



# Data Mining

Lab - 1

23010101407

## 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 [10]: df = pd.read_csv("titanic.csv")
df
```

Out[10]:	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	0	PC 17599 7
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803 5
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450
...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536 1
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053 3
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	Nan	1	2	W./C. 6607 2
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369 3
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376

891 rows × 12 columns



## Step-3 Read csv or excel File

```
In [14]: df = pd.read_csv("titanic.csv")
df
```

Out[14]:

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

891 rows × 12 columns



## Step-4 Print Data from csv or excel File

```
In [16]: df = pd.read_csv("titanic.csv")
df
```

Out[16]:	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	0	PC 17599 7
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803 5
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450
...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536 1
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053 3
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	Nan	1	2	W./C. 6607 2
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369 3
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376

891 rows × 12 columns

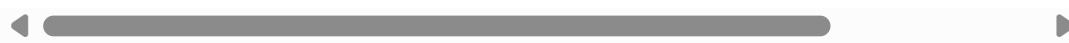


## Step-5 See the First 10 Rows

In [24]: `tp=df.head(10)`  
`tp`

Out[24]:

	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 7.0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	0	PC 17599 71.0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282 7.0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803 53.0
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450 8.0
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877 8.0
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463 51.0
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909 21.0
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742 11.0
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736 30.0



## Step-6 See the Last 10 Rows

In [26]:

```
tp=df.tail(10)
tp
```

Out[26]:

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



## Step-7 Data type of each columns

In [48]:

```
df.dtypes
```

```
Out[48]: 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 [32]: tp=df.info()
tp

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          -----          ----  
 0   PassengerId 891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name          891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare          891 non-null    float64
 10  Cabin         204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

## Step-9 Access a specific column

```
In [53]: cl=df[["Name", "Parch"]]
cl
```

Out[53]:

	Name	Parch
0	Braund, Mr. Owen Harris	0
1	Cumings, Mrs. John Bradley (Florence Briggs Th... ...	0
2	Heikkinen, Miss. Laina	0
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0
4	Allen, Mr. William Henry	0
...	...	...
886	Montvila, Rev. Juozas	0
887	Graham, Miss. Margaret Edith	0
888	Johnston, Miss. Catherine Helen "Carrie"	2
889	Behr, Mr. Karl Howell	0
890	Dooley, Mr. Patrick	0

891 rows × 2 columns

## Step-10 Access rows by their integer location

In [44]: `ri=df.iloc[1]`  
`ri`Out[44]: `PassengerId` 2  
`Survived` 1  
`Pclass` 1  
`Name` Cumings, Mrs. John Bradley (Florence Briggs Th...  
`Sex` female  
`Age` 38.0  
`SibSp` 1  
`Parch` 0  
`Ticket` PC 17599  
`Fare` 71.2833  
`Cabin` C85  
`Embarked` C  
`Name: 1, dtype: object`

## Step-11 Delete a specific Column

In [55]: `ri=df.drop("Parch",axis=1,inplace=True)`  
`ri`

## Step-12 Create a new Column

```
In [61]: df["isCabin"] =~ df['Cabin'].isnull()
```

Out[61]:

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

891 rows × 12 columns



## Step-13 Perform Condition Selection on DataFrame

```
In [65]: df[(df['Pclass']==1)]  
df[(df['Age']>25)]
```

Out[65]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Ticket	Fare
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	STON/O2. 3101282	7.925
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	113803	53.100
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	373450	8.050
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	17463	51.862
...	...	...	...	...	...	...	...	...	...
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	C.A./SOTON 34068	10.500
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	382652	29.125
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	211536	13.000
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	111369	30.000
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	370376	7.750

413 rows × 12 columns



In [73]: df[(df['Pclass']==1) &amp; (df['Sex']=='male')]

Out[73]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Ticket	Fare
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	17463	51.8625
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	113788	35.5000
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	19950	263.0000
30	31	0	1	Uruchurtu, Don. Manuel E	male	40.0	0	PC 17601	27.7208
34	35	0	1	Meyer, Mr. Edgar Joseph	male	28.0	1	PC 17604	82.1708
...	...	...	...	...	...	...	...	...	...
839	840	1	1	Marechal, Mr. Pierre	male	NaN	0	11774	29.7000
857	858	1	1	Daly, Mr. Peter Denis	male	51.0	0	113055	26.5500
867	868	0	1	Roebling, Mr. Washington Augustus II	male	31.0	0	PC 17590	50.4958
872	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	695	5.0000
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	111369	30.0000

122 rows × 12 columns



## Step-14 Compute the sum of value

In [75]: `df["Age"].sum()`

Out[75]: 21205.17

## Step-15 Compute the mean of value

In [77]: `df["Age"].mean()`

Out[77]: 29.69911764705882

## Step-16 Count non-null value (column)

```
In [79]: df["Age"].count()
```

```
Out[79]: 714
```

## Step-17 Find Minimum or Maximum values

```
In [83]: df["Age"].max()  
df["Age"].min()
```

```
Out[83]: 0.42
```

```
In [ ]:
```



**Darshan**  
UNIVERSITY

## Data Mining

### Lab - 3

23010101407

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

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

```
In [7]: titanic = pd.read_csv("titanic.csv")
```

```
In [9]: titanic.head(5)
```

Out[9]:

	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
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

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

In [12]:

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

print(Nominal)
print(Ordinal)
print(binary)
print(Numeric)
```

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

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

In [15]:

```
print("survived assymrtric count -----")
print(titanic['Survived'].value_counts())
print('=====')
print("gender count symmetric count -----")
print(titanic['Sex'].value_counts())
```

```
survived assymrtric count -----
Survived
0      549
1      342
Name: count, dtype: int64
=====
=
gender count symmetric count -----
Sex
male      577
female    314
Name: count, dtype: int64
```

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

```
In [18]: quantitative = ['PassengerId', 'Survived', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare']
for col in quantitative:
    print(col,":")
    print("mean =",titanic[col].mean())
    print("Standard deviation =",titanic[col].std())
    print("minimum =",titanic[col].min())
    print("mode =",titanic[col].mode()[0])
    print("Max =",titanic[col].max())
    print("range =",titanic[col].max() - titanic[col].min())
    print("-----")
```

```
PassengerId :  
mean = 446.0  
Standard deviation = 257.3538420152301  
minimum = 1  
mode = 1  
Max = 891  
range = 890  
-----  
Survived :  
mean = 0.3838383838383838  
Standard deviation = 0.4865924542648585  
minimum = 0  
mode = 0  
Max = 1  
range = 1  
-----  
Pclass :  
mean = 2.308641975308642  
Standard deviation = 0.8360712409770513  
minimum = 1  
mode = 3  
Max = 3  
range = 2  
-----  
Age :  
mean = 29.69911764705882  
Standard deviation = 14.526497332334044  
minimum = 0.42  
mode = 24.0  
Max = 80.0  
range = 79.58  
-----  
SibSp :  
mean = 0.5230078563411896  
Standard deviation = 1.1027434322934275  
minimum = 0  
mode = 0  
Max = 8  
range = 8  
-----  
Parch :  
mean = 0.38159371492704824  
Standard deviation = 0.8060572211299559  
minimum = 0  
mode = 0  
Max = 6  
range = 6  
-----  
Fare :  
mean = 32.204207968574636  
Standard deviation = 49.693428597180905  
minimum = 0.0  
mode = 8.05  
Max = 512.3292  
range = 512.3292  
-----
```

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

```
In [21]: print("passenger class frequency::")
print(titanic['Pclass'].value_counts())
```

```
passenger class frequency::
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 [24]: print("describe numeric attribute::")
print(titanic.describe())
print('-----')
print("describe ALL numeric attribute::")
print(titanic.describe(include = 'all'))
print('-----')
print("describe Object numeric attribute::")
print(titanic.describe(include = 'object'))
print('-----')
```

describe numeric attribute::

	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

-----

describe ALL numeric attribute::

	PassengerId	Survived	Pclass	Name	Sex	\
count	891.000000	891.000000	891.000000	891	891	
unique	Nan	Nan	Nan	891	2	
top	Nan	Nan	Nan	Braund, Mr. Owen Harris	male	
freq	Nan	Nan	Nan	1	577	
mean	446.000000	0.383838	2.308642	Nan	Nan	
std	257.353842	0.486592	0.836071	Nan	Nan	
min	1.000000	0.000000	1.000000	Nan	Nan	
25%	223.500000	0.000000	2.000000	Nan	Nan	
50%	446.000000	0.000000	3.000000	Nan	Nan	
75%	668.500000	1.000000	3.000000	Nan	Nan	
max	891.000000	1.000000	3.000000	Nan	Nan	

	Age	SibSp	Parch	Ticket	Fare	Cabin	\
count	714.000000	891.000000	891.000000	891	891.000000	204	
unique	Nan	Nan	Nan	681	Nan	147	
top	Nan	Nan	Nan	347082	Nan	B96 B98	
freq	Nan	Nan	Nan	7	Nan	4	
mean	29.699118	0.523008	0.381594	Nan	32.204208	Nan	
std	14.526497	1.102743	0.806057	Nan	49.693429	Nan	
min	0.420000	0.000000	0.000000	Nan	0.000000	Nan	
25%	20.125000	0.000000	0.000000	Nan	7.910400	Nan	
50%	28.000000	0.000000	0.000000	Nan	14.454200	Nan	
75%	38.000000	1.000000	0.000000	Nan	31.000000	Nan	
max	80.000000	8.000000	6.000000	Nan	512.329200	Nan	

Embarked

count	889
unique	3
top	S
freq	644
mean	Nan
std	Nan
min	Nan
25%	Nan
50%	Nan
75%	Nan
max	Nan

```
-----
describe Object numeric attribute:::
          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 [27]: print("corelation numeric col:")
print(titanic.corr(numeric_only=True))
```

corelation numeric col:

	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

```
In [29]: print("corelation numeric col:")
print(titanic.corr(numeric_only=True))
```

corelation numeric col:

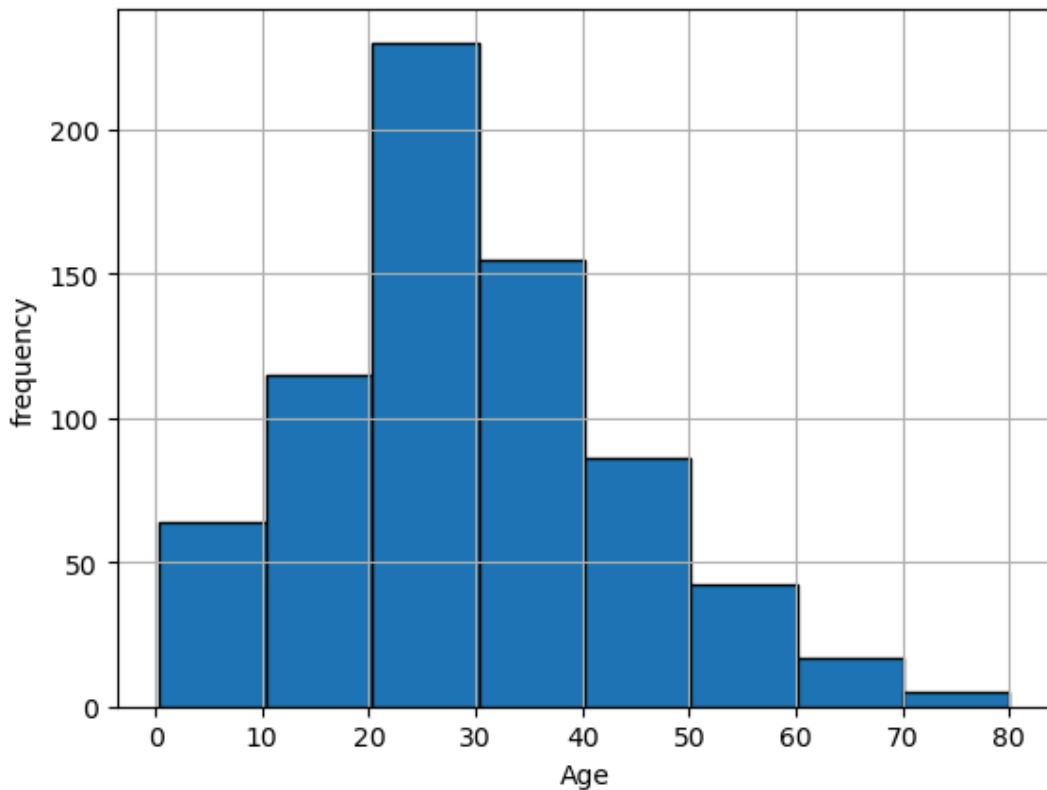
	PassengerId	Survived	Pclass	Age	SibSp	Parch	\
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-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 [32]: import matplotlib.pyplot as plt
titanic['Age'].dropna().hist(bins=8,edgecolor='black')
plt.xlabel('Age')
plt.ylabel('frequency')
plt.show()
```



```
In [ ]:
```

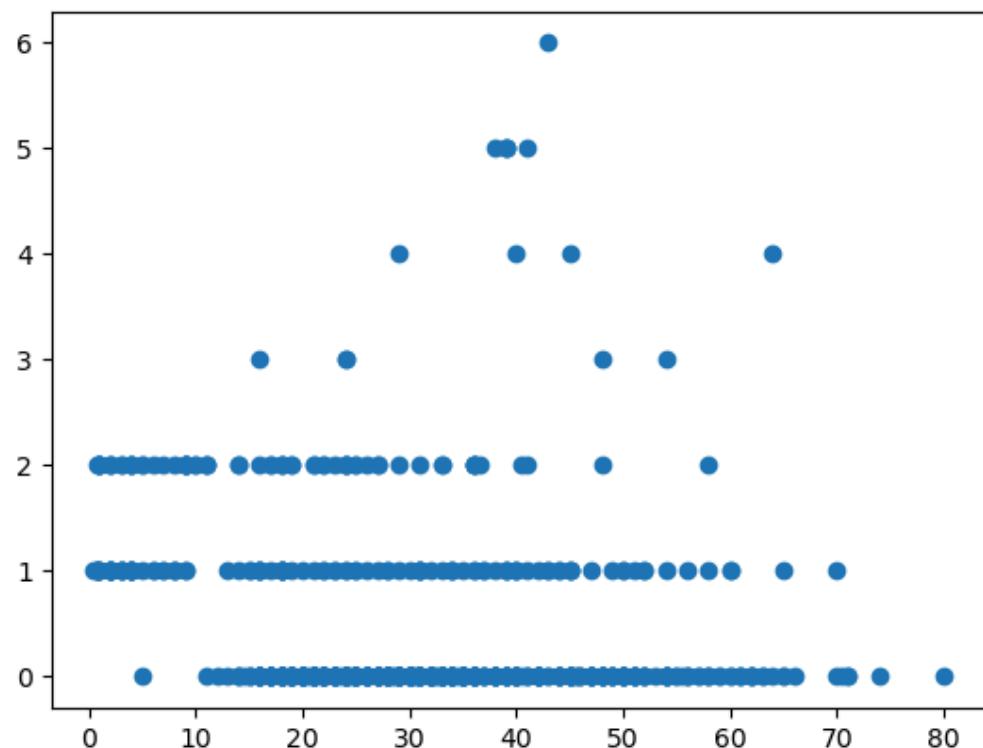
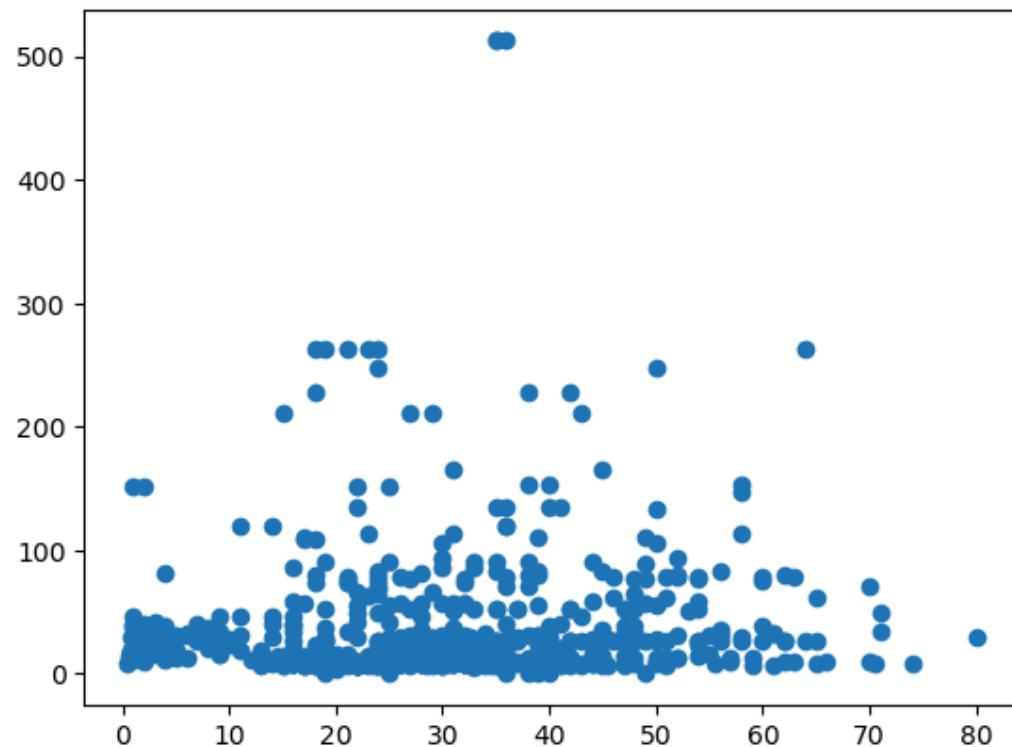
10) A boxplot can also be used to show the distribution of values for each attribute.

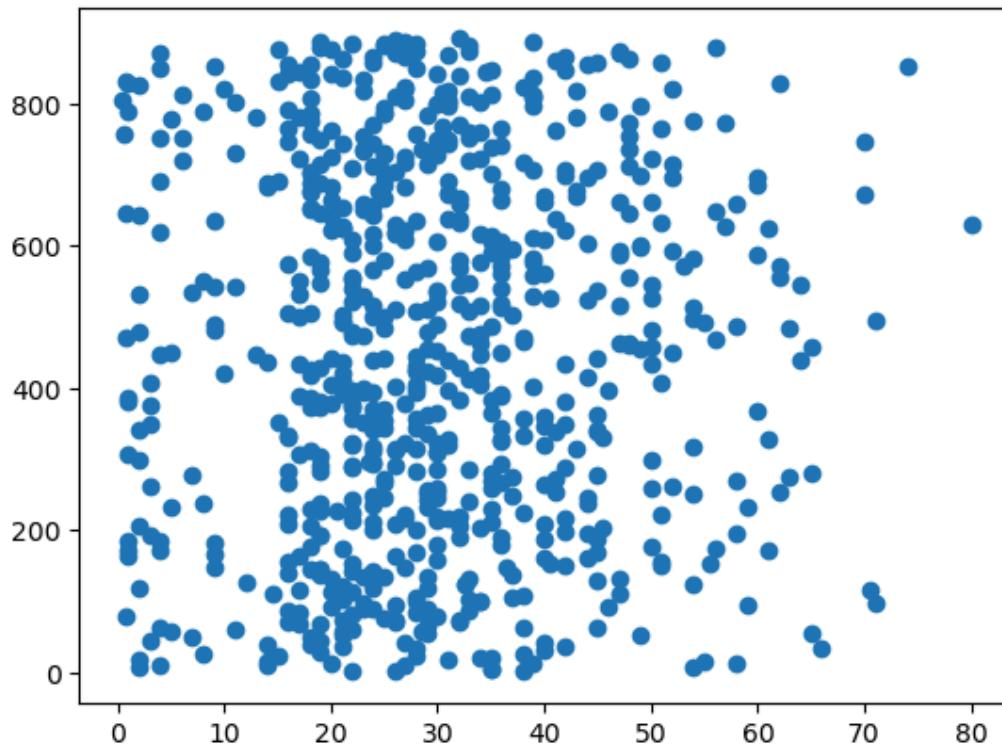
```
In [ ]:
```

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

```
In [38]: plt.scatter(titanic['Age'],titanic['Fare'])
plt.show()

plt.scatter(titanic['Age'],titanic['Parch'])
plt.show()
plt.scatter(titanic['Age'],titanic['PassengerId'])
plt.show()
```





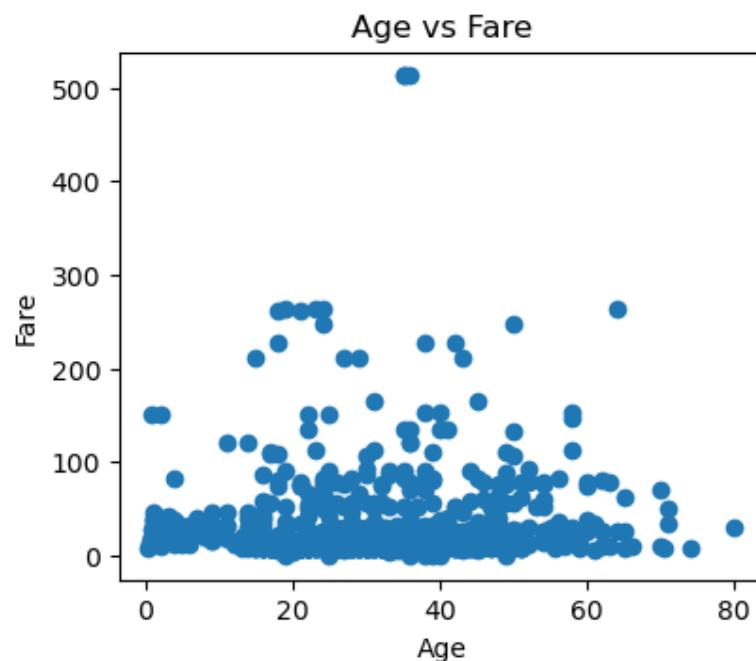
```
In [46]: pairs = [
    {'Age', 'Fare'},
    {'Age', 'sibsp'},
    {'Age', 'Parch'},
    {'Fare', 'sibsp'},
    {'Fare', 'Parch'},
]
plt.figure(figsize=(15,8))

for i, (x,y) in enumerate(pairs):
    plt.subplot(2,3,i+1)
    plt.scatter(titanic[x],titanic[y])
    plt.xlabel(x)
    plt.ylabel(y)
    plt.title(f'{x} vs {y}')
    plt.grid(true)

plt.show()
```

```
NameError                                                 Traceback (most recent call last)
Cell In[46], line 16
      14         plt.ylabel(y)
      15         plt.title(f'{x} vs {y}')
--> 16         plt.grid(true)
      19 plt.show()

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



In [ ]:



# Darshan UNIVERSITY

## Data Mining

### Lab - 4

23010101407

#### Step 1. Import the necessary libraries

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

#### Step 2. Import the dataset from this address.

#### Step 3. Assign it to a variable called chipo.

```
In [5]: url = 'https://raw.githubusercontent.com/justmarkham/DAT8/master/data/chipotle.txt'  
chipo = pd.read_csv(url, sep='\t')
```

#### Step 4. See the first 10 entries

```
In [7]: chipo.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 [13]: # Solution 1
chipo.shape[0]
```

Out[13]: 4622

```
In [15]: # 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 [17]: chipo.shape[1]
```

Out[17]: 5

## Step 7. Print the name of all the columns.

```
In [19]: chipo.columns
```

```
Out[19]: Index(['order_id', 'quantity', 'item_name', 'choice_description',
       'item_price'],
       dtype='object')
```

## Step 8. How is the dataset indexed?

```
In [21]: chipo.index
```

```
Out[21]: RangeIndex(start=0, stop=4622, step=1)
```

## Step 9. Number of Unique Items ?

```
In [29]: print(chipo.nunique())
print(chipo['item_name'].nunique())
```

```
order_id          1834
quantity           9
item_name          50
choice_description 1043
item_price          78
dtype: int64
50
```

## Step 10. Which was the most-ordered item?

```
In [60]: c = chipo.groupby('item_name')
c = c.sum()
c = c.sort_values(['quantity'], ascending = False)
c.head(1)
```

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

## Step 11. How many items were ordered in total?

```
In [48]: chipo['quantity'].sum()
```

```
Out[48]: 4972
```

## Step 12. Turn the item price into a float

### Step 12.a. Check the item price type

```
In [56]: chipo['item_price'].dtype
```

```
Out[56]: dtype('O')
```

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

```
In [70]: dollarization = lambda x: float(x[1:-1])
chipo.item_price = chipo.item_price.apply(dollarization)
```

```

-----
TypeError                                         Traceback (most recent call last)
Cell In[70], line 2
      1 dollarization = lambda x: float(x[1:-1])
----> 2 chipo.item_price = chipo.item_price.apply(dollarization)

File ~/anaconda3/Lib/site-packages/pandas/core/series.py:4924, in Series.apply(self, func, convert_dtype, args, by_row, **kwargs)
    4789 def apply(
    4790     self,
    4791     func: AggFuncType,
    (...),
    4796     **kwargs,
    4797 ) -> DataFrame | Series:
    4798     """
    4799     Invoke function on values of Series.
    4800
    (...),
    4915     dtype: float64
    4916     """
    4917     return SeriesApply(
    4918         self,
    4919         func,
    4920         convert_dtype=convert_dtype,
    4921         by_row=by_row,
    4922         args=args,
    4923         kwargs=kwargs,
-> 4924     ).apply()

File ~/anaconda3/Lib/site-packages/pandas/core/apply.py:1427, in SeriesApply.apply(self)
    1424     return self.apply_compat()
    1426 # self.func is Callable
-> 1427 return self.apply_standard()

File ~/anaconda3/Lib/site-packages/pandas/core/apply.py:1507, in SeriesApply.apply_standard(self)
    1501 # row-wise access
    1502 # apply doesn't have a `na_action` keyword and for backward compat reason
    s
    1503 # we need to give `na_action="ignore"` for categorical data.
    1504 # TODO: remove the `na_action="ignore"` when that default has been change
d in
    1505 # Categorical (GH51645).
    1506 action = "ignore" if isinstance(obj.dtype, CategoricalDtype) else None
-> 1507 mapped = obj._map_values(
    1508     mapper=curried, na_action=action, convert=self.convert_dtype
    1509 )
    1511 if len(mapped) and isinstance(mapped[0], ABCSeries):
    1512     # GH#43986 Need to do list(mapped) in order to get treated as nested
    1513     # See also GH#25959 regarding EA support
    1514     return obj._constructor_expanddim(list(mapped), index=obj.index)

File ~/anaconda3/Lib/site-packages/pandas/core/base.py:921, in IndexOpsMixin._map_values(self, mapper, na_action, convert)
    918 if isinstance(arr, ExtensionArray):
    919     return arr.map(mapper, na_action=na_action)
--> 921 return algorithms.map_array(arr, mapper, na_action=na_action, convert=convert)

```

```

File ~\anaconda3\Lib\site-packages\pandas\core\algorithms.py:1743, in map_array(a
rr, mapper, na_action, convert)
    1741 values = arr.astype(object, copy=False)
    1742 if na_action is None:
-> 1743     return lib.map_infer(values, mapper, convert=convert)
    1744 else:
    1745     return lib.map_infer_mask(
    1746         values, mapper, mask=isna(values).view(np.uint8), convert=convert
    1747     )

File lib.pyx:2972, in pandas._libs.lib.map_infer()

Cell In[70], line 1, in <lambda>(x)
----> 1 dollarization = lambda x: float(x[1:-1])
      2 chipo.item_price = chipo.item_price.apply(dollarization)

TypeError: 'float' object is not subscriptable

```

### Step 12.c. Check the item price type

```
In [72]: chipo['item_price'].dtype
```

```
Out[72]: dtype('float64')
```

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

```
In [78]: final_revenue = (chipo['item_price'] * chipo['quantity']).sum()
print(final_revenue)
```

```
39237.02
```

### Step 15. How many orders were made ?

```
In [84]: chipo['order_id'].nunique()
```

```
Out[84]: 1834
```

### Step 17. How many different choice descriptions are there?

```
In [86]: chipo['choice_description'].nunique()
```

```
Out[86]: 1043
```

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

```
In [92]: c = chipo.groupby('item_name')['quantity'].sum()
c[c>100]
```

```
Out[92]: 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 [103...]: # Solution 1
chipo['revenue'] = (chipo['item_price'] * chipo['quantity'])
order_grouped = chipo.groupby('order_id').sum()
order_grouped['revenue'].mean()
```

```
Out[103...]: 21.39423118865867
```

```
In [ ]: # Solution 2
```

```
Out[ ]: 21.394231188658654
```

```
In [ ]:
```

# Lab - 5 - Data Preprocessing

23010101407

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

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

```
In [5]: data = pd.read_csv('titanic.csv')  
data
```

Out[5]:

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

891 rows × 12 columns



In [7]:

```
data2 = data.copy()
data2.head()
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
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

In [9]: `data2.tail()`

Out[9]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
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

In [11]: `data2.nunique()`

```
Out[11]: PassengerId    891
          Survived      2
          Pclass        3
          Name         891
          Sex           2
          Age          88
          SibSp         7
          Parch         7
          Ticket       681
          Fare         248
          Cabin       147
          Embarked      3
          dtype: int64
```

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

```
In [14]: data_dropnaAll = data2.dropna(how='all')
          data_dropna = data2.dropna()
          data_dropna.count()
          # axis:
          # Specifies whether to drop rows or columns.
          # 0 or 'index' (default): Drops rows containing missing values.
          # 1 or 'columns': Drops columns containing missing values.
```

```
Out[14]: PassengerId    183
          Survived      183
          Pclass        183
          Name         183
          Sex           183
          Age           183
          SibSp         183
          Parch         183
          Ticket       183
          Fare          183
          Cabin        183
          Embarked      183
          dtype: int64
```

```
In [16]: data_fillna = data2.fillna({'Age' : 35,'Cabin' : 'Not Available'})
          # data_fillna = data2.fillna(35)
          data_fillna
```

Out[16]:

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

891 rows × 12 columns



In [ ]:

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

```
In [20]: age_mean = data2.Age.mean()  
age_mean  
fill_data = data2.fillna({'Age' : age_mean})  
fill_data
```

Out[20]:	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticl
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A 211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.000000	1	0	PC 175
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/C 31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	3734
...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W. 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	3703

891 rows × 12 columns

In [22]: `data2['Age'].min()`

Out[22]: 0.42

```
In [24]: data['Age'].max()
```

```
Out[24]: 80.0
```

```
In [26]: data_interpolation = data2.interpolate()  
data_interpolation
```

```
C:\Users\dholn\AppData\Local\Temp\ipykernel_18036\3603189312.py:1: FutureWarning:  
DataFrame.interpolate with object dtype is deprecated and will raise in a future  
version. Call obj.infer_objects(copy=False) before interpolating instead.  
  data_interpolation = data2.interpolate()
```

Out[26]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
<b>0</b>	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171
<b>1</b>	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	0	PC 17599 7
<b>2</b>	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282
<b>3</b>	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803 5
<b>4</b>	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450
...	...	...	...	...	...	...	...	...	...
<b>886</b>	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536 1
<b>887</b>	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053 3
<b>888</b>	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	22.5	1	2	W./C. 6607 2
<b>889</b>	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369 3
<b>890</b>	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376

891 rows × 12 columns



In [28]: `data2.fillna(data2.Age.mean(), inplace=True)`

In [30]: `minAge = data2.Age.min()  
maxAge = data2.Age.max()`

```
data2['MinMaxAge'] = (data2['Age']-minAge)/(maxAge-minAge)
data2
```

Out[30]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticl
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A 211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 175
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/C 31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	3734
...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W. 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	3703

891 rows × 13 columns



In [32]: `data2.describe(include='all')`

Out[32]:

	<b>PassengerId</b>	<b>Survived</b>	<b>Pclass</b>	<b>Name</b>	<b>Sex</b>	<b>Age</b>	<b>SibSp</b>
<b>count</b>	891.000000	891.000000	891.000000	891	891	891.000000	891.000000
<b>unique</b>	NaN	NaN	NaN	891	2	NaN	NaN
<b>top</b>	NaN	NaN	NaN	Braund, Mr. Owen Harris	male	NaN	NaN
<b>freq</b>	NaN	NaN	NaN	1	577	NaN	NaN
<b>mean</b>	446.000000	0.383838	2.308642	NaN	NaN	29.699118	0.523008
<b>std</b>	257.353842	0.486592	0.836071	NaN	NaN	13.002015	1.102743
<b>min</b>	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.000000
<b>25%</b>	223.500000	0.000000	2.000000	NaN	NaN	22.000000	0.000000
<b>50%</b>	446.000000	0.000000	3.000000	NaN	NaN	29.699118	0.000000
<b>75%</b>	668.500000	1.000000	3.000000	NaN	NaN	35.000000	1.000000
<b>max</b>	891.000000	1.000000	3.000000	NaN	NaN	80.000000	8.000000



In [34]:

```
maxAge = data2.Age.max()
noOfDigits = len(str(int(maxAge)))
noOfDigits
```

Out[34]: 2

In [36]:

```
data2['AgeDS'] = data2['Age']/(10**noOfDigits)
```

In [38]:

```
data2
```

Out[38]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A 211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.000000	1	0	PC 175
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/C 31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	3734
...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W. 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	3703

891 rows × 14 columns



In [40]:

```
mins = data2.Age.min()
stds = data2.Age.std()
data2['ageScore'] = (data2['Age'] - mins)/stds
data2
```

Out[40]:	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticl
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A 211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.000000	1	0	PC 175
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/C 31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	3734
...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W. 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	3703

891 rows × 15 columns



In [ ]:

In [ ]:



# Data Mining

## Lab - 6

23010101407

```
# 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
<code>np.mean(X, axis=0)</code>	Compute mean of each column (feature-wise mean).
<code>X - np.mean(X, axis=0)</code>	Centering the data (zero mean).
<code>np.cov(X, rowvar=False)</code>	Compute covariance matrix for features.
<code>np.linalg.eigh(cov_mat)</code>	Get eigenvalues and eigenvectors (for symmetric matrices).
<code>np.argsort(values)[::-1]</code>	Sort values in descending order.
<code>np.dot(X, eigenvectors)</code>	Project original data onto new axes.

## Step 1: Load the Iris Dataset

```
In [5]: import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt
```

```
In [6]: iris = pd.read_csv('iris.csv')  
demo = iris.copy()  
demo
```

Out[6]:

	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 [8]: x = demo.drop(columns="species")
y = demo['species'].map({
    'setosa' : 0,
    'versicolor' : 1,
    'virginica' : 2
})

print("original shape :: ", x.shape)
```

original shape :: (150, 4)

## Step 2: Standardize the data (zero mean)

```
In [12]: x_meaned = x - np.mean(x, axis=0)
print(x_meaned[:5])
```

	sepal_length	sepal_width	petal_length	petal_width
0	-0.743333	0.442667	-2.358	-0.999333
1	-0.943333	-0.057333	-2.358	-0.999333
2	-1.143333	0.142667	-2.458	-0.999333
3	-1.243333	0.042667	-2.258	-0.999333
4	-0.843333	0.542667	-2.358	-0.999333

## Step 3: Compute the Covariance Matrix

```
In [26]: cov_mat = np.cov(x_meaned, rowvar=False)
print(cov_mat)
print(" covariance of matrix shape : ", cov_mat.shape)
```

```
[[ 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]]
covariance of matrix shape : (4, 4)
```

## Step 4: Compute eigenvalues and eigenvectors

```
In [24]: eigen_values,eigen_vector = np.linalg.eigh(cov_mat)
print("eigen_values",eigen_values)
print('\neigen_vector\n',eigen_vector[:, :2])

eigen_values [0.02383509 0.0782095 0.24267075 4.22824171]

eigen_vector
[[ 0.31548719  0.58202985]
 [-0.3197231 -0.59791083]
 [-0.47983899 -0.07623608]
 [ 0.75365743 -0.54583143]]
```

## Step 5: Compute eigenvalues and eigenvectors in descending order

```
In [35]: sorted_index = np.argsort(eigen_values[::-1])
sorted_eigenvalues = eigen_values[sorted_index]
sorted_eigenvecotores = eigen_vector[:,sorted_index]
print(sorted_index)
print(sorted_eigenvalues)
print(sorted_eigenvecotores)

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

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

```
In [45]: k = 2
eigenvector_subset = sorted_eigenvecotores[:,0:k]
print(eigenvector_subset)

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

## Step 7: Project the data onto the top k eigenvectors

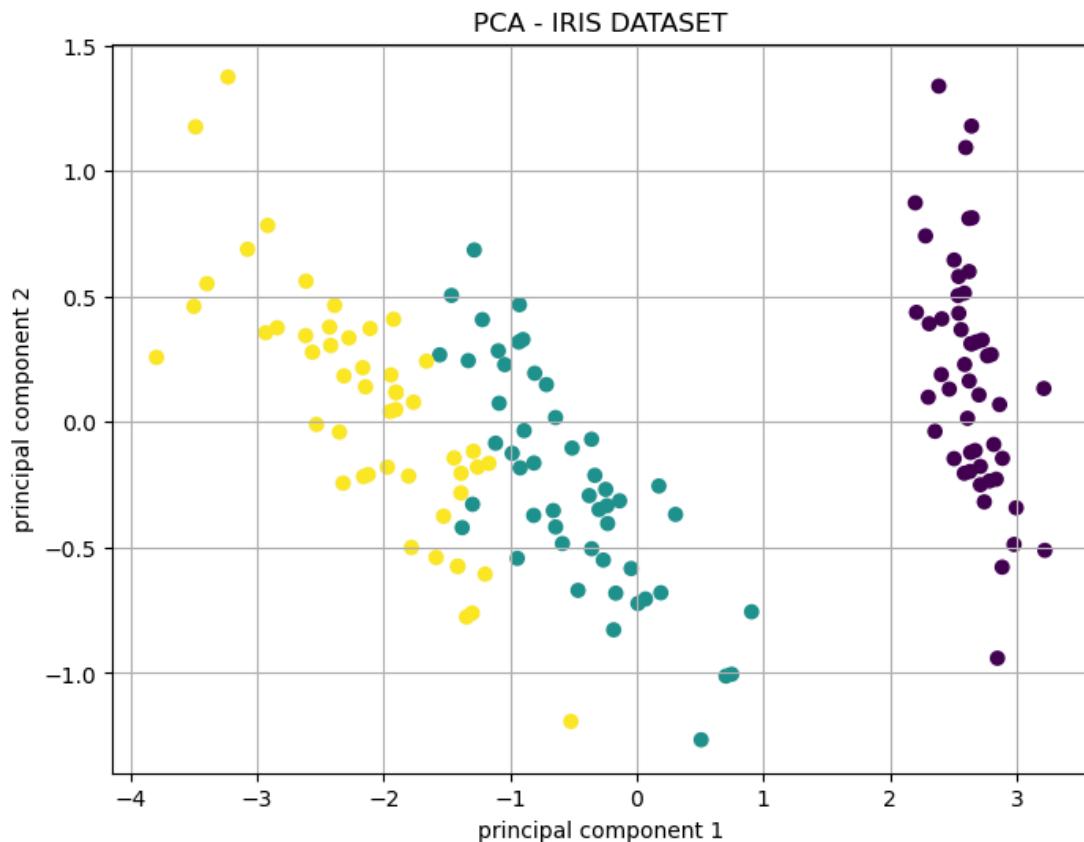
```
In [49]: x_reduced = np.dot(x_meaned, eigenvector_subset)
print("Reduced data shape :: ",x_reduced.shape)
```

Reduced data shape :: (150, 2)

## Step 8: Plot the PCA-Reduced Data

```
In [57]: plt.figure(figsize=(8,6))
plt.scatter(x_reduced[:,0],x_reduced[:,1],c=y)
plt.xlabel('principal component 1')
plt.ylabel('principal component 2')
plt.title('PCA - IRIS DATASET')
plt.grid()
plt.plot()
```

Out[57]: []



## 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]: `equal_fre`

```
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 [71]: `210/3`

Out[71]: `70.0`

In [77]: `5+70`

Out[77]: `75`

In [79]: `90+70`

Out[79]: `160`

In [ ]:



## Data Mining

### Lab - 7 (Part 2)

23010101407

#### Step 1: Load the Dataset

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

In [1]:

```
import pandas as pd

data = pd.read_csv('Tdata.csv')
data.head()
```

Out[1]:

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

#### Step 2: Drop the 'Transaction' Column

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

In [4]:

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

#### Step 3: Count Single Items

See how many transactions include each item.

In [7]:

```
data.sum()
```

```
Out[7]: 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 [10]: from itertools import combinations

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

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

    return result
```

## Step 5: Run Apriori

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

```
In [13]: frequent_itemsets=find_frequent_itemsets(data,min_support=0.5)
for itemset,support in frequent_itemsets:
    print(f"set(itemset)}-> support: {support}")

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

## Step 6 Display as a DataFrame

```
In [15]: result_data = pd.DataFrame(frequent_itemsets,columns=['itemset','support'])
result_data
```

Out[15]:

	itemset	support
0	(bread)	0.83
1	(butter)	0.50
2	(milk)	0.50
3	(bread, butter)	0.50
4	(bread, milk)	0.50

In [ ]:

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

In [ ]:

## Extra : - > Define Apriori Function without itertools

In [ ]:

## \* Apply Apriori Algorithm

(1) TID items

TID	items	Ans :- C, itemset minsup
100	1 3 4	{1} 2
200	2 3 5	{2} 3
300	1 2 3 5	{3} 3
400	2 5	{4} 1 X
		{5} 3

Scan

L<sub>1</sub> items support

{1}	2
{2}	3
{3}	3
{4,5}	3

### frequent - 2

C <sub>2</sub>	items	support	L <sub>2</sub>	items	support
{1,2}	1 X		{1,3}	2	
{1,3}	2		{2,3}	2	
{1,5}	1 X		{2,5}	3	
{2,3}	2		{3,5}	2	
{2,5}	3				
{3,5}	2				

### frequent - 3

C <sub>3</sub>	item	support	L <sub>3</sub>	item	support
{1,2,3}	1 X		{2,3,5}	2	
{1,3,5}	1 X				
{2,3,5}	2				

## => Rule Generation

Association Rule support confidence (%)

$2^3 \rightarrow 5$	2	$2/2 = 1$	100%
$3^5 \rightarrow 2$	2	$2/2 = 1$	100%
$2^5 \rightarrow 3$	2	$2/3 = 0.66$	66%
$2 \rightarrow 3^5$	2	$2/3 = 0.66$	66%
$3 \rightarrow 2^5$	2	$2/3 = 0.66$	66%
$5 \rightarrow 2^3$	2	$2/3 = 0.66$	66%

Strong association  $\rightarrow 2^3 \rightarrow 5$   
 $3^5 \rightarrow 2$

②

TID

Items

Min-sup=3

2

Bread, milk

3

Bread, diaper, beer, egg

4

milk, diaper, beer, cola

5

milk, diaper, beer, cola

Bread, milk, diaper, cola

Frequent 1

C1	Items	Support	L1	Items	Support
	{Bread}	3		{bread}	3
	{diaper}	4		{diaper}	4
	{beer}	3		{beer}	3
	{egg}	1	X		
	{cola}	3		{cola}	3
	{milk}	4		{milk}	4

## frequent - 2

$L_2$ Items	support	$L_2$ Items	support
{bread, diaper}	2 X	{diaper, beer}	3
{bread, beer}	1 X	{diaper, cola}	3
{bread, cola}	1 X	{diaper, milk}	3
{bread, milk}	2 X	{cola, milk}	3
{diaper, beer}	3		
{diaper, cola}	3		
{diaper, milk}	3		
{beer, cola}	2 X		
{beer, milk}	2 X		
{cola, milk}	3		

$L_3$ Items	support	$L_3$ Items	support
{diaper, beer, cola}	2 X	{diaper, cola, milk}	3
{diaper, beer, milk}	2 X		
{diaper, cola, milk}	3		

## → Rule Generation

Association Rule	Support	confi.	%
diaper → cola ^ milk	2	$3/4 = 0.75$	75%
cola → diaper ^ milk	3	$3/3 = 1$	100%
milk → diaper ^ cola	3	$3/4 = 0.75$	75%
diaper ^ cola → milk	3	$3/3 = 1$	100%
diaper ^ milk → cola	3	$3/3 = 1$	100%
cola ^ milk → diaper	3	$3/3 = 1$	100%

## Lab-8

### \* FP - Growth Exemple

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

→ Step 1 : freq itemset min. support 2/5

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

→ Step 2 : transaction with item sorted based on frequency, and ignoring the infrequent items  
{K:5, E:4, O:3, M:3, Y:3}

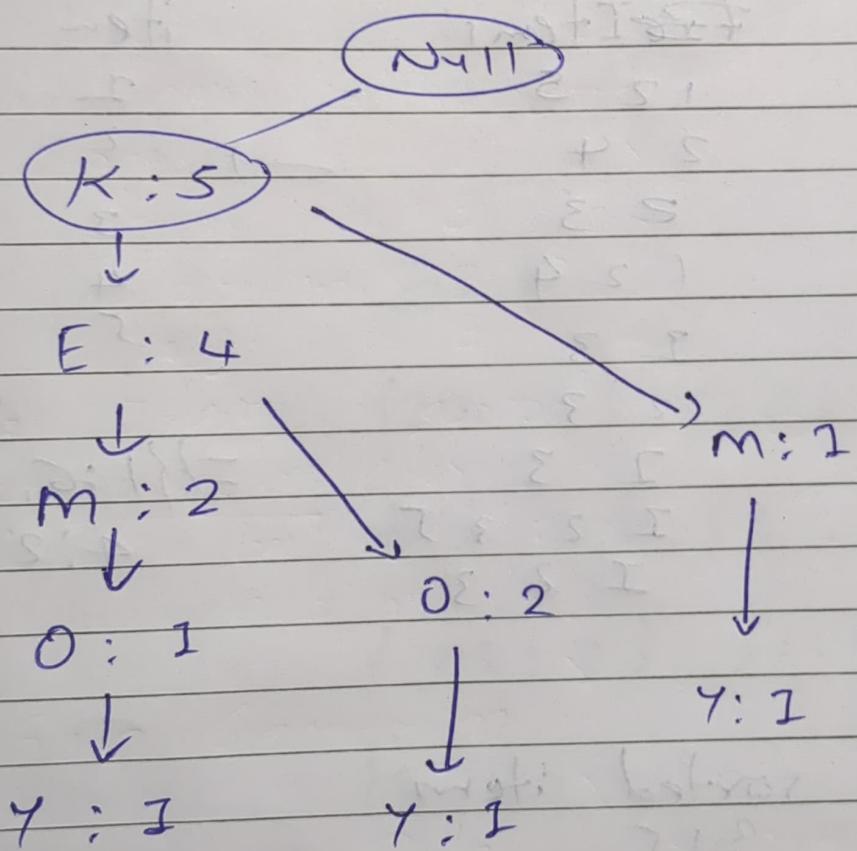
TID

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

## ~) Conditional pattern Base



item

Y

Conditional pattern Base Fptree

{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=4S}

K

—

frequent pattern gen

$\{K, Y: 3\}$

$\{K, O: 3\}$

$\{K, M: 3\}$

$\{K, E: 4\}$

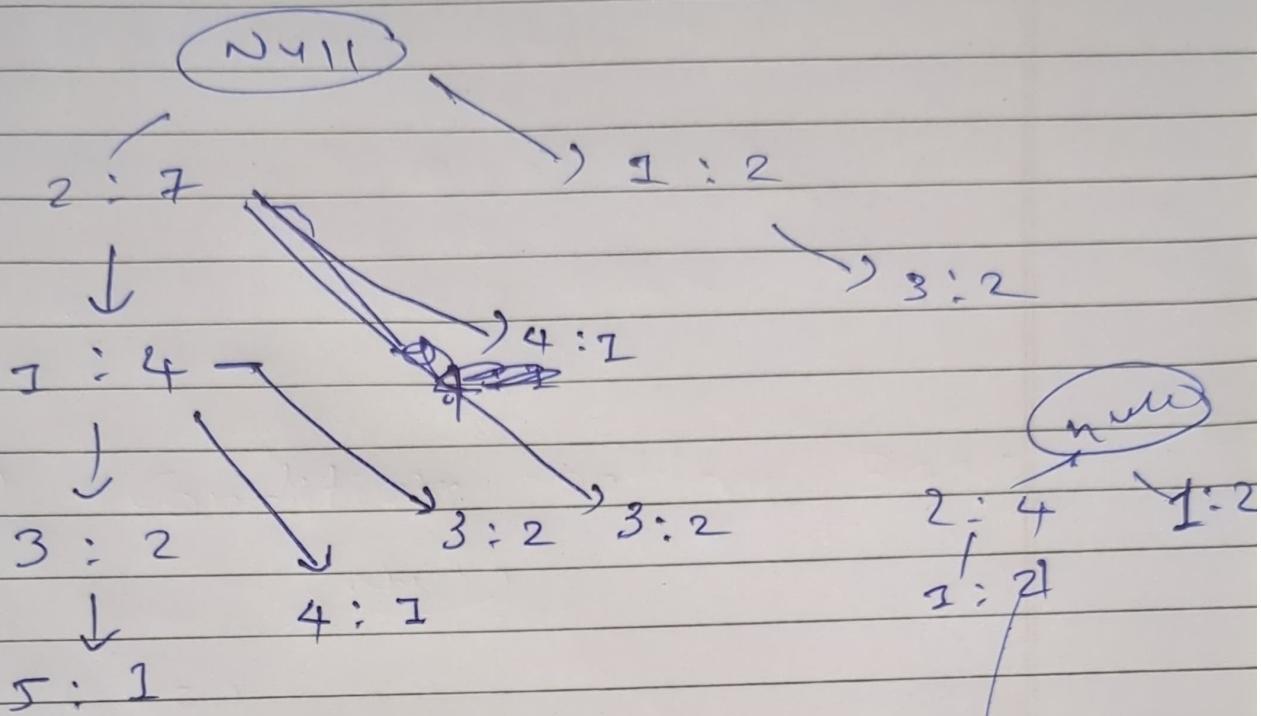
Ex: 2

IID	Items	item	freq
1	1 2 5	1	6
2	2 4	2	7
3	2 3	3	6
4	1 2 4	4	2
5	1 3	5	2
6	1 2 3		
7	1 3		
8	1 2 3 5		
9	1 2 3		

$\Rightarrow \{1: 6, 2: 7, 3: 6, 4: 2, 5: 2\}$

→ sorted

IID	sorted items
1	2 1 5
2	2 4
3	2 3
4	2 1 4
5	1 3
6	1 2 3
7	1 3
8	2 1 3 5
9	2 1 3



item condition

pp

5

$\{2:1\}, \{2:3:1\}$

$\{1:2\}$

4

$\{2:1:2\}, \{2:1\}$

$\{2:2\}$

3

$\{2:1:2\}, \{1:2\}, \{2:2\}$

$\{2:4\}$   
 $\{1:2\}$

2

$\{2:4\}$

$\{2:4\}$

free.

$\{2, 5:2\}, \{1;5:2\}, \{2, 1, 5:2\}$

$\{2, 4:2\}$

$\{2, 3:4\}, \{1, 3:2\}$

$\{2, 2:4\}$

# Data Mining - Lab - 2

## Numpy & Perform Data Exploration with Pandas

23010101407

---

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

### Step 2. Create a 1D array of numbers

```
In [5]: # array ma badha element same data type na Leva na
arr = np.array([1,4,2,5,6,3])
arr
```

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

```
In [21]: # jetli range hoy enathi ek ocho ave
# jyathi start krvu hoy te coma seprated dy devanu (2,10) to otp 2 to 9 no array
arr = np.arange(10)
type(arr) #type of arr
arr.dtype # datatype ape int, float etc...
arr.size # size ape
arr.ndim # ---> no. of dimantion
```

```
Out[21]: 1
```

## Step 3. Reshape 1D to 2D Array

```
In [11]: arr2D = np.array([[1,2,3],[4,5,6],[7,8,9]])
arr2D
```

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

```
In [13]: # (row,col)
arr2D= np.arange(12).reshape(3,4)
arr2D
```

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

## Step 4. Create a Linspace array

```
In [23]: #auto data type float
#starting point,ending point,no. of output(apde ketla joi ch e)
#starting and ending point both are inclisive(bey array ma ave)
arr = np.linspace(0,5,10)
arr
```

```
Out[23]: array([0.          , 0.55555556, 1.1111111, 1.66666667, 2.22222222,
 2.77777778, 3.33333333, 3.88888889, 4.44444444, 5.          ])
```

## Step 5. Create a Random Numbered Array

```
In [37]: arr = np.random.rand(5)
arr
```

```
Out[37]: array([0.92066394, 0.18001599, 0.27637333, 0.82266963, 0.04610274])
```

```
In [ ]:
```

## Step 6. Create a Random Integer Array

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

```
In [ ]:
```

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

```
In [39]: arr = np.array([1,4,2,5,6,3])
arr
```

```
Out[39]: array([1, 4, 2, 5, 6, 3])
```

```
In [41]: arr = np.array([1,4,2,5,6,3])
arr.max()
```

Out[41]: 6

```
In [43]: arr = np.array([1,4,2,5,6,3])
arr.min()
```

Out[43]: 1

```
In [47]: #index ape and index 0 thi start thy
arr = np.array([1,4,2,5,6,3])
arr.argmax()
```

Out[47]: 4

```
In [49]: arr = np.array([1,4,2,5,6,3])
arr.argmin()
```

Out[49]: 0

## Step 8. Indexing in 1D Array

```
In [55]: arr = np.array([1,4,2,5,6,3])
arr[0]
```

Out[55]: 1

## Step 9. Indexing in 2D Array

```
In [57]: arr2D= np.arange(12).reshape(3,4)
arr2D[1,0]
```

Out[57]: 4

## Step 10. Conditional Selection

```
In [63]: arr= np.arange(12)
arr
arr[arr%2==0]
```

Out[63]: array([ 0, 2, 4, 6, 8, 10])

In [ ]:

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

## Pandas

### Step 1. Import the necessary libraries

```
In [1]: 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 [3]: df = pd.read_csv("https://raw.githubusercontent.com/justmarkham/DAT8/master/datasets/ml-lenses.csv")
```

```
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 4. See the first 25 entries**

```
In [5]: tp=df.head(25)
```

```
tp
```

Out[5]:

	<b>user_id</b>	<b>age</b>	<b>gender</b>	<b>occupation</b>	<b>zip_code</b>
<b>0</b>	1	24	M	technician	85711
<b>1</b>	2	53	F	other	94043
<b>2</b>	3	23	M	writer	32067
<b>3</b>	4	24	M	technician	43537
<b>4</b>	5	33	F	other	15213
<b>5</b>	6	42	M	executive	98101
<b>6</b>	7	57	M	administrator	91344
<b>7</b>	8	36	M	administrator	05201
<b>8</b>	9	29	M	student	01002
<b>9</b>	10	53	M	lawyer	90703
<b>10</b>	11	39	F	other	30329
<b>11</b>	12	28	F	other	06405
<b>12</b>	13	47	M	educator	29206
<b>13</b>	14	45	M	scientist	55106
<b>14</b>	15	49	F	educator	97301
<b>15</b>	16	21	M	entertainment	10309
<b>16</b>	17	30	M	programmer	06355
<b>17</b>	18	35	F	other	37212
<b>18</b>	19	40	M	librarian	02138
<b>19</b>	20	42	F	homemaker	95660
<b>20</b>	21	26	M	writer	30068
<b>21</b>	22	25	M	writer	40206
<b>22</b>	23	30	F	artist	48197
<b>23</b>	24	21	F	artist	94533
<b>24</b>	25	39	M	engineer	55107

## Step 5. See the last 10 entries

In [7]:

```
tp=df.tail(10)
tp
```

Out[7]:

	<b>user_id</b>	<b>age</b>	<b>gender</b>	<b>occupation</b>	<b>zip_code</b>
<b>933</b>	934	61	M	engineer	22902
<b>934</b>	935	42	M	doctor	66221
<b>935</b>	936	24	M	other	32789
<b>936</b>	937	48	M	educator	98072
<b>937</b>	938	38	F	technician	55038
<b>938</b>	939	26	F	student	33319
<b>939</b>	940	32	M	administrator	02215
<b>940</b>	941	20	M	student	97229
<b>941</b>	942	48	F	librarian	78209
<b>942</b>	943	22	M	student	77841

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

```
In [11]: df["user_id"].count()
```

```
Out[11]: 943
```

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

```
In [17]: # 1 is col  
#0 is row  
df.shape[1]
```

```
Out[17]: 5
```

## Step 8. Print the name of all the columns.

```
In [19]: df.columns
```

```
Out[19]: Index(['user_id', 'age', 'gender', 'occupation', 'zip_code'], dtype='object')
```

## Step 9. How is the dataset indexed?

```
In [21]: df.index
```

```
Out[21]: RangeIndex(start=0, stop=943, step=1)
```

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

```
In [29]: df.dtypes
```

```
Out[29]: user_id      int64
          age         int64
          gender     object
          occupation  object
          zip_code   object
          dtype: object
```

## Step 11. Print only the occupation column

```
In [25]: cl = df[["user_id","occupation"]]
cl
```

	user_id	occupation
0	1	technician
1	2	other
2	3	writer
3	4	technician
4	5	other
...	...	...
938	939	student
939	940	administrator
940	941	student
941	942	librarian
942	943	student

943 rows × 2 columns

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

```
In [45]: df["occupation"].nunique()
```

```
Out[45]: 21
```

## Step 13. What is the most frequent occupation?

```
In [51]: #mode is give to occurrences
          #when i give enter a 0 so give me a first occurrences
          df["occupation"].mode()[0]
```

```
Out[51]: 'student'
```

## Step 14. Summarize the DataFrame.

```
In [39]: tp=df.describe()
tp
```

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

## Step 15. Summarize all the columns

```
In [53]: tp=df.info()
tp

<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 16. Summarize only the occupation column

```
In [55]: df[ "occupation"].info()

<class 'pandas.core.series.Series'>
RangeIndex: 943 entries, 0 to 942
Series name: occupation
Non-Null Count  Dtype  
----- 
943 non-null    object  
dtypes: object(1)
memory usage: 7.5+ KB
```

## Step 17. What is the mean age of users?

```
In [59]: df[ "age"].mean()
```

Out[59]: 34.05196182396607

## Step 18. What is the age with least occurrence?

```
In [61]: counts = df['age'].value_counts()  
  
least_common = counts[counts == counts.min()]  
least_common
```

```
Out[61]: 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! 