In [58]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")# ignoring warnings
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:

In [59]:
```
data=pd.read_csv("/home/placement/Desktop/nio/Titanic.csv")# reading the file
```

In [60]:
```
data.describe()# describing the data
```

Out[60]:

	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 [61]:
```
data
```

Out[61]:

	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 [62]:
```
data.head(10) #printing the starting 10 values
```

Out[62]:

	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
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

In [63]:
```
data["Pclass"].unique()# printing unique values in Pcalss
```

Out[63]:
```
array([3, 1, 2])
```

In [64]:
```
data["Survived"].unique()# printing unique values in Survived
```

Out[64]:
```
array([0, 1])
```

In [65]:
```
data["Parch"].unique()# printing unique values in Pcalss Survived
```

Out[65]:
```
array([0, 1, 2, 5, 3, 4, 6])
```

In [66]:
```
data["Cabin"].unique()# printing unique values in Cabin
```

Out[66]:
```
array([nan, 'C85', 'C123', 'E46', 'G6', 'C103', 'D56', 'A6',
       'C23 C25 C27', 'B78', 'D33', 'B30', 'C52', 'B28', 'C83', 'F33',
       'F G73', 'E31', 'A5', 'D10 D12', 'D26', 'C110', 'B58 B60', 'E101',
       'F E69', 'D47', 'B86', 'F2', 'C2', 'E33', 'B19', 'A7', 'C49', 'F4',
       'A32', 'B4', 'B80', 'A31', 'D36', 'D15', 'C93', 'C78', 'D35',
       'C87', 'B77', 'E67', 'B94', 'C125', 'C99', 'C118', 'D7', 'A19',
       'B49', 'D', 'C22 C26', 'C106', 'C65', 'E36', 'C54',
       'B57 B59 B63 B66', 'C7', 'E34', 'C32', 'B18', 'C124', 'C91', 'E40',
       'T', 'C128', 'D37', 'B35', 'E50', 'C82', 'B96 B98', 'E10', 'E44',
       'A34', 'C104', 'C111', 'C92', 'E38', 'D21', 'E12', 'E63', 'A14',
       'B37', 'C30', 'D20', 'B79', 'E25', 'D46', 'B73', 'C95', 'B38',
       'B39', 'B22', 'C86', 'C70', 'A16', 'C101', 'C68', 'A10', 'E68',
       'B41', 'A20', 'D19', 'D50', 'D9', 'A23', 'B50', 'A26', 'D48',
       'E58', 'C126', 'B71', 'B51 B53 B55', 'D49', 'B5', 'B20', 'F G63',
       'C62 C64', 'E24', 'C90', 'C45', 'E8', 'B101', 'D45', 'C46', 'D30',
       'E121', 'D11', 'E77', 'F38', 'B3', 'D6', 'B82 B84', 'D17', 'A36',
       'B102', 'B69', 'E49', 'C47', 'D28', 'E17', 'A24', 'C50', 'B42',
       'C148'], dtype=object)
```

In [67]:
```
data["SibSp"].unique()# printing unique values in SibSp
```

Out[67]:
```
array([1, 0, 3, 4, 2, 5, 8])
```

In [68]:
```
data["Age"].unique()# printing unique values in age
```

Out[68]:
```
array([22.  , 38.  , 26.  , 35.  ,   nan, 54.  ,  2.  , 27.  , 14.  ,
        4.  , 58.  , 20.  , 39.  , 55.  , 31.  , 34.  , 15.  , 28.  ,
        8.  , 19.  , 40.  , 66.  , 42.  , 21.  , 18.  ,  3.  ,  7.  ,
       49.  , 29.  , 65.  , 28.5 ,  5.  , 11.  , 45.  , 17.  , 32.  ,
       16.  , 25.  ,  0.83, 30.  , 33.  , 23.  , 24.  , 46.  , 59.  ,
       71.  , 37.  , 47.  , 14.5 , 70.5 , 32.5 , 12.  ,  9.  , 36.5 ,
       51.  , 55.5 , 40.5 , 44.  ,  1.  , 61.  , 56.  , 50.  , 36.  ,
       45.5 , 20.5 , 62.  , 41.  , 52.  , 63.  , 23.5 ,  0.92, 43.  ,
       60.  , 10.  , 64.  , 13.  , 48.  ,  0.75, 53.  , 57.  , 80.  ,
       70.  , 24.5 ,  6.  ,  0.67, 30.5 ,  0.42, 34.5 , 74.  ])
```

