Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Data Mining

Lab - 1

Raj Vekariya | 23010101298

Introduction to Pandas Library Function:

Step-1 Import the pandas Libraries

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

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

In [ ]:
```
```

Step-3 Read csv or excel File

In [3]:
```
df=pd.read_csv("titanic.csv")
df
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7

891 rows × 12 columns

Step-4 Print Data from csv or excel File

In [ ]:

Step-5 See the First 10 Rows

In [5]:
```
df.head(10)
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250(
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283:
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925(
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100(
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050(
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.458:
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.862:
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.075(
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.133:
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.070:

Step-6 See the Last 10 Rows

In [4]:
```
df.tail(10)
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
881	882	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	7552	10.
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.

Step-7 Data type of each columns

In [14]:
```
df.dtypes
```

Out[14]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

Step-8 Display Summary Information

In [6]:
```
df.describe()
```

Out[6]:

	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

Step-9 Access a specific column

In [18]:
```
df['Pclass']
```

Out[18]:
```
0      3
1      1
2      3
3      1
4      3
      ..
886    2
887    1
888    3
889    1
890    3
Name: Pclass, Length: 891, dtype: int64
```

In [31]:
```
df.shape
```

Out[31]:
```
(891, 12)
```

In [33]:
```
df.shape[0]
```

Out[33]:
```
891
```

In [35]:
```
df.shape[1]
```

Out[35]:
```
12
```

Step-10 Access rows by their integer location

In [37]:
```
df.iloc[0]
```

Out[37]:
```
PassengerId                     1
Survived                        0
Pclass                          3
Name           Braund, Mr. Owen Harris
Sex                            male
Age                            22.0
SibSp                             1
Parch                             0
Ticket                    A/5 21171
Fare                           7.25
Cabin                           NaN
Embarked                          S
Name: 0, dtype: object
```

Step-11 Delete a specific Column

In [7]:
```
df.drop("Embarked",axis="columns",inplace=True)
```

In [8]:
```
df
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7

891 rows × 11 columns

Step-12 Create a new Column

In [45]:
```
df["isCabin"]= ~ df["Cabin"].isnull()
```

In [47]:
```
df
```

Out[47]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7

891 rows × 12 columns

Step-13 Perform Condition Selection on DataFrame

In [78]:
```
df[df["Pclass"] ==3 ]
```

Out[78]:

	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	7.
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.
...	...	...	...	...	...	...	...	...	...	
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	7552	10.
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.

491 rows × 12 columns

Step-14 Compute the sum of value

In [52]:
```
df["Fare"].sum()
```

Out[52]: 28693.9493

Step-15 Compute the mean of value

In [56]:
```
df["Fare"].mean()
```

Out[56]: 32.204207968574636

Step-16 Count non-null value (column)

In [74]:
```
(~df.isnull()).sum()
```

Out[74]:
```
PassengerId    891
Survived       891
Pclass         891
Name           891
Sex            891
Age            714
SibSp          891
Parch          891
Ticket         891
Fare           891
Cabin          204
isCabin        891
dtype: int64
```

In [72]:
```
df.count()
```

Out[72]:
```
PassengerId    891
Survived       891
Pclass         891
Name           891
Sex            891
Age            714
SibSp          891
Parch          891
Ticket         891
Fare           891
Cabin          204
isCabin        891
dtype: int64
```

Step-17 Find Minimun or Maximum values

In [66]:
```
df["Fare"].min()
```

Out[66]: 0.0

In [64]:
```
df["Fare"].max()
```

Out[64]: 512.3292

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Data Mining

Lab - 2

Raj Vekariya | 23010101298

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 [6]:
```
a=np.arange(11)
print(a)
print(type(a))
```

```
[ 0  1  2  3  4  5  6  7  8  9 10]
<class 'numpy.ndarray'>
```

In [9]:
```
a=np.arange(2,9)
a
```

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

Step 3. Reshape 1D to 2D Array

In [13]:
```
a=np.arange(12).reshape(3,4)
a
```

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

Step 4. Create a Linspace array

In [14]:
```
np.linspace(0,5,20)
```

Out[14]:
```
array([0.        , 0.26315789, 0.52631579, 0.78947368, 1.05263158,
       1.31578947, 1.57894737, 1.84210526, 2.10526316, 2.36842105,
       2.63157895, 2.89473684, 3.15789474, 3.42105263, 3.68421053,
       3.94736842, 4.21052632, 4.47368421, 4.73684211, 5.        ])
```

Step 5. Create a Random Numbered Array

In [15]:
```
np.random.rand(2)
```

Out[15]:
```
array([0.7406834 , 0.56917623])
```

In [16]:
```
np.random.rand(2,4)
```

Out[16]:
```
array([[0.56916381, 0.52930315, 0.95050776, 0.39259408],
       [0.82080692, 0.18362901, 0.88810763, 0.47867718]])
```

Step 6. Create a Random Integer Array

In [21]:
```
np.random.randint(1,100,size=10)
```

Out[21]:
```
array([94,  4, 99, 55, 51,  3, 49, 89, 26, 14])
```

In [23]:
```
np.random.randint(1,100,size=(2,4))
```

Out[23]:
```
array([[39, 27, 96, 78],
       [19, 66, 22, 47]])
```

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

In [25]:
```
arr=np.random.randint(1,100,size=10)
arr
```

Out[25]:
```
array([ 2,  6, 14, 88, 39, 59, 46, 81, 16, 31])
```

In [26]:
```
arr.min()
```

Out[26]:
```
2
```

In [27]:
```
arr.max()
```

Out[27]:
```
88
```

In [28]:
```
arr.argmax()
```

Out[28]:
```
3
```

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

Out[29]:
```
0
```

Step 8. Indexing in 1D Array

In [30]:
```
arr[8]
```

Out[30]:
```
16
```

In [31]:
```
arr[1:5]
```

Out[31]:
```
array([ 6, 14, 88, 39])
```

Step 9. Indexing in 2D Array

In [33]:
```
arr2d=arr.reshape(2,5)
arr2d
```

Out[33]:
```
array([[ 2,  6, 14, 88, 39],
       [59, 46, 81, 16, 31]])
```

In [34]:
```
arr2d[0]
```

Out[34]:
```
array([ 2,  6, 14, 88, 39])
```

In [40]:
```
arr2d[0][1]
arr2d[0,1]
```

Out[40]:
```
6
```

In [41]:
```
arr2d[:1,2:]
```

Out[41]:
```
array([[14, 88, 39]])
```

In [53]:
```
n=np.arange(15).reshape(3,5)
n
```

Out[53]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

In [56]:
```
n[1:,2:]
```

Out[56]:
```
array([[ 7,  8,  9],
       [12, 13, 14]])
```

Step 10. Conditional Selection

In [43]:
```
arr[arr>4]
```

Out[43]:
```
array([ 6, 14, 88, 39, 59, 46, 81, 16, 31])
```

In [44]:
```
arr2d[arr2d>2]
```

Out[44]:
```
array([ 6, 14, 88, 39, 59, 46, 81, 16, 31])
```

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

Pandas

Step 1. Import the necessary libraries

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

Step 2. Import the dataset from this address.

In [3]:
```
df=pd.read_csv("user.csv",sep="|")
df
```

Out[3]:

	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 × 5 columns

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

In [4]:
```
df.set_index("user_id")
```

Out[4]:

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 [65]:
```
df.head(25)
```

Out[65]:

	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
10	11	39	F	other	30329
11	12	28	F	other	06405
12	13	47	M	educator	29206
13	14	45	M	scientist	55106
14	15	49	F	educator	97301
15	16	21	M	entertainment	10309
16	17	30	M	programmer	06355
17	18	35	F	other	37212
18	19	40	M	librarian	02138
19	20	42	F	homemaker	95660
20	21	26	M	writer	30068
21	22	25	M	writer	40206
22	23	30	F	artist	48197
23	24	21	F	artist	94533
24	25	39	M	engineer	55107

Step 5. See the last 10 entries

In [66]:
```
df.tail(10)
```

Out[66]:

	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 [76]:
```
df.shape[0]
```

Out[76]:
```
943
```

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

In [71]:
```
df.shape[1]
```

Out[71]:
```
5
```

Step 8. Print the name of all the columns.

In [90]:
```
df.columns
```

Out[90]:
```
Index(['user_id', 'age', 'gender', 'occupation', 'zip_code'], dtype='object')
```

Step 9. How is the dataset indexed?

In [8]:
```
df.index
```

Out[8]:
```
RangeIndex(start=0, stop=943, step=1)
```

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 943 entries, 0 to 942
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   user_id     943 non-null    int64 
 1   age         943 non-null    int64 
 2   gender      943 non-null    object
 3   occupation  943 non-null    object
 4   zip_code    943 non-null    object
dtypes: int64(2), object(3)
memory usage: 37.0+ KB
```

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

In [9]:
```
df.dtypes
```

Out[9]:
```
user_id        int64
age            int64
gender        object
occupation    object
zip_code      object
dtype: object
```

Step 11. Print only the occupation column

In [10]:
```
df['occupation']
```

Out[10]:
```
0         technician
1              other
2             writer
3         technician
4              other
            ...     
938          student
939    administrator
940          student
941        librarian
942          student
Name: occupation, Length: 943, dtype: object
```

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

In [15]:
```
len(set(df['occupation']))
```

Out[15]:
```
21
```

Step 13. What is the most frequent occupation?

In [16]:
```
df['occupation'].mode()
```

Out[16]:
```
0    student
Name: occupation, dtype: object
```

In [17]:
```
df['occupation'].value_counts().idxmax()
```

Out[17]:
```
'student'
```

Step 14. Summarize the DataFrame.

In [12]:
```
df.describe()
```

Out[12]:

	user_id	age
count	943.000000	943.000000
mean	472.000000	34.051962
std	272.364951	12.192740
min	1.000000	7.000000
25%	236.500000	25.000000
50%	472.000000	31.000000
75%	707.500000	43.000000
max	943.000000	73.000000

Step 15. Summarize all the columns

In [27]:
```
df.describe(include='all')
```

Out[27]:

	user_id	age	gender	occupation	zip_code
count	943.000000	943.000000	943	943	943
unique	NaN	NaN	2	21	795
top	NaN	NaN	M	student	55414
freq	NaN	NaN	670	196	9
mean	472.000000	34.051962	NaN	NaN	NaN
std	272.364951	12.192740	NaN	NaN	NaN
min	1.000000	7.000000	NaN	NaN	NaN
25%	236.500000	25.000000	NaN	NaN	NaN
50%	472.000000	31.000000	NaN	NaN	NaN
75%	707.500000	43.000000	NaN	NaN	NaN
max	943.000000	73.000000	NaN	NaN	NaN

Step 16. Summarize only the occupation column

In [19]:
```
df['occupation'].describe()
```

Out[19]:
```
count         943
unique         21
top       student
freq          196
Name: occupation, dtype: object
```

Step 17. What is the mean age of users?

In [20]:
```
df['age'].mean()
```

Out[20]:
```
34.05196182396607
```

Step 18. What is the age with least occurrence?

In [26]:
```
df['age'].value_counts().idxmin()
```

Out[26]:
```
7
```

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

In [ ]:
```

```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Data Mining

Lab - 3

Raj Vekariya | 23010101298

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

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

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

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7

891 rows × 12 columns

In [6]:
```
df.head(5)
```

Out[6]:

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

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

In [8]:
```
nominal=['Name','Sex','Cabin','Embarked','Ticket']
ordinal=['Pclass']
binary=['Sex','Survived']
numeric=['Age','Fare','Sibsp','Parch']

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

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

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

In [11]:
```
print("Survived values(asymmetric binary)")
print(df['Survived'].value_counts())
print("---------------------------------")
```

```
print("Gender count(symmetric binary)")
print(df['Sex'].value_counts())
```

```
Survived values(asymmetric binary)
0    549
1    342
Name: Survived, dtype: int64
--------------------------------
Gender count(symmetric binary)
male      577
female    314
Name: Sex, dtype: int64
```

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

In [14]:
```
quantitative=['PassengerId','Survived','Pclass','Age','SibSp','Parch','Fare']

for i in quantitative:
    print(":::",i,":::")
    print("Mean",df[i].mean())
    print("Standard Deviation",df[i].std())
    print("Minimum",df[i].min())
    print("Maximum",df[i].max())
    print("Mode",df[i].mode()[0])
    print("Range",df[i].max()-df[i].min())
    print("--------------------------------")
```

```
::: PassengerId :::
Mean 446.0
Standard Deviation 257.3538420152301
Minimum 1
Maximum 891
Mode 1
Range 890
------------------------------
::: Survived :::
Mean 0.3838383838383838
Standard Deviation 0.4865924542648575
Minimum 0
Maximum 1
Mode 0
Range 1
------------------------------
::: Pclass :::
Mean 2.308641975308642
Standard Deviation 0.836071240977049
Minimum 1
Maximum 3
Mode 3
Range 2
------------------------------
::: Age :::
Mean 29.69911764705882
Standard Deviation 14.526497332334042
Minimum 0.42
Maximum 80.0
Mode 24.0
Range 79.58
------------------------------
::: SibSp :::
Mean 0.5230078563411896
Standard Deviation 1.1027434322934317
Minimum 0
Maximum 8
Mode 0
Range 8
------------------------------
::: Parch :::
Mean 0.38159371492704824
Standard Deviation 0.8060572211299483
Minimum 0
Maximum 6
Mode 0
Range 6
------------------------------
::: Fare :::
Mean 32.204207968574636
Standard Deviation 49.6934285971809
Minimum 0.0
Maximum 512.3292
Mode 8.05
Range 512.3292
------------------------------
```

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

In [19]:
```
print("Frequency of Pclass\n",df['Pclass'].value_counts())
```

```
Frequency of Pclass
 3    491
1    216
2    184
Name: Pclass, dtype: int64
```

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

In [23]:
```
print("Summary for numeric attributes:")
print(df.describe())

print("\nSummary for categorical attributes")
print(df.describe(include="object"))
```

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

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

Summary for categorical attributes
                           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
```

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

In [33]:
```
print("Covariance matrix:")
print(df.cov())
```

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

                Parch         Fare
PassengerId -0.342697   161.883369
Survived     0.032017     6.221787
Pclass       0.012429   -22.830196
Age         -2.344191    73.849030
SibSp        0.368739     8.748734
Parch        0.649728     8.661052
Fare         8.661052  2469.436846
```

In [34]:
```
print("Correlation matrix:")
print(df.corr())
```

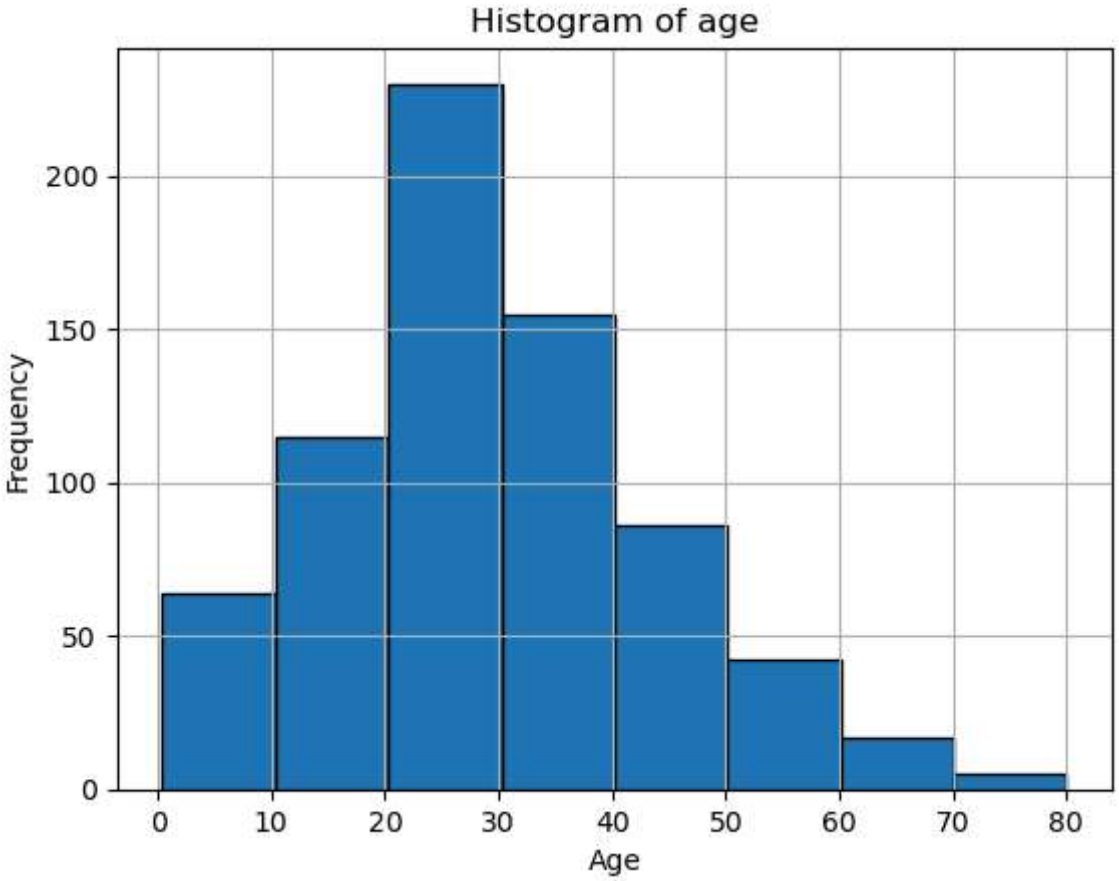
```
Correlation matrix:
             PassengerId  Survived    Pclass       Age     SibSp     Parch  \
PassengerId     1.000000 -0.005007 -0.035144  0.036847 -0.057527 -0.001652
Survived       -0.005007  1.000000 -0.338481 -0.077221 -0.035322  0.081629
Pclass         -0.035144 -0.338481  1.000000 -0.369226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

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

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

In [6]:
```
import matplotlib.pyplot as plt
```

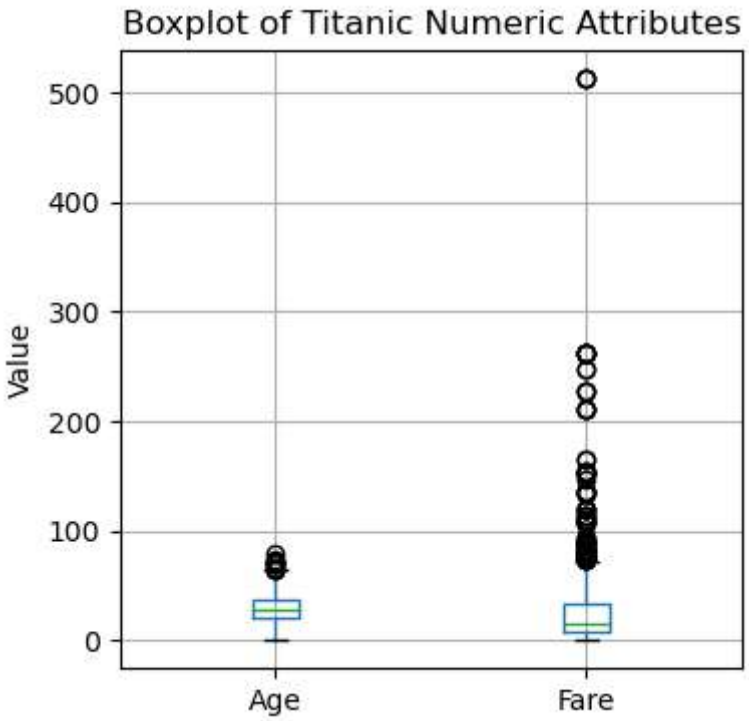
In [37]:
```
df['Age'].dropna().hist(bins=8,edgecolor='black')
plt.title("Histogram of age")
```

```
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



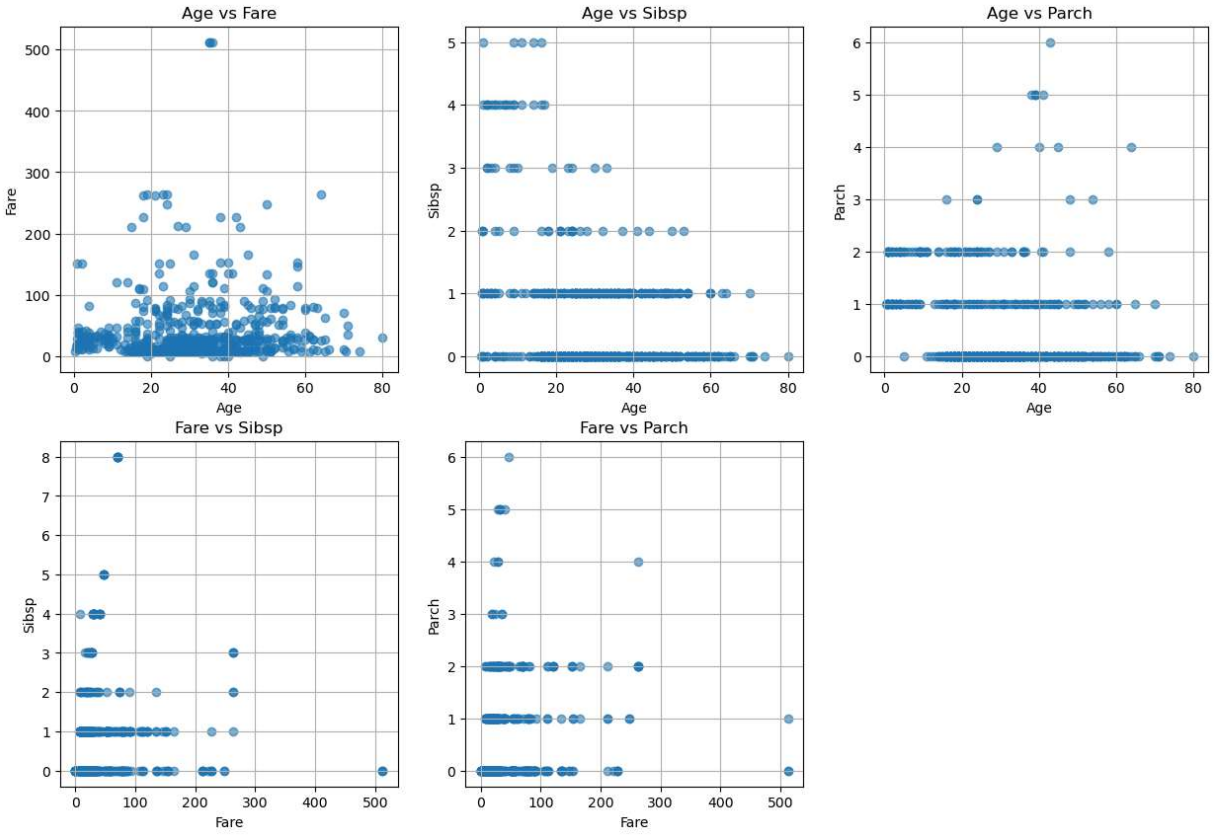
10) A boxplot can also be used to show the distribution of values for each attribute.

In [16]:
```
numeric = ['Age', 'Fare']
data = df[numeric].dropna()

plt.figure(figsize = (4,4))
data.boxplot(column = numeric)
plt.title("Boxplot of Titanic Numeric Attributes")
plt.ylabel("Value")
plt.grid(True)
plt.show()
```



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

In [15]:
```
pairs = [('Age', 'Fare'), ('Age', 'SibSp'), ('Age', 'Parch'), ('Fare', 'SibSp'), ('

plt.figure(figsize = (15, 10))

for i, (x, y) in enumerate(pairs):
    plt.subplot(2, 3, i + 1)
    plt.scatter(df[x], df[y], alpha=0.6)
    plt.xlabel(x.capitalize())
    plt.ylabel(y.capitalize())
    plt.title(f"{x.capitalize()} vs {y.capitalize()}")
    plt.grid(True)
```


Age vs Fare
Fare
Age
Age vs Sibsp
Sibsp
Age
Age vs Parch
Parch
Age
Fare vs Sibsp
Sibsp
Fare
Fare vs Parch
Parch
Fare

In [ ]:

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Data Mining

Lab - 4

Raj Vekariya | 23010101298

Step 1. Import the necessary libraries

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

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called chipo.

In [3]:
```
url="https://raw.githubusercontent.com/justmarkham/DAT8/master/data/chipotle.tsv"
chipo=pd.read_csv(url,sep='\t')
```

Step 4. See the first 10 entries

In [4]:
```
chipo.head(10)
```

Out[4]:

	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 [5]:
```
# Solution 1
chipo.shape[0]
```

Out[5]: 4622

In [6]:
```
# 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 [7]:
```
chipo.shape[1]
```

Out[7]: 5

Step 7. Print the name of all the columns.

In [8]:
```
chipo.columns
```

Out[8]:
```
Index(['order_id', 'quantity', 'item_name', 'choice_description',
       'item_price'],
      dtype='object')
```

Step 8. How is the dataset indexed?

In [9]:
```
chipo.index
```

Out[9]:
```
RangeIndex(start=0, stop=4622, step=1)
```

Step 9. Number of Unique Items ?

In [17]:
```
chipo['item_name'].nunique()
```

Out[17]: 50

Step 10. Which was the most-ordered item?

In [14]:
```
chipo.groupby('item_name').sum().sort_values(['quantity'],ascending=False).head(1)
```

Out[14]:

item_name	order_id	quantity
Chicken Bowl	713926	761

Step 11. How many items were orderd in total?

In [11]:
```
chipo['quantity'].sum()
```

Out[11]: 4972

Step 12. Turn the item price into a float

Step 12.a. Check the item price type

In [18]:
```
chipo['item_price'].dtypes
```

Out[18]:
```
dtype('O')
```

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

In [24]:
```
chipo.item_price=chipo.item_price.apply(lambda x:float(x[1:-1]))
```

Step 12.c. Check the item price type

In [25]:
```
chipo['item_price'].dtypes
```

Out[25]:
```
dtype('float64')
```

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

In [29]:
```
revenue=(chipo['item_price']*chipo['quantity']).sum()
print("Revenue was: $",revenue)
```

```
Revenue was: $ 39237.02
```

Step 15. How many orders were made ?

In [32]:
```
chipo['order_id'].value_counts().count()
```

Out[32]:
```
1834
```

Step 17. How many different choice descriptions are there?

In [33]:
```
chipo['choice_description'].nunique()
```

Out[33]:
```
1043
```

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

In [34]:
```
items=chipo.groupby('item_name')['quantity'].sum()
items[items>100]
```

Out[34]:
```
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 [39]:
```
# Solution 1
chipo['revenue']=chipo['item_price']*chipo['quantity']
order=chipo.groupby(by=['order_id']).sum()
order['revenue'].mean()
```

Out[39]:
```
21.39423118865867
```

In [41]:
```
# Solution 2
chipo.groupby(by=['order_id']).sum()['revenue'].mean()
```

Out[41]:
```
21.39423118865867
```

In [ ]:
```

```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Data Mining

Lab - 5

Raj Vekariya | 23010101298

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

In [1]:
```
import pandas as pd
```

In [27]:
```
df=pd.read_csv("titanic.csv")
df
```

Out[27]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7

891 rows × 12 columns

In [10]:
```
df.tail(5)
```

Out[10]:

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

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

In [ ]:
```
df.dropna(inplace=True)
df.dropna(how='all', axis=0) #if all the entries are NaN, drop the row
df.dropna(how='any', axis=0) # if any entry is NaN, drop the row
df
```

Out[ ]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.862
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.700
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.550
...	...	...	...	...	...	...	...	...	...	.
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.554
872	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	695	5.000
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.158
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.000
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.000

183 rows × 12 columns

In [40]:
```
# df.fillna(0, inplace=True)
# df
mean_data=df['Age'].mean()
mode_cabin=df['Cabin'].mode()
mc=mode_cabin[[0][0]]
mode_data=mc[0:3]
df.fillna({'Age': mean_data, 'Cabin': mode_data}, inplace=True)
df
```

Out[40]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450
...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	211536
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	112053
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./C. 6607
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	111369
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	370376

891 rows × 12 columns

In [19]:
```
df.interpolate(inplace=True)
df
```

Out[19]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fa
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.28
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.10
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	22.5	1	2	W./C. 6607	23.45
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75

891 rows × 12 columns

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

In [41]:
```
dataminmax=df.copy()
max_age=df['Age'].max()
min_age=df['Age'].min()
scale_age=(df['Age']-min_age)/(max_age-min_age)
dataminmax['MinMax']=scale_age
dataminmax
```

Out[41]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450
...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	211536
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	112053
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./C. 6607
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	111369
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	370376

891 rows × 13 columns

In [42]:
```
#deciaml scaling
datadecimal=df.copy()
scale_age_decimal=df['Age']/100
datadecimal['Decimal']=scale_age_decimal
datadecimal
```

Out[42]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450
...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	211536
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	112053
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./C. 6607
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	111369
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	370376

891 rows × 13 columns

In [43]:
```
data_zscore=df.copy()
mean_age=df['Age'].mean()
std_age=df['Age'].std()
zscore_age=(df['Age']-mean_age)/std_age
data_zscore['ZScore']=zscore_age
data_zscore
```

Out[43]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450
...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	211536
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	112053
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./C. 6607
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	111369
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	370376

891 rows × 13 columns

In [ ]:

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Data Mining

Lab - 6

Raj Vekariya | 23010101298

```
# 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.

Step 1: Load the Iris Dataset

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

Out[4]:

	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 [5]:
```
df.shape
```

Out[5]: (150, 5)

Step 2: Standardize the data (zero mean)

In [6]:
```
X=df.drop(columns='species')
Y=df['species'].map({'setosa': 0, 'virginica': 1, 'versicolor': 2})
print("Dimension of X:", X.shape)
```

```
Dimension of X: (150, 4)
```

In [7]:
```
X_meaned=X-np.mean(X,axis=0)
print("Data after centering(first 5 rows):\n",X_meaned[:5])
print(X_meaned.shape)
```

```
Data after centering(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
(150, 4)
```

Step 3: Compute the Covariance Matrix

In [8]:
```
cov_mat=np.cov(X_meaned, rowvar=False)
print("Covariance matrix shape:", cov_mat.shape)
print("Covariance matrix:\n", cov_mat)
```

```
Covariance matrix shape: (4, 4)
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 [9]:
```
eigen_values, eigen_vectors = np.linalg.eig(cov_mat)
print("Eigenvalues:\n", eigen_values)
print("Eigenvectors:\n", eigen_vectors)
```

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

Step 5: Compute eigenvalues and eigenvectors

In [11]:
```
sorted_index=np.argsort(eigen_values)[::-1]
sorted_eigenvalues = eigen_values[sorted_index]
sorted_eigenvectors=eigen_vectors[:, sorted_index]
print("Sort Index:", sorted_index)
print("Sorted Eigenvalues:\n", sorted_eigenvalues)
print("Sorted Eigenvectors:\n", sorted_eigenvectors)
```

```
Sort Index: [0 1 2 3]
Sorted Eigenvalues:
 [4.22824171 0.24267075 0.0782095  0.02383509]
Sorted Eigenvectors:
 [[ 0.36138659 -0.65658877 -0.58202985  0.31548719]
 [-0.08452251 -0.73016143  0.59791083 -0.3197231 ]
 [ 0.85667061  0.17337266  0.07623608 -0.47983899]
 [ 0.3582892   0.07548102  0.54583143  0.75365743]]
```

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

In [12]:
```
k=2
eigenvector_subset=sorted_eigenvectors[:,0:k]
print(eigenvector_subset.shape)
```

```
(4, 2)
```

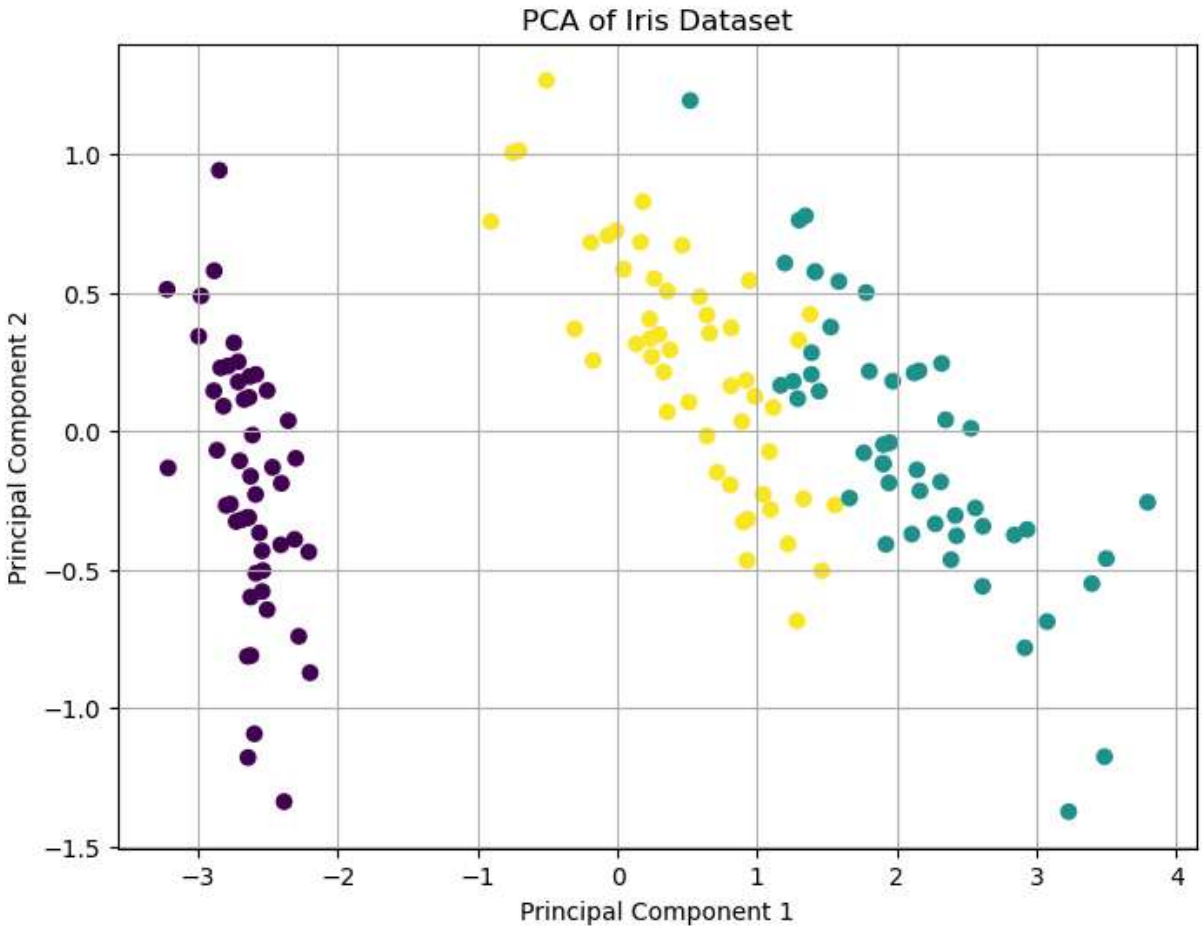
Step 7: Project the data onto the top k eigenvectors

In [13]:
```
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 [14]:
```
plt.figure(figsize=(8, 6))
plt.scatter(X_reduced[:,0], X_reduced[:,1], c=Y, cmap='viridis')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of 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 [ ]:

Lab : 7
Date : 14/6/25

Unit - 3

Apriori Algorithm :

Support : $\frac{x\ UB}{|T|}$ (Total item)

Confidence : $C = \frac{A \cup B}{A}$

Que :

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

minimum support = 2

↓ scan D

C_1

Item set	Min. support	
{1}	2	
{2}	3	
{3}	3	
{4}	1	×
{5}	3	

(1)

(2) L_1 → 2 >=

IS	M.S
{1}	2
{2}	3
{3}	3
{5}	3

scan D →

C_2

Itemset	Min. sup	
{1,2}	1	×
{1,3}	2	
{1,5}	1	×
{2,3}	2	
{2,5}	3	
{3,5}	2	

(3)

(4) L_2 →

IS	M.S.
{1,3}	2
{2,3}	2
{2,5}	3
{3,5}	2

scan D →

↓

(3 nij pair one ej lesi)

Item set	Min sup	
{1, 2, 3}	1	×
{1, 3, 5}	1	×
{2, 3, 5}	2	

⇒ Rules Generation

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

$\frac{A \cup B}{A} = \frac{2^3 \wedge 5}{5} = \frac{2}{3}$ (support count) $= 0.66$

$\frac{A \rightarrow B\ (A \cup B)}{A} = \frac{2^3 \wedge 5}{2^3} = \frac{2}{2} = 1$

(1) $2^3 \rightarrow 5 = \frac{A \rightarrow B}{A} = \frac{2^3 \wedge 5}{2^3} = \frac{2}{2} = 1$

(2) $3^5 \rightarrow 2 = \frac{A \rightarrow B}{A} = \frac{3^5 \wedge 2}{3^5} = \frac{2}{2} = 1$

(3) $2^5 \rightarrow 3 = \frac{A \rightarrow B}{A} = \frac{2^5 \wedge 3}{2^5} = \frac{2}{3} = 0.66$

(4) $2 \rightarrow 3^5 = \frac{A \rightarrow B}{A} = \frac{2^3 \wedge 5}{2} = \frac{2}{3} = 0.66$

vision

(5) $3 \rightarrow 2\wedge 5 = \frac{A \rightarrow B}{A} = \frac{3\wedge 2\wedge 5}{3} = \frac{2}{3} = 0.66$

(6) $5 \rightarrow 2\wedge 3 = \frac{A \rightarrow B}{A} = \frac{5\wedge 2\wedge 3}{5} = \frac{2}{3} = 0.66$

Que:

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

C_1

Item set	Min Support
{Br}	3
{M}	4
{Be}	3
{E}	1 ×
{D}	4
{C}	3

$L_1 \longrightarrow$

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

C_2

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

vision

L_2

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

C_3

IS	M.S
{Br, M, O}	1
{Br, M, Be}	0
{Br, M, C}	1
{Br, Be, O}	1
{Br, O, C}	1
{M, Be, O}	2
{M, Be, C}	2
{M, O, C}	~~3~~
{Be, O, C}	2

$\xrightarrow{L_3}$

IS	MS
{M, Be, O}	2
{M, Be, C}	2
{M, O, C}	3
{Be, O, C}	2

C_4

IS	MS
{M, Be, O, C}	2

?

⇒

Rules Generation	support	confidence	%
C ^ BE ^ D → M		2/2 = 1	100 %
M ^ BE ^ D → C		2/2 = 1	100 %
M ^ C ^ D → BE		2/3 = 0.66	66 %
M ^ BE ^ C → D		2/2 = ~~0.66~~ 1	100 %
M ^ BE → D ^ C		2/2 = 1	100 %
M ^ D → BE ^ C		2/3 = 0.66	66 %
M ^ C → D ^ BE		2/3 = 0.66	66 %
M → C ^ BE ^ D		2/4 = 0.5	50 %
C → M ^ BE ^ D		2/3 = 0.66	66 %
BE → M ^ C ^ D		2/3 = 0.66	66 %
D → M ^ BE		2/4 = 0.5	50 %
D ^ C → M ^ BE		2/3 = 0.66	66 %
BE ^ C → M ^ D		2/2 = 1	100 %
D ^ BE → M ^ C		2/3 = 0.66	66 %

vision

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Data Mining

Lab - 7 (Part 2)

Raj Vekariya | 23010101298

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
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(columns='Transaction')
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 [8]:
```
df.sum()
```

Out[8]:
```
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 [13]:
```
from itertools import combinations
def AprioriFun(df, minSupport):
    n=len(df)
    result=[]

    for i in [1,2,3]:
        for items in combinations(df.columns,i):
            mask=df[list(items)].all(axis=1)
            support=mask.sum()/n
            print(f"{frozenset(items)}->{round(support,2)}")
            if support>=minSupport:
                result.append((frozenset(items),round(support,2)))

    return result
```

Step 5: Run Apriori

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

In [15]:
```
frequent_itemsets=AprioriFun(df,minSupport=0.5)

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

```
frozenset({'bread'})->0.83
frozenset({'butter'})->0.5
frozenset({'coffee'})->0.33
frozenset({'eggs'})->0.33
frozenset({'jam'})->0.33
frozenset({'milk'})->0.5
frozenset({'butter', 'bread'})->0.5
frozenset({'coffee', 'bread'})->0.17
frozenset({'eggs', 'bread'})->0.17
frozenset({'jam', 'bread'})->0.17
frozenset({'milk', 'bread'})->0.5
frozenset({'butter', 'coffee'})->0.0
frozenset({'butter', 'eggs'})->0.0
frozenset({'butter', 'jam'})->0.17
frozenset({'milk', 'butter'})->0.33
frozenset({'eggs', 'coffee'})->0.17
frozenset({'coffee', 'jam'})->0.17
frozenset({'milk', 'coffee'})->0.0
frozenset({'eggs', 'jam'})->0.17
frozenset({'milk', 'eggs'})->0.17
frozenset({'milk', 'jam'})->0.0
frozenset({'butter', 'coffee', 'bread'})->0.0
frozenset({'butter', 'bread', 'eggs'})->0.0
frozenset({'butter', 'jam', 'bread'})->0.17
frozenset({'milk', 'butter', 'bread'})->0.33
frozenset({'eggs', 'coffee', 'bread'})->0.0
frozenset({'coffee', 'jam', 'bread'})->0.0
frozenset({'milk', 'coffee', 'bread'})->0.0
frozenset({'eggs', 'jam', 'bread'})->0.0
frozenset({'milk', 'eggs', 'bread'})->0.17
frozenset({'milk', 'jam', 'bread'})->0.0
frozenset({'butter', 'coffee', 'eggs'})->0.0
frozenset({'butter', 'coffee', 'jam'})->0.0
frozenset({'milk', 'butter', 'coffee'})->0.0
frozenset({'butter', 'jam', 'eggs'})->0.0
frozenset({'milk', 'butter', 'eggs'})->0.0
frozenset({'milk', 'butter', 'jam'})->0.0
frozenset({'eggs', 'coffee', 'jam'})->0.17
frozenset({'milk', 'eggs', 'coffee'})->0.0
frozenset({'milk', 'coffee', 'jam'})->0.0
frozenset({'milk', 'eggs', 'jam'})->0.0
{'bread'}->support:0.83
{'butter'}->support:0.5
{'milk'}->support:0.5
{'butter', 'bread'}->support:0.5
{'milk', 'bread'}->support:0.5
```

Step 6 Display as a DataFrame

In [16]:
```
result_df=pd.DataFrame(frequent_itemsets,columns=['Itemset','Support'])
result_df
```

Out[16]:

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

In [ ]:

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

In [ ]:

Extra : - > Define Apriori Function without itertools

In [ ]:

In [ ]:

-:- FP - growth

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 U Y
5	C E I K O

Min support = 3

step : 1 Freq. 1-itemsets

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

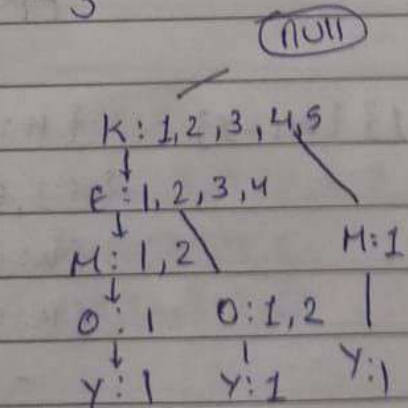
step:2 item sorted

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

TID	sorted item
1	K E M O Y
2	K E O Y
3	K E M
4	K M Y
5	K E O

step:3

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

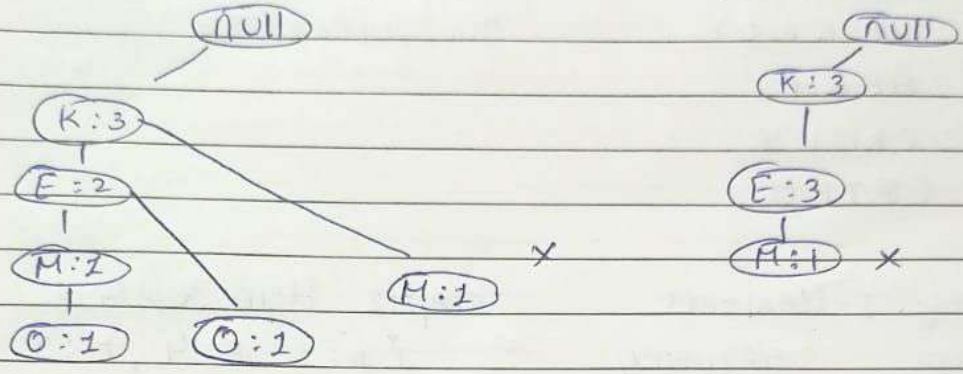


Item	Conditional
Y	{K E M O:1} {K E O:1} {K M:1}
O	{K E M:1} {K E:2}
M	{K E:2} {K:1}
E	{K:4}
K	-

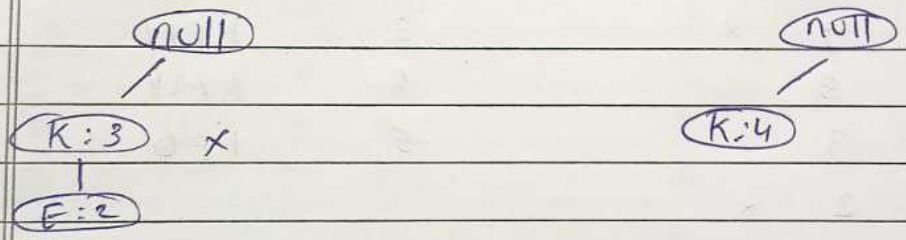
vision

Step:4

Item	condition PB.	condition FP-tree
Y	{KEMO:1} {KEO:1} {KM:1}	{K:3}
O	{KEM:1} {KE:2}	{K:3, E:3}



M	{KE:2} {K:1}	{K:3}
E	{K:4}	{K:4}



Step:5

Item	condition P.B	FP-tree	Frequency Pattern.
Y	{KEMO:1} {KEO:1} {KM:1}	{K:3}	{K, Y:3}
O	{KEM:1} {KE:2}	{K:3, E:3}	{K, O:3} {E, O:3} {K, E, O:3}
M	{KE:2} {K:1}	{K:3}	{K, M:3}
E	{K:4}	{K:4}	{K, E:4}
K	-	-	-

② Min. support - 3

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 freq - 1 - itemsets

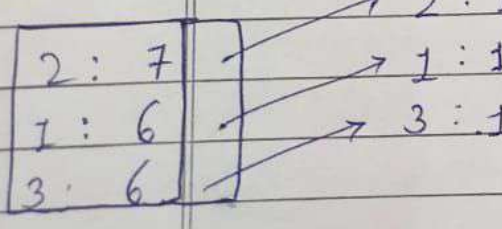
item	frequency
1	6
2	7
3	6
4	2 x
5	2 x

step:2 item sorted

{1:6, 2:7, 3:6}

TID	sorted item
1	2 1
2	2
3	2 3
4	2 1
5	1 3
6	2 3
7	1 3
8	2 1 3
9	2 1 3

step:3



Item	conditional
~~2~~	~~{2, 1}~~ ~~{2, }~~ ~~{2}~~
3	{2, 1 : 2}, {2 : 2}, {1 : 2}
~~1~~	{2 : 2}
2	—

vision

step : 4

Item	conditional	FP-tree
3	{2, 1 : 2}, {2 : 2}, {1 : 2}	
1	{2 : 2}