[(https://www.darshan.ac.in/)](https://www.darshan.ac.in/)

Data Mining

Lab - 3

Name : Gohel krish Vishalbhai

EnrNo : 22010101060

Roll No. : 216

batch : A6

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

In [1]:
```
import pandas as pd
```

In [3]:
```
df = pd.read_csv("titanic.csv")
df
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500
...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500

891 rows × 12 columns

In [4]:
```
df.head(5)
```

Out[4]:

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

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

In [7]:
```
print("Nominal")

print(df['PassengerId'])
print()
print(df['Name'])
print()
print(df['Embarked'])
print()
print(df['Ticket'])
print()
print(df['Cabin'])
print()
```

```
Nominal
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

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

0      S
1      C
2      S
3      S
4      S
      ..
886    S
887    S
888    S
889    C
890    Q
Name: Embarked, Length: 891, dtype: object

0             A/5 21171
1              PC 17599
2      STON/O2. 3101282
3                113803
4                373450
             ...
886              211536
887              112053
888          W./C. 6607
889              111369
890              370376
Name: Ticket, Length: 891, dtype: object

0       NaN
1       C85
2       NaN
3      C123
```

```
4        NaN
        ...
886      NaN
887      B42
888      NaN
889     C148
890      NaN
Name: Cabin, Length: 891, dtype: object
```

In [9]:
```
print("Ordinal")

print(df["Pclass"])
```

```
Ordinal
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 [10]:
```
print("Binary")

print(df['Sex'])
print()
print(df['Survived'])
```

```
Binary
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

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
```

In [17]:
```
print("Numeric")

print(df['PassengerId'])
print()
print(df['Fare'])
print()
print(df['SibSp'])
print()
print(df['Age'])
print()
print(df['Parch'])
print()
```

```
Numeric
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

0       7.2500
1      71.2833
2       7.9250
3      53.1000
4       8.0500
        ...
886    13.0000
887    30.0000
888    23.4500
889    30.0000
890     7.7500
Name: Fare, Length: 891, dtype: float64

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

0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888     NaN
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64

0      0
1      0
2      0
3      0
```

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

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

In [20]:
```
print("symatric")

print(df['Sex'])
```

```
symatric
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 [22]:
```
print("asymatric")

print(df['Survived'])
```

```
asymatric
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 [27]:
```
from pandas.api.types import is_numeric_dtype
for col in df.columns:
    if is_numeric_dtype(df[col].dtype):
        print(col," : ")
        print("\t Max : ",df[col].max())
        print("\t Min : ",df[col].min())
        print("\t SD  : ",df[col].std())
        print("\t Range : ",df[col].max()-df[col].min())
        print("\t Average : ",df[col].mean())
        if col != "PassengerId " :
            print("\t Mode :",df[col].mode()[0])
```

```
PassengerId  :
         Max :  891
         Min :  1
         SD  :  257.3538420152301
         Range :  890
         Average :  446.0
         Mode : 1
Survived  :
         Max :  1
         Min :  0
         SD  :  0.4865924542648585
         Range :  1
         Average :  0.3838383838383838
         Mode : 0
Pclass  :
         Max :  3
         Min :  1
         SD  :  0.8360712409770513
         Range :  2
         Average :  2.308641975308642
         Mode : 3
Age  :
         Max :  80.0
         Min :  0.42
         SD  :  14.526497332334044
         Range :  79.58
         Average :  29.69911764705882
         Mode : 24.0
SibSp  :
         Max :  8
         Min :  0
         SD  :  1.1027434322934275
         Range :  8
         Average :  0.5230078563411896
         Mode : 0
Parch  :
         Max :  6
         Min :  0
         SD  :  0.8060572211299559
         Range :  6
         Average :  0.38159371492704824
         Mode : 0
Fare  :
         Max :  512.3292
         Min :  0.0
         SD  :  49.693428597180905
         Range :  512.3292
         Average :  32.204207968574636
         Mode : 8.05
```

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

In [28]:
```
df.Pclass.value_counts()
```

Out[28]:
```
Pclass
3    491
1    216
2    184
Name: count, 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 [29]:
```
df.describe(include="all")
```

Out[29]:

	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 [36]:
```
# df[['PassengerId','Survived', 'Pclass' , 'Age', 'SibSp','Parch','Fare']].cov
# df[['PassengerId','Survived', 'Pclass' , 'Age', 'SibSp','Parch','Fare']].cor
df.cov(numeric_only=True)
```

Out[36]:

	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 [37]:
```
# df[['PassengerId','Survived', 'Pclass' , 'Age', 'SibSp','Parch','Fare']].cor
df.cov(numeric_only=True)
```

Out[37]:

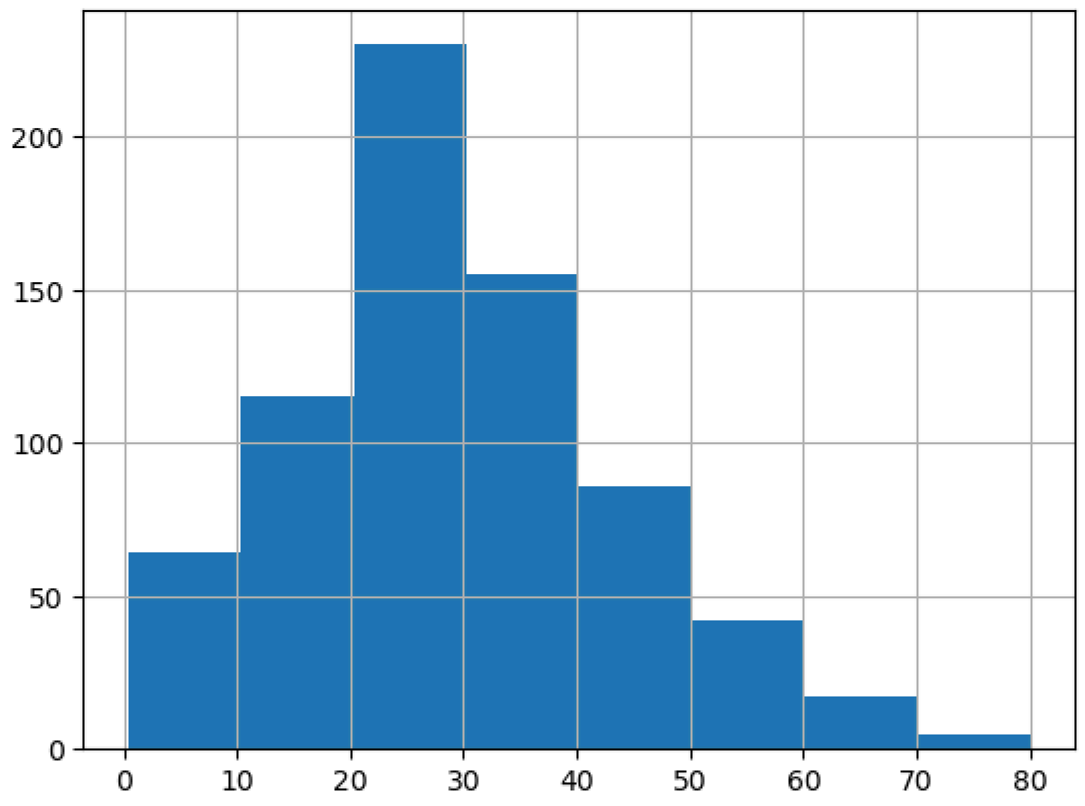
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	66231.000000	-0.626966	-7.561798	138.696504	-16.325843	-0.342697	161.883369
Survived	-0.626966	0.236772	-0.137703	-0.551296	-0.018954	0.032017	6.221787
Pclass	-7.561798	-0.137703	0.699015	-4.496004	0.076599	0.012429	-22.830196
Age	138.696504	-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

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

In [38]:
```
import matplotlib.pyplot as plt
```

In [39]:
```
df['Age'].hist(bins=8)
```

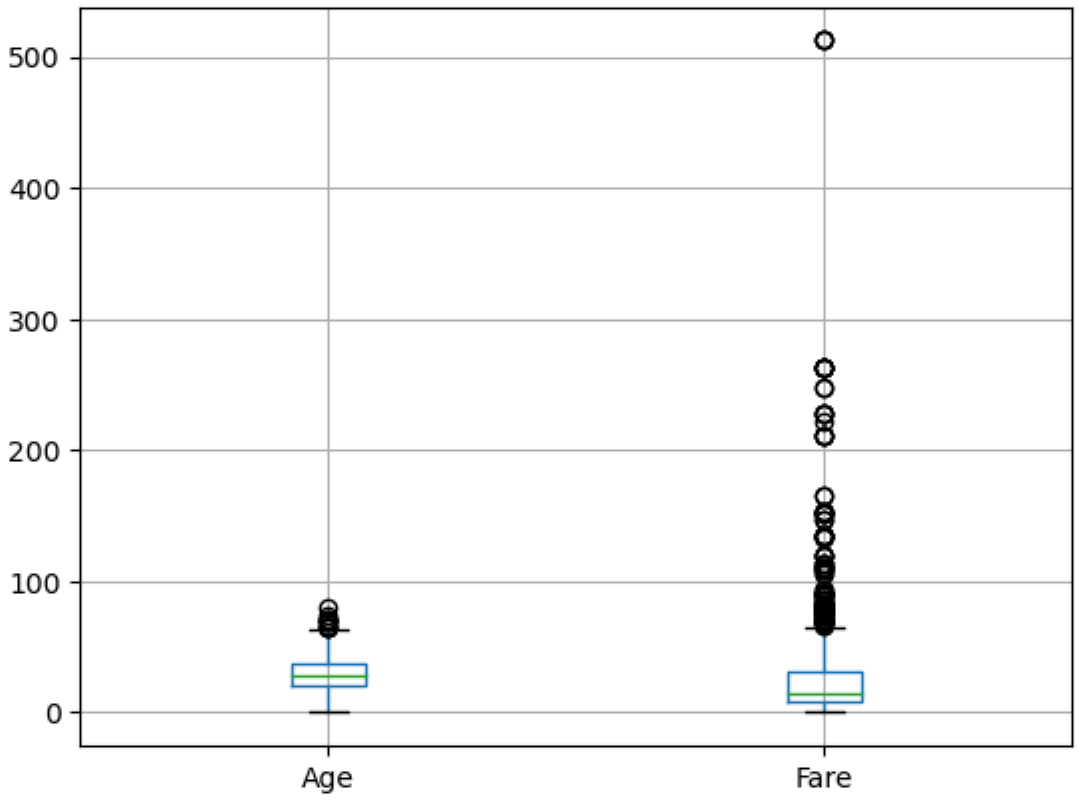
Out[39]:
```
<Axes: >
```



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

```
In [40]: df.boxplot(['Age','Fare'])
```

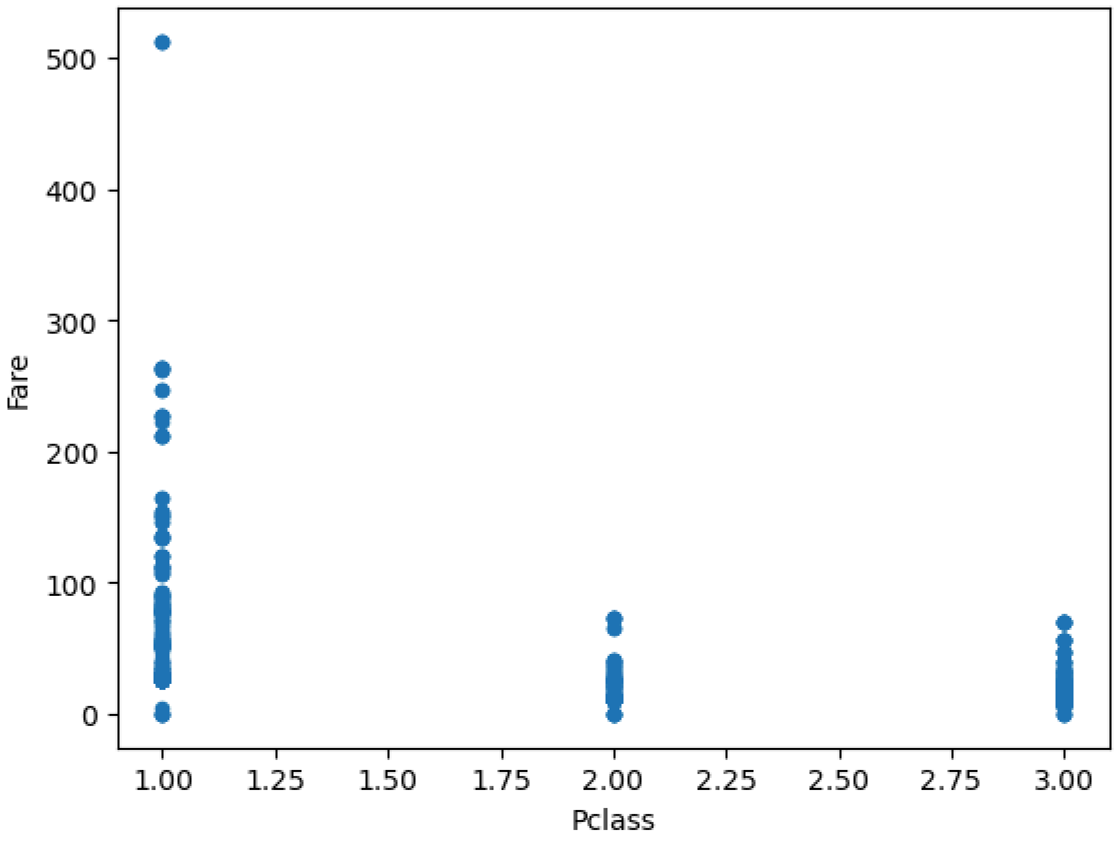
```
Out[40]: <Axes: >
```



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

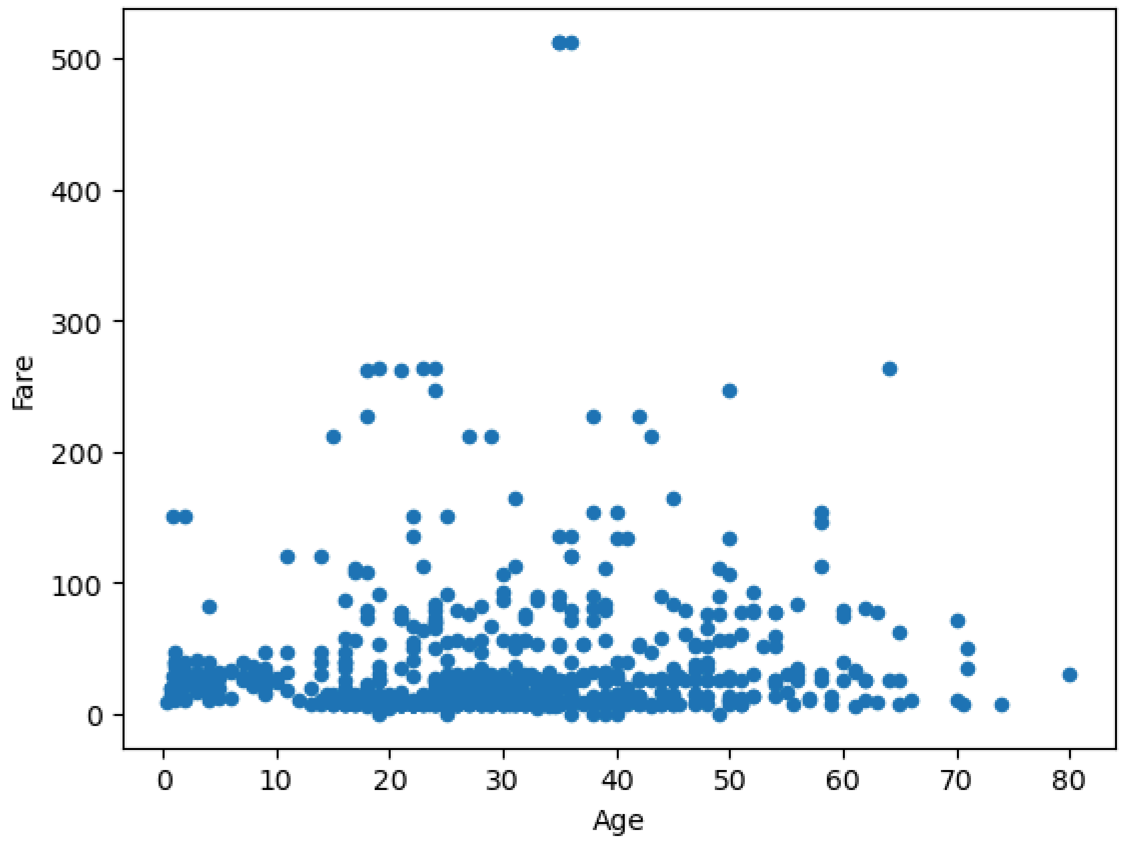
In [41]:
```
df.plot.scatter(x="Pclass",y="Fare")
```

Out[41]:
```
<Axes: xlabel='Pclass', ylabel='Fare'>
```



```
In [42]: df.plot.scatter(x="Age",y="Fare")
```

```
Out[42]: <Axes: xlabel='Age', ylabel='Fare'>
```


Fare
Age