In [69]:
```
data["Fare"].unique()# printing unique values in Fare
```

Out[69]:
```
array([  7.25  ,  71.2833,   7.925 ,  53.1   ,   8.05  ,   8.4583,
        51.8625,  21.075 ,  11.1333,  30.0708,  16.7   ,  26.55  ,
        31.275 ,   7.8542,  16.    ,  29.125 ,  13.    ,  18.    ,
         7.225 ,  26.    ,   8.0292,  35.5   ,  31.3875, 263.    ,
         7.8792,   7.8958,  27.7208, 146.5208,   7.75  ,  10.5   ,
        82.1708,  52.    ,   7.2292,  11.2417,   9.475 ,  21.    ,
        41.5792,  15.5   ,  21.6792,  17.8   ,  39.6875,   7.8   ,
        76.7292,  61.9792,  27.75  ,  46.9   ,  80.    ,  83.475 ,
        27.9   ,  15.2458,   8.1583,   8.6625,  73.5   ,  14.4542,
        56.4958,   7.65  ,  29.    ,  12.475 ,   9.    ,   9.5   ,
         7.7875,  47.1   ,  15.85  ,  34.375 ,  61.175 ,  20.575 ,
        34.6542,  63.3583,  23.    ,  77.2875,   8.6542,   7.775 ,
        24.15  ,   9.825 ,  14.4583, 247.5208,   7.1417,  22.3583,
         6.975 ,   7.05  ,  14.5   ,  15.0458,  26.2833,   9.2167,
        79.2   ,   6.75  ,  11.5   ,  36.75  ,   7.7958,  12.525 ,
        66.6   ,   7.3125,  61.3792,   7.7333,  69.55  ,  16.1   ,
        15.75  ,  20.525 ,  55.    ,  25.925 ,  33.5   ,  30.6958,
        25.4667,  28.7125,   0.    ,  15.05  ,  39.    ,  22.025 ,
        50.    ,   8.4042,   6.4958,  10.4625,  18.7875,  31.    ,
       113.275 ,  27.    ,  76.2917,  90.    ,   9.35  ,  13.5   ,
         7.55  ,  26.25  ,  12.275 ,   7.125 ,  52.5542,  20.2125,
        86.5   , 512.3292,  79.65  , 153.4625, 135.6333,  19.5   ,
        29.7   ,  77.9583,  20.25  ,  78.85  ,  91.0792,  12.875 ,
         8.85  , 151.55  ,  30.5   ,  23.25  ,  12.35  , 110.8833,
       108.9   ,  24.    ,  56.9292,  83.1583, 262.375 ,  14.    ,
       164.8667, 134.5   ,   6.2375,  57.9792,  28.5   , 133.65  ,
        15.9   ,   9.225 ,  35.    ,  75.25  ,  69.3   ,  55.4417,
       211.5   ,   4.0125, 227.525 ,  15.7417,   7.7292,  12.    ,
       120.    ,  12.65  ,  18.75  ,   6.8583,  32.5   ,   7.875 ,
        14.4   ,  55.9   ,   8.1125,  81.8583,  19.2583,  19.9667,
        89.1042,  38.5   ,   7.725 ,  13.7917,   9.8375,   7.0458,
         7.5208,  12.2875,   9.5875,  49.5042,  78.2667,  15.1   ,
         7.6292,  22.525 ,  26.2875,  59.4   ,   7.4958,  34.0208,
        93.5   , 221.7792, 106.425 ,  49.5   ,  71.    ,  13.8625,
         7.8292,  39.6   ,  17.4   ,  51.4792,  26.3875,  30.    ,
        40.125 ,   8.7125,  15.    ,  33.    ,  42.4   ,  15.55  ,
        65.    ,  32.3208,   7.0542,   8.4333,  25.5875,   9.8417,
         8.1375,  10.1708, 211.3375,  57.    ,  13.4167,   7.7417,
         9.4833,   7.7375,   8.3625,  23.45  ,  25.9292,   8.6833,
```

```
        8.5167,   7.8875,  37.0042,   6.45  ,   6.95  ,   8.3   ,
        6.4375,  39.4   ,  14.1083,  13.8583,  50.4958,   5.    ,
        9.8458,  10.5167])
```

In [70]:
```
data #describing all the data
```

Out[70]:

	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 [71]:
```
data["Embarked"].unique()# printing unique values in embarked
```

Out[71]:
```
array(['S', 'C', 'Q', nan], dtype=object)
```

In [72]:
```
list(data)# shows the cols present in the data
```

Out[72]:
```
['PassengerId',
 'Survived',
 'Pclass',
 'Name',
 'Sex',
 'Age',
 'SibSp',
 'Parch',
 'Ticket',
 'Fare',
 'Cabin',
 'Embarked']
```

In [73]:
```
data1=data.drop(["PassengerId","Name","Cabin","Ticket","SibSp","Parch"],axis=1)# eliminating the unwanted co
```

In [74]:
```
data1
```

Out[74]:

	Survived	Pclass	Sex	Age	Fare	Embarked
0	0	3	male	22.0	7.2500	S
1	1	1	female	38.0	71.2833	C
2	1	3	female	26.0	7.9250	S
3	1	1	female	35.0	53.1000	S
4	0	3	male	35.0	8.0500	S
...	...	...	...	...	...	...
886	0	2	male	27.0	13.0000	S
887	1	1	female	19.0	30.0000	S
888	0	3	female	NaN	23.4500	S
889	1	1	male	26.0	30.0000	C
890	0	3	male	32.0	7.7500	Q

