In [27]:

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab

from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [28]:

```
passengers= pd.read_excel(r"C:\Users\Shrikant\Desktop\IMS Project\train data.xlsx")
passengers.head()
```

Out[28]:

	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 [29]:

```
passengers= passengers.drop('PassengerId', axis=1)
passengers= passengers.drop('Name', axis=1)
passengers= passengers.drop('Cabin', axis=1)
passengers= passengers.drop('Embarked', axis=1)
passengers= passengers.drop('Fare', axis=1)
passengers= passengers.drop('Ticket', axis=1)
passengers.head()
```

Out[29]:

	Survived	Pclass	Sex	Age	SibSp	Parch
0	0	3	male	22.0	1	0
1	1	1	female	38.0	1	0
2	1	3	female	26.0	0	0
3	1	1	female	35.0	1	0
4	0	3	male	35.0	0	0

In [30]:

```
passengers.shape
```

Out[30]:

```
(891, 6)
```

In [31]:

```
passengers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Pclass    891 non-null    int64  
 2   Sex       891 non-null    object 
 3   Age       714 non-null    float64
 4   SibSp     891 non-null    int64  
 5   Parch     891 non-null    int64  
dtypes: float64(1), int64(4), object(1)
memory usage: 41.9+ KB
```

In [32]:

```
print(passengers.isnull().sum())
```

```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
dtype: int64
```

In [33]:

```
passengers['Age']= passengers.Age.fillna(passengers.Age.mean())
passengers.head()
```

Out[33]:

	Survived	Pclass	Sex	Age	SibSp	Parch
0	0	3	male	22.0	1	0
1	1	1	female	38.0	1	0
2	1	3	female	26.0	0	0
3	1	1	female	35.0	1	0
4	0	3	male	35.0	0	0

In [34]:

```
passengers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Pclass    891 non-null    int64  
 2   Sex       891 non-null    object 
 3   Age       891 non-null    float64
 4   SibSp     891 non-null    int64  
 5   Parch     891 non-null    int64  
dtypes: float64(1), int64(4), object(1)
memory usage: 41.9+ KB
```

In [35]:

```
passengers.Survived.value_counts()
```

Out[35]:

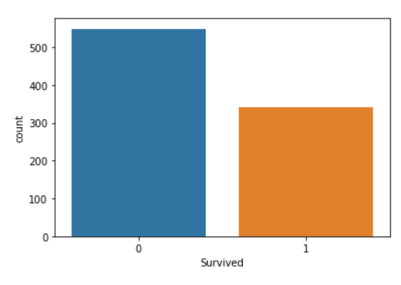
```
0    549
1    342
Name: Survived, dtype: int64
```

In [36]:

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

Out[36]:

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



- Number of people who survived are less than the number of people who died.

In [37]:

```
passengers.groupby('Survived').mean()
```

Out[37]:

Survived	Pclass	Age	SibSp	Parch
0	2.531876	30.415100	0.553734	0.329690
1	1.950292	28.549778	0.473684	0.464912

In [38]:

```
passengers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Pclass    891 non-null    int64  
 2   Sex       891 non-null    object 
 3   Age       891 non-null    float64
 4   SibSp     891 non-null    int64  
 5   Parch     891 non-null    int64  
dtypes: float64(1), int64(4), object(1)
memory usage: 41.9+ KB
```

In [39]:

```
X= passengers.drop('Survived', axis= 1)
Y= passengers[['Survived']]
X_train, X_test, Y_train, Y_test= train_test_split(X,Y, test_size=0.30, random_state=1)
```

In [40]:

```
train= pd.concat([X_train, Y_train], axis=1)
train.head()
```

Out[40]:

	Pclass	Sex	Age	SibSp	Parch	Survived
114	3	female	17.000000	0	0	0
874	2	female	28.000000	1	0	1
76	3	male	29.699118	0	0	0
876	3	male	20.000000	0	0	0
674	2	male	29.699118	0	0	0

Hypothesis

- H0= There is no relationship between Survival and Pclass, Sex, Age, SipSp, Parch.

VS

- H1= There is relationship between Survival and Pclass, Sex, Age, SipSp, Parch.

In [41]:

```
import statsmodels.formula.api as smf
result= smf.logit('Survived~ Pclass + C(Sex) + Age + SibSp + Parch', data= train).fit()
result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.419434
         Iterations 6
```

Out[41]:

Logit Regression Results

Dep. Variable:	Survived	No. Observations:	623
Model:	Logit	Df Residuals:	617
Method:	MLE	Df Model:	5
Date:	Sat, 23 Jul 2022	Pseudo R-squ.:	0.3605
Time:	12:46:54	Log-Likelihood:	-261.31
converged:	True	LL-Null:	-408.62
Covariance Type:	nonrobust	LLR p-value:	1.434e-61

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	5.4195	0.602	9.007	0.000	4.240	6.599
C(Sex)[T.male]	-2.9127	0.245	-11.902	0.000	-3.392	-2.433
Pclass	-1.2281	0.149	-8.263	0.000	-1.519	-0.937
Age	-0.0429	0.009	-4.523	0.000	-0.062	-0.024
SibSp	-0.3802	0.140	-2.710	0.007	-0.655	-0.105
Parch	0.0393	0.139	0.282	0.778	-0.234	0.312

In [42]:

```
result= smf.logit('Survived~ Pclass + C(Sex) + Age + SibSp', data= train).fit()
result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.419498
         Iterations 6
```

Out[42]:

Logit Regression Results

Dep. Variable:	Survived	No. Observations:	623
Model:	Logit	Df Residuals:	618
Method:	MLE	Df Model:	4
Date:	Sat, 23 Jul 2022	Pseudo R-squ.:	0.3604
Time:	12:46:54	Log-Likelihood:	-261.35
converged:	True	LL-Null:	-408.62
Covariance Type:	nonrobust	LLR p-value:	1.628e-62

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	5.4434	0.596	9.138	0.000	4.276	6.611
C(Sex)[T.male]	-2.9250	0.241	-12.135	0.000	-3.397	-2.453
Pclass	-1.2289	0.149	-8.271	0.000	-1.520	-0.938
Age	-0.0431	0.009	-4.544	0.000	-0.062	-0.024
SibSp	-0.3679	0.133	-2.770	0.006	-0.628	-0.108

- We reject H0 and conclude that there is a relationship between Sex, Pclass, Age, SibSp and Survived.

In [43]:

```
result.params
```

Out[43]:

```
Intercept         5.443434
C(Sex)[T.male]   -2.925003
Pclass           -1.228858
Age              -0.043081
SibSp            -0.367871
dtype: float64
```

In [47]:

```
train.head()
```

Out[47]:

	Pclass	Sex	Age	SibSp	Parch	Survived
114	3	female	17.000000	0	0	0
874	2	female	28.000000	1	0	1
76	3	male	29.699118	0	0	0
876	3	male	20.000000	0	0	0
674	2	male	29.699118	0	0	0

In [49]:

```
train['Probability']= result.predict(train)
train.head()
```

Out[49]:

	Pclass	Sex	Age	SibSp	Parch	Survived	Probability
114	3	female	17.000000	0	0	0	0.735845
874	2	female	28.000000	1	0	1	0.804016
76	3	male	29.699118	0	0	0	0.079614
876	3	male	20.000000	0	0	0	0.116114
674	2	male	29.699118	0	0	0	0.228157

- 73% chances that the person will die.

In [50]:

```
train['Predicted']= np.where(train['Probability']>=0.7,1,0)
train.head()
```

Out[50]:

	Pclass	Sex	Age	SibSp	Parch	Survived	Probability	Predicted
114	3	female	17.000000	0	0	0	0.735845	1
874	2	female	28.000000	1	0	1	0.804016	1
76	3	male	29.699118	0	0	0	0.079614	0
876	3	male	20.000000	0	0	0	0.116114	0
674	2	male	29.699118	0	0	0	0.228157	0

In [53]:

```
from sklearn.metrics import confusion_matrix
matrix= confusion_matrix(train['Predicted'], train['Survived'])
matrix
```

Out[53]:

```
array([[386, 107],
       [ 10, 120]], dtype=int64)
```

In [56]:

```
Accuracy_Train=((386+120)/(623)*100)
Accuracy_Train
```

Out[56]:

```
81.2199036918138
```

- Model accuracy is 81.21%. We can say that the model is a good fit.

In [55]:

```
from sklearn.metrics import classification_report
print(classification_report(train['Survived'], train['Predicted']))
```

```
              precision    recall  f1-score   support

           0       0.78      0.97      0.87       396
           1       0.92      0.53      0.67       227

    accuracy                           0.81       623
   macro avg       0.85      0.75      0.77       623
weighted avg       0.83      0.81      0.80       623
```

- Accuracy of people who are likely to die captured by Model is 97% (Sensitivity)
- Accuracy of people who are likely to survive Capture by Model is 53% (specificity)
- Accuracy of Predicted not survived And often Correct is 78%
- Accuracy of Predicted survived And often Correct is 92%

In [57]:

```
test=pd.concat([X_test, Y_test], axis=1)
test.head()
```

Out[57]:

	Pclass	Sex	Age	SibSp	Parch	Survived
862	1	female	48.000000	0	0	1
223	3	male	29.699118	0	0	0
84	2	female	17.000000	0	0	1
680	3	female	29.699118	0	0	0
535	2	female	7.000000	0	2	1

In [58]:

```
test['Probability']= result.predict(test)
test.head()
```

Out[58]:

	Pclass	Sex	Age	SibSp	Parch	Survived	Probability
862	1	female	48.000000	0	0	1	0.895360
223	3	male	29.699118	0	0	0	0.079614
84	2	female	17.000000	0	0	1	0.904939
680	3	female	29.699118	0	0	0	0.617133
535	2	female	7.000000	0	2	1	0.936085

In [59]:

```
test['Predicted']= np.where(test['Probability']>=0.7,1,0)
test.head()
```

Out[59]:

	Pclass	Sex	Age	SibSp	Parch	Survived	Probability	Predicted
862	1	female	48.000000	0	0	1	0.895360	1
223	3	male	29.699118	0	0	0	0.079614	0
84	2	female	17.000000	0	0	1	0.904939	1
680	3	female	29.699118	0	0	0	0.617133	0
535	2	female	7.000000	0	2	1	0.936085	1

In [61]:

```
from sklearn.metrics import confusion_matrix
matrix= confusion_matrix(test['Predicted'], test['Survived'])
matrix
```

Out[61]:

```
array([[146,  62],
       [  7,  53]], dtype=int64)
```

In [62]:

```
Accuracy_test= ((146+53)/(268)*100)
Accuracy_test
```

Out[62]:

```
74.25373134328358
```

- Accuracy of the model performance on test data is 74.25%, so we can say that the model is a good performance model.

In [65]:

```
from sklearn.metrics import classification_report
print(classification_report(test['Survived'],test['Predicted']))
```

```
              precision    recall  f1-score   support

           0       0.70      0.95      0.81       153
           1       0.88      0.46      0.61       115

    accuracy                           0.74       268
   macro avg       0.79      0.71      0.71       268
weighted avg       0.78      0.74      0.72       268
```

- Accuracy of people who are likely to die captured by Model is 95%
- Accuracy of people who are likely to survive Capture by Model is 46%
- Accuracy of Predicted not survived And often Correct is 70%
- Accuracy of Predicted survived And often Correct is 88%

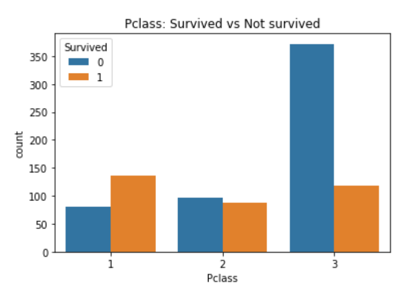
In [67]:

```
sns.countplot('Pclass', hue='Survived', data=passengers)
plt.title('Pclass: Survived vs Not survived')
```

```
C:\Users\Shrikant\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[67]:

```
Text(0.5, 1.0, 'Pclass: Survived vs Not survived')
```



- Majority of people from 3rd class died as compared to the other two classes.
- Number of people survived in 1st class is more than the number of people died.
- Class was given preference.

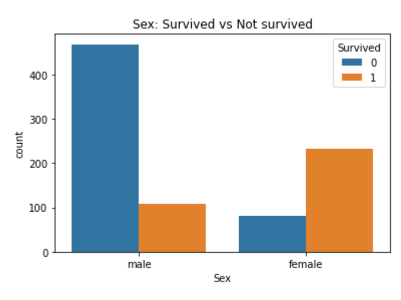
In [68]:

```
sns.countplot('Sex', hue='Survived', data=passengers)
plt.title('Sex: Survived vs Not survived')
```

```
C:\Users\Shrikant\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[68]:

```
Text(0.5, 1.0, 'Sex: Survived vs Not survived')
```



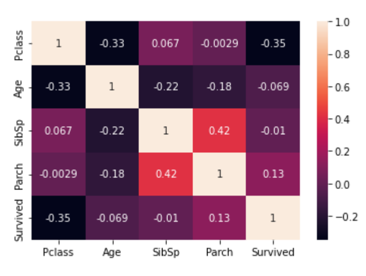
- Number of females survived is more than males.
- This indicates that sex was given preference.

In [69]:

```
train= pd.concat([X_train, Y_train], axis=1)
sns.heatmap(train.corr(), annot=True)
```

Out[69]:

```
<AxesSubplot:>
```



- There is a positive correlation between Parch and Survived.

In [70]:

```
pd.crosstab([passengers.Sex, passengers.Survived], passengers.Pclass, margins= True).style.background_gradient(cmap='summer_r')
```

Out[70]:

	Pclass	1	2	3	All
Sex	Survived				
female	0	3	6	72	81
	1	91	70	72	233
male	0	77	91	300	468
	1	45	17	47	109
All		216	184	491	891

- Number of women survived were 233, out of which 91 belonged to 1st class, 70 belonged to 2nd class and 72 belonged to 3rd class.
- Number of men survived were 109 out of which 45 belonged to 1st class, 17 belonged to 2nd class and 47 belonged to 3rd class.
- Majority people who died were males belongging to 3rd class (300 people).