In [2]:
```
import pyforest
```

In [4]:
```
data = pd.read_csv("D:\\Software Very Important\\Exploratory Data Analysis\\titani
data
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875
...	...	...	...	...	...	...	...	...	...	...
413	1305	0	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500
414	1306	1	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000
415	1307	0	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500
416	1308	0	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500
417	1309	0	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583

418 rows × 12 columns

In [5]:
```
data.shape
```

Out[5]:
```
(418, 12)
```

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Survived     418 non-null    int64  
 2   Pclass       418 non-null    int64  
 3   Name         418 non-null    object 
 4   Sex          418 non-null    object 
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64  
 7   Parch        418 non-null    int64  
 8   Ticket       418 non-null    object 
 9   Fare         417 non-null    float64
 10  Cabin        91 non-null     object 
 11  Embarked     418 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
```

In [7]:
```
data.isna().sum()
```

Out[7]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [8]:
```
data.describe()
```

Out[8]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	0.363636	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.481622	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	0.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	0.000000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	0.000000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	1.000000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	1.000000	3.000000	76.000000	8.000000	9.000000	512.329200

In [9]:
```
data.dtypes
```

Out[9]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [10]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
data['Sex'] = label_encoder.fit_transform(data['Sex'])
data['Sex'].value_counts()
```

Out[10]:
```
1    266
0    152
Name: Sex, dtype: int64
```

In [11]:
```
list_to_drop=["Cabin","Name"]
data= data.drop(list_to_drop,axis=1)
data
```

Out[11]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	892	0	3	1	34.5	0	0	330911	7.8292	Q
1	893	1	3	0	47.0	1	0	363272	7.0000	S
2	894	0	2	1	62.0	0	0	240276	9.6875	Q
3	895	0	3	1	27.0	0	0	315154	8.6625	S
4	896	1	3	0	22.0	1	1	3101298	12.2875	S
...	...	...	...	...	...	...	...	...	...	...
413	1305	0	3	1	NaN	0	0	A.5. 3236	8.0500	S
414	1306	1	1	0	39.0	0	0	PC 17758	108.9000	C
415	1307	0	3	1	38.5	0	0	SOTON/O.Q. 3101262	7.2500	S
416	1308	0	3	1	NaN	0	0	359309	8.0500	S
417	1309	0	3	1	NaN	1	1	2668	22.3583	C

418 rows × 10 columns

In [12]:
```
data['Age'].median()
```

Out[12]:
```
27.0
```

In [15]:
```
data['Age']=data['Age'].fillna(value=27)
data
```

Out[15]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	892	0	3	1	34.5	0	0	330911	7.8292	Q
1	893	1	3	0	47.0	1	0	363272	7.0000	S
2	894	0	2	1	62.0	0	0	240276	9.6875	Q
3	895	0	3	1	27.0	0	0	315154	8.6625	S
4	896	1	3	0	22.0	1	1	3101298	12.2875	S
...	...	...	...	...	...	...	...	...	...	...
413	1305	0	3	1	28.0	0	0	A.5. 3236	8.0500	S
414	1306	1	1	0	39.0	0	0	PC 17758	108.9000	C
415	1307	0	3	1	38.5	0	0	SOTON/O.Q. 3101262	7.2500	S
416	1308	0	3	1	28.0	0	0	359309	8.0500	S
417	1309	0	3	1	28.0	1	1	2668	22.3583	C

418 rows × 10 columns

In [17]:
```
data['Fare'].median()
```

Out[17]:
```
14.4542
```

In [18]:
```
data['Fare']=data['Fare'].fillna(value=14)
data
```

Out[18]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	892	0	3	1	34.5	0	0	330911	7.8292	Q
1	893	1	3	0	47.0	1	0	363272	7.0000	S
2	894	0	2	1	62.0	0	0	240276	9.6875	Q
3	895	0	3	1	27.0	0	0	315154	8.6625	S
4	896	1	3	0	22.0	1	1	3101298	12.2875	S
...	...	...	...	...	...	...	...	...	...	...
413	1305	0	3	1	28.0	0	0	A.5. 3236	8.0500	S
414	1306	1	1	0	39.0	0	0	PC 17758	108.9000	C
415	1307	0	3	1	38.5	0	0	SOTON/O.Q. 3101262	7.2500	S
416	1308	0	3	1	28.0	0	0	359309	8.0500	S
417	1309	0	3	1	28.0	1	1	2668	22.3583	C

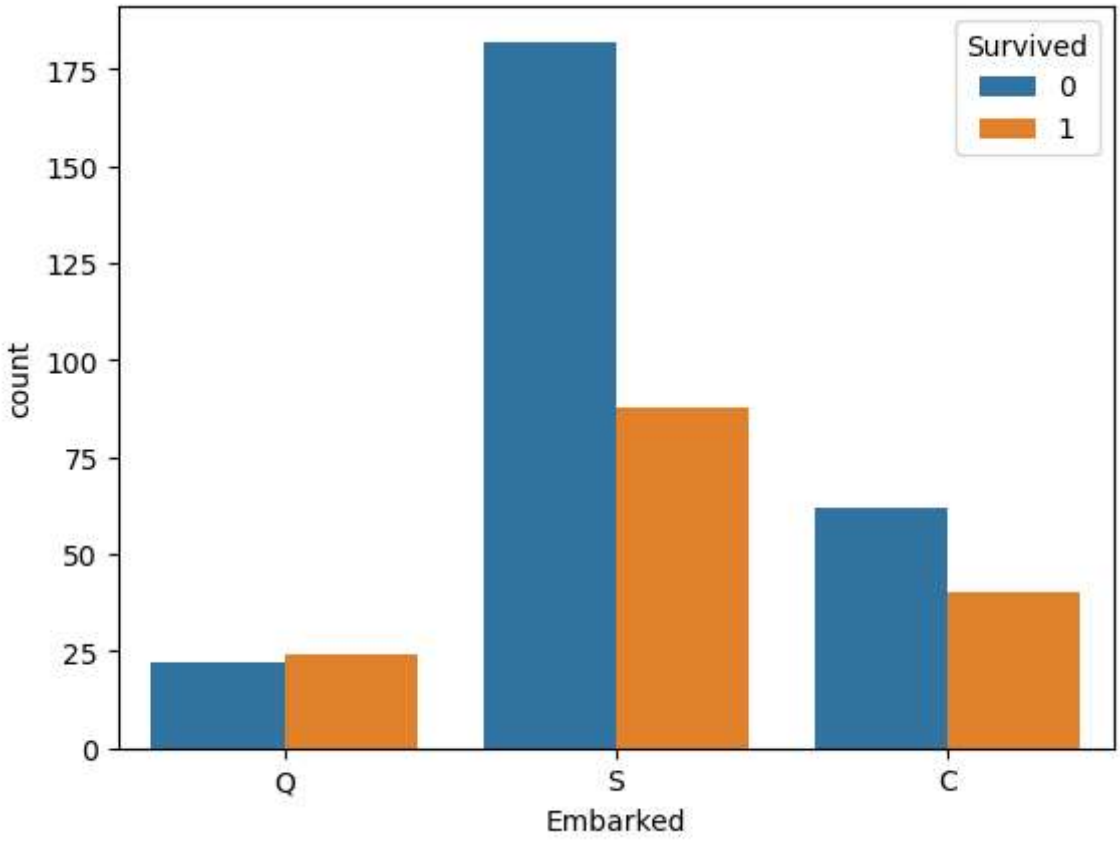
418 rows × 10 columns

In [19]:
```
data.isna().sum()
```

Out[19]:
```
PassengerId    0
Survived       0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

In [29]:
```
sns.countplot(data['Embarked'],hue=data['Survived'])
plt.show()
```

```
C:\Users\bhava\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only va
lid positional argument will be `data`, and passing other arguments without an exp
licit keyword will result in an error or misinterpretation.
```

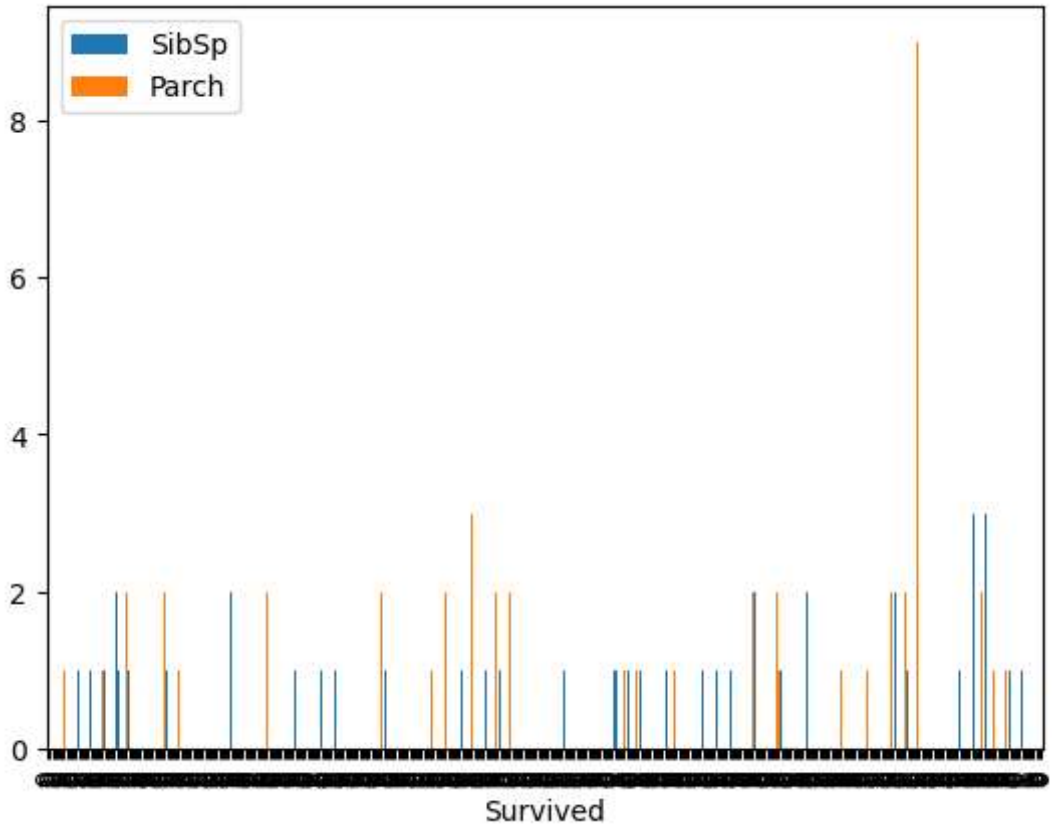


In [27]:
```
data.corr()
```

Out[27]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fa
PassengerId	1.000000	-0.023245	-0.026751	0.023245	-0.031309	0.003818	0.043080	0.0086
Survived	-0.023245	1.000000	-0.108615	-1.000000	0.005592	0.099943	0.159120	0.1920
Pclass	-0.026751	-0.108615	1.000000	0.108615	-0.460143	0.001087	0.018721	-0.5773
Sex	0.023245	-1.000000	0.108615	1.000000	-0.005592	-0.099943	-0.159120	-0.1920
Age	-0.031309	0.005592	-0.460143	-0.005592	1.000000	-0.073820	-0.044191	0.3372
SibSp	0.003818	0.099943	0.001087	-0.099943	-0.073820	1.000000	0.306895	0.1719
Parch	0.043080	0.159120	0.018721	-0.159120	-0.044191	0.306895	1.000000	0.2303
Fare	0.008642	0.192049	-0.577326	-0.192049	0.337282	0.171920	0.230331	1.0000

In [32]:
```
data.plot(x="Survived" , y=['SibSp','Parch'],kind="bar")
plt.show()
```

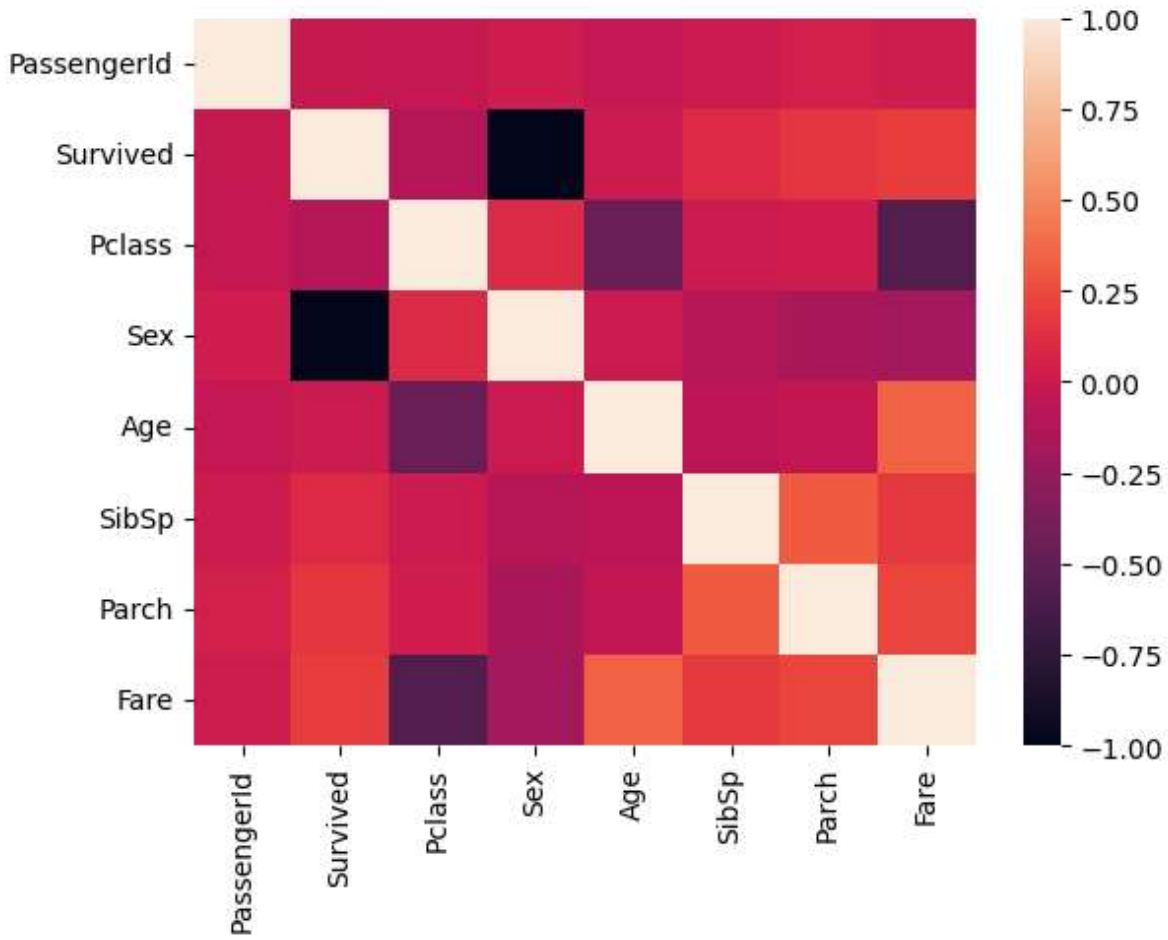


In [34]:
```
correlation=data.corr()
correlation['Survived'].sort_values(ascending=False)
```

Out[34]:
```
Survived       1.000000
Fare           0.192049
Parch          0.159120
SibSp          0.099943
Age            0.005592
PassengerId   -0.023245
Pclass        -0.108615
Sex           -1.000000
Name: Survived, dtype: float64
```

In [35]:
```
sns.heatmap(data.corr())
```

Out[35]: <AxesSubplot:>



In [37]:
```
data
```

Out[37]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	892	0	3	1	34.5	0	0	330911	7.8292	Q
1	893	1	3	0	47.0	1	0	363272	7.0000	S
2	894	0	2	1	62.0	0	0	240276	9.6875	Q
3	895	0	3	1	27.0	0	0	315154	8.6625	S
4	896	1	3	0	22.0	1	1	3101298	12.2875	S
...	...	...	...	...	...	...	...	...	...	...
413	1305	0	3	1	28.0	0	0	A.5. 3236	8.0500	S
414	1306	1	1	0	39.0	0	0	PC 17758	108.9000	C
415	1307	0	3	1	38.5	0	0	SOTON/O.Q. 3101262	7.2500	S
416	1308	0	3	1	28.0	0	0	359309	8.0500	S
417	1309	0	3	1	28.0	1	1	2668	22.3583	C

418 rows × 10 columns

In [66]:
```
# data['Family']=data['SibSp']
# data=data.drop(['SibSp','Parch'],axis=1)
```

```
# data=data.drop('PassengerId',axis=1)
# data=data.drop('Embarked',axis=1)
data=data.drop('Ticket',axis=1)
```

In [67]:
```
data
```

Out[67]:

	Survived	Pclass	Sex	Age	Fare	Family
0	0	3	1	34.5	7.8292	0
1	1	3	0	47.0	7.0000	1
2	0	2	1	62.0	9.6875	0
3	0	3	1	27.0	8.6625	0
4	1	3	0	22.0	12.2875	1
...	...	...	...	...	...	...
413	0	3	1	28.0	8.0500	0
414	1	1	0	39.0	108.9000	0
415	0	3	1	38.5	7.2500	0
416	0	3	1	28.0	8.0500	0
417	0	3	1	28.0	22.3583	1

418 rows × 6 columns

In [68]:
```
x=data.drop('Survived',axis=1).values
y=data['Survived'].values
```

In [69]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [70]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=100)
```

In [71]:
```
from sklearn.metrics import accuracy_score
```

In [72]:
```
lr=LogisticRegression()
lr.fit(x_train,y_train)
lrpred=lr.predict(x_test)
```

In [73]:
```
accuracy_score(y_test,lrpred)
```

Out[73]:
```
1.0
```