891 rows × 6 columns

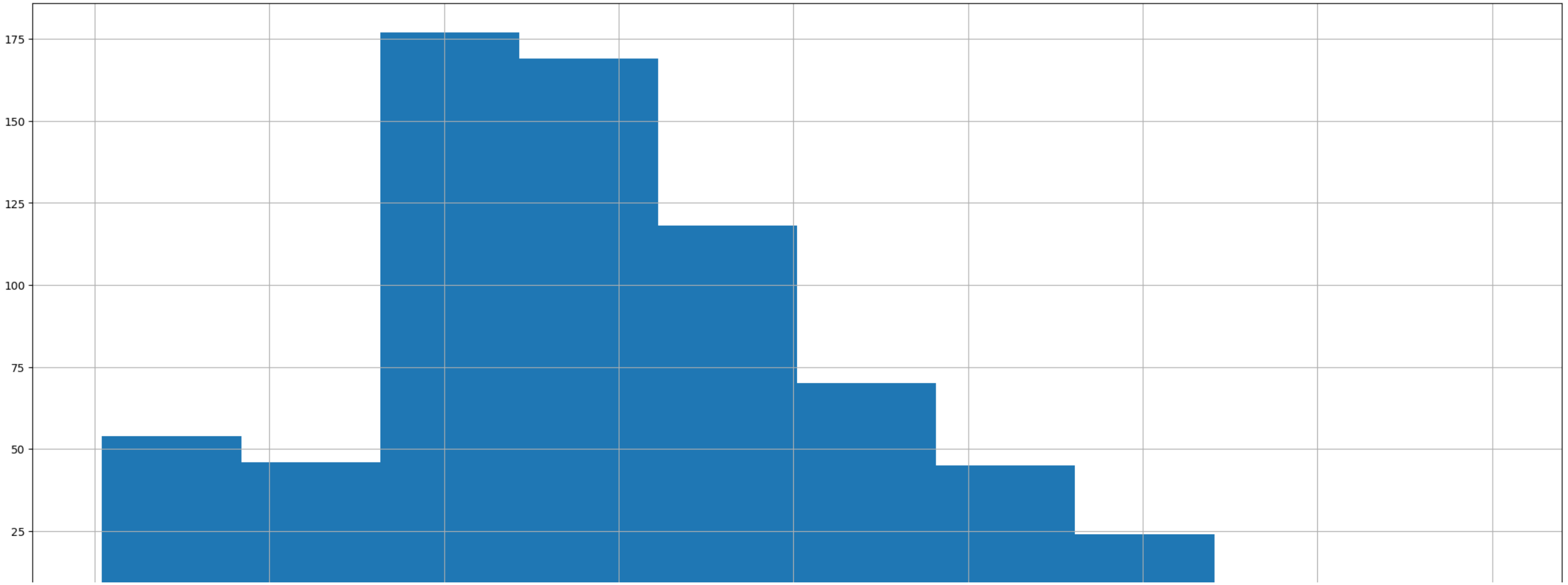
In [75]:
```
data1.isna().sum()# finding the count null values in data
```

Out[75]:
```
Survived      0
Pclass        0
Sex           0
Age         177
Fare          0
Embarked      2
dtype: int64
```

In [76]:
```
data1.shape
```

Out[76]:
```
(891, 6)
```

In [77]:
```
data1['Age'].hist(figsize=(25,10))
plt.show()# ploting the histogram
```



In [78]:
```
dage=data1.groupby(["Age"]).sum()# groupby the age cols
```

In [79]:
```
dage
```

Out[79]:

	Survived	Pclass	Fare
Age			
0.42	1	3	8.5167
0.67	1	2	14.5000
0.75	2	6	38.5166
0.83	2	4	47.7500
0.92	1	1	151.5500
...	...	...	...
70.00	0	3	81.5000
70.50	0	3	7.7500
71.00	0	2	84.1584
74.00	0	3	7.7750
80.00	1	1	30.0000

88 rows × 3 columns

In [80]:
```
data1=data1.fillna(data1.median())
data1 # filling the null values with median
```

Out[80]:

	Survived	Pclass	Sex	Age	Fare	Embarked
0	0	3	male	22.0	7.2500	S
1	1	1	female	38.0	71.2833	C
2	1	3	female	26.0	7.9250	S
3	1	1	female	35.0	53.1000	S
4	0	3	male	35.0	8.0500	S
...	...	...	...	...	...	...
886	0	2	male	27.0	13.0000	S
887	1	1	female	19.0	30.0000	S
888	0	3	female	28.0	23.4500	S
889	1	1	male	26.0	30.0000	C
890	0	3	male	32.0	7.7500	Q

891 rows × 6 columns

In [81]:
```
data1.isna().sum()# finding the if any values
```

Out[81]:
```
Survived    0
Pclass      0
Sex         0
Age         0
Fare        0
Embarked    2
dtype: int64
```

In [82]:
```
data1
```

Out[82]:

	Survived	Pclass	Sex	Age	Fare	Embarked
0	0	3	male	22.0	7.2500	S
1	1	1	female	38.0	71.2833	C
2	1	3	female	26.0	7.9250	S
3	1	1	female	35.0	53.1000	S
4	0	3	male	35.0	8.0500	S
...	...	...	...	...	...	...
886	0	2	male	27.0	13.0000	S
887	1	1	female	19.0	30.0000	S
888	0	3	female	28.0	23.4500	S
889	1	1	male	26.0	30.0000	C
890	0	3	male	32.0	7.7500	Q

891 rows × 6 columns

In [ ]:
```

```

In [ ]:
```
#y=data1["Age"]
```

In [ ]:
```
#y
```

In [83]:
```
# data1.fillna(35,inplace=True) for replacing null values with integers as this salso
```

In [85]:
```
data1["Sex"]=data1["Sex"].map({"male":1,"female":0})# mapping the values of col-sex
data1["Pclass"].unique()
```

Out[85]:
```
array([3, 1, 2])
```

In [86]:
```
data1
```

Out[86]:

	Survived	Pclass	Sex	Age	Fare	Embarked
0	0	3	1	22.0	7.2500	S
1	1	1	0	38.0	71.2833	C
2	1	3	0	26.0	7.9250	S
3	1	1	0	35.0	53.1000	S
4	0	3	1	35.0	8.0500	S
...	...	...	...	...	...	...
886	0	2	1	27.0	13.0000	S
887	1	1	0	19.0	30.0000	S
888	0	3	0	28.0	23.4500	S
889	1	1	1	26.0	30.0000	C
890	0	3	1	32.0	7.7500	Q

891 rows × 6 columns

In [87]:
```
data1["Pclass"]=data1["Pclass"].map({1:"F",2:"S",3:"Third"})#MAPPING THE VALUES OF COL Pclass
```

```

```

In [88]:
```
data1
```

Out[88]:

