Lab - 3 - Data Exploration

1) First, you need to read the titanic dataset from local disk and display first five records

In [3]:

```
import pandas as pd
```

In [4]:

```
data=pd.read_csv("titanic.csv")
```

In [5]:

```
data.head(5)
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cal
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	N
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	N
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	N

2) Identify Nominal, Ordinal, Binary and Numeric attributes from data sets and display all values.

In [6]:

```
print("Nominal")
data["Name"]
```

```
Nominal
```

Out[6]:

```
0                                Braund, Mr. Owen Harris
1      Cumings, Mrs. John Bradley (Florence Briggs Th...
2                                 Heikkinen, Miss. Laina
3           Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                               Allen, Mr. William Henry
                             ...
886                                Montvila, Rev. Juozas
887                         Graham, Miss. Margaret Edith
888             Johnston, Miss. Catherine Helen "Carrie"
889                                Behr, Mr. Karl Howell
890                                  Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```

In [7]:

```
print("Odinal")
data["Pclass"]
```

```
Odinal
```

Out[7]:

```
0      3
1      1
2      3
3      1
4      3
      ..
886    2
887    1
888    3
889    1
890    3
Name: Pclass, Length: 891, dtype: int64
```

In [8]:

```
print("Binary")
data["Sex"]
```

```
Binary
```

Out[8]:

```
0        male
1      female
2      female
3      female
4        male
       ...
886      male
887    female
888    female
889      male
890      male
Name: Sex, Length: 891, dtype: object
```

In [9]:

```
print("Numeric")
data["PassengerId"]
```

```
Numeric
```

Out[9]:

```
0        1
1        2
2        3
3        4
4        5
      ...
886    887
887    888
888    889
889    890
890    891
Name: PassengerId, Length: 891, dtype: int64
```

3) Identify symmetric and asymmetric binary attributes from data sets and display all values.

In [13]:

```
print("symmetric")
data["Sex"]
```

```
symmetric
```

Out[13]:

```
0        male
1      female
2      female
3      female
4        male
       ...
886      male
887    female
888    female
889      male
890      male
Name: Sex, Length: 891, dtype: object
```

In [14]:

```
print("asymmetric")
data["Survived"]
```

```
asymmetric
```

Out[14]:

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

4) For each quantitative attribute, calculate its average, standard deviation, minimum, mode, range and maximum values.

In [25]:

```
from pandas.api.types import is_numeric_dtype

for col in data.columns:
    if is_numeric_dtype(data[col]):
        print('%s' % (col))
        print('\t Mean = %.2f' % data[col].mean())
        print("\t Standerd deviation = %.2f" % data[col].std())
        print('\t Minimum = %.2f' % data[col].min())
        print('\t Maximum = %.2f' % data[col].max())
        print('\t Range = %.2f' % (data[col].max()-data[col].min()))
        if col!="PassengerId":
            print("\t Mode : ",data[col].mode()[0])
```

```
PassengerId
         Mean = 446.00
         Standerd deviation = 257.35
         Minimum = 1.00
         Maximum = 891.00
         Range = 890.00
Survived
         Mean = 0.38
         Standerd deviation = 0.49
         Minimum = 0.00
         Maximum = 1.00
         Range = 1.00
         Mode :  0
Pclass
         Mean = 2.31
         Standerd deviation = 0.84
         Minimum = 1.00
         Maximum = 3.00
         Range = 2.00
         Mode :  3
Age
         Mean = 29.70
         Standerd deviation = 14.53
         Minimum = 0.42
         Maximum = 80.00
         Range = 79.58
         Mode :  24.0
SibSp
         Mean = 0.52
         Standerd deviation = 1.10
         Minimum = 0.00
         Maximum = 8.00
         Range = 8.00
         Mode :  0
Parch
         Mean = 0.38
         Standerd deviation = 0.81
         Minimum = 0.00
         Maximum = 6.00
         Range = 6.00
         Mode :  0
Fare
         Mean = 32.20
         Standerd deviation = 49.69
         Minimum = 0.00
         Maximum = 512.33
         Range = 512.33
         Mode :  8.05
```

6) For the qualitative attribute (class), count the frequency for each of its distinct values.

In [30]:

```
data['PassengerId'].value_counts()
```

Out[30]:

```
1      1
599    1
588    1
589    1
590    1
      ..
301    1
302    1
303    1
304    1
891    1
Name: PassengerId, Length: 891, dtype: int64
```

In [31]:

```
data['Pclass'].value_counts()
```

Out[31]:

```
3    491
1    216
2    184
Name: Pclass, dtype: int64
```

7) It is also possible to display the summary for all the attributes simultaneously in a table using the describe() function. If an attribute is quantitative, it will display its mean, standard deviation and various quantiles (including minimum, median, and maximum) values. If an attribute is qualitative, it will display its number of unique values and the top (most frequent) values.

In [35]:

```
data.describe(include='all')
```

Out[35]:

	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

8) For multivariate statistics, you can compute the covariance and correlation between pairs of attributes.

In [37]:

```
data.cov()
```

Out[37]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	66231.000000	-0.626966	-7.561798	138.696504	-16.325843	-0.342697	161.883369
Survived	-0.626966	0.236772	-0.137703	-0.551296	-0.018954	0.032017	6.221787
Pclass	-7.561798	-0.137703	0.699015	-4.496004	0.076599	0.012429	-22.830196
Age	138.696504	-0.551296	-4.496004	211.019125	-4.163334	-2.344191	73.849030
SibSp	-16.325843	-0.018954	0.076599	-4.163334	1.216043	0.368739	8.748734
Parch	-0.342697	0.032017	0.012429	-2.344191	0.368739	0.649728	8.661052
Fare	161.883369	6.221787	-22.830196	73.849030	8.748734	8.661052	2469.436846

In [40]:

```
data.corr()
```

Out[40]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

9) Display the histogram for Age attribute by discretizing it into 8 separate bins and counting the frequency for each bin.

In [41]:

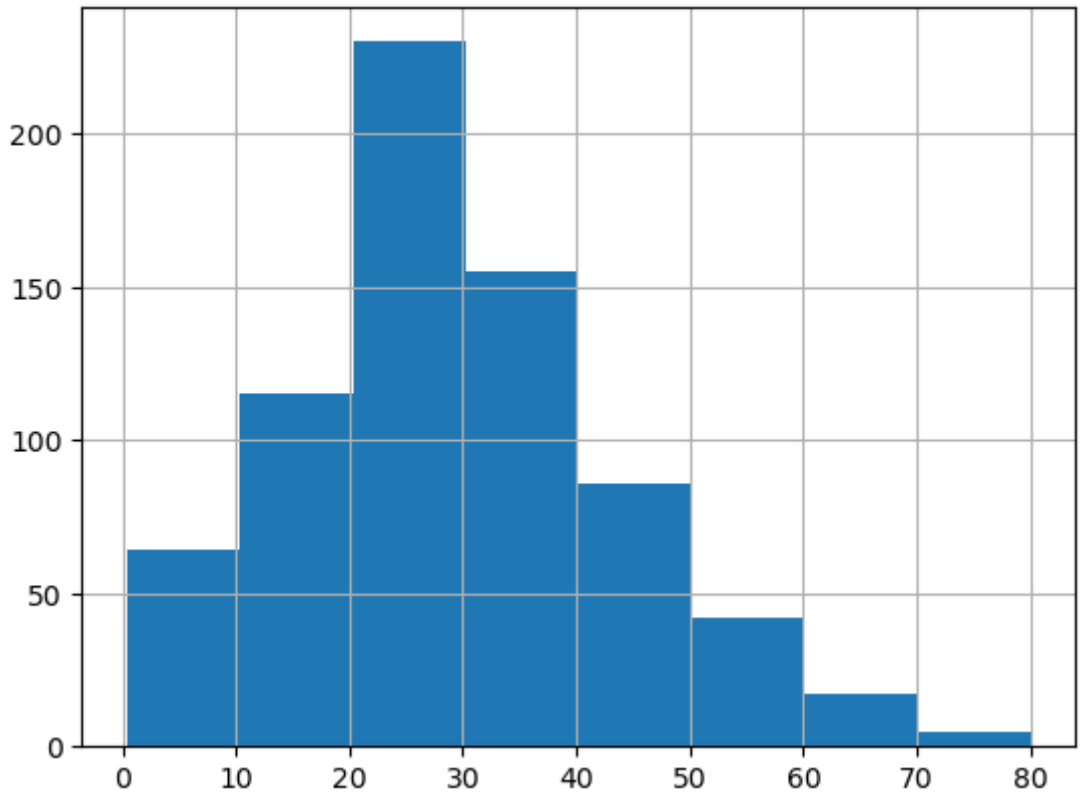
```
import matplotlib.pyplot as plt
```

In [44]:

```
data['Age'].hist(bins=8)
```

Out[44]:

```
<AxesSubplot:>
```



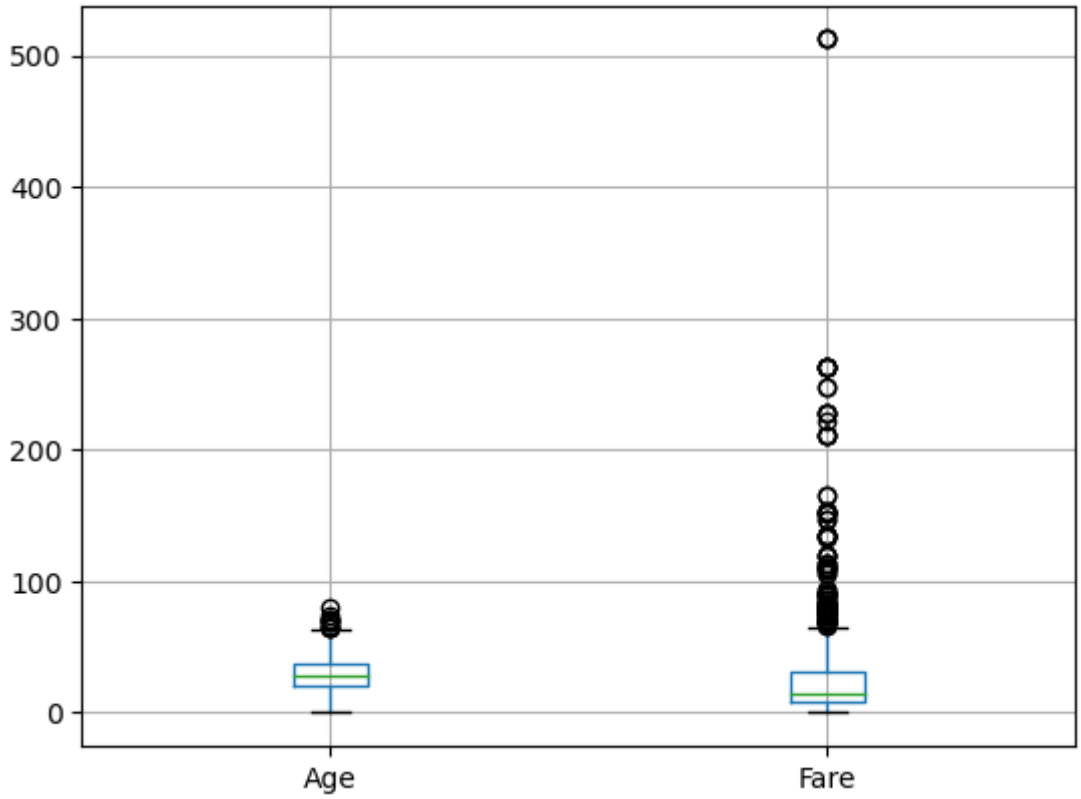
10) A boxplot can also be used to show the distribution of values for each attribute.

In [50]:

```
data[['Age','Fare']].boxplot()
```

Out[50]:

<AxesSubplot:>



11) Display scatter plot for any 5 pair of attributes , we can use a scatter plot to visualize their joint distribution.

In [55]:

```
plt.scatter(data['Age'],data['Fare'])
```

Out[55]:

```
<matplotlib.collections.PathCollection at 0x7fc87805f220>
```