Import Library

In [132…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')
from scipy.stats import chi2_contingency
from matplotlib.pyplot import figure
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PowerTransformer
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,precis
from sklearn.preprocessing import StandardScaler


from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV,GridSearchCV, KFold
from sklearn.model_selection import RepeatedStratifiedKFold
```

Data import and Understand

In [2]:

```
data_1 = pd.read_csv('C:\\Users\\DELL\\Downloads\\train.csv')
```

In [3]:

```
data = data_1.copy()
```

In [4]:

```
data.head(2)
```

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

In [5]:

```
print('Rows of dataset:-',data.shape[0])
print('Columns of dataset:-',data.shape[1])
```

```
Rows of dataset:- 891
Columns of dataset:- 12
```

summary of dataset

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ship

This is dataset about 891 titanic passenger who are travel in this dataset 891 rows and 12 columns

Features descriptions

1). PassengerId :- the unique identifier for each passenger.

2). Survived :- which passenger are survived and which passenger are not survived 0 in the sence passenger did not survived 1 in the sence passenger survived

3). Pclass:- class of the passenger 1:- in the sense first class 2:- in the sence second class 3:- in the sence third class

4). Name :- Name of passenger who are travel in titanic ship

5). Sex:- Sex of paasenger who are travel in titanic ship

6). Age:- All age of passenger who are travel in titanic ship

7). SibSp:- passenger is travel alone or with family

8). Ticket:- ticket number of passenger

9). Fare :- fare of passenger who are travel in titanic ship

10). Cabin:- cabin number of passenger who are travel in ship

11). C --> chebourge , S --> southampton , Q --> Queenstown

Issues with dataset

1). Dirty data:-

Age --> 177 missing values and datatype is float in age column (Completion) Cabin --> 687 missing values in cabin column (Completion) Embarked --> 2 missing values in mbarked column (Completion)

2). Messy data:-

Ticket and Cabin :- missed two data type in single feature

Columns types:-

1). Numerical :- PassengerId , SibSp, Parch, Fare, Age.

2). Categorical:- Embarked, Sex, Survived , Pclass, Name.

3). Mixed:- Ticket, Cabin.

In [6]:
```
# check duplicates data
data.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
data['Embarked'].value_counts()
```

Out[7]:
```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [8]:
```
# check all missing of the dataset
data.isnull().sum()
```

Out[8]:
```
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 [9]:
```
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
```

Exploratory Data Analysis

Univariate Analysis on Numerical columns

Age:-

conclusions:-

1) Age is (almost) normal distributed

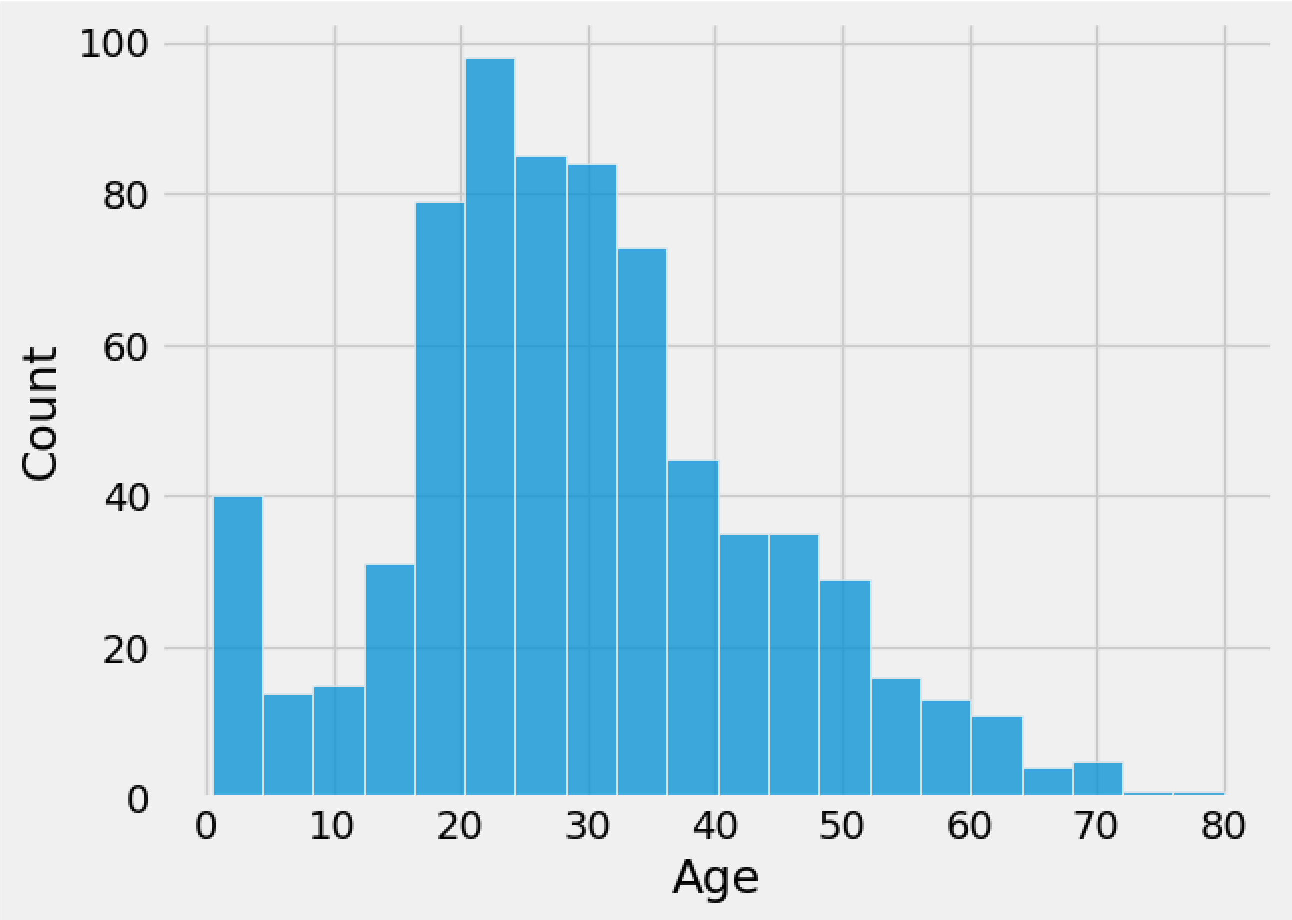
2) 20% missing values

3) There are some outliers

In [10]:
```
data['Age'].describe()
```

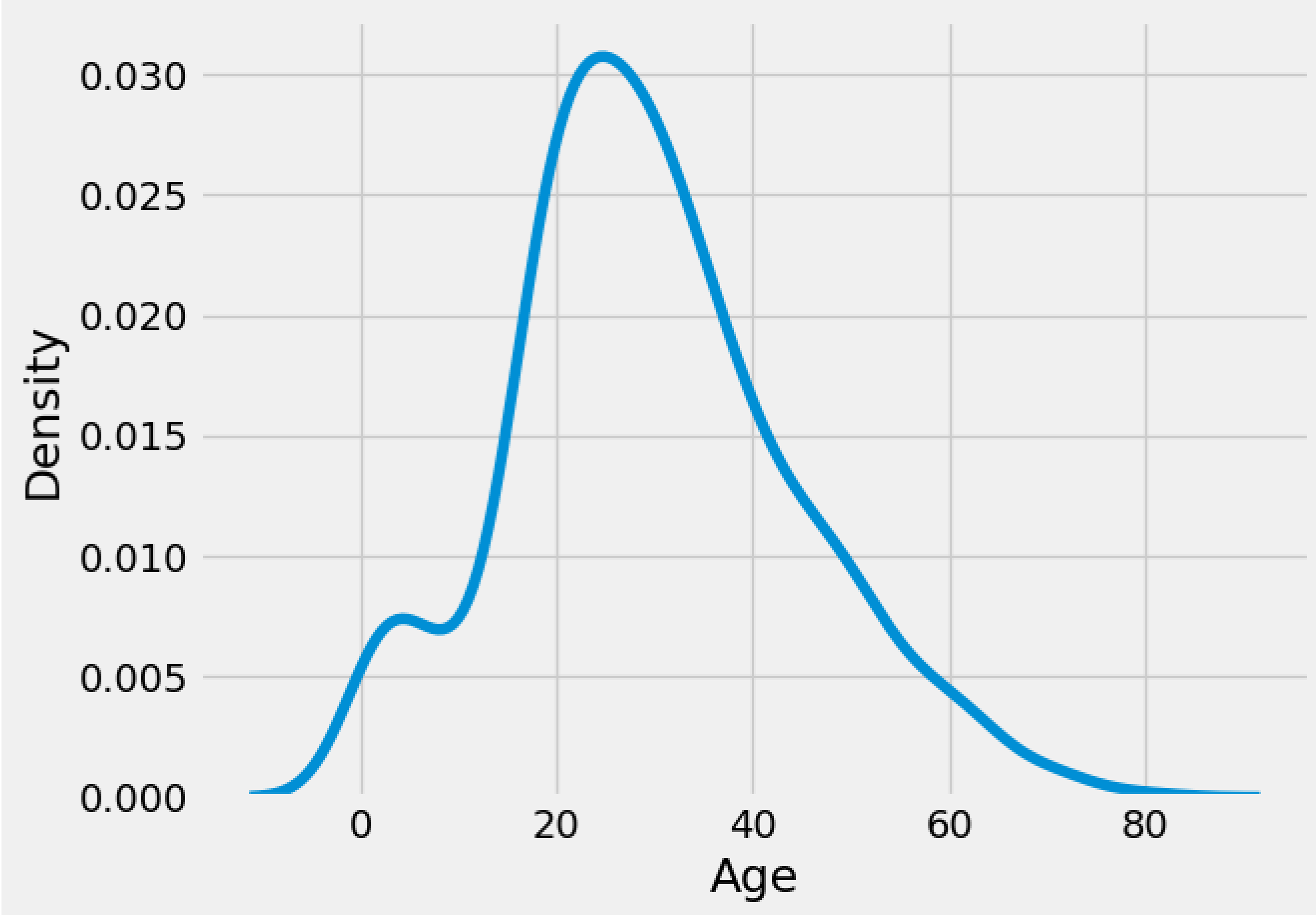
Out[10]:
```
count    714.000000
mean      29.699118
std       14.526497
min        0.420000
25%       20.125000
50%       28.000000
75%       38.000000
max       80.000000
Name: Age, dtype: float64
```

In [11]:
```
sns.histplot(data['Age'])
plt.show()
```

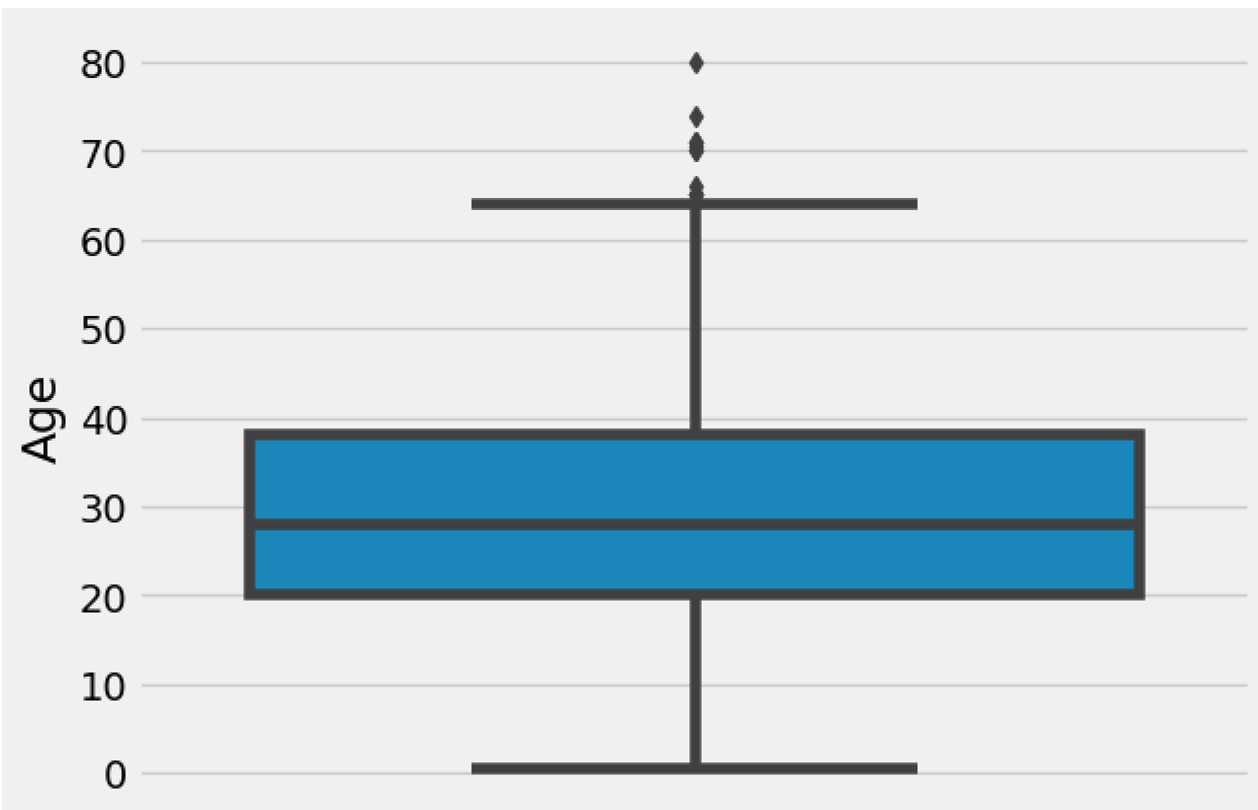


In [12]:
```
sns.distplot(data['Age'],hist=False)
plt.show()
```

Loading [MathJax]/extensions/Safe.js



In [13]:
```
sns.boxplot(data=data,y='Age')
plt.show()
```



In [14]:
```
data[data['Age']>65]
```

Out[14]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
33	34	0	2	Wheadon, Mr. Edward H	male	66.0	0	0	C.A. 24579	10.5000	NaN	S
96	97	0	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.6542	A5	C
116	117	0	3	Connors, Mr. Patrick	male	70.5	0	0	370369	7.7500	NaN	Q
493	494	0	1	Artagaveytia, Mr. Ramon	male	71.0	0	0	PC 17609	49.5042	NaN	C
630	631	1	1	Barkworth, Mr. Algernon Henry Wilson	male	80.0	0	0	27042	30.0000	A23	S
672	673	0	2	Mitchell, Mr. Henry Michael	male	70.0	0	0	C.A. 24580	10.5000	NaN	S
745	746	0	1	Crosby, Capt. Edward Gifford	male	70.0	1	1	WE/P 5735	71.0000	B22	S
851	852	0	3	Svensson, Mr. Johan	male	74.0	0	0	347060	7.7750	NaN	S

In [15]:

```
data['Age'].skew()
```

Out[15]:

```
0.38910778230082704
```

In [16]:

```
(data['Age'].isnull().sum()/data['Age'].shape[0])*100
```

Out[16]:

