NAME: SHAIK MOHID BABU

REG. NO: 21BCE9569

Branch: Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning

Campus: VIT-AP

Email: mohidbabu.21bce9569@vitapstudent.ac.in

Data Preprocessing

- Import the Libraries.
- Importing the dataset.
- Checking for Null Values.
- Data Visualization.
- Outlier Detection
- Splitting Dependent and Independent variables
- Perform Encoding
- Feature Scaling.
- Splitting Data into Train and Test

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 MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

2.Importing the dataset.

In [2]:

```
data=pd.read_csv("Titanic-dataset.csv")
```

In [3]:

```
data
```

Out[3]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [4]:

```
data.head()
```

Out[4]:

	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 [5]:

```
data.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]:

```
data.describe()
```

Out[6]:

	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
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

3. Checking for Null Values.

In [7]:

```
data.isnull()
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	True	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False

891 rows × 12 columns

In [8]:

```
data.isnull().any()
```

Out[8]:

```
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 [9]:

```
data.isnull().sum()
```

Out[9]:

```
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 [10]:

```
len(data["Cabin"])
```

Out[10]:

```
891
```

In [11]:

```
data["Cabin"].isnull().sum()
```

Out[11]:

```
687
```

Since the Cabin column contains 77% null values, it is advisable to remove this column for a more efficient dataset.

In [12]:

```
data.drop(columns=["Cabin"],inplace=True)
```

In [13]:

```
data.head()
```

Out[13]:

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

In [14]:

```
data.isnull().sum()
```

Out[14]:

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

In [15]:

```
len(data["Age"])
```

Out[15]:

```
891
```

In [16]:

```
data["Age"].isnull().sum()
```

Out[16]:

```
177
```

To handle the 19% null values in the "Age" column, we are applying imputation by replacing them with the mean value of the existing data in the "Age" column.

In [17]:

```
data["Age"].fillna(data["Age"].median(),inplace=True)
```

In [18]:

```
data["Age"]
```

Out[18]:

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    28.0
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64
```

In [19]:

```
data["Age"].isnull().sum()
```

Out[19]:

```
0
```

In [20]:

```
data.isnull().sum()
```

Out[20]:

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

Removing the null values present in the Embarked column

In [21]:

```
data["Embarked"].value_counts()
```

Out[21]:

```
Embarked
S    644
C    168
Q     77
Name: count, dtype: int64
```

In [22]:

```
data.dropna(subset=["Embarked"],inplace=True)
```

In [23]:

```
data
```

Out[23]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S
...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	28.0	1	2	W./C. 6607	23.4500	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	Q

889 rows × 11 columns

In [24]:

```
data.isnull().sum()
```

Out[24]:

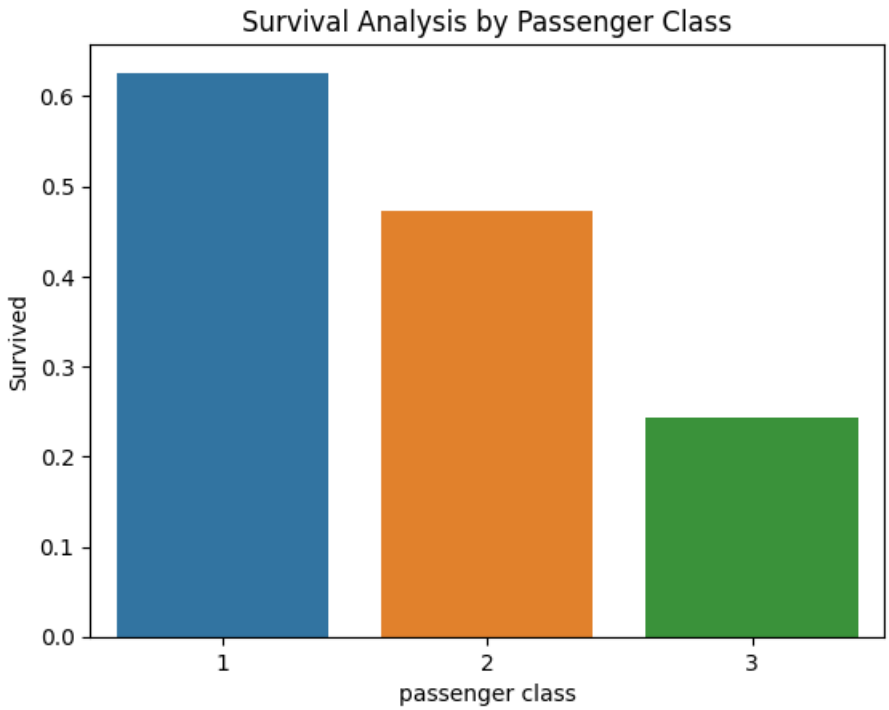
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

4. Data Visualization.

In [25]:

```
sns.barplot(x="Pclass",y="Survived",data=data,ci=0)
plt.xlabel("passenger class")
plt.ylabel("Survived")
plt.title("Survival Analysis by Passenger Class")
plt.show()
```

```
C:\Users\MOHID BABU SHAIK\AppData\Local\Temp\ipykernel_30528\1628323338.py:1: FutureWarning:

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

  sns.barplot(x="Pclass",y="Survived",data=data,ci=0)
```




```
Inference: Assessing Passenger Class Impact on Survival Rates
```

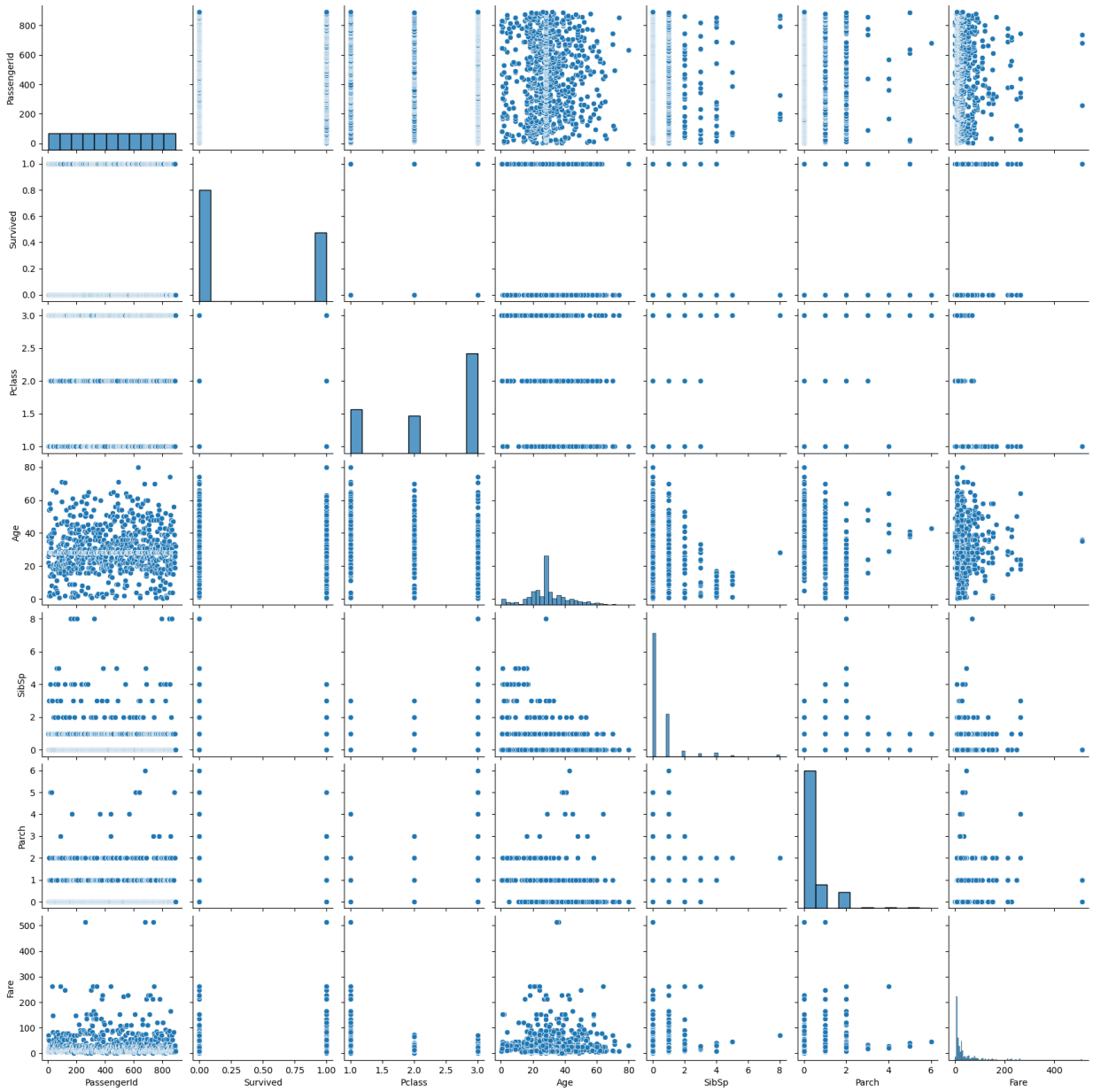
In [26]:

```
sns.pairplot(data)
```

```
C:\Users\MOHID BABU SHAIK\miniconda3\lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The fig
ure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

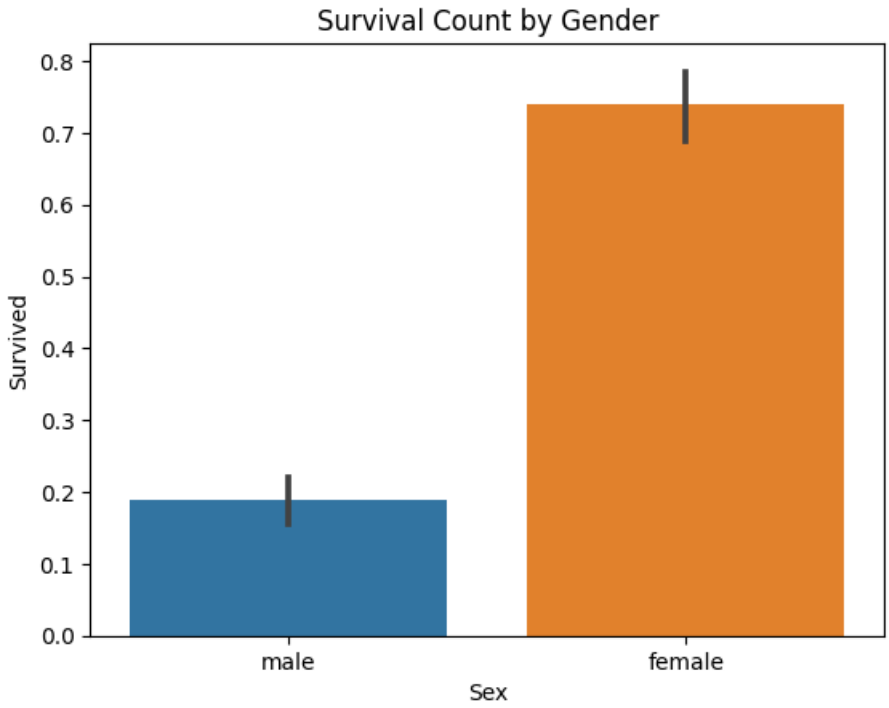
Out[26]:

```
<seaborn.axisgrid.PairGrid at 0x22ce362b520>
```



In [27]:

```
sns.barplot(x="Sex",y="Survived",data=data)
plt.xlabel("Sex")
plt.ylabel("Survived")
plt.title("Survival Count by Gender")
plt.show()
```




```
Inference: Based on the analysis of the data, it becomes evident that a significantly higher percentage of
females survived the crash when compared to their male counterparts
```

In [28]:

```
(data.select_dtypes(include=['number'])).corr()
```

Out[28]:

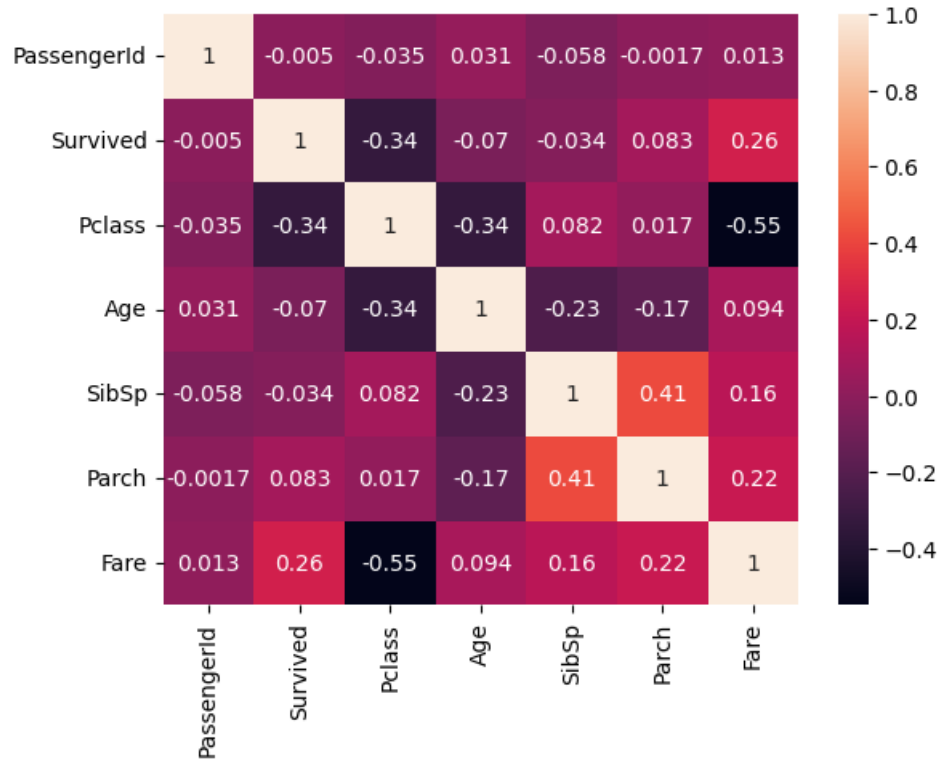
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005028	-0.035330	0.031319	-0.057686	-0.001657	0.012703
Survived	-0.005028	1.000000	-0.335549	-0.069822	-0.034040	0.083151	0.255290
Pclass	-0.035330	-0.335549	1.000000	-0.336512	0.081656	0.016824	-0.548193
Age	0.031319	-0.069822	-0.336512	1.000000	-0.232543	-0.171485	0.093707
SibSp	-0.057686	-0.034040	0.081656	-0.232543	1.000000	0.414542	0.160887
Parch	-0.001657	0.083151	0.016824	-0.171485	0.414542	1.000000	0.217532
Fare	0.012703	0.255290	-0.548193	0.093707	0.160887	0.217532	1.000000

In [29]:

```
sns.heatmap((data.select_dtypes(include=['number'])).corr(),annot=True)
```

Out[29]:

```
<Axes: >
```

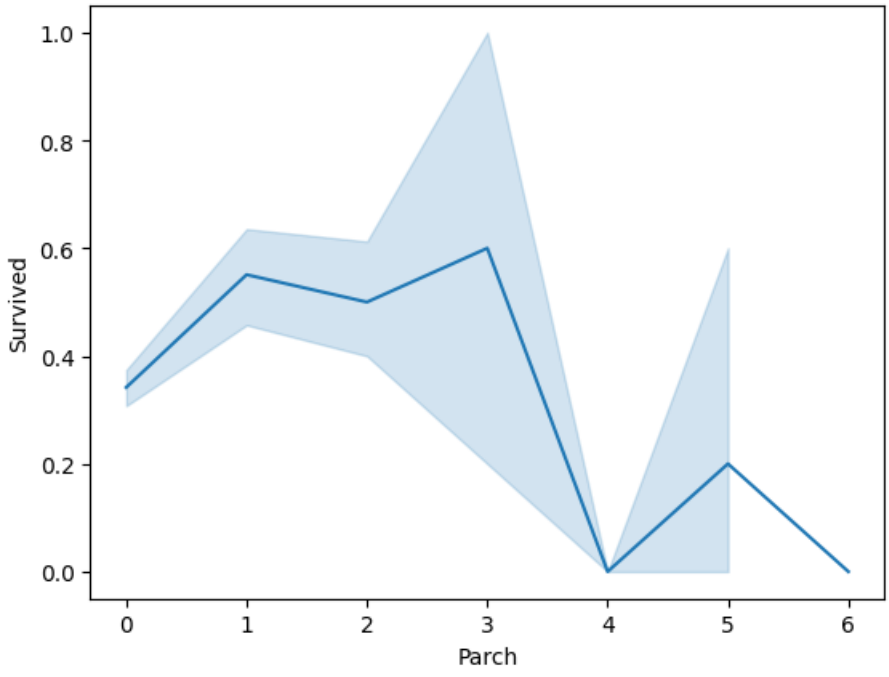


In [30]:

```
sns.lineplot(x="Parch",y="Survived",data=data)
```

Out[30]:

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



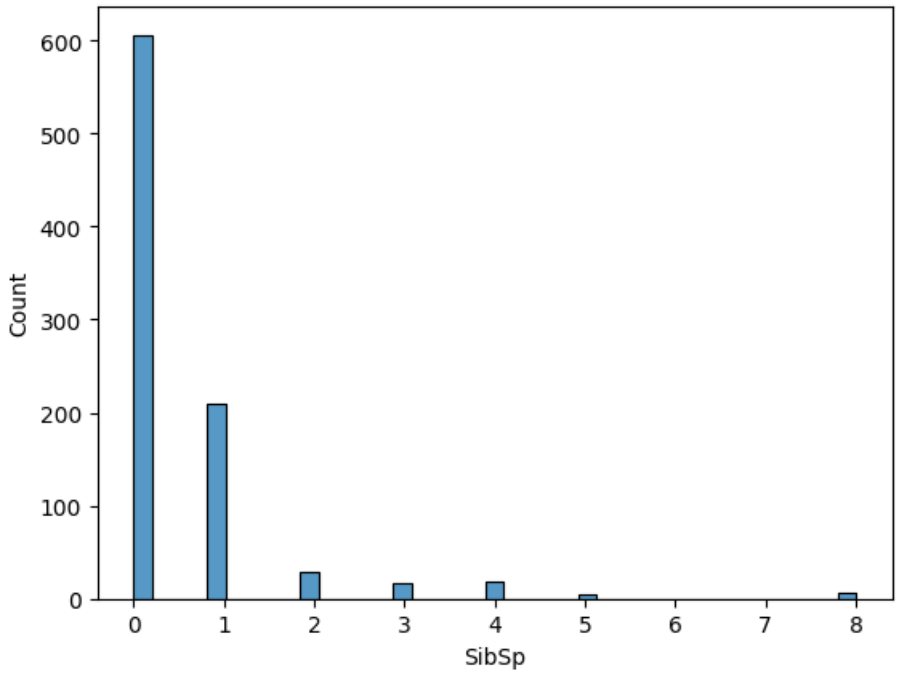

```
Inference: This graph illustrates the number of parents and children aboard who survived.
```

In [31]:

```
sns.histplot(data["SibSp"])
```

Out[31]:

```
<Axes: xlabel='SibSp', ylabel='Count'>
```



Inference: This graph illustrates the number of siblings and spouses aboard the ship.

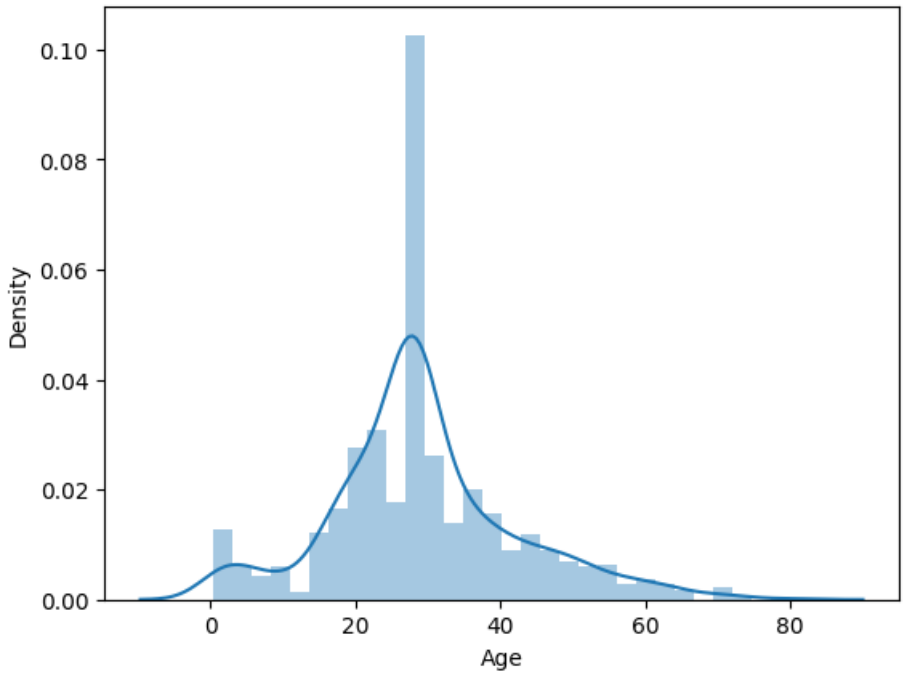
In [32]:

```
sns.distplot(data.Age)
```

```
C:\Users\MOHID BABU SHAIK\AppData\Local\Temp\ipykernel_30528\4003340619.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(data.Age)
```

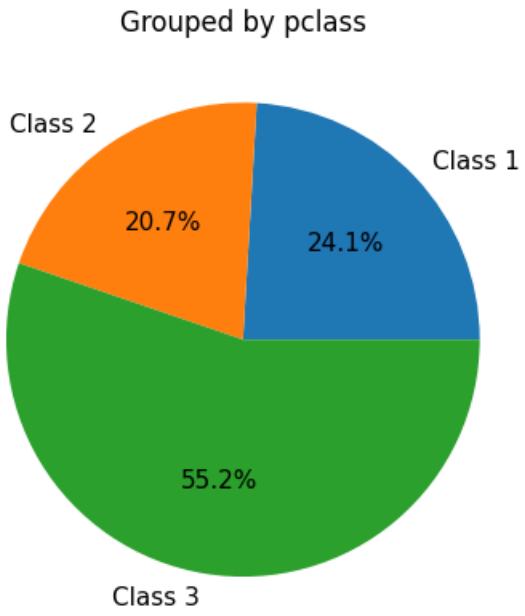
Out[32]:

```
<Axes: xlabel='Age', ylabel='Density'>
```



In [33]:

```
pclass_count = data.groupby('Pclass')['Pclass'].count()
plt.title('Grouped by pclass')
plt.pie(pclass_count.values, labels=['Class 1', 'Class 2', 'Class 3'],  autopct='%1.1f%%', textprops={'fontsize':1
plt.show()
```



In [34]:

```
plt.title('Age vs Survived')
sns.violinplot(x ="Sex", y ="Age", hue ="Survived",data = data, split = True)
```

Out[34]:

```
<Axes: title={'center': 'Age vs Survived'}, xlabel='Sex', ylabel='Age'>
```

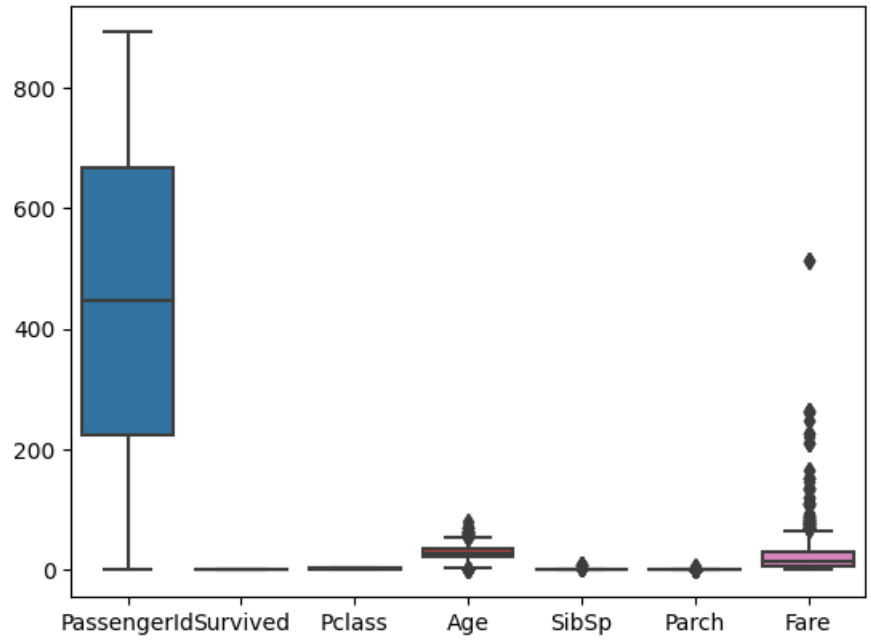
5. Outlier Detection

In [35]:

```
sns.boxplot(data)
```

Out[35]:

<Axes: >

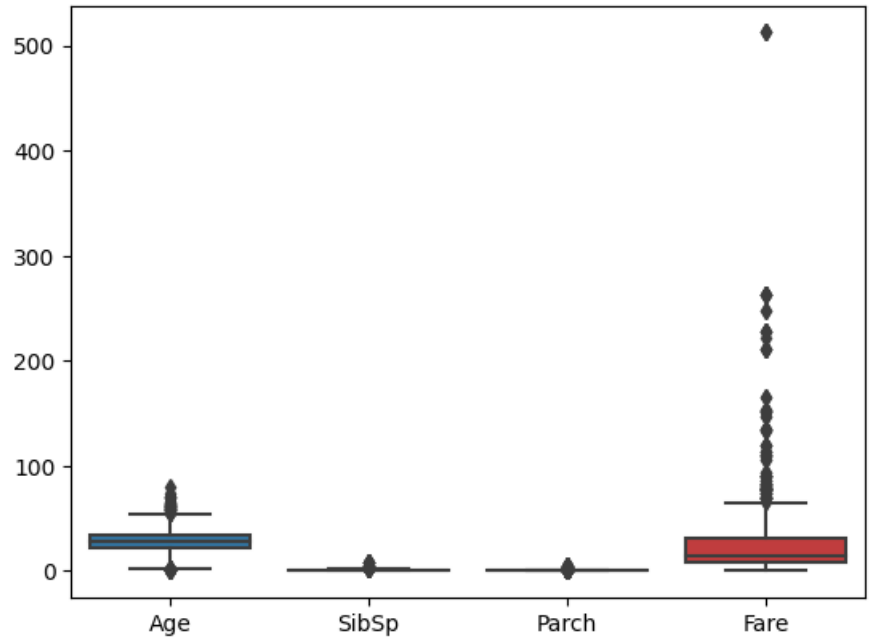


In [36]:

```
sns.boxplot(data.iloc[:,3:])
```

Out[36]:

<Axes: >



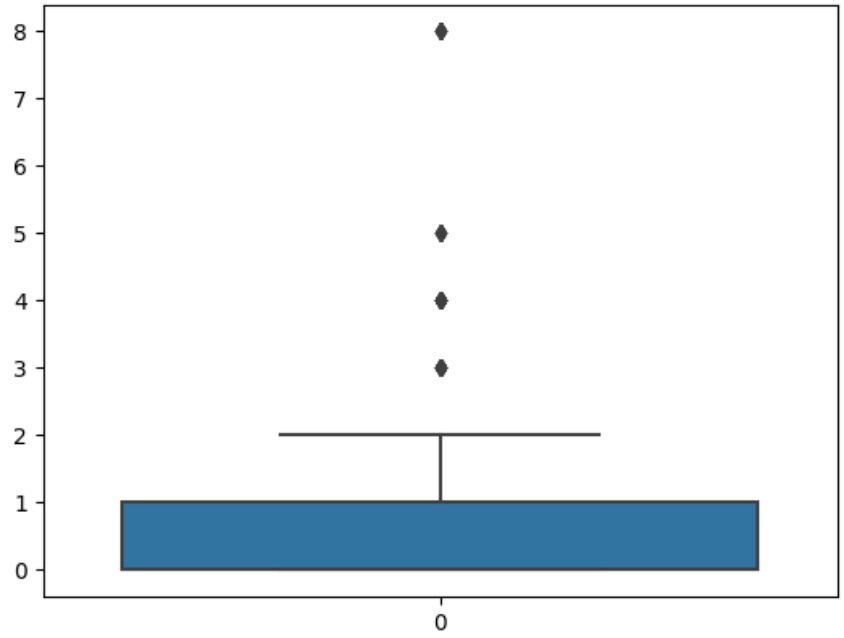
Outlier present in the SibSp Column

In [37]:

```
sns.boxplot(data.SibSp)
```

Out[37]:

```
<Axes: >
```



In [38]:

```
q1 = data.SibSp.quantile(0.25)
q3 = data.SibSp.quantile(0.75)
```

In [39]:

```
print(q1)
print(q3)
```

```
0.0
1.0
```

In [40]:

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

```
1.0
```

In [41]:

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

```
2.5
```

In [42]:

```
data["SibSp"].median()
```

Out[42]:

```
0.0
```

In [43]:

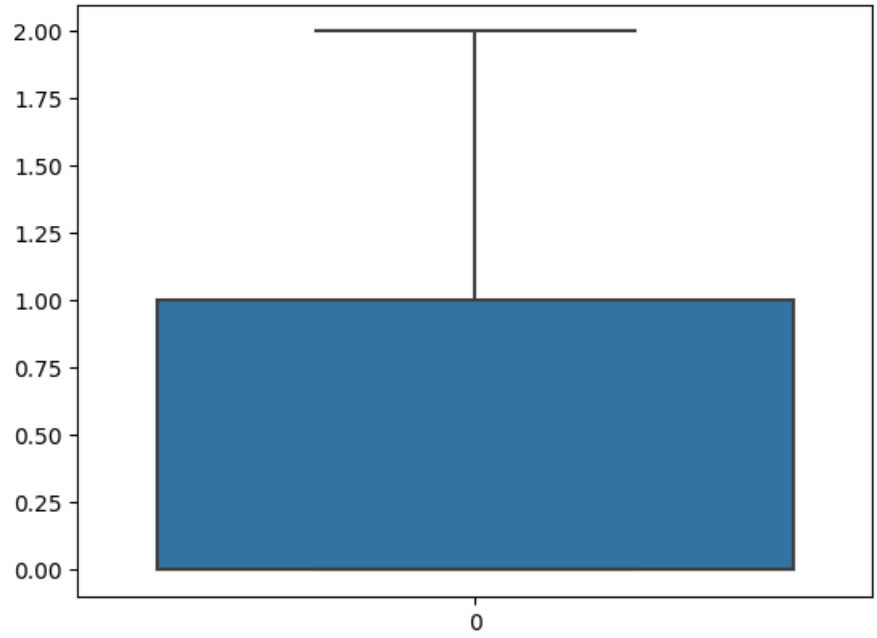
```
data["SibSp"]=np.where(data['SibSp']>upper_limit,0,data['SibSp'])
```

In [44]:

```
sns.boxplot(data.SibSp)
```

Out[44]:

```
<Axes: >
```



In [45]:

```
data.shape
```

Out[45]:

```
(889, 11)
```

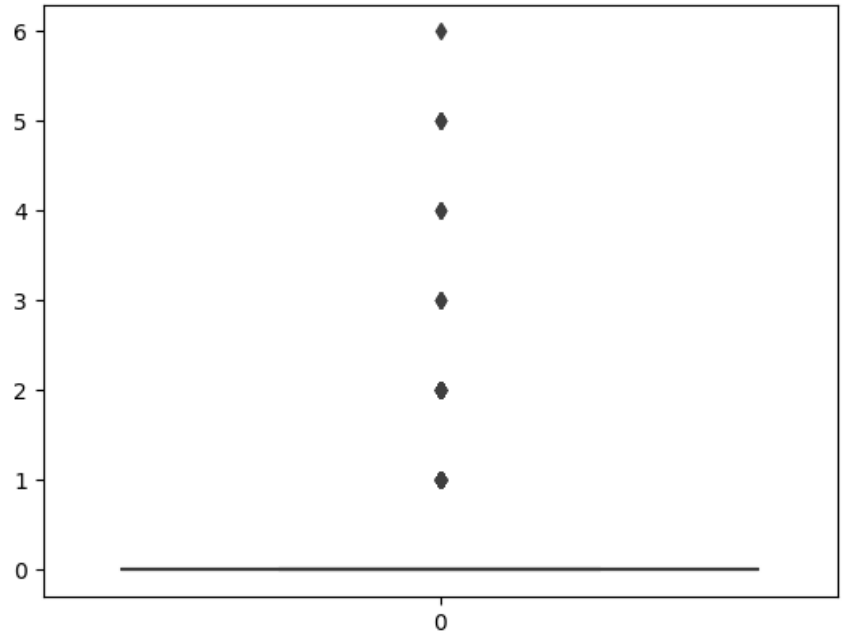
Outlier Present in the Parch Column

In [46]:

```
sns.boxplot(data.Parch)
```

Out[46]:

<Axes: >



In [47]:

```
q1 = data.Parch.quantile(0.25)
q3 = data.Parch.quantile(0.75)
```

In [48]:

```
print(q1)
print(q3)
```

```
0.0
0.0
```

In [49]:

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

```
0.0
```

In [50]:

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

```
0.0
```

In [51]:

```
data["Parch"].median()
```

Out[51]:

```
0.0
```

In [52]:

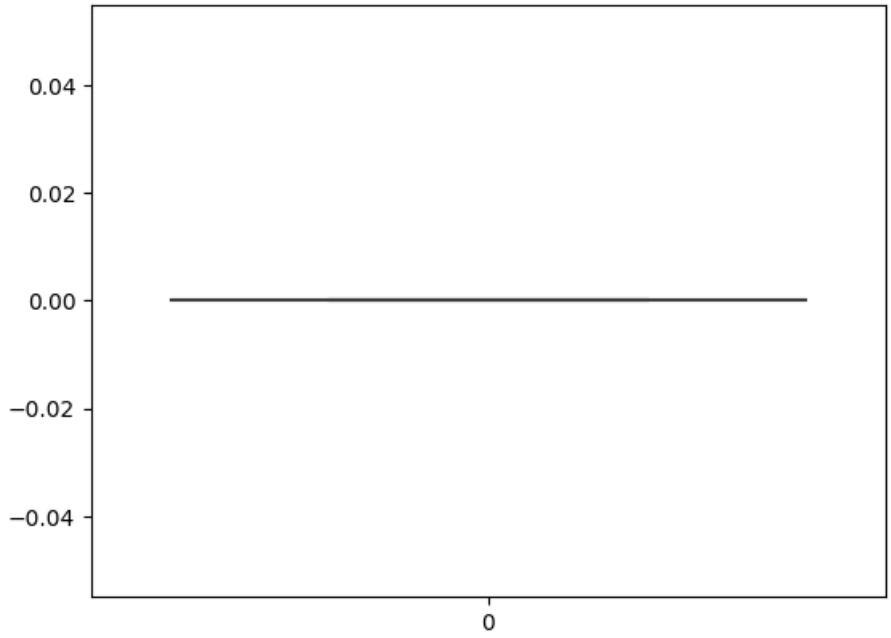
```
data["Parch"]=np.where(data['Parch']>upper_limit,0,data['Parch'])
```

In [53]:

```
sns.boxplot(data.Parch)
```

Out[53]:

<Axes: >



Outlier Present in the Age column

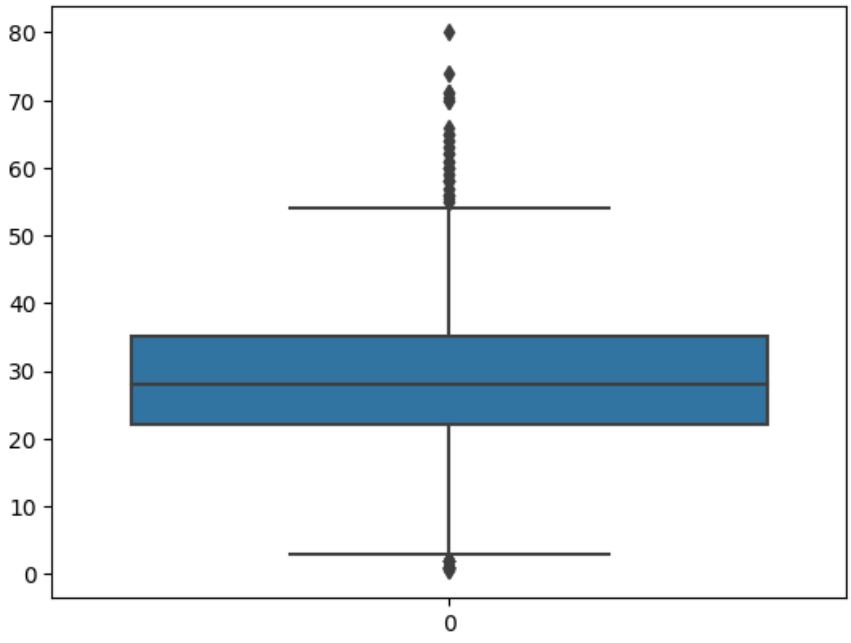
Removing the Outlier using the IQR Method(replaced with Median)

In [54]:

```
sns.boxplot(data.Age)
```

Out[54]:

<Axes: >



In [55]:

```
q1 = data.Age.quantile(0.25)
q3 = data.Age.quantile(0.75)
```

In [56]:

```
print(q1)
print(q3)
```

```
22.0
35.0
```

In [57]:

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

```
13.0
```

In [58]:

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

```
54.5
```

In [59]:

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

```
2.5
```

In [60]:

```
data.Age.median()
```

Out[60]:

```
28.0
```

In [61]:

```
data1=data.copy()
```

In [62]:

```
data1["Age"]=np.where(data1["Age"]>upper_limit,28,data1["Age"])
```

In [63]:

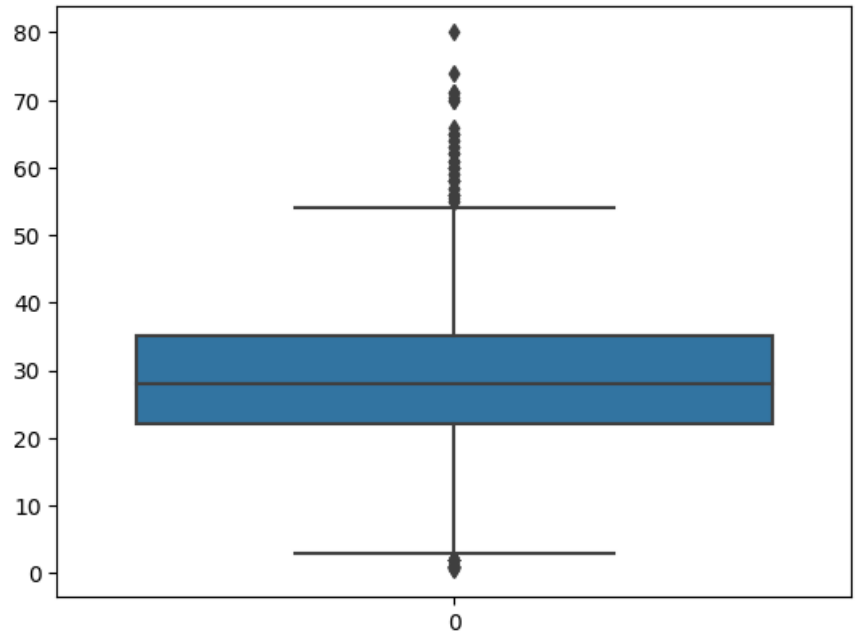
```
data1["Age"]=np.where(data1["Age"]<lower_limit,28,data1["Age"])
```

In [64]:

```
sns.boxplot(data.Age)
```

Out[64]:

<Axes: >



Removing the outlier using the IQR method(dropping the outliers)

In [65]:

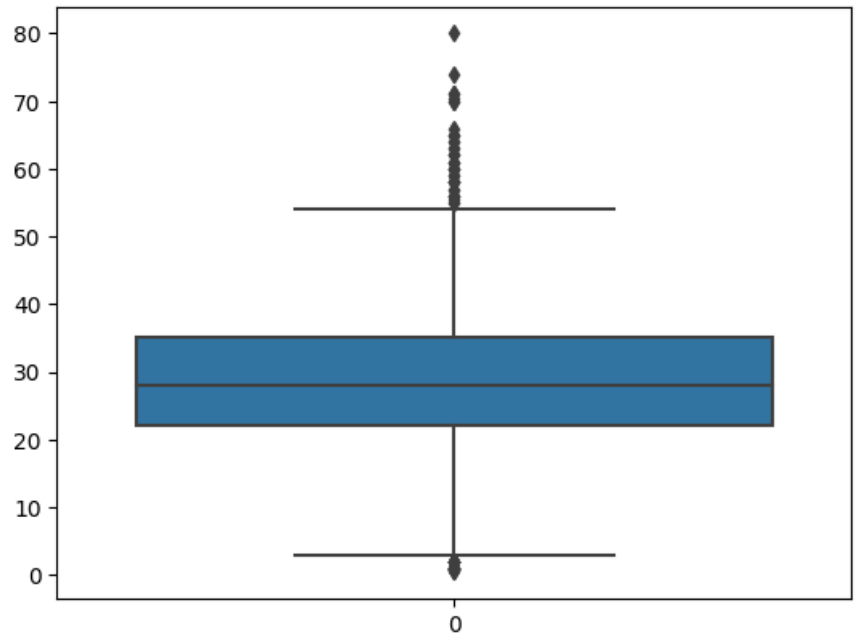
```
data2=data.copy()
```

In [66]:

```
sns.boxplot(data2.Age)
```

Out[66]:

<Axes: >



In [67]:

```
q1 = data2.Age.quantile(0.25)
q3 = data2.Age.quantile(0.75)
```

In [68]:

```
print(q1)
print(q3)
```

```
22.0
35.0
```

In [69]:

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

```
13.0
```

In [70]:

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

```
54.5
```

In [71]:

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

```
2.5
```

In [72]:

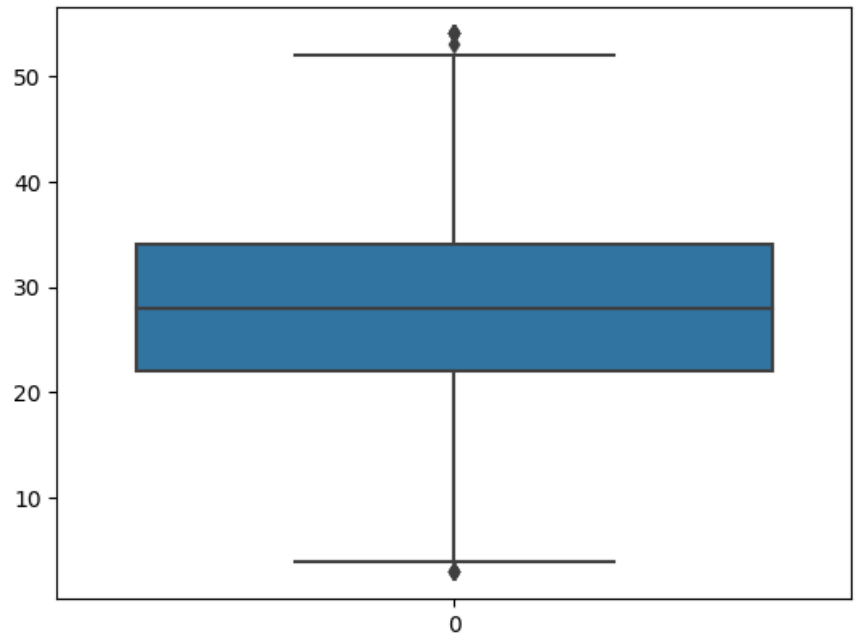
```
data2 = data2[(lower_limit < data2["Age"]) & (data2["Age"] < upper_limit)]
```

In [73]:

```
sns.boxplot(data2.Age)
```

Out[73]:

```
<Axes: >
```



In [74]:

```
data2.shape
```

Out[74]:

```
(824, 11)
```

In [75]:

```
data.shape
```

Out[75]:

```
(889, 11)
```

Removing the Outlier using Z-score

In [76]:

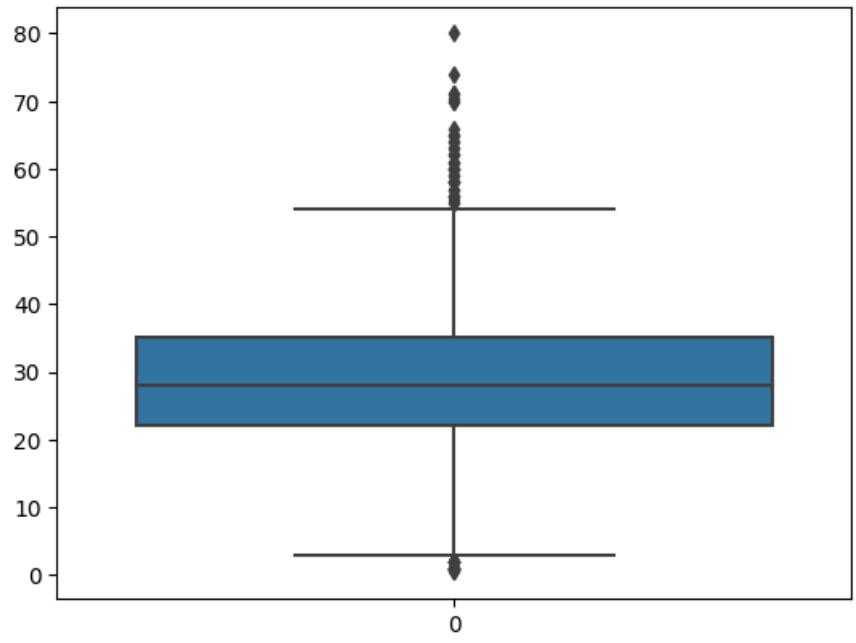
```
data3=data.copy()
```

In [77]:

```
sns.boxplot(data3.Age)
```

Out[77]:

```
<Axes: >
```



In [78]:

```
from scipy import stats
```

In [79]:

```
age_zscore = stats.zscore(data3.Age)
age_zscore
```

Out[79]:

```
0     -0.563674
1      0.669217
2     -0.255451
3      0.438050
4      0.438050
         ...
886   -0.178396
887   -0.794841
888   -0.101340
889   -0.255451
890    0.206883
Name: Age, Length: 889, dtype: float64
```

In [80]:

```
data3 = data3[np.abs(age_zscore)<=3]
```

In [81]:

```
data3
```

Out[81]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S
...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	28.0	1	0	W./C. 6607	23.4500	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	Q

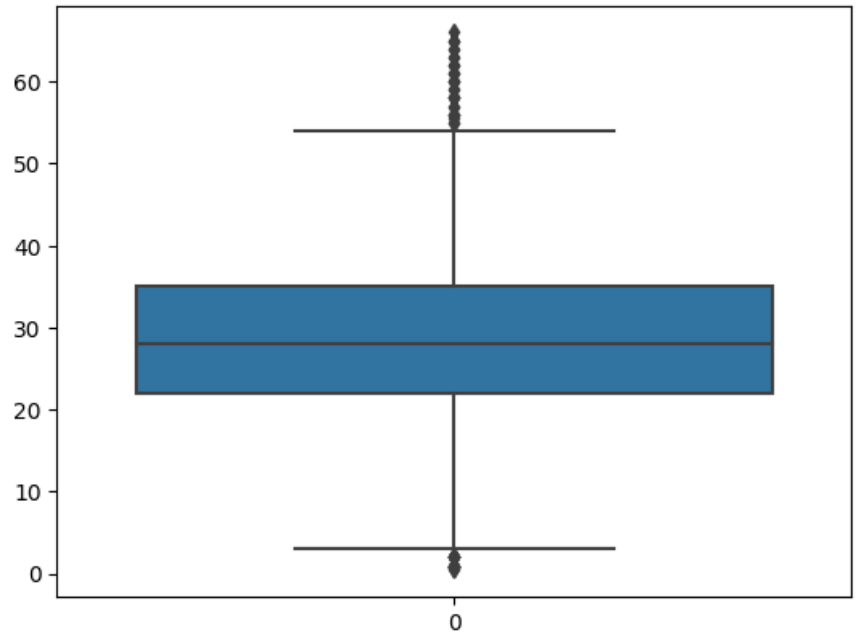
882 rows × 11 columns

In [82]:

```
sns.boxplot(data3.Age)
```

Out[82]:

```
<Axes: >
```



In [83]:

```
data3.shape
```

Out[83]:

```
(882, 11)
```

Removing the Outlier using Percentile Method

In [84]:

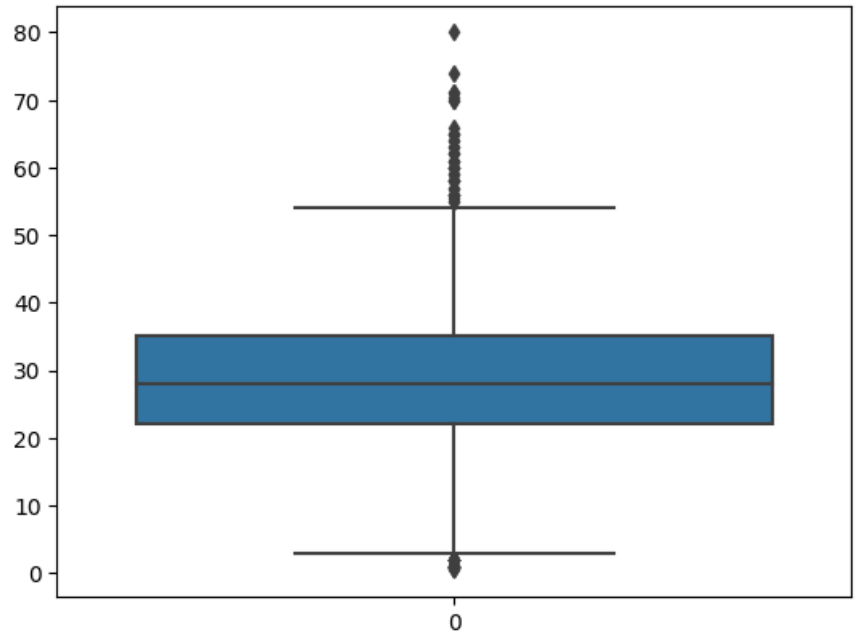
```
data4=data.copy()
```

In [85]:

```
sns.boxplot(data4.Age)
```

Out[85]:

```
<Axes: >
```



In [86]:

```
p99=data4.Age.quantile(0.99)
```

In [87]:

```
p99
```

Out[87]:

```
65.0
```

In [88]:

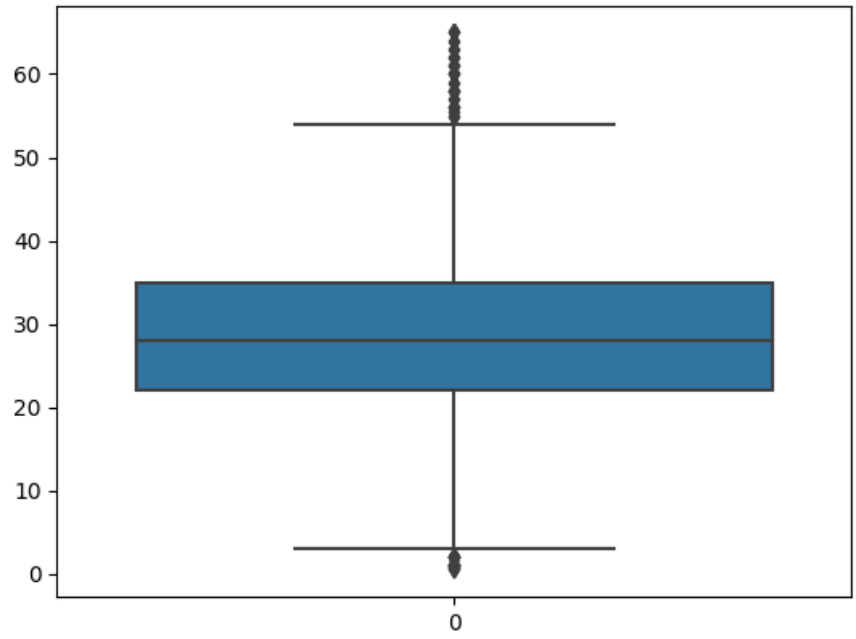
```
data4 = data4[data4.Age<=p99]
```

In [89]:

```
sns.boxplot(data4.Age)
```

Out[89]:

<Axes: >



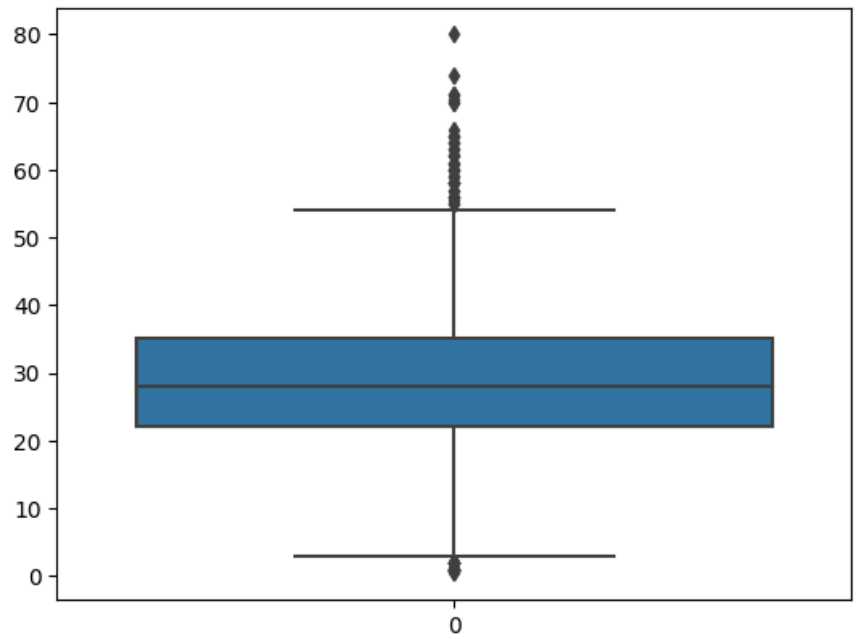
The IQR method for outlier removal consistently yields superior results compared to the other methods

In [90]:

```
sns.boxplot(data.Age)
```

Out[90]:

<Axes: >



In [91]:

```
q1 = data.Age.quantile(0.25)
q3 = data.Age.quantile(0.75)
```

In [92]:

```
print(q1)
print(q3)
```

```
22.0
35.0
```

In [93]:

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

```
13.0
```

In [94]:

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

```
2.5
```

In [95]:

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

```
54.5
```

In [96]:

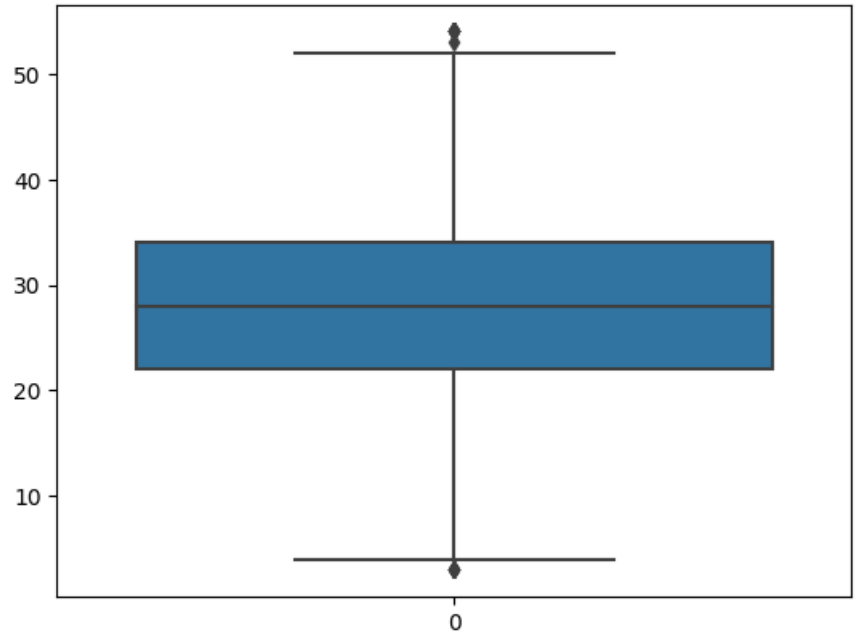
```
data = data[(lower_limit < data["Age"]) & (data["Age"] < upper_limit)]
```

In [97]:

```
sns.boxplot(data.Age)
```

Out[97]:

```
<Axes: >
```



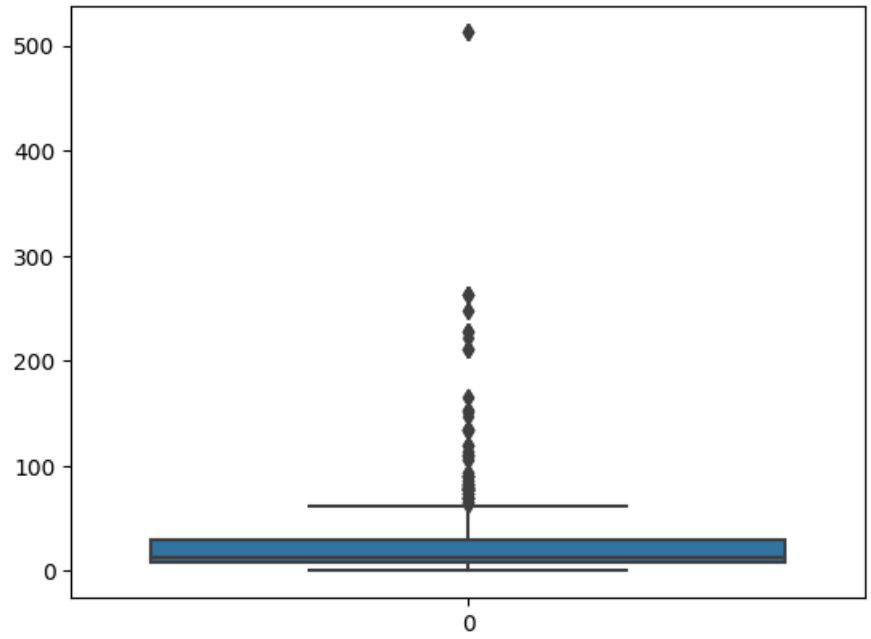
Outlier in Fare Column

In [98]:

```
sns.boxplot(data.Fare)
```

Out[98]:

```
<Axes: >
```



Removing the Outlier using IQR Method(by replacing them with the median value)

In [99]:

```
data5=data.copy()
```

In [100]:

```
q1 = data5.Fare.quantile(0.25)
q3 = data2.Fare.quantile(0.75)
```

In [101]:

```
print(q1)
print(q3)
```

```
7.8958
30.017699999999998
```

In [102]:

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

```
22.121899999999997
```

In [103]:

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

```
63.20054999999999
```

In [104]:

```
data5.Fare.median()
```

Out[104]:

```
13.20835
```

In [105]:

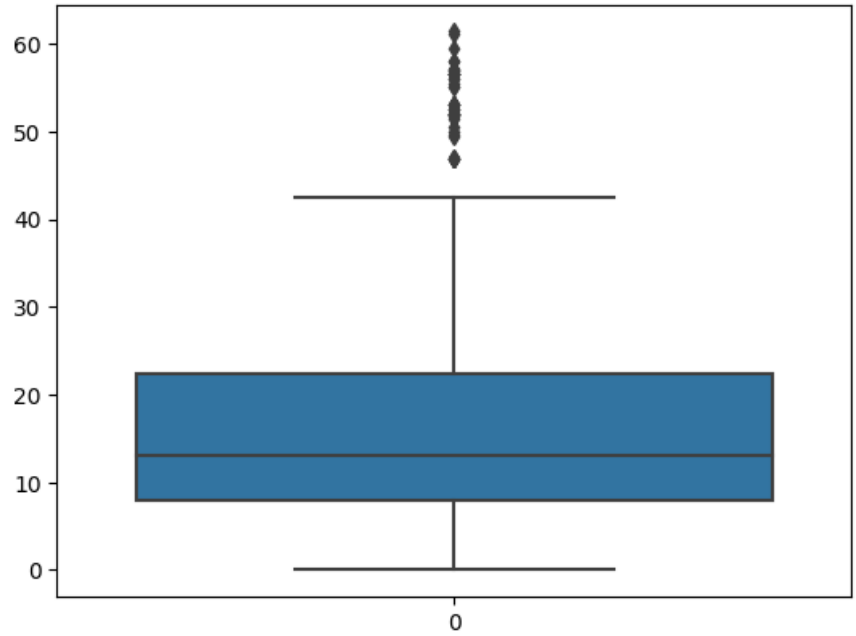
```
data5["Fare"]=np.where(data5["Fare"]>upper_limit,13,data5["Fare"])
```

In [106]:

```
sns.boxplot(data5.Fare)
```

Out[106]:

```
<Axes: >
```



Removing the Outlier using the IQR method(dropping the outliers)

In [107]:

```
df=data.copy()
```

In [108]:

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

In [109]:

```
print(q1)
print(q3)
```

```
7.8958
30.017699999999998
```

In [110]:

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

```
22.121899999999997
```

In [111]:

```
upper_limit1 = q3+1.5*IQR
print(upper_limit)
```

```
63.20054999999999
```

In [112]:

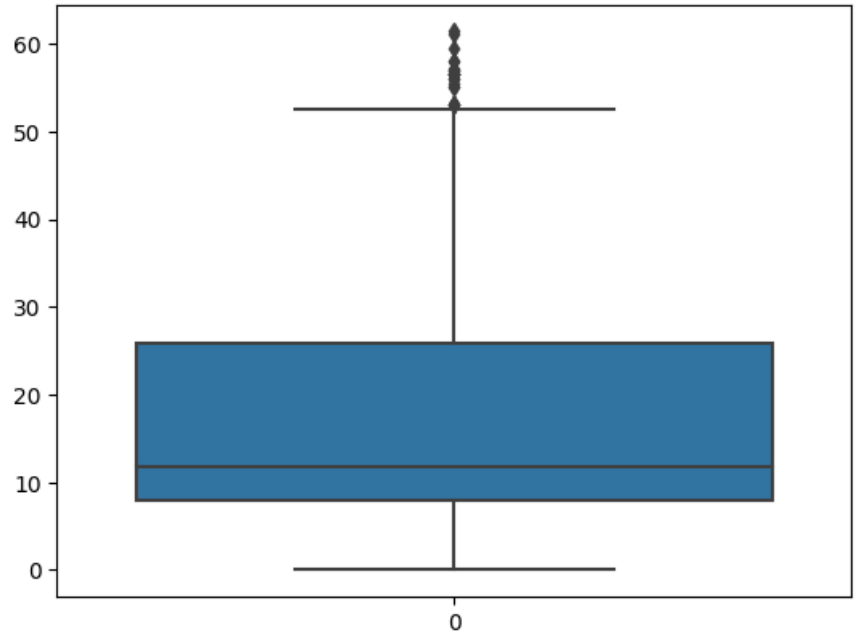
```
df = df[df["Fare"] < upper_limit1]
```

In [113]:

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

Out[113]:

```
<Axes: >
```



Removing the Outlier using the Z-score

In [114]:

```
df1=data.copy()
```

In [115]:

```
fare_zscore=stats.zscore(df1.Fare)
```

In [116]:

```
fare_zscore
```

Out[116]:

```
0     -0.484194
1      0.798324
2     -0.470674
3      0.434132
4     -0.468171
         ...
886   -0.369028
887   -0.028536
888   -0.159725
889   -0.028536
890   -0.474179
Name: Fare, Length: 824, dtype: float64
```

In [117]:

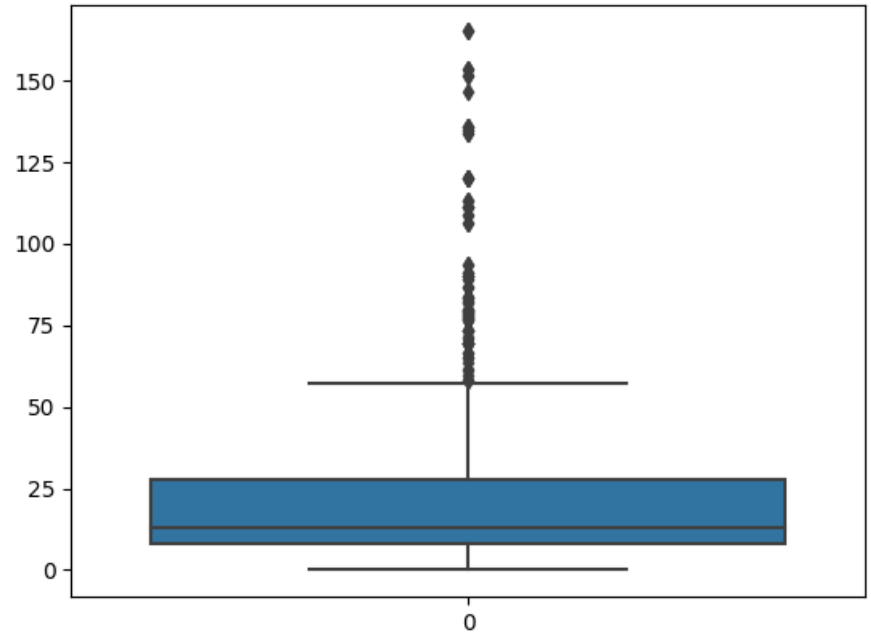
```
df1=df1[np.abs(fare_zscore)<=3]
```

In [118]:

```
sns.boxplot(df1.Fare)
```

Out[118]:

```
<Axes: >
```



Removing the Outlier by using the Percentile Method

In [119]:

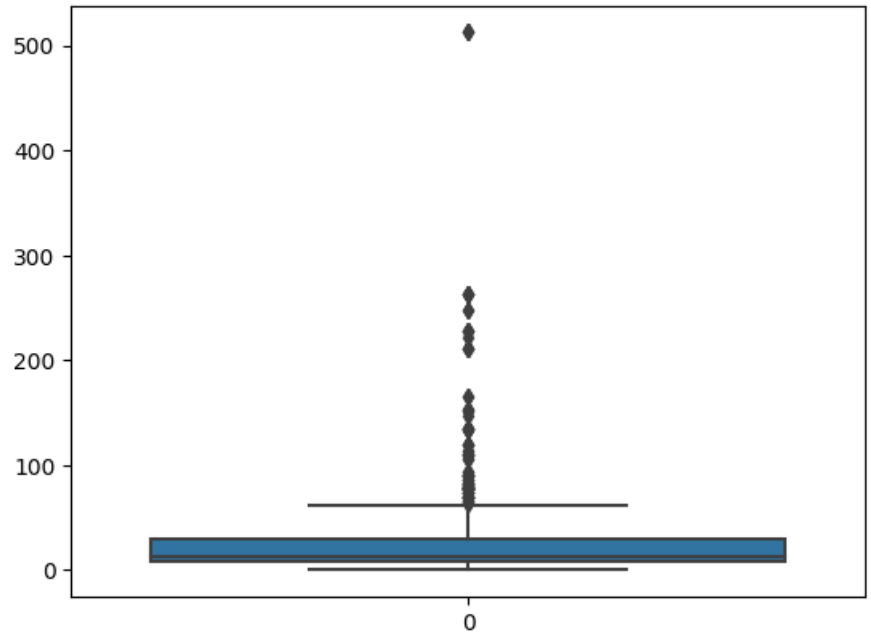
```
df2=data.copy()
```

In [120]:

```
sns.boxplot(df2.Fare)
```

Out[120]:

```
<Axes: >
```



In [121]:

```
p99 = df2.Fare.quantile(0.99)
p99
```

Out[121]:

```
247.5208
```

In [122]:

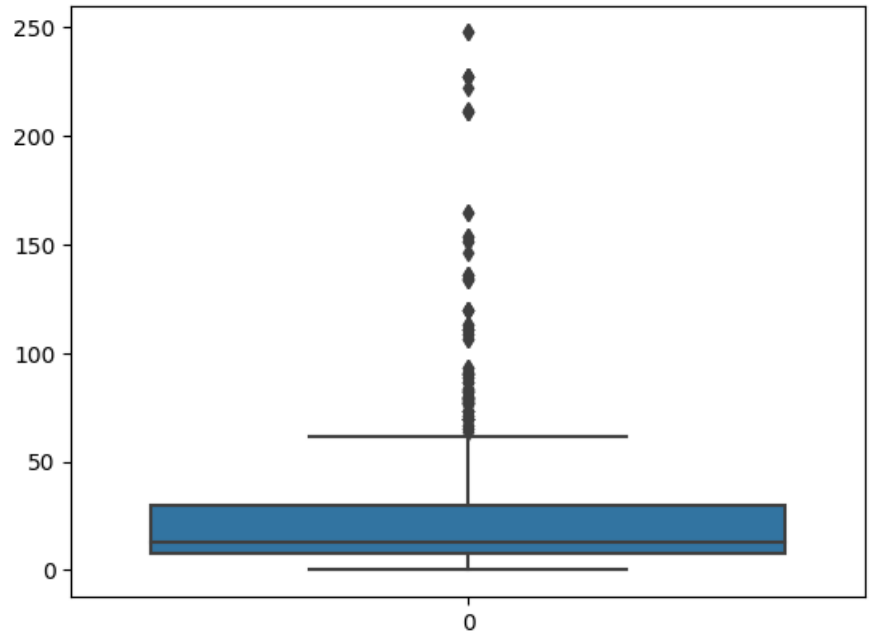
```
df2 = df2[df2.Fare<=p99]
```

In [123]:

```
sns.boxplot(df2.Fare)
```

Out[123]:

<Axes: >



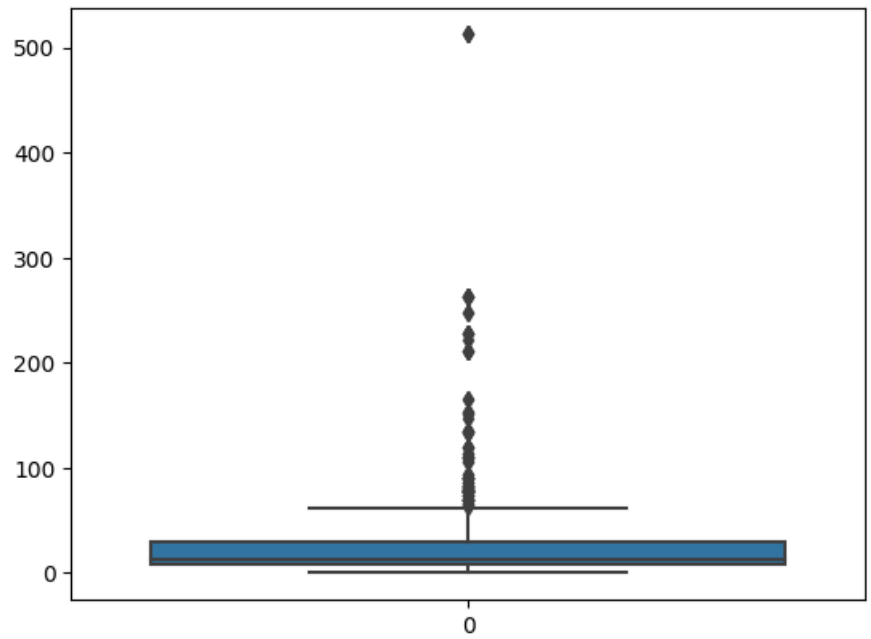
Among all this method IQR method is better

In [124]:

```
sns.boxplot(data.Fare)
```

Out[124]:

<Axes: >



In [125]:

```
q1 = data.Fare.quantile(0.25)
q3 = data.Fare.quantile(0.75)
```

In [126]:

```
print(q1)
print(q3)
```

```
7.8958
30.017699999999998
```

In [127]:

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

```
22.121899999999997
```

In [128]:

```
upper_limit1 = q3+1.5*IQR
print(upper_limit)
```

```
63.20054999999999
```

In [129]:

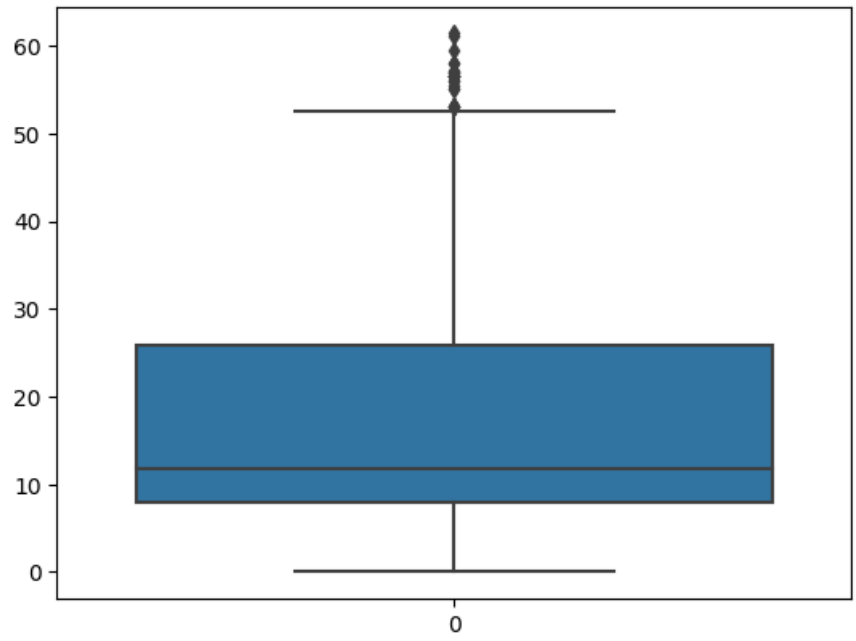
```
data = data[data["Fare"] < upper_limit]
```

In [130]:

```
sns.boxplot(data.Fare)
```

Out[130]:

```
<Axes: >
```



6. Separate dependent and independent variables

In [131]:

```
data.head()
```

Out[131]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S
5	6	0	3	Moran, Mr. James	male	28.0	0	0	330877	8.4583	Q

In [132]:

```
x = data[["Pclass", "Sex", "Age", "SibSp", "Parch", "Fare", "Embarked"]]
```

In [133]:

```
x
```

Out[133]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	22.0	1	0	7.2500	S
2	3	female	26.0	0	0	7.9250	S
3	1	female	35.0	1	0	53.1000	S
4	3	male	35.0	0	0	8.0500	S
5	3	male	28.0	0	0	8.4583	Q
...	...	...	...	...	...	...	...
886	2	male	27.0	0	0	13.0000	S
887	1	female	19.0	0	0	30.0000	S
888	3	female	28.0	1	0	23.4500	S
889	1	male	26.0	0	0	30.0000	C
890	3	male	32.0	0	0	7.7500	Q

718 rows × 7 columns

In [134]:

```
x.shape
```

Out[134]:

```
(718, 7)
```

In [135]:

```
type(x)
```

Out[135]:

```
pandas.core.frame.DataFrame
```

In [136]:

```
y=data["Survived"]
```

In [137]:

```
y
```

Out[137]:

```
0      0
2      1
3      1
4      0
5      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 718, dtype: int64
```

In [138]:

```
type(y)
```

Out[138]:

```
pandas.core.series.Series
```

In [139]:

```
y.shape
```

Out[139]:

```
(718,)
```

7. Encoding

Label Encoder on Gender column

In [140]:

```
le=LabelEncoder()
```

In [141]:

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

```
C:\Users\MOHID BABU SHAIK\AppData\Local\Temp\ipykernel_30528\566333558.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
ng.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/inde
xing.html#returning-a-view-versus-a-copy)
  x["Sex"]=le.fit_transform(x["Sex"])
```

In [142]:

```
x["Sex"]
```

Out[142]:

```
0      1
2      0
3      0
4      1
5      1
      ..
886    1
887    0
888    0
889    1
890    1
Name: Sex, Length: 718, dtype: int32
```

In [143]:

```
x
```

Out[143]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	22.0	1	0	7.2500	S
2	3	0	26.0	0	0	7.9250	S
3	1	0	35.0	1	0	53.1000	S
4	3	1	35.0	0	0	8.0500	S
5	3	1	28.0	0	0	8.4583	Q
...	...	...	...	...	...	...	...
886	2	1	27.0	0	0	13.0000	S
887	1	0	19.0	0	0	30.0000	S
888	3	0	28.0	1	0	23.4500	S
889	1	1	26.0	0	0	30.0000	C
890	3	1	32.0	0	0	7.7500	Q

718 rows × 7 columns

One Hot encoding on Embarked column

In [144]:

```
data["Embarked"].unique()
```

Out[144]:

```
array(['S', 'Q', 'C'], dtype=object)
```

In [145]:

```
Embarked = pd.get_dummies(x["Embarked"], drop_first=True).astype(int)
```

In [146]:

```
Embarked
```

Out[146]:

	Q	S
0	0	1
2	0	1
3	0	1
4	0	1
5	1	0
...	...	...
886	0	1
887	0	1
888	0	1
889	0	0
890	1	0

718 rows × 2 columns

In [147]:

```
x=pd.concat([x,Embarked],axis=1)
```

In [148]:

```
x
```

Out[148]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Q	S
0	3	1	22.0	1	0	7.2500	S	0	1
2	3	0	26.0	0	0	7.9250	S	0	1
3	1	0	35.0	1	0	53.1000	S	0	1
4	3	1	35.0	0	0	8.0500	S	0	1
5	3	1	28.0	0	0	8.4583	Q	1	0
...	...	...	...	...	...	...	...	...	...
886	2	1	27.0	0	0	13.0000	S	0	1
887	1	0	19.0	0	0	30.0000	S	0	1
888	3	0	28.0	1	0	23.4500	S	0	1
889	1	1	26.0	0	0	30.0000	C	0	0
890	3	1	32.0	0	0	7.7500	Q	1	0

718 rows × 9 columns

In [149]:

```
x.drop(["Embarked"],axis=1,inplace=True)
```

In [150]:

```
x.head()
```

Out[150]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Q	S
0	3	1	22.0	1	0	7.2500	0	1
2	3	0	26.0	0	0	7.9250	0	1
3	1	0	35.0	1	0	53.1000	0	1
4	3	1	35.0	0	0	8.0500	0	1
5	3	1	28.0	0	0	8.4583	1	0

8. Spliting into training and testing set

In [151]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=0)
```

In [152]:

```
x_train,x_test,y_train,y_test
```

Out[152]:

```
(     Pclass  Sex   Age  SibSp  Parch     Fare  Q  S
 596       2    0  28.0      0      0  33.0000  0  1
 194       1    0  44.0      0      0  27.7208  0  0
 441       3    1  20.0      0      0   9.5000  0  1
 95        3    1  28.0      0      0   8.0500  0  1
 112       3    1  22.0      0      0   8.0500  0  1
 ..      ...  ...   ...    ...    ...      ... .. ..
 880       2    0  25.0      0      0  26.0000  0  1
 235       3    0  28.0      0      0   7.5500  0  1
 785       3    1  25.0      0      0   7.2500  0  1
 701       1    1  35.0      0      0  26.2875  0  1
 854       2    0  44.0      1      0  26.0000  0  1

 [502 rows x 8 columns],
      Pclass  Sex   Age  SibSp  Parch     Fare  Q  S
 623       3    1  21.0      0      0   7.8542  0  1
 805       3    1  31.0      0      0   7.7750  0  1
 302       3    1  19.0      0      0   0.0000  0  1
 175       3    1  18.0      1      0   7.8542  0  1
 612       3    0  28.0      1      0  15.5000  1  0
 ..      ...  ...   ...    ...    ...      ... .. ..
 17        2    1  28.0      0      0  13.0000  0  1
 197       3    1  42.0      0      0   8.4042  0  1
 610       3    0  39.0      1      0  31.2750  0  1
 335       3    1  28.0      0      0   7.8958  0  1
 577       1    0  39.0      1      0  55.9000  0  1

 [216 rows x 8 columns],
 596    1
 194    1
 441    0
 95     0
 112    0
       ..
 880    1
 235    0
 785    0
 701    1
 854    0
 Name: Survived, Length: 502, dtype: int64,
 623    0
 805    0
 302    0
 175    0
 612    1
       ..
 17     1
 197    0
 610    0
 335    0
 577    1
 Name: Survived, Length: 216, dtype: int64)
```

In [153]:

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

Out[153]:

```
((502, 8), (216, 8), (502,), (216,))
```

9. Feature Scaling

In [154]:

```
ms=MinMaxScaler()
```

In [155]:

```
x_train_scaled = ms.fit_transform(x_train)
x_test_scaled = ms.fit_transform(x_test)
```

In [156]:

```
x_train_scaled
```

Out[156]:

```
array([[0.5       , 0.        , 0.49019608, ..., 0.55555556, 0.        ,
        1.        ],
       [0.        , 0.        , 0.80392157, ..., 0.46668013, 0.        ,
        0.        ],
       [1.        , 1.        , 0.33333333, ..., 0.15993266, 0.        ,
        1.        ],
       ...,
       [1.        , 1.        , 0.43137255, ..., 0.12205387, 0.        ,
        1.        ],
       [0.        , 1.        , 0.62745098, ..., 0.44255051, 0.        ,
        1.        ],
       [0.5       , 0.        , 0.80392157, ..., 0.43771044, 0.        ,
        1.        ]])
```

In [157]:

```
x_test_scaled
```

Out[157]:

```
array([[1.        , 1.        , 0.35294118, ..., 0.12796192, 0.        ,
        1.        ],
       [1.        , 1.        , 0.54901961, ..., 0.12667158, 0.        ,
        1.        ],
       [1.        , 1.        , 0.31372549, ..., 0.        , 0.        ,
        1.        ],
       ...,
       [1.        , 0.        , 0.70588235, ..., 0.50953743, 0.        ,
        1.        ],
       [1.        , 1.        , 0.49019608, ..., 0.12863967, 0.        ,
        1.        ],
       [0.        , 0.        , 0.70588235, ..., 0.91073197, 0.        ,
        1.        ]])
```