In [34]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from warnings import filterwarnings
filterwarnings(action='ignore')
```

In [35]:
```
test= pd.read_csv(r"C:\Users\ADMIN\Downloads\test.csv")
train = pd.read_csv(r"C:\Users\ADMIN\Downloads\train.csv")
```

In [36]:
```
test.head()
```

Out[36]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embark
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	

In [37]:
```
test.shape
```

Out[37]:
```
(418, 11)
```

In [38]:
```
train.shape
```

Out[38]:
```
(891, 12)
```

In [39]:
```
test.isnull().sum()
```

Out[39]:
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [40]:
```
train.isnull().sum()
```

Out[40]:
```
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 [41]:
```
train.describe(include="all")
```

Out[41]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch
count	891.000000	891.000000	891.000000	891	891	714.000000	891.000000	891.000000
unique	NaN	NaN	NaN	891	2	NaN	NaN	NaN
top	NaN	NaN	NaN	Braund, Mr. Owen Harris	male	NaN	NaN	NaN
freq	NaN	NaN	NaN	1	577	NaN	NaN	NaN
mean	446.000000	0.383838	2.308642	NaN	NaN	29.699118	0.523008	0.381594
std	257.353842	0.486592	0.836071	NaN	NaN	14.526497	1.102743	0.806057
min	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	NaN	NaN	20.125000	0.000000	0.000000
50%	446.000000	0.000000	3.000000	NaN	NaN	28.000000	0.000000	0.000000
75%	668.500000	1.000000	3.000000	NaN	NaN	38.000000	1.000000	0.000000
max	891.000000	1.000000	3.000000	NaN	NaN	80.000000	8.000000	6.000000

In [58]:
```
train.groupby('Survived')
```

Out[58]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001FBD82508C0>
```

In [64]:
```
train = pd.read_csv(r"C:\Users\ADMIN\Downloads\train.csv")
train.corr
```

Out[64]:
```
<bound method DataFrame.corr of      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]>
```

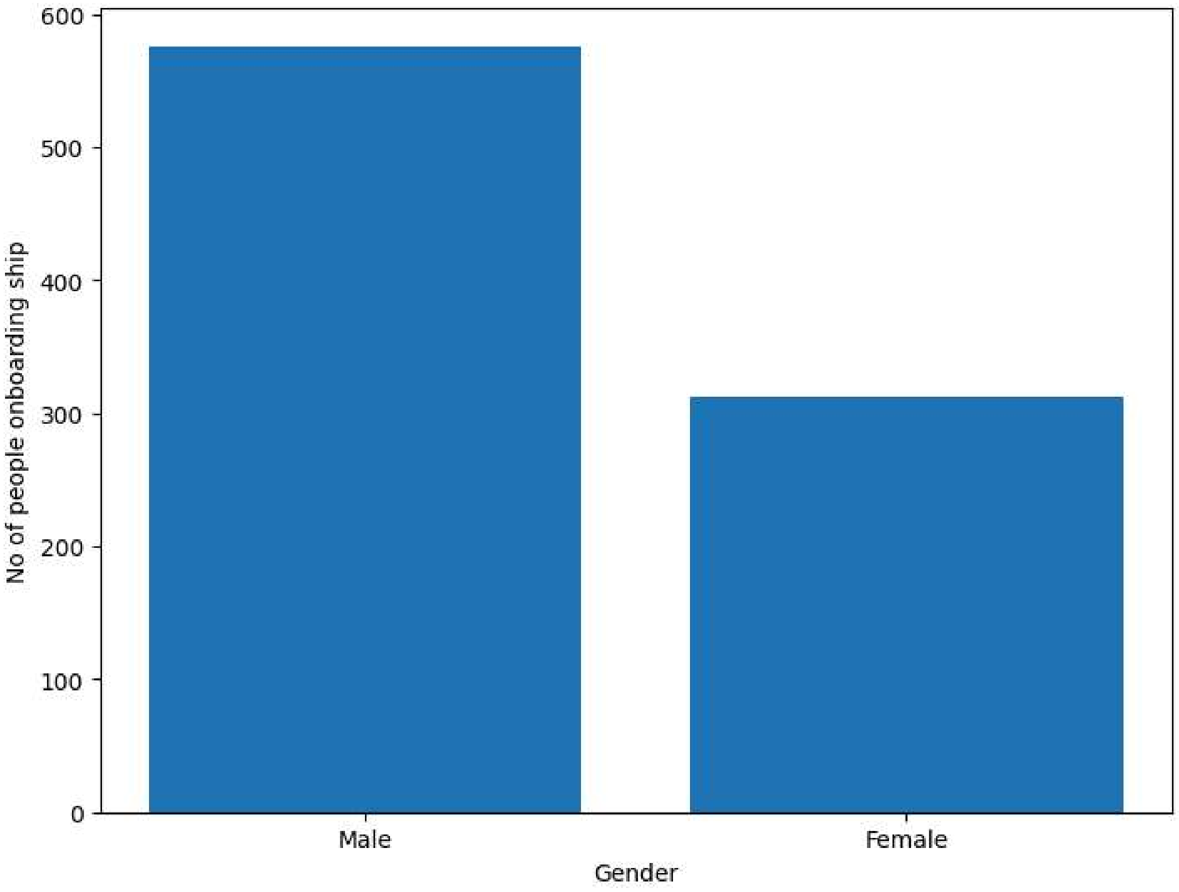
In [65]:
```
male_ind = len(train[train['Sex'] == 'male'])
print("No of Males in Titanic:",male_ind)
```

```
No of Males in Titanic: 577
```

In [66]:
```
female_ind = len(train[train['Sex'] == 'female'])
print("No of Females in Titanic:",female_ind)
```

```
No of Females in Titanic: 314
```

In [101]:
```
#Plotting
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
gender = ['Male','Female']
index = [576,312]
ax.bar(gender,index)
plt.xlabel("Gender")
plt.ylabel("No of people onboarding ship")
plt.show()
```



In [68]:
```
alive = len(train[train['Survived'] == 1])
dead = len(train[train['Survived'] == 0])
```

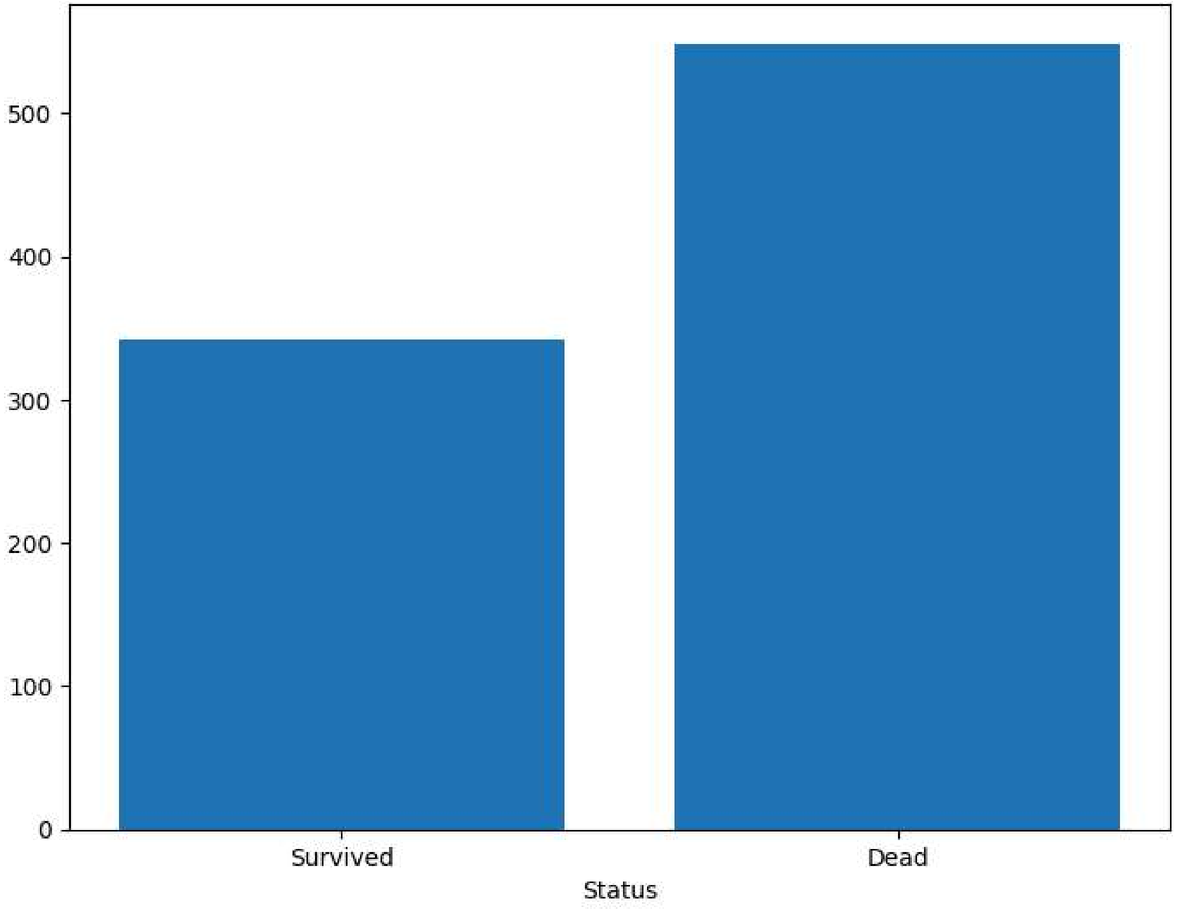
In [69]:
```
train.groupby('Sex')[['Survived']].mean()
```

Out[69]:

	Survived
Sex	
female	0.742038
male	0.188908

In [70]:

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
status = ['Survived','Dead']
ind = [alive,dead]
ax.bar(status,ind)
plt.xlabel("Status")
plt.show()
```



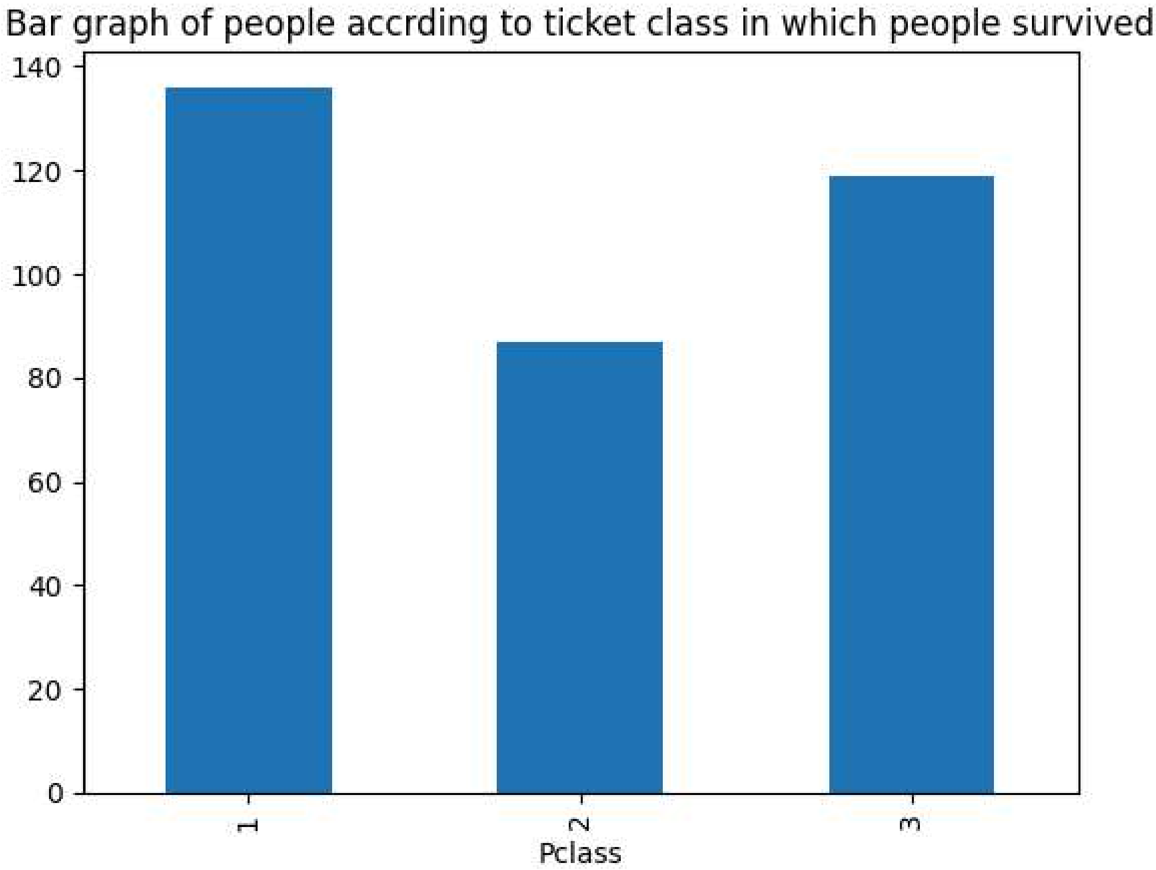
In [71]:

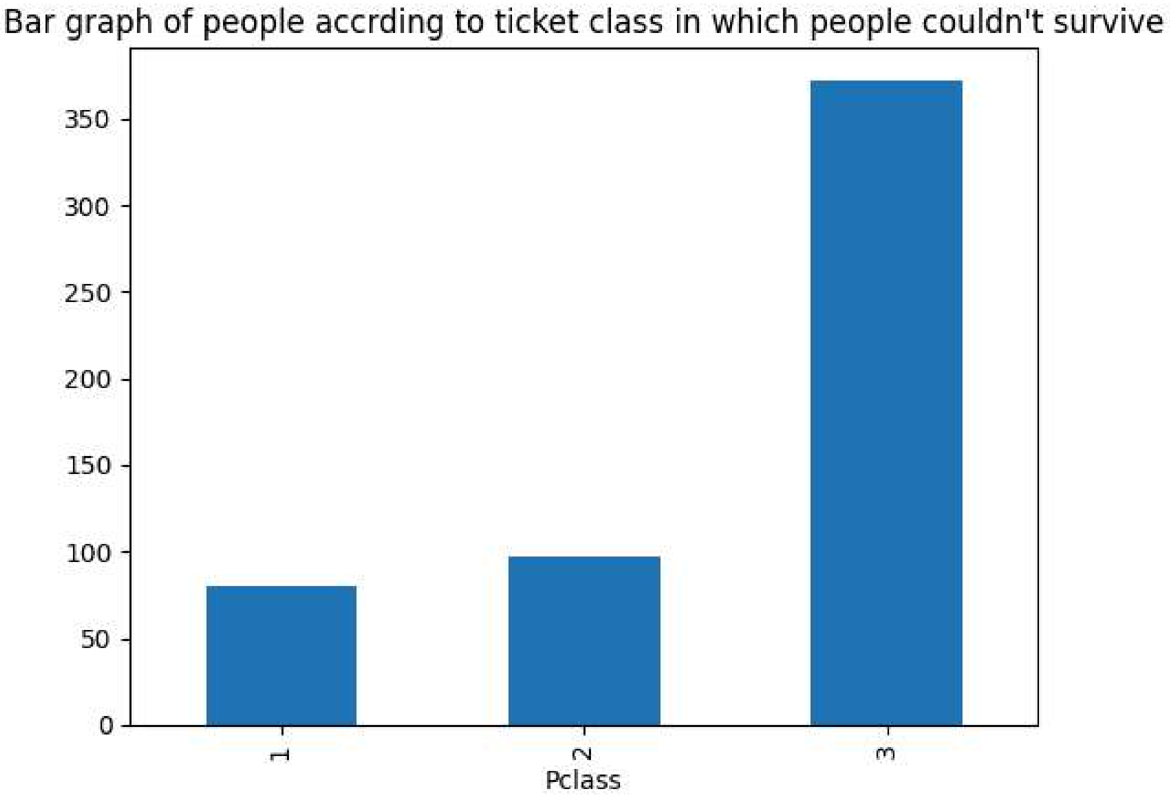
```
plt.figure(1)
train.loc[train['Survived'] == 1, 'Pclass'].value_counts().sort_index().plot.ba
plt.title('Bar graph of people accrding to ticket class in which people survive


plt.figure(2)
train.loc[train['Survived'] == 0, 'Pclass'].value_counts().sort_index().plot.ba
plt.title('Bar graph of people accrding to ticket class in which people couldn\
```

Out[71]:

```
Text(0.5, 1.0, "Bar graph of people accrding to ticket class in which people
couldn't survive")
```




Bar graph of people accrding to ticket class in which people couldn't survive
350
300
250
200
150
100
50
0
1
2
3
Pclass

In [72]:

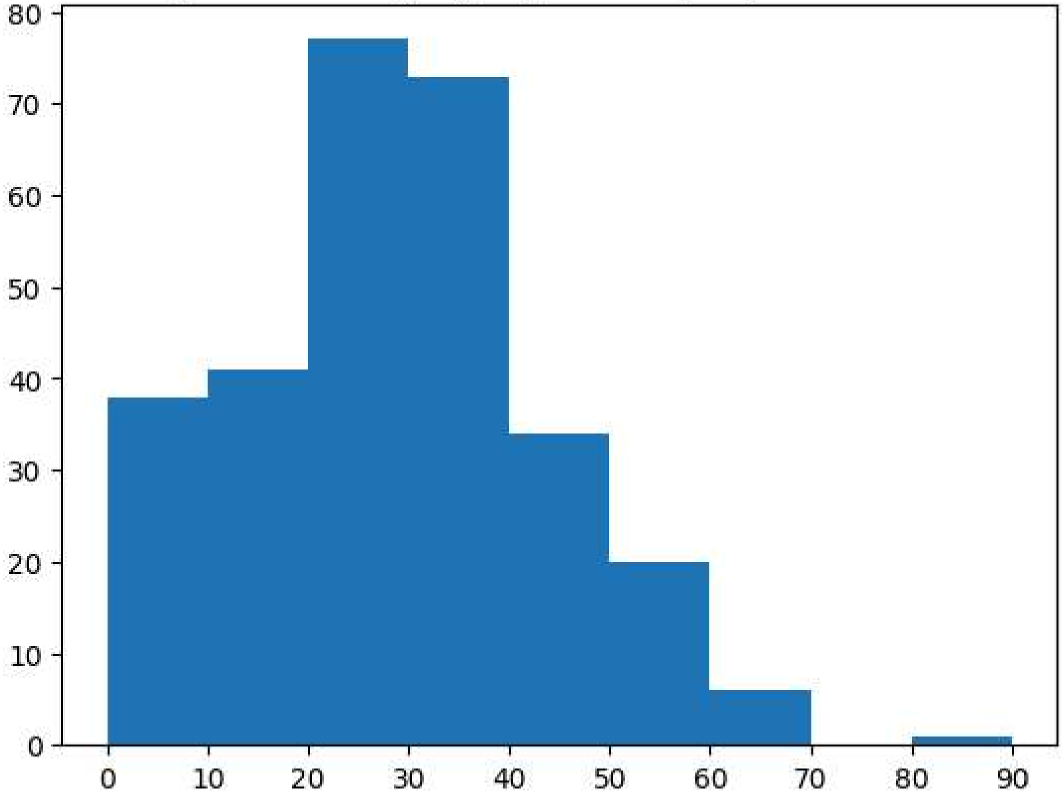
```
plt.figure(1)
age  = train.loc[train.Survived == 1, 'Age']
plt.title('The histogram of the age groups of the people that had survived')
plt.hist(age, np.arange(0,100,10))
plt.xticks(np.arange(0,100,10))


plt.figure(2)
age  = train.loc[train.Survived == 0, 'Age']
plt.title('The histogram of the age groups of the people that coudn\'t survive
plt.hist(age, np.arange(0,100,10))
plt.xticks(np.arange(0,100,10))
```

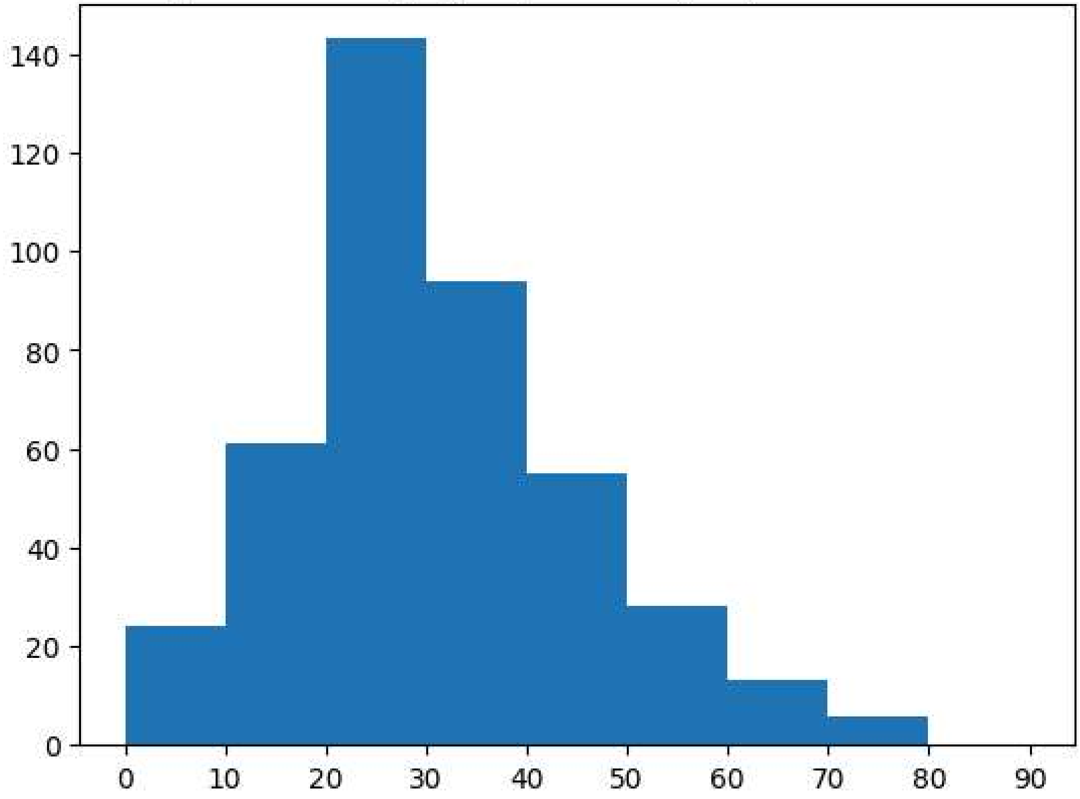
Out[72]:

```
([<matplotlib.axis.XTick at 0x1fbdbb73ad0>,
  <matplotlib.axis.XTick at 0x1fbdbb73aa0>,
  <matplotlib.axis.XTick at 0x1fbdbb72420>,
  <matplotlib.axis.XTick at 0x1fbdbbc46b0>,
  <matplotlib.axis.XTick at 0x1fbdbbc4ce0>,
  <matplotlib.axis.XTick at 0x1fbdbbc5640>,
  <matplotlib.axis.XTick at 0x1fbdbbc5eb0>,
  <matplotlib.axis.XTick at 0x1fbdbbc67e0>,
  <matplotlib.axis.XTick at 0x1fbdbbc71a0>,
  <matplotlib.axis.XTick at 0x1fbdbbc5970>],
 [Text(0, 0, '0'),
  Text(10, 0, '10'),
  Text(20, 0, '20'),
  Text(30, 0, '30'),
  Text(40, 0, '40'),
  Text(50, 0, '50'),
  Text(60, 0, '60'),
  Text(70, 0, '70'),
  Text(80, 0, '80'),
  Text(90, 0, '90')])
```

The histogram of the age groups of the people that had survived



The histogram of the age groups of the people that coudn't survive



In [73]:
```
train[["SibSp", "Survived"]].groupby(['SibSp'], as_index=False).mean().sort_va
```

Out[73]:

	SibSp	Survived
1	1	0.535885
2	2	0.464286
0	0	0.345395
3	3	0.250000
4	4	0.166667
5	5	0.000000
6	8	0.000000

In [74]:
```
train[["Pclass", "Survived"]].groupby(['Pclass'], as_index=False).mean().sort_
```

Out[74]:

	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

In [75]:
```
train[["Age", "Survived"]].groupby(['Age'], as_index=False).mean().sort_values
```

Out[75]:

	Age	Survived
0	0.42	1.0
1	0.67	1.0
2	0.75	1.0
3	0.83	1.0
4	0.92	1.0
...	...	...
83	70.00	0.0
84	70.50	0.0
85	71.00	0.0
86	74.00	0.0
87	80.00	1.0

88 rows × 2 columns

In [76]:

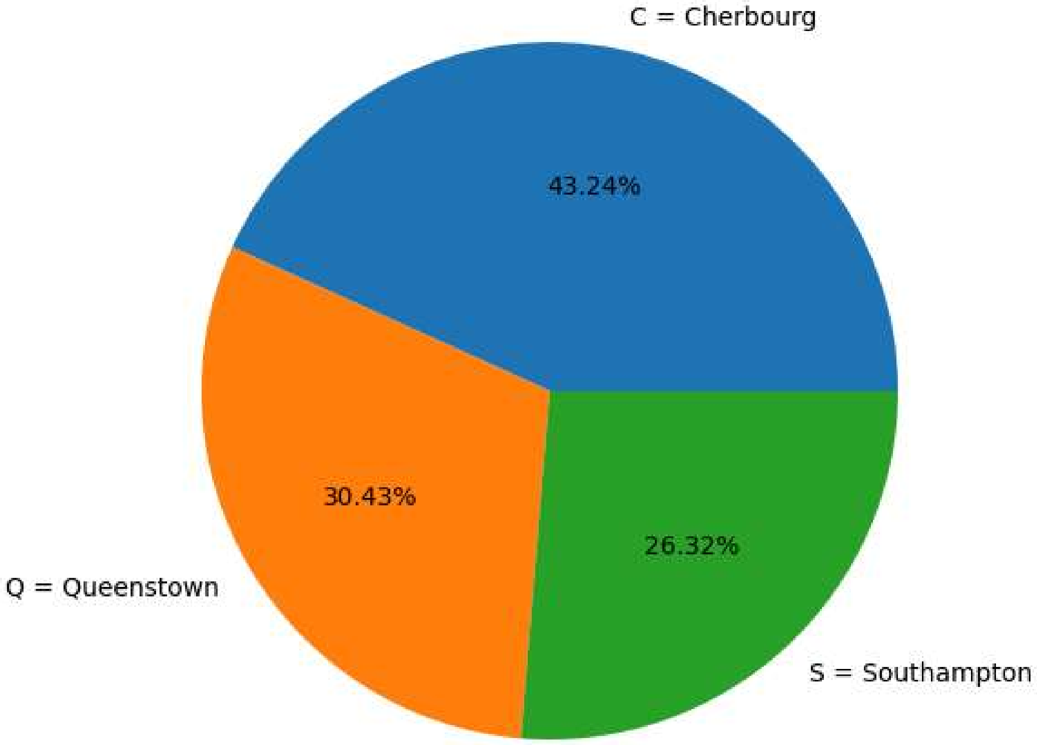
```
train[["Embarked", "Survived"]].groupby(['Embarked'], as_index=False).mean().s
```

Out[76]:

	Embarked	Survived
0	C	0.553571
1	Q	0.389610
2	S	0.336957

In [77]:

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
l = ['C = Cherbourg', 'Q = Queenstown', 'S = Southampton']
s = [0.553571,0.389610,0.336957]
ax.pie(s, labels = l,autopct='%1.2f%%')
plt.show()
```



In [78]:

```
test.describe(include="all")
```

Out[78]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
count	418.000000	418.000000	418	418	332.000000	418.000000	418.000000	418	417.0
unique	NaN	NaN	418	2	NaN	NaN	NaN	363	
top	NaN	NaN	Kelly, Mr. James	male	NaN	NaN	NaN	PC 17608	
freq	NaN	NaN	1	266	NaN	NaN	NaN	5	
mean	1100.500000	2.265550	NaN	NaN	30.272590	0.447368	0.392344	NaN	35.6
std	120.810458	0.841838	NaN	NaN	14.181209	0.896760	0.981429	NaN	55.9
min	892.000000	1.000000	NaN	NaN	0.170000	0.000000	0.000000	NaN	0.0
25%	996.250000	1.000000	NaN	NaN	21.000000	0.000000	0.000000	NaN	7.8
50%	1100.500000	3.000000	NaN	NaN	27.000000	0.000000	0.000000	NaN	14.4
75%	1204.750000	3.000000	NaN	NaN	39.000000	1.000000	0.000000	NaN	31.5
max	1309.000000	3.000000	NaN	NaN	76.000000	8.000000	9.000000	NaN	512.3

In [79]:

```
#Droping Useless Columns
train = train.drop(['Ticket'], axis = 1)
test = test.drop(['Ticket'], axis = 1)
```

In [80]:

```
train = train.drop(['Cabin'], axis = 1)
test = test.drop(['Cabin'], axis = 1)
```

In [81]:

```
train = train.drop(['Name'], axis = 1)
test = test.drop(['Name'], axis = 1)
```

In [82]:

```
#Feature Selection
column_train=['Age','Pclass','SibSp','Parch','Fare','Sex','Embarked']
#training values
X=train[column_train]
#target value
Y=train['Survived']
```

In [83]:

```
X['Age'].isnull().sum()
X['Pclass'].isnull().sum()
X['SibSp'].isnull().sum()
X['Parch'].isnull().sum()
X['Fare'].isnull().sum()
X['Sex'].isnull().sum()
X['Embarked'].isnull().sum()
```

Out[83]:

```
2
```

In [84]:
```
X['Age']=X['Age'].fillna(X['Age'].median())
X['Age'].isnull().sum()
```

Out[84]:
```
0
```

In [85]:
```
X['Embarked'] = train['Embarked'].fillna(method ='pad')
X['Embarked'].isnull().sum()
```

Out[85]:
```
0
```

In [86]:
```
d={'male':0, 'female':1}
X['Sex']=X['Sex'].apply(lambda x:d[x])
X['Sex'].head()
```

Out[86]:
```
0    0
1    1
2    1
3    1
4    0
Name: Sex, dtype: int64
```

In [87]:
```
e={'C':0, 'Q':1 ,'S':2}
X['Embarked']=X['Embarked'].apply(lambda x:e[x])
X['Embarked'].head()
```

Out[87]:
```
0    2
1    0
2    2
3    2
4    2
Name: Embarked, dtype: int64
```

In [88]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3,random_st
```

In [89]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,Y_train)
Y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,Y_pred))
```

```
Accuracy Score: 0.7574626865671642
```

In [90]:
```
from sklearn.metrics import accuracy_score,confusion_matrix
confusion_mat = confusion_matrix(Y_test,Y_pred)
print(confusion_mat)
```

```
[[130  26]
 [ 39  73]]
```

In [91]:

```
from sklearn.svm import SVC
model1 = SVC()
model1.fit(X_train,Y_train)

pred_y = model1.predict(X_test)

from sklearn.metrics import accuracy_score
print("Acc=",accuracy_score(Y_test,pred_y))
```

```
Acc= 0.6604477611940298
```

In [92]:

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_repo
confusion_mat = confusion_matrix(Y_test,pred_y)
print(confusion_mat)
print(classification_report(Y_test,pred_y))
```

```
[[149   7]
 [ 84  28]]
              precision    recall  f1-score   support

           0       0.64      0.96      0.77       156
           1       0.80      0.25      0.38       112

    accuracy                           0.66       268
   macro avg       0.72      0.60      0.57       268
weighted avg       0.71      0.66      0.61       268
```

In [93]:

```
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier(n_neighbors=5)
model2.fit(X_train,Y_train)
y_pred2 = model2.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred2))
```

```
Accuracy Score: 0.6604477611940298
```

In [94]:

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_repo
confusion_mat = confusion_matrix(Y_test,y_pred2)
print(confusion_mat)
print(classification_report(Y_test,y_pred2))
```

```
[[127  29]
 [ 62  50]]
              precision    recall  f1-score   support

           0       0.67      0.81      0.74       156
           1       0.63      0.45      0.52       112

    accuracy                           0.66       268
   macro avg       0.65      0.63      0.63       268
weighted avg       0.66      0.66      0.65       268
```

In [95]:

```
from sklearn.naive_bayes import GaussianNB
model3 = GaussianNB()
model3.fit(X_train,Y_train)
y_pred3 = model3.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred3))
```

```
Accuracy Score: 0.7686567164179104
```

In [96]:

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_repo
confusion_mat = confusion_matrix(Y_test,y_pred3)
print(confusion_mat)
print(classification_report(Y_test,y_pred3))
```

```
[[129  27]
 [ 35  77]]
              precision    recall  f1-score   support

           0       0.79      0.83      0.81       156
           1       0.74      0.69      0.71       112

    accuracy                           0.77       268
   macro avg       0.76      0.76      0.76       268
weighted avg       0.77      0.77      0.77       268
```

In [97]:

```
from sklearn.tree import DecisionTreeClassifier
model4 = DecisionTreeClassifier(criterion='entropy',random_state=7)
model4.fit(X_train,Y_train)
y_pred4 = model4.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred4))
```

```
Accuracy Score: 0.7425373134328358
```

In [98]:

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_repo
confusion_mat = confusion_matrix(Y_test,y_pred4)
print(confusion_mat)
print(classification_report(Y_test,y_pred4))
```

```
[[132  24]
 [ 45  67]]
              precision    recall  f1-score   support

           0       0.75      0.85      0.79       156
           1       0.74      0.60      0.66       112

    accuracy                           0.74       268
   macro avg       0.74      0.72      0.73       268
weighted avg       0.74      0.74      0.74       268
```

In [99]:

```
results = pd.DataFrame({
    'Model': ['Logistic Regression','Support Vector Machines', 'Naive Bayes','K
    'Score': [0.75,0.66,0.76,0.66,0.74]})

result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
result_df.head(9)
```

Out[99]:

```
                          Model
Score
 0.76               Naive Bayes
 0.75       Logistic Regression
 0.74             Decision Tree
 0.66   Support Vector Machines
 0.66                       KNN
```

In [ ]: