

```
In [11]: # STEP1: Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
In [12]: # STEP2: Importing DataSets
train_data = pd.read_csv('./train.csv')
data_test = pd.read_csv('./test.csv')
train_data.head()
```

Out[12]:

	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 [13]: # STEP3: Print summary information
train_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
```

```
In [14]: data_test.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
#   Column      Non-Null Count  Dtype
---  -
0   PassengerId  418 non-null    int64
1   Pclass       418 non-null    int64
2   Name         418 non-null    object
3   Sex          418 non-null    object
4   Age          332 non-null    float64
5   SibSp        418 non-null    int64
6   Parch        418 non-null    int64
7   Ticket       418 non-null    object
8   Fare         417 non-null    float64
9   Cabin        91 non-null     object
10  Embarked     418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.1+ KB
```

```
In [15]: # STEP4: Display Shape
train_data.shape
```

Out[15]: (891, 12)

```
In [16]: data_test.shape
```

Out[16]: (418, 11)

```
In [17]: # STEP5: Checking null values
train_data.isnull().sum()
```

Out[17]: 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 [18]: data\_test.isnull().sum()

Out[18]: PassengerId 0  
Pclass 0  
Name 0  
Sex 0  
Age 86  
SibSp 0  
Parch 0  
Ticket 0  
Fare 1  
Cabin 327  
Embarked 0  
dtype: int64

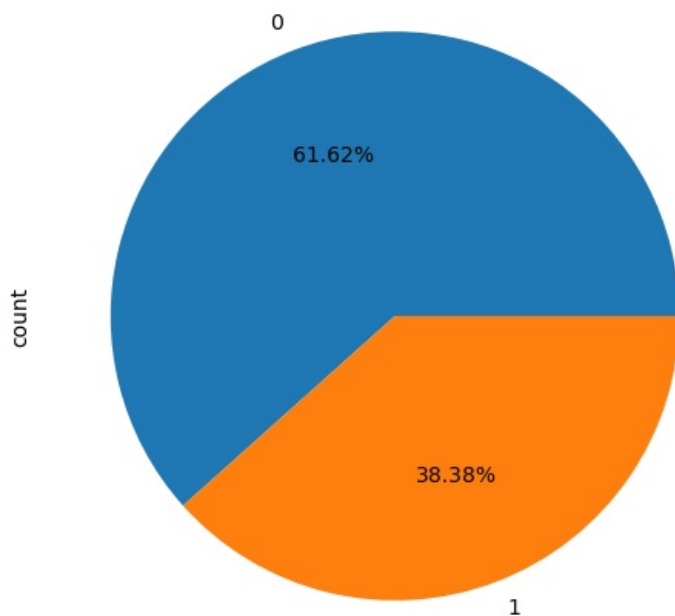
In [19]: # STEP6: making a discription of datasets  
train\_data.describe(include="all")

Out[19]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
count	891.000000	891.000000	891.000000	891	891	714.000000	891.000000	891.000000	891	891.000000	204	889
unique	NaN	NaN	NaN	891	2	NaN	NaN	NaN	681	NaN	147	3
top	NaN	NaN	NaN	Braund, Mr. Owen Harris	male	NaN	NaN	NaN	347082	NaN	B96 B98	S
freq	NaN	NaN	NaN	1	577	NaN	NaN	NaN	7	NaN	4	644
mean	446.000000	0.383838	2.308642	NaN	NaN	29.699118	0.523008	0.381594	NaN	32.204208	NaN	NaN
std	257.353842	0.486592	0.836071	NaN	NaN	14.526497	1.102743	0.806057	NaN	49.693429	NaN	NaN
min	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.000000	0.000000	NaN	0.000000	NaN	NaN
25%	223.500000	0.000000	2.000000	NaN	NaN	20.125000	0.000000	0.000000	NaN	7.910400	NaN	NaN
50%	446.000000	0.000000	3.000000	NaN	NaN	28.000000	0.000000	0.000000	NaN	14.454200	NaN	NaN
75%	668.500000	1.000000	3.000000	NaN	NaN	38.000000	1.000000	0.000000	NaN	31.000000	NaN	NaN
max	891.000000	1.000000	3.000000	NaN	NaN	80.000000	8.000000	6.000000	NaN	512.329200	NaN	NaN

In [20]: # STEP7 : let us check the overall survival ratio  
fig = plt.figure(figsize=(6,6))  
train\_data['Survived'].value\_counts().plot.pie(autopct = '%1.2f%%')

Out[20]: <Axes: ylabel='count'>



```
In [21]: # checking no.of males
male_ind = len(train_data[train_data['Sex'] == 'male'])
print("No of Males in Titanic:",male_ind)
```

No of Males in Titanic: 577

```
In [22]: # checking no.of females
female_ind = len(train_data[train_data['Sex'] == 'female'])
print("No of Females in Titanic:",female_ind)
```

No of Females in Titanic: 314

```
In [23]: train_data['Cabin'] = train_data['Cabin'].fillna('U0')
```

```
In [24]: # STEP8 : adding libraries to find age and survived people
from sklearn.ensemble import RandomForestRegressor

age_df = train_data[['Age', 'Survived', 'Fare', 'Parch', 'SibSp', 'Pclass']]
age_df_notnull = age_df.loc[(train_data['Age'].notnull())]
age_df_isnull = age_df.loc[(train_data['Age'].isnull())]
X = age_df_notnull.values[:,1:]
Y = age_df_notnull.values[:,0]
# use RandomForestRegression to train data
RFR = RandomForestRegressor(n_estimators=1000,n_jobs=-1)
RFR.fit(X,Y)
predictAges = RFR.predict(age_df_isnull.values[:,1:])
train_data.loc[train_data['Age'].isnull(), ['Age']] = predictAges
```

```
In [25]: # STEP9: checking complementary data
```

```
train_data.info()
```

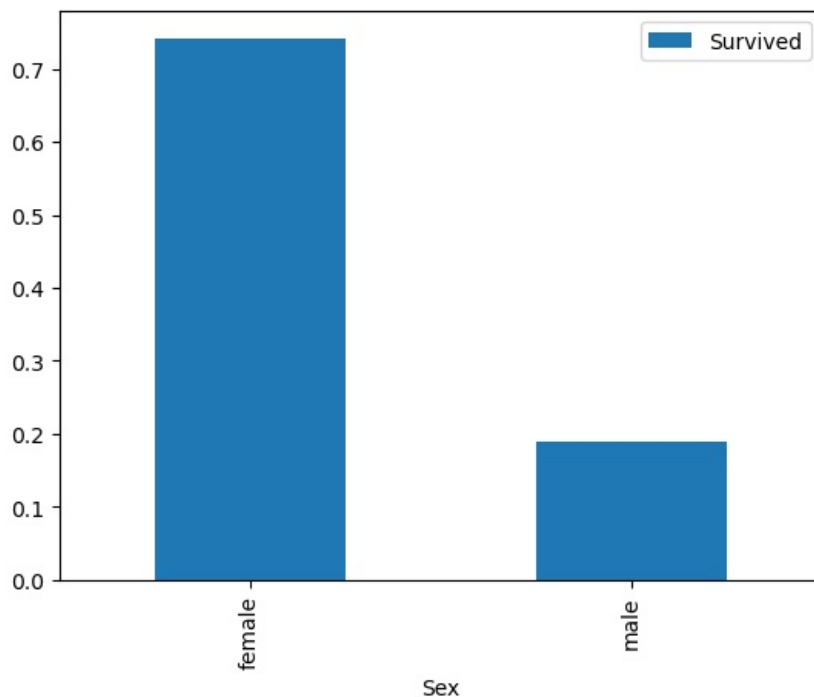
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
#   Column      Non-Null Count  Dtype  
---  -
0   PassengerId  891 non-null   int64  
1   Survived     891 non-null   int64  
2   Pclass       891 non-null   int64  
3   Name         891 non-null   object  
4   Sex          891 non-null   object  
5   Age          891 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        891 non-null   object  
11  Embarked     889 non-null   object  
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
In [26]: train_data.groupby(['Sex', 'Survived'])['Survived'].count()
```

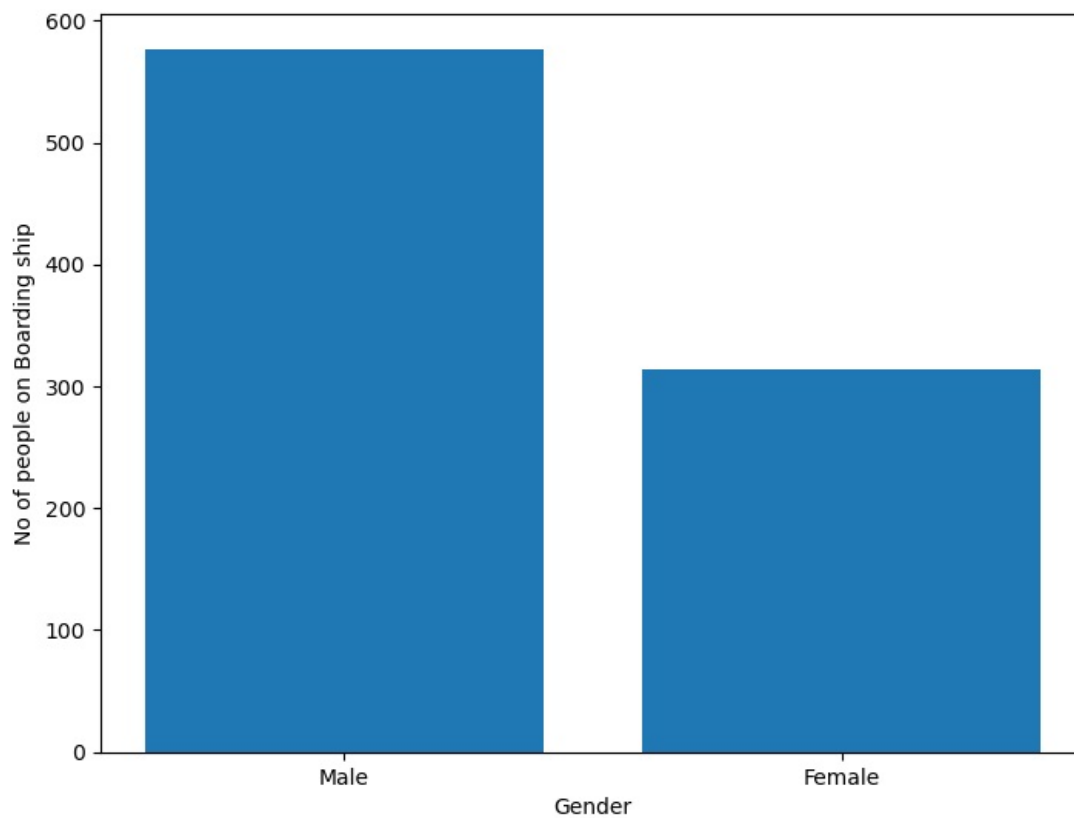
```
Out[26]: Sex      Survived
female 0           81
        1          233
male    0          468
        1          109
Name: Survived, dtype: int64
```

```
In [27]: # making a barplot
survived_by_sex = train_data[['Sex', 'Survived']].groupby('Sex').mean()
type(survived_by_sex)
survived_by_sex.plot.bar()
```

```
Out[27]: <Axes: xlabel='Sex'>
```



```
In [28]: #Plotting the no. of people on boarding in ship
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
gender = ['Male', 'Female']
index = [577, 314]
ax.bar(gender, index)
plt.xlabel("Gender")
plt.ylabel("No of people on Boarding ship")
plt.show()
```



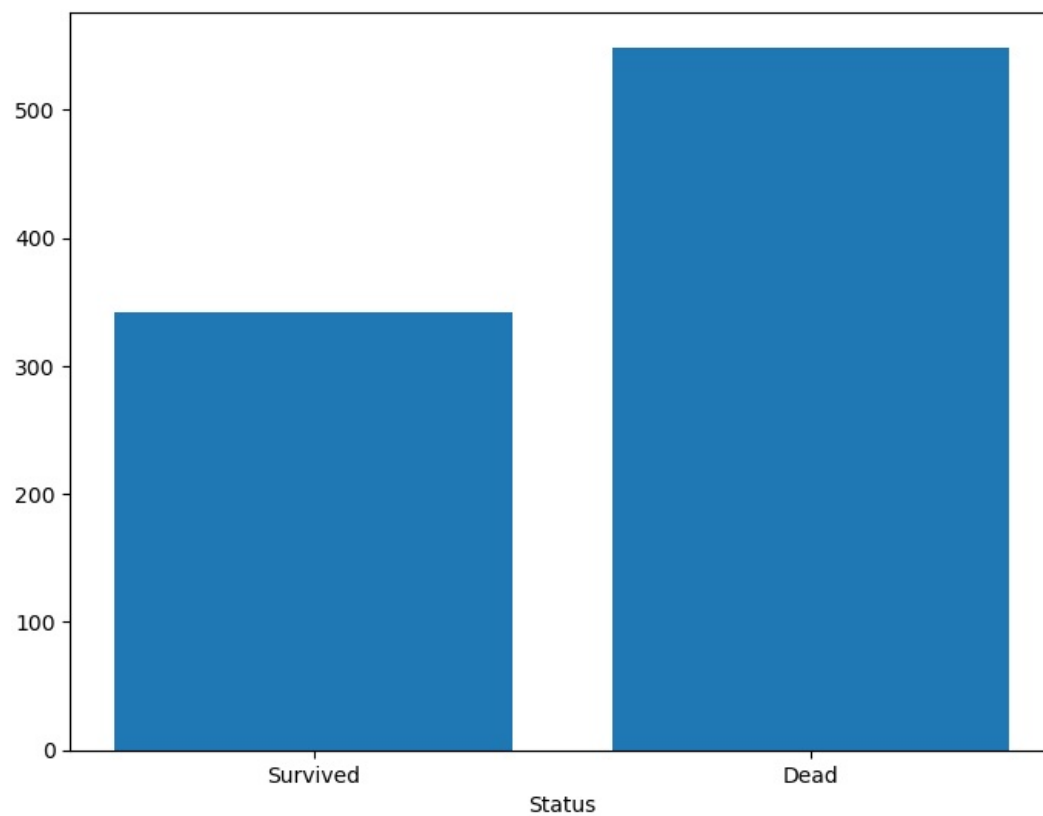
```
In [29]: # STEP10: checking alives and dead
alive = len(train_data[train_data['Survived'] == 1])
dead = len(train_data[train_data['Survived'] == 0])
```

```
In [30]: train_data.groupby('Sex')[['Survived']].mean()
```

```
Out[30]:
```

Survived	
Sex	
female	0.742038
male	0.188908

```
In [31]: # plotting alive and dead
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()
```

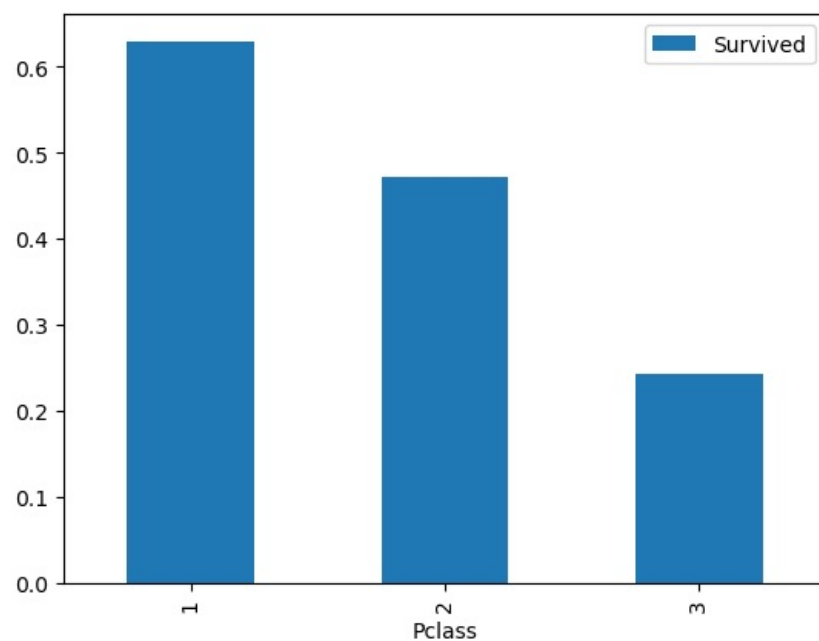


```
In [32]: # STEP11: relationship between cabin class and survival or not Pclass
train_data.groupby(['Pclass', 'Survived'])['Pclass'].count()
```

```
Out[32]: Pclass  Survived
1         0          80
         1         136
2         0          97
         1          87
3         0         372
         1         119
Name: Pclass, dtype: int64
```

```
In [33]: train_data[['Pclass', 'Survived']].groupby(['Pclass']).mean().plot.bar()
```

```
Out[33]: <Axes: xlabel='Pclass'>
```



```
In [34]: train_data.groupby(['Sex', 'Pclass', 'Survived'])['Survived'].count()
```

```
Out[34]:
```

Sex	Pclass	Survived
female	1	0
		1
	2	0
		1
	3	0
		1
male	1	0
		1
	2	0
		1
	3	0
		1

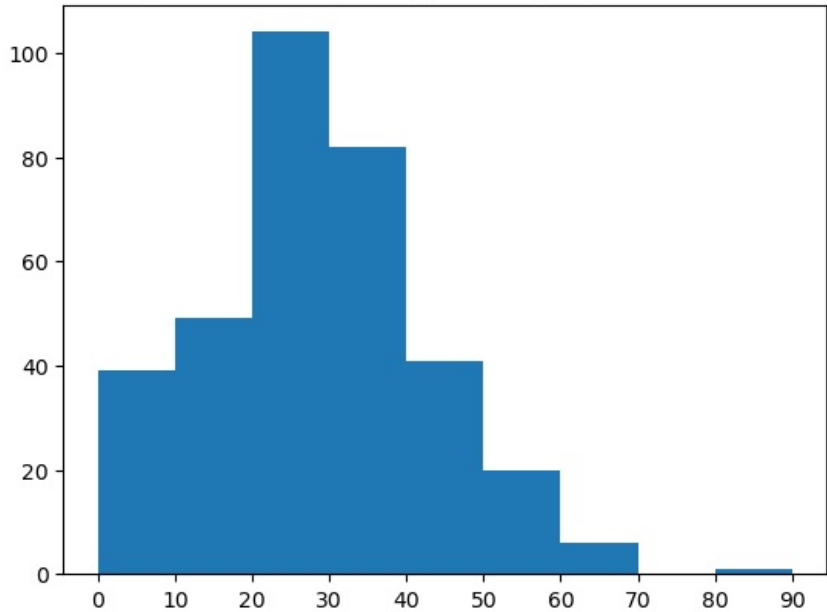
Name: Survived, dtype: int64

```
In [35]: plt.figure(1)
age = train_data.loc[train_data.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))

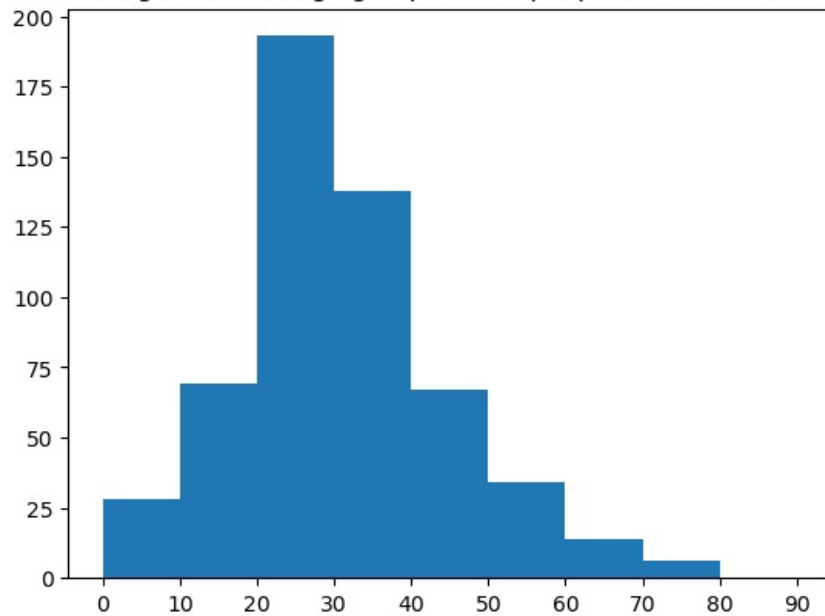
plt.figure(2)
age = train_data.loc[train_data.Survived == 0, 'Age']
plt.title('The histogram of the age groups of the people that couldn\'t survive')
plt.hist(age, np.arange(0,100,10))
plt.xticks(np.arange(0,100,10))
```

```
Out[35]: ([<matplotlib.axis.XTick at 0x1ccdb15e050>,
<matplotlib.axis.XTick at 0x1ccdb174fd0>,
<matplotlib.axis.XTick at 0x1ccdaf7db50>,
<matplotlib.axis.XTick at 0x1ccdb1bd810>,
<matplotlib.axis.XTick at 0x1ccdb1bea50>,
<matplotlib.axis.XTick at 0x1ccdb1c9a10>,
<matplotlib.axis.XTick at 0x1ccdb1cba90>,
<matplotlib.axis.XTick at 0x1ccdb1d1b10>,
<matplotlib.axis.XTick at 0x1ccdb1d3bd0>,
<matplotlib.axis.XTick at 0x1ccdaf09f10>],
[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



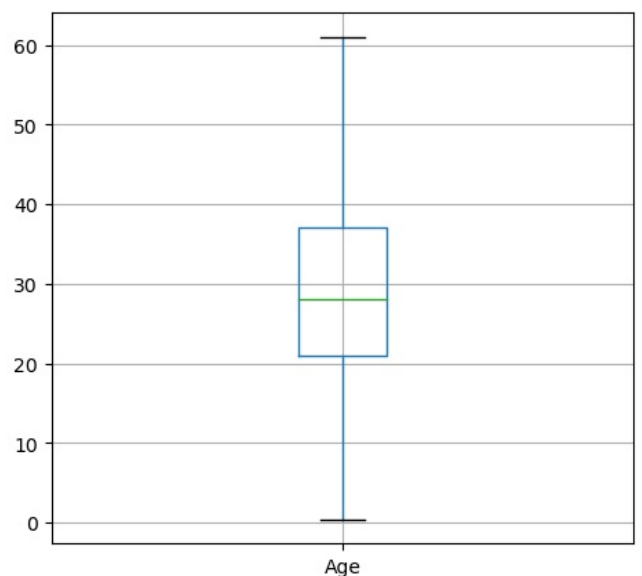
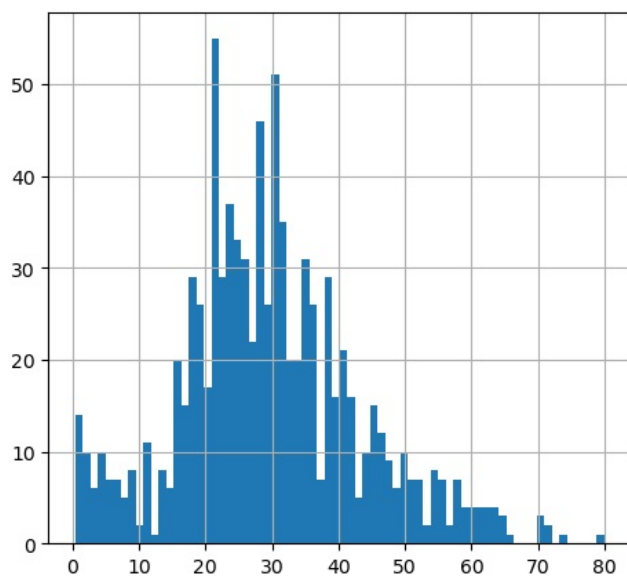
The histogram of the age groups of the people that couldn't survive



```
In [36]: plt.figure(figsize=(12,5))
plt.subplot(121)
train_data['Age'].hist(bins=70)

plt.subplot(122)
train_data.boxplot(column='Age', showfliers=False)
```

Out[36]: <Axes: >



```
In [37]: import seaborn as sns
```

```
In [38]: facet = sns.FacetGrid(train_data,hue='Survived',aspect=4)
facet.map(sns.kdeplot, 'Age', shade=True)
facet.set(xlim=(0, train_data['Age'].max()))
facet.add_legend()
```



```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\axisgrid.py:848: FutureWarning:
```

```
`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.
```

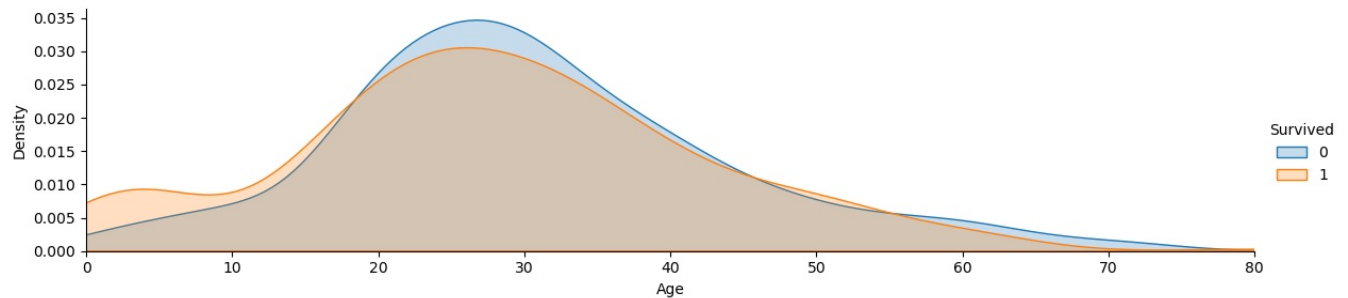
```
func(*plot_args, **plot_kwargs)
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\axisgrid.py:848: FutureWarning:
```

```
`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.
```

```
func(*plot_args, **plot_kwargs)
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed
to tight
```

```
self.figure.tight_layout(*args, **kwargs)
<seaborn.axisgrid.FacetGrid at 0x1ccdc8f52d0>
```

Out[38]:



```
In [39]: train_data[["SibSp", "Survived"]].groupby(['SibSp'], as_index=False).mean().sort_values(by='Survived', ascending=False)
```

Out[39]:

	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 [40]: train_data[["Pclass", "Survived"]].groupby(['Pclass'], as_index=False).mean().sort_values(by='Survived', ascending=False)
```

Out[40]:

	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

```
In [41]: train_data[["Age", "Survived"]].groupby(['Age'], as_index=False).mean().sort_values(by='Age', ascending=True)
```

Out[41]:

	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
...	...	...
171	70.00	0.0
172	70.50	0.0
173	71.00	0.0
174	74.00	0.0
175	80.00	1.0

176 rows × 2 columns

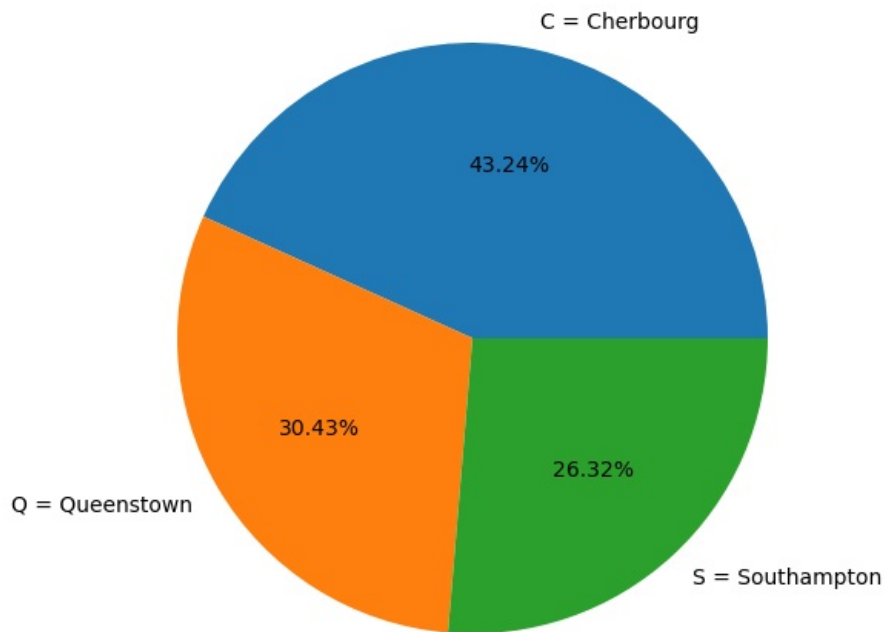
```
In [42]: train_data[["Embarked", "Survived"]].groupby(['Embarked'], as_index=False).mean().sort_values(by='Survived', ascending=False)
```

Out[42]:

	Embarked	Survived
0	C	0.553571
1	Q	0.389610
2	S	0.336957

0	C	0.553571
1	Q	0.389610
2	S	0.336957

```
In [43]: 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 [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js