Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Roll No - 130

Enrollment No - 24010101694

Lab - 1

Karan Sonagara

In [ ]:

Introduction to Pandas Library Function:

In [ ]:
```
The Pandas library is a powerful and popular Python library used for data manipulation and analysis.
```

Step-1 Import the pandas Libraries

In [4]:
```
import pandas as pd
```

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

In [30]:
```
import pandas as pd
```

Step-3 Read csv or excel File

In [27]:
```
##read CSV File

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

Out[27]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

Step-4 Print Data from csv or excel File

In [32]:
```
##read CSV File

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

Out[32]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

Step-5 See the First 10 Rows

In [36]:
```
df = pd.read_csv("titanic.csv")
df.head(10)
```

Out[36]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

Step-6 See the Last 10 Rows

In [43]:
```
df = pd.read_csv("titanic.csv")
df.tail(10)
```

Out[43]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
881	882	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.8958	NaN	S
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	7552	10.5167	NaN	S
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.5000	NaN	S
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.0500	NaN	S
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	NaN	Q
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

Step-7 Data type of each columns

In [194…

```
df.dtypes
```

Out[194…
```
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 [45]:
```
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
```

Step-9 Access a specific column

In [83]:
```
df["Age"]
```

Out[83]:
```
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

In [99]:
```
df = pd.read_csv("titanic.csv")
df.iloc[2]
```

Out[99]:
```
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 [121…
```
df.drop("Sex",axis=1,inplace=True)
```

Out[121...

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	35.0	0	0	373450	8.0500	NaN
...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	27.0	0	0	211536	13.0000	NaN
887	888	1	1	Graham, Miss. Margaret Edith	19.0	0	0	112053	30.0000	B42
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	NaN	1	2	W./C. 6607	23.4500	NaN
889	890	1	1	Behr, Mr. Karl Howell	26.0	0	0	111369	30.0000	C148
890	891	0	3	Dooley, Mr. Patrick	32.0	0	0	370376	7.7500	NaN

891 rows × 10 columns

Step-12 Create a new Column

In [126...

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

Out[126...

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	isCabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	False
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	True
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	False
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	True
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S	False
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S	True
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S	False
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C	True
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q	False

891 rows × 13 columns

Step-13 Perform Condition Selection on DataFrame

In [150...

```
df[df['Pclass']==1]
```

Out[150...

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	isCabin
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	True
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	True
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S	True
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S	True
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.5542	D35	S	True
872	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	695	5.0000	B51 B53 B55	S	True
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583	C50	C	True
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S	True
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C	True

216 rows × 13 columns

Step-14 Compute the sum of value

In [136...

```
df["Age"].sum()
```

Out[136… 21205.17

Step-15 Compute the mean of value

In [138…
```
df["Age"].mean()
```

Out[138… 29.69911764705882

Step-16 Count non-null value (column)

In [152…
```
df["Age"].count()
```

Out[152… 714

Step-17 Find Minimun or Maximum values

In [142…
```
df["Age"].min()
```

Out[142… 0.42

In [144…
```
df["Age"].max()
```

Out[144… 80.0

In [214…
```
df[(df['Pclass']==1) & (df['Age']>25)]
```

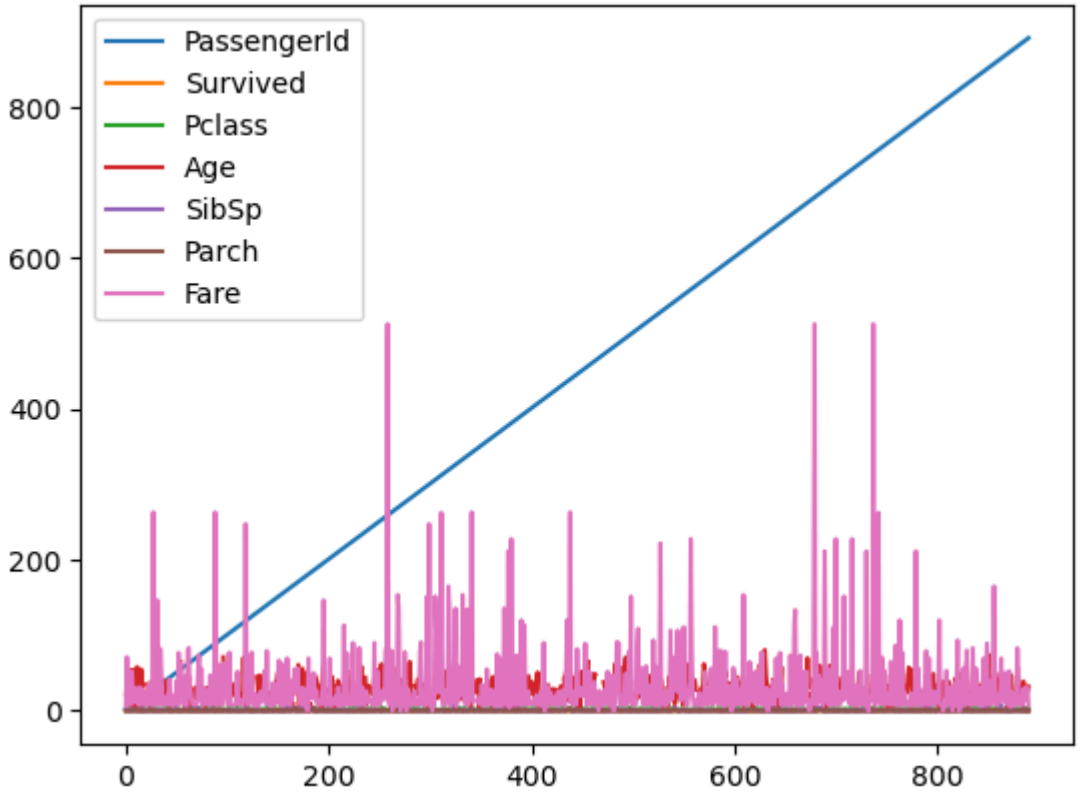
Out[214...

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	isCabin
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	True
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	True
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S	True
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S	True
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...
867	868	0	1	Roebling, Mr. Washington Augustus II	male	31.0	0	0	PC 17590	50.4958	A24	S	True
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.5542	D35	S	True
872	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	695	5.0000	B51 B53 B55	S	True
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583	C50	C	True
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C	True

144 rows × 13 columns

In [222...

```
import matplotlib.pyplot as plt
import pandas as df
df = pd.read_csv('titanic.csv')
df.plot()
plt.show()
```


PassengerId
Survived
Pclass
Age
SibSp
Parch
Fare
800
600
400
200
0
0
200
400
600
800

In [ ]:

In [ ]:

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Roll No - 130

Enrollment No - 24010101694

Lab - 2

Karan Sonagara

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 [1]:
```
import numpy as np
```

Step 2. Create a 1D array of numbers

In [19]:
```
arr = np.array([1, 2, 3])
arr
```

Out[19]:
```
array([1, 2, 3])
```

Step 3. Reshape 1D to 2D Array

In [11]:
```
arr3 = np.arange(12).reshape(3, 4)
arr3
```

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

Step 4. Create a Linspace array

In [7]:
```
arr4 = np.linspace(1, 1, 5)
arr4
```

Out[7]:
```
array([1., 1., 1., 1., 1.])
```

Step 5. Create a Random Numbered Array

In [9]:
```
arr5 = np.random.rand(3, 4)
arr5
```

Out[9]:
```
array([[0.17642223, 0.4339188 , 0.76623246, 0.24111004],
       [0.99867265, 0.65568381, 0.74620793, 0.42338742],
       [0.59718242, 0.44156543, 0.22621162, 0.85787908]])
```

Step 6. Create a Random Integer Array

In [27]:
```
arr6 = np.random.randint(2, 10, 10)
arr6
```

Out[27]:
```
array([5, 4, 8, 7, 3, 3, 5, 9, 2, 7])
```

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

In [29]:
```
arr6.argmin()
```

Out[29]: 8

In [31]:
```
arr6.min()
```

Out[31]: 2

In [33]:
```
arr6.max()
```

Out[33]: 9

In [35]:
```
arr6.argmax()
```

Out[35]: 7

Step 8. Indexing in 1D Array

In [53]:
```
arr8 = np.array([1, 2, 3, 4, 5, 6 ,7, 8, 9 , 10])
arr8[:7:2]
```

Out[53]:
```
array([1, 3, 5, 7])
```

Step 9. Indexing in 2D Array

In [71]:
```
arr9 = np.arange(12).reshape(3, 4)
arr9
```

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

In [79]:
```
arr9[::2]
```

Out[79]:
```
array([[ 0,  1,  2,  3],
       [ 8,  9, 10, 11]])
```

In [106…
```
arr9
```

Out[106…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [122…
```
arr9[:3:2]
```

Out[122…
```
array([[ 0,  1,  2,  3],
       [ 8,  9, 10, 11]])
```

Step 10. Conditional Selection

In [130…
```
arr3 = np.arange(100)
arr3[arr3>50]
```

Out[130…
```
array([51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
```

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

Pandas

Step 1. Import the necessary libraries

In [158…
```
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 [160…
```
users = pd.read_csv("https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user",sep="|",index_col='user_id')
users
```

Out[160…

user_id	age	gender	occupation	zip_code
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 [166…

```
df = pd.read_csv("https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user",sep='|')
df.head(10)
```

Out[166…

	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
5	6	42	M	executive	98101
6	7	57	M	administrator	91344
7	8	36	M	administrator	05201
8	9	29	M	student	01002
9	10	53	M	lawyer	90703

Step 5. See the last 10 entries

In [170…

```
df = pd.read_csv("https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user",sep='|')
df.tail(10)
```

Out[170…

	user_id	age	gender	occupation	zip_code
933	934	61	M	engineer	22902
934	935	42	M	doctor	66221
935	936	24	M	other	32789
936	937	48	M	educator	98072
937	938	38	F	technician	55038
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

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

In [172…
```
users.shape
```

Out[172…
```
(943, 4)
```

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

In [225…
```
users.count()
```

Out[225…
```
age           943
gender        943
occupation    943
zip_code      943
dtype: int64
```

Step 8. Print the name of all the columns.

In [178…
```
df = pd.read_csv("https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user",sep='|')
df.columns
```

Out[178…
```
Index(['user_id', 'age', 'gender', 'occupation', 'zip_code'], dtype='object')
```

Step 9. How is the dataset indexed?

In [182…
```
users.index
```

Out[182…
```
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 [184…
```
users.dtypes
```

Out[184…
```
age            int64
gender        object
occupation    object
zip_code      object
dtype: object
```

Step 11. Print only the occupation column

In [193…
```
users.occupation
users['occupation']
```

Out[193…
```
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 [202…
```
users['occupation'].nunique()
users['occupation'].unique()
```

Out[202…
```
array(['technician', 'other', 'writer', 'executive', 'administrator',
       'student', 'lawyer', 'educator', 'scientist', 'entertainment',
       'programmer', 'librarian', 'homemaker', 'artist', 'engineer',
       'marketing', 'none', 'healthcare', 'retired', 'salesman', 'doctor'],
      dtype=object)
```

Step 13. What is the most frequent occupation?

In [208…
```
users.occupation.value_counts().head(1)
```

Out[208…
```
occupation
student    196
Name: count, dtype: int64
```

Step 14. Summarize the DataFrame.

In [210…
```
users.describe()
```

Out[210…

	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 [213…

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

Out[213…

	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 [215…
```
users.occupation.describe()
```

Out[215…
```
count          943
unique          21
top       student
freq          196
Name: occupation, dtype: object
```

Step 17. What is the mean age of users?

In [219…
```
users.age.mean()
```

Out[219…
```
34.05196182396607
```

Step 18. What is the age with least occurrence?

In [221…
```
users.age.value_counts().tail()
```

Out[221…
```
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! 🚀

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Roll No - 130

Enrollment No - 24010101694

Lab - 3

Karan Sonagara

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

In [2]:
```
import pandas as pd
```

In [4]:
```
import numpy as ny
```

In [10]:
```
df = pd.read_csv("titanic.csv")
df.head(5)
```

Out[10]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

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

In [25]:
```
nominal = ["Name","Cabin","Embarked","Ticket","Sex"]
ordinal = ["Pclass"]
numeric = ["Age","Fare","SibSp","Parch"]
binary = ["Sex","Survived"]

print("Nominal",nominal)
print("Ordinal",ordinal)
print("Numeric",numeric)
print("Binary",binary)
```

```
Nominal ['Name', 'Cabin', 'Embarked', 'Ticket', 'Sex']
Ordinal ['Pclass']
Numeric ['Age', 'Fare', 'SibSp', 'Parch']
Binary ['Sex', 'Survived']
```

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

In [47]:
```
print("Survived values (Asymmetric Binary):")
print(df['Survived'].value_counts())
```

```
print("Sex Values (Symmetric Value):")
print(df['Sex'])
```

```
Survived values (Asymmetric Binary):
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 [65]:
```
arr = ["PassengerId","Survived","Pclass","SibSp","Parch","Fare"]
for i in arr:
    print(':::',i,':::',sep="|")
    print("   Mean",df[i].mean())
    print("   Min",df[i].min())
    print("   Mode",df[i].mode())
    print("   Max",df[i].max())
    print("   Standard Deviation",df[i].std())
    print("   Ranging",df[i].max()-1)
```

```
:::|PassengerId|:::
   Mean 446.0
   Min 1
   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
   Max 891
   Standard Deviation 257.3538420152301
   Ranging 890
:::|Survived|:::
   Mean 0.3838383838383838
   Min 0
   Mode 0    0
Name: Survived, dtype: int64
   Max 1
   Standard Deviation 0.4865924542648585
   Ranging 0
:::|Pclass|:::
   Mean 2.308641975308642
   Min 1
   Mode 0    3
Name: Pclass, dtype: int64
   Max 3
   Standard Deviation 0.8360712409770513
   Ranging 2
:::|SibSp|:::
   Mean 0.5230078563411896
   Min 0
   Mode 0    0
Name: SibSp, dtype: int64
   Max 8
   Standard Deviation 1.1027434322934275
```

```
   Ranging 7
:::|Parch|:::
   Mean 0.38159371492704824
   Min 0
   Mode 0    0
Name: Parch, dtype: int64
   Max 6
   Standard Deviation 0.8060572211299559
   Ranging 5
:::|Fare|:::
   Mean 32.204207968574636
   Min 0.0
   Mode 0    8.05
Name: Fare, dtype: float64
   Max 512.3292
   Standard Deviation 49.693428597180905
   Ranging 511.3292
```

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

In [70]:
```
df["Pclass"].value_counts()
```

Out[70]:
```
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 [92]:
```
df.describe(include=["object"])
```

Out[92]:

	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 [94]:
```
df.describe()
```

Out[94]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [88]:
```
df.describe(include="all")
```

Out[88]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
count	891.000000	891.000000	891.000000	891	891	714.000000	891.000000	891.000000	891	891.000000	204	889
unique	NaN	NaN	NaN	891	2	NaN	NaN	NaN	681	NaN	147	3
top	NaN	NaN	NaN	Braund, Mr. Owen Harris	male	NaN	NaN	NaN	347082	NaN	B96 B98	S
freq	NaN	NaN	NaN	1	577	NaN	NaN	NaN	7	NaN	4	644
mean	446.000000	0.383838	2.308642	NaN	NaN	29.699118	0.523008	0.381594	NaN	32.204208	NaN	NaN
std	257.353842	0.486592	0.836071	NaN	NaN	14.526497	1.102743	0.806057	NaN	49.693429	NaN	NaN
min	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.000000	0.000000	NaN	0.000000	NaN	NaN
25%	223.500000	0.000000	2.000000	NaN	NaN	20.125000	0.000000	0.000000	NaN	7.910400	NaN	NaN
50%	446.000000	0.000000	3.000000	NaN	NaN	28.000000	0.000000	0.000000	NaN	14.454200	NaN	NaN
75%	668.500000	1.000000	3.000000	NaN	NaN	38.000000	1.000000	0.000000	NaN	31.000000	NaN	NaN
max	891.000000	1.000000	3.000000	NaN	NaN	80.000000	8.000000	6.000000	NaN	512.329200	NaN	NaN

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

In [105…

```
print("\nCorrelation Matrix:\n")
df.corr(numeric_only=True)
```

```
Correlation Matrix:
```

Out[105...

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

In [109...

```
print(" Covariance Matrix:\n")
df.cov(numeric_only=True)
```

```
Covariance Matrix:
```

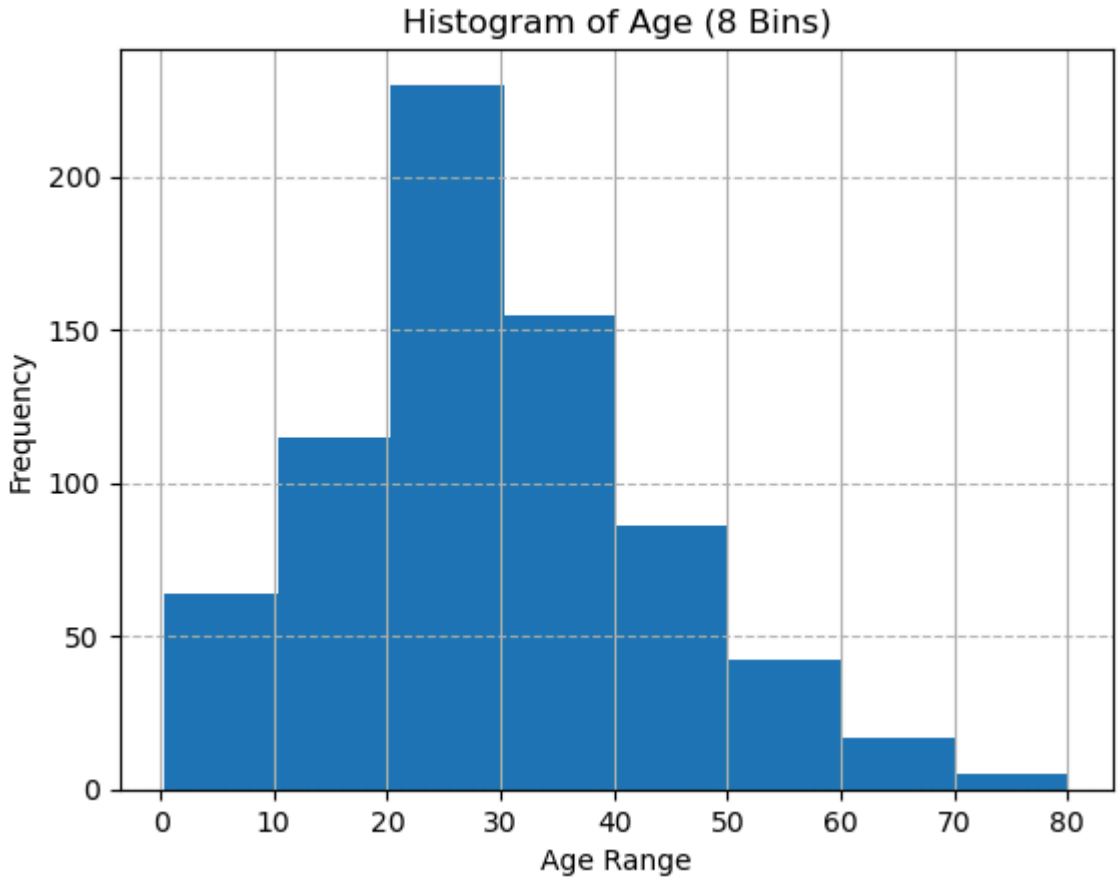
Out[109...

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	66231.000000	-0.626966	-7.561798	138.696504	-16.325843	-0.342697	161.883369
Survived	-0.626966	0.236772	-0.137703	-0.551296	-0.018954	0.032017	6.221787
Pclass	-7.561798	-0.137703	0.699015	-4.496004	0.076599	0.012429	-22.830196
Age	138.696504	-0.551296	-4.496004	211.019125	-4.163334	-2.344191	73.849030
SibSp	-16.325843	-0.018954	0.076599	-4.163334	1.216043	0.368739	8.748734
Parch	-0.342697	0.032017	0.012429	-2.344191	0.368739	0.649728	8.661052
Fare	161.883369	6.221787	-22.830196	73.849030	8.748734	8.661052	2469.436846

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

In [166…
```
import matplotlib.pyplot as plt
```

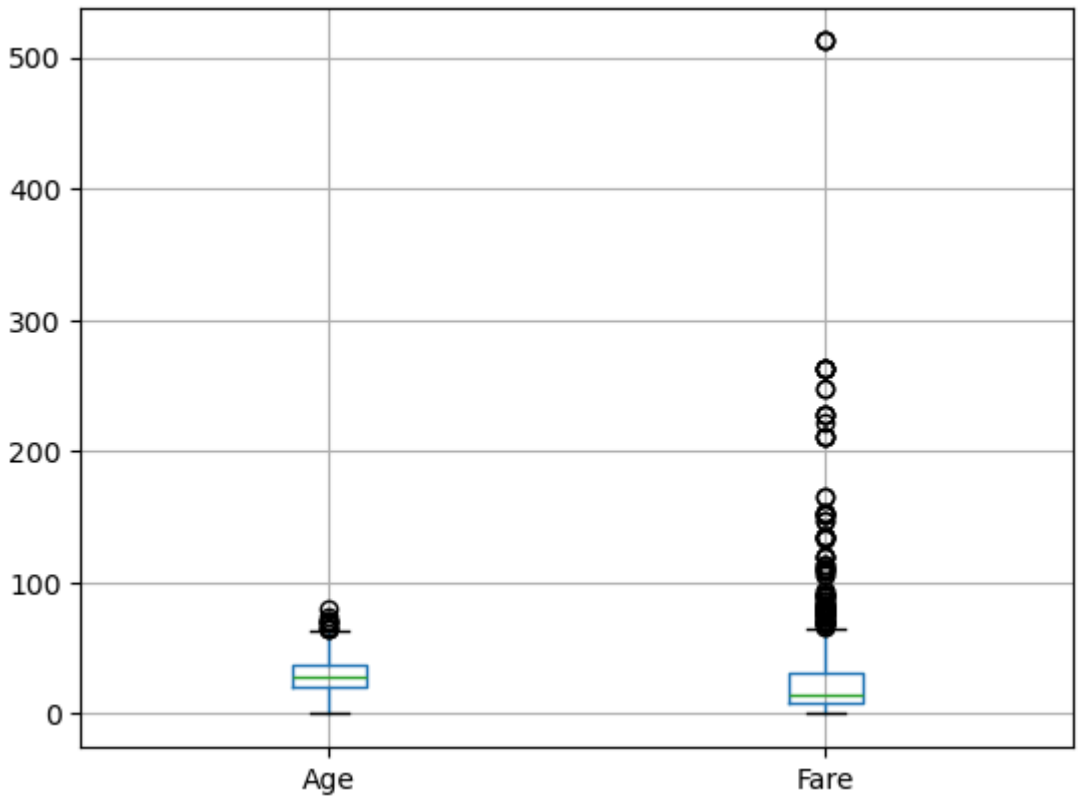
In [143…
```
df['Age'].dropna().hist(bins=8)
plt.title('Histogram of Age (8 Bins)')
plt.xlabel('Age Range')
plt.ylabel('Frequency')
plt.grid(axis='y', linestyle='--', alpha=0.9)
plt.show()
```



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

In [176…
```
df.boxplot(['Age','Fare'])
```

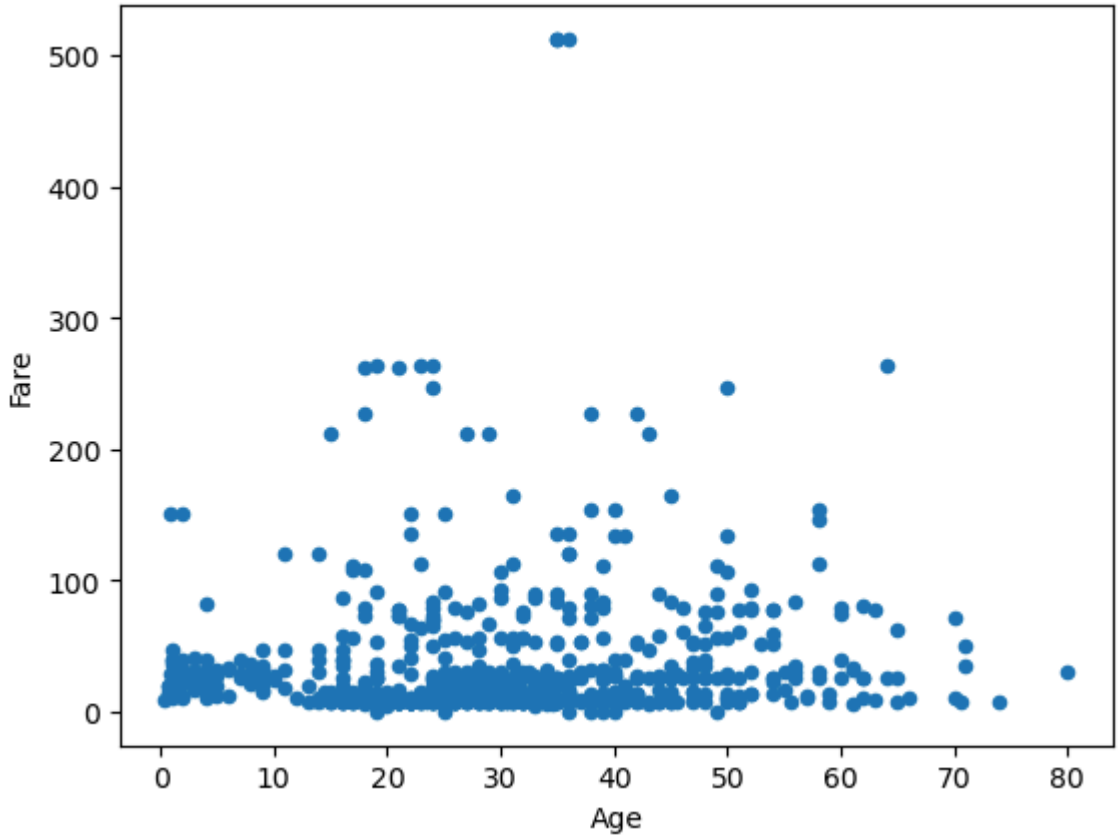
Out[176… <Axes: >

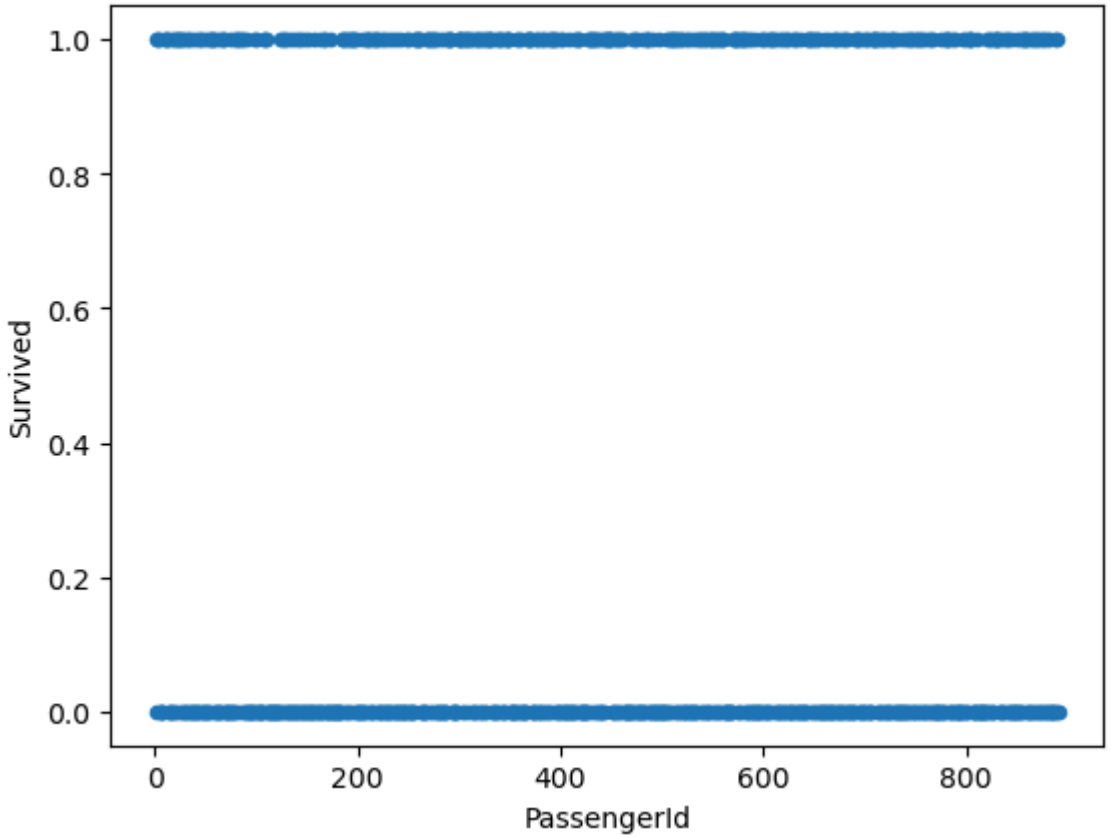


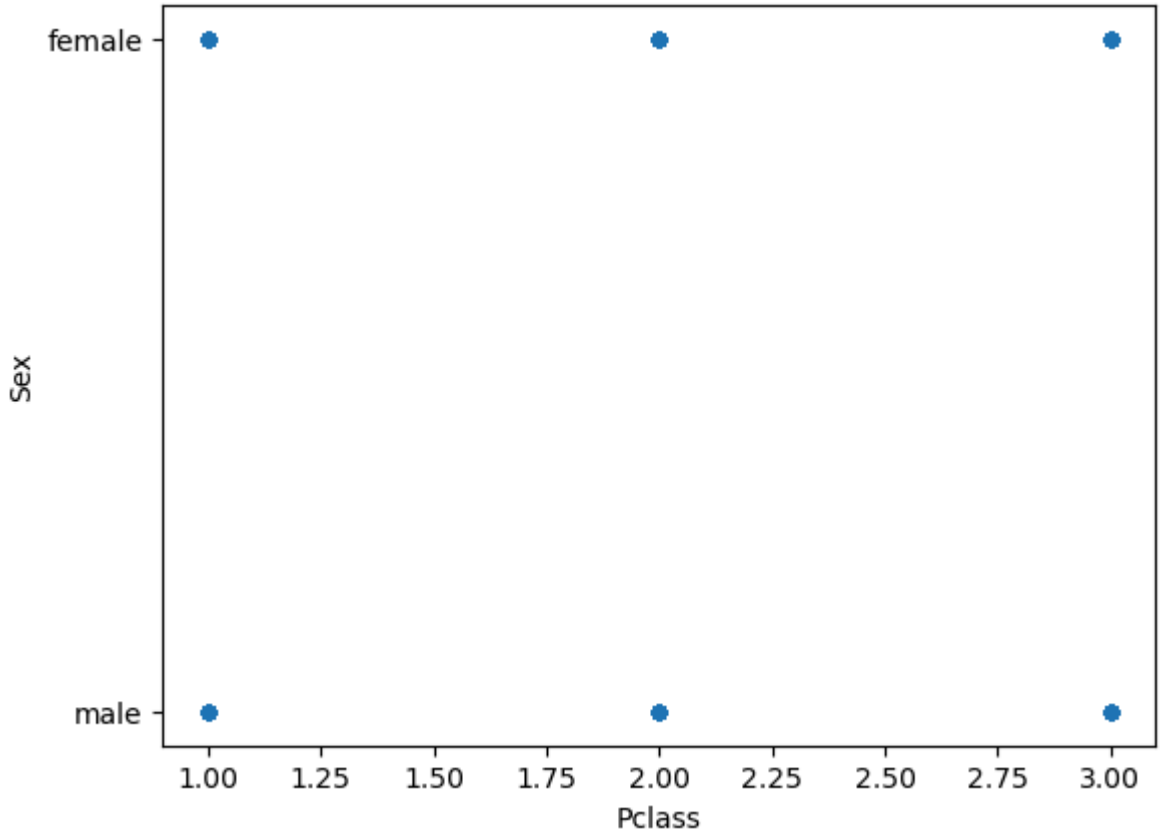
11) Display scatter plot for any 5 pair of attributes , we can use a scatter plot to visualize their joint distribution.

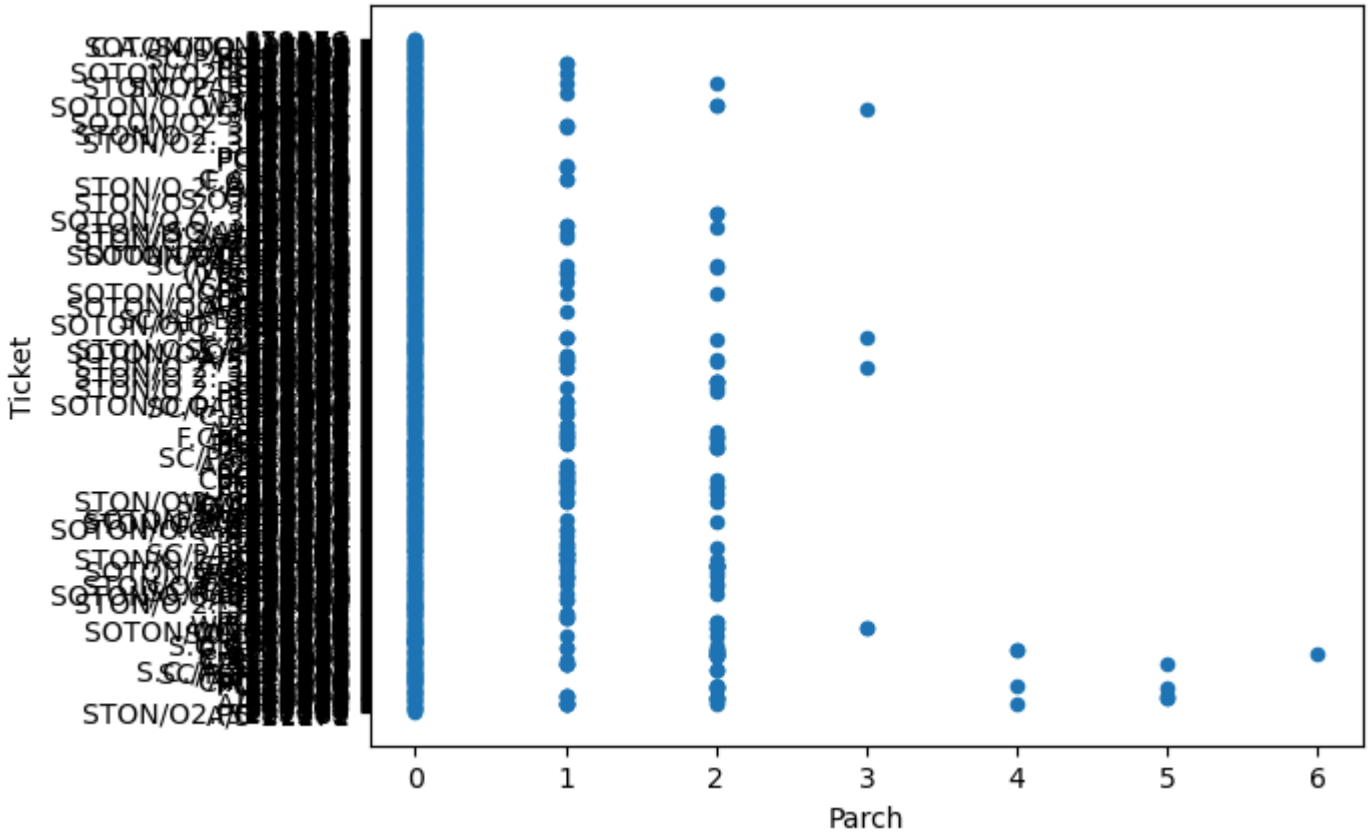
In [190…
```
df.plot.scatter(x="Age",y="Fare")
df.plot.scatter(x="PassengerId",y="Survived")
df.plot.scatter(x="Pclass",y="Sex")
df.plot.scatter(x="Parch",y="Ticket")
```

Out[190… <Axes: xlabel='Parch', ylabel='Ticket'>


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


1.0
0.8
0.6
0.4
0.2
0.0
Survived
0
200
400
600
800
PassengerId


female
male
Sex
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Pclass



In [ ]:

Roll No - 130

Enrollment No - 24010101694

Lab - 4

Karan Sonagara

Step 1. Import the necessary libraries

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

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called chipo.

In [5]:
```
df = pd.read_csv("https://raw.githubusercontent.com/justmarkham/DAT8/master/data/chipotle.tsv", sep='\t', encoding='cp1258')
df
```

Out[5]:

	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 [7]:
```
df.head(10)
```

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
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 [9]:
```
# Solution 1
print(df.shape[0])
```

```
4622
```

In [13]:
```
# Solution 2
df.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 [20]:
```
print(df.shape[1])
```

```
5
```

Step 7. Print the name of all the columns.

In [16]:
```
df.columns
```

Out[16]:
```
Index(['order_id', 'quantity', 'item_name', 'choice_description',
       'item_price'],
      dtype='object')
```

Step 8. How is the dataset indexed?

In [34]:
```
df.index
```

Out[34]:
```
RangeIndex(start=0, stop=4622, step=1)
```

Step 9. Number of Unique Items ?

In [50]:
```
df["item_name"].nunique()
```

Out[50]: 50

Step 10. Which was the most-ordered item?

In [72]:
```
c = df.groupby('item_name')
c = c.sum()
c = c.sort_values(['quantity'], ascending = False)
c[['quantity','order_id']].head(1)
```

Out[72]:

item_name	quantity	order_id
Chicken Bowl	761	713926

Step 11. How many items were orderd in total?

In [86]:
```
cf = df.quantity.sum()
cf
```

Out[86]: 4972

Step 12. Turn the item price into a float

Step 12.a. Check the item price type

In [52]:
```
df["item_price"].dtype
```

Out[52]: dtype('O')

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

In [5]:
```
df['item_price'] = df['item_price'].apply(lambda x: float(x.replace('$', '')) )
```

Step 12.c. Check the item price type

In [7]:
```
df["item_price"].dtype
```

Out[7]:
```
dtype('float64')
```

In [9]:
```
df['item_price']
```

Out[9]:
```
0        2.39
1        3.39
2        3.39
3        2.39
4       16.98
        ...
4617    11.75
4618    11.75
4619    11.25
4620     8.75
4621     8.75
Name: item_price, Length: 4622, dtype: float64
```

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

In [15]:
```
gt = (df['item_price'] * df['quantity']).sum()
print(gt)
```

```
39237.02
```

Step 15. How many orders were made ?

In [104…
```
total_orders = df['order_id'].nunique()
print("Total number of orders:", total_orders)
```

```
Total number of orders: 1834
```

Step 17. How many different choice descriptions are there?

In [56]:
```
df["choice_description"].nunique()
```

Out[56]: 1043

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

In [100…
```
item_totals = df.groupby('item_name')['quantity'].sum()
popular_items = item_totals[item_totals > 100]
print(popular_items)
```

```
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 [33]:
```
# Solution 1
df["item_revenue"] = df["item_price"] * df["quantity"]
rpd = df.groupby('order_id')['item_revenue'].sum()
ar = rpd.mean()
print(f"Averege Revenue Order: ${ar:.2f}")
```

```
Averege Revenue Order: $21.39
```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Roll No - 130

Enrollment No - 24010101694

Lab - 5

Karan Sonagara

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

In [2]:
```
import pandas as pd
import numpy as ny
```

In [4]:
```
df = pd.read_csv("titanic.csv", encoding='cp1258')
df
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [8]:
```
df.tail(5)
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

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

In [10]:
```
data_withdropna = df.dropna(how='all')  #how = any # how = all # axis=1
data_withdropna
```

Out[10]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [20]:
```
data_withfillna = df.fillna({'Age':30,'ICabin':"Not Avail"}) #({'Age':30,'ICabin':"Not Avail"})
data_withfillna
```

Out[20]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	30.0	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [30]:
```
datawithfillna = df.copy()
meanAge = datawithfillna['Age'].mean()
datawithfillna['Age'] = datawithfillna['Age'].fillna(meanAge)
datawithfillna
```

Out[30]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	28.0	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [34]:
```
datawithfillna = df.copy()
medianAge = datawithfillna['Age'].median()
datawithfillna['Age'] = datawithfillna['Age'].fillna(medianAge)
datawithfillna
```

Out[34]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	28.0	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [38]:
```
data_interpolate = df.copy()
data_interpolate['NewAge'] = data_interpolate['Age'].interpolate()
data_interpolate
```

Out[38]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	NewAge
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	22.0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	38.0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	26.0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	35.0
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	35.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S	27.0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S	19.0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S	22.5
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C	26.0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q	32.0

891 rows × 13 columns

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

In [40]:
```
data_minmax = df.copy()
age_main = data_minmax['Age']
```

```
age_min = age_main.min()
age_max = age_main.max()
data_minmax['MinMaxAge'] = (age_main-age_min) / (age_max-age_main)
data_minmax
```

Out[40]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	MinMaxAge
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0.372069
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	0.894762
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0.473704
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	0.768444
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0.768444
...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S	0.501509
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S	0.304590
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S	NaN
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C	0.473704
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q	0.657917

891 rows × 13 columns

In [64]:
```
data_max = df.copy()
age_main = data_max['Age']
age_max = age_main.abs().max()
temp = len(str(int(age_max)))
data_max["DecimalScaleAge"] = age_main/(10**temp)
data_max
```

Out[64]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	DecimalScaleAge
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0.22
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	0.38
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0.26
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	0.35
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0.35
...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S	0.27
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S	0.19
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S	NaN
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C	0.26

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	DecimalScaleAge
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q	0.32

891 rows × 13 columns

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Roll No - 130

Enrollment No - 24010101694

Lab - 6

Karan Sonagara

🔍 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.

Function	Purpose
`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.

Step 1: Load the Iris Dataset

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib as plt
```

In [2]:
```
iris=pd.read_csv("iris.csv")
iris
```

Out[2]:

	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 [9]:
```
x=iris.drop(columns="species")
y=iris["species"].map({
    'setosa':0,
    'versicolor':1,
    'virginica':2
})
print(x.shape)
```

```
(150, 4)
```

In [24]:
```
arrcov = [40,80,45,65,10,55]
print(np.cov(arrcov,rowvar=False))
```

```
574.1666666666667
```

Step 2: Standardize the data (zero mean)

In [13]:
```
x_meaned= x-np.mean(x,axis=0)
print(x_meaned)
x_meaned.shape
```

```
     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
..            ...          ...           ...          ...
145      0.856667    -0.057333         1.442     1.100667
146      0.456667    -0.557333         1.242     0.700667
147      0.656667    -0.057333         1.442     0.800667
148      0.356667     0.342667         1.642     1.100667
149      0.056667    -0.057333         1.342     0.600667

[150 rows x 4 columns]
```

Out[13]: (150, 4)

Step 3: Compute the Covariance Matrix

In [15]:
```
cov_mat=np.cov(x_meaned, rowvar=False)
print('Coveriance Matrix',cov_mat)
print('Coveriance Matrix',cov_mat.shape)
```

```
Coveriance 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]]
Coveriance Matrix (4, 4)
```

Step 4: Compute eigenvalues and eigenvectors

In [19]:
```
np.linalg.eigh(cov_mat)
```

Out[19]:
```
EighResult(eigenvalues=array([0.02383509, 0.0782095 , 0.24267075, 4.22824171]), eigenvectors=array([[ 0.31548719,  0.5820298
5,  0.65658877, -0.36138659],
       [-0.3197231 , -0.59791083,  0.73016143,  0.08452251],
       [-0.47983899, -0.07623608, -0.17337266, -0.85667061],
       [ 0.75365743, -0.54583143, -0.07548102, -0.3582892 ]]))
```

Step 5: Compute eigenvalues and eigenvectors

In [21]:
```
eigen_values,eigen_vector=np.linalg.eigh(cov_mat)
print("eigenvalues:\n",eigen_values)
print("eigenvectors:\n",eigen_vector)
```

```
eigenvalues:
 [0.02383509 0.0782095  0.24267075 4.22824171]
eigenvectors:
 [[ 0.31548719  0.58202985  0.65658877 -0.36138659]
 [-0.3197231  -0.59791083  0.73016143  0.08452251]
 [-0.47983899 -0.07623608 -0.17337266 -0.85667061]
 [ 0.75365743 -0.54583143 -0.07548102 -0.3582892 ]]
```

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

In [23]:
```
sorted_ind=np.argsort(eigen_values)[::-1]
sorted_eigenvalues=eigen_values[sorted_ind]
sorted_eigenvectors=eigen_vector[:,sorted_ind]
k=2
vector_subset=sorted_eigenvectors[:,0:k]
vector_subset.shape
```

Out[23]:
```
(4, 2)
```

Step 7: Project the data onto the top k eigenvectors

In [25]:
```
X_reduced=np.dot(x_meaned, vector_subset)
print(X_reduced)
k.shape
```

```
[[ 2.68412563  0.31939725]
 [ 2.71414169 -0.17700123]
 [ 2.88899057 -0.14494943]
 [ 2.74534286 -0.31829898]
 [ 2.72871654  0.32675451]
 [ 2.28085963  0.74133045]
 [ 2.82053775 -0.08946138]
 [ 2.62614497  0.16338496]
 [ 2.88638273 -0.57831175]
 [ 2.6727558  -0.11377425]
 [ 2.50694709  0.6450689 ]
 [ 2.61275523  0.01472994]
 [ 2.78610927 -0.235112  ]
 [ 3.22380374 -0.51139459]
 [ 2.64475039  1.17876464]
 [ 2.38603903  1.33806233]
 [ 2.62352788  0.81067951]
 [ 2.64829671  0.31184914]
 [ 2.19982032  0.87283904]
 [ 2.5879864   0.51356031]
 [ 2.31025622  0.39134594]
 [ 2.54370523  0.43299606]
 [ 3.21593942  0.13346807]
 [ 2.30273318  0.09870885]
 [ 2.35575405 -0.03728186]
 [ 2.50666891 -0.14601688]
 [ 2.46882007  0.13095149]
 [ 2.56231991  0.36771886]
 [ 2.63953472  0.31203998]
 [ 2.63198939 -0.19696122]
 [ 2.58739848 -0.20431849]
 [ 2.4099325   0.41092426]
 [ 2.64886233  0.81336382]
 [ 2.59873675  1.09314576]
 [ 2.63692688 -0.12132235]
 [ 2.86624165  0.06936447]
 [ 2.62523805  0.59937002]
 [ 2.80068412  0.26864374]
 [ 2.98050204 -0.48795834]
 [ 2.59000631  0.22904384]
 [ 2.77010243  0.26352753]
```

```
[ 2.84936871 -0.94096057]
[ 2.99740655 -0.34192606]
[ 2.40561449  0.18887143]
[ 2.20948924  0.43666314]
[ 2.71445143 -0.2502082 ]
[ 2.53814826  0.50377114]
[ 2.83946217 -0.22794557]
[ 2.54308575  0.57941002]
[ 2.70335978  0.10770608]
[-1.28482569  0.68516047]
[-0.93248853  0.31833364]
[-1.46430232  0.50426282]
[-0.18331772 -0.82795901]
[-1.08810326  0.07459068]
[-0.64166908 -0.41824687]
[-1.09506066  0.28346827]
[ 0.74912267 -1.00489096]
[-1.04413183  0.2283619 ]
[ 0.0087454  -0.72308191]
[ 0.50784088 -1.26597119]
[-0.51169856 -0.10398124]
[-0.26497651 -0.55003646]
[-0.98493451 -0.12481785]
[ 0.17392537 -0.25485421]
[-0.92786078  0.46717949]
[-0.66028376 -0.35296967]
[-0.23610499 -0.33361077]
[-0.94473373 -0.54314555]
[-0.04522698 -0.58383438]
[-1.11628318 -0.08461685]
[-0.35788842 -0.06892503]
[-1.29818388 -0.32778731]
[-0.92172892 -0.18273779]
[-0.71485333  0.14905594]
[-0.90017437  0.32850447]
[-1.33202444  0.24444088]
[-1.55780216  0.26749545]
[-0.81329065 -0.1633503 ]
[ 0.30558378 -0.36826219]
[ 0.06812649 -0.70517213]
[ 0.18962247 -0.68028676]
```

```
[-0.13642871 -0.31403244]
[-1.38002644 -0.42095429]
[-0.58800644 -0.48428742]
[-0.80685831  0.19418231]
[-1.22069088  0.40761959]
[-0.81509524 -0.37203706]
[-0.24595768 -0.2685244 ]
[-0.16641322 -0.68192672]
[-0.46480029 -0.67071154]
[-0.8908152  -0.03446444]
[-0.23054802 -0.40438585]
[ 0.70453176 -1.01224823]
[-0.35698149 -0.50491009]
[-0.33193448 -0.21265468]
[-0.37621565 -0.29321893]
[-0.64257601  0.01773819]
[ 0.90646986 -0.75609337]
[-0.29900084 -0.34889781]
[-2.53119273 -0.00984911]
[-1.41523588 -0.57491635]
[-2.61667602  0.34390315]
[-1.97153105 -0.1797279 ]
[-2.35000592 -0.04026095]
[-3.39703874  0.55083667]
[-0.52123224 -1.19275873]
[-2.93258707  0.3555    ]
[-2.32122882 -0.2438315 ]
[-2.91675097  0.78279195]
[-1.66177415  0.24222841]
[-1.80340195 -0.21563762]
[-2.1655918   0.21627559]
[-1.34616358 -0.77681835]
[-1.58592822 -0.53964071]
[-1.90445637  0.11925069]
[-1.94968906  0.04194326]
[-3.48705536  1.17573933]
[-3.79564542  0.25732297]
[-1.30079171 -0.76114964]
[-2.42781791  0.37819601]
[-1.19900111 -0.60609153]
[-3.49992004  0.4606741 ]
```

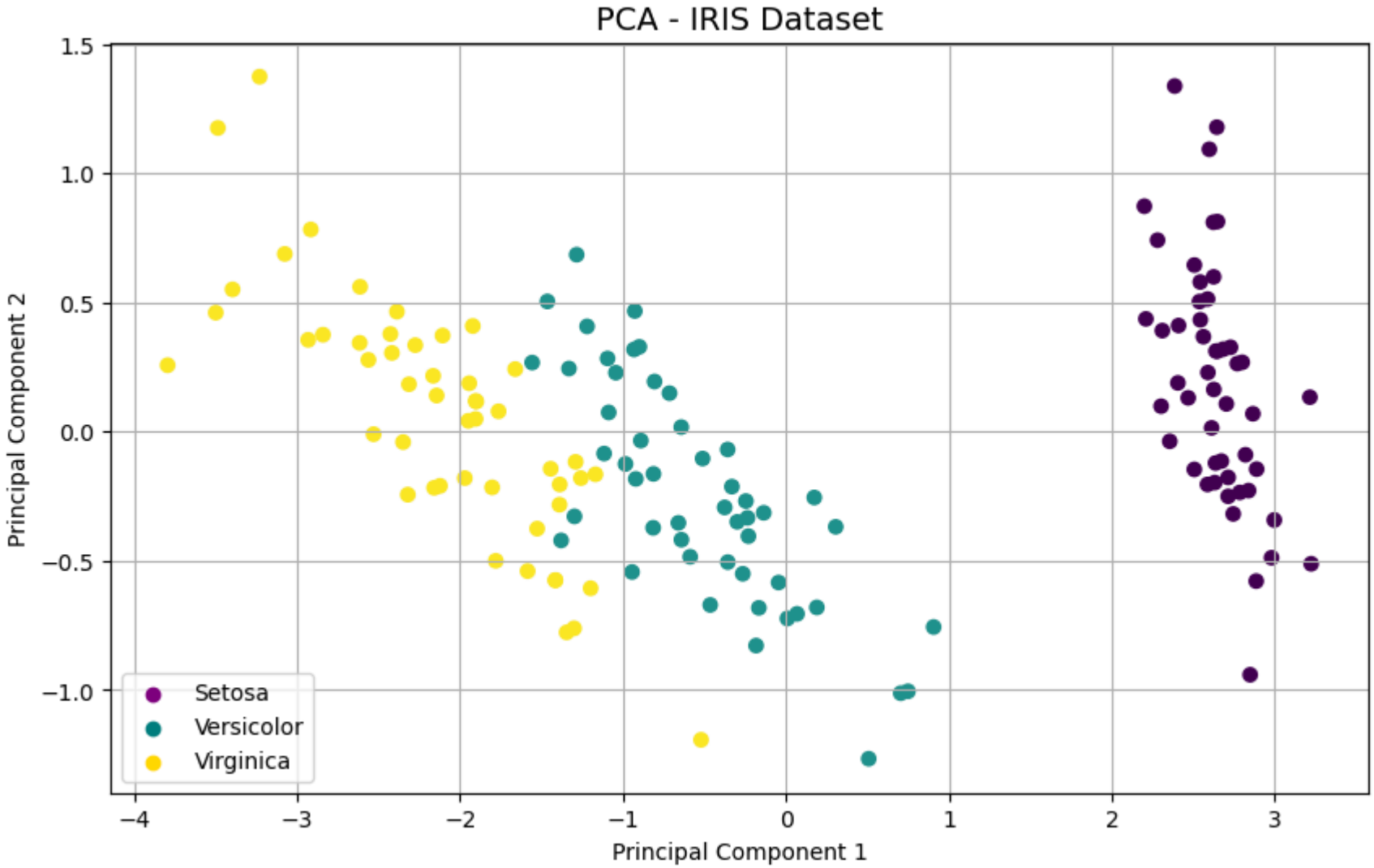
```
 [-1.38876613 -0.20439933]
 [-2.2754305   0.33499061]
 [-2.61409047  0.56090136]
 [-1.25850816 -0.17970479]
 [-1.29113206 -0.11666865]
 [-2.12360872 -0.20972948]
 [-2.38800302  0.4646398 ]
 [-2.84167278  0.37526917]
 [-3.23067366  1.37416509]
 [-2.15943764 -0.21727758]
 [-1.44416124 -0.14341341]
 [-1.78129481 -0.49990168]
 [-3.07649993  0.68808568]
 [-2.14424331  0.1400642 ]
 [-1.90509815  0.04930053]
 [-1.16932634 -0.16499026]
 [-2.10761114  0.37228787]
 [-2.31415471  0.18365128]
 [-1.9222678   0.40920347]
 [-1.41523588 -0.57491635]
 [-2.56301338  0.2778626 ]
 [-2.41874618  0.3047982 ]
 [-1.94410979  0.1875323 ]
 [-1.52716661 -0.37531698]
 [-1.76434572  0.07885885]
 [-1.90094161  0.11662796]
 [-1.39018886 -0.28266094]]
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[25], line 3
      1 X_reduced=np.dot(x_meaned, vector_subset)
      2 print(X_reduced)
----> 3 k.shape

AttributeError: 'int' object has no attribute 'shape'
```

Step 8: Plot the PCA-Reduced Data

In [27]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
scatter = plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=y, cmap='viridis')
plt.title("PCA - IRIS Dataset", fontsize=14)
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.grid(True)
# Optional: Custom legend
colors = ['purple', 'teal', 'gold']
labels = ['Setosa', 'Versicolor', 'Virginica']
for i in range(3):
    plt.scatter([], [], c=colors[i], label=labels[i])
plt.legend()
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 [31]:
```
data=[5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215]
print('Sotred data:',data)
```

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

18/07/25 Task 130

LAB - 7 (Part - A)

Page No.
Date

* Apriori Algorithm

(1)

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

Step-1 Item Set

Itemset	min Sup
{1}	2
{2}	3
{3}	3
{4}	1 ✗
{5}	3

Step 2 Minimum Count less than

ItemSet	minSup
{1}	2
{2}	3
{3}	3
{4}	3

Page No.
Date

Step 3 Itemset pairing

Itemset
{1,2}
{1 3}
{1 5}
{2,3}
{2, 5}
{3, 5}

Step 4 min Set Compare Value

Itemset	minSup
~~{1 2 3}~~	~~1~~ x
~~{1 3 5}~~	~~1~~ x
{2 3 5}	2

Step 5 Final Table Parity

Itemset	minSup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

Itemset	minSup
{1 2}	1 x
{1 3}	2
{1 5}	1 x
{2 3}	2
{2 5}	3
{3 5}	2

Step 6 final Table Answer Comparing and Condition

Itemset	minSup
{1 2 3}	1 x
{1 3 5}	1 x
{2 3 5}	2

Step2 Rules Generation

Association Rule	Support	Confidence	Confidence (1.)
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%

(2)

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

Step1 Total ItemSet Count

ItemSet	min Sup	ItemSet	minSup
B = {1,5}	3	B = {1,5}	3
M = {2,3,4,5}	4	→ M = {2,3,4}	4
D = {2,3,4}	4	D = {2,3,4}	3
Be = {2,4,5}	3	Be = {2,4,5}	3
C = {3,4,5}	3	C = {3,4,5}	3
E = {2}	1		

Page No.
Date

Step2 1, 2, 3, 4, 5

ItemSet	Items
{1 2}	1, 2
{1 3}	1, 3, 4, 6
{1 4}	2, 3, 4, 5
{1 5}	2, 3, 4, 5
{2, 3}	1, 2, 3, 5
{2 4}	
{2 5}	
{3 4}	
{3 5}	
{4 5}	

Step3 Compare of Two pair of Set

ItemSet	minSup
{1 2}	2 x
{1 3}	2 x
{1 4}	1 x
{1, 5}	1 x
{2 3}	3
{2 4}	3
{3 4}	3
{3 5}	3
{4 5}	2 x

→

Item Set	minSup
{2 3}	3
{2 4}	3
{3 4}	3
{3 5}	3

Page No.
Date

Step 4

ItemSet	MinSup
{2,3,4}	2 x
{2,3,5}	3
{3,4,5}	2 x

Step 5

Rules Generation

Association Rule	Support	Confidence	Confidence (%)
2^3 → 5	3	3/3 = 1	100%
3^5 → 2	3	3/3 = 1	100%
2^5 → 3	3	3/3 = 1	100%
2 → 3^5	3	3/4 = 0.75	75%
3 → 2^5	3	3/4 = 0.75	75%
5 → 2^3	3	3/3 = 1	100%

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Roll No - 130

Enrollment No - 24010101694

Lab - 7 (Part 2)

Karan Sonagara

Step 1: Load the Dataset

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

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib as plt
df=pd.read_csv("Tdata.csv")
df
```

Out[4]:

	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
5	T6	0	0	1	1	1	0

Step 2: Drop the 'Transaction' Column

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

In [7]:
```
df = df.drop("Transaction",axis=1)
df
```

Out[7]:

	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 [40]:
```
df.sum()
```

Out[40]:
```
Transaction    T1T2T3T4T5T6
bread                     5
butter                    3
coffee                    2
eggs                      2
jam                       2
milk                      3
dtype: object
```

Step 4: Define Apriori Function

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

In [43]:
```
from itertools import combinations

def findf(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)
            support = mask.sum() / n
            if support >= min_support:
                result.append((frozenset(items),round(support,2)))

    return result


finals = findf(df, min_support=0.5)

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

```
{'Transaction'} -> support: 1.0
{'bread'} -> support: 0.83
{'butter'} -> support: 0.5
{'milk'} -> support: 0.5
{'Transaction', 'bread'} -> support: 0.83
{'butter', 'Transaction'} -> support: 0.5
{'Transaction', 'milk'} -> support: 0.5
{'butter', 'bread'} -> support: 0.5
{'milk', 'bread'} -> support: 0.5
{'butter', 'Transaction', 'bread'} -> support: 0.5
{'Transaction', 'milk', 'bread'} -> support: 0.5
```

Step 5: Run Apriori

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

In [52]:
```
finals = findf(df, min_support=0.6)

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

```
{'Transaction'} -> support: 1.0
{'bread'} -> support: 0.83
{'Transaction', 'bread'} -> support: 0.83
```

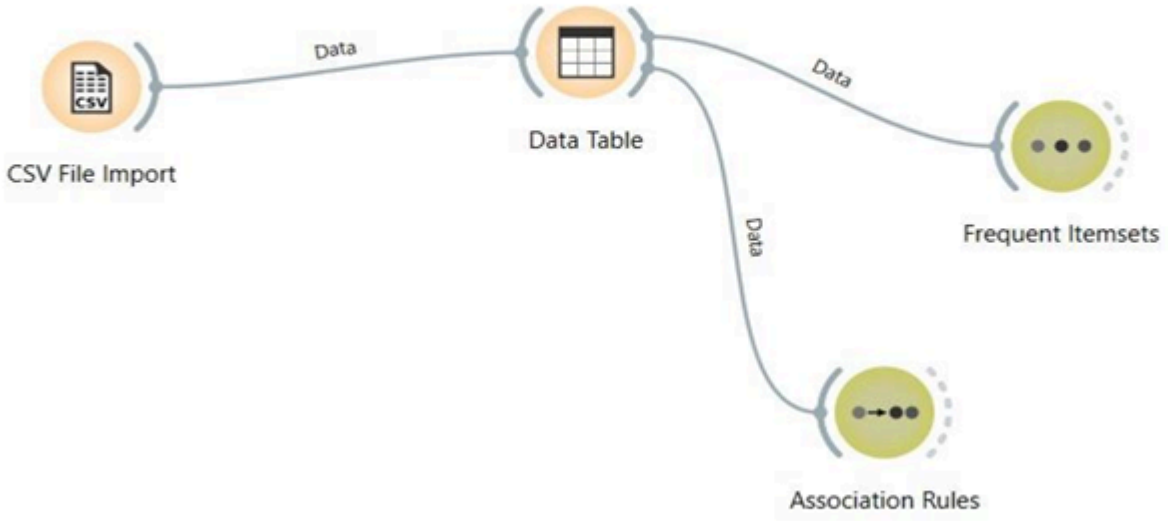
Step 6 Display as a DataFrame

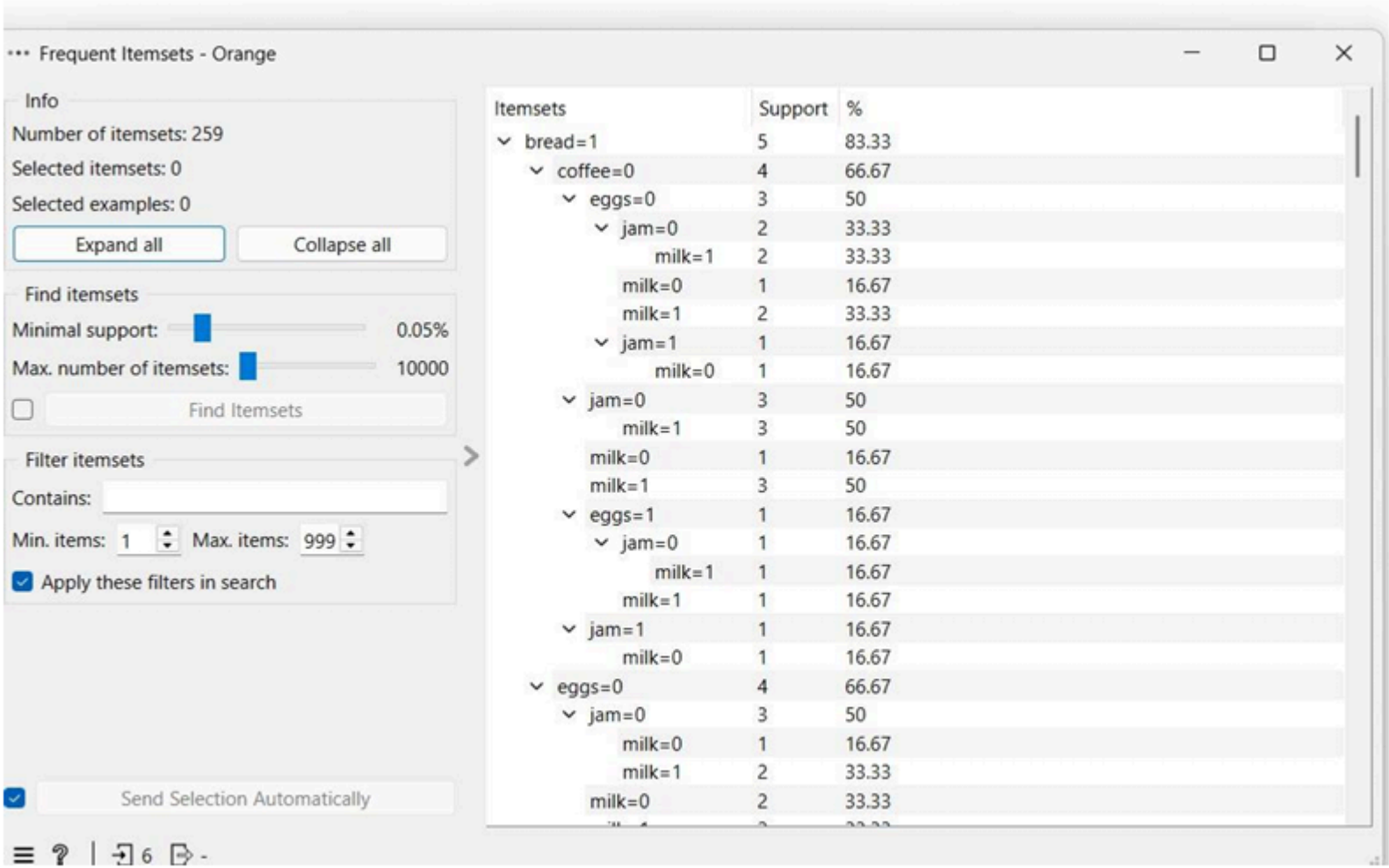
In [50]:
```
resulr = pd.DataFrame(finals,columns=["Itemset","Support"])
resulr
```

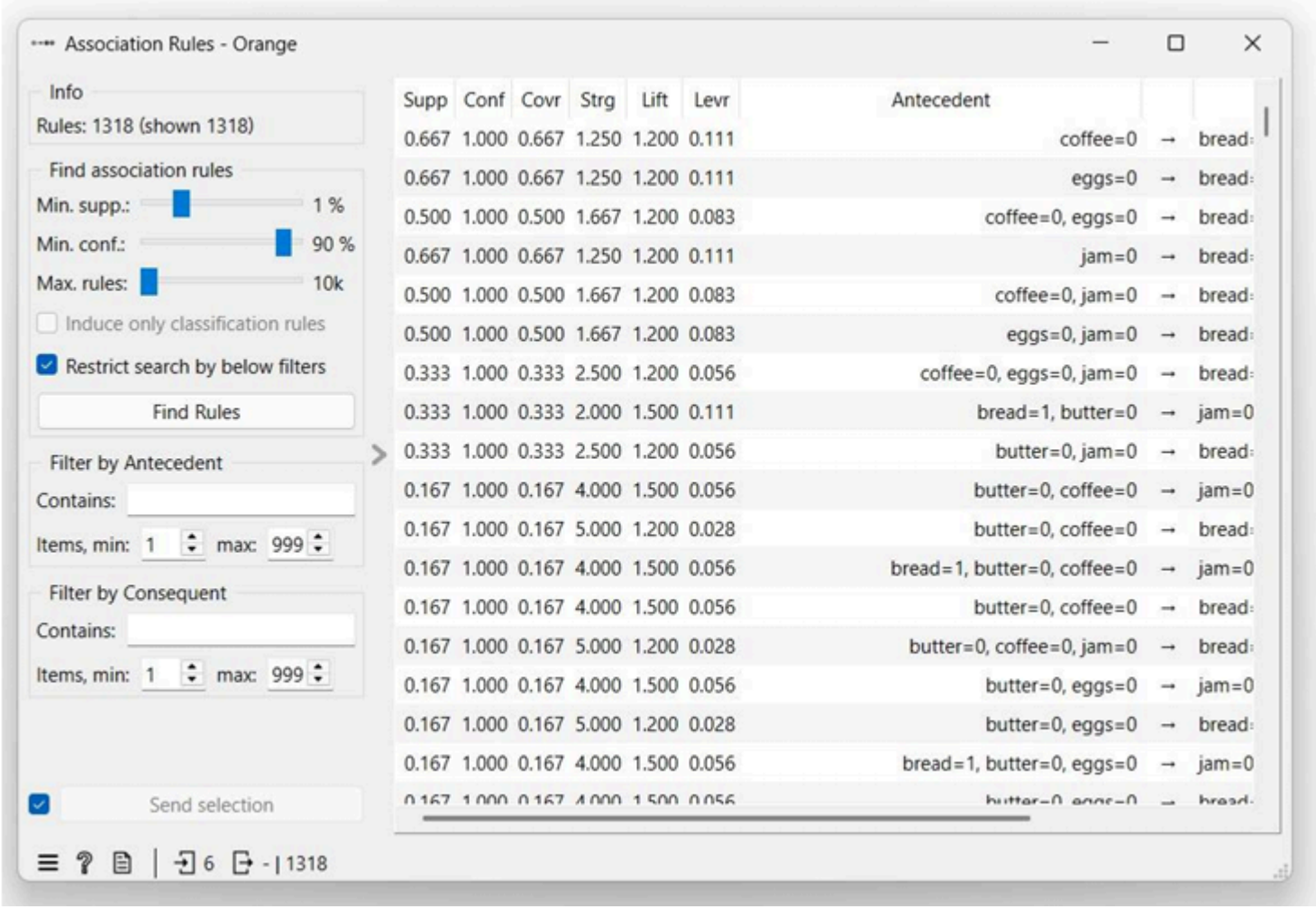
Out[50]:

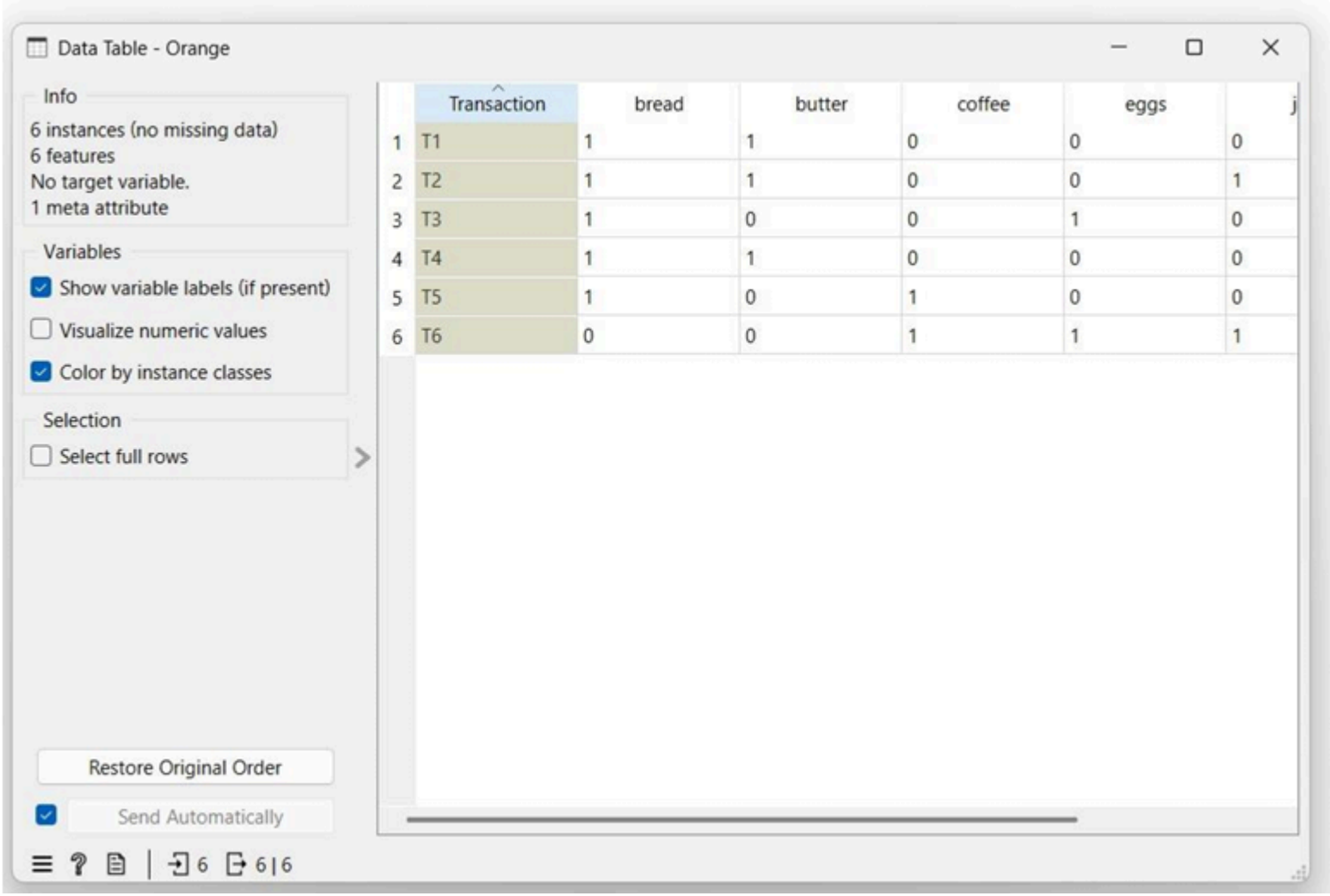
	Itemset	Support
0	(Transaction)	1.00
1	(bread)	0.83
2	(butter)	0.50
3	(milk)	0.50
4	(Transaction, bread)	0.83
5	(butter, Transaction)	0.50
6	(Transaction, milk)	0.50
7	(butter, bread)	0.50
8	(milk, bread)	0.50
9	(butter, Transaction, bread)	0.50
10	(Transaction, milk, bread)	0.50

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


Data
Data
Data
CSV File Import
Data Table
Frequent Itemsets
Association Rules


Frequent Itemsets - Orange
Info
Number of itemsets: 259
Selected itemsets: 0
Selected examples: 0
Expand all
Collapse all
Find itemsets
Minimal support: 0.05%
Max. number of itemsets: 10000
Find Itemsets
Filter itemsets
Contains:
Min. items: 1
Max. items: 999
Apply these filters in search
Send Selection Automatically
6
Itemsets Support %
bread=1 5 83.33
coffee=0 4 66.67
eggs=0 3 50
jam=0 2 33.33
milk=1 2 33.33
milk=0 1 16.67
milk=1 2 33.33
jam=1 1 16.67
milk=0 1 16.67
jam=0 3 50
milk=1 3 50
milk=0 1 16.67
milk=1 3 50
eggs=1 1 16.67
jam=0 1 16.67
milk=1 1 16.67
milk=1 1 16.67
jam=1 1 16.67
milk=0 1 16.67
eggs=0 4 66.67
jam=0 3 50
milk=0 1 16.67
milk=1 2 33.33
milk=0 2 33.33


Association Rules - Orange
Info
Rules: 1318 (shown 1318)
Find association rules
Min. supp.: 1 %
Min. conf.: 90 %
Max. rules: 10k
Induce only classification rules
Restrict search by below filters
Find Rules
Filter by Antecedent
Contains:
Items, min: 1 max: 999
Filter by Consequent
Contains:
Items, min: 1 max: 999
Send selection
6 - | 1318
Supp Conf Covr Strg Lift Levr Antecedent
0.667 1.000 0.667 1.250 1.200 0.111 coffee=0 → bread
0.667 1.000 0.667 1.250 1.200 0.111 eggs=0 → bread
0.500 1.000 0.500 1.667 1.200 0.083 coffee=0, eggs=0 → bread
0.667 1.000 0.667 1.250 1.200 0.111 jam=0 → bread
0.500 1.000 0.500 1.667 1.200 0.083 coffee=0, jam=0 → bread
0.500 1.000 0.500 1.667 1.200 0.083 eggs=0, jam=0 → bread
0.333 1.000 0.333 2.500 1.200 0.056 coffee=0, eggs=0, jam=0 → bread
0.333 1.000 0.333 2.000 1.500 0.111 bread=1, butter=0 → jam=0
0.333 1.000 0.333 2.500 1.200 0.056 butter=0, jam=0 → bread
0.167 1.000 0.167 4.000 1.500 0.056 butter=0, coffee=0 → jam=0
0.167 1.000 0.167 5.000 1.200 0.028 butter=0, coffee=0 → bread
0.167 1.000 0.167 4.000 1.500 0.056 bread=1, butter=0, coffee=0 → jam=0
0.167 1.000 0.167 4.000 1.500 0.056 butter=0, coffee=0 → bread
0.167 1.000 0.167 5.000 1.200 0.028 butter=0, coffee=0, jam=0 → bread
0.167 1.000 0.167 4.000 1.500 0.056 butter=0, eggs=0 → jam=0
0.167 1.000 0.167 5.000 1.200 0.028 butter=0, eggs=0 → bread
0.167 1.000 0.167 4.000 1.500 0.056 bread=1, butter=0, eggs=0 → jam=0



	Transaction	bread	butter	coffee	eggs	j
1	T1	1	1	0	0	0
2	T2	1	1	0	0	1
3	T3	1	0	0	1	0
4	T4	1	1	0	0	0
5	T5	1	0	1	0	0
6	T6	0	0	1	1	1

130 2401010169

LAB-8

(A) Generate the frequent pattern Tree from given dataset min_sup ≥ 3.

(1)

TID	Items
1	E K M N O Y
2	O E K N O Y
3	A E K M
4	C K M O Y
5	C E I K O

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

Step -1

item	frequent	
A	1	x
C	2	x
D	1	x
E	4	✓
I	1	x
K	5	✓
M	3	✓
N	2	x
O	3	✓
U	1	x
Y	3	✓

Step - 2

TID	Sorted Items
1	K E M O Y
2	K E O Y
3	K E M
4	K M Y
5	K E O

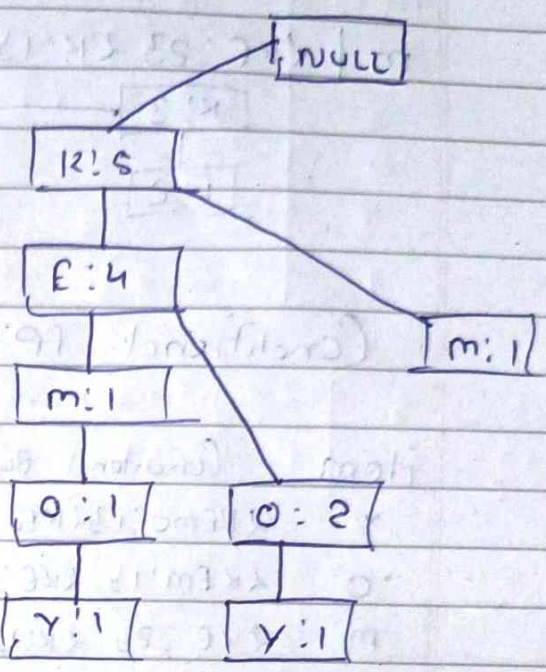
→ K E m o y

K E O y

K E M

K m y

K E O

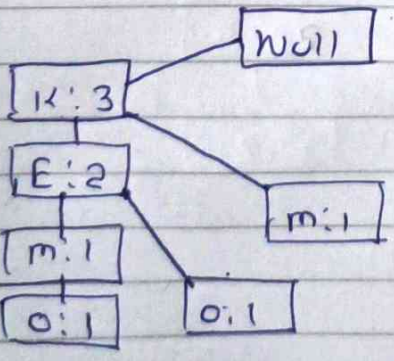


→

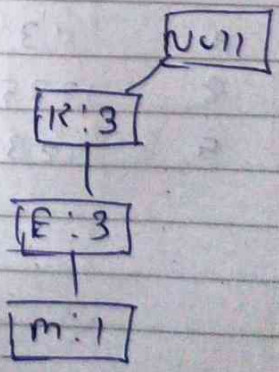
item	Conditional	Pattern	Base
Y	{KEMO:1}	{KEO:1}	{KM:1}
O	{KEM:1}	{KE:2}	
m	{KE:2}	{K:1}	
E	{K:4}		
K	—		

→

Y	{KEMO:1} {KEO:1} {KM:1}	{K:3}
O	{KEM:1} {KE:1} {KE:2}	{K:3}{E:2}

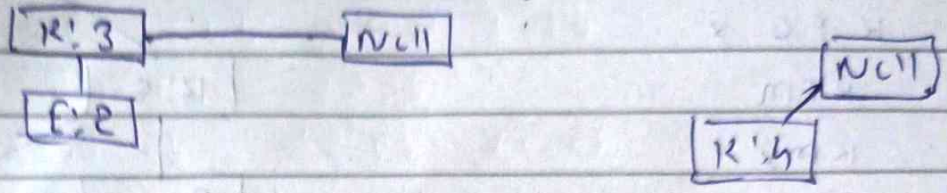


X



X

m | ⟨KE:2⟩ ⟨K:1⟩ | ⟨K:3⟩ | E | ⟨K:4⟩ | ⟨K:4⟩



Conditional FP-Tree and freq Pattern

item	Conditional Base Pattern	FP-Tree	freq - Path Generated
Y	⟨KEMO:1⟩⟨KEO:1⟩⟨KM:1⟩	⟨K:3⟩	⟨K,Y:3⟩
O	⟨KEM:1⟩ ⟨KE:2⟩	⟨K:3⟩ ⟨E:2⟩	⟨K,O:3⟩⟨E,O:3⟩
M	⟨KE:2⟩ ⟨K:1⟩	⟨K:3⟩	⟨K,M:3⟩
E	⟨K:4⟩	⟨K:4⟩	⟨K,E:4⟩
K	—	—	—

(2)

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

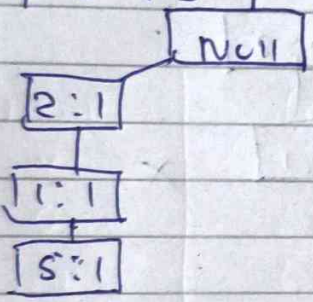
Step - 1

1	6
2	7
3	6
4	2
5	2

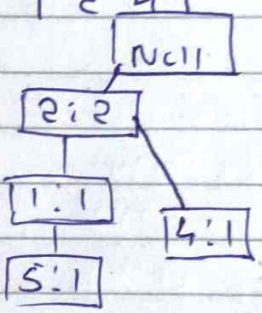
Step-2

TID	Items
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 5

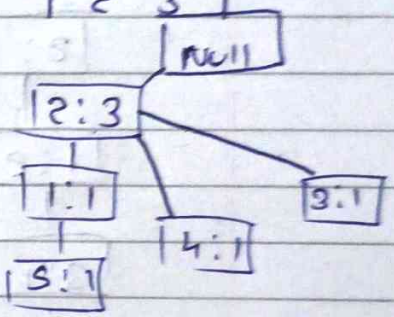
→ | 2 1 5 |



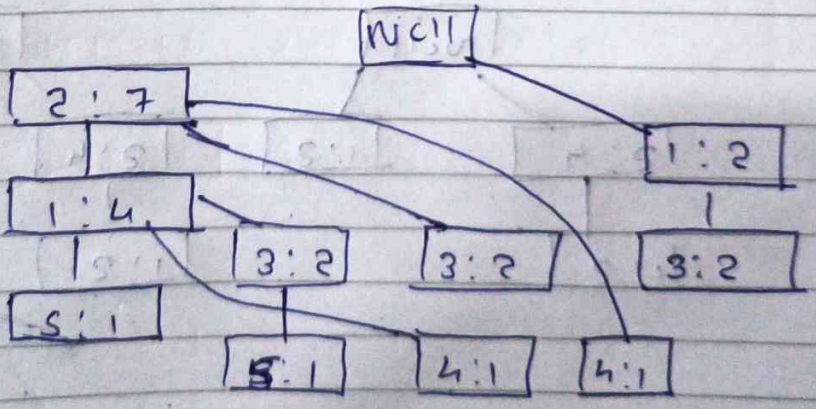
| 2 4 |



| 2 3 |

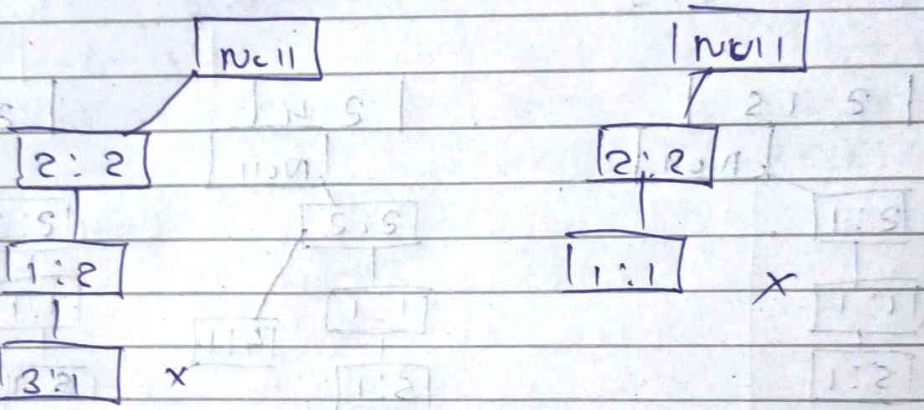


→ | 2 1 4 and all . . . |



Item	Conditional Pattern Base
S	{2,1:1} {2,1,3:1}
4	{2,1:1} {2:1}
3	{2,1:2} {2:2} {1:2}
1	{2:7}

(1)	S	{2,1:1} {2,1,3:1}	{2:2} {1:2}
(2)	4	{2,1:1} {2:1}	{2:2}



(3)	3	{2,1:2} {2:2} {1:2}	{2:4} {1:2}
(4)	1	{2:2} {3}	{2:1}