Data Mining Lab - 1 Rudra - 23010101048

Introduction to Pandas Library Function:

Step-1 Import the pandas Libraries

```
import pandas as pd
```

Step-2 Import the dataset from this:....

Step-3 Read csv or excel File

```
df = pd.read_csv('titanic.csv')
```

Step-4 Print Data from csv or excel File

```
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
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
..     ...               ...      ...   ...      ...
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

Step-5 See the First 10 Rows

```
df.head(10)
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3
5            6         0       3
6            7         0       1
7            8         0       3
8            9         1       3
9           10         1       2
```

```
                                                 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
5                                    Moran, Mr. James     male    NaN
0
6                             McCarthy, Mr. Timothy J     male   54.0
0
7                      Palsson, Master. Gosta Leonard     male    2.0
3
8  Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female   27.0
0
9                 Nasser, Mrs. Nicholas (Adele Achem)  female   14.0
1

   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
5      0            330877   8.4583   NaN        Q
6      0             17463  51.8625   E46        S
7      1            349909  21.0750   NaN        S
8      2            347742  11.1333   NaN        S
9      0            237736  30.0708   NaN        C
```

Step-6 See the Last 10 Rows

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

```
     PassengerId  Survived  Pclass
Name  \
881          882         0       3                           Markun, Mr.
Johann
882          883         0       3               Dahlberg, Miss. Gerda
Ulrika
883          884         0       2             Banfield, Mr. Frederick
James
```

```
884          885         0       3                     Sutehall, Mr.
Henry Jr
885          886         0       3       Rice, Mrs. William (Margaret
Norton)
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
881    male  33.0      0      0            349257   7.8958   NaN
S
882  female  22.0      0      0              7552  10.5167   NaN
S
883    male  28.0      0      0  C.A./SOTON 34068  10.5000   NaN
S
884    male  25.0      0      0   SOTON/OQ 392076   7.0500   NaN
S
885  female  39.0      0      5            382652  29.1250   NaN
Q
886    male  27.0      0      0            211536  13.0000   NaN
S
887  female  19.0      0      0            112053  30.0000   B42
S
888  female   NaN      1      2        W./C. 6607  23.4500   NaN
S
889    male  26.0      0      0            111369  30.0000  C148
C
890    male  32.0      0      0            370376   7.7500   NaN
Q
```

Step-7 Data type of each columns

```
df.dtypes
```

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
```

```
SibSp             int64
Parch             int64
Ticket           object
Fare            float64
Cabin            object
Embarked         object
dtype: object
```

Step-8 Display Summary Information

```
df.describe()
```

```
       PassengerId    Survived      Pclass         Age        SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

```
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
```

Step-9 Access a specific column

```
df['Age']
```

```
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
```

```
df.Age
```

```
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
```

Step-10 Access rows by their integer location

```
#accessing from index 25
df.iloc[25]
```

```
PassengerId                                                 26
Survived                                                     1
Pclass                                                       3
Name            Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...
```

```
Sex                                                          female
Age                                                            38.0
SibSp                                                             1
Parch                                                             5
Ticket                                                       347077
Fare                                                        31.3875
Cabin                                                           NaN
Embarked                                                          S
Name: 25, dtype: object
```

Step-11 Delete a specific Column

```
df.drop("Age", axis = 1)
#to delete column axis = 1, only delete for particular cell (not
permanently)
#to delete row axis = 0
#to delete permenantly set inplace = True
```

```
     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  SibSp
Parch  \
0                              Braund, Mr. Owen Harris     male      1
0
1    Cumings, Mrs. John Bradley (Florence Briggs Th...   female      1
0
2                               Heikkinen, Miss. Laina   female      0
0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)   female      1
0
4                             Allen, Mr. William Henry     male      0
0
..                                                 ...      ...    ...
...
886                              Montvila, Rev. Juozas     male      0
0
887                       Graham, Miss. Margaret Edith   female      0
```

```
0
888                Johnston, Miss. Catherine Helen "Carrie"
2
889                                   Behr, Mr. Karl Howell
0
890                                     Dooley, Mr. Patrick
0

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

[891 rows x 11 columns]
```

```
df.drop("Age", axis = 1, inplace = True)
```

```
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  SibSp
Parch  \
0                              Braund, Mr. Owen Harris    male      1
0
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female      1
0
2                               Heikkinen, Miss. Laina  female      0
0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female      1
0
```

```
4                                Allen, Mr. William Henry    male       0
0
..                                                    ...     ...     ...
...
886                                 Montvila, Rev. Juozas    male       0
0
887                          Graham, Miss. Margaret Edith  female       0
0
888              Johnston,  Miss. Catherine Helen "Carrie"  female       1
2
889                                 Behr, Mr. Karl Howell    male       0
0
890                                   Dooley, Mr. Patrick    male       0
0

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

[891 rows x 11 columns]
```

Step-12 Create a new Column

```
df['Amount'] = df['Fare']*100    #multiply fare to 100 and store value
in amount column
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   SibSp
Parch  \
0                              Braund, Mr. Owen Harris     male       1
0
1    Cumings, Mrs. John Bradley (Florence Briggs Th...   female       1
0
2                               Heikkinen, Miss. Laina   female       0
0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)   female       1
0
4                           Allen, Mr. William Henry     male       0
0
..                                                 ...      ...     ...
...
886                            Montvila, Rev. Juozas     male       0
0
887                      Graham, Miss. Margaret Edith   female       0
0
888          Johnston, Miss. Catherine Helen "Carrie"   female       1
2
889                              Behr, Mr. Karl Howell     male       0
0
890                                Dooley, Mr. Patrick     male       0
0

                Ticket      Fare Cabin Embarked    Amount
0            A/5 21171    7.2500   NaN        S    725.00
1             PC 17599   71.2833   C85        C   7128.33
2    STON/O2. 3101282    7.9250   NaN        S    792.50
3               113803   53.1000  C123        S   5310.00
4               373450    8.0500   NaN        S    805.00
..                 ...       ...   ...      ...       ...
886             211536   13.0000   NaN        S   1300.00
887             112053   30.0000   B42        S   3000.00
888         W./C. 6607   23.4500   NaN        S   2345.00
889             111369   30.0000  C148        C   3000.00
890             370376    7.7500   NaN        Q    775.00

[891 rows x 12 columns]
```

Step-13 Perform Condition Selection on DataFrame

```
df[df['Fare']>20]
```

```
                                                  Name     Sex  SibSp
Parch  \
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female      1
0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female      1
0
6                              McCarthy, Mr. Timothy J    male      0
0
7                       Palsson, Master. Gosta Leonard    male      3
1
9                    Nasser, Mrs. Nicholas (Adele Achem)  female      1
0
..                                                  ...     ...    ...
...
880       Shelley, Mrs. William (Imanita Parrish Hall)  female      0
1
885                 Rice, Mrs. William (Margaret Norton)  female      0
5
887                        Graham, Miss. Margaret Edith  female      0
0
888            Johnston, Miss. Catherine Helen "Carrie"  female      1
2
889                                Behr, Mr. Karl Howell    male      0
0

          Ticket     Fare Cabin Embarked   Amount
1      PC 17599  71.2833   C85        C  7128.33
3         113803  53.1000  C123        S  5310.00
6          17463  51.8625   E46        S  5186.25
7         349909  21.0750   NaN        S  2107.50
9         237736  30.0708   NaN        C  3007.08
..           ...      ...   ...      ...      ...
880       230433  26.0000   NaN        S  2600.00
885       382652  29.1250   NaN        Q  2912.50
887       112053  30.0000   B42        S  3000.00
888  W./C. 6607  23.4500   NaN        S  2345.00
889       111369  30.0000  C148        C  3000.00
```

```
[376 rows x 12 columns]
```

```
df[df['Sex'] == 'female']
```

```
     PassengerId  Survived  Pclass  \
1              2         1       1
2              3         1       3
3              4         1       1
8              9         1       3
9             10         1       2
..           ...       ...     ...
880           881         1       2
882           883         0       3
885           886         0       3
887           888         1       1
888           889         0       3

                                                  Name     Sex   SibSp
Parch  \
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female       1
0
2                                Heikkinen, Miss. Laina  female       0
0
3          Futrelle, Mrs. Jacques Heath (Lily May Peel)  female       1
0
8    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female       0
2
9                  Nasser, Mrs. Nicholas (Adele Achem)  female       1
0
..                                                 ...     ...     ...
...
880       Shelley, Mrs. William (Imanita Parrish Hall)  female       0
1
882                          Dahlberg, Miss. Gerda Ulrika  female       0
0
885                Rice, Mrs. William (Margaret Norton)  female       0
5
887                          Graham, Miss. Margaret Edith  female       0
0
888           Johnston, Miss. Catherine Helen "Carrie"  female       1
2

                Ticket     Fare Cabin Embarked   Amount
1            PC 17599  71.2833   C85        C  7128.33
2    STON/O2. 3101282   7.9250   NaN        S   792.50
3               113803  53.1000  C123        S  5310.00
8               347742  11.1333   NaN        S  1113.33
9               237736  30.0708   NaN        C  3007.08
..                 ...      ...   ...      ...      ...
```

```
880             230433  26.0000   NaN        S  2600.00
882               7552  10.5167   NaN        S  1051.67
885             382652  29.1250   NaN        Q  2912.50
887             112053  30.0000   B42        S  3000.00
888         W./C. 6607  23.4500   NaN        S  2345.00

[314 rows x 12 columns]
```

```
df[(df['Fare'] > 20) & (df['Fare'] < 50)]
```

```
     PassengerId  Survived  Pclass  \
7              8         0       3
9             10         1       2
11            12         1       1
13            14         0       3
16            17         0       3
..           ...       ...     ...
880           881         1       2
885           886         0       3
887           888         1       1
888           889         0       3
889           890         1       1

                                                Name     Sex  SibSp
Parch  \
7                     Palsson, Master. Gosta Leonard    male      3
1
9                  Nasser, Mrs. Nicholas (Adele Achem)  female      1
0
11                         Bonnell, Miss. Elizabeth  female      0
0
13                       Andersson, Mr. Anders Johan    male      1
5
16                              Rice, Master. Eugene    male      4
1
..                                               ...     ...    ...    ..
.
880  Shelley, Mrs. William (Imanita Parrish Hall)  female      0
1
885            Rice, Mrs. William (Margaret Norton)  female      0
5
887                    Graham, Miss. Margaret Edith  female      0
0
888        Johnston, Miss. Catherine Helen "Carrie"  female      1
2
889                            Behr, Mr. Karl Howell    male      0
0

         Ticket     Fare Cabin Embarked   Amount
7        349909  21.0750   NaN        S  2107.50
```

```
9         237736  30.0708   NaN         C  3007.08
11        113783  26.5500  C103         S  2655.00
13        347082  31.2750   NaN         S  3127.50
16        382652  29.1250   NaN         Q  2912.50
..           ...      ...   ...       ...      ...
880       230433  26.0000   NaN         S  2600.00
885       382652  29.1250   NaN         Q  2912.50
887       112053  30.0000   B42         S  3000.00
888   W./C. 6607  23.4500   NaN         S  2345.00
889       111369  30.0000  C148         C  3000.00

[215 rows x 12 columns]
```

Step-14 Compute the sum of value

```
df.Fare.sum()

28693.9493
```

Step-15 Compute the mean of value

```
df.Fare.mean()

32.204207968574636
```

Step-16 Count non-null value (column)

```
df.count()

PassengerId     891
Survived        891
Pclass          891
Name            891
Sex             891
SibSp           891
Parch           891
Ticket          891
Fare            891
Cabin           204
Embarked        889
Amount          891
dtype: int64
```

Step-17 Find Minimun or Maximum values

```
df.Fare.max()
```

```
512.3292
```

```
df.Fare.min()
```

```
0.0
```

Numpy & Perform Data Exploration with Pandas

Numpy

1) NumPy (Numerical Python) is a powerful open-source library in Python used for numerical and scientific computing. 2) It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on them efficiently. 3) NumPy is highly optimized and written in C, making it much faster than using regular Python lists for numerical operations. 4) It serves as the foundation for many other Python libraries in data science and machine learning, like pandas, TensorFlow, and scikit-learn. 5) With features like broadcasting, vectorization, and integration with C/C++ code, NumPy allows for cleaner and faster code in numerical computations.

Step 1. Import the Numpy library

```
import numpy as np
```

Step 2. Create a 1D array of numbers

```
#arr = np.arange(20)
#arr = np.arange(5,20)
arr = np.array([1,2,3,4,5])
arr
```

```
array([1, 2, 3, 4, 5])
```

Step 3. Reshape 1D to 2D Array

```
arr1=np.arange(12).reshape(3,4)
#arange(total).reshape(argument1,argument =total)
arr1
#print(type(arr1))  #array type
#arr1.dtype
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

Step 4. Create a Linspace array

```
#linspace uses the number of samples (num) as a parameter, whereas
arange uses the step size.
#linspace includes the endpoint by default, while arange does not.
np.linspace(2,5) #numpy.linspace(start, stop, num=50)
```

```
array([2.        , 2.06122449, 2.12244898, 2.18367347, 2.24489796,
       2.30612245, 2.36734694, 2.42857143, 2.48979592, 2.55102041,
       2.6122449 , 2.67346939, 2.73469388, 2.79591837, 2.85714286,
       2.91836735, 2.97959184, 3.04081633, 3.10204082, 3.16326531,
       3.2244898 , 3.28571429, 3.34693878, 3.40816327, 3.46938776,
       3.53061224, 3.59183673, 3.65306122, 3.71428571, 3.7755102 ,
       3.83673469, 3.89795918, 3.95918367, 4.02040816, 4.08163265,
       4.14285714, 4.20408163, 4.26530612, 4.32653061, 4.3877551 ,
       4.44897959, 4.51020408, 4.57142857, 4.63265306, 4.69387755,
       4.75510204, 4.81632653, 4.87755102, 4.93877551, 5.        ])
```

Step 5. Create a Random Numbered Array

```
np.random.rand(4,5) #.rand(row,column)
```

```
array([[0.44038032, 0.67471774, 0.44878604, 0.12097114, 0.16000432],
       [0.90871693, 0.89649164, 0.25501194, 0.0460539 , 0.93094023],
       [0.02461087, 0.85730924, 0.95664616, 0.04115632, 0.84472724],
       [0.81159055, 0.86620271, 0.68160496, 0.03558224, 0.47387525]])
```

Step 6. Create a Random Integer Array

```
np.random.randint(1,100,size=10)
```

```
array([ 9,  8,  8, 94, 88, 74,  2, 20, 56, 95])
```

```
#2d array
np.random.randint(1,100,size=(4,5))
```

```
array([[19, 96, 20, 86, 19],
       [ 9, 67, 22, 23, 64],
       [90, 85, 27, 14, 57],
       [16, 64, 30, 38, 68]])
```

Step 7. Create a 1D Array and get Max,Min,ArgMax,ArgMin

```
arr=np.random.randint(20,50,size=10)
arr
```

```
array([48, 37, 21, 33, 29, 43, 38, 22, 43, 34])
```

```
arr.max()
```

```
48
```

```
arr.min()
```

```
21
```

```
#The numpy.argmax() function returns indices of the max element of the array in a particular axis.
arr.argmax()
```

```
0
```

```
#The numpy.argmin() method returns indices of the min element of the array in a particular axis.
arr.argmin()
```

```
2
```

Step 8. Indexing in 1D Array

```
arr[5]
```

```
43
```

Step 9. Indexing in 2D Array

```
arr2= arr.reshape(2,5)
arr2
```

```
array([[48, 37, 21, 33, 29],
       [43, 38, 22, 43, 34]])
```

```
arr2[0]
```

```
array([48, 37, 21, 33, 29])
```

```
arr2[1][2]
```

```
22
```

Step 10. Conditional Selection

```
arr3=np.random.randint(1,20,5)
arr3
```

```
array([13,  5, 19,  3,  3])
```

```
arr3[arr3 > 5]
```

```
array([13, 19])
```

🔥 You did it! 10 exercises down — you're on fire! 🔥

Pandas

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

```
users=pd.read_csv("https://raw.githubusercontent.com/justmarkham/
DAT8/master/data/u.user",sep="|",index_col="user_id")
users
```

```
          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
...       ...    ...             ...      ...
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

[943 rows x 4 columns]
```

Step 4. See the first 25 entries

```
users.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

```
#default value 5
users.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?

```
users.shape[0]
#users.shape #-----> ama bai ave row nd col
```

```
(943, 4)
```

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

```
users.shape[1]
```

```
4
```

Step 8. Print the name of all the columns.

```
users.columns
```

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

Step 9. How is the dataset indexed?

```
users.index
```

```
Index([  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?

```
users.dtypes
```

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

Step 11. Print only the occupation column

```
#users.occupation
users['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?

```
users.occupation.nunique()
```

```
21
```

Step 13. What is the most frequent occupation?

```
#users.occupation.value_counts()
users.occupation.value_counts().head(1)
```

```
occupation
student    196
Name: count, dtype: int64
```

Step 14. Summarize the DataFrame.

```
users.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

```
users.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

```
users['occupation'].describe()
```

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

Step 17. What is the mean age of users?

```
users['age'].mean()
```

```
34.05196182396607
```

Step 18. What is the age with least occurrence?

```
users.age.value_counts().tail()
```

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

You're not just learning, you're mastering it. Keep aiming higher! 🔥

Data Mining Lab – 3 Rudra - 23010101048

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')
```

```
FileNotFoundError                         Traceback (most recent call
last)
Cell In[1], line 2
      1 import pandas as pd
----> 2 df = pd.read_csv('titanic.csv')

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1026,
in read_csv(filepath_or_buffer, sep, delimiter, header, names,
index_col, usecols, dtype, engine, converters, true_values,
false_values, skipinitialspace, skiprows, skipfooter, nrows,
na_values, keep_default_na, na_filter, verbose, skip_blank_lines,
parse_dates, infer_datetime_format, keep_date_col, date_parser,
date_format, dayfirst, cache_dates, iterator, chunksize, compression,
thousands, decimal, lineterminator, quotechar, quoting, doublequote,
escapechar, comment, encoding, encoding_errors, dialect, on_bad_lines,
delim_whitespace, low_memory, memory_map, float_precision,
storage_options, dtype_backend)
   1013 kwds_defaults = _refine_defaults_read(
   1014     dialect,
   1015     delimiter,
   (...)
   1022     dtype_backend=dtype_backend,
   1023 )
   1024 kwds.update(kwds_defaults)
-> 1026 return _read(filepath_or_buffer, kwds)

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:620,
in _read(filepath_or_buffer, kwds)
    617  validate_names(kwds.get("names", None))
    619 # Create the parser.
--> 620 parser = TextFileReader(filepath_or_buffer, **kwds)
    622 if chunksize or iterator:
    623     return parser

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1620,
in TextFileReader.__init__(self, f, engine, **kwds)
   1617     self.options["has_index_names"] = kwds["has_index_names"]
   1619 self.handles: IOHandles | None = None
-> 1620 self._engine = self._make_engine(f, self.engine)
```

```
File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1880,
in TextFileReader._make_engine(self, f, engine)
   1878     if "b" not in mode:
   1879         mode += "b"
-> 1880 self.handles = get_handle(
   1881     f,
   1882     mode,
   1883     encoding=self.options.get("encoding", None),
   1884     compression=self.options.get("compression", None),
   1885     memory_map=self.options.get("memory_map", False),
   1886     is_text=is_text,
   1887     errors=self.options.get("encoding_errors", "strict"),
   1888     storage_options=self.options.get("storage_options", None),
   1889 )
   1890 assert self.handles is not None
   1891 f = self.handles.handle

File ~\anaconda3\Lib\site-packages\pandas\io\common.py:873, in
get_handle(path_or_buf, mode, encoding, compression, memory_map,
is_text, errors, storage_options)
    868 elif isinstance(handle, str):
    869     # Check whether the filename is to be opened in binary
mode.
    870     # Binary mode does not support 'encoding' and 'newline'.
    871     if ioargs.encoding and "b" not in ioargs.mode:
    872         # Encoding
--> 873         handle = open(
    874             handle,
    875             ioargs.mode,
    876             encoding=ioargs.encoding,
    877             errors=errors,
    878             newline="",
    879         )
    880     else:
    881         # Binary mode
    882         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory: 'titanic.csv'

df

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
last)
Cell In[3], line 1
----> 1 df

NameError: name 'df' is not defined
```

```
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                            Bra  nd, 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.

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

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

```
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.

```
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.

```
        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.

```
              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
```

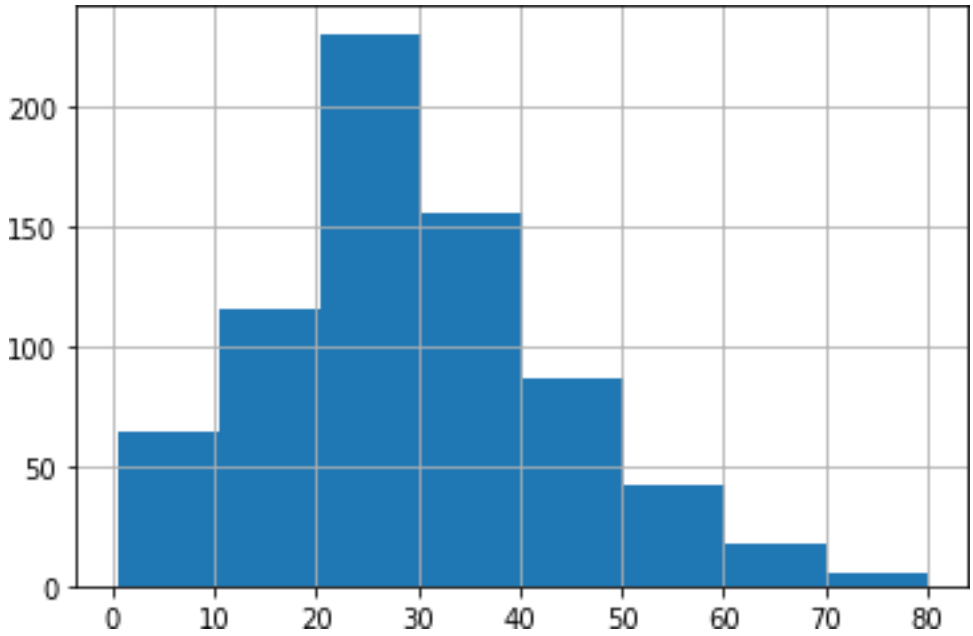
```
Age          -2.344191    73.849030
SibSp         0.368739     8.748734
Parch         0.649728     8.661052
Fare          8.661052  2469.436846
```

```
             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
```

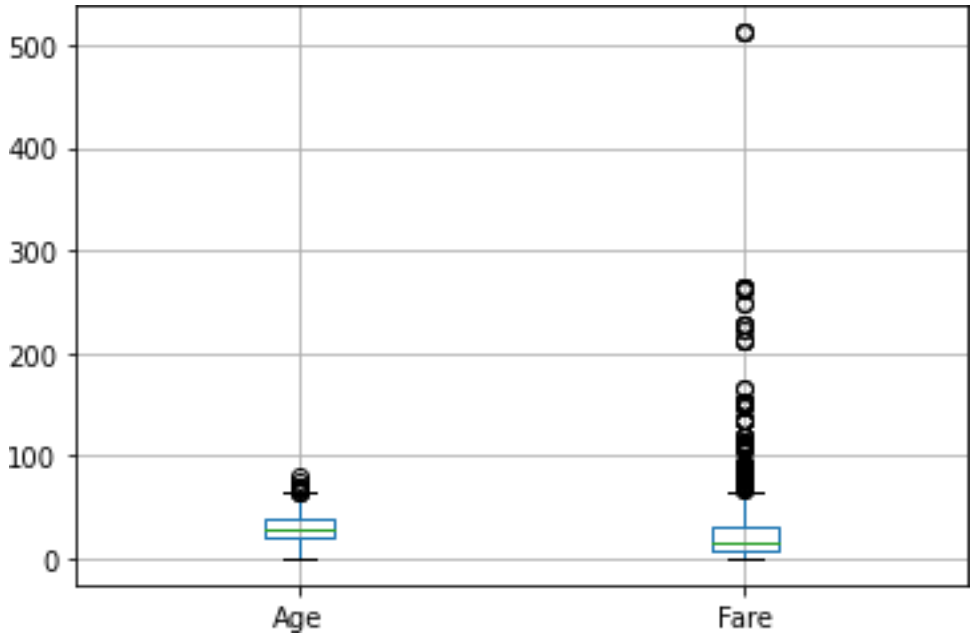
9) Display the histogram for Age attribute by discretizing it into 8 separate bins and counting the frequency for each bin.

```
<AxesSubplot:>
```



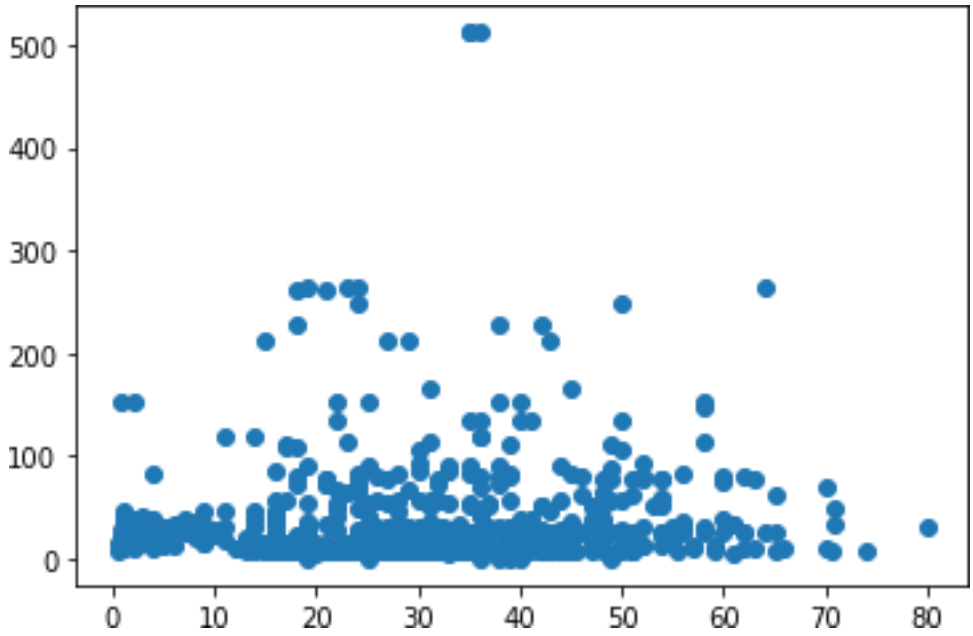
10) A boxplot can also be used to show the distribution of values for each attribute.

```
<AxesSubplot:>
```



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

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



Data Mining Lab - 4 Rudra - 23010101048

Step 1. Import the necessary libraries

```
import pandas as pd
import numpy as np
```

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called chipo.

```
chipo=pd.read_csv("https://raw.githubusercontent.com/justmarkham/
DAT8/master/data/chipotle.tsv",sep="\t")
chipo
```

```
      order_id  quantity                              item_name  \
0            1         1           Chips and Fresh Tomato Salsa
1            1         1                                   Izze
2            1         1                       Nantucket Nectar
3            1         1  Chips and Tomatillo-Green Chili Salsa
4            2         2                           Chicken Bowl
...        ...       ...                                    ...
4617      1833         1                      Steak Burrito
4618      1833         1                      Steak Burrito
4619      1834         1                 Chicken Salad Bowl
4620      1834         1                 Chicken Salad Bowl
4621      1834         1                 Chicken Salad Bowl

                                choice_description item_price
0                                              NaN     $2.39
1                                      [Clementine]     $3.39
2                                           [Apple]     $3.39
3                                              NaN     $2.39
4     [Tomatillo-Red Chili Salsa (Hot), [Black Beans...    $16.98
...                                              ...       ...
4617  [Fresh Tomato Salsa, [Rice, Black Beans, Sour ...    $11.75
4618  [Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...    $11.75
4619  [Fresh Tomato Salsa, [Fajita Vegetables, Pinto...    $11.25
4620  [Fresh Tomato Salsa, [Fajita Vegetables, Lettu...     $8.75
4621  [Fresh Tomato Salsa, [Fajita Vegetables, Pinto...     $8.75

[4622 rows x 5 columns]
```

Step 4. See the first 10 entries

```
chipo.head(10)
```

```
   order_id  quantity                              item_name  \
0         1         1           Chips and Fresh Tomato Salsa
1         1         1                                   Izze
```

```
2          1          1                            Nantucket Nectar
3          1          1  Chips and Tomatillo-Green Chili Salsa
4          2          2                               Chicken Bowl
5          3          1                               Chicken Bowl
6          3          1                              Side of Chips
7          4          1                              Steak Burrito
8          4          1                            Steak Soft Tacos
9          5          1                               Steak Burrito
```

```
                                     choice_description item_price
0                                                   NaN     $2.39
1                                           [Clementine]     $3.39
2                                                [Apple]     $3.39
3                                                   NaN     $2.39
4  [Tomatillo-Red Chili Salsa (Hot), [Black Beans...     $16.98
5  [Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...     $10.98
6                                                   NaN     $1.69
7  [Tomatillo Red Chili Salsa, [Fajita Vegetables...     $11.75
8  [Tomatillo Green Chili Salsa, [Pinto Beans, Ch...      $9.25
9  [Fresh Tomato Salsa, [Rice, Black Beans, Pinto...      $9.25
```

Step 5. What is the number of observations in the dataset?

```
# Solution 1
chipo.shape[0]
```

```
4622
```

```
# Solution 2
chipo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4622 entries, 0 to 4621
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   order_id            4622 non-null   int64 
 1   quantity            4622 non-null   int64 
 2   item_name           4622 non-null   object
 3   choice_description  3376 non-null   object
 4   item_price          4622 non-null   object
dtypes: int64(2), object(3)
memory usage: 180.7+ KB
```

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

```
# 1 for columns
# 0 for rows
chipo.shape[1]
```

```
5
```

Step 7. Print the name of all the columns.

```
chipo.columns
```

```
Index(['order_id', 'quantity', 'item_name', 'choice_description',
       'item_price'],
      dtype='object')
```

Step 8. How is the dataset indexed?

```
chipo.index
```

```
RangeIndex(start=0, stop=4622, step=1)
```

Step 9. Number of Unique Items ?

```
chipo.item_name.nunique()
```

```
50
```

Step 10. Which was the most-ordered item?

```
c=chipo.groupby('item_name')
#c.get_group('Chicken Bowl')
c=c.sum()
c=c.sort_values(['quantity'],ascending=False)
c.head(1)
```

```
               order_id quantity \
item_name
Chicken Bowl    713926        761

                                                choice_description \
item_name
Chicken Bowl  [Tomatillo-Red Chili Salsa (Hot), [Black Beans...

                                                        item_price
item_name
Chicken Bowl  $16.98 $10.98 $11.25 $8.75 $8.49 $11.25 $8.75 ...
```

Step 11. How many items were orderd in total?

```
chipo['quantity'].sum()
```

```
4972
```

Step 12. Turn the item price into a float

Step 12.a. Check the item price type

```
chipo['item_price'].dtypes
```

```
dtype('O')
```

Step 12.b. Create a lambda function and change the type of item price

```
a="$2.34"
print(type(a))
```

```
<class 'str'>
```

```
x=a[1:]
print(x)
```

```
2.34
```

```
y=float(x)
print(y)
print(type(y))
```

```
2.34
<class 'float'>
```

```
m = lambda x: float(x[1:])
chipo['item_price']=chipo['item_price'].apply(m)
```

Step 12.c. Check the item price type

```
chipo['item_price'].dtypes
```

```
dtype('float64')
```

Step 14. How much was the revenue for the period in the dataset?

```
revenue=(chipo['quantity']*chipo['item_price']).sum()
print("revenue was :$",revenue)
```

```
revenue was :$ 39237.02
```

Step 15. How many orders were made ?

```
chipo['order_id'].nunique()
```

```
1834
```

Step 17. How many different choice descriptions are there?

```
chipo['choice_description'].nunique()
```

```
1043
```

Step 18. What items have been ordered more than 100 times?

```
chipo.groupby('item_name')['quantity'].sum().loc[lambda x:x>100]
```

```
item_name
Bottled Water                    211
Canned Soda                      126
Canned Soft Drink                351
Chicken Bowl                     761
Chicken Burrito                  591
Chicken Salad Bowl               123
Chicken Soft Tacos               120
Chips                            230
Chips and Fresh Tomato Salsa     130
Chips and Guacamole              506
Side of Chips                    110
Steak Bowl                       221
Steak Burrito                    386
Name: quantity, dtype: int64
```

Step 19. What is the average revenue amount per order?

```
# Solution 1
chipo['revenue']=chipo['quantity']*chipo['item_price']
c=chipo.groupby('order_id')['revenue'].sum().mean()
c
```

```
21.39423118865867
```

```
# Solution 2


21.394231188658654
```

Data Mining Lab - 5 Rudra – 23010101048

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.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
```

```
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
```

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

```
#je je record ni andar ek pan record missing hse ee record delete karse
df.dropna()
```

```
     PassengerId  Survived  Pclass  \
1              2         1       1
3              4         1       1
6              7         0       1
10            11         1       3
11            12         1       1
..           ...       ...     ...
871           872         1       1
872           873         0       1
879           880         1       1
887           888         1       1
889           890         1       1

                                                  Name     Sex   Age
SibSp  \
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
1
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
1
6                              McCarthy, Mr. Timothy J    male  54.0
0
10                     Sandstrom, Miss. Marguerite Rut  female   4.0
1
11                            Bonnell, Miss. Elizabeth  female  58.0
0
..                                                 ...     ...   ...
...
871   Beckwith, Mrs. Richard Leonard (Sallie Monypeny)  female  47.0
1
872                           Carlsson, Mr. Frans Olof    male  33.0
0
879      Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)  female  56.0
0
887                        Graham, Miss. Margaret Edith  female  19.0
0
889                               Behr, Mr. Karl Howell    male  26.0
0

     Parch    Ticket     Fare        Cabin  Embarked
1        0  PC 17599  71.2833          C85        C
3        0    113803  53.1000         C123        S
6        0     17463  51.8625          E46        S
10       1   PP 9549  16.7000           G6        S
```

```
11        0    113783  26.5500          C103        S
..      ...       ...      ...           ...      ...
871       1     11751  52.5542           D35        S
872       0       695   5.0000  B51 B53 B55        S
879       1     11767  83.1583           C50        C
887       0    112053  30.0000           B42        S
889       0    111369  30.0000          C148        C

[183 rows x 12 columns]
```

```
df.dropna(inplace=True)

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
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
..     ...               ...      ...   ...      ...
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

```
df.dropna(axis=1)
```

```
     PassengerId  Survived  Pclass  \
1              2         1       1
3              4         1       1
6              7         0       1
10            11         1       3
11            12         1       1
..           ...       ...     ...
871          872         1       1
872          873         0       1
879          880         1       1
887          888         1       1
889          890         1       1

                                                  Name     Sex   Age
SibSp  \
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
1
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
1
6                             McCarthy, Mr. Timothy J    male  54.0
0
10                    Sandstrom, Miss. Marguerite Rut  female   4.0
1
11                           Bonnell, Miss. Elizabeth  female  58.0
0
..                                                 ...     ...   ...
...
871   Beckwith, Mrs. Richard Leonard (Sallie Monypeny)  female  47.0
1
872                           Carlsson, Mr. Frans Olof    male  33.0
0
```

```
879      Potter, Mrs.  Thomas Jr (Lily Alexenia Wilson)  female  56.0
0
887                         Graham, Miss. Margaret Edith  female  19.0
0
889                              Behr, Mr. Karl Howell    male  26.0
0

     Parch    Ticket     Fare        Cabin Embarked
1        0  PC 17599  71.2833          C85        C
3        0    113803  53.1000         C123        S
6        0     17463  51.8625          E46        S
10       1   PP 9549  16.7000           G6        S
11       0    113783  26.5500         C103        S
..     ...       ...      ...          ...      ...
871      1     11751  52.5542          D35        S
872      0       695   5.0000  B51 B53 B55        S
879      1     11767  83.1583          C50        C
887      0    112053  30.0000          B42        S
889      0    111369  30.0000         C148        C

[183 rows x 12 columns]
```

```
df.dropna(how= 'any') #[any|all]
```

```
     PassengerId  Survived  Pclass  \
1              2         1       1
3              4         1       1
6              7         0       1
10            11         1       3
11            12         1       1
..           ...       ...     ...
871           872         1       1
872           873         0       1
879           880         1       1
887           888         1       1
889           890         1       1

                                                  Name      Sex    Age
SibSp  \
1    Cumings, Mrs. John Bradley (Florence Briggs Th...   female  38.0
1
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)   female  35.0
1
6                              McCarthy, Mr. Timothy J    male  54.0
0
10                    Sandstrom, Miss. Marguerite Rut   female   4.0
1
11                           Bonnell, Miss. Elizabeth   female  58.0
0
..                                                ...     ...   ...
```

```
...
871    Beckwith, Mrs. Richard Leonard (Sallie Monypeny)  female  47.0
1
872                            Carlsson, Mr. Frans Olof    male  33.0
0
879       Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)  female  56.0
0
887                         Graham, Miss. Margaret Edith  female  19.0
0
889                                Behr, Mr. Karl Howell    male  26.0
0

     Parch    Ticket     Fare        Cabin Embarked
1        0  PC 17599  71.2833          C85        C
3        0    113803  53.1000         C123        S
6        0     17463  51.8625          E46        S
10       1   PP 9549  16.7000           G6        S
11       0    113783  26.5500         C103        S
..     ...       ...      ...          ...      ...
871      1     11751  52.5542          D35        S
872      0       695   5.0000  B51 B53 B55        S
879      1     11767  83.1583          C50        C
887      0    112053  30.0000          B42        S
889      0    111369  30.0000         C148        C

[183 rows x 12 columns]

df.fillna("Darshan")

     PassengerId  Survived  Pclass  \
1              2         1       1
3              4         1       1
6              7         0       1
10            11         1       3
11            12         1       1
..           ...       ...     ...
871          872         1       1
872          873         0       1
879          880         1       1
887          888         1       1
889          890         1       1

                                                 Name     Sex    Age
SibSp  \
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
1
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
1
6                              McCarthy, Mr. Timothy J    male  54.0
0
```

```
10                      Sandstrom, Miss. Marguerite Rut  female    4.0
1
11                             Bonnell, Miss. Elizabeth  female   58.0
0
..                                                   ...     ...    ...
...
871   Beckwith, Mrs. Richard Leonard (Sallie Monypeny)  female   47.0
1
872                            Carlsson, Mr. Frans Olof    male   33.0
0
879       Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)  female   56.0
0
887                         Graham, Miss. Margaret Edith  female   19.0
0
889                              Behr, Mr. Karl How◂ ll    male   26.0
0

     Parch    Ticket      Fare        Cabin Embarked
1        0  PC 17599   71.2833          C85        C
3        0    113803   53.1000         C123        S
6        0     17463   51.8625          E46        S
10       1   PP 9549   16.7000           G6        S
11       0    113783   26.5500         C103        S
..     ...       ...       ...          ...      ...
871      1     11751   52.5542          D35        S
872      0       695    5.0000  B51 B53 B55        S
879      1     11767   83.1583          C50        C
887      0    112053   30.0000          B42        S
889      0    111369   30.0000         C148        C

[183 rows x 12 columns]
```

```
#df.fillna({"Age":0,"Cabin":"NA"})
#df.fillna({"Age":(df['Age'].mean()),"Cabin":"NA"}) #float ma value
avse
#df.fillna({"Age":int(df['Age'].mean()),"Cabin":"NA"})
df.fillna({"Age":df.groupby("Sex")['Age'].transform('mean')})
```

```
     PassengerId  Survived  Pclass  \
1              2         1       1
3              4         1       1
6              7         0       1
10            11         1       3
11            12         1       1
..           ...       ...     ...
871           872         1       1
872           873         0       1
879           880         1       1
887           888         1       1
889           890         1       1
```

```
                                                    Name      Sex    Age
SibSp  \
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
1
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
1
6                              McCarthy, Mr. Timothy J    male  54.0
0
10                    Sandstrom, Miss. Marguerite Rut  female   4.0
1
11                           Bonnell, Miss. Elizabeth  female  58.0
0
..                                                 ...     ...   ...
...
871   Beckwith, Mrs. Richard Leonard (Sallie Monypeny)  female  47.0
1
872                           Carlsson, Mr. Frans Olof    male  33.0
0
879      Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)  female  56.0
0
887                        Graham, Miss. Margaret Edith  female  19.0
0
889                             Behr, Mr. Karl How ll    male  26.0
0

     Parch    Ticket     Fare        Cabin Embarked
1        0  PC 17599  71.2833          C85        C
3        0    113803  53.1000         C123        S
6        0     17463  51.8625          E46        S
10       1   PP 9549  16.7000           G6        S
11       0    113783  26.5500         C103        S
..     ...       ...      ...          ...      ...
871      1     11751  52.5542          D35        S
872      0       695   5.0000  B51 B53 B55        S
879      1     11767  83.1583          C50        C
887      0    112053  30.0000          B42        S
889      0    111369  30.0000         C148        C

[183 rows x 12 columns]
```

```
df.groupby("Survived")['Age'].mean()
```

```
Survived
0    41.350000
1    32.905854
Name: Age, dtype: float64
```

```
#group by data free
gdf=
```

```
#df.groupby("Sex")['Age'].transform('mean')
df.groupby("Sex")['Age'].transform('median')
```

```
1       32.25
3       32.25
6       37.00
10      32.25
11      32.25
        ...
871     32.25
872     37.00
879     32.25
887     32.25
889     37.00
Name: Age, Length: 183, dtype: float64
```

```
#df.interpolate().isnull().sum()
df.interpolate()['Age'].std()
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_6900\1450143585.py:2:
FutureWarning: DataFrame.interpolate with object dtype is deprecated
and will raise in a future version. Call obj.infer_objects(copy=False)
before interpolating instead.
  df.interpolate()['Age'].std()
```

```
15.643865966849717
```

```
#method='linear' (default)
#method='time' (used for time series)
#method='polynomial', order=2
#limit_direction='forward' or 'backward' (control fill direction)
```

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

```
df['NewAge']=df["Age"]+5

df
```

```
     PassengerId  Survived  Pclass  \
1              2         1       1
3              4         1       1
6              7         0       1
10            11         1       3
11            12         1       1
..           ...       ...     ...
871          872         1       1
872          873         0       1
879          880         1       1
```

```
887           888         1       1
889           890         1       1

                                                    Name      Sex    Age
SibSp  \
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
1
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
1
6                              McCarthy, Mr. Timothy J    male  54.0
0
10                    Sandstrom, Miss. Marguerite Rut  female   4.0
1
11                           Bonnell, Miss. Elizabeth  female  58.0
0
..                                                 ...     ...   ...
...
871   Beckwith, Mrs. Richard Leonard (Sallie Monypeny)  female  47.0
1
872                            Carlsson, Mr. Frans Olof    male  33.0
0
879      Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)  female  56.0
0
887                        Graham, Miss. Margaret Edith  female  19.0
0
889                             Behr, Mr. Karl Howell    male  26.0
0

     Parch    Ticket     Fare        Cabin Embarked  NewAge
1        0  PC 17599  71.2833          C85        C    43.0
3        0    113803  53.1000         C123        S    40.0
6        0     17463  51.8625          E46        S    59.0
10       1   PP 9549  16.7000           G6        S     9.0
11       0    113783  26.5500         C103        S    63.0
..     ...       ...      ...          ...      ...     ...
871      1     11751  52.5542          D35        S    52.0
872      0       695   5.0000  B51 B53 B55        S    38.0
879      1     11767  83.1583          C50        C    61.0
887      0    112053  30.0000          B42        S    24.0
889      0    111369  30.0000         C148        C    31.0

[183 rows x 13 columns]
```

```
# Min-Max Scaling
df['AGE_MinMax'] = (df['Age'] - df['Age'].min()) / (df['Age'].max() -
df['Age'].min())
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   AGE_MinMax
0        0         A/5 21171    7.2500   NaN        S     0.271174
1        0          PC 17599   71.2833   C85        C     0.472229
2        0  STON/O2. 3101282    7.9250   NaN        S     0.321438
3        0            113803   53.1000  C123        S     0.434531
4        0            373450    8.0500   NaN        S     0.434531
..     ...               ...       ...   ...      ...          ...
886      0            211536   13.0000   NaN        S     0.334004
887      0            112053   30.0000   B42        S     0.233476
888      2        W./C. 6607   23.4500   NaN        S          NaN
889      0            111369   30.0000  C148        C     0.321438
890      0            370376    7.7500   NaN        Q     0.396833

[891 rows x 13 columns]
```

```
# Decimal Scaling
data_ds=df.copy()
age_ds=data_ds['Age']
max_age=age_ds.max()
temp = len(str(int(max_age)))
print(max_age)
print(temp)
data_ds['DecimalScale']=age_ds/(10 ** temp)
data_ds[['Age', 'DecimalScale']]
```

```
80.0
2
```

```
      Age  DecimalScale
0    22.0          0.22
1    38.0          0.38
2    26.0          0.26
3    35.0          0.35
4    35.0          0.35
..    ...           ...
886  27.0          0.27
887  19.0          0.19
888   NaN           NaN
889  26.0          0.26
890  32.0          0.32

[891 rows x 2 columns]
```

```
#using Z-score
data_zs = df.copy()
age_zs = data_zs['Age']
mean_age = age_zs.mean()
std_age = age_zs.std()
data_zs['ZScoreAge'] = (age_zs - mean_age) / std_age
data_zs[['Age', 'ZScoreAge']]
```

```
      Age  ZScoreAge
0    22.0  -0.530005
1    38.0   0.571430
2    26.0  -0.254646
3    35.0   0.364911
4    35.0   0.364911
..    ...        ...
886  27.0  -0.185807
887  19.0  -0.736524
888   NaN        NaN
889  26.0  -0.254646
890  32.0   0.158392
```

```
[891 rows x 2 columns]
```

Data Mining Lab - 6 Rudra - 23010101048

Dimensionality Reduction using NumPy

□ What is Data Reduction?

Data reduction refers to the process of reducing the amount of data that needs to be processed and stored, while preserving the essential patterns in the data.

Why do we reduce data?

- To reduce computational cost.
- To remove noise and redundant features.
- To improve model performance and training time.
- To visualize high-dimensional data in 2D or 3D.

Common data reduction techniques include:

- Principal Component Analysis (PCA)
- Feature selection
- Sampling

□ What is Principal Component Analysis (PCA)?

PCA is a **dimensionality reduction technique** that transforms a dataset into a new coordinate system. It identifies the **directions (principal components)** where the variance of the data is maximized.

Key Concepts:

- **Principal Components**: New features (linear combinations of original features) capturing most variance.
- **Eigenvectors & Eigenvalues**: Used to compute these principal directions.
- **Covariance Matrix**: Measures how features vary with each other.

PCA helps in **visualizing high-dimensional data**, **noise reduction**, and **speeding up algorithms**.

□ NumPy Functions Summary for PCA

Function	Purpose
`np.mean(X, axis=0)`	Compute mean of each column (feature-wise mean).
`X - np.mean(X, axis=0)`	Centering the data (zero mean).
`np.cov(X, rowvar=False)`	Compute covariance matrix for features.
`np.linalg.eigh(cov_mat)`	Get eigenvalues and eigenvectors (for symmetric matrices).

Function	Purpose
`np.argsort(values)[::-1]`	Sort values in descending order.
`np.dot(X, eigenvectors)`	Project original data onto new axes.

Step 1: Load the Iris Dataset

```
from sklearn.datasets import load_iris
import numpy as np
import matplotlib.pyplot as plt

iris =load_iris()
X=iris.data
Y=iris.target

print("Original shape :",X.shape)
```

```
Original shape : (150, 4)
```

Step 2: Standardize the data (zero mean)

```
X_meaned = X- np.mean(X, axis=0)
print("Data after centering (first 5 rows): \n",X_meaned[:5])
```

```
Data after centering (first 5 rows):
 [[-0.74333333  0.44266667 -2.358      -0.99933333]
 [-0.94333333 -0.05733333 -2.358      -0.99933333]
 [-1.14333333  0.14266667 -2.458      -0.99933333]
 [-1.24333333  0.04266667 -2.258      -0.99933333]
 [-0.84333333  0.54266667 -2.358      -0.99933333]]
```

Step 3: Compute the Covariance Matrix

```
cov_mat = np.cov(X_meaned, rowvar=False)
print("Covariance Matrix shape:", cov_mat.shape)
print(cov_mat)
```

```
Covariance Matrix shape: (4, 4)
[[ 0.68569351 -0.042434    1.27431544  0.51627069]
 [-0.042434    0.18997942 -0.32965638 -0.12163937]
 [ 1.27431544 -0.32965638  3.11627785  1.2956094 ]
 [ 0.51627069 -0.12163937  1.2956094   0.58100626]]
```

Step 4: Compute eigenvalues and eigenvectors

```
eigen_values, eigen_vectors = np.linalg.eigh(cov_mat)  #linear algebra
= linag
print("Eigenvalues:\n", eigen_values)
print("\nEigenvectors (first 2):\n", eigen_vectors[:,:2])
```

```
Eigenvalues:
 [0.02383509 0.0782095  0.24267075 4.22824171]

Eigenvectors (first 2):
 [[ 0.31548719  0.58202985]
 [-0.3197231  -0.59791083]
 [-0.47983899 -0.07623608]
 [ 0.75365743 -0.54583143]]
```

Step 5: Compute eigenvalues and eigenvectors in descending order

```
sorted_index = np.argsort(eigen_values)[::-1]
sorted_eigenvalues = eigen_values[sorted_index]
sorted_eigenvectors = eigen_vectors[:,sorted_index]
print("sort index",sorted_index)
print("sort eigenvalues",sorted_eigenvalues)
print("sort eigenvectors",sorted_eigenvectors)
```

```
sort index [3 2 1 0]
sort eigenvalues [4.22824171 0.24267075 0.0782095  0.02383509]
sort eigenvectors [[-0.36138659  0.65658877  0.58202985  0.31548719]
 [ 0.08452251  0.73016143 -0.59791083 -0.3197231 ]
 [-0.85667061 -0.17337266 -0.07623608 -0.47983899]
 [-0.3582892  -0.07548102 -0.54583143  0.75365743]]
```

Step 6: Select the top k eigenvectors (top 2)

```
k = 2
eigenvector_subset = sorted_eigenvectors[:,0:k] #[row : coloumn]
print(eigenvector_subset.shape)
```

```
(4, 2)
```

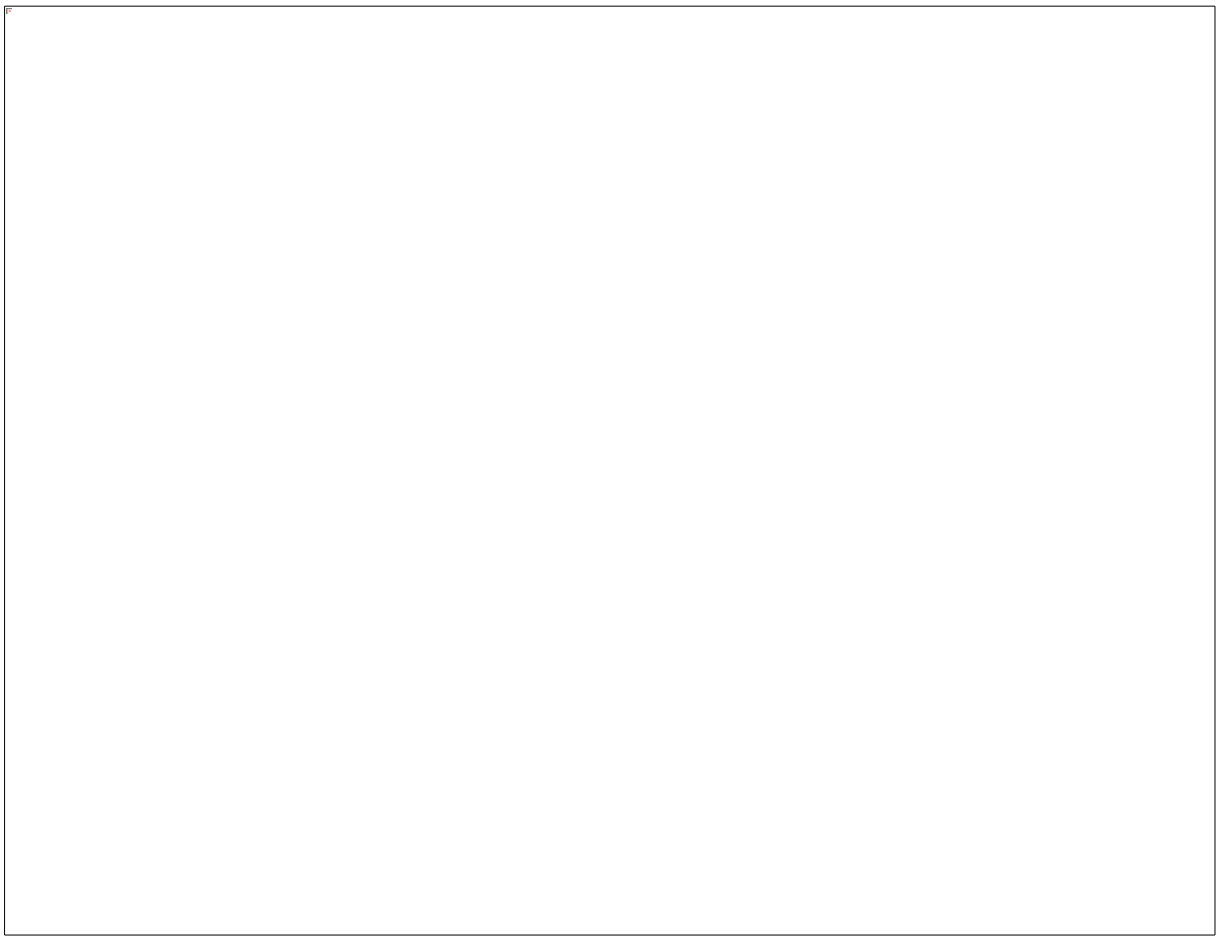
Step 7: Project the data onto the top k eigenvectors

```
X_reduced = np.dot(X_meaned, eigenvector_subset)
print("Reduced shape:", X_reduced.shape)

Reduced shape: (150, 2)
```

Step 8: Plot the PCA-Reduced Data

```
plt.figure(figsize=(8,6))
plt.scatter(X_reduced[:,0], X_reduced[:,1], c=Y, cmap='viridis')
plt.xlabel('principle component 1')
plt.ylabel('principle component 2')
plt.title('PCA - IRIS Dataset')
plt.grid(True)
plt.show()
```



Extra - Bining Method

5,10,11,13,15,35,50,55,72,92,204,215.

Partition them into three bins by each of the following methods:

(a) equal-frequency (equal-depth) partitioning

(b) equal-width partitioning

```
Sorted Data: [5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215]

(a) Equal-Frequency Bins:
Bin 1: [5, 10, 11, 13]
Bin 2: [15, 35, 50, 55]
Bin 3: [72, 92, 204, 215]

(b) Equal-Width Bins:
Bin 1: [5, 10, 11, 13, 15, 35, 50, 55, 72]
Bin 2: [92]
Bin 3: [204, 215]
```

Data Mining Lab – 7 (Part-2) Rudra - 23010101048

Step 1: Load the Dataset

Load the `Tdata.csv` file and display the first few rows.

```
import pandas as pd

data=pd.read_csv("Tdata.csv")

data.head(5)
```

	Transaction	bread	butter	coffee	eggs	jam	milk
0	T1	1	1	0	0	0	1
1	T2	1	1	0	0	1	0
2	T3	1	0	0	1	0	1
3	T4	1	1	0	0	0	1
4	T5	1	0	1	0	0	0

Step 2: Drop the 'Transaction' Column

We're only interested in the items (not the transaction IDs).

```
data.drop('Transaction',axis=1,inplace=True)

data
```

	bread	butter	coffee	eggs	jam	milk
0	1	1	0	0	0	1
1	1	1	0	0	1	0
2	1	0	0	1	0	1
3	1	1	0	0	0	1
4	1	0	1	0	0	0
5	0	0	1	1	1	0

Step 3: Count Single Items

See how many transactions include each item.

```
data.sum()
```

```
bread      5
butter     3
coffee     2
eggs       2
jam        2
milk       3
dtype: int64
```

Step 4: Define Apriori Function

This function finds frequent itemsets of size 1, 2, and 3 with minimum support.

```
from itertools import combinations
def find_frequent_itemsets(df,min_support):
    n=len(df)
    result=[]

    for k in [1,2,3]:
        for items in combinations(df.columns,k):
            mask=df[list(items)].all(axis=1)
            # print("items",items,"mask",mask.sum())
            support=mask.sum()/n
            if support>=min_support:
                result.append((frozenset(items),round(support,2)))
    return result
```

Step 5: Run Apriori

Set `min_support = 0.6` and display the frequent itemsets.

```
frequent_itemsets=find_frequent_itemsets(data,min_support=0.5)

for itemset, support in frequent_itemsets:
    print(f"{set(itemset)}-> support:{support}")
```

```
{'bread'}-> support:0.83
{'butter'}-> support:0.5
{'milk'}-> support:0.5
{'butter', 'bread'}-> support:0.5
{'bread', 'milk'}-> support:0.5
```

Step 6 Display as a DataFrame

```
result_data=pd.DataFrame(frequent_itemsets,columns=["Itemset","Support
"])
result_data
```

```
            Itemset  Support
0           (bread)      0.83
1          (butter)      0.50
2            (milk)      0.50
3   (butter, bread)      0.50
4     (bread, milk)      0.50
```

Orange Tool : - >Generate Same Frequent Patterns in Orange tools

Extra : - > Define Apriori Function without itertools

LAB - 7.

Apriori algo.

Min. sup. = 2

①

TID	Items.
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

→ C_1

ItemSet	Min. Sup.	
{1}	2	
{2}	3	
{3}	3	
{4}	1	*
{5}	3	

→ L_1

ItemSet	Min. Sup.
{1}	2
{2}	3
{3}	3
{5}	3

→ C_2

ItemSet	Min. Sup.	
{1 2}	1	*
{1 3}	2	
{1 5}	1	*
{2 3}	2	
{2 5}	3	
{3 5}	2	

→ L_2

ItemSet	Min. Sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

ItemSet	Min. Sup	
{1 2 3}	1	*
{1 3 5}	1	*
{2 3 5}	2	

Rules Generation

Association Rule	Support	Confidence	Confidence (%)
2 ^ 3 → 5	2	2/2 = 1	100 %
3 ^ 5 → 2	2	2/2 = 1	100 %
2 ^ 5 → 3	2	2/3 = 0.66	66 %
2 → 3 ^ 5	2	2/3 = 0.66	66 %
3 → 2 ^ 5	2	2/3 = 0.66	66 %
5 → 2 ^ 3	2	2/3 = 0.66	66 %

$$\frac{A \cup B}{A} = \frac{2\hat{}3\hat{}5}{5} = \frac{2}{3} = 0.66$$

⇒ (1) $2\hat{}3 \longrightarrow 5 = \frac{A \longrightarrow B}{A} = \frac{2\hat{}3\hat{}5}{2\hat{}3} = \frac{2}{2} = 1.$

- Similarly with others.

(2)

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Cola
4	Milk, Diaper, Beer, Cola
5	Bread, Milk, Diaper, Cola

C_1

IS	MS
{Br}	3
{M}	4
{Be}	3
{E}	1 ×
{D}	4
{C}	3

4 →

IS	MS
{Br}	3
{M}	4
{Be}	3
{D}	4
{C}	3

C_2

IS	MS
{Br, M}	2
{Br Be}	1
{Br D}	2
{Br C}	1
{M Be}	2
{M D}	3
{M C}	3
{Be D}	3
{Be C}	2
{D C}	3.

L_2

{Br, M}	2
{Br, D}	2
{M, Be}	2
{M D}	3
{M C}	3
{Be D}	3
{Be C}	2
{D C}	3

C_3

IS	MS
{Br M D}	1
{Br M Be}	0
{Br M C}	1
{Br Be D}	1
{Br D C}	1
{M Be D}	2
{M Be C}	2
{M D C}	3
{Be D C}	2

$L_3 \rightarrow$

IS	MS
{M Be D}	2
{M Be C}	2
{M D C}	3
{Be D C}	2

C_4

IS	MS
{M Be D C}	2

* Rules Generation

Association Rule	Support	Confidence	%
C ^ BE ^ D → M		2/2 = 1	100 %
M ^ BE ^ D → C		2/2 = 1	100 %
M ^ C ^ D → Be		2/3 = 0.66	66 %
M ^ Be ^ C → D		2/2 = 1	100 %
M ^ Be → D ^ C		2/2 = 1	100 %
M ^ D → Be ^ C		2/3 = 0.66	66 %
M ^ C → D ^ Be		2/3 = 0.66	66 %
M → C ^ Be ^ D	2/4 = 0.5	~~2/3 = 0.66~~	50 %
C → M ^ Be ^ D	2/3 = 0.66	~~2/4 = 0.5~~	66 %
Be → M ^ C ^ D		2/3 = 0.66	66 %
D → M ^ Be ^ C	2/4 = 0.5	~~2/3 = 0.66~~	50 %
D ^ C → M ^ Be		2/3 = 0.66	66 %
Be ^ C → M ^ D		2/2 = 1	100 %
D ^ Be → M ^ C		2/3 = 0.66	66 %

LAB - 8:

★ FP Tree

(1)

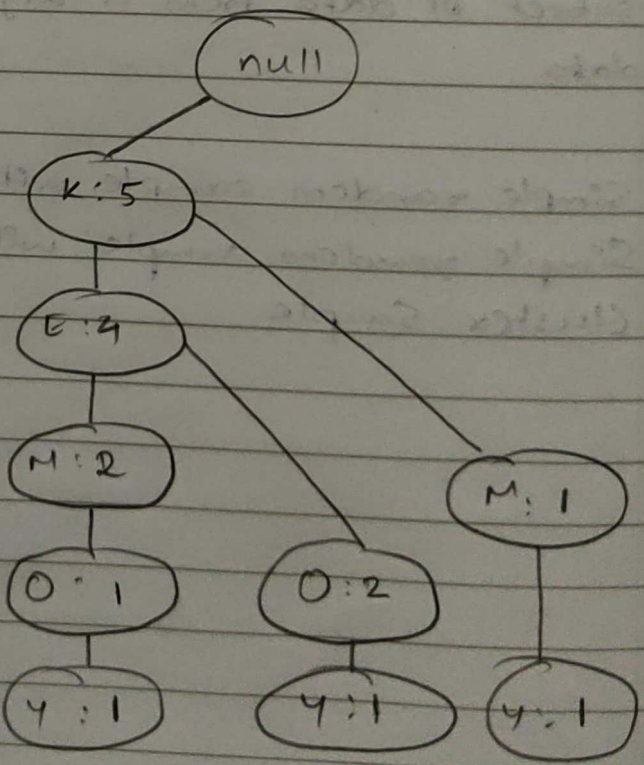
TID	Items
1	EKMNOY
2	DEKNOY
3	AEKM
4	CKMUY
5	CEIKO

Items	Freq.
A	1
C	2
D	1
E	4
K	5
M	3
N	2
O	3
Y	3
U	1
I	1

Frequent Patterns -

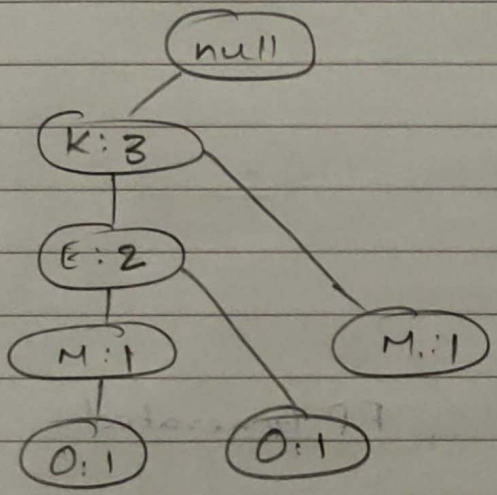
K : 5 , E : 4 , M : 3 , O : 3 , Y : 3

1	KEMOY
2	KEOY
3	KEM
4	KMY
5	KEO



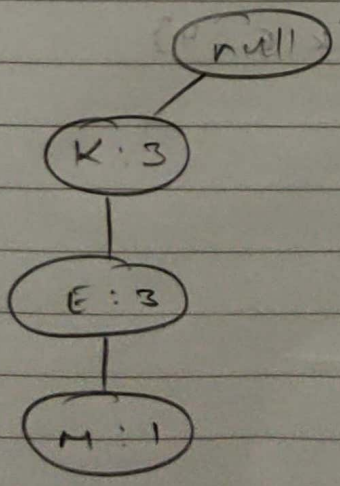
Item	Patterns
Y	{KEMO : 1}, {KEO : 1}, {KM : 1}
O	{KEM : 1}, {KE : 2}
M	{KE : 2}, {K : 1}
E	{K : 4}
K	—

1. Y {KEMO : 1}, {KEO : 1}, {KM : 1}



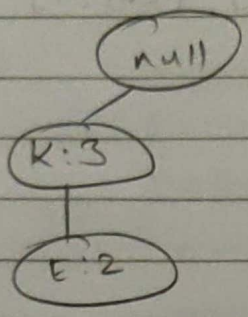
{K : 3}

2. O {KEM : 1} {KE : 2}



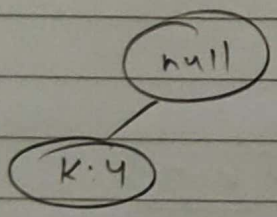
{K, E : 3}

3. M {KE:2} {K:1}



{K:3}

4. E {K:4}



{K:4}

- Items. FP obtained FP Generated

Y	{K:3}	{K,Y:3}
O	{K,E:3}	{K,O:3}, {E,O:3}, {K,E,O:3}
M	{K:3}	{K,M:3}
E	{K:4}	{K,E:4}

(2)

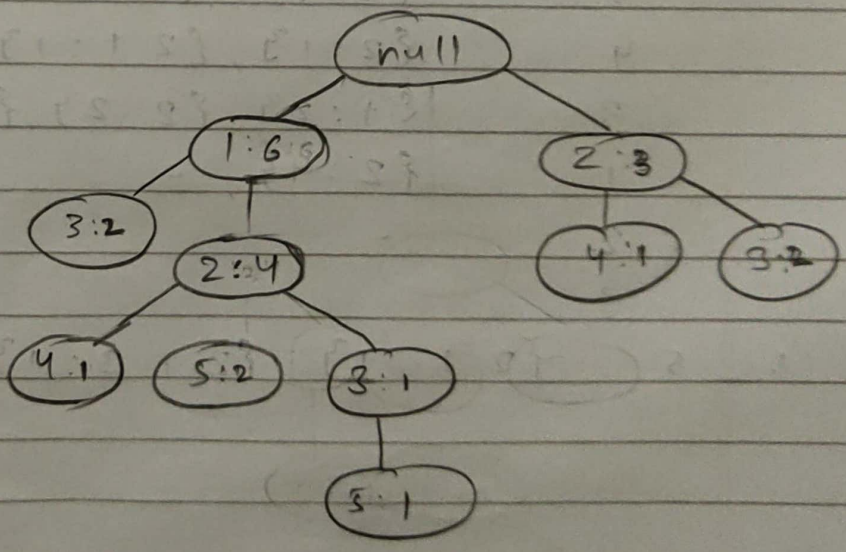
TID.	Items.
1	125
2	24
3	23
4	124
5	13
6	23
7	13
8	1235
9	123

Items	Freq.
1	6
2	7
3	6
4	2
5	2

Frequent Patterns-

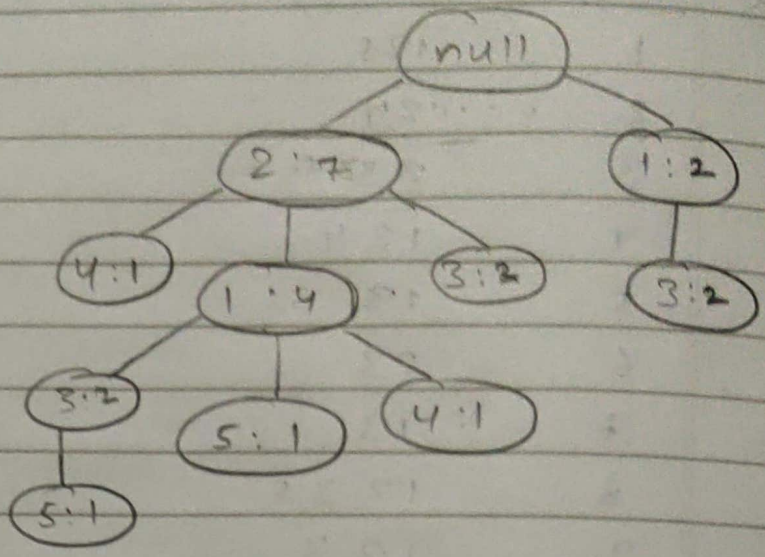
2:7, 1:6, 3:6, 4:2, 5:2

1	125
2	24
3	23
4	124
5	13
6	23
7	13
8	1235
9	123



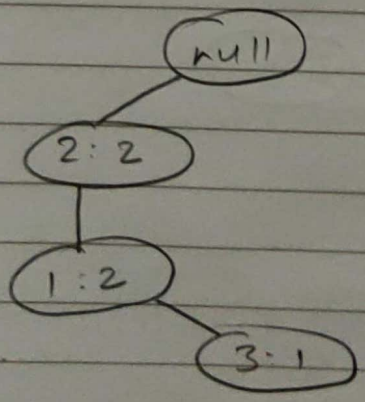
Frequent Patterns:

1	2 1 5
2	2 4
3	2 3
4	2 1 4
5	1 3
6	2 3
7	1 3
8	2 1 3 5
9	2 1 3



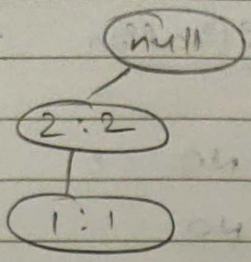
Items.	Patterns.
5	{2 1 : 1}, {2 1 3 : 1}
4	{2 : 1}, {2 1 : 1}
3	{1 : 2}, {2 : 2}, {2 1 : 2}
1	{2 : 4}, -
2	-

1. 5 {2 1 : 1}, {2 1 3 : 1}



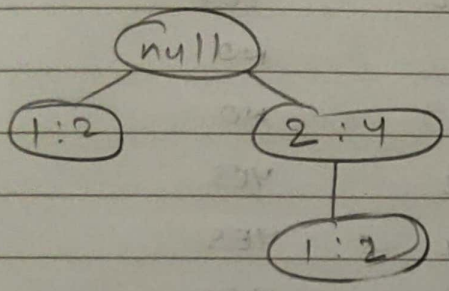
{2 : 2, 1 : 2}

2. 4 {2:1}, {2 1:1}



{2:2}

3. 3 {1:2}, {2:2}, {2 1:2}



{1:2, 2:4, 1:2}

4. {2:4}

Items.	FP obtained	FP generated
5	{2:2, 1:2}	{2,5:2}, {1,5:2}
4	{2:2}	{2,4:2}
3	{1:2, 2:4, 1:2}	{1,3:2}, {2,3:4}, {1,3:2}
1	{2:4}	{1,2:4}