	Survived	Pclass	Sex	Age	Fare	Embarked
0	0	Third	1	22.0	7.2500	S
1	1	F	0	38.0	71.2833	C
2	1	Third	0	26.0	7.9250	S
3	1	F	0	35.0	53.1000	S
4	0	Third	1	35.0	8.0500	S
...	...	...	...	...	...	...
886	0	S	1	27.0	13.0000	S
887	1	F	0	19.0	30.0000	S
888	0	Third	0	28.0	23.4500	S
889	1	F	1	26.0	30.0000	C
890	0	Third	1	32.0	7.7500	Q

891 rows × 6 columns

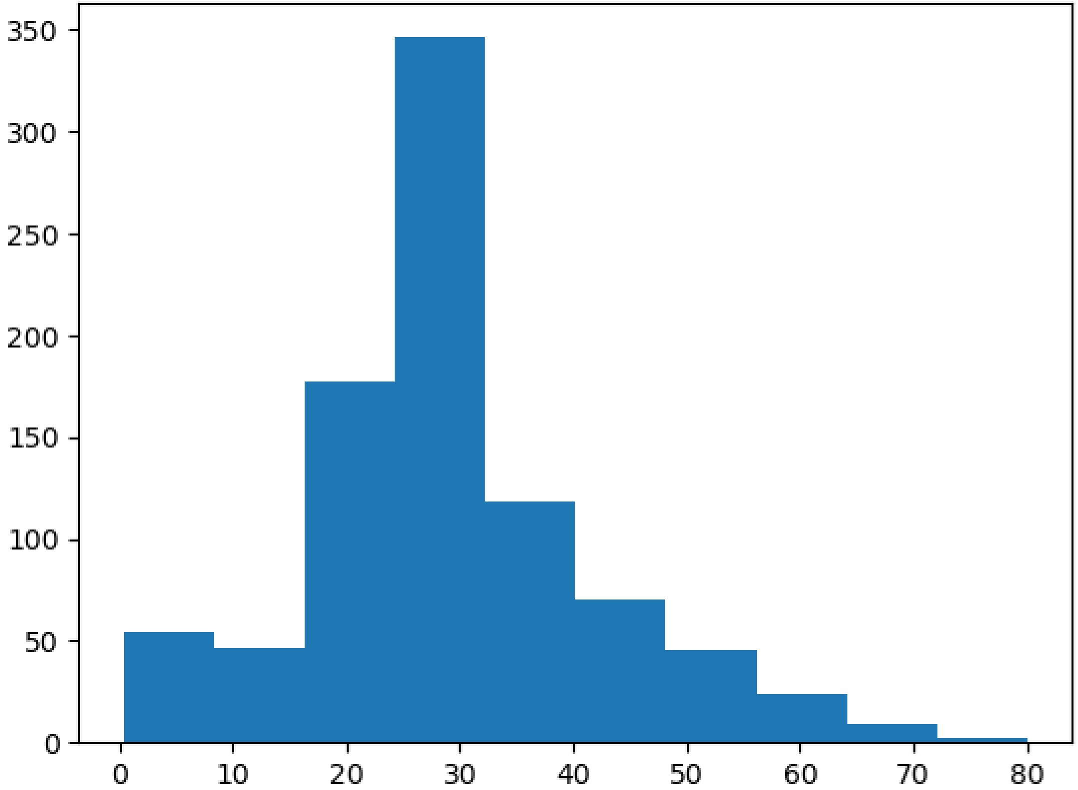
In [89]:
```
data1.isna().sum()
```

Out[89]:
```
Survived    0
Pclass      0
Sex         0
Age         0
Fare        0
Embarked    2
dtype: int64
```

In [ ]:

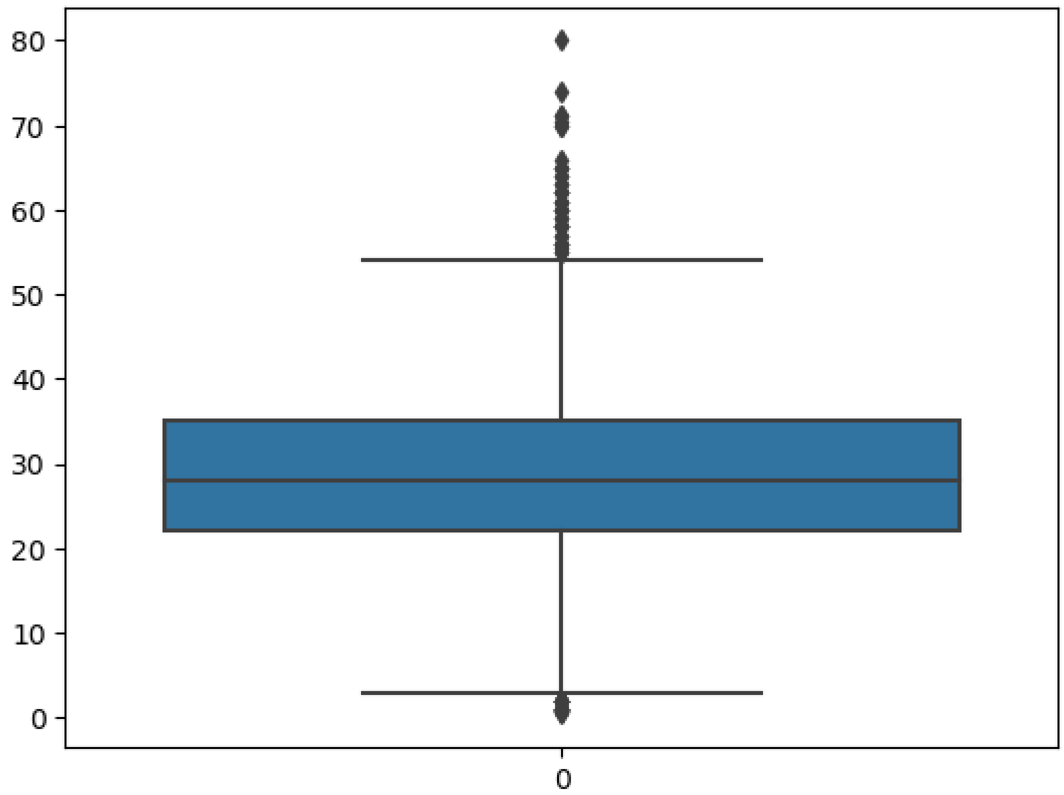
In [90]:
```
plt.hist(data1["Age"]) #plotting histogram of age
```

Out[90]:
```
(array([ 54.,  46., 177., 346., 118.,  70.,  45.,  24.,   9.,   2.]),
 array([ 0.42 ,  8.378, 16.336, 24.294, 32.252, 40.21 , 48.168, 56.126,
        64.084, 72.042, 80.   ]),
 <BarContainer object of 10 artists>)
```



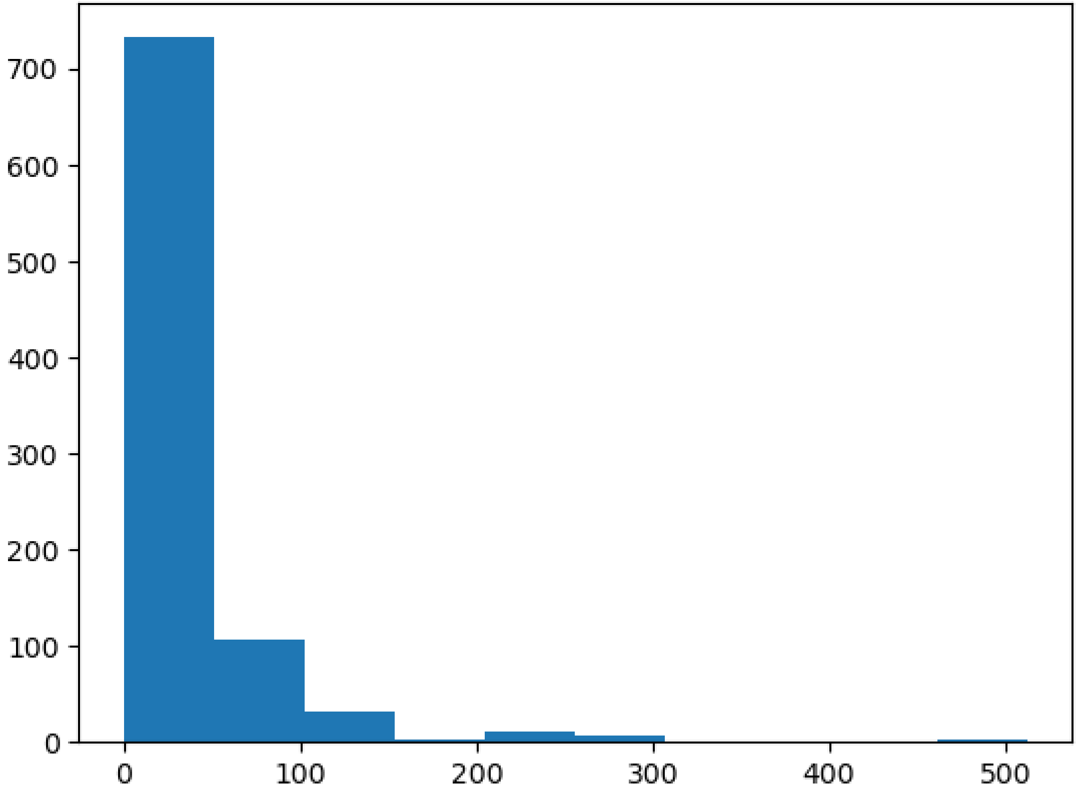
In [91]:
```
sns.boxplot(data1.Age)# printing the boxplot of age
```

Out[91]: <Axes: >

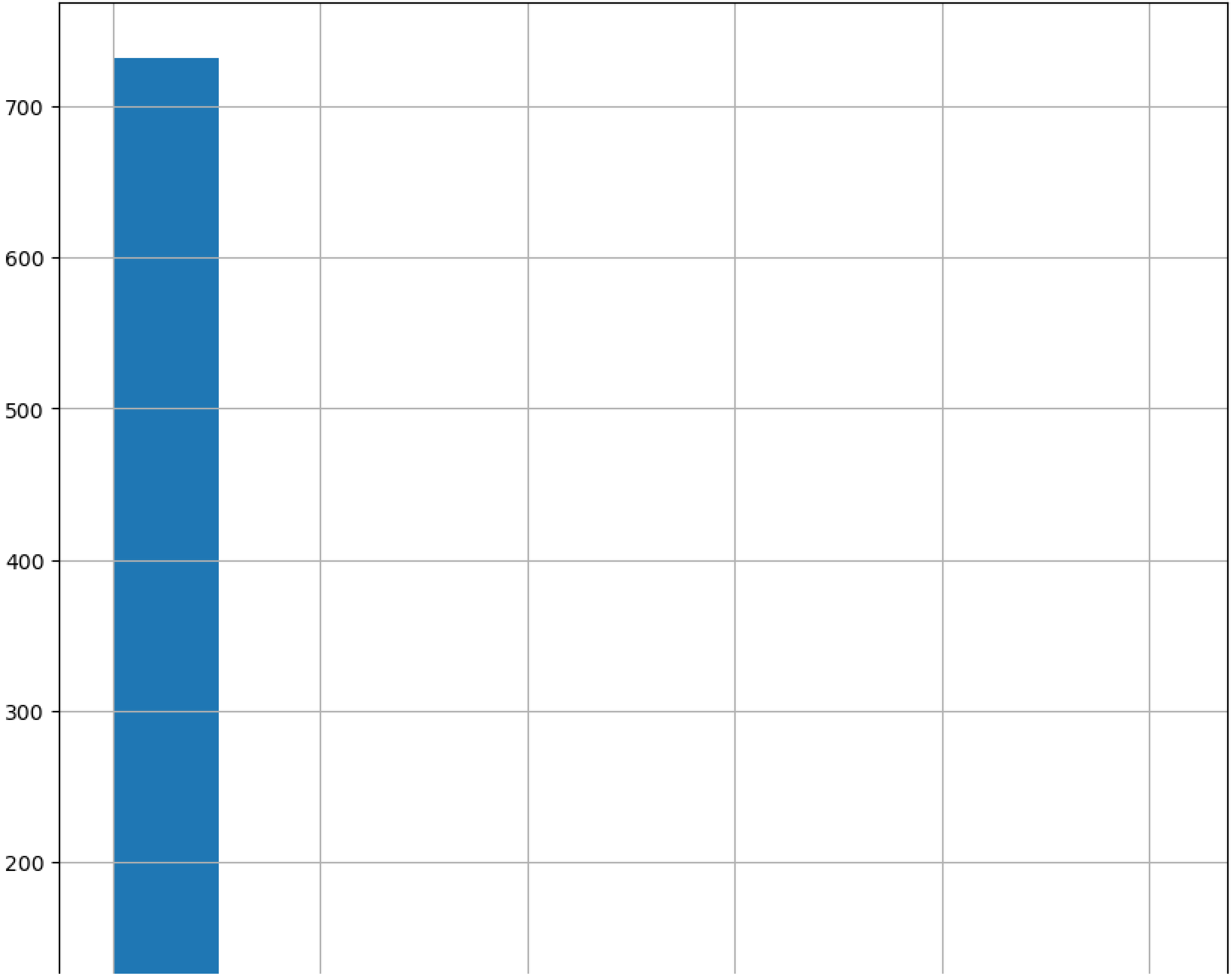


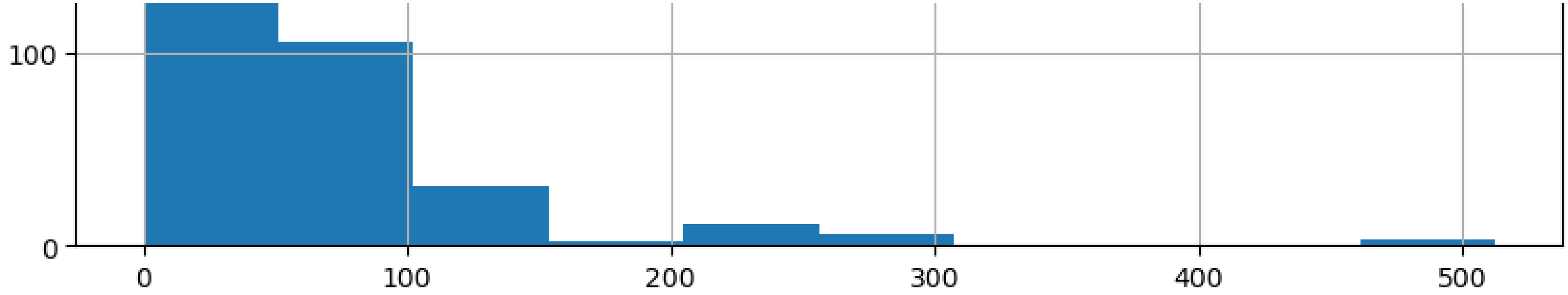
In [92]:
```
plt.hist(data1["Fare"])
```

Out[92]:
```
(array([732., 106.,  31.,   2.,  11.,   6.,   0.,   0.,   0.,   3.]),
 array([  0.     ,  51.23292, 102.46584, 153.69876, 204.93168, 256.1646 ,
        307.39752, 358.63044, 409.86336, 461.09628, 512.3292 ]),
 <BarContainer object of 10 artists>)
```



In [93]:
```
data1['Fare'].hist(figsize=(10,10))
plt.show()# ploting the histogram
```





In [94]:
```
data1.groupby(["Age"]).count()
```

Out[94]:

Age	Survived	Pclass	Sex	Fare	Embarked
0.42	1	1	1	1	1
0.67	1	1	1	1	1
0.75	2	2	2	2	2
0.83	2	2	2	2	2
0.92	1	1	1	1	1
...	...	...	...	...	...
70.00	2	2	2	2	2
70.50	1	1	1	1	1
71.00	2	2	2	2	2
74.00	1	1	1	1	1
80.00	1	1	1	1	1

88 rows × 5 columns

In [ ]:

In [95]:
```
data1.groupby(["Age"]).sum()
```

Out[95]:

	Survived	Sex	Fare
Age			
0.42	1	1	8.5167
0.67	1	1	14.5000
0.75	2	0	38.5166
0.83	2	2	47.7500
0.92	1	1	151.5500
...	...	...	...
70.00	0	2	81.5000
70.50	0	1	7.7500
71.00	0	2	84.1584
74.00	0	1	7.7750
80.00	1	1	30.0000

88 rows × 3 columns

In [96]:
```
data1=pd.get_dummies(data1)#replacing the strings wih integers
```

In [97]:
```
data1.isna().sum()# checking if any null values are presented
```

Out[97]:
```
Survived        0
Sex             0
Age             0
Fare            0
Pclass_F        0
Pclass_S        0
Pclass_Third    0
Embarked_C      0
Embarked_Q      0
Embarked_S      0
dtype: int64
```

In [ ]:

In [98]:
```
data1
```

Out[98]:

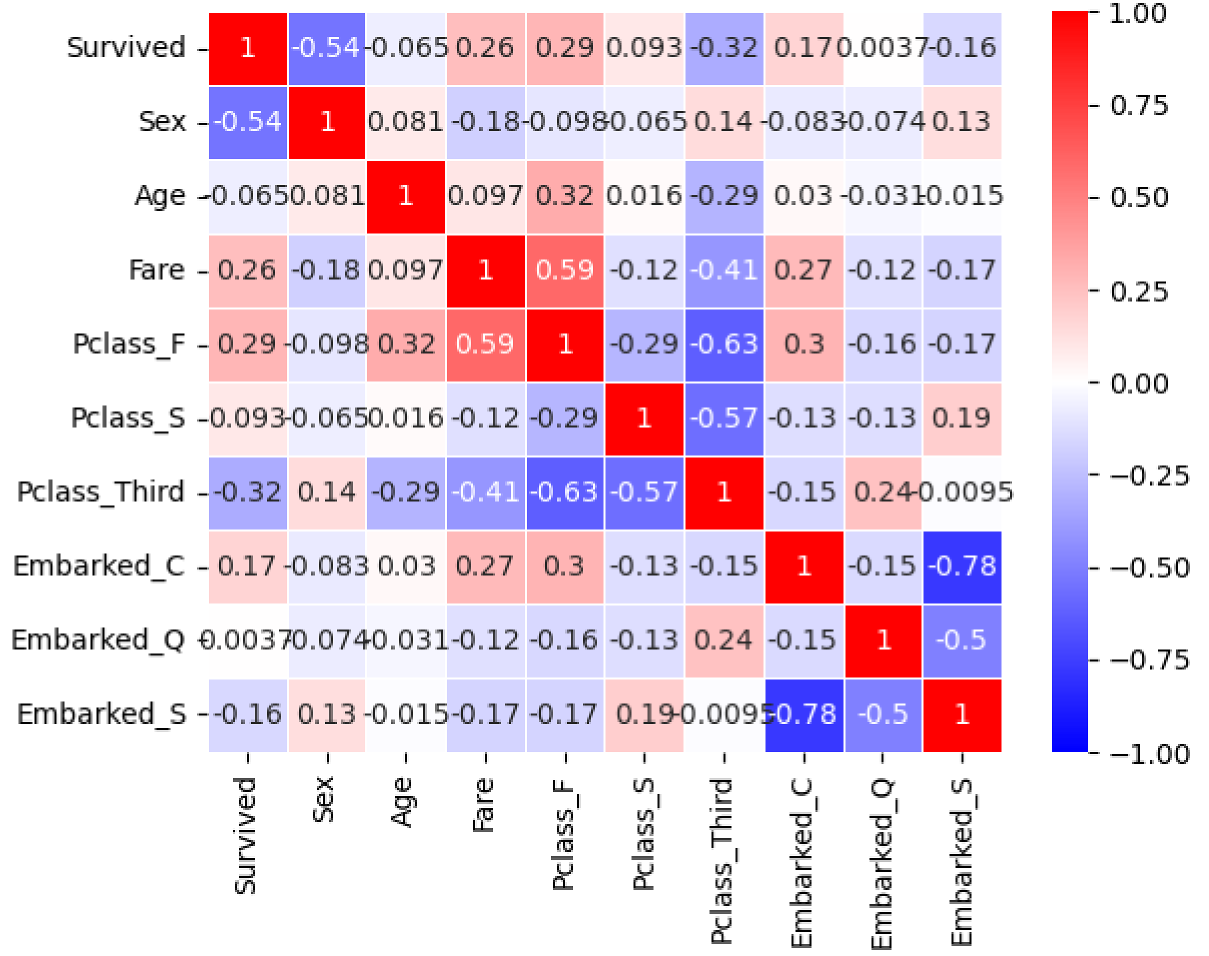
	Survived	Sex	Age	Fare	Pclass_F	Pclass_S	Pclass_Third	Embarked_C	Embarked_Q	Embarked_S
0	0	1	22.0	7.2500	0	0	1	0	0	1
1	1	0	38.0	71.2833	1	0	0	1	0	0
2	1	0	26.0	7.9250	0	0	1	0	0	1
3	1	0	35.0	53.1000	1	0	0	0	0	1
4	0	1	35.0	8.0500	0	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...	...
886	0	1	27.0	13.0000	0	1	0	0	0	1
887	1	0	19.0	30.0000	1	0	0	0	0	1
888	0	0	28.0	23.4500	0	0	1	0	0	1
889	1	1	26.0	30.0000	1	0	0	1	0	0
890	0	1	32.0	7.7500	0	0	1	0	1	0

891 rows × 10 columns

In [99]:
```
cor=data1.corr()# USING THE CORRELATION FUNCTION
```

In [100]:
```
sns.heatmap(cor,vmax=1,vmin=-1,annot=True,linewidths=.5,cmap="bwr")# PRINTIN THE HEATMMAP OF COR DATA
```

Out[100]: <Axes: >



In [101]:
```
data1.groupby("Survived").count()
```

Out[101]:

Survived	Sex	Age	Fare	Pclass_F	Pclass_S	Pclass_Third	Embarked_C	Embarked_Q	Embarked_S
0	549	549	549	549	549	549	549	549	549
1	342	342	342	342	342	342	342	342	342

In [102]:
```
y=data1["Survived"]
x=data1.drop("Survived",axis=1)# REMOVING AND SAVING THE SURVIVED COL FOR THE PREDICTED VALES
```

In [103]:
```
y
```

Out[103]:
```
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 [ ]:
```

```

In [104]:
```
cor
```

Out[104]:

	Survived	Sex	Age	Fare	Pclass_F	Pclass_S	Pclass_Third	Embarked_C	Embarked_Q	Embarked_S
Survived	1.000000	-0.543351	-0.064910	0.257307	0.285904	0.093349	-0.322308	0.168240	0.003650	-0.155660
Sex	-0.543351	1.000000	0.081163	-0.182333	-0.098013	-0.064746	0.137143	-0.082853	-0.074115	0.125722
Age	-0.064910	0.081163	1.000000	0.096688	0.323896	0.015831	-0.291955	0.030248	-0.031415	-0.014665
Fare	0.257307	-0.182333	0.096688	1.000000	0.591711	-0.118557	-0.413333	0.269335	-0.117216	-0.166603
Pclass_F	0.285904	-0.098013	0.323896	0.591711	1.000000	-0.288585	-0.626738	0.296423	-0.155342	-0.170379
Pclass_S	0.093349	-0.064746	0.015831	-0.118557	-0.288585	1.000000	-0.565210	-0.125416	-0.127301	0.192061
Pclass_Third	-0.322308	0.137143	-0.291955	-0.413333	-0.626738	-0.565210	1.000000	-0.153329	0.237449	-0.009511
Embarked_C	0.168240	-0.082853	0.030248	0.269335	0.296423	-0.125416	-0.153329	1.000000	-0.148258	-0.778359
Embarked_Q	0.003650	-0.074115	-0.031415	-0.117216	-0.155342	-0.127301	0.237449	-0.148258	1.000000	-0.496624
Embarked_S	-0.155660	0.125722	-0.014665	-0.166603	-0.170379	0.192061	-0.009511	-0.778359	-0.496624	1.000000

In [105]:
```
data1
```

Out[105]:

	Survived	Sex	Age	Fare	Pclass_F	Pclass_S	Pclass_Third	Embarked_C	Embarked_Q	Embarked_S
0	0	1	22.0	7.2500	0	0	1	0	0	1
1	1	0	38.0	71.2833	1	0	0	1	0	0
2	1	0	26.0	7.9250	0	0	1	0	0	1
3	1	0	35.0	53.1000	1	0	0	0	0	1
4	0	1	35.0	8.0500	0	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...	...
886	0	1	27.0	13.0000	0	1	0	0	0	1
887	1	0	19.0	30.0000	1	0	0	0	0	1
888	0	0	28.0	23.4500	0	0	1	0	0	1
889	1	1	26.0	30.0000	1	0	0	1	0	0
890	0	1	32.0	7.7500	0	0	1	0	1	0

891 rows × 10 columns

In [106]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
#splits data into 33% testing and 66% training data
```

In [113]:
```
x_test
```

Out[113]:

	Sex	Age	Fare	Pclass_F	Pclass_S	Pclass_Third	Embarked_C	Embarked_Q	Embarked_S
709	1	28.0	15.2458	0	0	1	1	0	0
439	1	31.0	10.5000	0	1	0	0	0	1
840	1	20.0	7.9250	0	0	1	0	0	1
720	0	6.0	33.0000	0	1	0	0	0	1
39	0	14.0	11.2417	0	0	1	1	0	0
...	...	...	...	...	...	...	...	...	...
715	1	19.0	7.6500	0	0	1	0	0	1
525	1	40.5	7.7500	0	0	1	0	1	0
381	0	1.0	15.7417	0	0	1	1	0	0
140	0	28.0	15.2458	0	0	1	1	0	0
173	1	21.0	7.9250	0	0	1	0	0	1

295 rows × 9 columns

In [114]:
```
y_test
```

Out[114]:
```
709    1
439    0
840    0
720    1
39     1
      ..
715    0
525    0
381    1
140    0
173    0
Name: Survived, Length: 295, dtype: int64
```

In [115]:
```
x_train
```

Out[115]:

	Sex	Age	Fare	Pclass_F	Pclass_S	Pclass_Third	Embarked_C	Embarked_Q	Embarked_S
6	1	54.0	51.8625	1	0	0	0	0	1
718	1	28.0	15.5000	0	0	1	0	1	0
685	1	25.0	41.5792	0	1	0	1	0	0
73	1	26.0	14.4542	0	0	1	1	0	0
882	0	22.0	10.5167	0	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...
106	0	21.0	7.6500	0	0	1	0	0	1
270	1	28.0	31.0000	1	0	0	0	0	1
860	1	41.0	14.1083	0	0	1	0	0	1
435	0	14.0	120.0000	1	0	0	0	0	1
102	1	21.0	77.2875	1	0	0	0	0	1

596 rows × 9 columns

In [ ]:
```

```

In [107]:
```
from sklearn.linear_model import LogisticRegression
reg=LogisticRegression()
reg.fit(x_train,y_train)
#importing logistic regression
```

Out[107]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [108]:
```
y_pred=reg.predict(x_test)
```

In [109]:
```
y_pred
```

Out[109]:
```
array([0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0])
```

In [110]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)# CONFUSIO MATRIX OF TRUE POSITIVE&NEGATIVE , FASLE POSITIVE & NEGAITVE
```

Out[110]:
```
array([[154,  21],
       [ 37,  83]])
```

In [111]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)#EFFICENCY OF THE CONFUSION MATRIX
```

Out[111]:
```
0.8033898305084746
```

In [ ]: