In [115]:
```
                                        #DUDDUKURI SAIKIRAN
                                        #21BCE9166
                                        #ASSIGNMENT-NO : 3
#Perform the Data preprocessing on Titanic Dataset
```

1. Import the necessary libraries :

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Import the dataset :

In [4]:
```
df = pd.read_csv("Titanic.csv")
```

In [5]:
```
print(df)
```

```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex   Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
..                                                 ...     ...   ...    ...
886                              Montvila, Rev. Juozas    male  27.0      0
887                       Graham, Miss. Margaret Edith  female  19.0      0
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                              Behr, Mr. Karl Howell    male  26.0      0
890                                Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     ...               ...      ...   ...      ...
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

In [10]:
```
#no.of rows and columns
df.shape
```

Out[10]:
```
(891, 12)
```

In [9]:
```
#getting mean,median,percentiles count,std,min and of dataset using descibe() function
df.describe()
```

Out[9]:

	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

In [6]:
```
#getting first five rows
df.head()
```

Out[6]:

	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 [7]:
```
#getting last five rows
df.tail()
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
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 [8]:
```
#info of the datset
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
```

3. Checking for Null values :

In [11]:
```
df.isnull().any()
```

Out[11]:
```
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 [11]:
```
#here there are null values in the column cabin,Age ans Embarked so we need to handle th
```

In [12]:
```
#lets check the sum of null values
```

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
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 [15]:
```
#there are 177 null values in Age column
#here are 687 null values in Cabin column
#here are 2 null values in Embarked column
```

In [16]:
```
#lets remove the null values in Age column
#Age column is having only numerical data so we can replace null values of age column wi
```

In [13]:
```
mean1 = df["Age"].mean()
```

In [14]:
```
mean1
```

Out[14]:
```
29.69911764705882
```

In [15]:
```
df["Age"]=df["Age"].fillna(mean1)
```

In [16]:
```
df["Age"]
```

Out[16]:
```
0      22.000000
1      38.000000
2      26.000000
3      35.000000
4      35.000000
         ...
886    27.000000
887    19.000000
888    29.699118
889    26.000000
890    32.000000
Name: Age, Length: 891, dtype: float64
```

In [20]:
```
#we have replaced the null values of age column  with mean of age column
```

In [21]:
```
#lets remove the null values in Cabin column
#Cabin column is having only categorical data so we can replace null values of Cabin col
```

In [17]:
```
mode1 = df["Cabin"].mode()
```

In [18]:
```
mode1
```

Out[18]:
```
0        B96 B98
1    C23 C25 C27
2             G6
dtype: object
```

In [19]:
```
df["Cabin"] = df["Cabin"].fillna(mode1[2])
```

In [20]:
```
df["Cabin"]
```

Out[20]:
```
0        G6
1       C85
2        G6
3      C123
4        G6
       ...
886      G6
887     B42
888      G6
889    C148
890      G6
Name: Cabin, Length: 891, dtype: object
```

In [21]:
```
#lets remove the null values in Embarked column
#Embarked column is having only categorical data so we can replace null values of Embark
```

In [22]:
```
mode2 = df["Embarked"].mode()
```

In [23]:
```
mode2
```

Out[23]:
```
0    S
dtype: object
```

In [24]:
```
df["Embarked"] = df["Embarked"].fillna(mode2[0])
```

In [25]:
```
df["Embarked"]
```

Out[25]:
```
0      S
1      C
2      S
3      S
4      S
      ..
886    S
887    S
888    S
889    C
890    Q
Name: Embarked, Length: 891, dtype: object
```

In [26]:
```
#Now lets check if there are null values are not
```

In [27]:
```
df.isnull().any()
```

Out[27]:
```
PassengerId    False
Survived       False
Pclass         False
Name           False
Sex            False
Age            False
SibSp          False
Parch          False
Ticket         False
Fare           False
Cabin          False
Embarked       False
dtype: bool
```

In [28]:
```
df.isnull().sum()
```

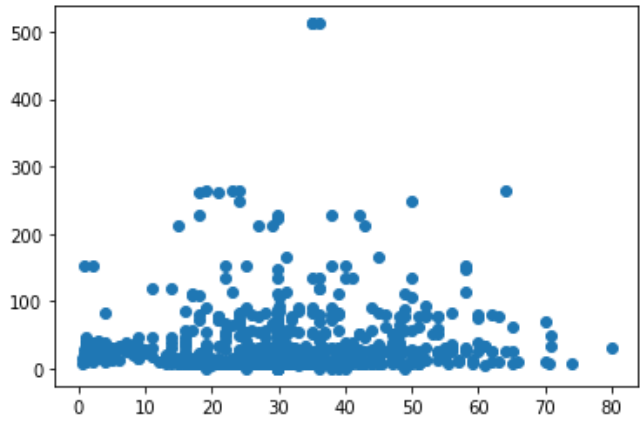
Out[28]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

In [29]:
```
#so we have removed the null values
```

4. Data Visualization :

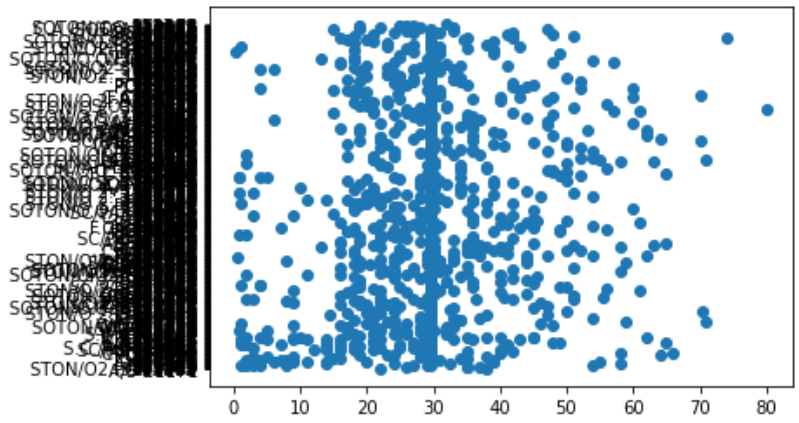
In [30]:
```
plt.scatter(x=df["Age"],y=df["Fare"])
```

Out[30]:
```
<matplotlib.collections.PathCollection at 0x1264a64f0>
```



In [31]:
```
plt.scatter(x=df["Age"],y=df["Ticket"])
```

Out[31]:
```
<matplotlib.collections.PathCollection at 0x1265d5a30>
```

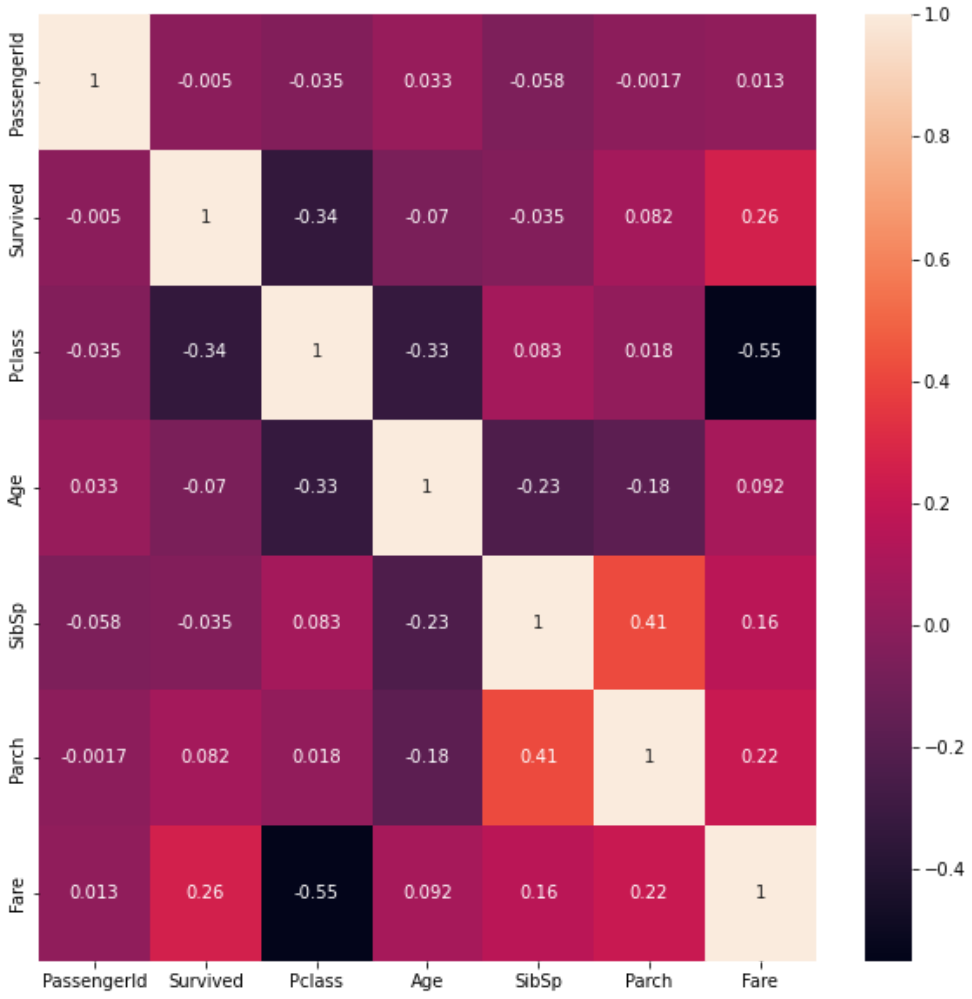


In [32]:
```
#lets us observe the correlation by drawing heat map
```

In [33]:
```
cor = df.corr()
```

In [34]:
```
plt.figure(figsize=(10,10))
sns.heatmap(cor,annot=True)
```

Out[34]: <AxesSubplot:>



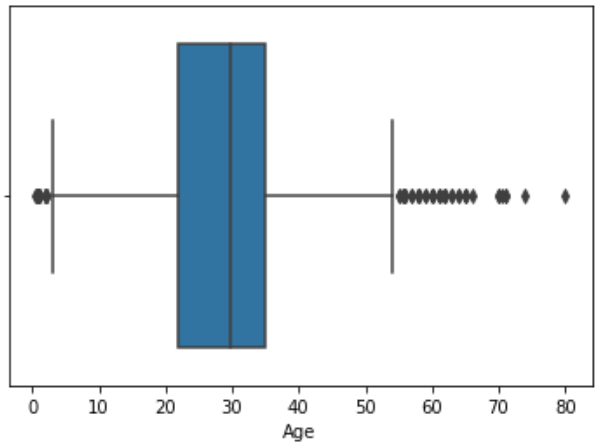
5. Outlier Detection :

In [35]:
```
#in order to know whether there are outliers or not we have to draw the box plot
#lets us draw the boxplot
```

In [36]:
```
sns.boxplot(df["Age"])
```

