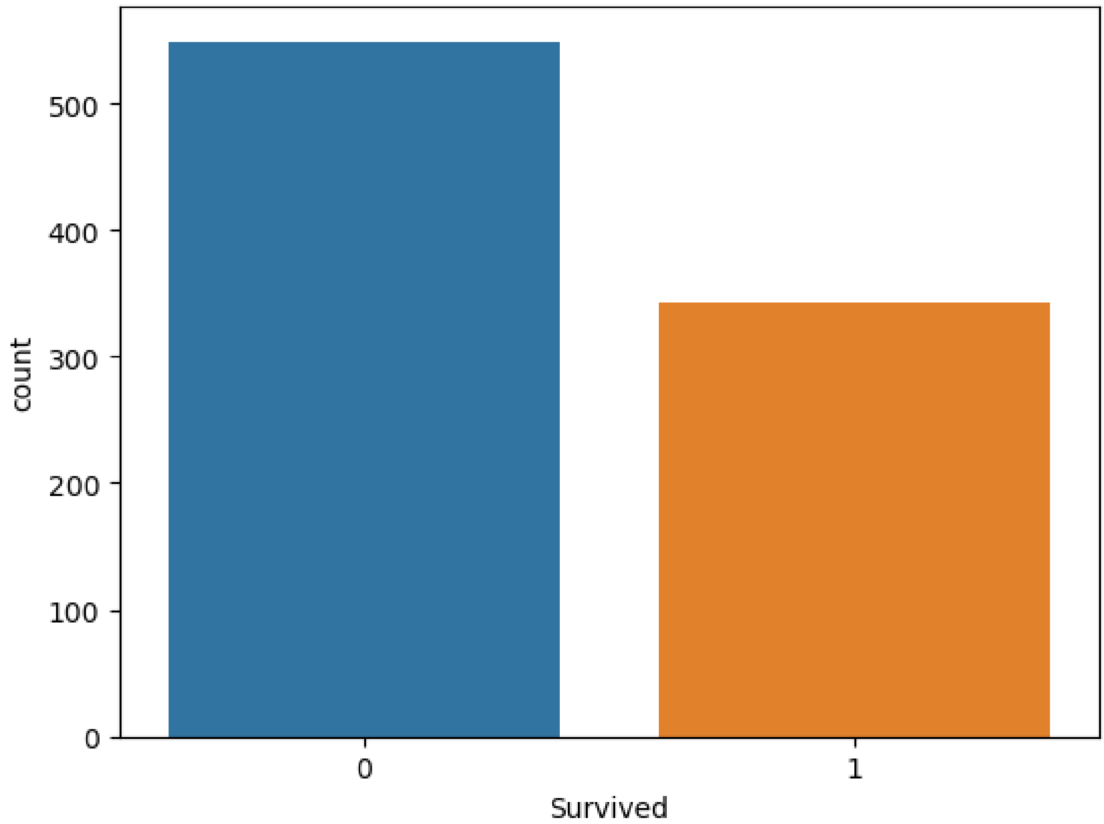
Step 4 - Data Visualization.

In [19]:

```
sns.countplot(x='Survived',data=dataset)
```

Out[19]:

```
<Axes: xlabel='Survived', ylabel='count'>
```



In [20]:

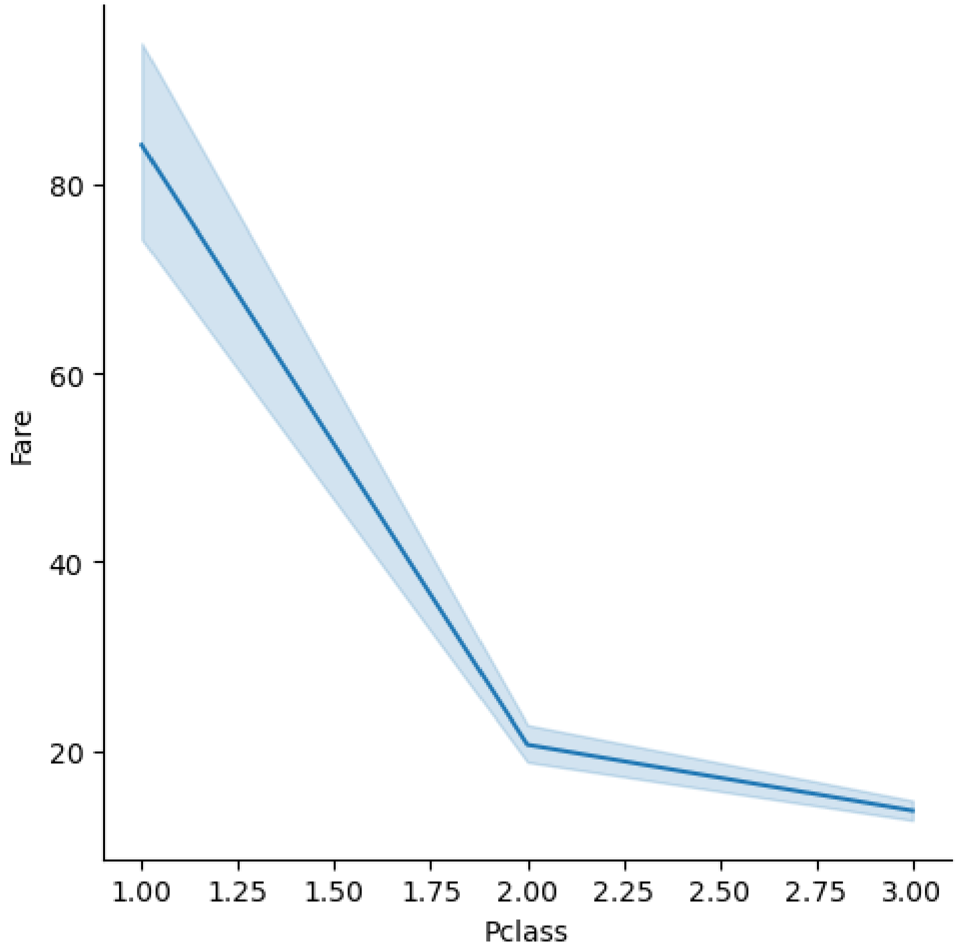
```
#From the above plot, it is clear that, majority of the people did not survive
```

In [21]:

```
sns.relplot(x='Pclass',y='Fare',data=dataset,kind='line')
```

Out[21]:

```
<seaborn.axisgrid.FacetGrid at 0x2869100f410>
```



In [22]:

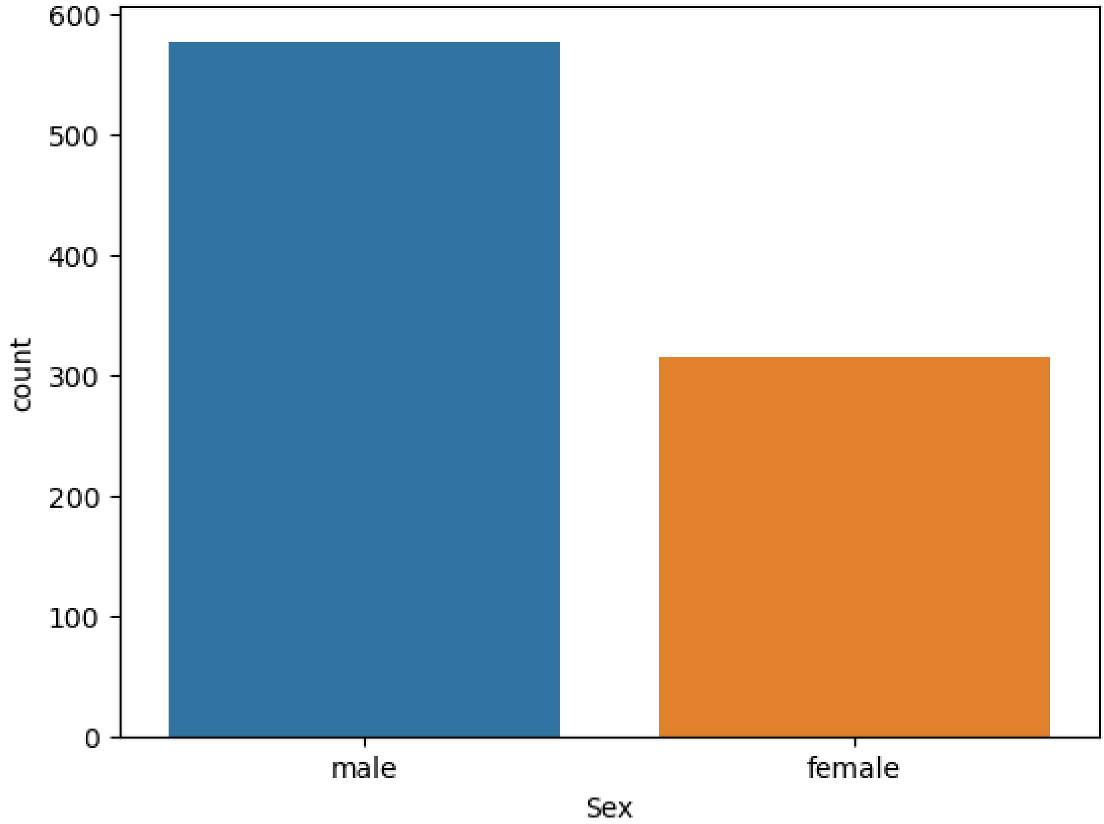
```
#from the graph it is clear that, the first class tickets costed more than the second a
```

In [23]:

```
sns.countplot(x='Sex',data=dataset)
```

Out[23]:

```
<Axes: xlabel='Sex', ylabel='count'>
```



In [24]:

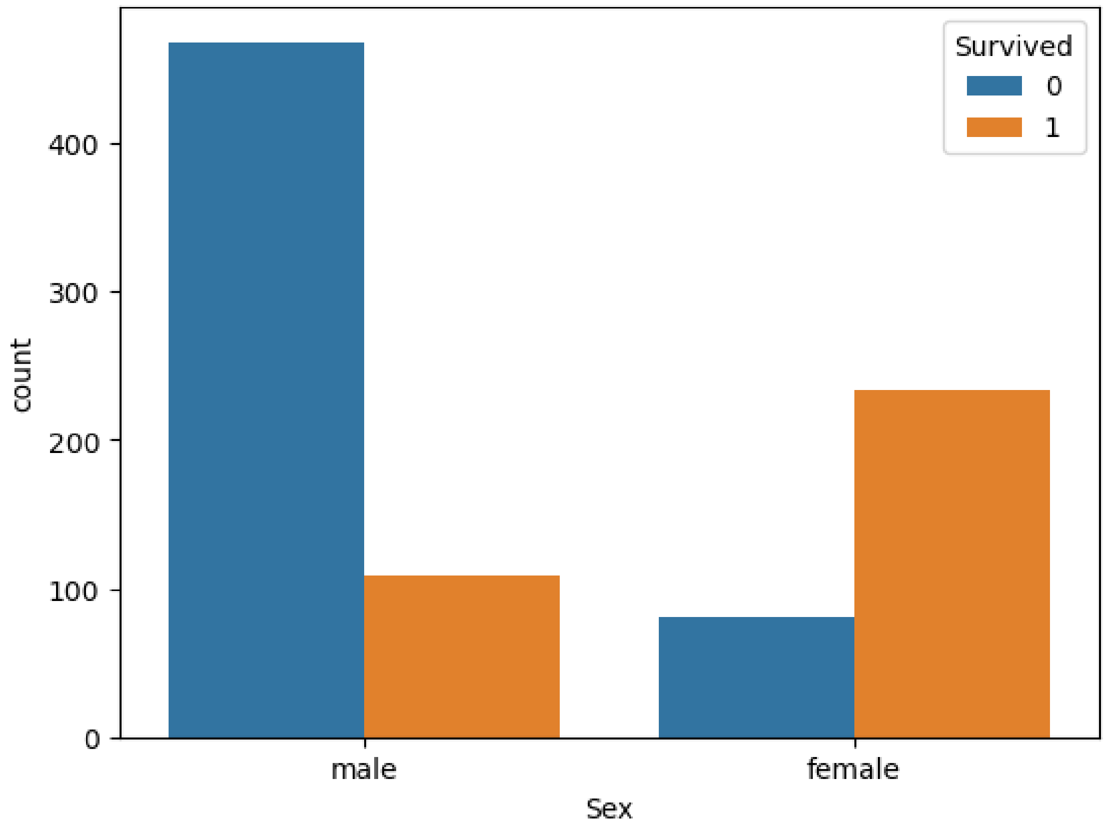
```
#from the plot, it is clear that, majority of the passengers were Male
```

In [25]:

```
sns.countplot(x='Sex',hue='Survived',data=dataset)
```

Out[25]:

```
<Axes: xlabel='Sex', ylabel='count'>
```



In [26]:

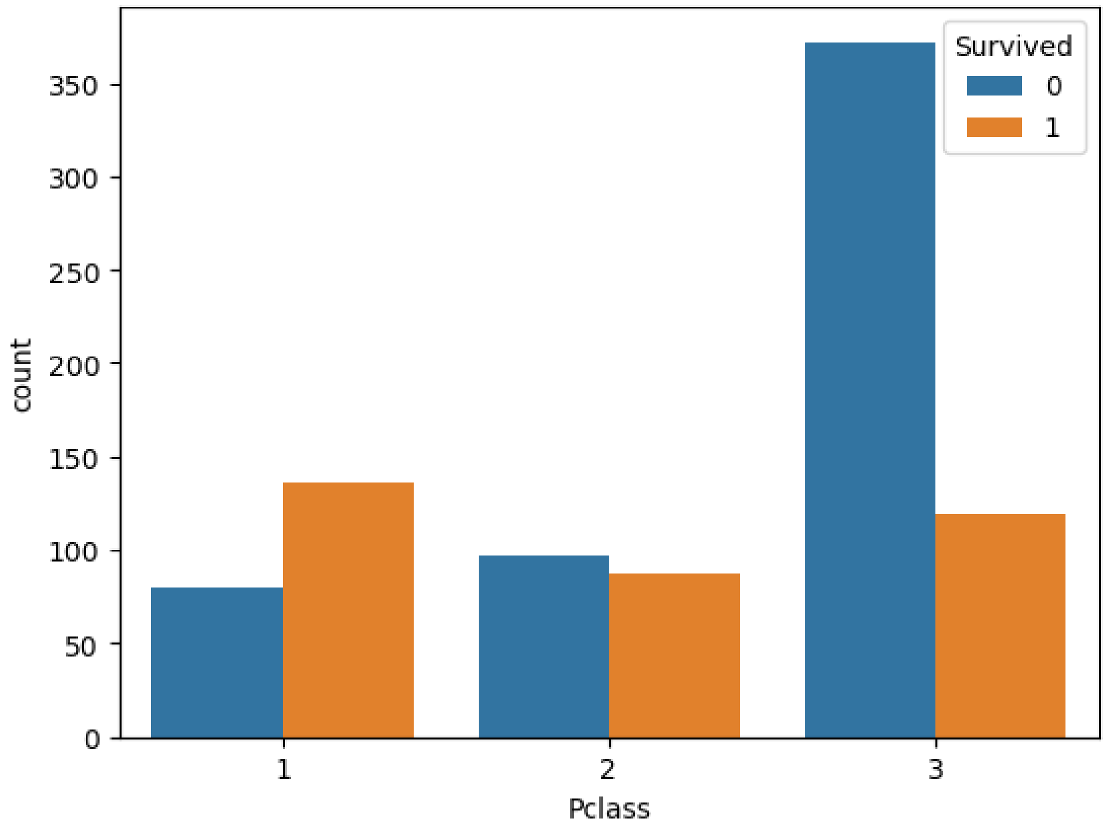
```
#from the plot it is clear that most of the survivers were female
```

In [27]:

```
sns.countplot(x='Pclass',hue='Survived',data=dataset)
```

Out[27]:

```
<Axes: xlabel='Pclass', ylabel='count'>
```



In [28]:

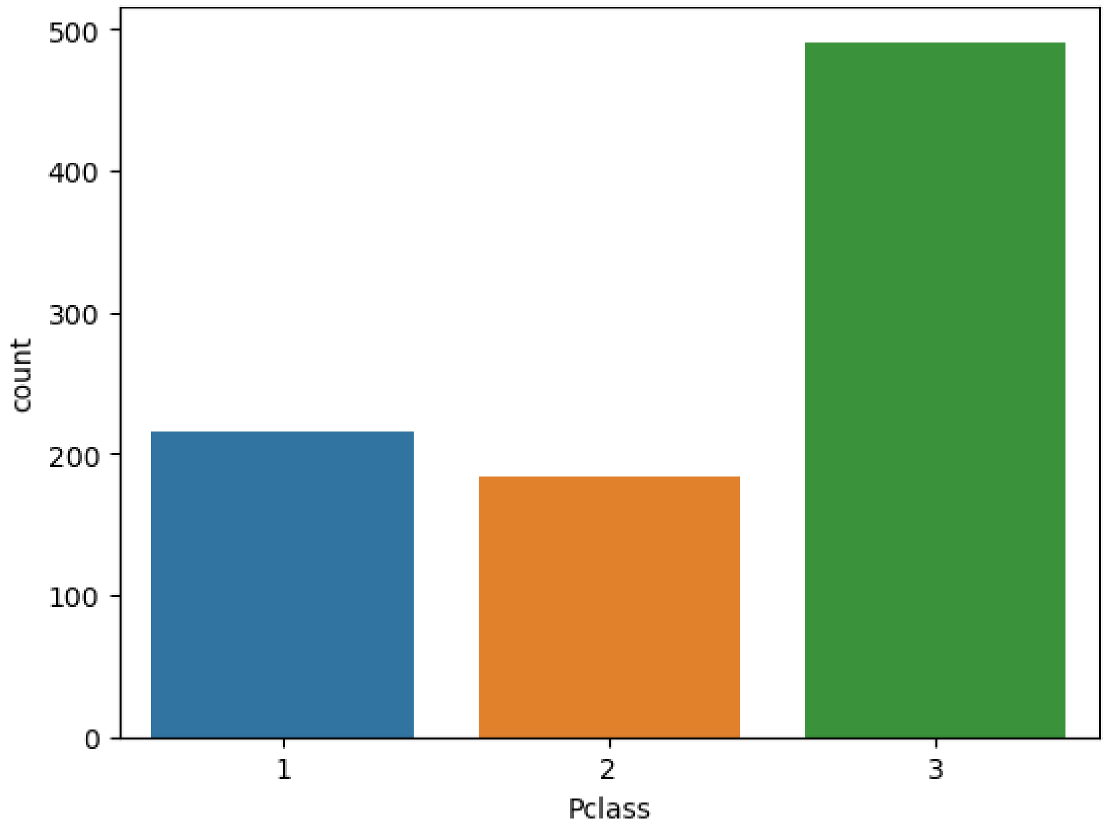
```
#from the above plot, we can infer that people who travelled in first class were the hi
```

In [29]:

```
sns.countplot(x='Pclass',data=dataset)
```

Out[29]:

```
<Axes: xlabel='Pclass', ylabel='count'>
```



In [30]:

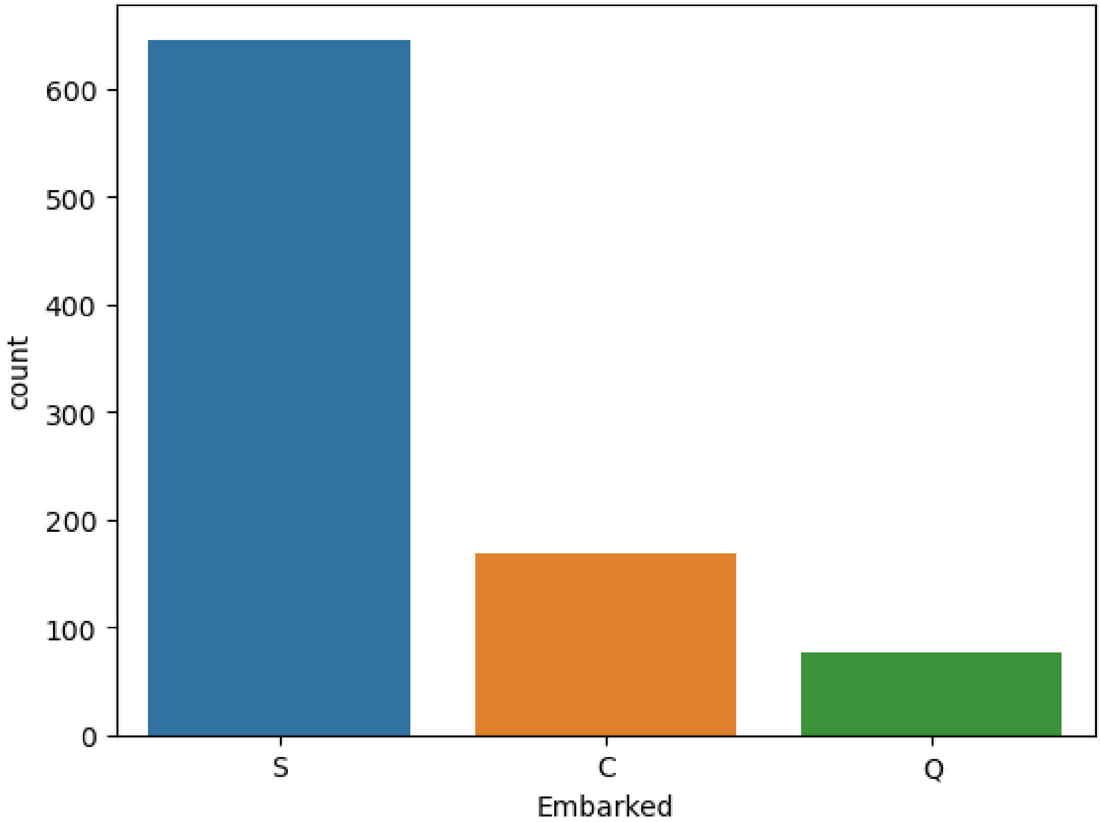
```
#from the plot, it is clear that majority of the people travelled in third class
```

In [31]:

```
sns.countplot(x='Embarked',data=dataset)
```

Out[31]:

```
<Axes: xlabel='Embarked', ylabel='count'>
```



In [32]:

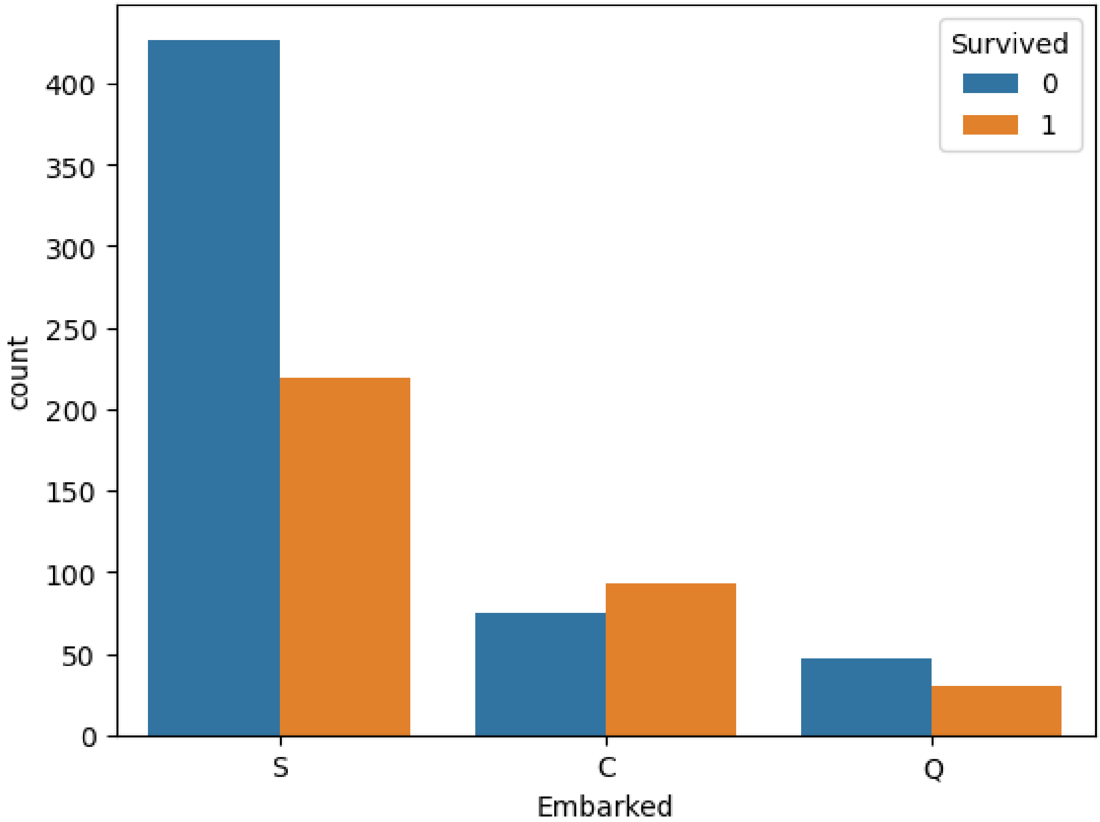
```
#from te above plot, we can infer that most the passengers embarked at Southampton
```

In [33]:

```
sns.countplot(x='Embarked',hue='Survived',data=dataset)
```

Out[33]:

```
<Axes: xlabel='Embarked', ylabel='count'>
```



In [34]:

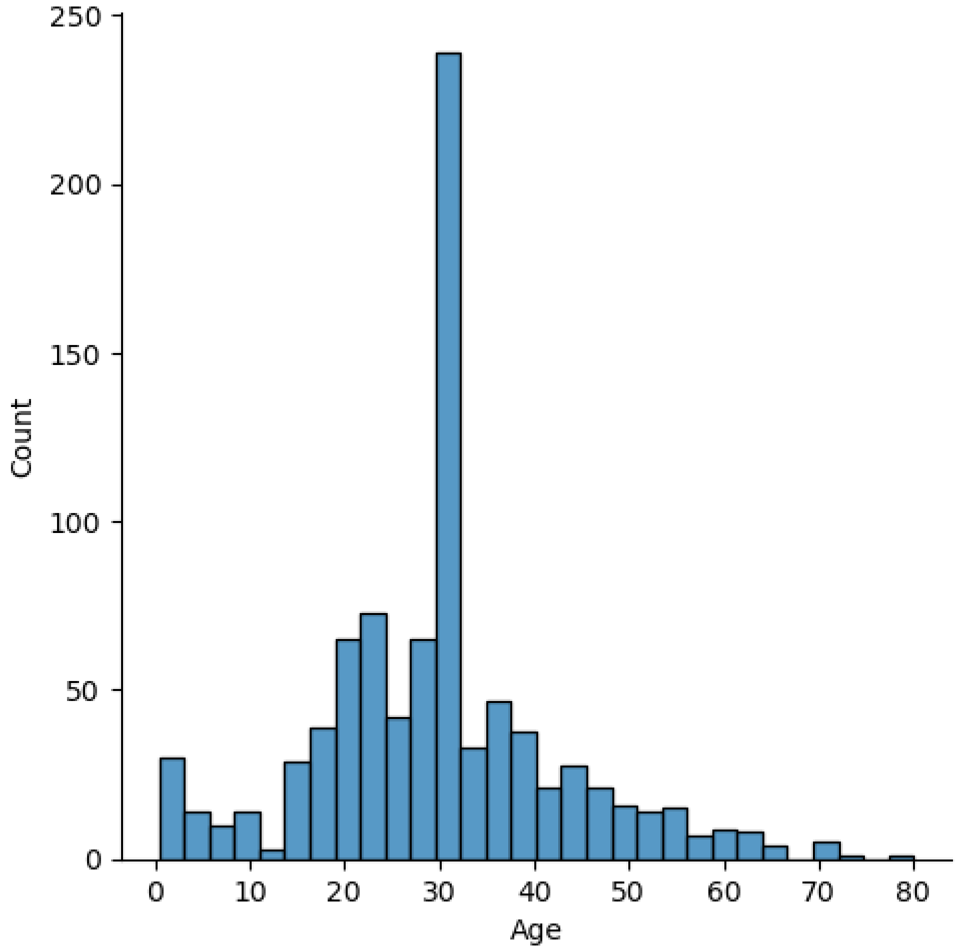
```
#from the plot, we can infer that, a large portion of survivers embarked at Southampton
```

In [35]:

```
sns.displot(dataset['Age'])
```

Out[35]:

<seaborn.axisgrid.FacetGrid at 0x2869139e4d0>



In [36]:

```
#from the above plot, it is clear that narly 1/4 th of the passengers belong to the age
```

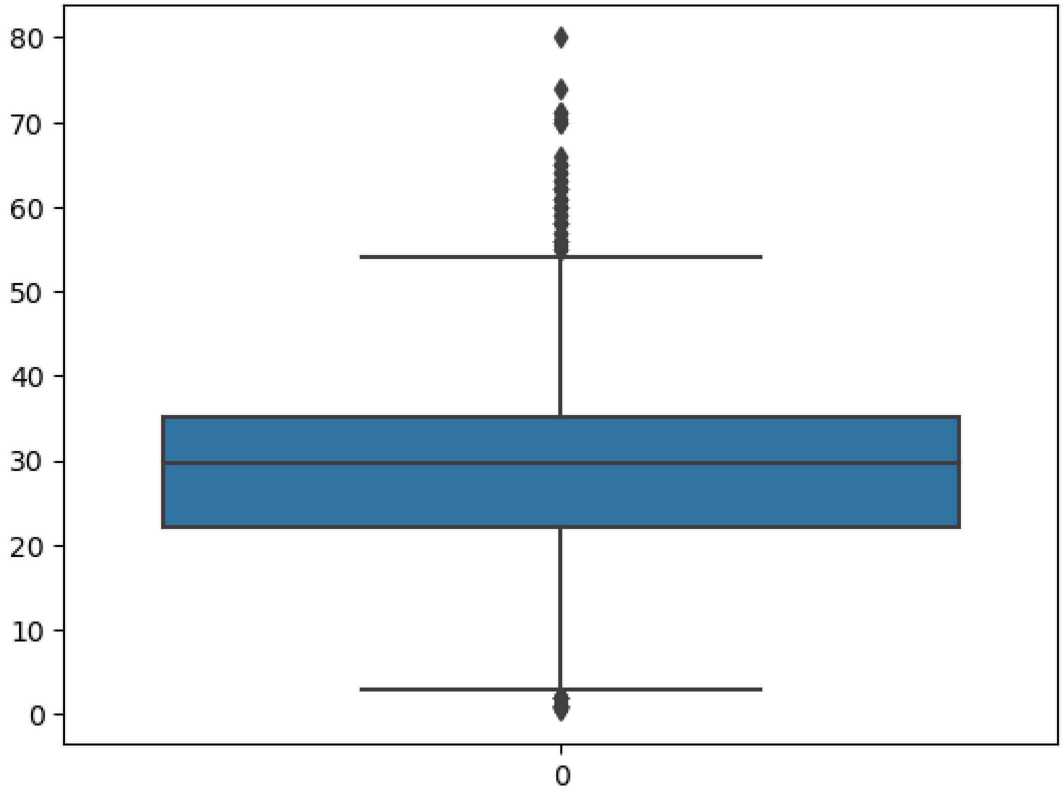
Step 5 - Outlier Detection.

In [37]:

```
sns.boxplot(dataset.Age)
```

Out[37]:

<Axes: >



In [38]:

```
q1 = dataset.Age.quantile(0.25)
q3 = dataset.Age.quantile(0.75)
iqr = q3 - q1
upperlimit = q3 + (1.5*iqr)
lowerlimit = q1 - (1.5*iqr)
```

In [39]:

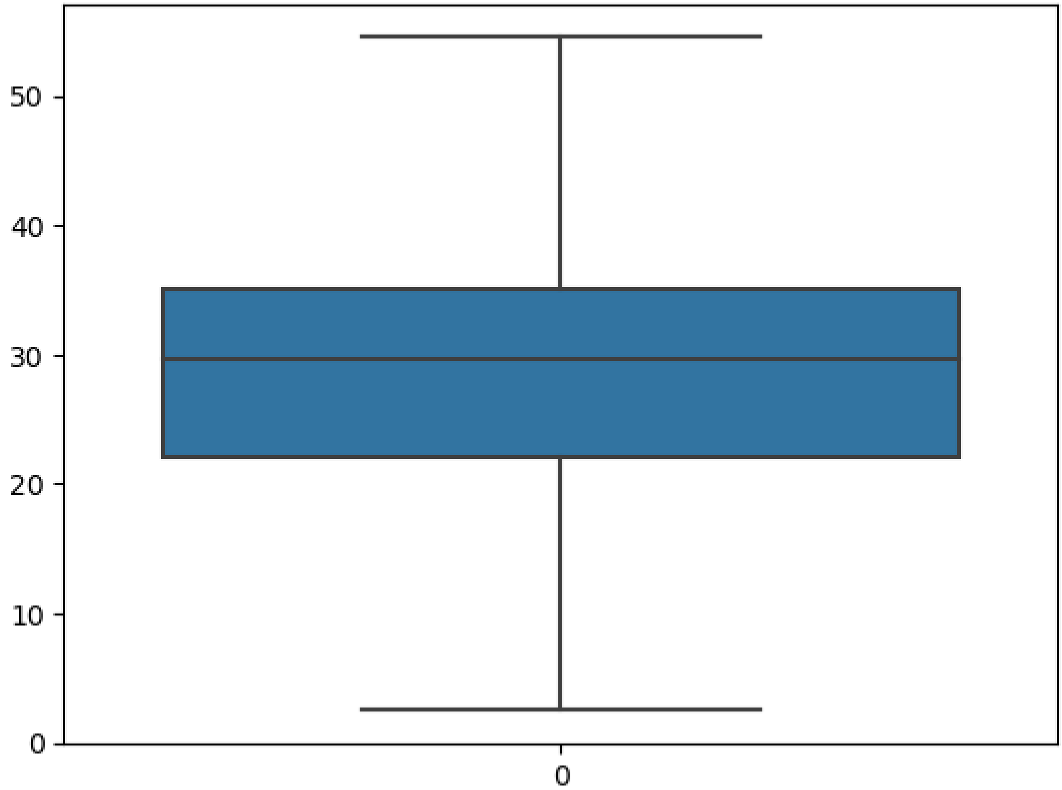
```
dataset['Age'] = np.where(dataset['Age'] > upperlimit , upperlimit,dataset['Age'])
dataset['Age'] = np.where(dataset['Age'] < lowerlimit, lowerlimit, dataset['Age'])
```

In [40]:

```
sns.boxplot(dataset['Age'])
```

Out[40]:

<Axes: >

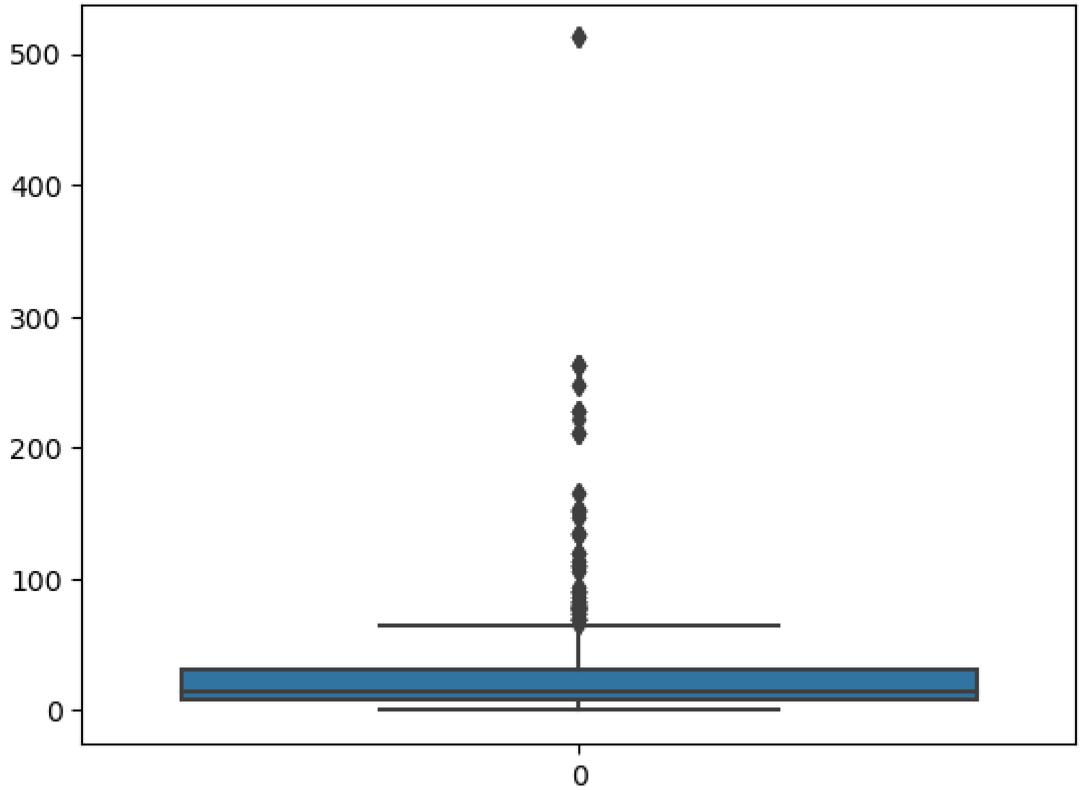


In [41]:

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

Out[41]:

```
<Axes: >
```



In [42]:

```
q1 = dataset.Fare.quantile(0.25)
q3 = dataset.Fare.quantile(0.75)
iqr = q3 - q1
upperlimit = q3 + (1.5*iqr)
lowerlimit = q1 - (1.5*iqr)
```

In [43]:

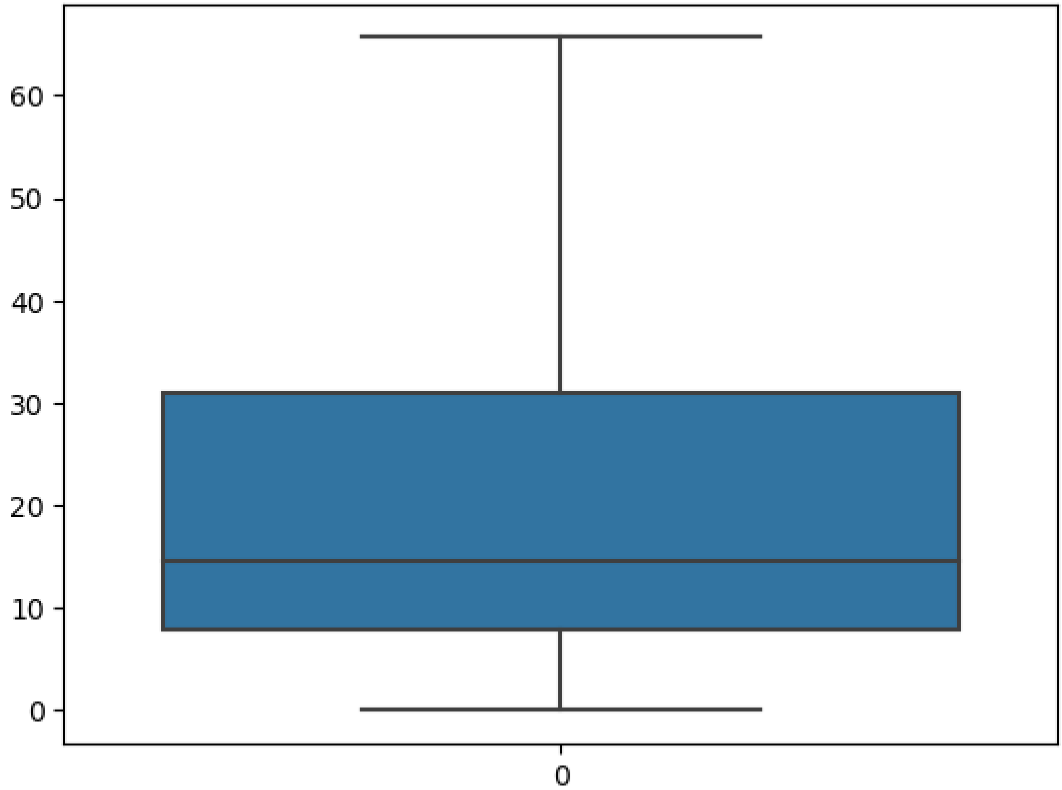
```
dataset['Fare'] = np.where(dataset['Fare'] > upperlimit , upperlimit,dataset['Fare'])
dataset['Fare'] = np.where(dataset['Fare'] < lowerlimit, lowerlimit, dataset['Fare'])
```

In [44]:

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

Out[44]:

<Axes: >



Step 6 - Splitting Dependent and Independent variables.

In [45]:

```
dataset.head()
```

Out[45]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	65.634
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050

In [46]:

```
x = dataset.iloc[: , 2:]
x.drop(columns=["Name","Ticket"],inplace= True) #independent variables
y = dataset.iloc[:,1:2] #dependent variables
```

Step 7 - Perform Encoding

In [47]:

```
le = LabelEncoder()
```

In [48]:

```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Pclass    891 non-null    int64  
 1   Sex       891 non-null    object 
 2   Age       891 non-null    float64
 3   SibSp     891 non-null    int64  
 4   Parch     891 non-null    int64  
 5   Fare      891 non-null    float64
 6   Embarked  891 non-null    object 
dtypes: float64(2), int64(3), object(2)
memory usage: 48.9+ KB
```

In [49]:

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

In [50]:

```
x['Embarked'] = le.fit_transform(x['Embarked'])
```

In [51]:

```
x.head()
```

Out[51]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	22.0	1	0	7.2500	2
1	1	0	38.0	1	0	65.6344	0
2	3	0	26.0	0	0	7.9250	2
3	1	0	35.0	1	0	53.1000	2
4	3	1	35.0	0	0	8.0500	2

Step 8 - Splitting Data into Train and Test.

In [52]:

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

In [53]:

```
x_train.shape , x_test.shape , y_train.shape , y_test.shape
```

Out[53]:

```
((712, 7), (179, 7), (712, 1), (179, 1))
```

Step 9 - Feature Scaling.

In [54]:

```
sc = StandardScaler()
```

In [55]:

```
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```