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

```
import pandas as pd
from pandas.api.types import is_numeric_dtype
df = pd.read_csv("titanic.csv") df.head(5)
```

```
                                                 Name     Sex   Age
SibSp  \
                             Braund, Mr. Owen Harris    male  22.0

  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0

                              Heikkinen, Miss. Laina  female  26.0

      Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0

                            Allen, Mr. William Henry    male  35.0

   PassengerId  Survived  Pclass  \
0            1         0
1            2         1
2            3         1
3            4         1
4            5         0
0
1
1
1
2
0
3
1
4
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
```

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

```
print("Nominal:: ")
print(df["Name"])
print(df["PassengerId"])
print(df["Ticket"])
print(df.Cabin)
print(df.Embarked)
print("Ordinal:: ")
print(df["Pclass"])
print(df["PassengerId"])

print("Binary:: ")
print(df["Sex"])
print(df["Survived"])

print("Numeric:: ")
print(df["Fare"])
print(df["Age"])
print(df["Parch"])
print(df["SibSp"])
```

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

```
print("Asymmetric:: ")
print(df["Survived"])

print("Symmetric:: ")
print(df["Sex"])
```

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

```
for i in df.columns:     if
is_numeric_dtype(df[i].dtype):
print(i)
        print("Maximum:: ", df[i].max())
print("Minimum:: ", df[i].min())
        print("Standard Deviation:: ", df[i].std())
print("Mean:: ", df[i].mean())

        if i != "PassengerId":
print("MODE:: ", df[i].mode()[0])

        print("Range:: ", df[i].max() - df[i],min())
```

```
PassengerId:
       Mean = 446.00
       Standard deviation = 257.35
       Minimum = 1.00
       Maximum = 891.00
Survived:
       Mean = 0.38
       Standard deviation = 0.49
       Minimum = 0.00
       Maximum = 1.00
Pclass:
       Mean = 2.31
       Standard deviation = 0.84
       Minimum = 1.00
       Maximum = 3.00
Age:
       Mean = 29.70
       Standard deviation = 14.53
       Minimum = 0.42
       Maximum = 80.00
SibSp:
       Mean = 0.52
       Standard deviation = 1.10
       Minimum = 0.00
       Maximum = 8.00
Parch:
       Mean = 0.38
       Standard deviation = 0.81
       Minimum = 0.00
       Maximum = 6.00
Fare:
       Mean = 32.20
       Standard deviation = 49.69
       Minimum = 0.00
       Maximum = 512.33
```

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

```
df["Pclass"].value_counts()
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.

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

```
        PassengerId    Survived      Pclass                          Name
Sex  \
count    891.000000  891.000000  891.000000                           891
891
unique          NaN         NaN         NaN                           891
2
top             NaN         NaN         NaN  Braund, Mr. Owen Harris
male
freq            NaN         NaN         NaN                             1
577
mean     446.000000    0.383838    2.308642                           NaN
NaN
std      257.353842    0.486592    0.836071                           NaN
NaN
min        1.000000    0.000000    1.000000                           NaN
NaN
25%      223.500000    0.000000    2.000000                           NaN
NaN
50%      446.000000    0.000000    3.000000                           NaN
NaN
75%      668.500000    1.000000    3.000000                           NaN
NaN
max      891.000000    1.000000    3.000000                           NaN
NaN

              Age        SibSp       Parch  Ticket         Fare
Cabin  \
count   714.000000  891.000000  891.000000     891  891.000000
```

```
204
unique          NaN         NaN         NaN     681         NaN
147
top             NaN         NaN         NaN  347082         NaN  B96
B98
freq            NaN         NaN         NaN       7         NaN
4
mean      29.699118    0.523008    0.381594     NaN   32.204208
NaN
std       14.526497    1.102743    0.806057     NaN   49.693429
NaN
min        0.420000    0.000000    0.000000     NaN    0.000000
NaN
25%       20.125000    0.000000    0.000000     NaN    7.910400
NaN
50%       28.000000    0.000000    0.000000     NaN   14.454200
NaN
75%       38.000000    1.000000    0.000000     NaN   31.000000
NaN
max       80.000000    8.000000    6.000000     NaN  512.329200
NaN
```

```
       Embarked
count       889
unique        3
top           S
freq        644
mean        NaN
std         NaN
min         NaN
25%         NaN
50%         NaN
75%         NaN
max         NaN
```

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

```
df.cov(numeric_only = True)
```

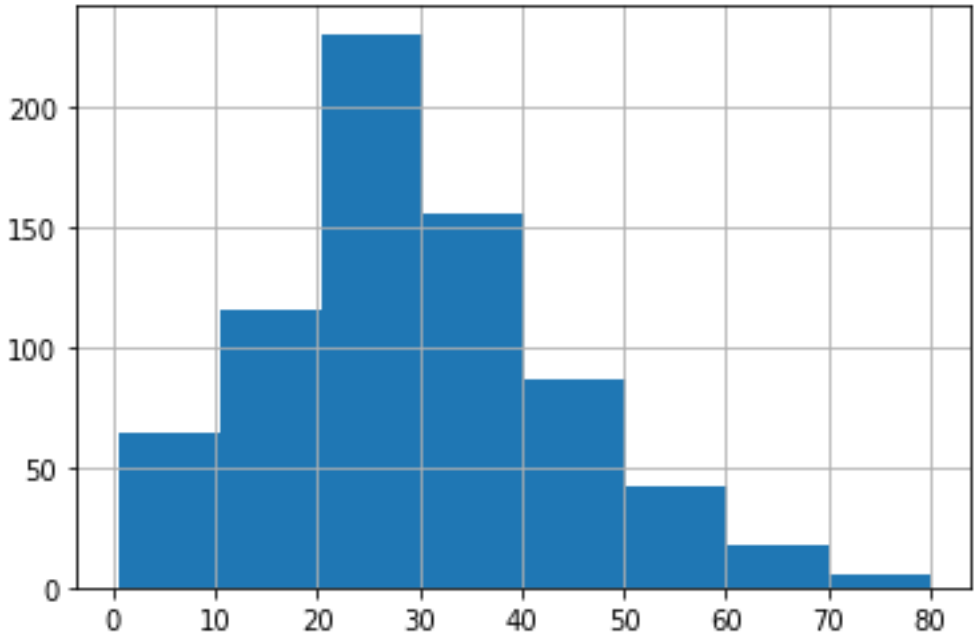
```
              PassengerId  Survived     Pclass          Age       SibSp
PassengerId  66231.000000 -0.626966  -7.561798  138.696504 -16.325843
\

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

                Parch         Fare
PassengerId -0.342697   161.883369
Survived     0.032017     6.221787
Pclass       0.012429   -22.830196
Age         -2.344191    73.849030
SibSp        0.368739     8.748734
Parch        0.649728     8.661052  Fare
8.661052  2469.436846
df.corr(numeric_only = True)
             PassengerId  Survived    Pclass       Age     SibSp
Parch  \
PassengerId     1.000000 -0.005007 -0.035144  0.036847 -0.057527 -
0.001652
Survived       -0.005007  1.000000 -0.338481 -0.077221 -0.035322
0.081629
Pclass         -0.035144 -0.338481  1.000000 -0.369226  0.083081
0.018443
Age             0.036847 -0.077221 -0.369226  1.000000 -0.308247 -
0.189119
SibSp          -0.057527 -0.035322  0.083081 -0.308247  1.000000
0.414838
Parch          -0.001652  0.081629  0.018443 -0.189119  0.414838
1.000000
Fare            0.012658  0.257307 -0.549500  0.096067  0.159651
0.216225
```

```
                Fare
PassengerId  0.012658
Survived     0.257307
Pclass      -0.549500
Age          0.096067
SibSp        0.159651
Parch        0.216225
Fare         1.000000
```

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

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

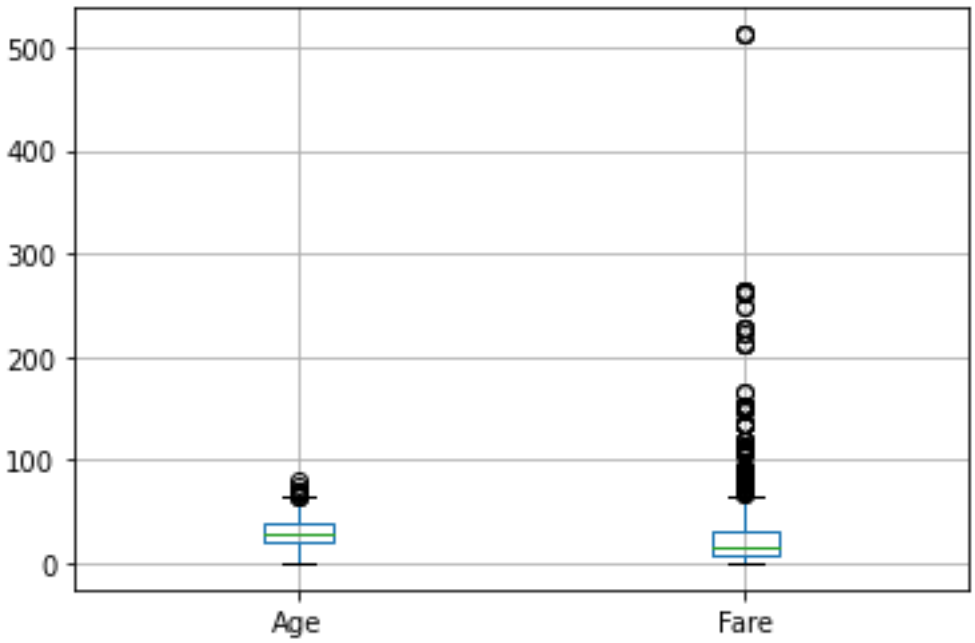
```
<AxesSubplot:>
```



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

```
df.boxplot(column=['Age','Fare'])
```

```
<AxesSubplot:>
```



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

```
df.plot.scatter(x='Age', y='Fare')
```

```
<matplotlib.collections.PathCollection at 0x7f939a5eadf0>
```