```
/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid posi
tional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[36]: <AxesSubplot:xlabel='Age'>



In [37]:
```
#we have outliers so lets remove the outliers
```

In [38]:
```
A_q1 = df.Age.quantile(0.25)
A_q3 = df.Age.quantile(0.75)
```

In [39]:
```
IQR = A_q3-A_q1
```

In [40]:
```
IQR
```

Out[40]: 13.0

In [41]:
```
upper_limit = A_q3+1.5*IQR
```

In [42]:
```
upper_limit
```

Out[42]: 54.5

In [43]:
```
med = df.Age.median()
```

In [44]:
```
med
```

Out[44]: 29.69911764705882

In [45]:
```
df["Age"] = np.where(df["Age"]>upper_limit,med,df["Age"])
```

In [46]:
```
lower_limit = A_q1-1.5*IQR
```

In [47]:
```
df["Age"] = np.where(df["Age"]<lower_limit,med,df["Age"])
```

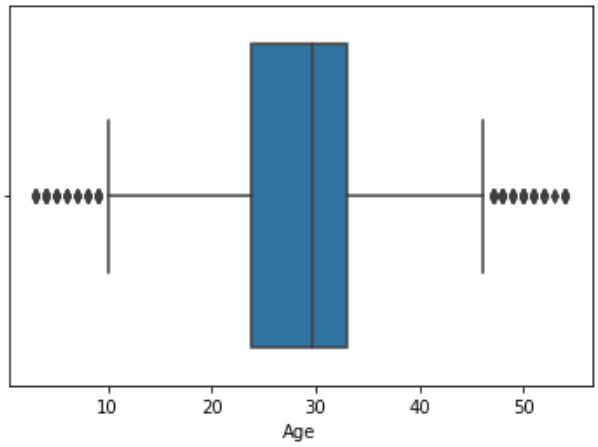
In [ ]:
```

```

In [48]:
```
sns.boxplot(df["Age"])
```

```
/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid posi
tional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[48]: <AxesSubplot:xlabel='Age'>



In [71]:
```
#still we have some more outliers lets again remove them
```

In [49]:
```
A_q1 = df.Age.quantile(0.25)
A_q3 = df.Age.quantile(0.75)
```

In [50]:
```
IQR = A_q3-A_q1
```

In [51]:
```
IQR
```

Out[51]: 9.25

In [52]:
```
upper_limit = A_q3+1.5*IQR
```

In [53]:
```
upper_limit
```

Out[53]: 46.875

In [54]:
```
med = df.Age.median()
```

In [55]:
```
med
```

Out[55]: 29.69911764705882

In [56]:
```
df["Age"] = np.where(df["Age"]>upper_limit,med,df["Age"])
```

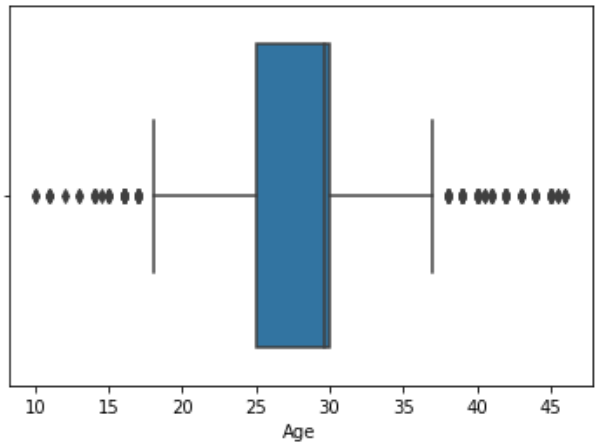
In [57]:
```
lower_limit = A_q1-1.5*IQR
```

In [58]:
```
df["Age"] = np.where(df["Age"]<lower_limit,med,df["Age"])
```

In [59]:
```
sns.boxplot(df["Age"])
```

```
/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid posi
tional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[59]: <AxesSubplot:xlabel='Age'>



In [60]:
```
#Lets use zscore method to remove some more outliers :
```

In [61]:
```
from scipy import stats
```

In [62]:
```
Age_zscore = stats.zscore(df.Age)
```

In [63]:
```
Age_zscore
```

Out[63]:
```
0     -1.000548
1      1.396086
2     -0.401390
3      0.946717
4      0.946717
         ...
886   -0.251600
887   -1.449917
888    0.152700
889   -0.401390
890    0.497348
Name: Age, Length: 891, dtype: float64
```

In [64]:
```
df_z=df[np.abs(Age_zscore)<=3]
```

In [65]:
```
df_z
```

Out[65]:

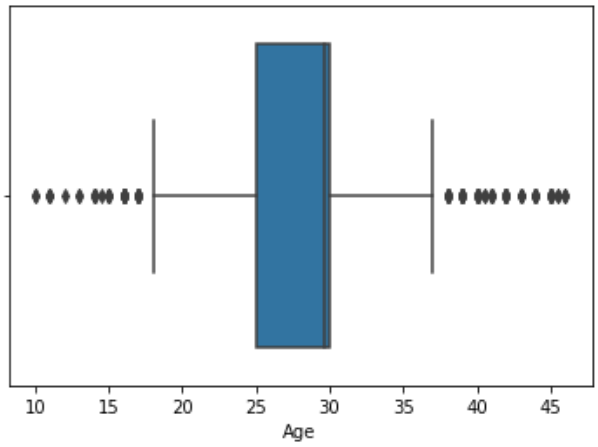
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	G6	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	G6	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450	8.0500	G6	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	211536	13.0000	G6	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./C. 6607	23.4500	G6	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	370376	7.7500	G6	Q

891 rows × 12 columns

In [66]:
```
sns.boxplot(df.Age)
```

```
/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid posi
tional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[66]: <AxesSubplot:xlabel='Age'>



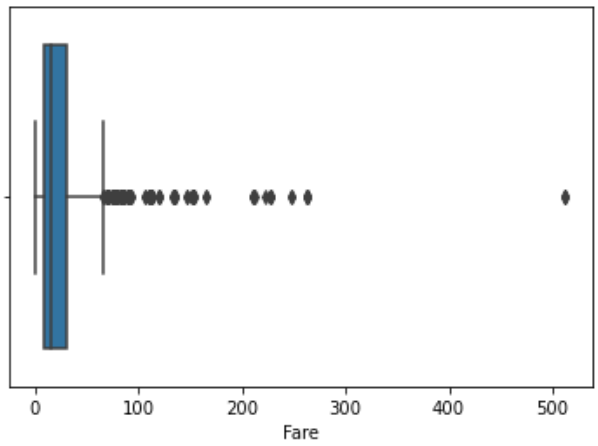
In [67]:
```
#we are able to remove the outliers only for some extend
#again and again some outliers are coming
#we can remove the columns but majority of the columns will eliminate from dataset which
#so let them as it is.
#for this dataset we are able to remove the outliers for some extend only
```

In [68]:
```
#Lets try for Fare columns :
```

In [69]:
```
sns.boxplot(df.Fare)
```

```
/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid posi
tional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[69]: <AxesSubplot:xlabel='Fare'>



In [70]:
```
#Here there are only outliers only at upperlimit so lets remove them
```

In [71]:
```
F_q1 = df.Fare.quantile(0.25)
F_q3 = df.Fare.quantile(0.75)
```

In [72]:
```
IQR = F_q3-F_q1
```

In [73]:
```
IQR
```

Out[73]:
```
23.0896
```

In [74]:
```
u_l = F_q3+1.5*IQR
```

In [75]:
```
l_l = F_q1-1.5*IQR
```

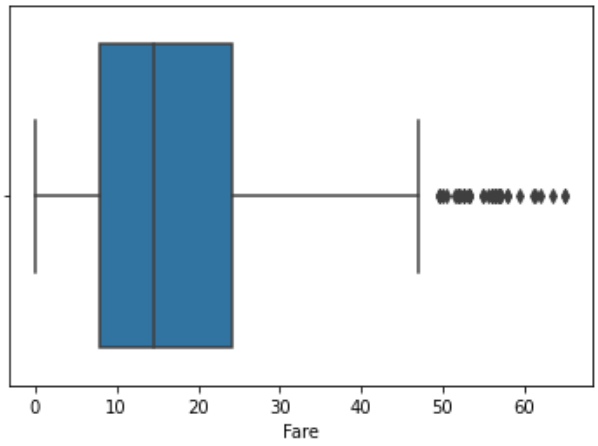
In [76]:
```
med = df["Fare"].median()
```

In [77]:
```
df["Fare"] = np.where(df["Fare"]>u_l,med,df["Fare"])
```

In [78]:
```
sns.boxplot(df["Fare"])
```

```
/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid posi
tional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[78]:
```
<AxesSubplot:xlabel='Fare'>
```



In [79]:
```
#here also only some outliers got removed
```

In [80]:
```
#lets try with zscore also
```

In [81]:
```
Fare_zscore = stats.zscore(df.Fare)
```

In [82]:
```
Fare_zscore
```

Out[82]:
```
0      -0.797554
1      -0.230556
2      -0.744429
3       2.811012
4      -0.734591
          ...
886    -0.345007
887     0.992956
888     0.477446
889     0.992956
890    -0.758202
Name: Fare, Length: 891, dtype: float64
```

In [83]:
```
df_z=df[np.abs(Fare_zscore)<=3]
```

In [84]:
```
df_z
```

Out[84]:

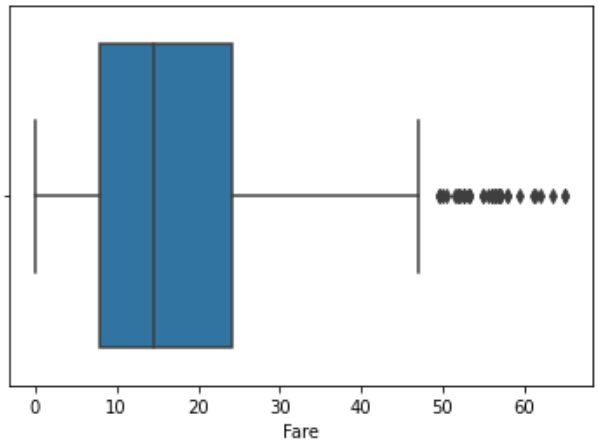
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	G6	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	14.4542	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	G6	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450	8.0500	G6	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	211536	13.0000	G6	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./C. 6607	23.4500	G6	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	370376	7.7500	G6	Q

869 rows × 12 columns

In [85]:
```
sns.boxplot(df.Fare)
```

```
/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid posi
tional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[85]: <AxesSubplot:xlabel='Fare'>



In [86]:
```
#by using zscore also we are able to remove the outliers upto some extend
```

6. Splitting Dependent and Independent variables :

In [87]:
```
#Before splittng lets us the remove the columns which are not necessary for further step
```

In [88]:
```
df.drop(['Name'],axis=1,inplace=True)
```

In [89]:
```
df
```

Out[89]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	male	22.000000	1	0	A/5 21171	7.2500	G6	S
1	2	1	1	female	38.000000	1	0	PC 17599	14.4542	C85	C
2	3	1	3	female	26.000000	0	0	STON/O2. 3101282	7.9250	G6	S
3	4	1	1	female	35.000000	1	0	113803	53.1000	C123	S
4	5	0	3	male	35.000000	0	0	373450	8.0500	G6	S
...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	male	27.000000	0	0	211536	13.0000	G6	S
887	888	1	1	female	19.000000	0	0	112053	30.0000	B42	S
888	889	0	3	female	29.699118	1	2	W./C. 6607	23.4500	G6	S
889	890	1	1	male	26.000000	0	0	111369	30.0000	C148	C
890	891	0	3	male	32.000000	0	0	370376	7.7500	G6	Q

891 rows × 11 columns

In [90]:
```
df.drop(['Ticket'],axis=1,inplace=True)
```

In [91]:
```
df
```

Out[91]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	1	0	3	male	22.000000	1	0	7.2500	G6	S
1	2	1	1	female	38.000000	1	0	14.4542	C85	C
2	3	1	3	female	26.000000	0	0	7.9250	G6	S
3	4	1	1	female	35.000000	1	0	53.1000	C123	S
4	5	0	3	male	35.000000	0	0	8.0500	G6	S
...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	male	27.000000	0	0	13.0000	G6	S
887	888	1	1	female	19.000000	0	0	30.0000	B42	S
888	889	0	3	female	29.699118	1	2	23.4500	G6	S
889	890	1	1	male	26.000000	0	0	30.0000	C148	C
890	891	0	3	male	32.000000	0	0	7.7500	G6	Q

891 rows × 10 columns

In [92]:
```
df.drop(["PassengerId"],axis=1,inplace=True)
```

In [93]:
```
df
```

Out[93]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	0	3	male	22.000000	1	0	7.2500	G6	S
1	1	1	female	38.000000	1	0	14.4542	C85	C
2	1	3	female	26.000000	0	0	7.9250	G6	S
3	1	1	female	35.000000	1	0	53.1000	C123	S
4	0	3	male	35.000000	0	0	8.0500	G6	S
...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.000000	0	0	13.0000	G6	S
887	1	1	female	19.000000	0	0	30.0000	B42	S
888	0	3	female	29.699118	1	2	23.4500	G6	S
889	1	1	male	26.000000	0	0	30.0000	C148	C
890	0	3	male	32.000000	0	0	7.7500	G6	Q

891 rows × 9 columns

In [94]:
```
df.drop(["Cabin"],axis=1,inplace=True)
```

In [95]:
```
df
```

Out[95]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.000000	1	0	7.2500	S
1	1	1	female	38.000000	1	0	14.4542	C
2	1	3	female	26.000000	0	0	7.9250	S
3	1	1	female	35.000000	1	0	53.1000	S
4	0	3	male	35.000000	0	0	8.0500	S
...	...	...	...	...	...	...	...	...
886	0	2	male	27.000000	0	0	13.0000	S
887	1	1	female	19.000000	0	0	30.0000	S
888	0	3	female	29.699118	1	2	23.4500	S
889	1	1	male	26.000000	0	0	30.0000	C
890	0	3	male	32.000000	0	0	7.7500	Q

891 rows × 8 columns

In [96]:
```
#Now lets split the data into dependent and independent variables
#x determines the independent variable
#y determines the dependent variable
```

In [97]:
```
#dependent variable
y=df["Survived"]
```

In [98]:
```
y
```

Out[98]:
```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

In [99]:
```

```

In [103]:
```
#independent variable
x=df.drop("Survived",axis=1)
```

In [104]:
```
x
```

Out[104]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	22.000000	1	0	7.2500	S
1	1	female	38.000000	1	0	14.4542	C
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
...	...	...	...	...	...	...	...
886	2	male	27.000000	0	0	13.0000	S
887	1	female	19.000000	0	0	30.0000	S
888	3	female	29.699118	1	2	23.4500	S
889	1	male	26.000000	0	0	30.0000	C
890	3	male	32.000000	0	0	7.7500	Q

891 rows × 7 columns

7. Encoding :

In [105]:
```
#let us encode the sex column
```

In [108]:
```
from sklearn.preprocessing import LabelEncoder
```

In [109]:
```
#create the object for label encoder
lr = LabelEncoder()
```

In [110]:
```
lr
```

Out[110]:
```
LabelEncoder()
```

In [112]:
```
x["Sex"] = lr.fit_transform(x["Sex"])
```

In [113]:
```
x["Sex"]
```

Out[113]:
```
0      1
1      0
2      0
3      0
4      1
      ..
886    1
887    0
888    0
889    1
890    1
Name: Sex, Length: 891, dtype: int64
```

In [114]:
```
#we have encoded the sex column
```

In [116]:
```
x.head()
```

Out[116]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	22.0	1	0	7.2500	S
1	1	0	38.0	1	0	14.4542	C
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

In [117]:
```
#lets encode the Embarked column
```

In [118]:
```
x["Embarked"] = lr.fit_transform(x["Embarked"])
```

In [119]:
```
x["Embarked"]
```

Out[119]:
```
0       2
1       0
2       2
3       2
4       2
      ..
886     2
887     2
888     2
889     0
890     1
Name: Embarked, Length: 891, dtype: int64
```

In [120]:
```
x.head()
```

Out[120]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	22.0	1	0	7.2500	2
1	1	0	38.0	1	0	14.4542	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

In [121]:
```
#now we have only numerical data
```

In [122]:
```
x["Sex"].nunique()
```

Out[122]: 2

In [123]:
```
x["Embarked"].nunique()
```

Out[123]: 3

In [124]:
```
y.head()
```

Out[124]:
```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

8. Splitting the train and test data :

In [127]:
```
from sklearn.model_selection import train_test_split
```

In [128]:
```
#80 percent for training and 20 percent for testing
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

In [129]:
```
x_train
```

Out[129]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
140	3	0	29.699118	0	2	15.2458	0
439	2	1	31.000000	0	0	10.5000	2
817	2	1	31.000000	1	1	37.0042	0
378	3	1	20.000000	0	0	4.0125	0
491	3	1	21.000000	0	0	7.2500	2
...	...	...	...	...	...	...	...
835	1	0	39.000000	1	1	14.4542	0
192	3	0	19.000000	1	0	7.8542	2
629	3	1	29.699118	0	0	7.7333	1
559	3	0	36.000000	1	0	17.4000	2
684	2	1	29.699118	1	1	39.0000	2

712 rows × 7 columns

In [130]:
```
y_train
```

Out[130]:
```
140    0
439    0
817    0
378    0
491    0
      ..
835    1
192    1
629    0
559    1
684    0
Name: Survived, Length: 712, dtype: int64
```

In [131]:
```
x_test
```

Out[131]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
495	3	1	29.699118	0	0	14.4583	0
648	3	1	29.699118	0	0	7.5500	2
278	3	1	29.699118	4	1	29.1250	1
31	1	0	29.699118	1	0	14.4542	0
255	3	0	29.000000	0	2	15.2458	0
...	...	...	...	...	...	...	...
780	3	0	13.000000	0	0	7.2292	0
837	3	1	29.699118	0	0	8.0500	2
215	1	0	31.000000	1	0	14.4542	0
833	3	1	23.000000	0	0	7.8542	2
372	3	1	19.000000	0	0	8.0500	2

179 rows × 7 columns

In [132]:
```
y_test
```

Out[132]:
```
495    0
648    0
278    0
31     1
255    1
      ..
780    1
837    0
215    1
833    0
372    0
Name: Survived, Length: 179, dtype: int64
```

In [133]:
```
x_train.shape
```

Out[133]: (712, 7)

In [134]:
```
x_test.shape
```

Out[134]: (179, 7)

In [135]:
```
y_train.shape
```

Out[135]: (712,)

In [136]:
```
y_test.shape
```

Out[136]: (179,)

9. Feature Scaling :

In [137]:
```
from sklearn.preprocessing import StandardScaler
```

In [138]:
```
#create object for standard scaler
sc = StandardScaler()
```

In [139]:
```
sc
```

Out[139]:
```
StandardScaler()
```

In [140]:
```
x_train = sc.fit_transform(x_train)
```

In [141]:
```
x_train
```

Out[141]:
```
array([[ 0.81925059, -1.37207547,  0.13485787, ...,  1.95926403,
        -0.17726299, -1.98156574],
       [-0.38096838,  0.72882288,  0.33193428, ..., -0.47741019,
        -0.54667438,  0.5790056 ],
       [-0.38096838,  0.72882288,  0.33193428, ...,  0.74092692,
         1.51640316, -1.98156574],
       ...,
       [ 0.81925059,  0.72882288,  0.13485787, ..., -0.47741019,
        -0.76203333, -0.70128007],
       [ 0.81925059, -1.37207547,  1.08940633, ..., -0.47741019,
        -0.00958083,  0.5790056 ],
       [-0.38096838,  0.72882288,  0.13485787, ...,  0.74092692,
         1.67175552,  0.5790056 ]])
```

In [142]:
```
x_test = sc.fit_transform(x_test)
```

In [143]:
```
x_test
```

Out[143]:
```
array([[ 0.86022947,  0.77344314,  0.22087115, ..., -0.46006628,
        -0.19571051, -1.80134224],
       [ 0.86022947,  0.77344314,  0.22087115, ..., -0.46006628,
        -0.76604362,  0.61394061],
       [ 0.86022947,  0.77344314,  0.22087115, ...,  0.88996427,
         1.01513799, -0.59370081],
       ...,
       [-1.50871015, -1.29291987,  0.40823562, ..., -0.46006628,
        -0.19604899, -1.80134224],
       [ 0.86022947,  0.77344314, -0.74399444, ..., -0.46006628,
        -0.74092958,  0.61394061],
       [ 0.86022947,  0.77344314, -1.32010947, ..., -0.46006628,
        -0.72476479,  0.61394061]])
```

In [ ]: