Lab - 2 - Data Exploration

Step 1. Import the necessary libraries

```
import pandas as pd
```

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called users and use the 'user_id' as index

```
df = pd.read_csv("users.txt",sep="|",index_col=[0])
```

Step 4. See the first 25 entries

```
df.head(25)
```

```
          age gender     occupation zip_code
user_id                                     
1          24      M     technician    85711
2          53      F          other    94043
3          23      M         writer    32067
4          24      M     technician    43537
5          33      F          other    15213
6          42      M      executive    98101
7          57      M  administrator    91344
8          36      M  administrator    05201
9          29      M        student    01002
10         53      M         lawyer    90703
11         39      F          other    30329
12         28      F          other    06405
13         47      M       educator    29206
14         45      M      scientist    55106
15         49      F       educator    97301
16         21      M  entertainment    10309
17         30      M     programmer    06355
18         35      F          other    37212
19         40      M      librarian    02138
20         42      F      homemaker    95660
21         26      M         writer    30068
22         25      M         writer    40206
23         30      F         artist    48197
24         21      F         artist    94533
25         39      M       engineer    55107
```

Step 5. See the last 10 entries

```
df.tail(10)
```

```
         age gender     occupation zip_code
user_id
934       61      M       engineer    22902
935       42      M         doctor    66221
936       24      M          other    32789
937       48      M       educator    98072
938       38      F     technician    55038
939       26      F        student    33319
940       32      M  administrator    02215
941       20      M        student    97229
942       48      F      librarian    78209
943       22      M        student    77841
```

Step 6. What is the number of observations in the dataset?

```
 #0 indicates row

df.shape[0]
```

```
943
```

Step 7. What is the number of columns in the dataset?

```
#1 indicates column

df.shape[1]
```

```
4
```

Step 8. Print the name of all the columns.

```
df.columns
```

```
Index(['age', 'gender', 'occupation', 'zip_code'], dtype='object')
```

Step 9. How is the dataset indexed?

```
# "the index" (aka "the labels")

df.index
```

```
Int64Index([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,
            ...
            934, 935, 936, 937, 938, 939, 940, 941, 942, 943],
           dtype='int64', name='user_id', length=943)
```

Step 10. What is the data type of each column?

```
df.dtypes
```

```
age            int64
gender        object
occupation    object
zip_code      object
dtype: object
```

Step 11. Print only the occupation column

```
df["occupation"]
```

```
user_id
1         technician
2              other
3             writer
4         technician
5              other
           ...
939          student
940    administrator
941          student
942        librarian
943          student
Name: occupation, Length: 943, dtype: object
```

Step 12. How many different occupations are in this dataset?

```
 #print number of unique occupation

print(df.occupation.nunique())


#print all unique occupation in tuple

print(df.occupation.unique())
```

```
21
['technician' 'other' 'writer' 'executive' 'administrator' 'student'
 'lawyer' 'educator' 'scientist' 'entertainment' 'programmer'
'librarian'
 'homemaker' 'artist' 'engineer' 'marketing' 'none' 'healthcare'
'retired'
 'salesman' 'doctor']
```

Step 13. What is the most frequent occupation?

```
'student'
```

Step 14. Summarize the DataFrame.

```
df.describe()
```

```
              age
count  943.000000
mean    34.051962
std     12.192740
min      7.000000
25%     25.000000
50%     31.000000
75%     43.000000
max     73.000000
```

Step 15. Summarize all the columns

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

```
               age gender occupation zip_code
count   943.000000    943        943      943
unique         NaN      2         21      795
top            NaN      M    student    55414
freq           NaN    670        196        9
mean     34.051962    NaN        NaN      NaN
std      12.192740    NaN        NaN      NaN
min       7.000000    NaN        NaN      NaN
25%      25.000000    NaN        NaN      NaN
50%      31.000000    NaN        NaN      NaN
75%      43.000000    NaN        NaN      NaN
max      73.000000    NaN        NaN      NaN
```

Step 16. Summarize only the occupation column

```
df.occupation.describe()
```

```
count         943
unique         21
top       student
freq          196
Name: occupation, dtype: object
```

Step 17. What is the mean age of users?

```
round(df.age.mean())
```

```
34
```

Step 18. What is the age with least occurrence?

```
df.age.value_counts().tail()
```

```
7     1
66    1
11    1
10    1
73    1
Name: age, dtype: int64
```

Lab - 3 - Data Exploration

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

```
import pandas as pd

df = pd.read_csv("titanic.csv")

df.head(5)

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

   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["Ticket"])
print(df["Embarked"])
print(df["Cabin"])
print("Ordinal")
print(df["Pclass"])
print("Binary")
print(df["Sex"])
print(df["Survived"])
print("Numeric")
print(df["SibSp"])
print(df["PassengerId"])
print(df["Fare"])
print(df["Age"])
print(df["Parch"])
```

```
Nominal
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             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      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       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
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
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
Numeric
```

```
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        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      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.

```
print("Symmetric",df["Sex"])
print("Asymmetric",df["Survived"])
```

```
Symmetric 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
Asymmetric 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.

```
from pandas.api.types import is_numeric_dtype
for column in df.columns:
    if(is_numeric_dtype(df[column])):
```

```python
        print(column,":")
        print("\tMean : ",df[column].mean())
        print("\tStandard Deviation : ",df[column].std())
        print("\tMinimum : ",df[column].min())
        print("\tRange : ",df[column].max()-df[column].min())
        print("\tMax : ",df[column].max())
        if column!="PassengerId":
            print("\tMode : ",df[column].mode()[0])
```

```
PassengerId :
	Mean :  446.0
	Standard Deviation :  257.3538420152301
	Minimum :  1
	Range :  890
	Max :  891
Survived :
	Mean :  0.3838383838383838
	Standard Deviation :  0.4865924542648585
	Minimum :  0
	Range :  1
	Max :  1
	Mode :  0
Pclass :
	Mean :  2.308641975308642
	Standard Deviation :  0.8360712409770513
	Minimum :  1
	Range :  2
	Max :  3
	Mode :  3
Age :
	Mean :  29.69911764705882
	Standard Deviation :  14.526497332334044
	Minimum :  0.42
	Range :  79.58
	Max :  80.0
	Mode :  24.0
SibSp :
	Mean :  0.5230078563411896
	Standard Deviation :  1.1027434322934275
	Minimum :  0
	Range :  8
	Max :  8
	Mode :  0
Parch :
	Mean :  0.38159371492704824
	Standard Deviation :  0.8060572211299559
	Minimum :  0
	Range :  6
	Max :  6
	Mode :  0
```

```
Fare :
     Mean :  32.2042079685746
     Standard Deviation :  49.693428597180905
     Minimum :  0.0
     Range :  512.3292
     Max :  512.3292
     Mode :  8.05
```

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()
```

```
              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()
```

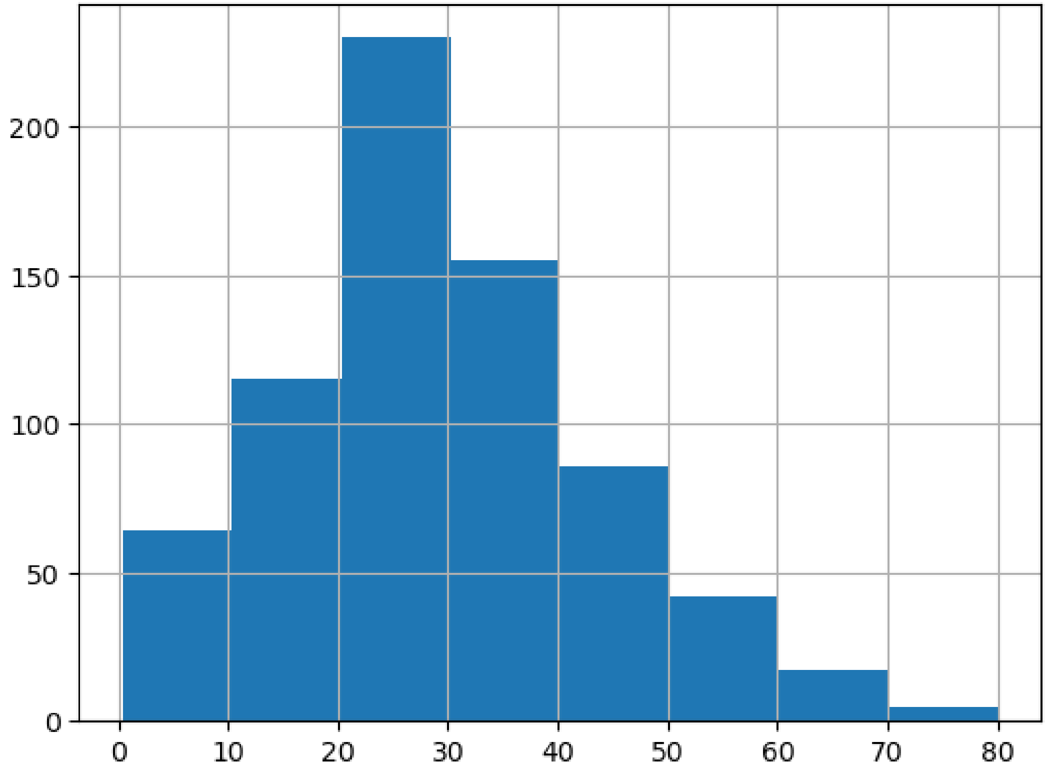
```
             PassengerId  Survived    Pclass       Age     SibSp
Parch  \
PassengerId     1.000000 -0.005007 -0.035144  0.036847 -0.057527 -
0.001652
Survived       -0.005007  1.000000 -0.338481 -0.077221 -0.035322
0.081629
Pclass         -0.035144 -0.338481  1.000000 -0.369226  0.083081
0.018443
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)
```

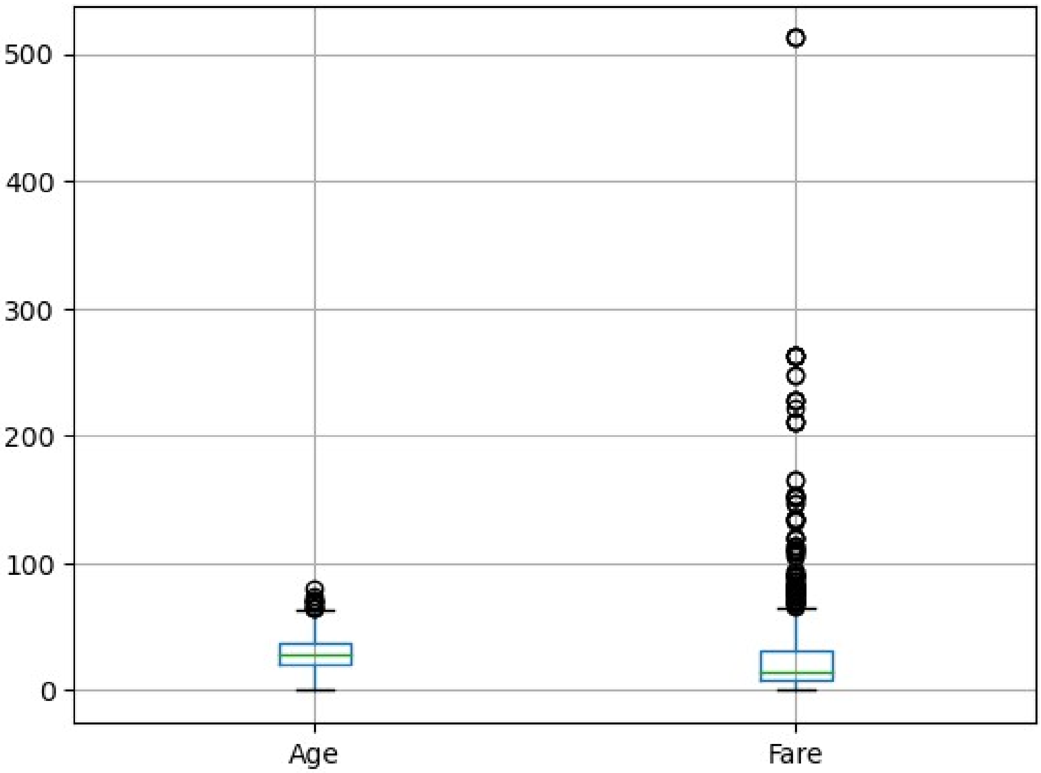
```
<AxesSubplot:>
```



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

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

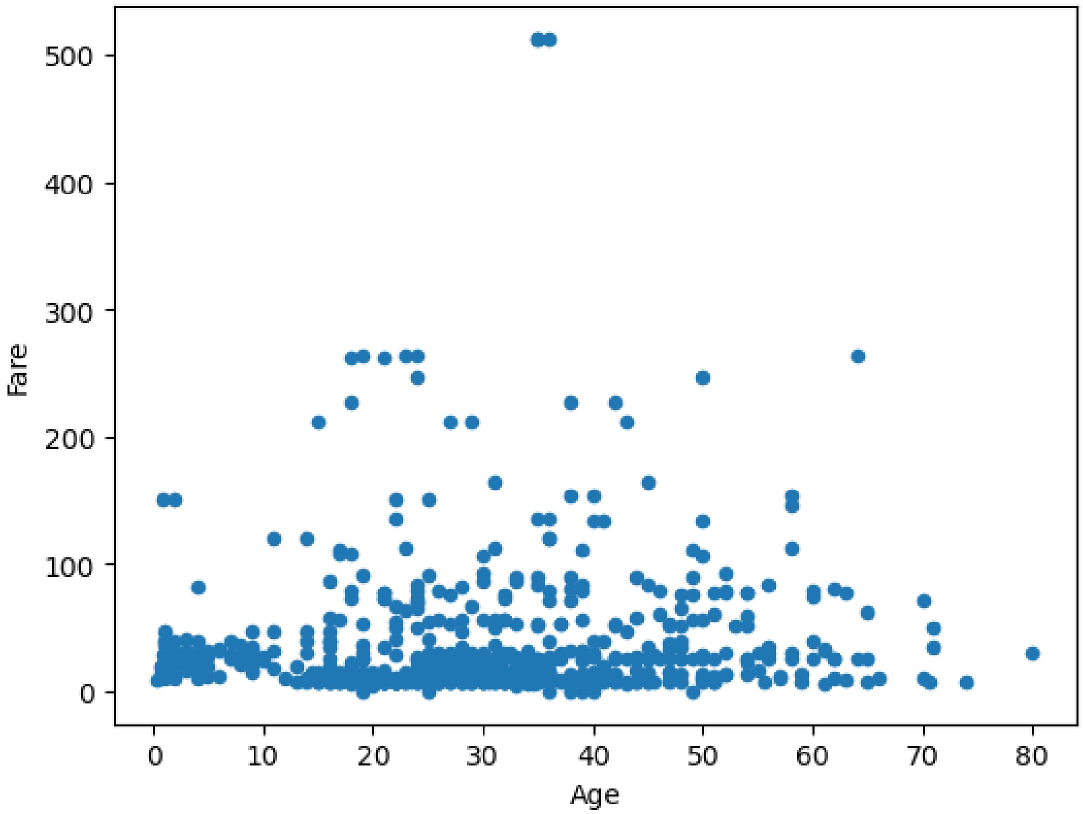
```
<AxesSubplot:>
```



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")
```

```
<AxesSubplot:xlabel='Age', ylabel='Fare'>
```


0
100
200
300
400
500
Fare
0
10
20
30
40
50
60
70
80
Age

Lab - 4 - Data Preprocessing

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

```
import pandas as pd

df = pd.read_csv("titanic.csv")

df.tail(5)
```

```
     PassengerId  Survived  Pclass
Name  \
886          887         0       2                      Montvila, Rev.
Juozas
887          888         1       1               Graham, Miss. Margaret
Edith
888          889         0       3  Johnston, Miss. Catherine Helen
"Carrie"
889          890         1       1                      Behr, Mr. Karl
Howell
890          891         0       3                        Dooley, Mr.
Patrick

        Sex   Age  SibSp  Parch      Ticket   Fare Cabin Embarked
886    male  27.0      0      0      211536  13.00   NaN        S
887  female  19.0      0      0      112053  30.00   B42        S
888  female   NaN      1      2  W./C. 6607  23.45   NaN        S
889    male  26.0      0      0      111369  30.00  C148        C
890    male  32.0      0      0      370376   7.75   NaN        Q
```

2) Handle Missing Values in data set [use dropna(), fillna(), and interpolate]

```
df.isnull().sum()
```

```
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
```

```
df.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
```

```
newdf = 
df.fillna({'Age':df['Age'].mean(),'Cabin':"Y28",'Embarked':"Y"})
```

```
newdf
```

```
     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  \
0                              Braund, Mr. Owen Harris    male  
22.000000   
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  
38.000000   
2                               Heikkinen, Miss. Laina  female  
```

```
26.000000
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female
35.000000
4                             Allen, Mr. William Henry    male
35.000000
..                                                 ...     ...
...
886                              Montvila, Rev. Juozas    male
27.000000
887                       Graham, Miss. Margaret Edith  female
19.000000
888           Johnston, Miss. Catherine Helen "Carrie"  female
29.699118
889                              Behr, Mr. Karl Howell    male
26.000000
890                                Dooley, Mr. Patrick    male
32.000000

     SibSp  Parch            Ticket     Fare Cabin Embarked
0        1      0         A/5 21171   7.2500   Y28        S
1        1      0          PC 17599  71.2833   C85        C
2        0      0  STON/O2. 3101282   7.9250   Y28        S
3        1      0            113803  53.1000  C123        S
4        0      0            373450   8.0500   Y28        S
..     ...    ...               ...      ...   ...      ...
886      0      0            211536  13.0000   Y28        S
887      0      0            112053  30.0000   B42        S
888      1      2        W./C. 6607  23.4500   Y28        S
889      0      0            111369  30.0000  C148        C
890      0      0            370376   7.7500   Y28        Q

[891 rows x 12 columns]
```

```
newdf.isnull().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

```
for i in df['Age']:
    if i<1:
        print(i)

0.83
0.92
0.75
0.75
0.67
0.42
0.83
```

3) Apply Scaling to AGE attribute with min max, decimal scaling and z score.

```
newdf['newAge'] = (newdf['Age']-
newdf['Age'].min())/(newdf['Age'].max()-newdf['Age'].min())

newdf.describe()

       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  891.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   13.002015    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   22.000000    0.000000
50%     446.000000    0.000000    3.000000   29.699118    0.000000
75%     668.500000    1.000000    3.000000   35.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare      newAge
count  891.000000  891.000000  891.000000
mean     0.381594   32.204208    0.367921
std      0.806057   49.693429    0.163383
min      0.000000    0.000000    0.000000
25%      0.000000    7.910400    0.271174
50%      0.000000   14.454200    0.367921
75%      0.000000   31.000000    0.434531
max      6.000000  512.329200    1.000000

newdf = df
newdf['newAge'] = df['Age']/10**len(str(int(df['Age'].max())))

df

     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  newAge
0        0         A/5 21171   7.2500   NaN        S    0.22
1        0          PC 17599  71.2833   C85        C    0.38
2        0  STON/O2. 3101282   7.9250   NaN        S    0.26
3        0            113803  53.1000  C123        S    0.35
4        0            373450   8.0500   NaN        S    0.35
..     ...               ...      ...   ...      ...     ...
886      0            211536  13.0000   NaN        S    0.27
887      0            112053  30.0000   B42        S    0.19
888      2        W./C. 6607  23.4500   NaN        S     NaN
889      0            111369  30.0000  C148        C    0.26
890      0            370376   7.7500   NaN        Q    0.32

[891 rows x 13 columns]

df.corr()
```

```
             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
newAge          0.036847 -0.077221 -0.369226  1.000000 -0.308247 -
0.189119

                 Fare    newAge
PassengerId  0.012658  0.036847
Survived     0.257307 -0.077221
Pclass      -0.549500 -0.369226
Age          0.096067  1.000000
SibSp        0.159651 -0.308247
Parch        0.216225 -0.189119
Fare         1.000000  0.096067
newAge       0.096067  1.000000
```

```
df[['Age','newAge']].corr()
```

```
        Age  newAge
Age     1.0     1.0
newAge  1.0     1.0
```