```
19.865319865319865
```

Fare:-

Conclusions:-

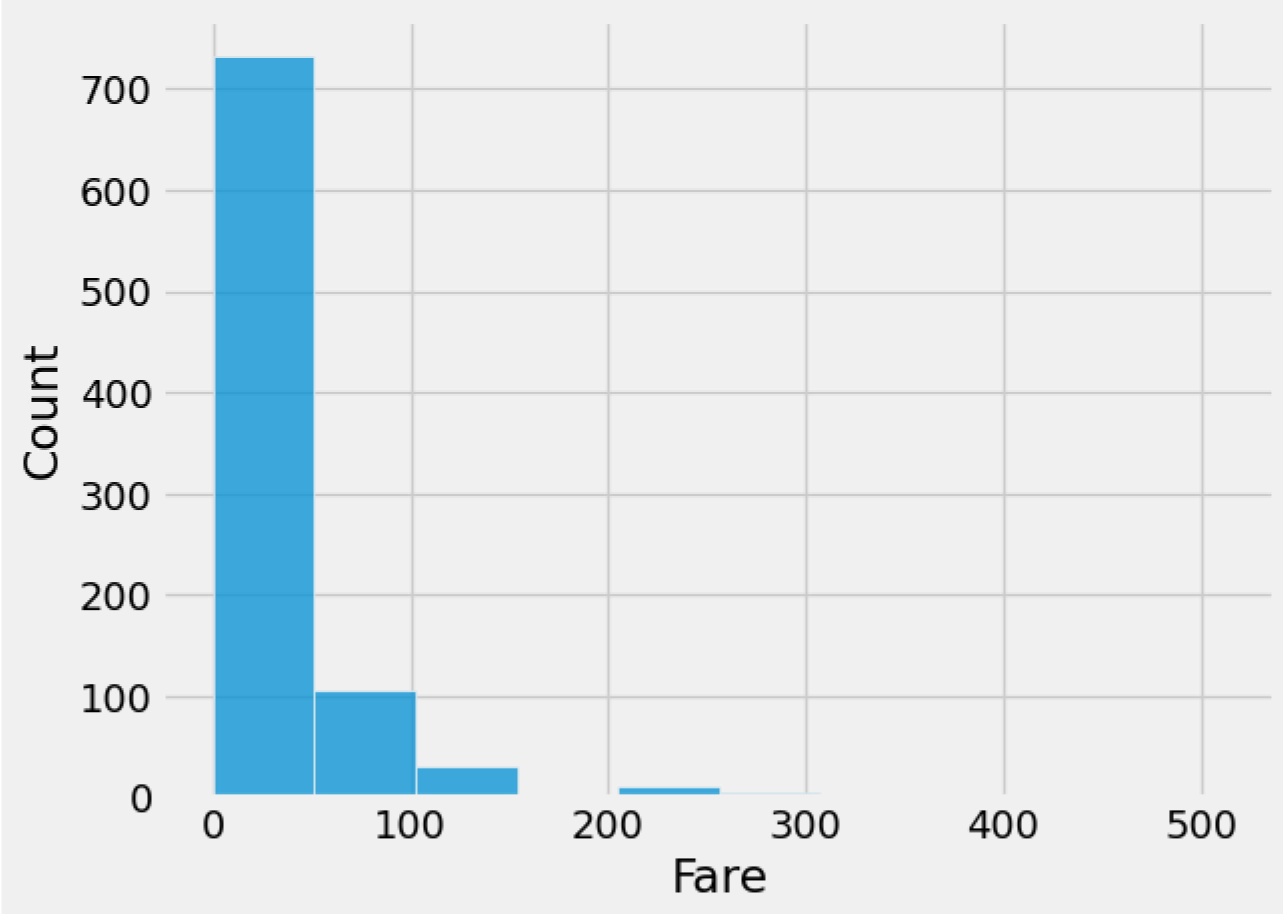
1). Fare is right skeward

2). Lot's of outliers

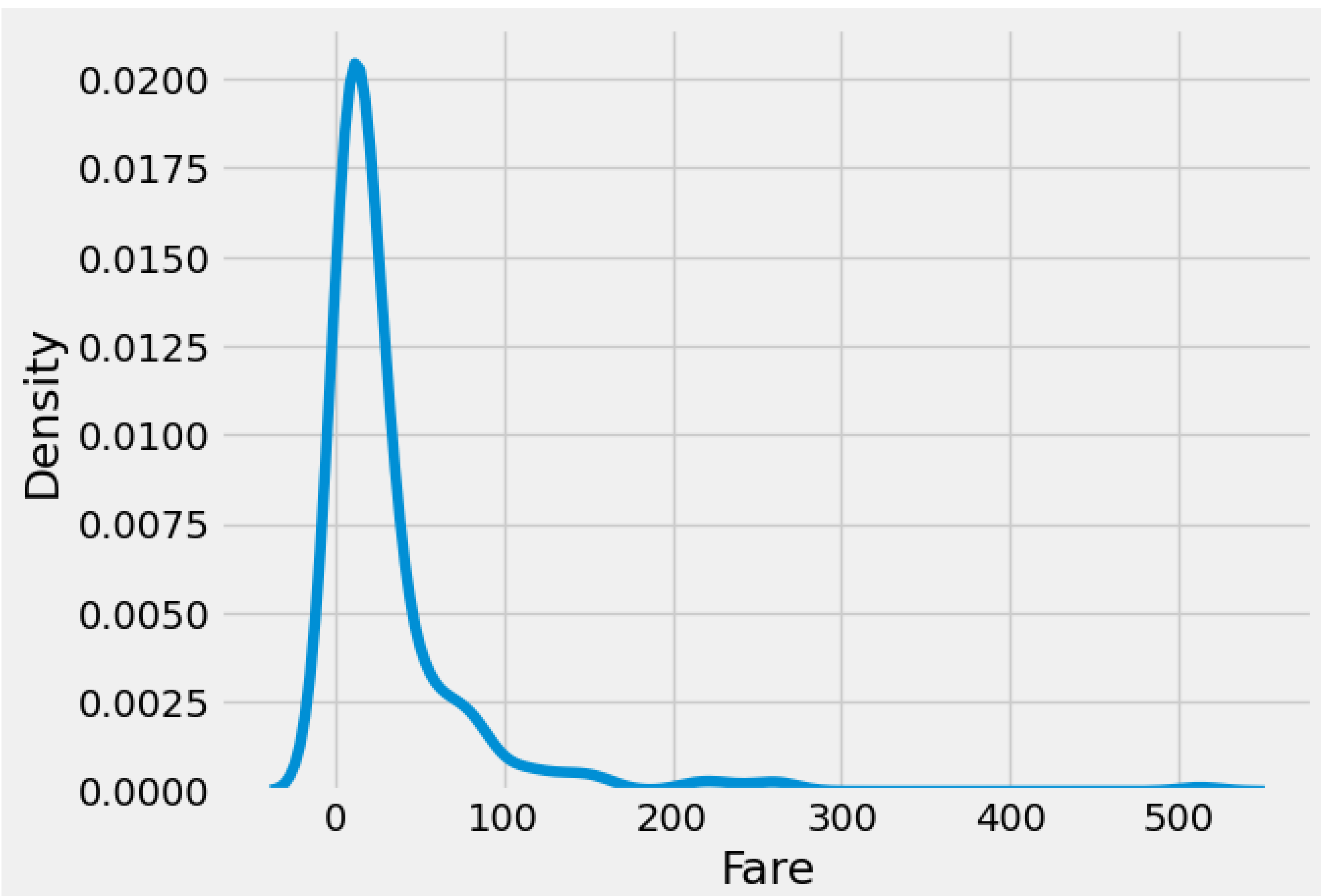
3). we can apply some transformation (feature engineering) becouse fare column is right skeward

In [17]:

```
sns.histplot(data['Fare'],bins=10)
plt.show()
```



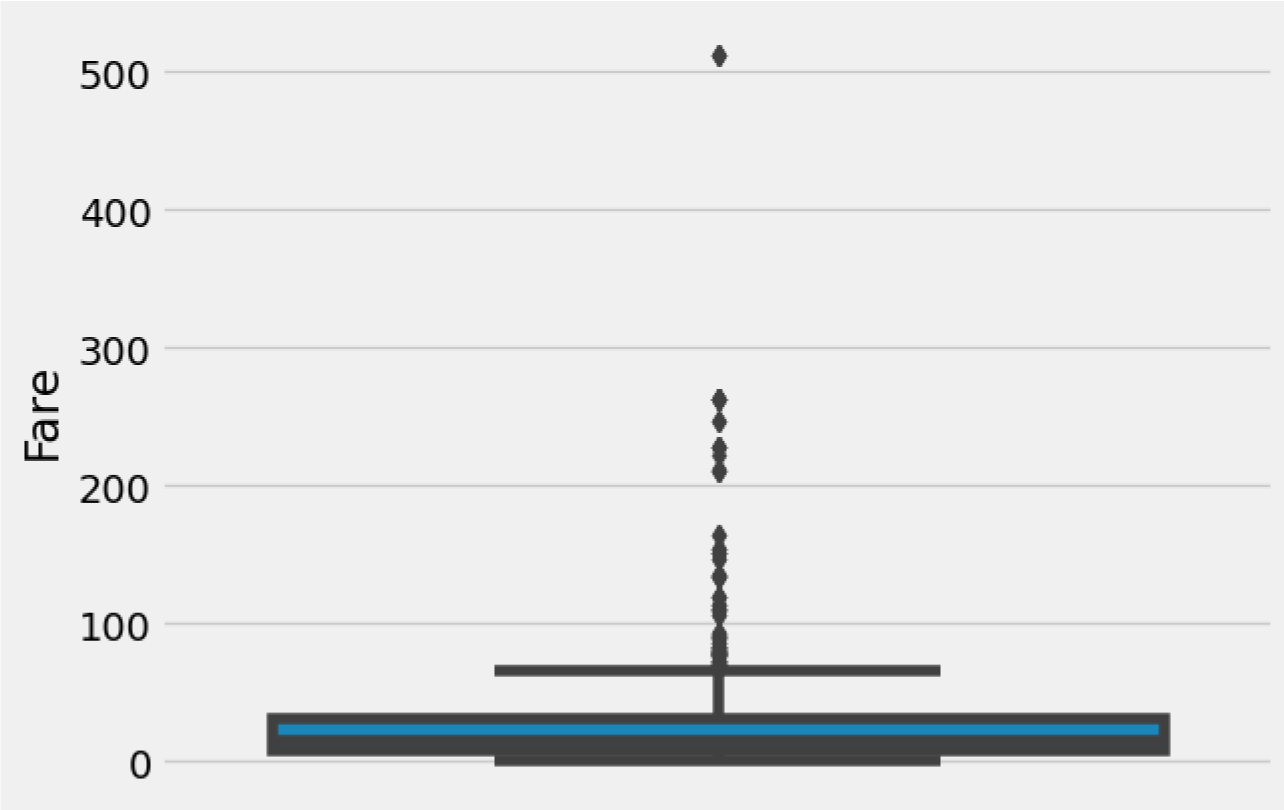
In [18]:
```
sns.distplot(data['Fare'],hist=False)
plt.show()
```



In [19]:
```
data['Fare'].skew()
```

Out[19]: 4.787316519674893

In [20]:
```
sns.boxplot(data=data,y='Fare')
plt.show()
```



In [21]:
```
data['Fare'].isnull().sum()
```

Out[21]: 0

In [22]:
```
data[data['Fare']>250]
```

Out[22]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000	C23 C25 C27	S
88	89	1	1	Fortune, Miss. Mabel Helen	female	23.0	3	2	19950	263.0000	C23 C25 C27	S
258	259	1	1	Ward, Miss. Anna	female	35.0	0	0	PC 17755	512.3292	NaN	C
311	312	1	1	Ryerson, Miss. Emily Borie	female	18.0	2	2	PC 17608	262.3750	B57 B59 B63 B66	C
341	342	1	1	Fortune, Miss. Alice Elizabeth	female	24.0	3	2	19950	263.0000	C23 C25 C27	S
438	439	0	1	Fortune, Mr. Mark	male	64.0	1	4	19950	263.0000	C23 C25 C27	S
679	680	1	1	Cardeza, Mr. Thomas Drake Martinez	male	36.0	0	1	PC 17755	512.3292	B51 B53 B55	C
737	738	1	1	Lesurer, Mr. Gustave J	male	35.0	0	0	PC 17755	512.3292	B101	C
742	743	1	1	Ryerson, Miss. Susan Parker "Suzette"	female	21.0	2	2	PC 17608	262.3750	B57 B59 B63 B66	C

Univariate Analysis on Categorical columns

Survived:-

1) survival rate of 0 is high campare to 1

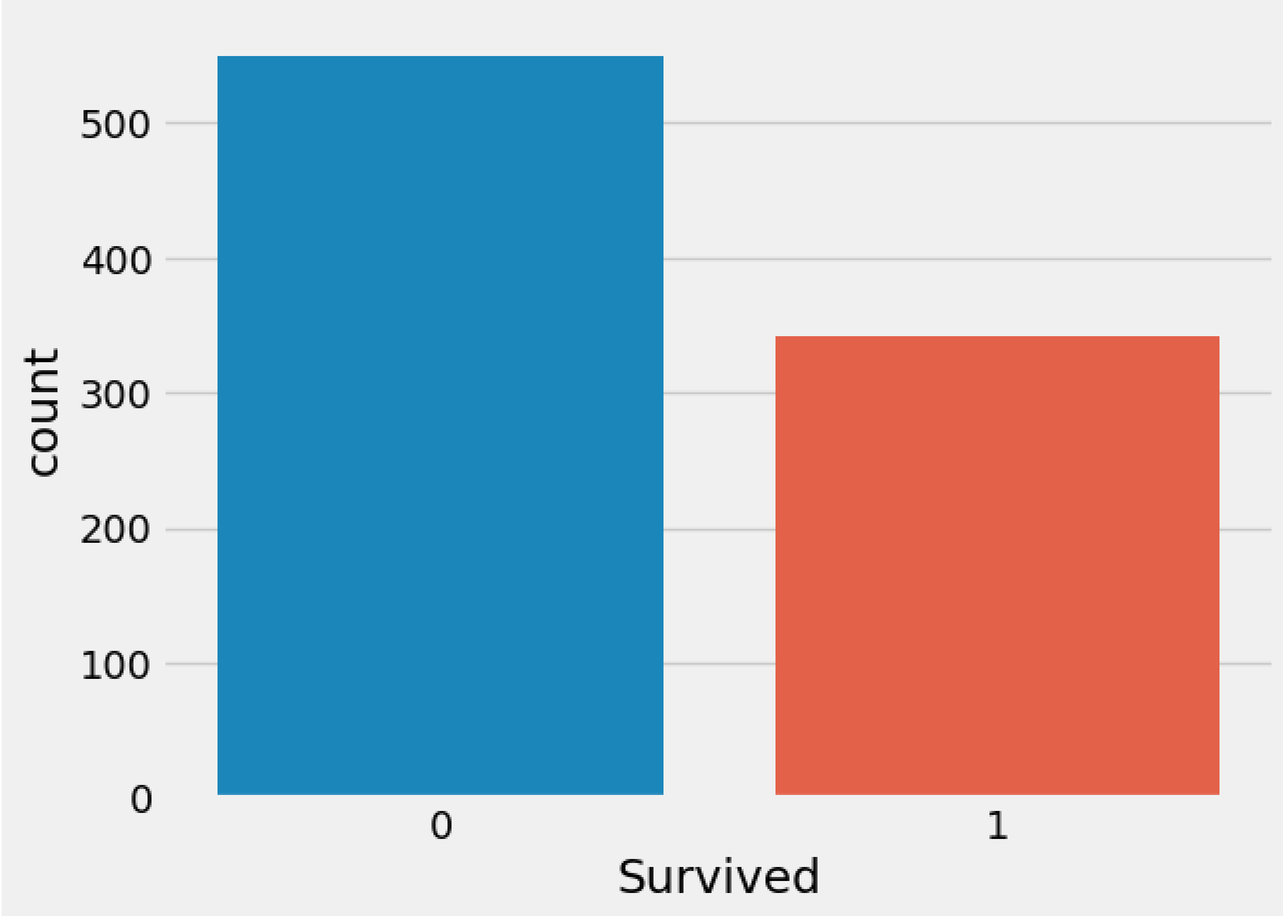
2) class is not inbalnced

In [23]:
```
data['Survived'].value_counts()/len(data['Survived'])*100
```

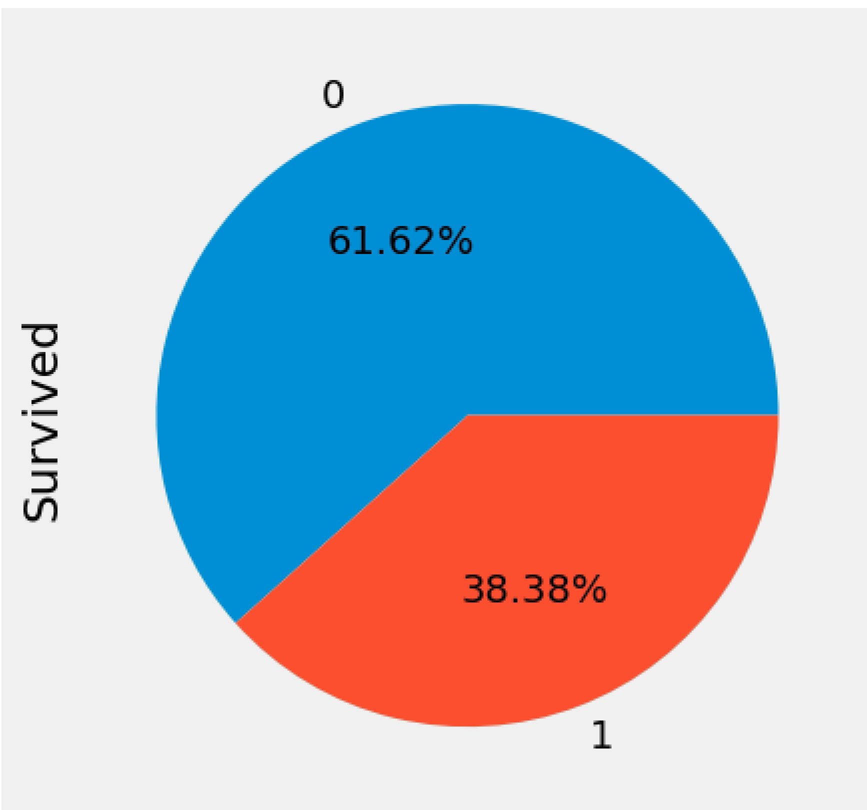
Out[23]:
```
0    61.616162
1    38.383838
Name: Survived, dtype: float64
```

In [24]:
```
sns.countplot(data['Survived'])
plt.show()
```

Loading [MathJax]/extensions/Safe.js



In [25]:
```
data['Survived'].value_counts().plot(kind='pie',autopct='%0.2f%%')
plt.show()
```



In [26]:
```
data['Survived'].isnull().sum()
```

Out[26]:
```
0
```

Pclass:

canclusion:- 1) 55% passenger were traveling in third class

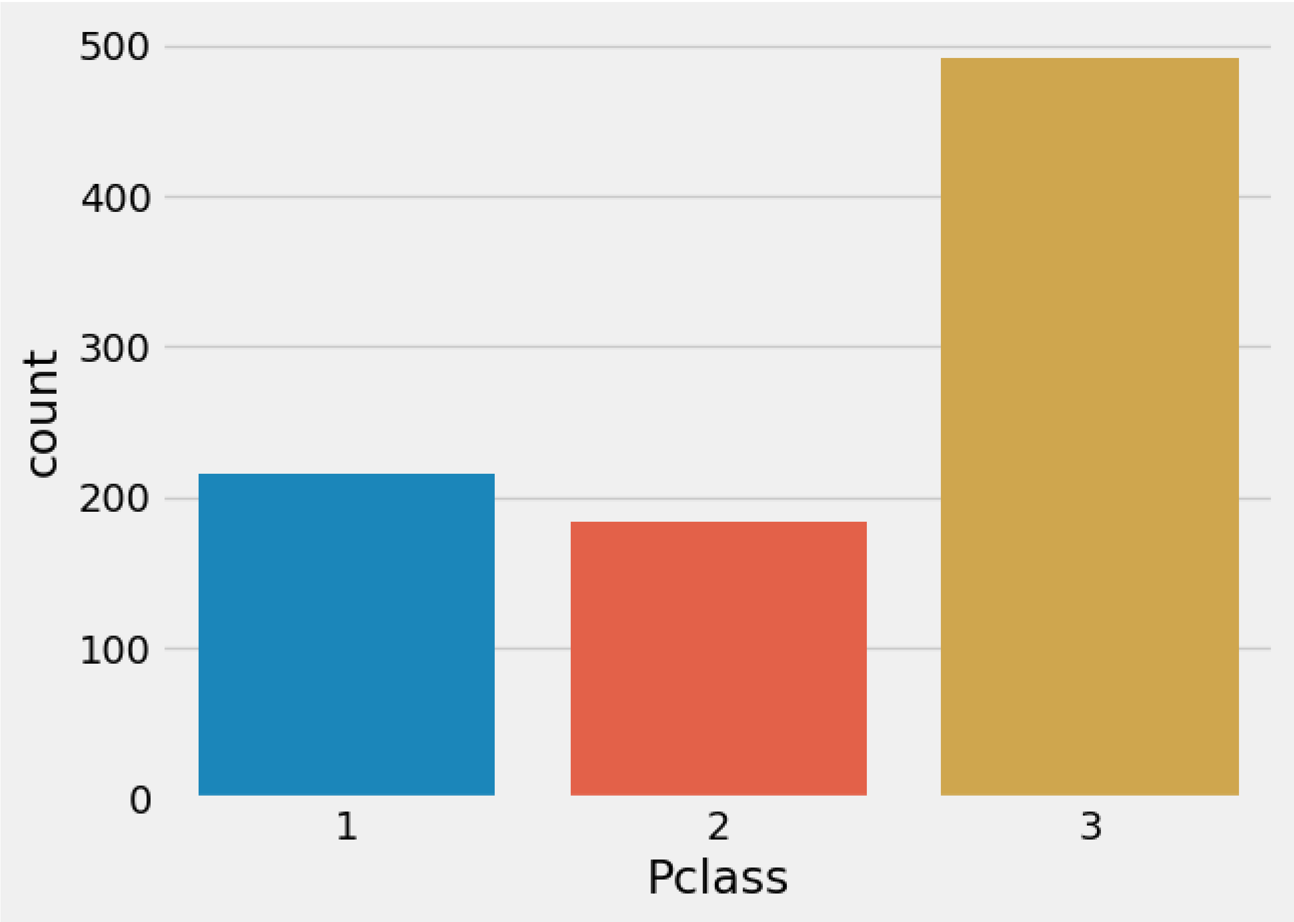
2) 24% passenger were travelling in second class

3) 20% passenger were travelling in first class

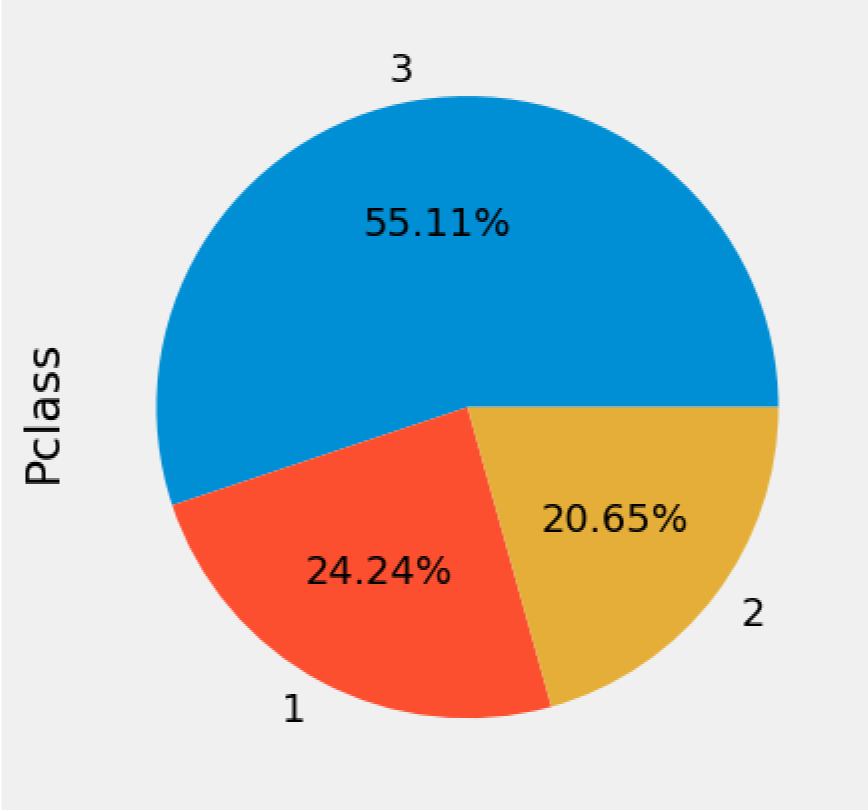
In [27]:
```
data['Pclass'].value_counts()
```

Out[27]:
```
3    491
1    216
2    184
Name: Pclass, dtype: int64
```

In [28]:
```
sns.countplot(data['Pclass'])
plt.show()
```



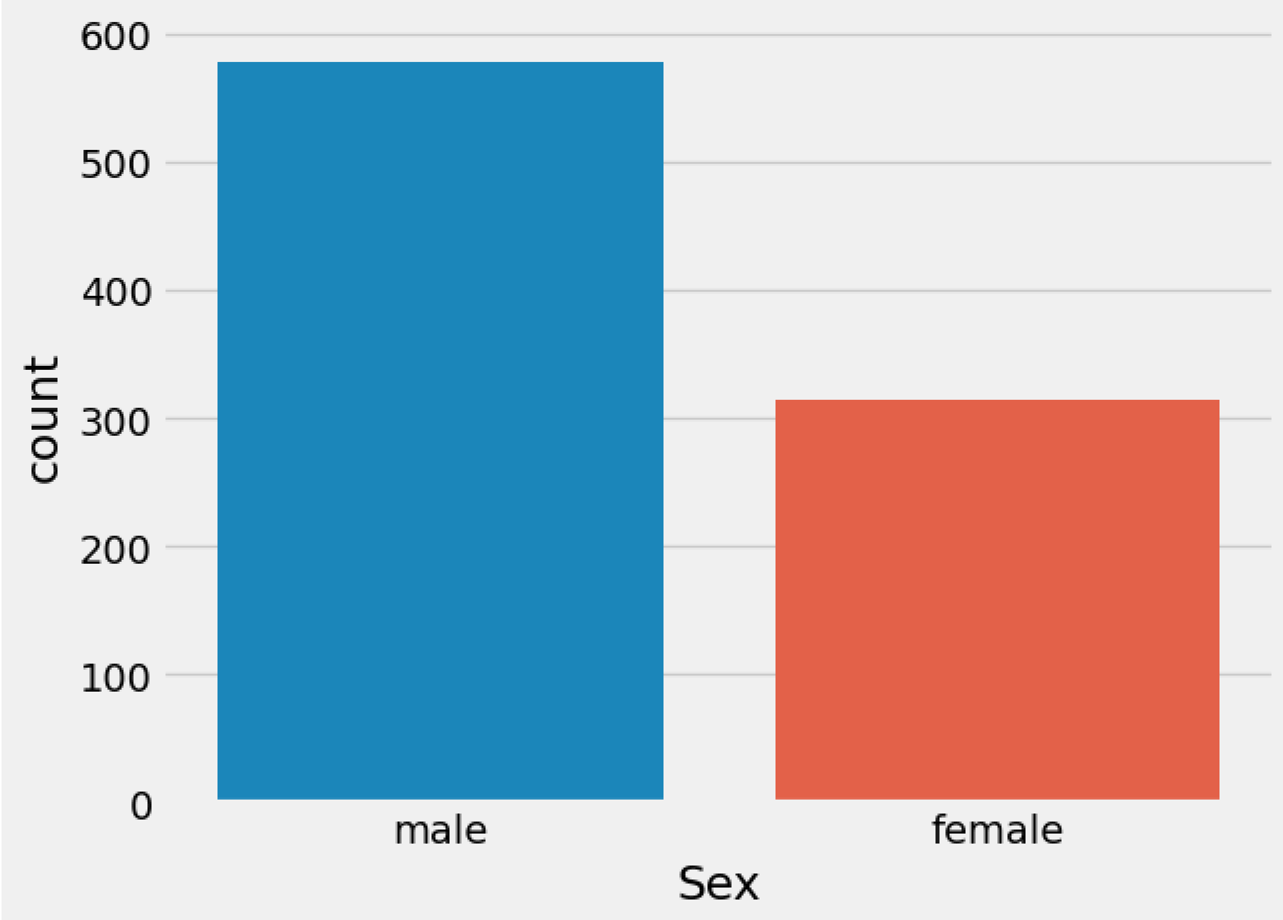
In [29]:
```
data['Pclass'].value_counts().plot(kind='pie',autopct='%0.2f%%')
plt.show()
```



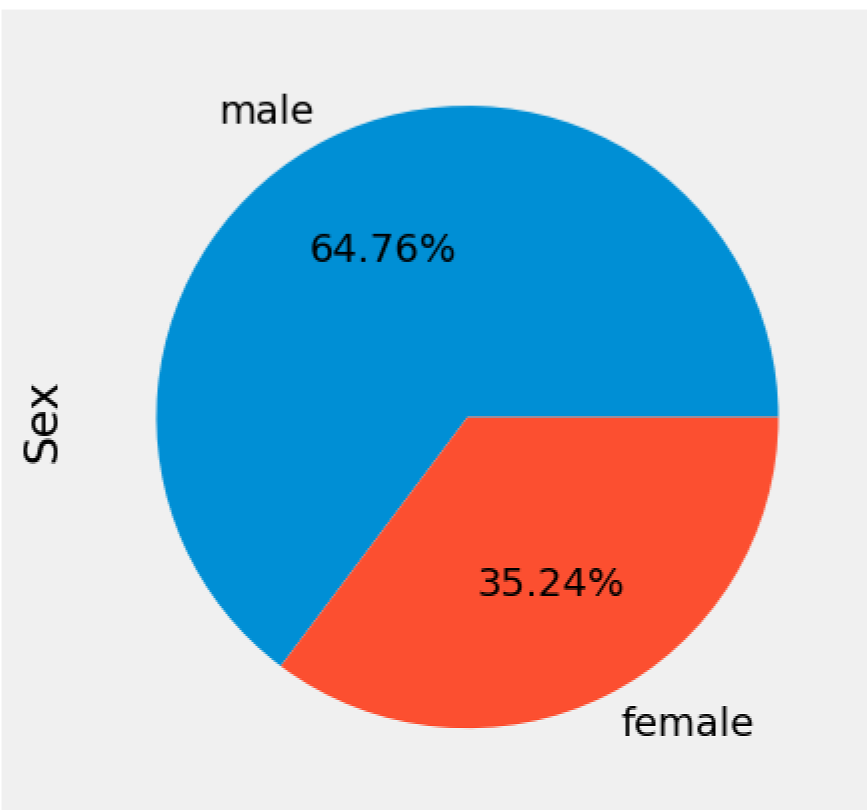
In [30]:
```
data['Pclass'].isnull().sum()
```
Out[30]:
```
0
```

In [31]:
```
data['Sex'].value_counts()
```
Out[31]:
```
male      577
female    314
Name: Sex, dtype: int64
```

In [32]:
```
sns.countplot(data['Sex'])
plt.show()
```



In [33]:
```
data['Sex'].value_counts().plot(kind='pie',autopct='%0.2f%%')
plt.show()
```



In [34]:
```
data['Sex'].isnull().sum()
```

Out[34]:
```
0
```

Loading [MathJax]/extensions/Safe.js

Embarked:-

conclusion:-

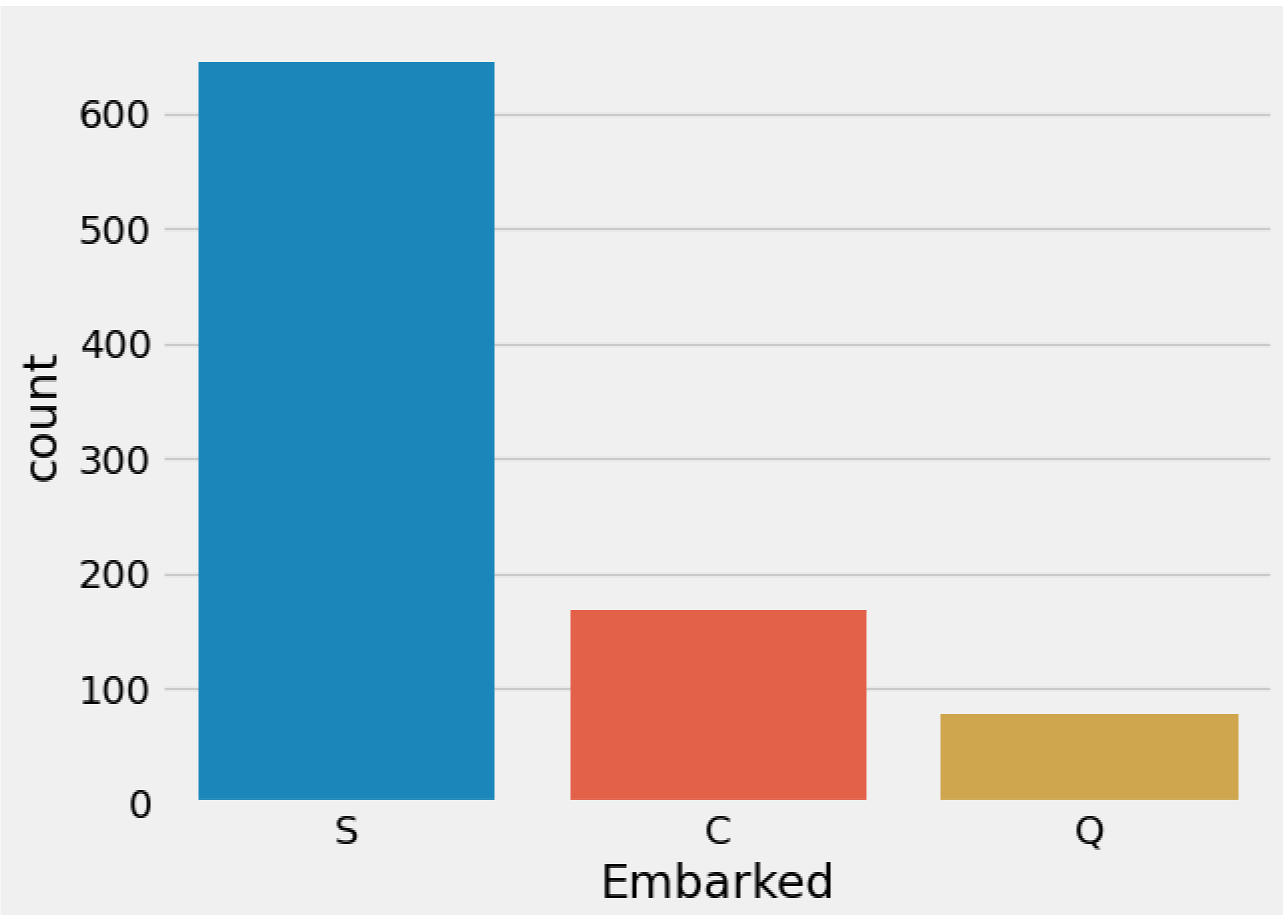
S :- Highest passenger travling from s

2 missing values

In [35]:
```
data['Embarked'].value_counts()
```

Out[35]:
```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [36]:
```
sns.countplot(data['Embarked'])
plt.show()
```



In [37]:
```
data['Embarked'].isnull().sum()
```

Out[37]:
```
2
```

Bivariate Analysis

Categorical vs Categorical

Conclusion:-

Highest not Survivel rate was in Pclass 3

Highest Survivel rate was in Pclass 1

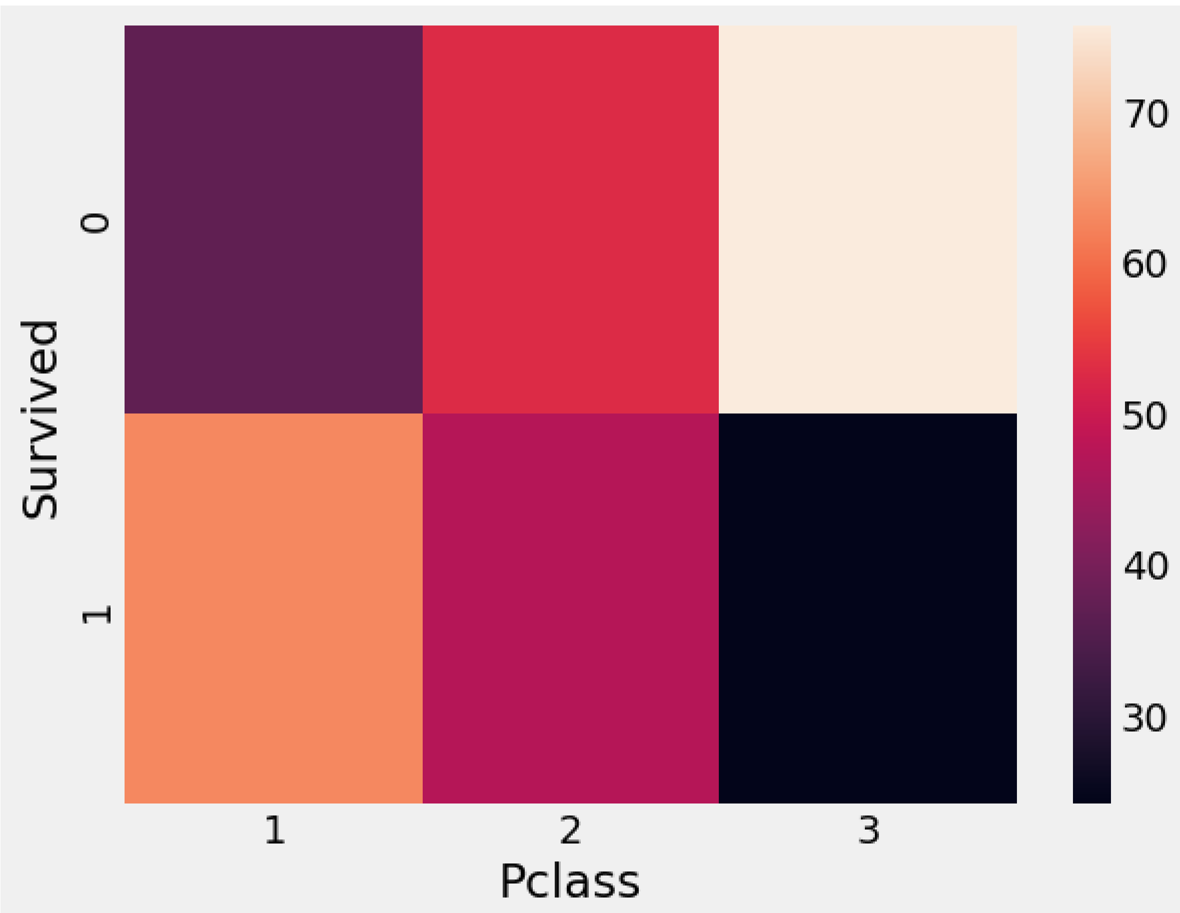
The Survival Rate of passengers from 1st class was highest. (The passengers from 1st and 2nd clsss were given priority while rescue)

In [38]:
```
pd.crosstab(data['Survived'],data['Pclass'],normalize='columns')*100
```

Out[38]:

Pclass	1	2	3
Survived			
0	37.037037	52.717391	75.763747
1	62.962963	47.282609	24.236253

In [39]:
```
sns.heatmap(pd.crosstab(data['Survived'],data['Pclass'],normalize='columns')*100)
plt.show()
```



In [40]:
```
sns.countplot(data['Pclass'],hue=data['Survived'])
plt.show()
```

In [41]:
```
p_s = data.groupby(by=['Pclass','Survived'])['Survived'].count()
```

In [42]:
```
print('1st class survived percentage:- %.2f%%'%(p_s[1][1]/(p_s[1][1]+p_s[1][0])*100))
print('2nd class survived percentage:- %.2f%%'%(p_s[2][1]/(p_s[2][1]+p_s[2][0])*100))
print('3rd class survived percentage:- %.2f%%'%(p_s[3][1]/(p_s[3][1]+p_s[3][0])*100))
```

```
1st class survived percentage:- 62.96%
2nd class survived percentage:- 47.28%
3rd class survived percentage:- 24.24%
```

Conclusion:-

1) The Survival Rate of Female passengers is higher as compared to male passengers (females were given priority while rescue)

In [43]:
```
pd.crosstab(data['Survived'],data['Sex'],normalize='columns')*100
```

Out[43]:

Sex	female	male
Survived		
0	25.796178	81.109185
1	74.203822	18.890815

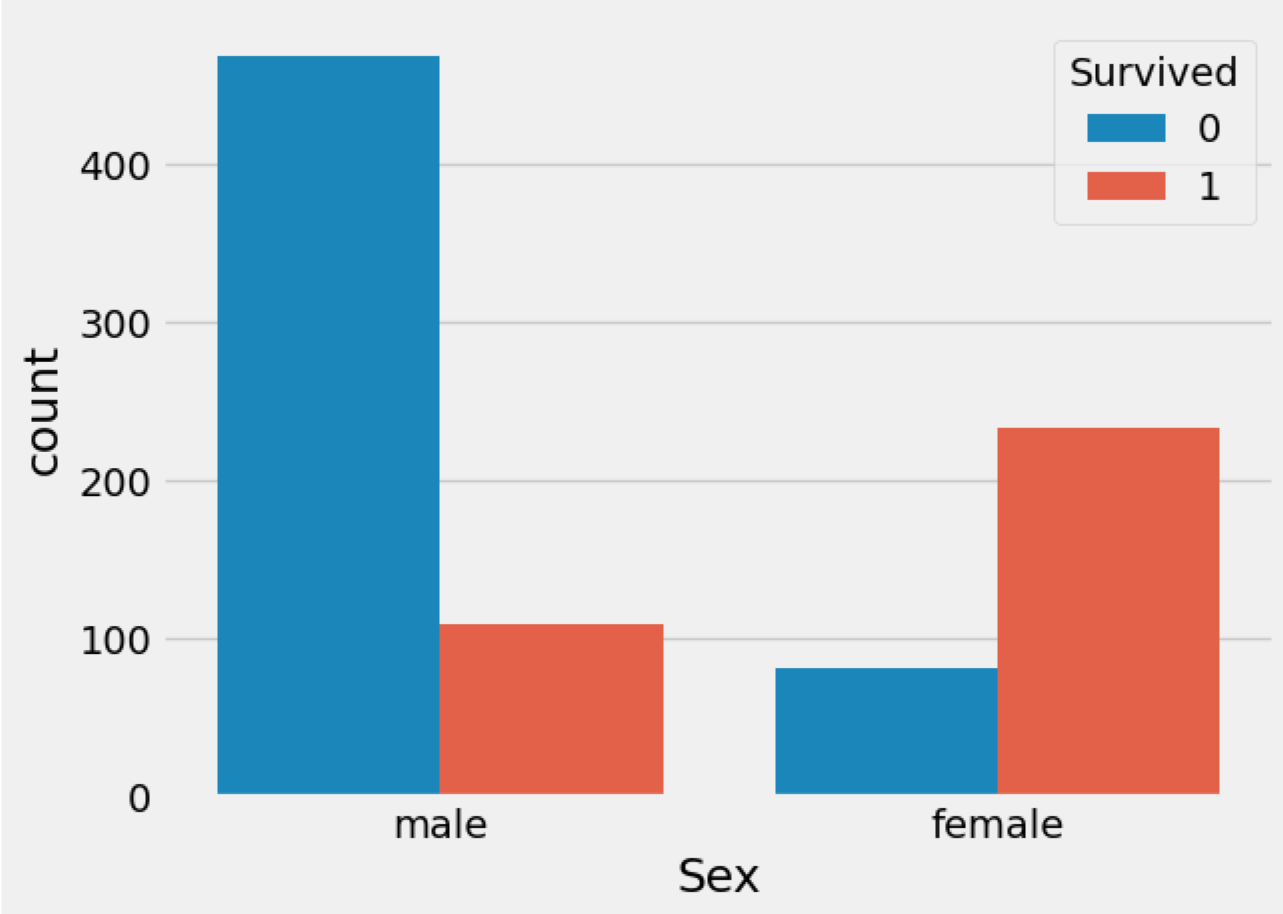
In [44]:
```
sns.heatmap(pd.crosstab(data['Survived'],data['Sex'],normalize='columns')*100)
plt.show()
```

In [45]:
```
data_sur = data.groupby(by=['Sex','Survived'])['Survived'].count()
```

In [46]:
```
print('Female Survived percentage:-%.2f%%'%(data_sur['female'][1]/(data_sur['female'][1]
print('Male Survived percentage:-%.2f%%'%(data_sur['male'][1]/(data_sur['male'][1]+data_
```

```
Female Survived percentage:-74.20%
Male Survived percentage:-18.89%
```

In [47]:
```
sns.countplot(data['Sex'],hue=data['Survived'])
plt.show()
```



Conclusion:-

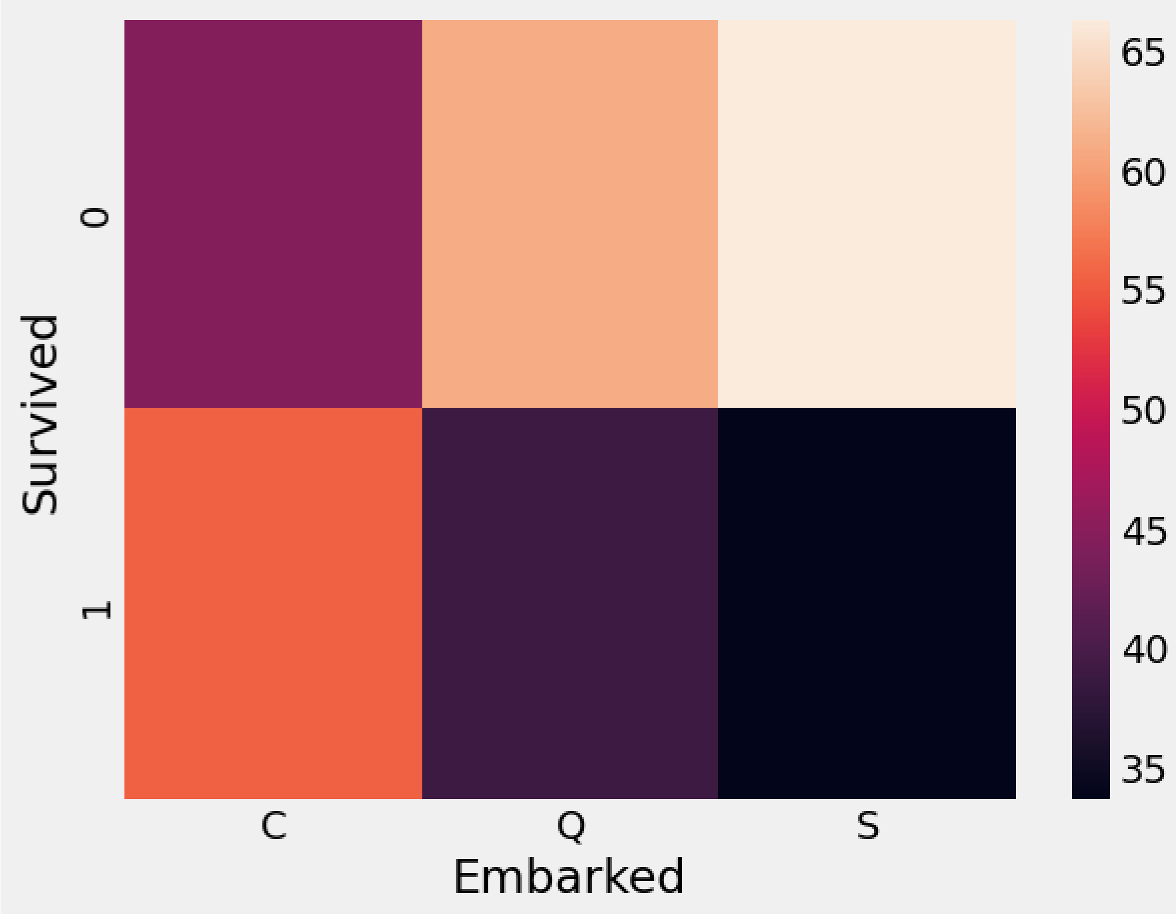
Although the Survival Rate for passengers boarded from chebourge was the highest while here more number of passengers from southampton we cannot say that was any priority on the basis of boarding station.

In [48]:
```
pd.crosstab(data['Survived'],data['Embarked'],normalize='columns')*100
```

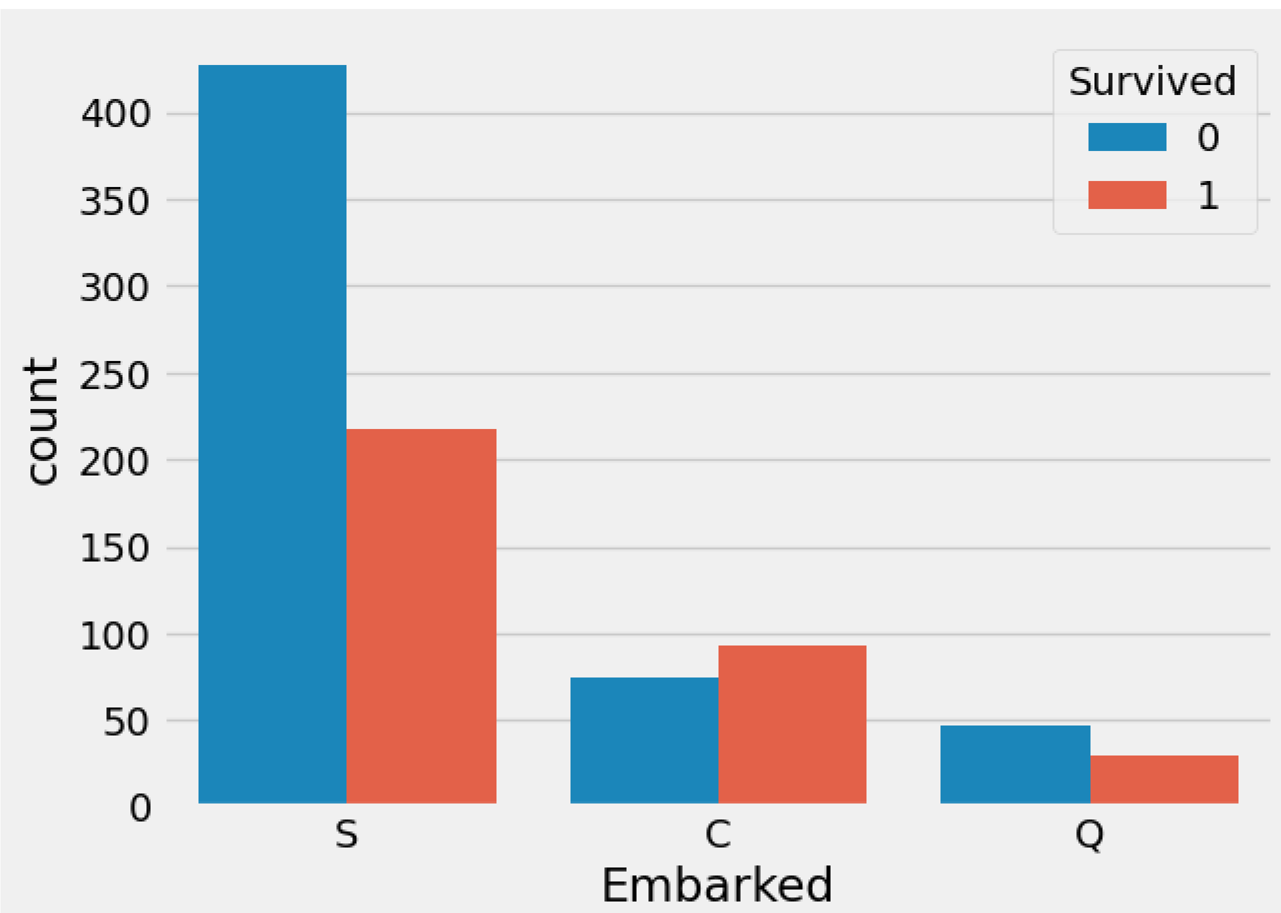
Out[48]:

Embarked	C	Q	S
Survived			
0	44.642857	61.038961	66.304348
1	55.357143	38.961039	33.695652

In [49]:
```
sns.heatmap(pd.crosstab(data['Survived'],data['Embarked'],normalize='columns')*100)
plt.show()
```



In [50]:
```
sns.countplot(data['Embarked'],hue=data['Survived'])
plt.show()
```



In [51]:
```
em_data = data.groupby(by=['Embarked','Survived'])['Survived'].count()
```

Loading [MathJax]/extensions/Safe.js

In [52]:
```
print('percentage of S:- %.2f%%'%(em_data['S'][1]/(em_data['S'][0]+em_data[0])*100))
print('percentage of Q:- %.2f%%'%(em_data['Q'][1]/(em_data[1]+em_data[0])*100))
print('pecentage of C:- %.2f%%'%(em_data['C'][1]/(em_data[1]+em_data[0])*100))
```

```
percentage of S:- 43.23%
percentage of Q:- 17.86%
pecentage of C:- 55.36%
```

Categorical Vs Numerical

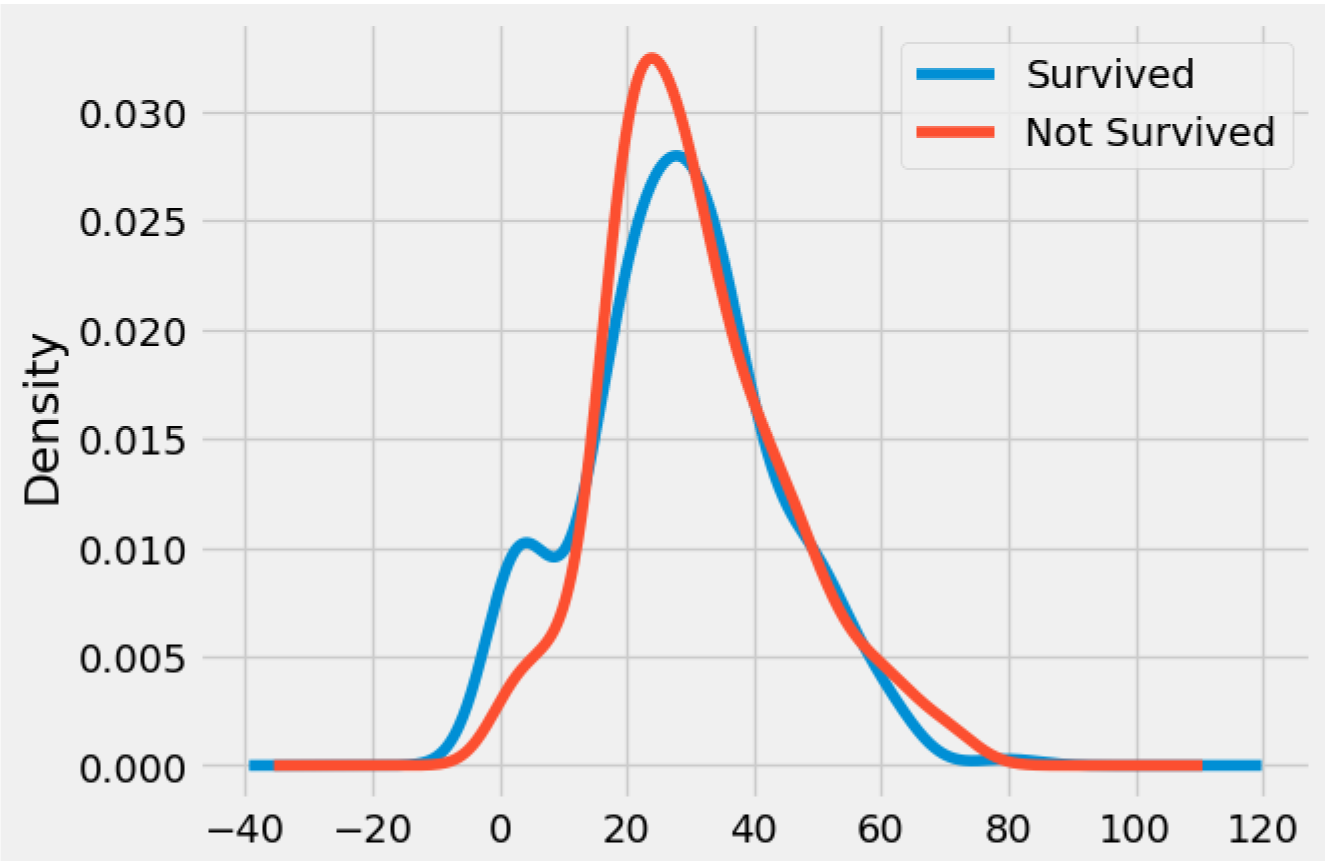
Conclusion:-

Those passengers whose age was between 0 to 20 were saved

An attempt was made to save the child

In [53]:
```
data[data['Survived'] == 1]['Age'].plot(kind='kde',label='Survived')
data[data['Survived'] == 0]['Age'].plot(kind='kde',label='Not Survived')

plt.legend()
plt.show()
```



In [54]:
```
print('Age of average Pclass 1:-',round(data[data['Pclass'] == 1]['Age'].mean(),2))
```

```
Age of average Pclass 1:- 38.23
```

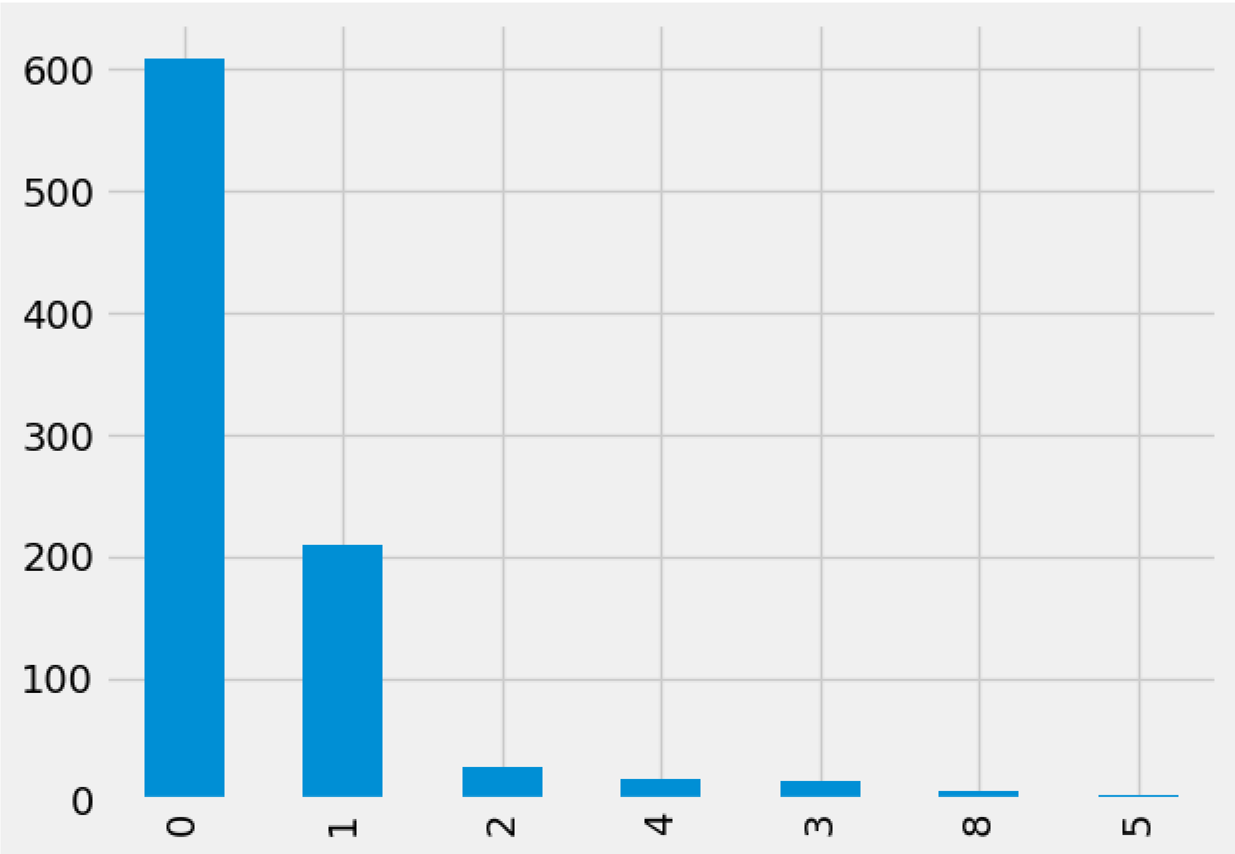
Feature Engineering

Loading [MathJax]/extensions/Safe.js

Conclusion:-

mostly pessenger were travelling alone

In [55]:
```
data['SibSp'].value_counts().plot(kind='bar')
plt.show()
```



Feature Engineering

Handle messy features

Ticket and Cabin column both in mixed data

Handle mixed data usning lambda function with split function

And change datatype of ticket and Cabin column

In [56]:
```
data.head(2)
```

Loading [MathJax]/extensions/Safe.js

Out[56]:

	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

In [57]:
```
data['Surname'] = data['Name'].apply(lambda x:x.split(',')[0])
```

In [58]:
```
data['Salutaion'] = data['Name'].apply(lambda x:x.split(',')[1]).apply(lambda x:x.strip(
```

In [59]:
```
data['Num_ticket'] = data['Ticket'].apply(lambda s: s.split()[-1])
```

In [60]:
```
print('Type of num_ticket column:-\n',data['Num_ticket'].dtypes)
```

```
Type of num_ticket column:-
 object
```

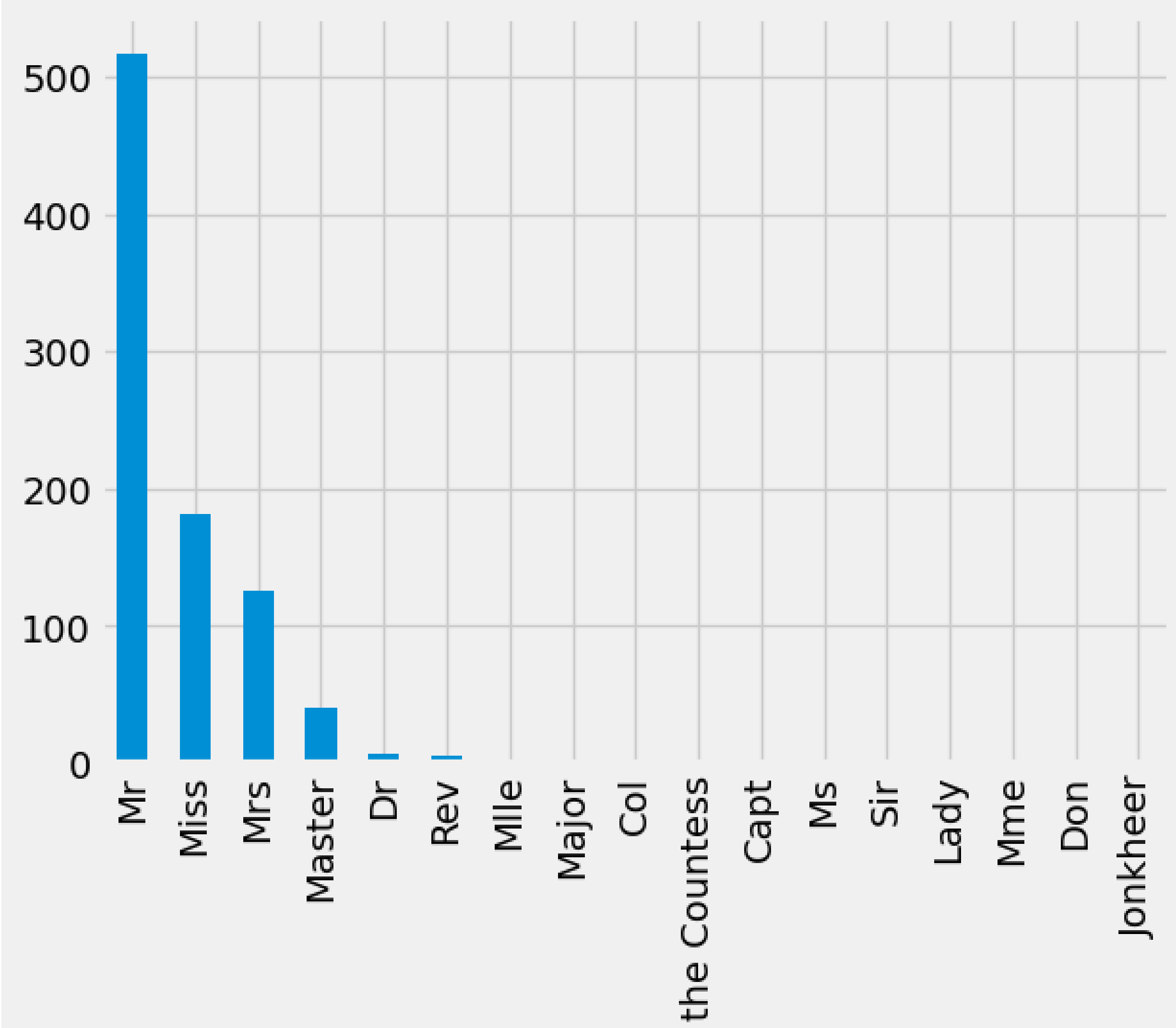
In [61]:
```
data['Num_ticket'] = pd.to_numeric(data['Num_ticket'],errors='coerce',
                                   downcast='integer')
```

In [62]:
```
data.head(1)
```

Out[62]:

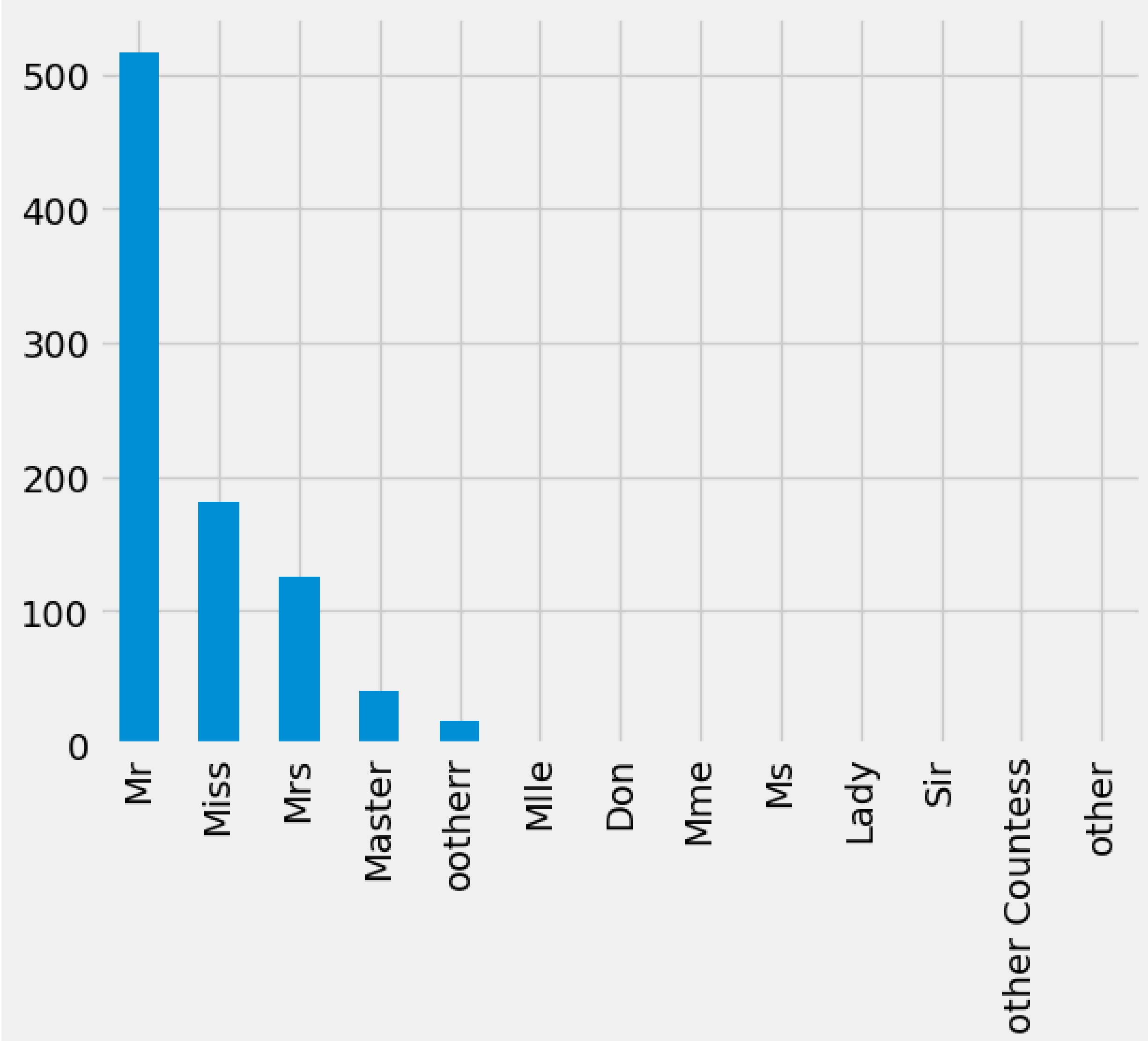
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Surname
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	Braund

In [63]:
```
data['Salutaion'].value_counts().plot(kind='bar')
plt.show()
```



In [64]:
```
data['Salutaion'] = data['Salutaion'].str.replace('Rev','other')
data['Salutaion'] = data['Salutaion'].str.replace('Dr','other')
data['Salutaion'] = data['Salutaion'].str.replace('Col','other')
data['Salutaion'] = data['Salutaion'].str.replace('Major','other')
data['Salutaion'] = data['Salutaion'].str.replace('Capt','other')
data['Salutaion'] = data['Salutaion'].str.replace('the','other')
data['Salutaion']= data['Salutaion'].str.replace('Jonkheer','other')
```

In [65]:
```
data['Salutaion'].value_counts().plot(kind='bar')
plt.show()
```



In [66]:
```
temp_df = data[data['Salutaion'].isin(['Mr','Miss','Mrs','Master','ootherr'])]
```

In [67]:
```
pd.crosstab(temp_df['Survived'],temp_df['Salutaion'],normalize='columns')*100
```

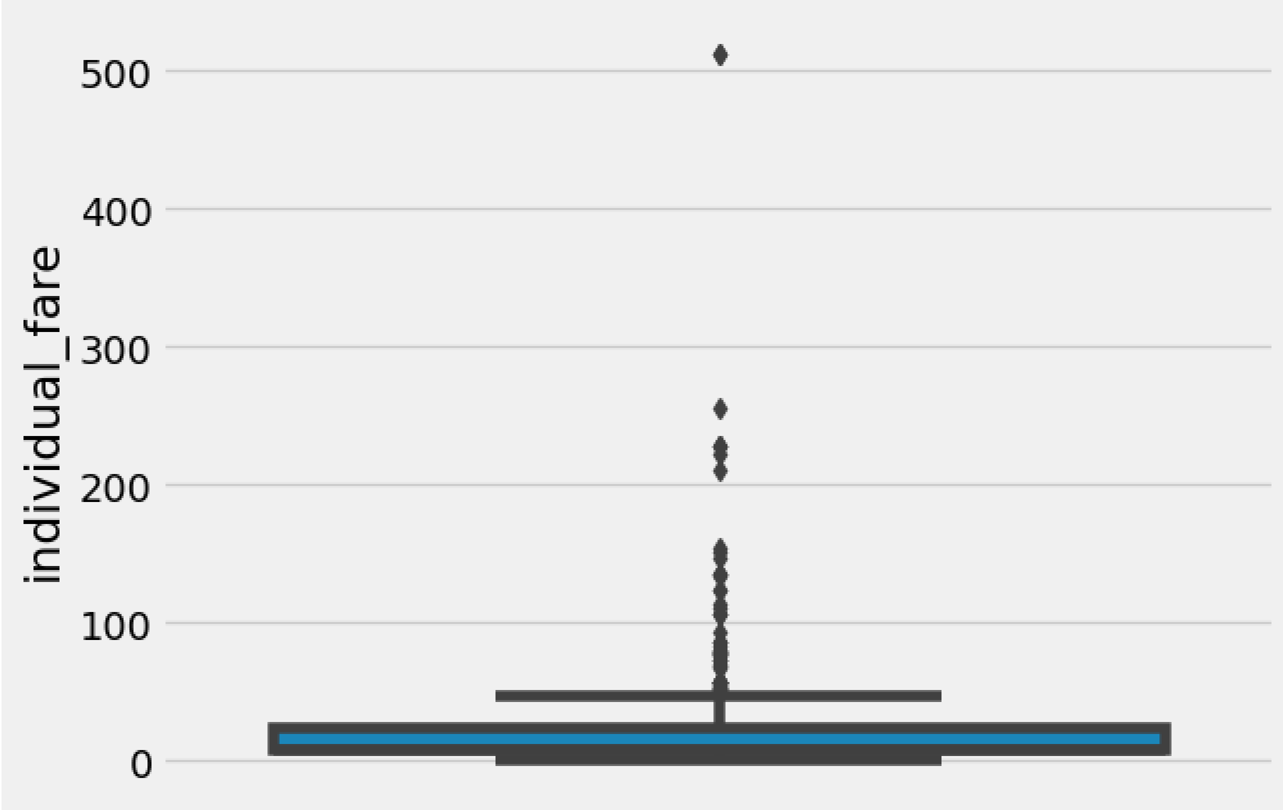
Out[67]:

Salutaion	Master	Miss	Mr	Mrs	ootherr
Survived					
0	42.5	30.21978	84.332689	20.8	72.222222
1	57.5	69.78022	15.667311	79.2	27.777778

In [68]:
```
data['ticket_cat'] = data['Ticket'].apply(lambda s: s.split()[0])
data['ticket_cat'] = np.where(data['ticket_cat'].str.isdigit(), np.nan,
                              data['ticket_cat'])
```

In [69]:
```
data['individual_fare'] = data['Fare']/(data['SibSp'] + data['Parch'] + 1)
```

In [70]:
```
sns.boxplot(data=data,y='individual_fare')
plt.show()
```



In [71]:
```
data[['individual_fare','Fare']].describe()
```

Out[71]:

	individual_fare	Fare
count	891.000000	891.000000
mean	19.916375	32.204208
std	35.841257	49.693429
min	0.000000	0.000000
25%	7.250000	7.910400
50%	8.300000	14.454200
75%	23.666667	31.000000
max	512.329200	512.329200

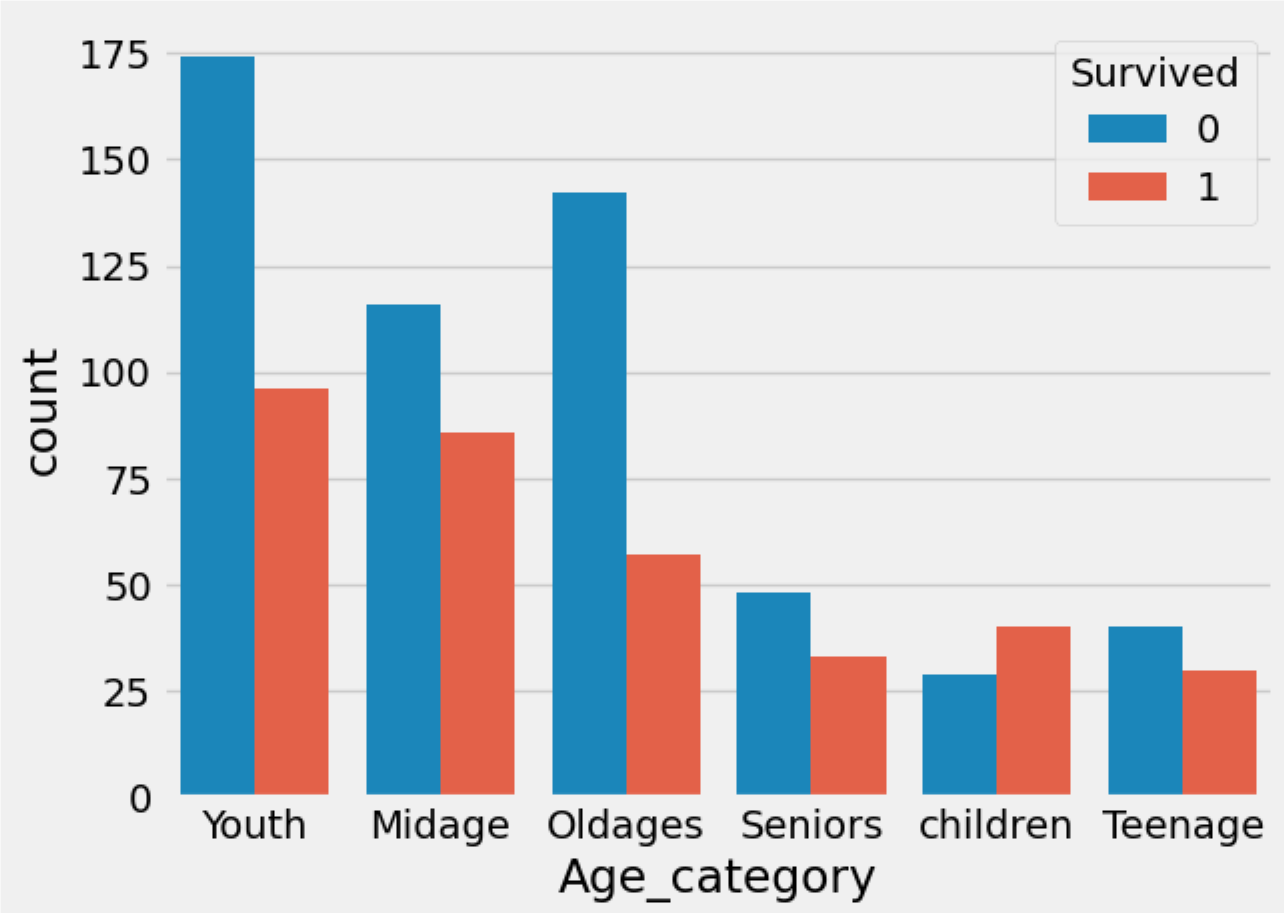
Conclusion:-

Children, Teenagers, and Senior citizens were given priority while rescue.

In [72]:
```
def age_category(age):
    if age <= 12:
        return 'children'
    elif age > 12 and age <=18:
        return 'Teenage'
    elif age > 18 and age <= 30:
        return 'Youth'
    elif age > 30 and age <=45:
        return 'Midage'
    elif age > 45 and age <=60:
        return 'Seniors'
    else:
        return 'Oldages'
```

In [73]:
```
data['Age_category'] = data['Age'].apply(age_category)
```

In [74]:
```
sns.countplot(data['Age_category'],hue=data['Survived'])
plt.show()
```



In [75]:
```
data['Age_category'].value_counts()
```

Out[75]:
```
Youth       270
Midage      202
Oldages     199
Seniors      81
Teenage      70
children     69
Name: Age_category, dtype: int64
```

In [76]:
```
Age_sur = data.groupby(by=['Age_category','Survived'])['Survived'].count()
```

In [77]:
```
print('MidAge survived percentage:-%.2f%%'%(Age_sur['Midage'][1]/(Age_sur['Midage'][1]+A
print('Oldages survived percentage:-%.2f%%'%(Age_sur['Oldages'][1]/(Age_sur['Oldages'][1
print('Seniors survived percentage:-%.2f%%'%(Age_sur['Seniors'][1]/(Age_sur['Seniors'][1
print('Youth survived percentage:-%.2f%%'%(Age_sur['Youth'][1]/(Age_sur['Youth'][1]+Age_
print('children survived percentage:-%.2f%%'%(Age_sur['children'][1]/(Age_sur['children'
print('TeenAge survived percentage :-%.2f%%'%(Age_sur['Teenage'][1]/(Age_sur['Teenage'][
```

```
MidAge survived percentage:-42.57%
Oldages survived percentage:-28.64%
Seniors survived percentage:-40.74%
Youth survived percentage:-35.56%
children survived percentage:-57.97%
TeenAge survived percentage :-42.86%
```

In [78]:
```
pd.crosstab(data['Age_category'],data['Survived'])
```

Out[78]:

Survived	0	1
Age_category		
Midage	116	86
Oldages	142	57
Seniors	48	33
Teenage	40	30
Youth	174	96
children	29	40

Perform some statistical test

In [79]:
```
obvserved = pd.crosstab(data['Age_category'],data['Survived'])
```

In [80]:
```
# H0 --> null hypothesis --> The survived not depend on Age_category
# H1 --> Alternate hypothesis --> The survived is Depends upon the Agecategory
chi,p,dof,expected = chi2_contingency(obvserved)
print('chi_square:-',chi)
print('p_value:-',p)
if p<0.05:
    print('Accept the H1')
else:
    print('Accept the H0')
```

```
chi_square:- 22.371876386576556
p_value:- 0.00044486673473075885
Accept the H1
```

Conclusion:-

small family survivel rate was high campare to orthers

In [81]:
```
data['family_size'] = data['SibSp'] + data['Parch'] + 1
```

In [82]:
```
def transform_family_size(num):
    if num == 1:
        return 'Alone'
    elif num>1 and num <5:
        return 'Small'
    else:
        return "Large"
```

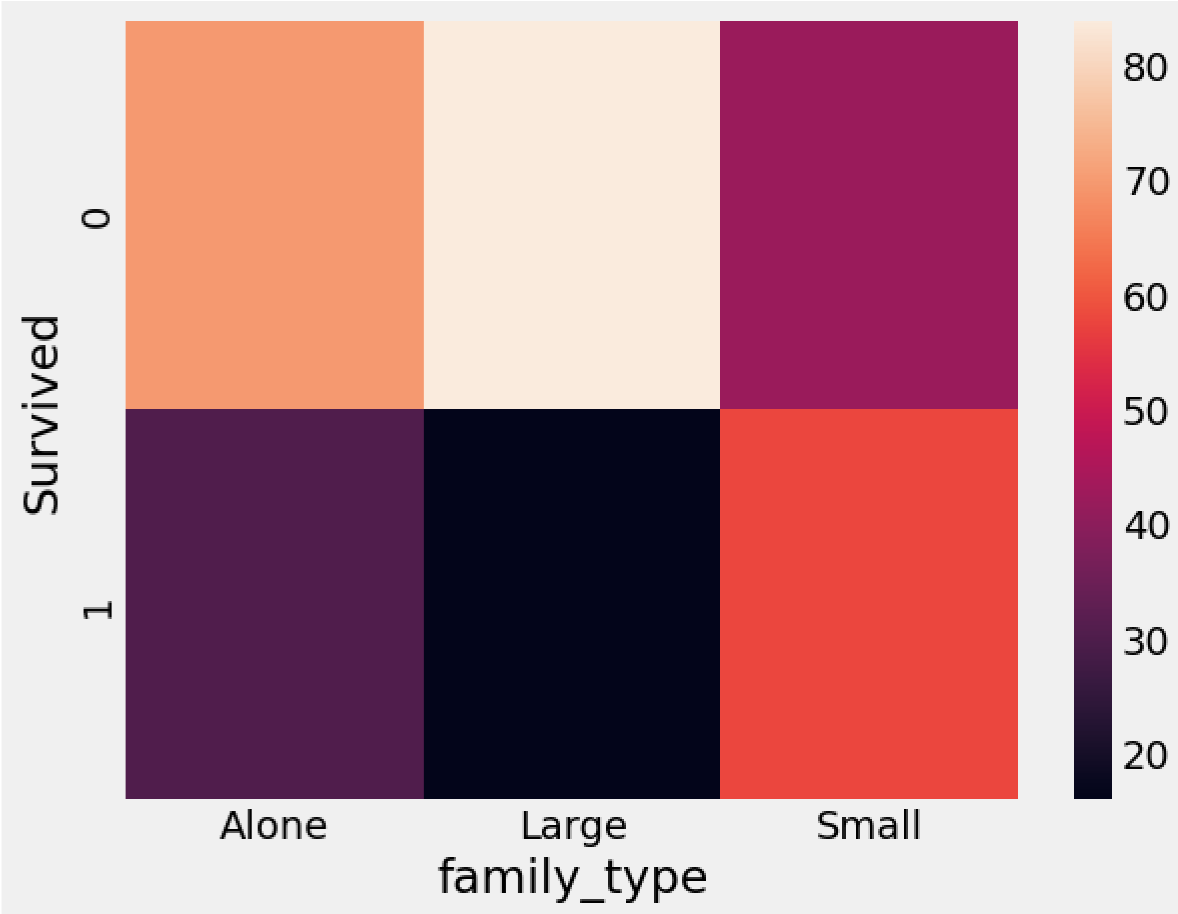
In [83]:
```
data['family_type'] = data['family_size'].apply(transform_family_size)
```

In [84]:
```
pd.crosstab(data['Survived'],data['family_type'],normalize='columns')*100
```

Out[84]:

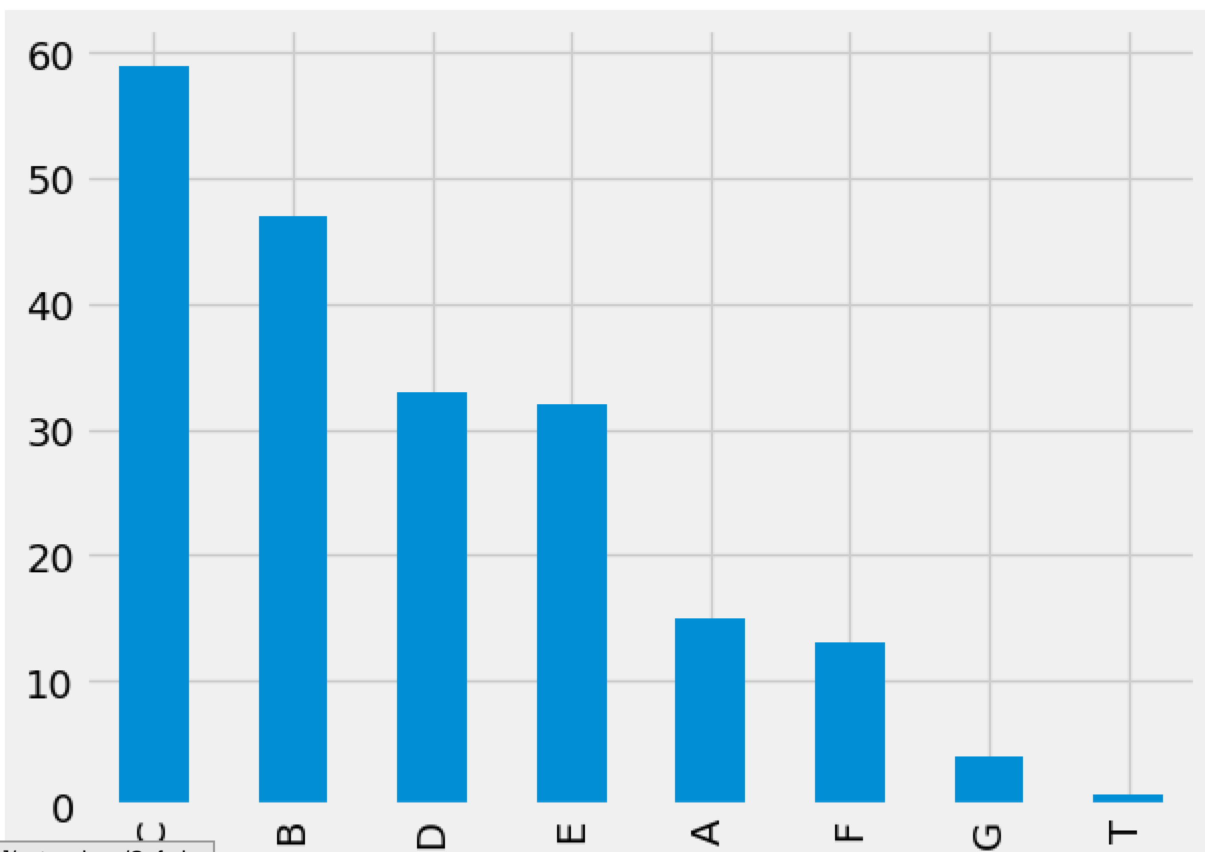
family_type	Alone	Large	Small
Survived			
0	69.646182	83.870968	42.123288
1	30.353818	16.129032	57.876712

In [85]:
```
sns.heatmap(pd.crosstab(data['Survived'],data['family_type'],normalize='columns')*100)
plt.show()
```



In [86]:
```
data['deck'] = data['Cabin'].str[0]
```

In [87]:
```
data['deck'].value_counts().plot(kind='bar')
plt.show()
```



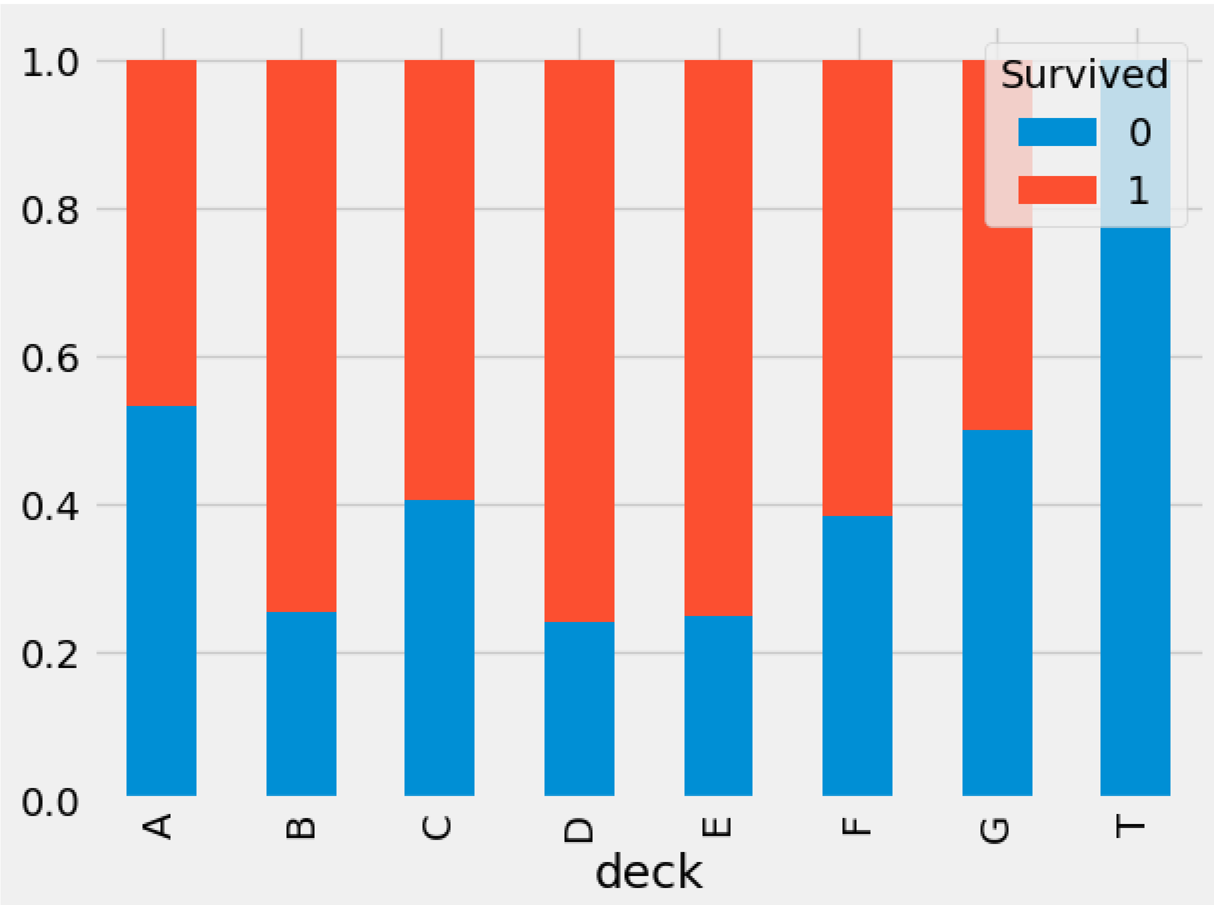
Loading [MathJax]/extensions/Safe.js

In [88]:
```
pd.crosstab(data['deck'],data['Pclass'])
```

Out[88]:

Pclass	1	2	3
deck			
A	15	0	0
B	47	0	0
C	59	0	0
D	29	4	0
E	25	4	3
F	0	8	5
G	0	0	4
T	1	0	0

In [89]:
```
pd.crosstab(data['deck'],data['Survived'],normalize='index').plot(kind='bar',stacked=Tru
plt.show()
```



Feture selection Manual

Drop all erelivent columns

In [90]:
```
data.head(2)
```

Out[90]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	...	Embarked	Surnam
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	...	S	Braur
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	...	C	Cuming

2 rows × 21 columns

In [91]:
```
data.drop(columns=['Ticket','Cabin','Name','ticket_cat','deck','family_type','Age_catego
                   'Num_ticket','family_size','PassengerId'],inplace=True)
```

Find coralation of all columns

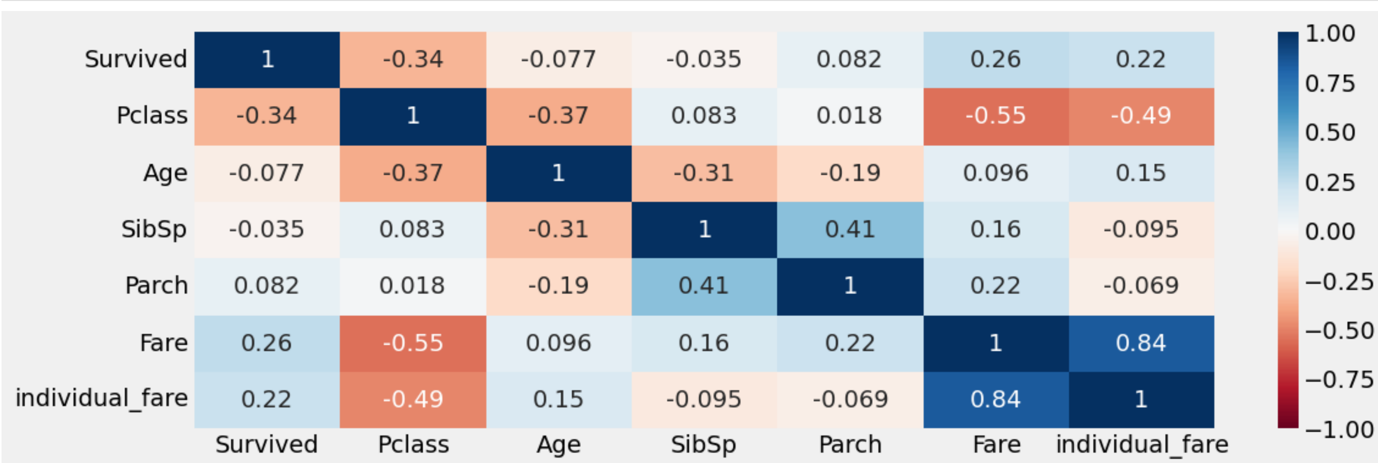
Using corr() function & Heatmap

In [92]:
```
data.corr()
```

Out[92]:

	Survived	Pclass	Age	SibSp	Parch	Fare	individual_fare
Survived	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307	0.221600
Pclass	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500	-0.485079
Age	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067	0.150763
SibSp	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651	-0.094682
Parch	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225	-0.068978
Fare	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000	0.840995
individual_fare	0.221600	-0.485079	0.150763	-0.094682	-0.068978	0.840995	1.000000

In [93]:
```
plt.figure(figsize=(12,4))
sns.heatmap(data.corr(),vmax=1,vmin=-1,cmap='RdBu',annot=True)
plt.show()
```



In [94]:
```
##  Drop Highly coralated columns
data.drop(columns=['individual_fare'],inplace=True)
```

Loading [MathJax]/extensions/Safe.js

In [95]:
```
data.head(2)
```

Out[95]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C

Missing values Imputation

Conculasion:-

1.) Age:- column in 177 missing values all missing values fill using median

2). Embarked :- column in 2 missing values fill missing values using mode

In [96]:
```
data.isnull().sum()
```

Out[96]:
```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

In [97]:
```
(data.isnull().sum()/data.shape[0])*100
```

Out[97]:
```
Survived     0.000000
Pclass       0.000000
Sex          0.000000
Age         19.865320
SibSp        0.000000
Parch        0.000000
Fare         0.000000
Embarked     0.224467
dtype: float64
```

In [98]:
```
for column in data:
    if data[column].dtype != "O":
        data[column].fillna(data[column].median(),inplace=True)
    else:
        data[column].fillna(data[column].mode()[0],inplace=True)
```

Outlier Detection and Removal

Conclusion:-

Age, fare these are columns in Outliers

Using IQR proximity rule Capping Outliers

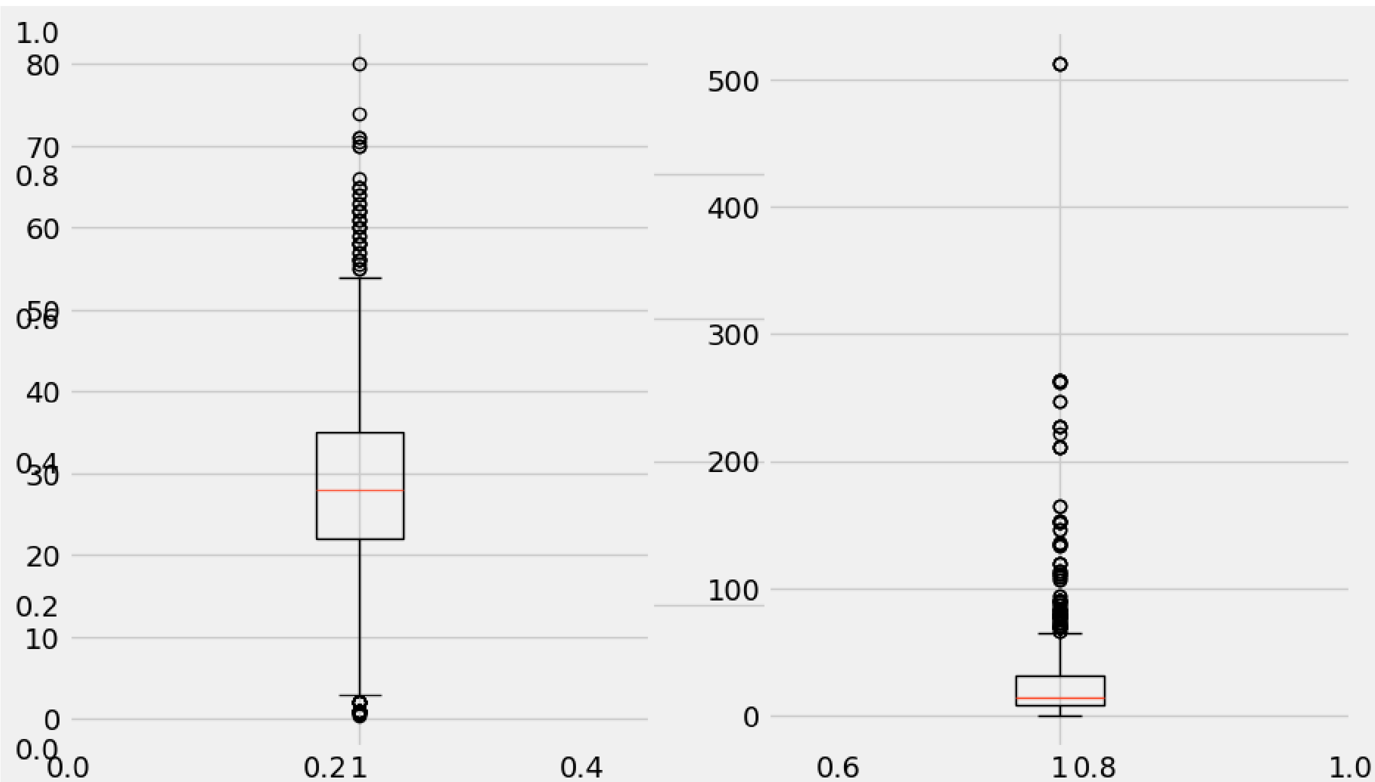
In [99]:
```
data.describe()
```

Out[99]:

	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.361582	0.523008	0.381594	32.204208
std	0.486592	0.836071	13.019697	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	22.000000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [100…
```
fig, ax = plt.subplots(figsize=(10,6))
ax1 = fig.add_subplot(1,2,1)
ax1.boxplot(data['Age'])
ax2 = fig.add_subplot(1,2,2)
ax2.boxplot(data['Fare'])

plt.show()
```



In [101…
```
for column in data[['Age','Fare']]:
    Q1 = np.percentile(data[column],25)
    Q3 = np.percentile(data[column],75)
    IQR = Q3-Q1
    Upper_bound = Q3+(1.5*IQR)
    Lower_bound = Q1-(1.5*IQR)
    data[column] = np.where(data[column]>Upper_bound,Upper_bound,data[column])
    data[column] = np.where(data[column]<Lower_bound,Lower_bound,data[column])
```

In [102…
```
# One Hot Encoding
```

In [103…
```
data = pd.get_dummies(data,drop_first=True)
```

In [104…
```
X = data.iloc[:,1:]
y = data.iloc[:,0]
```

In [105…
```
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=1,test_size=0.1)
```

In [106…
```
def Logistic_regression_model(X_train,X_test,y_train,y_test):
    """
    This function use for logistic regression model build & prediction
    """
    try:
        yeo = PowerTransformer()
        trf_1 = yeo.fit_transform(X_train)
        trf_2 = yeo.transform(X_test)

        scaler= StandardScaler()
        X_train_trf = scaler.fit_transform(trf_1)
        X_test_trf = scaler.transform(trf_2)

        Train_Standardized = pd.DataFrame(X_train_trf, columns = X_train.columns)
        Test_Standardized = pd.DataFrame(X_test_trf, columns = X_test.columns)
        print("*"*100)
        Lr = LogisticRegression()
        solvers = ['newton-cg', 'lbfgs', 'liblinear']
        penalty = ['l2']
        c_values = [100, 10, 1.0, 0.1, 0.01]

        grid = dict(solver=solvers,penalty=penalty,C=c_values)
        cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
        grid_search = GridSearchCV(estimator=Lr, param_grid=grid, n_jobs=-1, cv=10, scor
        grid_model = grid_search.fit(Train_Standardized,y_train)

        y_pred = grid_model.predict(Test_Standardized)
        print('Accuracy of the Logistic Regression:-%0.2f%%'%(accuracy_score(y_test,y_pr
        print('Precision_score:-',round(precision_score(y_test,y_pred),2))
        print('Recall_score:-',round(recall_score(y_test,y_pred),3))
        print('F1_score:-',round(f1_score(y_test,y_pred),2))
        print('Confusion matrix:-\n',confusion_matrix(y_test,y_pred))
        print(classification_report(y_test,y_pred))
    except Exception as E:
        print(E)

    except:
        print('Some Error')
    finally:
        print('Radhe Radhe')
```

In [107…
```
Logistic_regression_model(X_train,X_test,y_train,y_test)
```

```
****************************************************************************************************
Accuracy of the Logistic Regression:-0.77%
Precision_score:- 0.72
Recall_score:- 0.703
F1_score:- 0.71
Confusion matrix:-
 [[43 10]
 [11 26]]
              precision    recall  f1-score   support

           0       0.80      0.81      0.80        53
           1       0.72      0.70      0.71        37

    accuracy                           0.77        90
   macro avg       0.76      0.76      0.76        90
weighted avg       0.77      0.77      0.77        90

Radhe Radhe
```

In [124…

```
def Random_forest_classifiers_model(X_train,X_test,y_train,y_test):
    """
    This function use for Random forest model build & prediction
    """
    try:
        yeo = PowerTransformer()
        X_train_trf = yeo.fit_transform(X_train)
        X_test_trf = yeo.transform(X_test)

        print("*"*100)

        params = {
        'n_estimators': [100, 200, 500],
         'criterion': ['gini', 'entropy'],
        'min_samples_split': [1,2,4,5],
        'min_samples_leaf': [1,2,4,5],
       'max_leaf_nodes': [4,10,20,50,None]
        }

        grid_search = GridSearchCV(RandomForestClassifier(n_jobs=-1), params, n_jobs=-1,
        grid_search.fit(X_train_trf, y_train)

        print('Best score:', grid_search.best_score_)
        print('Best score:', grid_search.best_params_)

        Rm =RandomForestClassifier(n_estimators=100,criterion='entropy',max_leaf_nodes=N
                                  min_samples_split=4)
        print('*'*100)
        Rm.fit(X_train_trf,y_train)
        y_pred_grid = Rm.predict(X_test_trf)
        print('Accuracy of the Random forest:-',round(accuracy_score(y_test,y_pred_grid)
        print('Precision_score:-',round(precision_score(y_test,y_pred_grid),2))
        print('Recall_score:-',round(recall_score(y_test,y_pred_grid),2))
        print('F1_score:-',round(f1_score(y_test,y_pred_grid),2))
        print('Confusion matrix:-\n',confusion_matrix(y_test,y_pred_grid))
        print(classification_report(y_test,y_pred_grid))
    except Exception as E:
        print(E)

    except:
        print('Some Error')
    finally:
        print('Radhe Radhe')
```

Loading [MathJax]/extensions/Safe.js

In [125…
```
Random_forest_classifiers_model(X_train,X_test,y_train,y_test)
```

```
****************************************************************************************************
Best score: 0.875489645997003
Best score: {'criterion': 'entropy', 'max_leaf_nodes': 50, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 200}
****************************************************************************************************
Accuracy of the Random forest:- 0.77
Precision_score:- 0.79
Recall_score:- 0.59
F1_score:- 0.68
Confusion matrix:-
 [[47  6]
 [15 22]]
              precision    recall  f1-score   support

           0       0.76      0.89      0.82        53
           1       0.79      0.59      0.68        37

    accuracy                           0.77        90
   macro avg       0.77      0.74      0.75        90
weighted avg       0.77      0.77      0.76        90

Radhe Radhe
```

In [126…
```
def xgboostClassifier_model(X_train,X_test,y_train,y_test):
    """
    This function use for xgboostClassifier model build & prediction
    """
    try:
        yeo = PowerTransformer()
        X_train_trf = yeo.fit_transform(X_train)
        X_test_trf = yeo.transform(X_test)

        print("*"*100)
        params = {
          'n_estimators': [100, 200, 500],
           'learning_rate': [0.01,0.05,0.1],
           'booster': ['gbtree', 'gblinear'],
            'gamma': [0, 0.5, 1],
            'reg_alpha': [0, 0.5, 1],
            'reg_lambda': [0.5, 1, 5],
            'base_score': [0.2, 0.5, 1]
              }
        Xg_bost = GridSearchCV(XGBClassifier(n_jobs=-1), params, n_jobs=-1, cv=KFold(n_s
        Xg_bost.fit(X_train_trf, y_train)
        y_pred = Xg_bost.predict(X_test_trf)
        print('Accuracy of the xgboostClassifier model:-',round(accuracy_score(y_test,y_
        print('Precision_score:-',round(precision_score(y_test,y_pred),2))
        print('Recall_score:-',round(recall_score(y_test,y_pred),2))
        print('F1_score:-',round(f1_score(y_test,y_pred),2))
        print('Confusion matrix:-\n',confusion_matrix(y_test,y_pred))
        print(classification_report(y_test,y_pred))
    except Exception as E:
        print(E)

    except:
        print('Some Error')
```

Loading [MathJax]/extensions/Safe.js

```
    finally:
        print('Radhe Radhe')
```

In [127…

```
xgboostClassifier_model(X_train,X_test,y_train,y_test)
```

```
****************************************************************************************
************
Accuracy of the xgboostClassifier model:- 0.76
Precision_score:- 0.83
Recall_score:- 0.51
F1_score:- 0.63
Confusion matrix:-
 [[49  4]
 [18 19]]
              precision    recall  f1-score   support

           0       0.73      0.92      0.82        53
           1       0.83      0.51      0.63        37

    accuracy                           0.76        90
   macro avg       0.78      0.72      0.72        90
weighted avg       0.77      0.76      0.74        90

Radhe Radhe
```

In [130…

```
def ExtraTreesClassifier_model(X_train,X_test,y_train,y_test):
    """
    This function use for ExtraTreesClassifier model build & prediction
    """
    try:
        yeo = PowerTransformer()
        X_train_trf = yeo.fit_transform(X_train)
        X_test_trf = yeo.transform(X_test)

        print("*"*100)
        params = {
          'n_estimators': [100, 200, 500],
          'criterion': ['gini', 'entropy'],
         'min_samples_split': [1,2,4,5],
         'min_samples_leaf': [1,2,4,5],
        'max_leaf_nodes': [4,10,20,50,None]
            }
        gs3 = GridSearchCV(ExtraTreesClassifier(n_jobs=-1), params, n_jobs=-1, cv=KFold(
        gs3.fit(X_train_trf, y_train)

        print('Best score:', gs3.best_score_)
        print('Best score:', gs3.best_params_)
        print('*'*100)
        y_pred = gs3.predict(X_test_trf)

        print('Accuracy of the ExtraTreesClassifier model:-',round(accuracy_score(y_test
        print('Precision_score:-',round(precision_score(y_test,y_pred),2))
        print('Recall_score:-',round(recall_score(y_test,y_pred),2))
        print('F1_score:-',round(f1_score(y_test,y_pred),2))
        print('Confusion matrix:-\n',confusion_matrix(y_test,y_pred))
        print(classification_report(y_test,y_pred))
    except Exception as E:
        print(E)

    except:
        print('Some Error')
    finally:
        print('Radhe Radhe')
```

In [131…
```
ExtraTreesClassifier_model(X_train,X_test,y_train,y_test)
```

```
****************************************************************************************************
Best score: 0.8706590335395661
Best score: {'criterion': 'entropy', 'max_leaf_nodes': 20, 'min_samples_leaf': 1, 'min_samples_split': 4, 'n_estimators': 100}
****************************************************************************************************
Accuracy of the ExtraTreesClassifier model:- 0.74
Precision_score:- 0.77
Recall_score:- 0.54
F1_score:- 0.63
Confusion matrix:-
 [[47  6]
 [17 20]]
              precision    recall  f1-score   support

           0       0.73      0.89      0.80        53
           1       0.77      0.54      0.63        37

    accuracy                           0.74        90
   macro avg       0.75      0.71      0.72        90
weighted avg       0.75      0.74      0.73        90

Radhe Radhe
```

In [114…
```
def AdaBoostClassifier_model(X_train,X_test,y_train,y_test):
    """
    This function use for AdaBoostClassifier model build & prediction
    """
    try:
        yeo = PowerTransformer()
        X_train_trf = yeo.fit_transform(X_train)
        X_test_trf = yeo.transform(X_test)

        print("*"*100)
        parameters = {
                'n_estimators': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 20, 30]
                     }
        ab_clf = AdaBoostClassifier()
        clf = GridSearchCV(ab_clf, parameters, cv=5)
        clf.fit(X_train_trf, y_train)
        y_pred = clf.predict(X_test_trf)

        print('Accuracy of the AdaBoostClassifier model:-',round(accuracy_score(y_test,y
        print('Precision_score:-',round(precision_score(y_test,y_pred),2))
        print('Recall_score:-',round(recall_score(y_test,y_pred),2))
        print('F1_score:-',round(f1_score(y_test,y_pred),2))
        print('Confusion matrix:-\n',confusion_matrix(y_test,y_pred))
        print(classification_report(y_test,y_pred))
    except Exception as E:
        print(E)

    except:
        print('Some Error')
    finally:
        print('Radhe Radhe')
```

In [115…
```
AdaBoostClassifier_model(X_train,X_test,y_train,y_test)
```

Loading [MathJax]/extensions/Safe.js

```
****************************************************************************************************
Accuracy of the AdaBoostClassifier model:- 0.74
Precision_score:- 0.72
Recall_score:- 0.62
F1_score:- 0.67
Confusion matrix:-
 [[44  9]
 [14 23]]
              precision    recall  f1-score   support

           0       0.76      0.83      0.79        53
           1       0.72      0.62      0.67        37

    accuracy                           0.74        90
   macro avg       0.74      0.73      0.73        90
weighted avg       0.74      0.74      0.74        90

Radhe Radhe
```

In [137…

```python
def BaggingClassifier_model(X_train,X_test,y_train,y_test):
    """
    This function use for BaggingClassifier model build & prediction
    """
    try:
        yeo = PowerTransformer()
        X_train_trf = yeo.fit_transform(X_train)
        X_test_trf = yeo.transform(X_test)

        print("*"*100)
        param_grid = {
                  'base_estimator__max_depth' : [1, 2, 3, 4, 5],
                   'max_samples' : [0.05, 0.1, 0.2, 0.5]
                     }

        Bg = GridSearchCV(BaggingClassifier(DecisionTreeClassifier(),
                                    n_estimators = 100, max_features = 0.5),
                      param_grid, scoring = 'accuracy')
        Bg.fit(X_train_trf, y_train)
        print('*'*100)
        y_pred = Bg.predict(X_test_trf)
        print('Accuracy of the BaggingClassifier model:-',round(accuracy_score(y_test,y_
        print('Precision_score:-',round(precision_score(y_test,y_pred),2))
        print('Recall_score:-',round(recall_score(y_test,y_pred),2))
        print('F1_score:-',round(f1_score(y_test,y_pred),2))
        print('Confusion matrix:-\n',confusion_matrix(y_test,y_pred))
        print(classification_report(y_test,y_pred))
    except Exception as E:
        print(E)

    except:
        print('Some Error')
    finally:
        print('Radhe Radhe')
```

In [138…

```python
BaggingClassifier_model(X_train,X_test,y_train,y_test)
```

Loading [MathJax]/extensions/Safe.js

```
****************************************************************************************
************
****************************************************************************************
************
Accuracy of the BaggingClassifier model:- 0.77
Precision_score:- 0.81
Recall_score:- 0.57
F1_score:- 0.67
Confusion matrix:-
 [[48  5]
 [16 21]]
              precision    recall  f1-score   support

           0       0.75      0.91      0.82        53
           1       0.81      0.57      0.67        37

    accuracy                           0.77        90
   macro avg       0.78      0.74      0.74        90
weighted avg       0.77      0.77      0.76        90

Radhe Radhe
```

Accuracy of All Model

Logistic Regression :- 77%

Random forest :- 77%

xgboostClassifier :- 76%

ExtraTreesClassifier :- 74%

AdaBoostClassifier :- 74%

BaggingClassifier:- 77%

you can see almost all model accuracy is same

In [ ]: