योग: कर्मसु कौशलम्

Data Mining

Lab - 1

Nandani Padsumbiya | 02-06-2025

Introduction to Pandas Library Function:

Step-1 Import the pandas Libraries

In [3]:
```
import pandas as pd
```

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

In [ ]:
```

```

Step-3 Read csv or excel File

In [11]:
```
df=pd.read_csv('titanic.csv')
df
```

Out[11]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticl
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	175
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STC 31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	3734
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	3703

891 rows × 12 columns

Step-4 Print Data from csv or excel File

In [13]:

```
df
```

Out[13]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticl
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	175
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STO (31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	3734
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	3703

891 rows × 12 columns

Step-5 See the First 10 Rows

In [15]:

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

Out[15]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736

Step-6 See the Last 10 Rows

In [17]:

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

Out[17]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	T
881	882	0	3	Markun, Mr. Johann	male	33.0	0	0	34
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SC 3
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTO 39
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	38
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	21
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	11
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C.
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	11
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	37

Step-7 Data type of each columns

In [51]:

```
df.dtypes
```

Out[51]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
isCabin           bool
dtype: object
```

Step-8 Display Summary Information

In [25]:
```
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

In [53]:
```
df['Name']
```

Out[53]:
```
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
```

Step-10 Access rows by their integer location

In [29]:
```
df.iloc[2]
```

Out[29]:
```
PassengerId                          3
Survived                             1
Pclass                               3
Name            Heikkinen, Miss. Laina
Sex                             female
Age                               26.0
SibSp                                0
Parch                                0
Ticket                STON/O2. 3101282
Fare                             7.925
Cabin                              NaN
Embarked                             S
Name: 2, dtype: object
```

Step-11 Delete a specific Column

In [31]:
```
df.drop('Parch', axis=1)
# for row axis=0
# df.drop('Embarked', axis=1 , inplace=True)
```

Out[31]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	A/5 21171	7.
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	PC 17599	71.
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	STON/O2. 3101282	7.
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	113803	53.
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	373450	8.
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	211536	13.
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	112053	30.
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	W./C. 6607	23.
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	111369	30.
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	370376	7.

891 rows × 11 columns

Step-12 Create a new Column

In [39]:

```
df['isCabin']= ~df['Cabin'].isnull()
```

```
df
```

Out[39]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticl
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	175
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STC 31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	3734
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	3703

891 rows × 13 columns

Step-13 Perform Condition Selection on DataFrame

In [ ]:

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

Out[ ]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticl
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	175
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STC (31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	1138
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	3477
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	2377
...	...	...	...	...	...	...	...	...	
880	881	1	2	Shelley, Mrs. William (Imanita Parrish Hall)	female	25.0	0	1	2304
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	75
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	3826
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	1120
888	889	0	3	Johnston, Miss. Catherine	female	NaN	1	2	W 66

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticl
				Helen "Carrie"					

314 rows × 13 columns

In [89]:
```
df[(df['Pclass']==1) & (df['Age']>25)]
```

Out[89]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Tick
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	175
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	1138
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	174
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	1137
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	1137
...	...	...	...	...	...	...	...	...	
867	868	0	1	Roebling, Mr. Washington Augustus II	male	31.0	0	0	175
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	117
872	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	6
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	117
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	1113

144 rows × 13 columns

Step-14 Compute the sum of value

In [57]:
```
df.sum(numeric_only=True)
```

Out[57]:
```
PassengerId    397386.0000
Survived          342.0000
Pclass           2057.0000
Age             21205.1700
SibSp             466.0000
Parch             340.0000
Fare            28693.9493
isCabin           204.0000
dtype: float64
```

Step-15 Compute the mean of value

In [59]:
```
df.mean(numeric_only=True)
```

Out[59]:
```
PassengerId    446.000000
Survived         0.383838
Pclass           2.308642
Age             29.699118
SibSp            0.523008
Parch            0.381594
Fare            32.204208
isCabin          0.228956
dtype: float64
```

Step-16 Count non-null value (column)

In [61]:
```
df.count()
```

Out[61]:
```
PassengerId    891
Survived       891
Pclass         891
Name           891
Sex            891
Age            714
SibSp          891
Parch          891
Ticket         891
Fare           891
Cabin          204
Embarked       889
isCabin        891
dtype: int64
```

Step-17 Find Minimun or Maximum values

In [63]:
```
df.min(numeric_only=True)
```

Out[63]:
```
PassengerId       1
Survived          0
Pclass            1
Age            0.42
SibSp             0
Parch             0
Fare            0.0
isCabin       False
dtype: object
```

In [69]:
```
df.max(numeric_only=True)
```

Out[69]:
```
PassengerId         891
Survived              1
Pclass                3
Age                80.0
SibSp                 8
Parch                 6
Fare           512.3292
isCabin            True
dtype: object
```

In [77]:
```
# sort in descending order
df.sort_values('Age',ascending=False)
```

Out[77]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ti
630	631	1	1	Barkworth, Mr. Algernon Henry Wilson	male	80.0	0	0	2
851	852	0	3	Svensson, Mr. Johan	male	74.0	0	0	347
493	494	0	1	Artagaveytia, Mr. Ramon	male	71.0	0	0	1
96	97	0	1	Goldschmidt, Mr. George B	male	71.0	0	0	1
116	117	0	3	Connors, Mr. Patrick	male	70.5	0	0	370
...	...	...	...	...	...	...	...	...	
859	860	0	3	Razi, Mr. Raihed	male	NaN	0	0	
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8	2	
868	869	0	3	van Melkebeke, Mr. Philemon	male	NaN	0	0	34
878	879	0	3	Laleff, Mr. Kristo	male	NaN	0	0	34
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	

891 rows × 13 columns

Data Mining - Lab - 2

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

In [9]:
```
import numpy as np
```

Step 2. Create a 1D array of numbers

In [16]:
```
arr = np.array([1,2,3,4,5,6,7])
arr
```

Out[16]:
```
array([1, 2, 3, 4, 5, 6, 7])
```

In [20]:
```
type(arr)
```

Out[20]:
```
numpy.ndarray
```

In [22]:
```
arr1 = np.arange(10)
arr1
```

Out[22]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [24]:
```
#ndim uses for dimention
arr1.ndim
```

Out[24]: 1

In [26]:
```
#property size uses to get array size
arr1.size
```

Out[26]: 10

In [28]:
```
#uses to get datatype of individual element of array
arr1.dtype
```

Out[28]: dtype('int32')

In [11]:
```
arr2 = np.arange(101,201)
arr2
```

Out[11]:
```
array([101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113,
       114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126,
       127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139,
       140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152,
       153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165,
       166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178,
       179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191,
       192, 193, 194, 195, 196, 197, 198, 199, 200])
```

In [32]:
```
arr3 = np.array([[1,2],[1,3]])
arr3
```

Out[32]:
```
array([[1, 2],
       [1, 3]])
```

Step 3. Reshape 1D to 2D Array

In [42]:
```
# arr2.reshape(number of row , number of column)
arr4 = arr2.reshape(4,25)
arr4
```

Out[42]:
```
array([[101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113,
        114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125],
       [126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138,
        139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150],
       [151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163,
        164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175],
       [176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188,
        189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200]])
```

In [44]:
```
arr5 = np.arange(12).reshape(3,4)
arr5
```

Out[44]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [46]:
```
# aama error aavse kem ke arr pase data ma 10 element chhe ne row column thai
arr6 = np.arange(10).reshape(3,4)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[46], line 1
----> 1 arr6 = np.arange(10).reshape(3,4)

ValueError: cannot reshape array of size 10 into shape (3,4)
```

Step 4. Create a Linspace array

In [52]:
```
arr6 = np.linspace(1,10,3)
arr6
```

Out[52]:
```
array([ 1. ,  5.5, 10. ])
```

Step 5. Create a Random Numbered Array

In [54]:
```
n = np.random.rand(5)
n
```

Out[54]:
```
array([0.84100126, 0.26685111, 0.57066551, 0.74633546, 0.82257615])
```

In [ ]:

Step 6. Create a Random Integer Array

In [60]:
```
# arr = np.random.randint(start index , end index , number of element)
arr = np.random.randint(1,50,3)
arr
```

Out[60]:
```
array([43, 41, 43])
```

In [ ]:

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

In [15]:
```
arr2.min()
```

Out[15]:
```
101
```

In [17]:
```
arr2.max()
```

Out[17]: 200

In [19]:
```
#maximum array ni index return kare
arr2.argmax()
```

Out[19]: 99

In [23]:
```
#mininum array ni index return kare
arr2.argmin()
```

Out[23]: 0

In [ ]:

Step 8. Indexing in 1D Array

In [31]:
```
# np.random.rand(start point , end point , number of element)
arr= np.random.randint(1,20,11)
arr
```

Out[31]: array([4, 16, 18, 9, 17, 10, 19, 5, 11, 8, 2])

In [33]:
```
arr[6]
```

Out[33]: 19

In [37]:
```
#slicing
#arr[start:end:step]
#arr[:6:] or...
arr[:6]
```

Out[37]: array([4, 16, 18, 9, 17, 10])

In [39]:
```
arr[7::] or arr[7:]
```

Out[39]: array([5, 11, 8, 2])

In [41]:
```
#get alternet element
arr[::2]
```

Out[41]: array([4, 18, 17, 19, 11, 2])

Step 9. Indexing in 2D Array

In [43]:
```
arr = np.arange(15).reshape(3,5)
arr
```

Out[43]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

In [45]:
```
arr[1][2]
```

Out[45]:
```
7
```

In [47]:
```
arr[0][2]
```

Out[47]:
```
2
```

In [49]:
```
arr[1::]
```

Out[49]:
```
array([[ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

In [51]:
```
arr[:1:]
```

Out[51]:
```
array([[0, 1, 2, 3, 4]])
```

In [53]:
```
arr[:2:]
```

Out[53]:
```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

In [55]:
```
arr[::2]
```

Out[55]:
```
array([[ 0,  1,  2,  3,  4],
       [10, 11, 12, 13, 14]])
```

In [57]:
```
#arr[forrows,forcolumns]
arr[::2,::2]
```

Out[57]:
```
array([[ 0,  2,  4],
       [10, 12, 14]])
```

In [59]:
```
arr[::,::2]
```

Out[59]:
```
array([[ 0,  2,  4],
       [ 5,  7,  9],
       [10, 12, 14]])
```

In [61]:
```
arr[1::,::2]
```

Out[61]:
```
array([[ 5,  7,  9],
       [10, 12, 14]])
```

In [65]:
```
arr[:2:,1::3]
```

Out[65]:
```
array([[1, 4],
       [6, 9]])
```

Step 10. Conditional Selection

In [67]:
```
arr = np.random.randint(1,20,10)
arr
```

Out[67]:
```
array([ 8,  1, 18, 11,  5, 16, 18,  4, 18, 11])
```

In [69]:
```
arr[arr>10]
```

Out[69]:
```
array([18, 11, 16, 18, 18, 11])
```

In [71]:
```
arr1 = np.arange(15).reshape(3,5)
arr1
```

Out[71]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

In [73]:
```
arr1[arr1>10]
```

Out[73]:
```
array([11, 12, 13, 14])
```

In [75]:
```
arr1[(arr1>5)&(arr1<12)]
```

Out[75]:
```
array([ 6,  7,  8,  9, 10, 11])
```

In [77]:
```
arr1[(arr1==5)|(arr1==6)]
```

Out[77]:
```
array([5, 6])
```

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

Pandas

Step 1. Import the necessary libraries

In [97]:
```
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

In [103…
```
users = pd.read_csv("https://raw.githubusercontent.com/justmarkham/DAT8/master
users
```

Out[103…

	user_id|age|gender|occupation|zip_code
0	1|24|M|technician|85711
1	2|53|F|other|94043
2	3|23|M|writer|32067
3	4|24|M|technician|43537
4	5|33|F|other|15213
...	...
938	939|26|F|student|33319
939	940|32|M|administrator|02215
940	941|20|M|student|97229
941	942|48|F|librarian|78209
942	943|22|M|student|77841

943 rows × 1 columns

In [105…

```
users = pd.read_csv("https://raw.githubusercontent.com/justmarkham/DAT8/master
users
```

Out[105…

	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 × 4 columns

Step 4. See the first 25 entries

In [106…

```
users.head(25)
```

Out[106…

	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

In [109…
```
users.tail(10)
```

Out[109…

	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?

In [111…
```
# 0 for rows and 1 for columns
users.shape[0]
```

Out[111… 943

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

In [113…
```
users.shape[1]
```

Out[113… 4

Step 8. Print the name of all the columns.

In [115…
```
users.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 943 entries, 1 to 943
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   age         943 non-null    int64 
 1   gender      943 non-null    object
 2   occupation  943 non-null    object
 3   zip_code    943 non-null    object
dtypes: int64(1), object(3)
memory usage: 36.8+ KB
```

In [117…

```
users.columns
```

Out[117…

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

Step 9. How is the dataset indexed?

In [119…

```
users.index
```

Out[119…

```
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?

In [121…

```
users.dtypes
```

Out[121…

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

Step 11. Print only the occupation column

In [123…

```
users['occupation']
```

Out[123…
```
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
```

In [125…
```python
users.occupation
```

Out[125…
```
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?

In [129…
```python
users['occupation'].nunique()
```

Out[129…
```
21
```

In [133…
```python
users['occupation'].unique()
```

Out[133…
```
array(['technician', 'other', 'writer', 'executive', 'administrator',
       'student', 'lawyer', 'educator', 'scientist', 'entertainment',
       'programmer', 'librarian', 'homemaker', 'artist', 'engineer',
       'marketing', 'none', 'healthcare', 'retired', 'salesman', 'doctor'],
      dtype=object)
```

In [137…
```python
len(users['occupation'].unique())
```

Out[137…
```
21
```

Step 13. What is the most frequent occupation?

In [141…

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

Out[141…

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

Step 14. Summarize the DataFrame.

In [143…

```
users.describe()
```

Out[143…

	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

In [145…

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

Out[145…

	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

In [147…

```
users.occupation.describe()
```

Out[147…

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

In [149…

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

Out[149…

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

Step 17. What is the mean age of users?

In [151…

```
users['age'].mean(numeric_only=True)
```

Out[151…

```
34.05196182396607
```

Step 18. What is the age with least occurrence?

In [153…

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

Out[153…

```
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! ◈

In [ ]:

Data Mining

Lab - 3

Nandani padsumbiya | 2300101182 | 16/06/2025

In [7]:
```
import numpy as np
import pandas as pd
```

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

In [13]:
```
dt = pd.read_csv("titanic.csv")
dt
```

Out[13]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticl
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	175
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STC 31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	3734
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	3703

891 rows × 12 columns

In [15]:

```
dt.head(5)
```

Out[15]:

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

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

In [10]:

```
Nominal = ["Name","Ticket","cabin","Embarket"]
Ordinal = ["Pclass"]
Binary = ["Sex","Survived"]
Numeric = ["Passengerid","SibSp","Fare","Parch","Age"]
print("Nominal:",Nominal)
print("Ordinal:",Ordinal)
print("Binary:",Binary)
print("Numeric:",Numeric)
```

```
Nominal: ['Name', 'Ticket', 'cabin', 'Embarket']
Ordinal: ['Pclass']
Binary: ['Sex', 'Survived']
Numeric: ['Passengerid', 'SibSp', 'Fare', 'Parch', 'Age']
```

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

In [19]:

```
Symmetric = ["Sex"]
Asymmetric = ["Survived"]
print("symmentric:",Symmetric)
print("Asymmetric:",Asymmetric)
print("Sex (Binary Symmetric)")
print(dt['Sex'].value_counts())
```

```python
print(dt['Pclass'].value_counts())
```

```
symmentric: ['Sex']
Asymmetric: ['Survived']
Sex (Binary Symmetric)
Sex
male      577
female    314
Name: count, dtype: int64
Pclass
3    491
1    216
2    184
Name: count, dtype: int64
```

In [18]:
```python
print("Survived (Binary Asymmetric)")
print(dt['Survived'].value_counts())
```

```
Survived (Binary Asymmetric)
Survived
0    549
1    342
Name: count, dtype: int64
```

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

In [26]:
```python
qua = ["PassengerId","Survived","Pclass","Age","SibSp","Parch","Fare"]
for i in qua:
    print(i,"------------")
    print('\t mean / average :',dt[i].mean(numeric_only=True))
    print('\t standerd deviation :',dt[i].std(numeric_only=True))
    print('\t minimum :',dt[i].min(numeric_only=True))
    print('\t maximum :',dt[i].max(numeric_only=True))
    print('\t mode :',dt[i].mode())
    print('\t range :',(dt[i].max(numeric_only=True)-dt[i].min(numeric_only=Tr
```

```
PassengerId ------------
         mean / average : 446.0
         standerd deviation : 257.3538420152301
         minimum : 1
         maximum : 891
         mode : 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
         range : 890
Survived ------------
         mean / average : 0.3838383838383838
         standerd deviation : 0.4865924542648585
         minimum : 0
         maximum : 1
         mode : 0    0
Name: Survived, dtype: int64
         range : 1
Pclass ------------
         mean / average : 2.308641975308642
         standerd deviation : 0.8360712409770513
         minimum : 1
         maximum : 3
         mode : 0    3
Name: Pclass, dtype: int64
         range : 2
Age ------------
         mean / average : 29.69911764705882
         standerd deviation : 14.526497332334044
         minimum : 0.42
         maximum : 80.0
         mode : 0    24.0
Name: Age, dtype: float64
         range : 79.58
SibSp ------------
         mean / average : 0.5230078563411896
         standerd deviation : 1.1027434322934275
         minimum : 0
         maximum : 8
         mode : 0    0
Name: SibSp, dtype: int64
         range : 8
Parch ------------
         mean / average : 0.38159371492704824
         standerd deviation : 0.8060572211299559
         minimum : 0
```

```
         maximum : 6
         mode : 0    0
Name: Parch, dtype: int64
         range : 6
Fare ------------
         mean / average : 32.204207968574636
         standerd deviation : 49.693428597180905
         minimum : 0.0
         maximum : 512.3292
         mode : 0    8.05
Name: Fare, dtype: float64
         range : 512.3292
```

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

In [30]:
```python
dt['Pclass'].value_counts()
```

Out[30]:
```
Pclass
3    491
1    216
2    184
Name: count, dtype: int64
```

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

In [32]:
```python
dt.describe(include=['object'])
```

Out[32]:

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Braund, Mr. Owen Harris	male	347082	B96 B98	S
freq	1	577	7	4	644

In [36]:
```python
dt.describe(include='all')
```

Out[36]:

	PassengerId	Survived	Pclass	Name	Sex	Age	Sib
count	891.000000	891.000000	891.000000	891	891	714.000000	891.0000
unique	NaN	NaN	NaN	891	2	NaN	N
top	NaN	NaN	NaN	Braund, Mr. Owen Harris	male	NaN	N
freq	NaN	NaN	NaN	1	577	NaN	N
mean	446.000000	0.383838	2.308642	NaN	NaN	29.699118	0.5230
std	257.353842	0.486592	0.836071	NaN	NaN	14.526497	1.1027
min	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.0000
25%	223.500000	0.000000	2.000000	NaN	NaN	20.125000	0.0000
50%	446.000000	0.000000	3.000000	NaN	NaN	28.000000	0.0000
75%	668.500000	1.000000	3.000000	NaN	NaN	38.000000	1.0000
max	891.000000	1.000000	3.000000	NaN	NaN	80.000000	8.0000

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

In [38]:

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

Out[38]:

	PassengerId	Survived	Pclass	Age	SibSp	Par
PassengerId	66231.000000	-0.626966	-7.561798	138.696504	-16.325843	-0.3426
Survived	-0.626966	0.236772	-0.137703	-0.551296	-0.018954	0.0320
Pclass	-7.561798	-0.137703	0.699015	-4.496004	0.076599	0.0124
Age	138.696504	-0.551296	-4.496004	211.019125	-4.163334	-2.3441
SibSp	-16.325843	-0.018954	0.076599	-4.163334	1.216043	0.3687
Parch	-0.342697	0.032017	0.012429	-2.344191	0.368739	0.6497
Fare	161.883369	6.221787	-22.830196	73.849030	8.748734	8.6610

In [40]:

```
dt.corr(numeric_only=True)
```

Out[40]:

	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

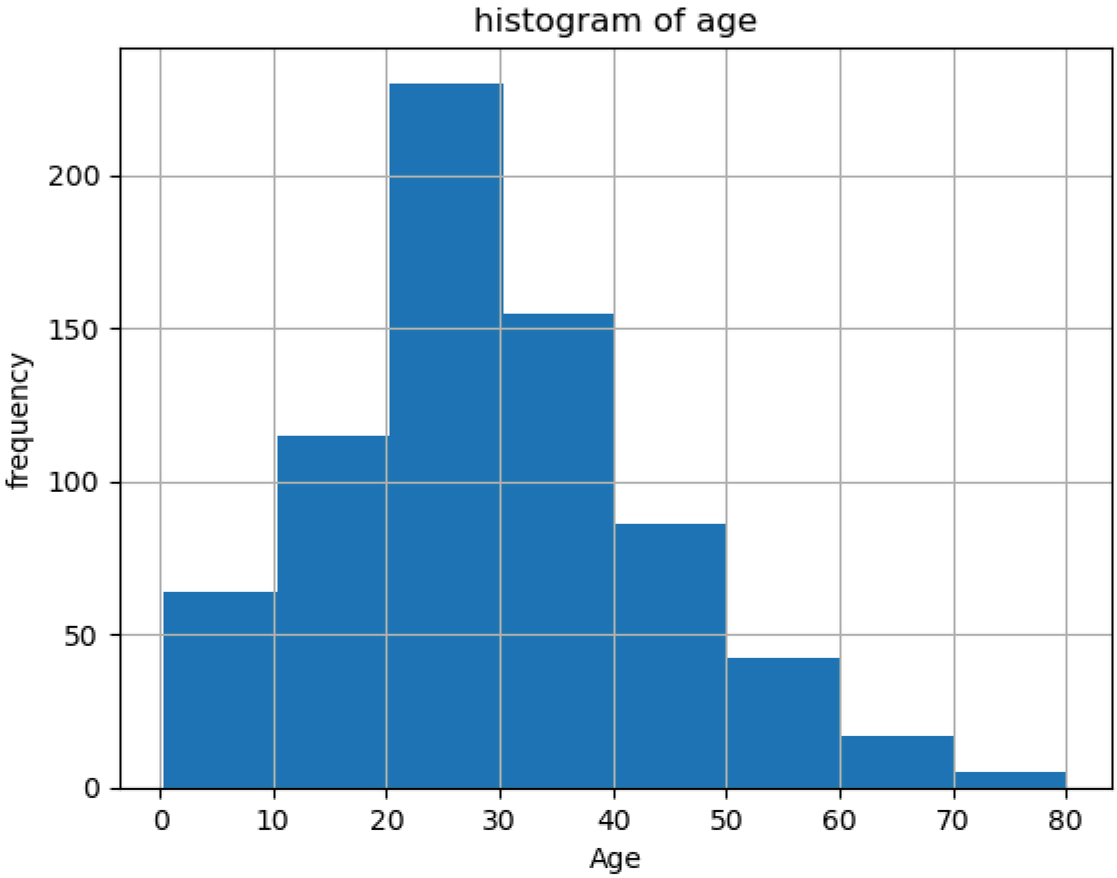
9) Display the histogram for Age attribute by discretizing it into 8 separate bins and counting the frequency for each bin.

In [44]:

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

In [48]:

```
dt['Age'].dropna().hist(bins=8)
plt.title("histogram of age")
plt.xlabel("Age")
plt.ylabel("frequency")
plt.show()
```

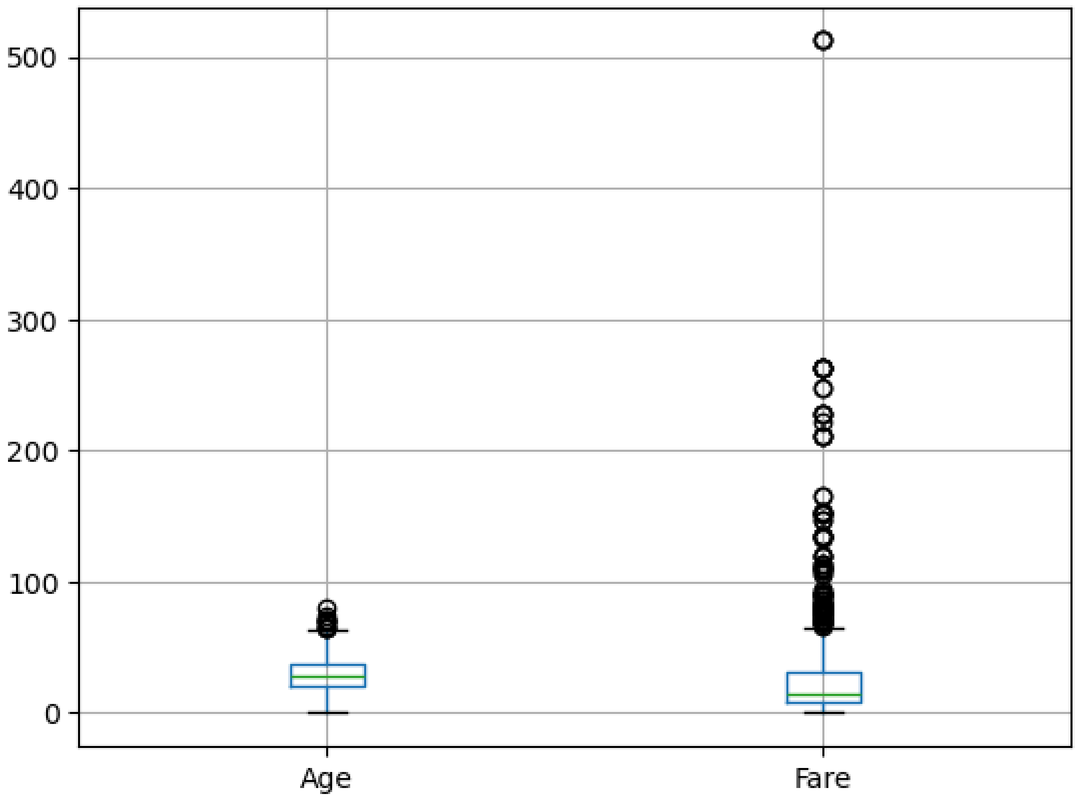


10) A boxplot can also be used to show the distribution of values for each attribute.

In [52]:
```
col = ['Age','Fare']
dt.boxplot(col)

#or dt.boxplot(['Age','Fare'])
```

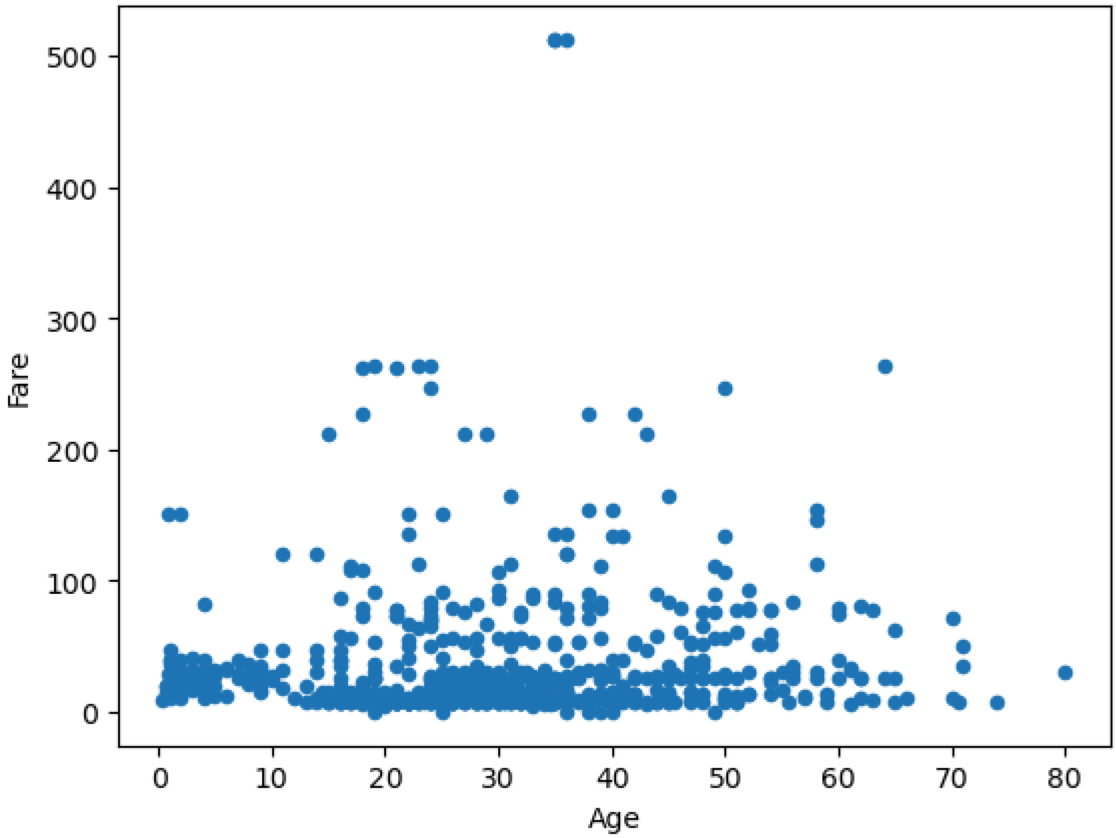
Out[52]: <Axes: >



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

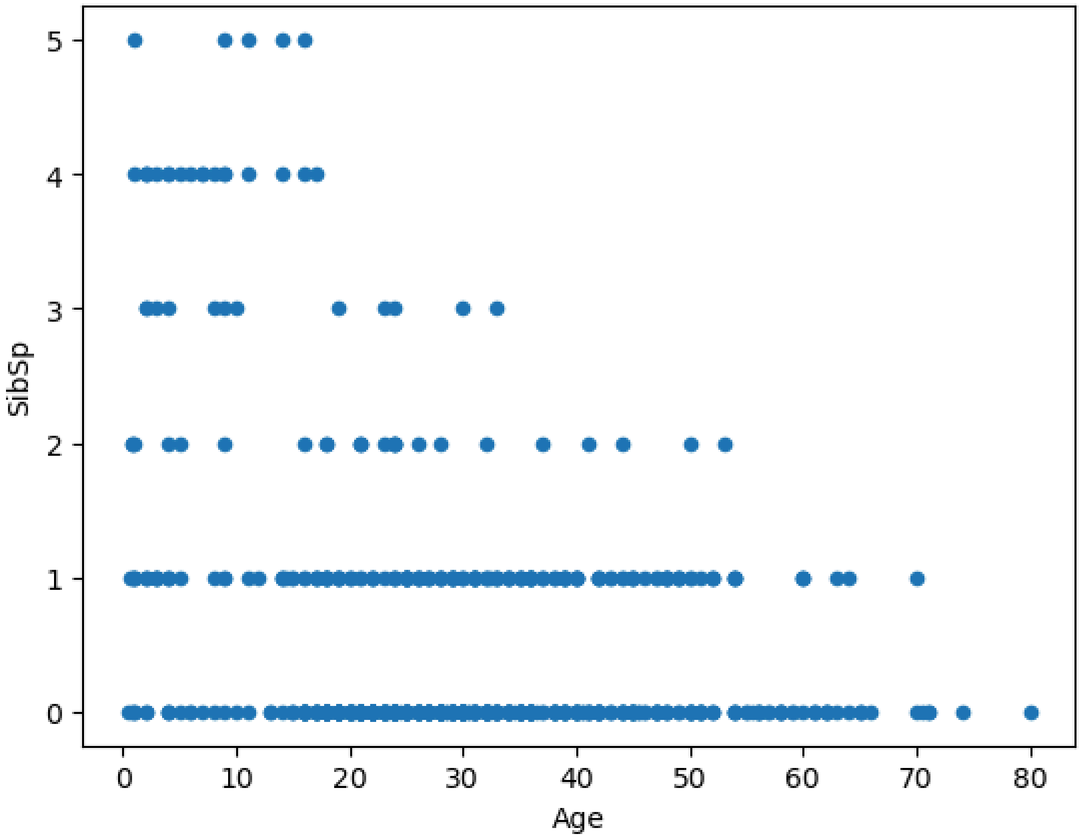
In [54]:
```
dt.plot.scatter('Age','Fare')
```

Out[54]:
```
<Axes: xlabel='Age', ylabel='Fare'>
```



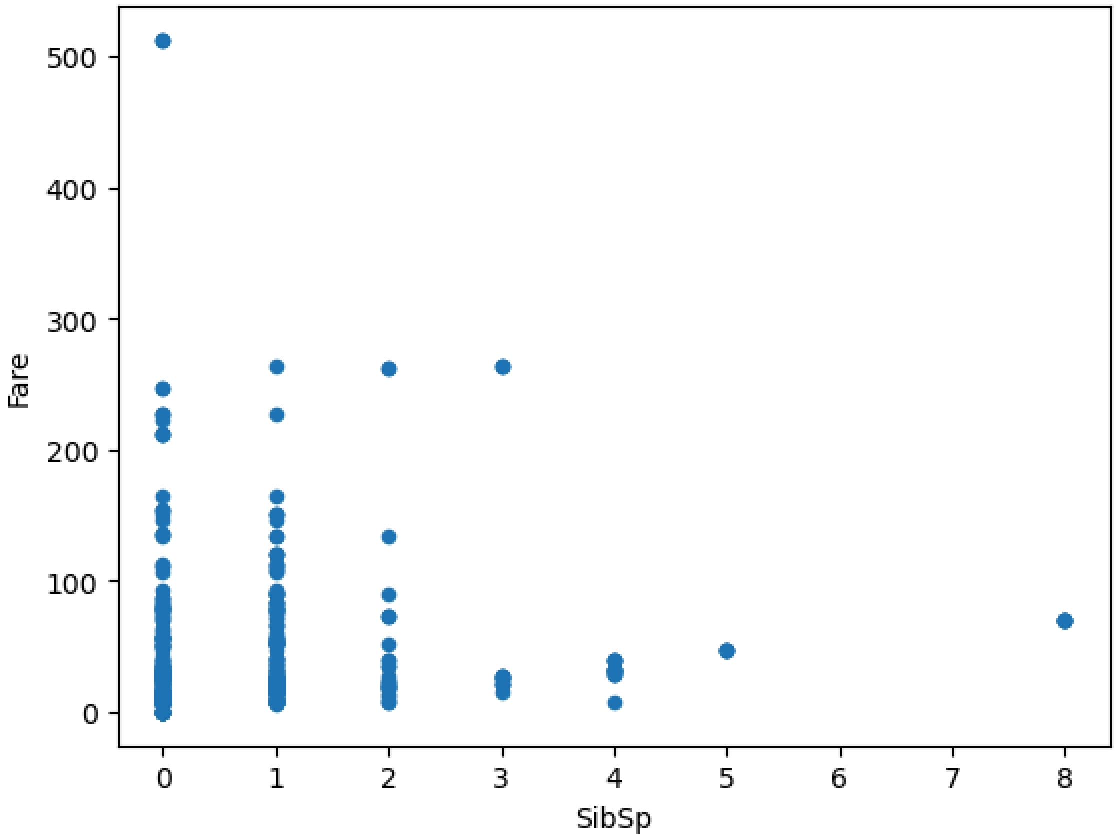
In [56]:
```
dt.plot.scatter('Age','SibSp')
```

Out[56]: <Axes: xlabel='Age', ylabel='SibSp'>



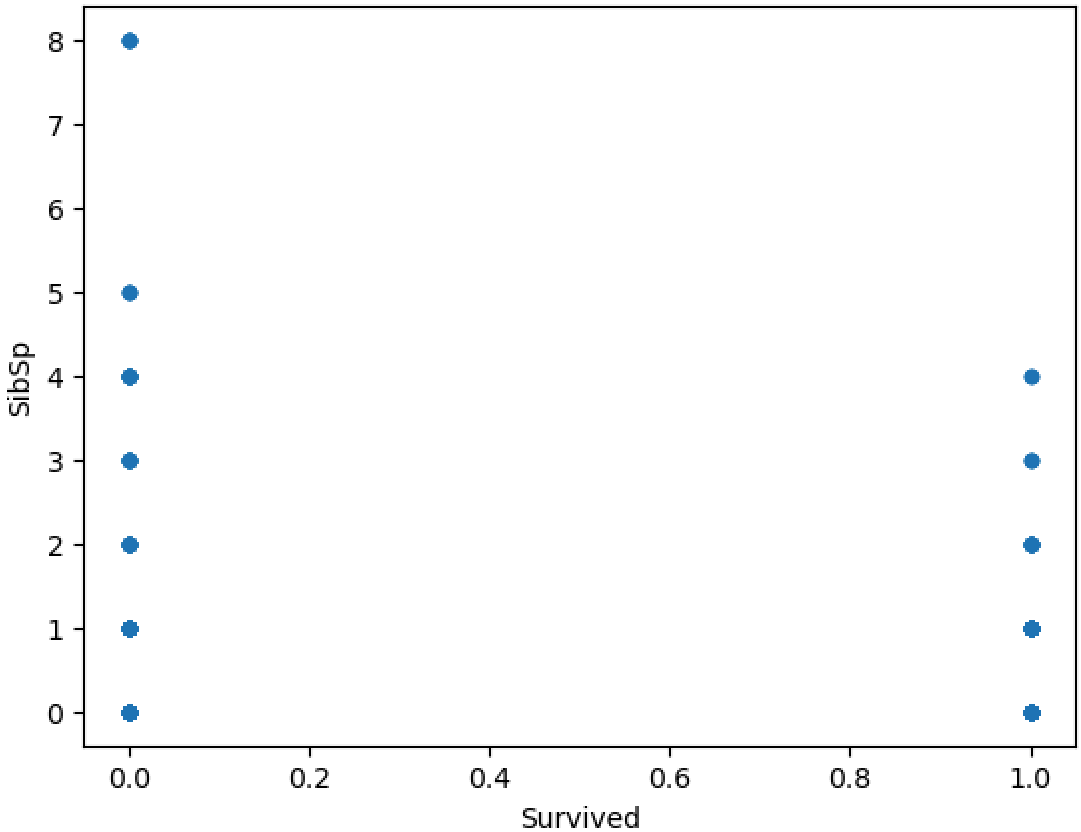
In [60]:
```
dt.plot.scatter('SibSp','Fare')
```

Out[60]:
```
<Axes: xlabel='SibSp', ylabel='Fare'>
```



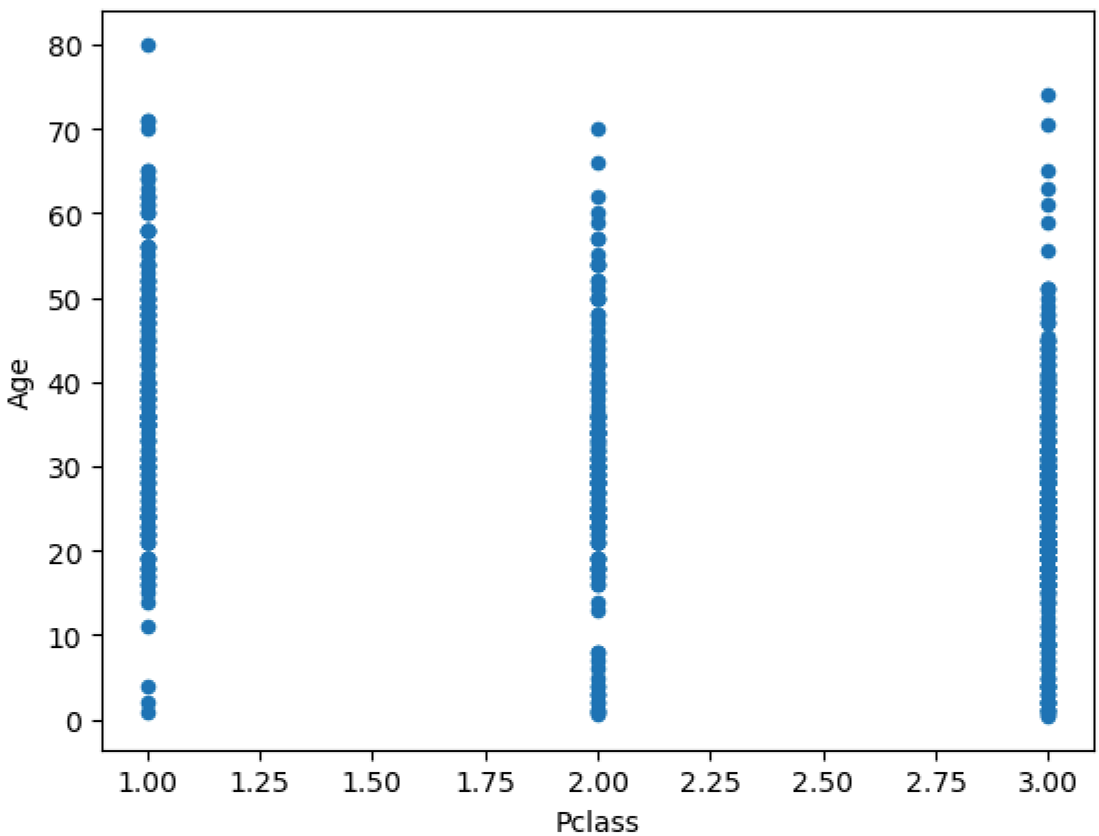
In [62]:
```
dt.plot.scatter('Survived','SibSp')
```

Out[62]: <Axes: xlabel='Survived', ylabel='SibSp'>



In [64]:
```
dt.plot.scatter(x='Pclass',y='Age')
```

Out[64]: <Axes: xlabel='Pclass', ylabel='Age'>


80
70
60
50
40
30
20
10
0
Age
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Pclass

Data Mining

Lab - 4

Nandani padsumbiya | 23010101182 | 23-06-2025

Step 1. Import the necessary libraries

In [3]:
```
import numpy as np
```

In [5]:
```
import pandas as pd
```

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called chipo.

In [7]:
```
# url = 'https://raw.githubusercontent.com/justmarkham/DAT8/master/data/chipot
```

In [7]:
```
chipo = pd.read_csv("https://raw.githubusercontent.com/justmarkham/DAT8/master
chipo
```

Out[7]:

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98
...	...	...	...	...	...
4617	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Sour ...	$11.75
4618	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...	$11.75
4619	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	$11.25
4620	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Lettu...	$8.75
4621	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	$8.75

4622 rows × 5 columns

Step 4. See the first 10 entries

In [18]:

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

Out[18]:

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98
5	3	1	Chicken Bowl	[Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...	$10.98
6	3	1	Side of Chips	NaN	$1.69
7	4	1	Steak Burrito	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	$11.75
8	4	1	Steak Soft Tacos	[Tomatillo Green Chili Salsa, [Pinto Beans, Ch...	$9.25
9	5	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Pinto...	$9.25

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

In [20]:

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

Out[20]: 4622

In [22]:

```
# 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?

In [24]:
```
chipo.shape[1]
```

Out[24]: 5

Step 7. Print the name of all the columns.

In [28]:
```
chipo.columns
```

Out[28]:
```
Index(['order_id', 'quantity', 'item_name', 'choice_description',
       'item_price'],
      dtype='object')
```

Step 8. How is the dataset indexed?

In [30]:
```
chipo.index
```

Out[30]: RangeIndex(start=0, stop=4622, step=1)

Step 9. Number of Unique Items ?

In [36]:
```
chipo.item_name.nunique()
```

Out[36]: 50

Step 10. Which was the most-ordered item?

In [42]:
```
c = chipo.groupby('item_name')
c = c.sum()
c = c.sort_values(['quantity'],ascending=False)
c.head(1)
```

Out[42]:

	order_id	quantity	choice_description	item_price
item_name				
Chicken Bowl	713926	761	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98 $10.98 $11.25 $8.75 $8.49 $11.25 $8.75 ...

In [44]:
```
c = chipo.groupby('item_name')
c.get_group("Chicken Bowl")
```

Out[44]:

	order_id	quantity	item_name	choice_description	item_price
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98
5	3	1	Chicken Bowl	[Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...	$10.98
13	7	1	Chicken Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	$11.25
19	10	1	Chicken Bowl	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	$8.75
26	13	1	Chicken Bowl	[Roasted Chili Corn Salsa (Medium), [Pinto Bea...	$8.49
...	...	...	...	...	...
4590	1825	1	Chicken Bowl	[Roasted Chili Corn Salsa, [Rice, Black Beans,...	$11.25
4591	1825	1	Chicken Bowl	[Tomatillo Red Chili Salsa, [Rice, Black Beans...	$8.75
4595	1826	1	Chicken Bowl	[Tomatillo Green Chili Salsa, [Rice, Black Bea...	$8.75
4599	1827	1	Chicken Bowl	[Roasted Chili Corn Salsa, [Cheese, Lettuce]]	$8.75
4604	1828	1	Chicken Bowl	[Fresh Tomato Salsa, [Rice, Black Beans, Chees...	$8.75

726 rows × 5 columns

Step 11. How many items were orderd in total?

In [46]:
```
chipo.quantity.sum()
```

Out[46]:
```
4972
```

Step 12. Turn the item price into a float

Step 12.a. Check the item price type

In [52]:
```
chipo['item_price'].dtype
#or chipo.item_price.dtypes
```

Out[52]:
```
dtype('O')
```

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

In [20]:
```
#dollarize = lambda x : float(x[1:-1]) -1 for till the last
#.apply is for applyig all rows
dollarize = lambda x : float(x[1:])
chipo.item_price = chipo.item_price.apply(dollarize)
```

In [22]:
```
chipo
```

Out[22]:

	order_id	quantity	item_name	choice_description	item_price	revenue
0	1	1	Chips and Fresh Tomato Salsa	NaN	2.39	$2.39
1	1	1	Izze	[Clementine]	3.39	$3.39
2	1	1	Nantucket Nectar	[Apple]	3.39	$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	2.39	$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	16.98	$16.98 $16.98
...	...	...	...	...	...	...
4617	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Sour ...	11.75	$11.75
4618	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...	11.75	$11.75
4619	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	11.25	$11.25
4620	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Lettu...	8.75	$8.75
4621	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	8.75	$8.75

4622 rows × 6 columns

Step 12.c. Check the item price type

In [24]:
```
chipo.item_price.dtype
```

Out[24]: dtype('float64')

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

In [26]:
```
chipo['revenue'] = (chipo['quantity']*chipo['item_price'])
chipo['revenue'].sum()
```

Out[26]: 39237.02

Step 15. How many orders were made ?

In [69]:
```
chipo.order_id.nunique()
```

Out[69]: 1834

In [71]:
```
orders = chipo.order_id.value_counts().count()
orders
```

Out[71]: 1834

Step 17. How many different choice descriptions are there?

In [73]:
```
chipo.choice_description.nunique()
```

Out[73]: 1043

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

In [89]:
```
df = chipo.groupby('item_name').quantity.sum()
df
```

```
Out[89]: item_name
6 Pack Soft Drink                          55
Barbacoa Bowl                              66
Barbacoa Burrito                           91
Barbacoa Crispy Tacos                      12
Barbacoa Salad Bowl                        10
Barbacoa Soft Tacos                        25
Bottled Water                             211
Bowl                                        4
Burrito                                     6
Canned Soda                               126
Canned Soft Drink                         351
Carnitas Bowl                              71
Carnitas Burrito                           60
Carnitas Crispy Tacos                       8
Carnitas Salad                              1
Carnitas Salad Bowl                         6
Carnitas Soft Tacos                        40
Chicken Bowl                              761
Chicken Burrito                           591
Chicken Crispy Tacos                       50
Chicken Salad                               9
Chicken Salad Bowl                        123
Chicken Soft Tacos                        120
Chips                                     230
Chips and Fresh Tomato Salsa              130
Chips and Guacamole                       506
Chips and Mild Fresh Tomato Salsa           1
Chips and Roasted Chili Corn Salsa         23
Chips and Roasted Chili-Corn Salsa         18
Chips and Tomatillo Green Chili Salsa      45
Chips and Tomatillo Red Chili Salsa        50
Chips and Tomatillo-Green Chili Salsa      33
Chips and Tomatillo-Red Chili Salsa        25
Crispy Tacos                                2
Izze                                       20
Nantucket Nectar                           29
Salad                                       2
Side of Chips                             110
Steak Bowl                                221
Steak Burrito                             386
Steak Crispy Tacos                         36
Steak Salad                                 4
Steak Salad Bowl                           31
Steak Soft Tacos                           56
Veggie Bowl                                87
Veggie Burrito                             97
Veggie Crispy Tacos                         1
Veggie Salad                                6
Veggie Salad Bowl                          18
Veggie Soft Tacos                           8
Name: quantity, dtype: int64
```

In [91]:
```
df[df>100]
```

Out[91]:
```
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?

In [30]:
```
# Solution 1
# Grouping by order_id and summing the revenue per order
df = chipo.groupby('order_id').revenue.sum()

average_revenue = df.mean()

print("Average revenue per order:", average_revenue)
```

```
Average revenue per order: 21.39423118865867
```

In [95]:
```
# Solution 2
avg_revenue = chipo['revenue'].sum()/chipo['order_id'].nunique()
avg_revenue
```

Out[95]:
```
21.39423118865867
```

In [ ]:
```

```

Lab - 4 - Data Preprocessing

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

In [3]:
```
import pandas as pd
```

In [4]:
```
df = pd.read_csv("titanic.csv")
df
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticl
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	175
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STC 31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	3734
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	3703

891 rows × 12 columns

In [5]:

```
df.tail(5)
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticke
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	21153
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	11205
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C 660
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	11136
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	37037

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

In [7]:

```
#delete records(row) which contains null values(not physible)(defaul delete ro
#with dropna
data_withdropna = df.copy()
data_withdropna = data_withdropna.dropna()
data_withdropna
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Tick
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	175
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	1138
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	174
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	95
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	1137
...	...	...	...	...	...	...	...	...	
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	117
872	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	6
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	117
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	1120
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	1113

183 rows × 12 columns

In [8]:

```
# axis=1 so it deletes column which contain 1 or more null value
data_withdropna = df.copy()
```

```
data_withdropna = data_withdropna.dropna(how='any',axis=1)
data_withdropna
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	1	0	A/5 21171	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	1	0	PC 17599	7
2	3	1	3	Heikkinen, Miss. Laina	female	0	0	STON/O2. 3101282	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	1	0	113803	5
4	5	0	3	Allen, Mr. William Henry	male	0	0	373450	
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	0	0	211536	1
887	888	1	1	Graham, Miss. Margaret Edith	female	0	0	112053	3
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	1	2	W./C. 6607	2
889	890	1	1	Behr, Mr. Karl Howell	male	0	0	111369	3
890	891	0	3	Dooley, Mr. Patrick	male	0	0	370376	

891 rows × 9 columns

In [9]:

```
# axis=1 so it deletes columns in which all the values are null
data_withdropna = df.copy()
```

```python
data_withdropna = data_withdropna.dropna(how='all',axis=1)
data_withdropna
```

Out[9]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticl
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	175
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STO 31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	3734
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	3703

891 rows × 12 columns

In [10]:

```python
#with fillna
#globally
```

```python
data_withfillna = df.copy()
data_withfillna = data_withfillna.fillna(1000) #----->insert 1000 where value
data_withfillna
```

Out[10]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	T
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	1
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	S 310
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	11
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	37
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	21
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	11
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	1000.0	1	2	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	11
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	37

891 rows × 12 columns

In [11]:

```python
#ColumnWise
```

```python
data_withfillna = df.copy()
data_withfillna = data_withfillna.fillna({'Age':100,'Cabin':'Not Available'})
data_withfillna
```

Out[11]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ti
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	21
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	17
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	ST 3101
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	100.0	1	2	W 6
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370

891 rows × 12 columns

In [12]:

```python
data_withfillna = df.copy()
```

```python
meanAge = data_withfillna.Age.mean()
print(meanAge)
data_withfillna = data_withfillna.fillna({'Age':meanAge,'Cabin':'Not Available
data_withfillna
```

```
29.69911764705882
```

Out[12]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0
...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0

891 rows × 12 columns

In [13]:

```
data_withfillna = df.copy()
meanAge = data_withfillna.Age.mean()
modeCabin = data_withfillna.Cabin.mode()[0] #mode can be multible but we will
print(meanAge)
print(modeCabin)
data_withfillna = data_withfillna.fillna({'Age':meanAge,'Cabin':modeCabin})
data_withfillna
```

```
29.69911764705882
B96 B98
```

Out[13]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0
...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0

891 rows × 12 columns

In [14]:

```
#interpolate only works on numeric not on strings and objects
#interpolate uses for replace columns null value according their up and down r
data_interpolate = df.copy()
data_interpolate = data_interpolate.interpolate()
data_interpolate
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_14212\2271286748.py:4: FutureWarnin
g: DataFrame.interpolate with object dtype is deprecated and will raise in a fu
ture version. Call obj.infer_objects(copy=False) before interpolating instead.
  data_interpolate = data_interpolate.interpolate()
```

Out[14]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticl
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	175
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STC (31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	3734
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	22.5	1	2	W 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	3703

891 rows × 12 columns

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

In [16]:
```
#With min-max
data = df.copy()

minAge = int(data.Age.min())
maxAge = int(data.Age.max())

print('Min Age =',minAge) # 0
print('Max Age =',maxAge) # 80

Ages = data.Age

data['MinMaxAge'] = (Ages - minAge) / (maxAge - minAge)
data
```

```
Min Age = 0
Max Age = 80
```

Out[16]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticl
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	175
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STO 3101
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W 6
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370

891 rows × 13 columns

In [37]:

```
# with decimal scaling
data = df.copy()
maxAge = data.Age.max()
noOfDigits = len(str(int(maxAge)))
Ages = data.Age
```

```python
print(noOfDigits)
data['DecimalScalingAge'] = Ages/(10**noOfDigits)
data
```

2

Out[37]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticl
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	175
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	ST((31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	3734
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	3703

891 rows × 13 columns

In [18]:
```
len('sdfghj')
```

Out[18]:
```
6
```

In [19]:
```
data_original = df.copy()
data_original = data_original.interpolate()
meanAge = data_original['Age'].mean()
stdAge = data_original['Age'].std()
data_original['AgeByZScore'] = (data_original['Age']-meanAge)/stdAge
data_original
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_14212\2574547447.py:2: FutureWarnin
g: DataFrame.interpolate with object dtype is deprecated and will raise in a fu
ture version. Call obj.infer_objects(copy=False) before interpolating instead.
  data_original = data_original.interpolate()
```

Out[19]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticl
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	175
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STC 31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	3734
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	22.5	1	2	W 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	3703

891 rows × 13 columns

In [ ]:

In [ ]:

Data Mining

Lab - 6

Nandani padsumbiya| 23010101182 | 07-07-2025


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

In [25]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Step 1: Load the Iris Dataset

In [17]:
```
iris = pd.read_csv('iris.csv')
iris
```

Out[17]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [27]:

```
print('Shape :',iris.shape)
```

```
Shape : (150, 5)
```

In [33]:

```
#map() is used to transform or remap a column’s values easily (set other value
X = iris.drop(columns="species")
Y = iris['species'].map({
    'setosa':0,
    'versicolor':1,
    'virginica':2
})

print('Original Shape :',X.shape)
```

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

Step 2: Standardize the data (zero mean)

In [39]:

```
X_meaned = X - np.mean(X, axis=0)
print('Data after mean(first 5 rows) :\n',X_meaned.head(5))
```

```
Data after mean(first 5 rows) :
    sepal_length  sepal_width  petal_length  petal_width
0     -0.743333     0.442667        -2.358    -0.999333
1     -0.943333    -0.057333        -2.358    -0.999333
2     -1.143333     0.142667        -2.458    -0.999333
3     -1.243333     0.042667        -2.258    -0.999333
4     -0.843333     0.542667        -2.358    -0.999333
```

Step 3: Compute the Covariance Matrix

In [47]:
```
#rowvar = row variance is by default true but we need to keep false here bcz w
cov_mat = np.cov(X_meaned,rowvar=False)
print('Covariance matrix shape :',cov_mat.shape)
```

```
Covariance matrix shape : (4, 4)
```

In [49]:
```
print('Covariance matrix :\n',cov_mat)
```

```
Covariance matrix :
 [[ 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

In [67]:
```
#linalg = linear algebra
#this fun returns two values 1st is eigenvalue and 2nd is eigenvector
eigen_values , eigen_vectors = np.linalg.eigh(cov_mat)

print('Eigenvalues :\n',eigen_values)
print('Eigenvectors (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: Sort eigenvalues and eigenvectors in descending order

In [69]:
```
sorted_index = np.argsort(eigen_value)[::-1]
sorted_eigenvalues = eigen_values[sorted_index]
sorted_eigenvactors = eigen_vectors[:,sorted_index]

print(sorted_index)
print(sorted_eigenvalues)
print(sorted_eigenvactors)
```

```
[3 2 1 0]
[4.22824171 0.24267075 0.0782095  0.02383509]
[[-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)

In [82]:
```
k = 2
eigenvector_subset = sorted_eigenvactors[:,0:k]
print(eigenvector_subset)
```

```
[[-0.36138659  0.65658877]
 [ 0.08452251  0.73016143]
 [-0.85667061 -0.17337266]
 [-0.3582892  -0.07548102]]
```

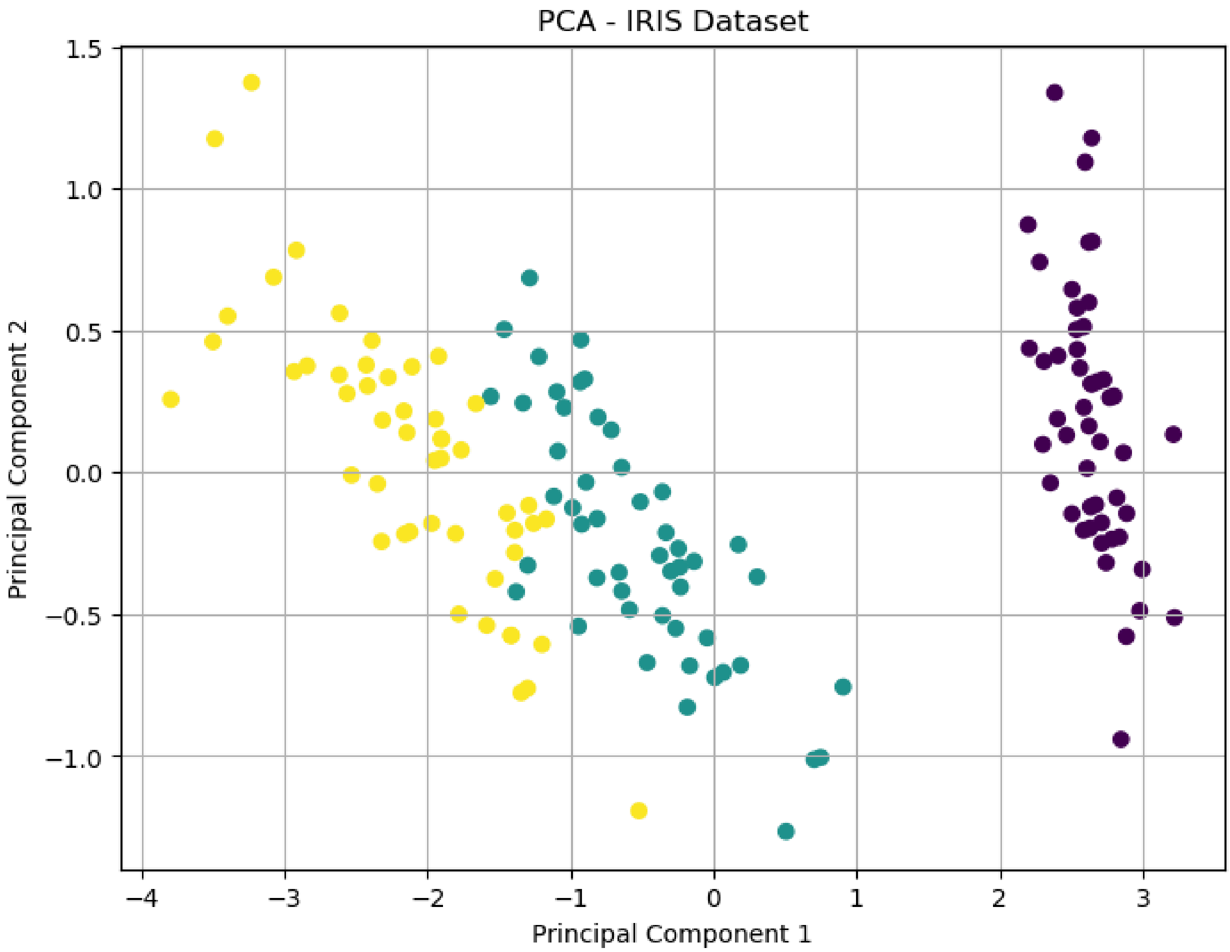
Step 7: Project the data onto the top k eigenvectors

In [84]:
```
X_reduced = np.dot(X_meaned,eigenvector_subset)
print('Reduced data shape :',X_reduced.shape)
```

```
Reduced data shape : (150, 2)
```

Step 8: Plot the PCA-Reduced Data

In [92]:
```
plt.figure(figsize=(8,6))
plt.scatter(X_reduced[:, 0],X_reduced[:, 1],c = Y)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal 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

In [23]:

```
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]
```

In [ ]:

Data Mining

Lab - 7 (Part 2)

Nandani padsumbiya | 23010101182

Step 1: Load the Dataset

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

In [41]:
```
import pandas as pd
df = pd.read_csv('Tdata.csv')
df.head()
```

Out[41]:

	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).

In [49]:
```
data = df.copy()
data_items = data.drop('Transaction',axis=1) #data.drop(column = ['Transaction
data_items
```

Out[49]:

	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.

In [51]:

```
data_items.sum()
```

Out[51]:

```
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.

In [80]:

```
from itertools import combinations

def find_frequent_itemsets(data,min_support):
    n = len(data)
    result = []

    for k in [1,2,3]:
        for items in combinations(data.columns,k):
            mask = data[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.

In [94]:
```python
frequent_itemsets = find_frequent_itemsets(data_items,min_support = 0.6)

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

```
items ('bread',) mask 5
items ('butter',) mask 3
items ('coffee',) mask 2
items ('eggs',) mask 2
items ('jam',) mask 2
items ('milk',) mask 3
items ('bread', 'butter') mask 3
items ('bread', 'coffee') mask 1
items ('bread', 'eggs') mask 1
items ('bread', 'jam') mask 1
items ('bread', 'milk') mask 3
items ('butter', 'coffee') mask 0
items ('butter', 'eggs') mask 0
items ('butter', 'jam') mask 1
items ('butter', 'milk') mask 2
items ('coffee', 'eggs') mask 1
items ('coffee', 'jam') mask 1
items ('coffee', 'milk') mask 0
items ('eggs', 'jam') mask 1
items ('eggs', 'milk') mask 1
items ('jam', 'milk') mask 0
items ('bread', 'butter', 'coffee') mask 0
items ('bread', 'butter', 'eggs') mask 0
items ('bread', 'butter', 'jam') mask 1
items ('bread', 'butter', 'milk') mask 2
items ('bread', 'coffee', 'eggs') mask 0
items ('bread', 'coffee', 'jam') mask 0
items ('bread', 'coffee', 'milk') mask 0
items ('bread', 'eggs', 'jam') mask 0
items ('bread', 'eggs', 'milk') mask 1
items ('bread', 'jam', 'milk') mask 0
items ('butter', 'coffee', 'eggs') mask 0
items ('butter', 'coffee', 'jam') mask 0
items ('butter', 'coffee', 'milk') mask 0
items ('butter', 'eggs', 'jam') mask 0
items ('butter', 'eggs', 'milk') mask 0
items ('butter', 'jam', 'milk') mask 0
items ('coffee', 'eggs', 'jam') mask 1
items ('coffee', 'eggs', 'milk') mask 0
items ('coffee', 'jam', 'milk') mask 0
items ('eggs', 'jam', 'milk') mask 0
{'bread'} support: 0.83
```

Step 6 Display as a DataFrame

In [96]:
```python
result_data = pd.DataFrame(frequent_itemsets,columns = ['Itemset','Support'])
result_data
```

Out[96]:

	Itemset	Support
0	(bread)	0.83

In [ ]:

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

In [ ]:

Extra : - > Define Apriori Function without itertools

In [104…

```
import pandas as pd

# Recreate your dataset
data = pd.DataFrame({
    'bread':  [1, 1, 1, 1, 1, 0],
    'butter': [1, 1, 0, 1, 0, 0],
    'coffee': [0, 0, 0, 0, 1, 1],
    'eggs':   [0, 0, 1, 0, 0, 1],
    'jam':  [0, 1, 0, 0, 0, 1],
    'milk': [1 ,0 ,1, 1, 0, 0]# guessing the last column is juice
})

min_support = 0.6  # for example, at least 30% of baskets

frequent_itemsets = find_frequent_itemsets(data, min_support)

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

```
items ('bread',) mask 5
items ('butter',) mask 3
items ('coffee',) mask 2
items ('eggs',) mask 2
items ('jam',) mask 2
items ('milk',) mask 3
items ('bread', 'butter') mask 3
items ('bread', 'coffee') mask 1
items ('bread', 'eggs') mask 1
items ('bread', 'jam') mask 1
items ('bread', 'milk') mask 3
items ('butter', 'coffee') mask 0
items ('butter', 'eggs') mask 0
items ('butter', 'jam') mask 1
items ('butter', 'milk') mask 2
items ('coffee', 'eggs') mask 1
items ('coffee', 'jam') mask 1
items ('coffee', 'milk') mask 0
items ('eggs', 'jam') mask 1
items ('eggs', 'milk') mask 1
items ('jam', 'milk') mask 0
items ('bread', 'butter', 'coffee') mask 0
items ('bread', 'butter', 'eggs') mask 0
items ('bread', 'butter', 'jam') mask 1
items ('bread', 'butter', 'milk') mask 2
items ('bread', 'coffee', 'eggs') mask 0
items ('bread', 'coffee', 'jam') mask 0
items ('bread', 'coffee', 'milk') mask 0
items ('bread', 'eggs', 'jam') mask 0
items ('bread', 'eggs', 'milk') mask 1
items ('bread', 'jam', 'milk') mask 0
items ('butter', 'coffee', 'eggs') mask 0
items ('butter', 'coffee', 'jam') mask 0
items ('butter', 'coffee', 'milk') mask 0
items ('butter', 'eggs', 'jam') mask 0
items ('butter', 'eggs', 'milk') mask 0
items ('butter', 'jam', 'milk') mask 0
items ('coffee', 'eggs', 'jam') mask 1
items ('coffee', 'eggs', 'milk') mask 0
items ('coffee', 'jam', 'milk') mask 0
items ('eggs', 'jam', 'milk') mask 0
{'bread'}: 0.83
```

```
In [ ]:
```