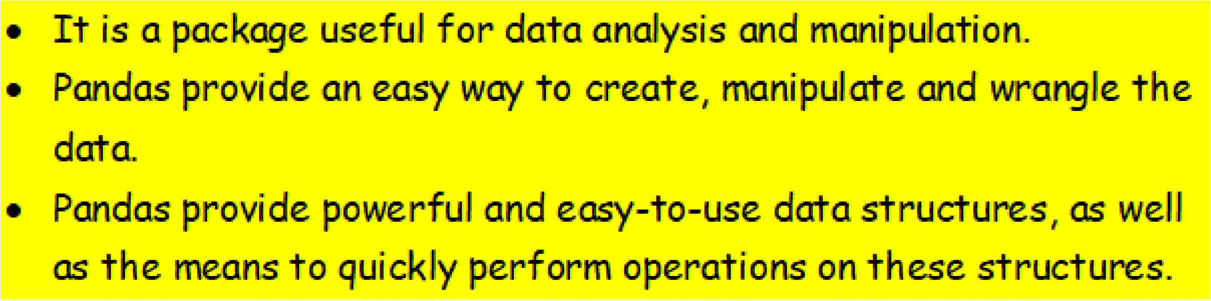
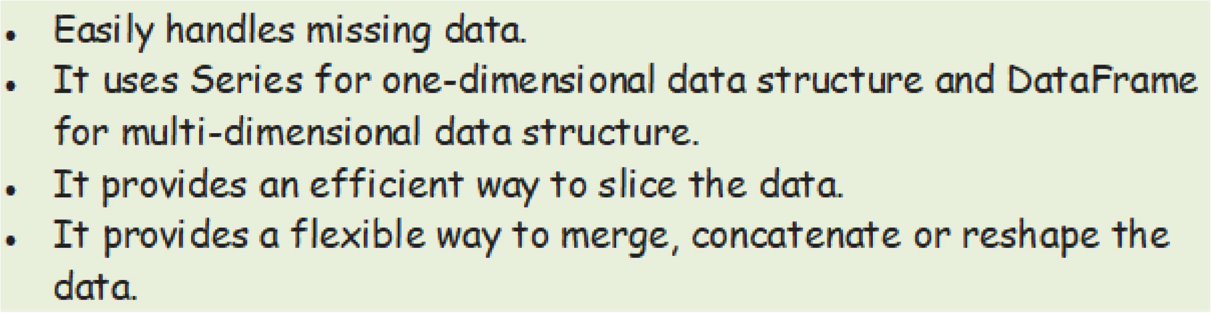
Module 2 Data Clening using Pandas Lab Practical

what is pandas?



- It is a package useful for data analysis and manipulation.
- Pandas provide an easy way to create, manipulate and wrangle the data.
- Pandas provide powerful and easy-to-use data structures, as well as the means to quickly perform operations on these structures.

Data scientists use Pandas for its following advantages:



- Easily handles missing data.
- It uses Series for one-dimensional data structure and DataFrame for multi-dimensional data structure.
- It provides an efficient way to slice the data.
- It provides a flexible way to merge, concatenate or reshape the data.

Note : Sample data and datasets used in this Notebook will be available in below github url

https://github.com/kundetivamsi2001/Datasets_AI-ML (https://github.com/kundetivamsi2001/Datasets_AI-ML)

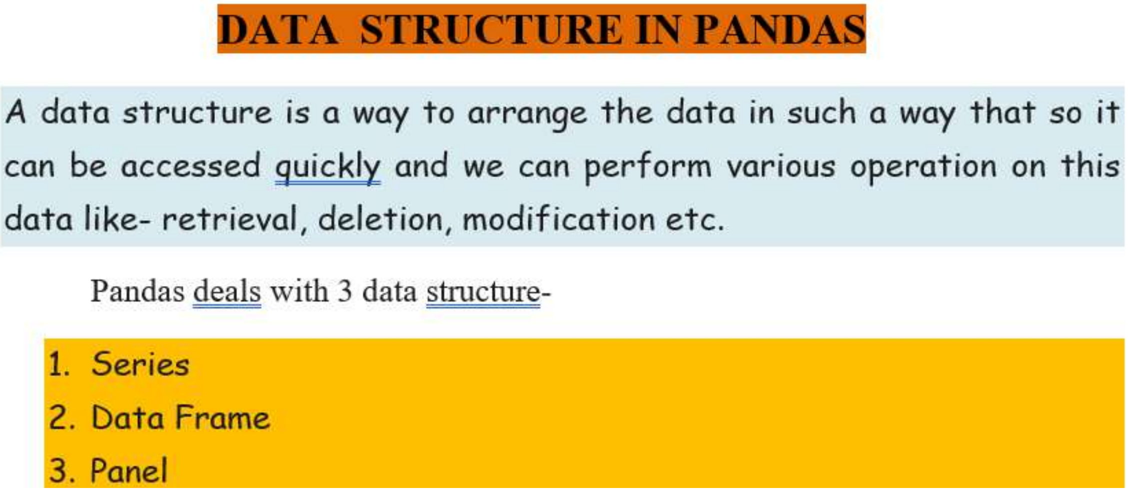
1. How Install and import pandas

In [2]:

```
!pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\vamsi2001\appdata\local\progr
ams\python\python39\lib\site-packages (1.3.3)
Requirement already satisfied: numpy>=1.17.3 in c:\users\vamsi2001\appdata\loca
l\programs\python\python39\lib\site-packages (from pandas) (1.22.4)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\vamsi2001\app
data\local\programs\python\python39\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in c:\users\vamsi2001\appdata\local
\programs\python\python39\lib\site-packages (from pandas) (2021.1)
Requirement already satisfied: six>=1.5 in c:\users\vamsi2001\appdata\local\pro
grams\python\python39\lib\site-packages (from python-dateutil>=2.7.3->pandas)
(1.16.0)
```

```
WARNING: Error parsing dependencies of bleach: Expected matching RIGHT_PARENTHE
SIS for LEFT_PARENTHESIS, after version specifier
    tinycss2 (>=1.1.0<1.2) ; extra == 'css'
             ~~~~~~~~~^
```

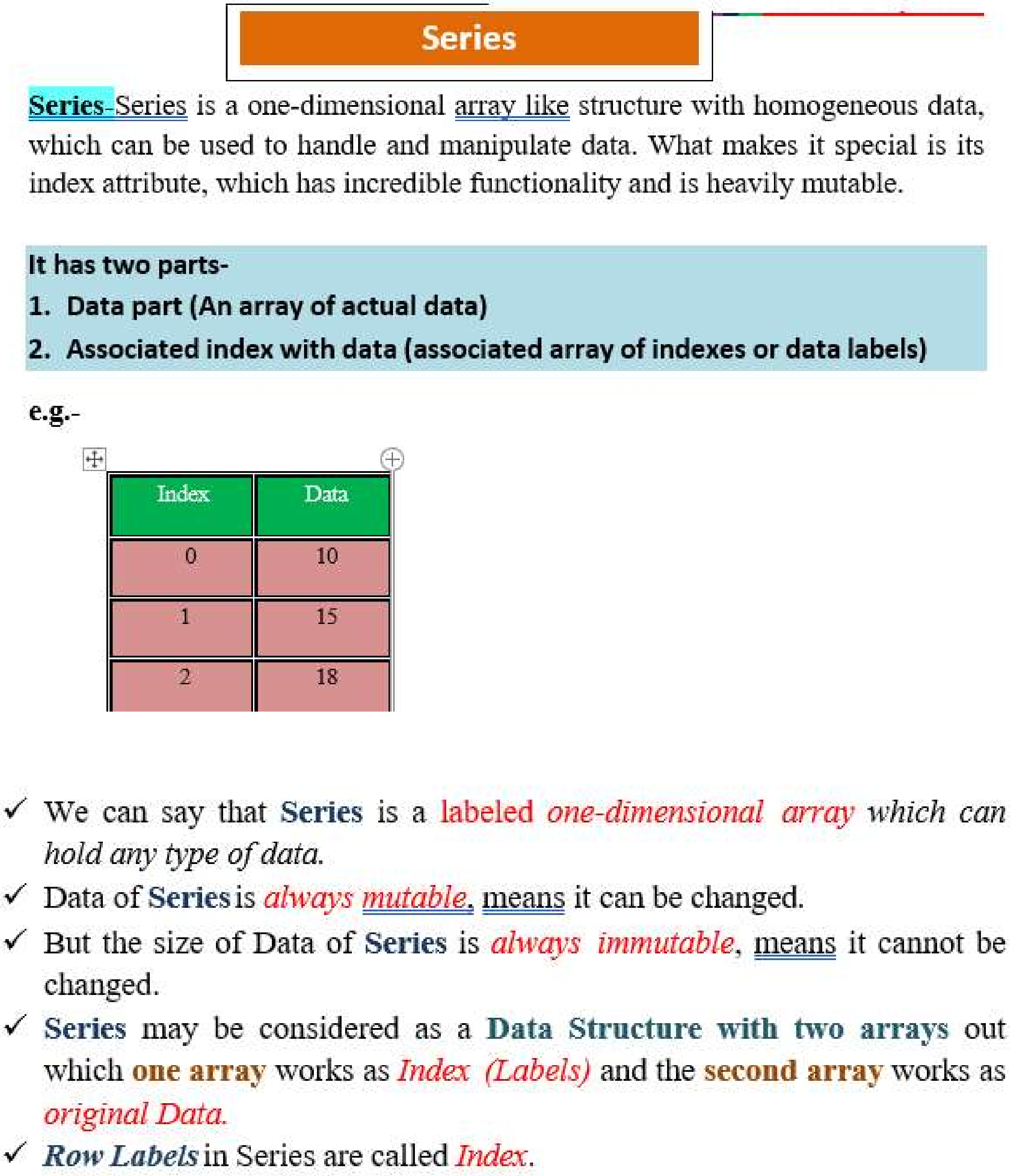


DATA STRUCTURE IN PANDAS

A data structure is a way to arrange the data in such a way that so it can be accessed quickly and we can perform various operation on this data like- retrieval, deletion, modification etc.

Pandas deals with 3 data structure-

1. Series
2. Data Frame
3. Panel



Series

Series-Series is a one-dimensional array like structure with homogeneous data, which can be used to handle and manipulate data. What makes it special is its index attribute, which has incredible functionality and is heavily mutable.

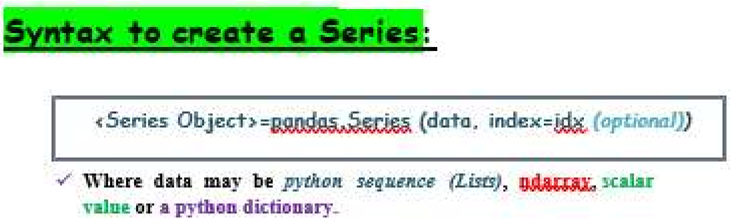
It has two parts-

1. **Data part (An array of actual data)**
2. **Associated index with data (associated array of indexes or data labels)**

e.g.-

Index	Data
0	10
1	15
2	18

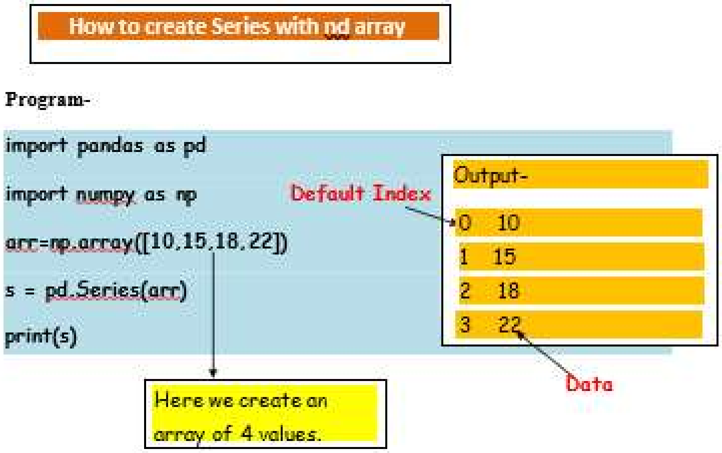
- ✓ We can say that **Series** is a labeled *one-dimensional array which can hold any type of data.*
- ✓ Data of **Series** is *always mutable,* means it can be changed.
- ✓ But the size of Data of **Series** is *always immutable*, means it cannot be changed.
- ✓ **Series** may be considered as a **Data Structure with two arrays** out which **one array** works as *Index (Labels)* and the **second array** works as *original Data.*
- ✓ ***Row Labels*** in Series are called *Index*.



Syntax to create a Series:

<Series Object>=pandas.Series (data, index=idx (optional))

✓ Where data may be *python sequence (Lists)*, ndarray, scalar value or a python dictionary.



How to create Series with nd array

Program-

```
import pandas as pd
import numpy as np
arr=np.array([10,15,18,22])
s = pd.Series(arr)
print(s)
```

Output-

```
0  10
1  15
2  18
3  22
```

In [3]:

```python
#how to create series from arrays in pandas
import pandas as pd
import numpy as np
arr=np.array([10,20,30,40,9,0,7,6,5,4,3,2,12,34,8798])
s=pd.Series(arr)
s
```

```
c:\users\vamsi2001\appdata\local\programs\python\python39\lib\site-packages\num
py\_distributor_init.py:30: UserWarning: loaded more than 1 DLL from .libs:
c:\users\vamsi2001\appdata\local\programs\python\python39\lib\site-packages\num
py\.libs\libopenblas.EL2C6PLE4ZYW3ECEVIV3OXXGRN2NRFM2.gfortran-win_amd64.dll
c:\users\vamsi2001\appdata\local\programs\python\python39\lib\site-packages\num
py\.libs\libopenblas64__v0.3.21-gcc_10_3_0.dll
  warnings.warn("loaded more than 1 DLL from .libs:"
```

Out[3]:

```
0       10
1       20
2       30
3       40
4        9
5        0
6        7
7        6
8        5
9        4
10       3
11       2
12      12
13      34
14    8798
dtype: int32
```

In [4]:

```
#create series with label indexing
import numpy as np
arr=np.array([1,2,3,4])
s=pd.Series(arr,index=['first','second','third','fourth'])
s
```

Out[4]:

```
first     1
second    2
third     3
fourth    4
dtype: int32
```

In [5]:

```
#Selection operator in series
s[1:4] # selecting data from index 1 to 3 i.e (n-1)
```

Out[5]:

```
second    2
third     3
fourth    4
dtype: int32
```

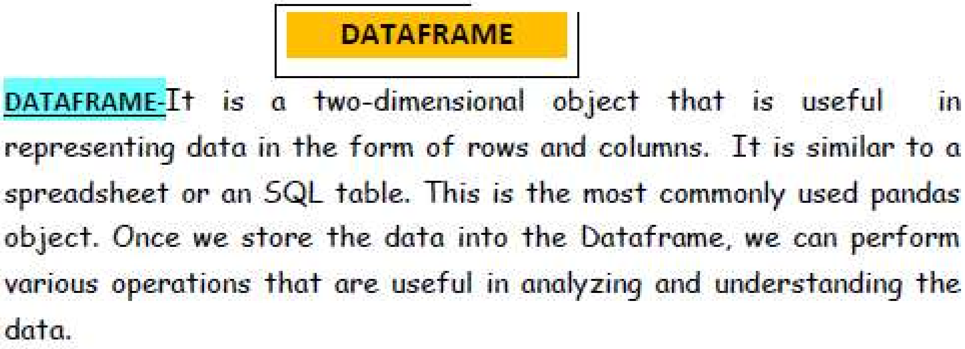
In [6]:

```
#Series can be created from list,tuple,dictionary
#create series from dictionary
dc={'Name':'Ram','Dept':'ECE', 'Marks':500,'Age':23}
sr=pd.Series(dc)
sr
```

Out[6]:

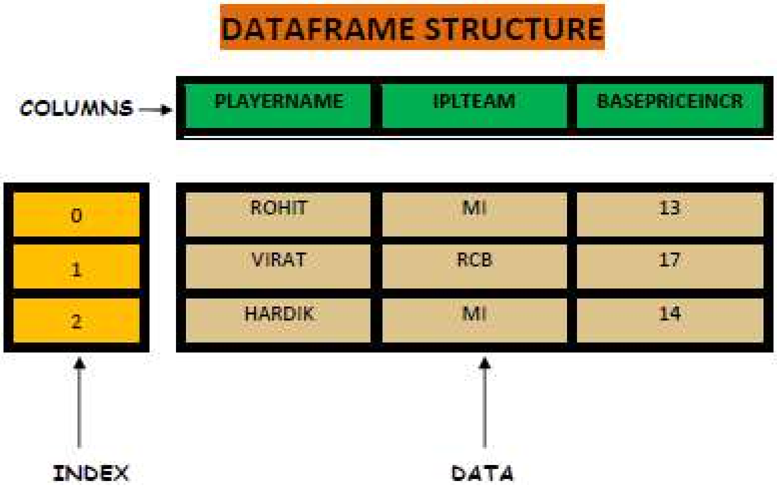
```
Name     Ram
Dept     ECE
Marks    500
Age       23
dtype: object
```

2. Most used Data structure in pandas is DataFrame



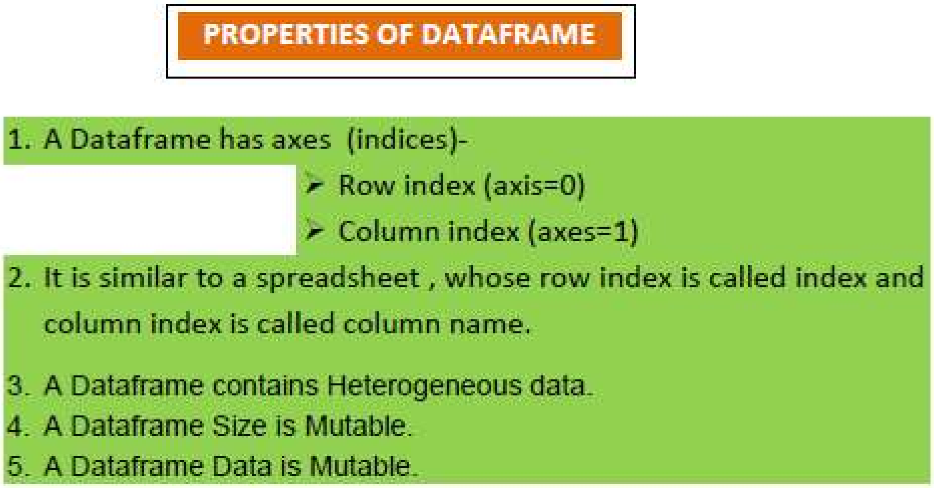
DATAFRAME

DATAFRAME-It is a two-dimensional object that is useful in representing data in the form of rows and columns. It is similar to a spreadsheet or an SQL table. This is the most commonly used pandas object. Once we store the data into the Dataframe, we can perform various operations that are useful in analyzing and understanding the the data.



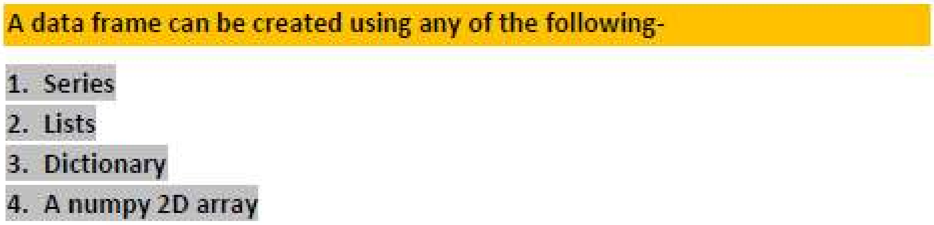
DATAFRAME STRUCTURE

	PLAYERNAME	IPLTEAM	BASEPRICEINCR
0	ROHIT	MI	13
1	VIRAT	RCB	17
2	HARDIK	MI	14



PROPERTIES OF DATAFRAME

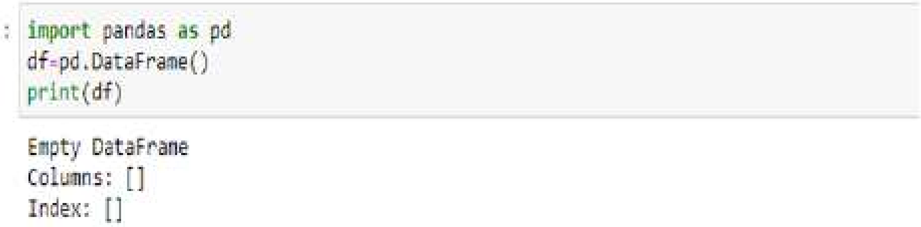
1. A Dataframe has axes (indices)-
   - Row index (axis=0)
   - Column index (axes=1)
2. It is similar to a spreadsheet , whose row index is called index and column index is called column name.
3. A Dataframe contains Heterogeneous data.
4. A Dataframe Size is Mutable.
5. A Dataframe Data is Mutable.



A data frame can be created using any of the following-

1. Series
2. Lists
3. Dictionary
4. A numpy 2D array

How to create Empty Dataframe



```
import pandas as pd
df=pd.DataFrame()
print(df)
```

```
Empty DataFrame
Columns: []
Index: []
```

In [7]:

```
#create dataframe from  series
#dc={'Name':'Ram','Dept':'ECE', 'Marks':500,'Age':23}
df=pd.DataFrame({'Name':['Ram','Raj','abc'],
                 'Dept':['ECE','CSE','me'],
                 'Marks':[500,550,89],
                 'Age':[20,19,23]})
df
```

Out[7]:

	Name	Dept	Marks	Age
0	Ram	ECE	500	20
1	Raj	CSE	550	19
2	abc	me	89	23

2.Creating of Dataframe ,row selction ,column selection

In [8]:

```python
import pandas as pd
data = {
    'ID': ['101', '102', '103', '104'],  # Should be int
    'Price': ['$1,000', '$2,500', '$3,750', '$4,100'],  # Should be float
    'Date': ['2024-01-01', '2024-02-15', '2024-03-20', '2024-04-10'],  # Sho
    'Category': ['fashion', 'Furniture', 'Decor', 'Electricals']  # Should r
}

df=pd.DataFrame(data)
df.head()
```

Out[8]:

	ID	Price	Date	Category
0	101	$1,000	2024-01-01	fashion
1	102	$2,500	2024-02-15	Furniture
2	103	$3,750	2024-03-20	Decor
3	104	$4,100	2024-04-10	Electricals

2.2 Select operation in data frame

To access the column data ,we can mention the column name as subscript.

e.g. - df[Price] This can also be done by using df.Price.

To access multiple columns we can write as df[[col1, col2,---]]

In [9]:

```python
# Access price column using select operator[]
#Note:Colunm name should be exactly same as in dataFrame(Case senstive)
df['Price']
```

Out[9]:

```
0    $1,000
1    $2,500
2    $3,750
3    $4,100
Name: Price, dtype: object
```

In [10]:

```python
#Access category column using '.'(period)
df.Category
```

Out[10]:

```
0        fashion
1      Furniture
2          Decor
3    Electricals
Name: Category, dtype: object
```

3.Basic methods like head(),tail(),info() loc(),iloc(),describe(),dtypes,shape

In [11]:

```
# To check the basic information of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   ID        4 non-null      object
 1   Price     4 non-null      object
 2   Date      4 non-null      object
 3   Category  4 non-null      object
dtypes: object(4)
memory usage: 256.0+ bytes
```

In [12]:

```
#To check the No.of columns and rows in dataset
df.shape # Note: shape is attribute not a method
```

Out[12]:

```
(4, 4)
```

In [13]:

```
#To find the datatypes of all columns(variables) in DataFrame
df.dtypes # Note: dtypes is attribute not a method
```

Out[13]:

```
ID          object
Price       object
Date        object
Category    object
dtype: object
```

In [14]:

```
# to chcek the top 5 rows of dataframe
df.head()
```

Out[14]:

	ID	Price	Date	Category
0	101	$1,000	2024-01-01	fashion
1	102	$2,500	2024-02-15	Furniture
2	103	$3,750	2024-03-20	Decor
3	104	$4,100	2024-04-10	Electricals

In [15]:

```
#To chcek the last 5 rows of the dataframe
df.tail()
```

Out[15]:

	ID	Price	Date	Category
0	101	$1,000	2024-01-01	fashion
1	102	$2,500	2024-02-15	Furniture
2	103	$3,750	2024-03-20	Decor
3	104	$4,100	2024-04-10	Electricals

3. 1 Pandas provide loc() and iloc() methods to access the subset from a data frame using row/column.

Accessing the data frame through loc()

It is used to access a group of rows and columns.

Syntax- Df.loc[StartRow : EndRow, StartColumn : EndColumn]

Note -If we pass : in row or column part then pandas provide the entire rows or columns respectively.

In [16]:

```
#loc method take index for rows and col_names for columns
df.loc[2:5,'Price':'Category']# from 2to5 rows and Price to category column
```

Out[16]:

	Price	Date	Category
2	$3,750	2024-03-20	Decor
3	$4,100	2024-04-10	Electricals

Accessing the data frame through iloc()

It is used to access a group of rows and columns based on numeric index value.

Syntax- Df.loc[StartRowindexs : EndRowindex, StartColumnindex : EndColumnindex]

Note -If we pass : in row or column part then pandas provide the entire rows or columns respectively.

In [17]:

```
df.iloc[1:3,2:4]# from 1 to 3 rows and 2 to 4 columns(index based )
```

Out[17]:

	Date	Category
1	2024-02-15	Furniture
2	2024-03-20	Decor

4.Data Cleaning like type conversions,inconsident data fixing,

In [18]:

```
#convert ID colunm to Integer type
df['ID']=df['ID'].astype('int')
#Covert Price column to float datatype by removing some special charectors
df['Price']=df['Price'].replace({'\$': '',',':''},regex=True).astype(float)
#Convert Data column to Datatime datatype
df['Date']=pd.to_datetime(df['Date'])
#now the datatypes of columns are correctly fixed
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   ID        4 non-null      int32
 1   Price     4 non-null      float64
 2   Date      4 non-null      datetime64[ns]
 3   Category  4 non-null      object
dtypes: datetime64[ns](1), float64(1), int32(1), object(1)
memory usage: 240.0+ bytes
```

In [19]:

```
#to display the basic descriptive statistics of the dataframe for Numerical
df.describe()
```

Out[19]:

	ID	Price
count	4.000000	4.00000
mean	102.500000	2837.50000
std	1.290994	1404.38302
min	101.000000	1000.00000
25%	101.750000	2125.00000
50%	102.500000	3125.00000
75%	103.250000	3837.50000
max	104.000000	4100.00000

5. Concatination of Data Frames

In [20]:

```
df1 = pd.DataFrame({'ID': [1, 2,3,4,5],
                    'Name': ['Ali', 'Bobby','Ramesh','sakshi','sahil'],
                    'Age': [25, 30,23,20,24]})
df2 = pd.DataFrame({'ID': [6,7,8,9],
                    'Name': ['Cherry', 'Mahesh','Preethi','Santosh'],
                    'Age': [35, 30,26,28]})
df1.head()

```

Out[20]:

	ID	Name	Age
0	1	Ali	25
1	2	Bobby	30
2	3	Ramesh	23
3	4	sakshi	20
4	5	sahil	24

In [21]:

```
df2.head()
```

Out[21]:

	ID	Name	Age
0	6	Cherry	35
1	7	Mahesh	30
2	8	Preethi	26
3	9	Santosh	28

In [22]:

```python
#combine 2 datafrmes row wise.
#df1 has 5 rows,df2 has 5 rows after concat total 10 rows
df_cat=pd.concat([df1,df2])# by default axis=0 (row wise)
df_cat

```

Out[22]:

	ID	Name	Age
0	1	Ali	25
1	2	Bobby	30
2	3	Ramesh	23
3	4	sakshi	20
4	5	sahil	24
0	6	Cherry	35
1	7	Mahesh	30
2	8	Preethi	26
3	9	Santosh	28

In [23]:

```python
#combine data frames column wise
#if df1 has 3 columns and df2 has 2 columns after concat total has 5 columns
df3 = pd.DataFrame({'City': ['HYD', 'BEN','VIJ','CHE'],
                    'Salary': [50000, 60000,70000,34000]})
dff=pd.concat([df2,df3],axis=1)#column wise
dff
```

Out[23]:

	ID	Name	Age	City	Salary
0	6	Cherry	35	HYD	50000
1	7	Mahesh	30	BEN	60000
2	8	Preethi	26	VIJ	70000
3	9	Santosh	28	CHE	34000

6.Check for duplicate values and remove them

In [24]:

```
#Create dataframe with employee information
import numpy as np
import pandas as pd
data = {
    "ID": [1, 2, 2, 3, 4, 5, 6, 7, 8, 9],
    "Name": ["Amit", "Priya", "Priya", "Rahul", "Sneha", "Vikram", "Raj", "A
    "Age": [25, 30, 30, 34, -5, 40, 35, 29, 150, 28],
    "Salary (INR)": [500000, 540000, 540000, 620000, 720000, np.nan, 800000,
    "City": ["Mumbai", "Delhi", "Delhi", "Bangalore", "Chennai", "Chennai",
    "Joining Date": ["2022-01-15", "2021-08-20", "2021-08-20", "2020-06-30",
    "Department": ["HR", "Finance", "Finance", "IT", "HR", "HR", "IT", "Fina
}

df_emp=pd.DataFrame(data)
# make of copy of original dataframe
df_cp=df_emp.copy()
```

In [25]:

```
#methos to check no of duplicate records
df_emp.duplicated().sum()

```

Out[25]:

```
1
```

In [26]:

```
#method to remove duplicate records
df_emp=df_emp.drop_duplicates()
df_emp
```

Out[26]:

	ID	Name	Age	Salary (INR)	City	Joining Date	Department
0	1	Amit	25	500000.0	Mumbai	2022-01-15	HR
1	2	Priya	30	540000.0	Delhi	2021-08-20	Finance
3	3	Rahul	34	620000.0	Bangalore	2020-06-30	IT
4	4	Sneha	-5	720000.0	Chennai	2019-11-25	HR
5	5	Vikram	40	NaN	Chennai	2018-03-14	HR
6	6	Raj	35	800000.0	Kolkata	2017-09-10	IT
7	7	Ananya	29	920000.0	Pune	2016-07-04	Finance
8	8	Kiran	150	1030000.0	Hyderabad	2015-05-21	IT
9	9	Neha	28	1140000.0	Ahmedabad	2014-12-11	HR

7. apply() method in python

7.1 apply() method is used to perform any custom function on the dataframe or any part of dataframe like columns

In [27]:

```python
# age column has negative values and also have values
#extreme  values like 150 years which is not correct in general
#lets fix that age values less than 0 and greater than 100 with its median
median_age = df_emp["Age"].median()

def age_correction(x):
    if x <= 0 or x > 100:
        return median_age
    return x

df_emp["Age"] = df_emp["Age"].apply(age_correction)
df_emp
```

Out[27]:

	ID	Name	Age	Salary (INR)	City	Joining Date	Department
0	1	Amit	25.0	500000.0	Mumbai	2022-01-15	HR
1	2	Priya	30.0	540000.0	Delhi	2021-08-20	Finance
3	3	Rahul	34.0	620000.0	Bangalore	2020-06-30	IT
4	4	Sneha	30.0	720000.0	Chennai	2019-11-25	HR
5	5	Vikram	40.0	NaN	Chennai	2018-03-14	HR
6	6	Raj	35.0	800000.0	Kolkata	2017-09-10	IT
7	7	Ananya	29.0	920000.0	Pune	2016-07-04	Finance
8	8	Kiran	30.0	1030000.0	Hyderabad	2015-05-21	IT
9	9	Neha	28.0	1140000.0	Ahmedabad	2014-12-11	HR

In [ ]:

```python

```

8. Check of missing values and handle them

In [28]:

```python
df_emp.isnull().sum()
```

Out[28]:

```
ID              0
Name            0
Age             0
Salary (INR)    1
City            0
Joining Date    0
Department      0
dtype: int64
```

We have 3 ways to handle missing values

1. Removing 2.Filling 3. Imputation

8.1 Remove or drop missing values

If the number of missing values are less when compared to size of dataframe we can remove them.If they are more we have to fill them or else we will lose information

Synatx : DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)

df.dropna():Dropping rows with NaN values (default behavior)

df.dropna(axis=1):Dropping columns with NaN values

df.dropna(subset=col_name): Drop rows based on given column conatins Nan values

df.dropna(how=any/all):Drop if any row contain single Nan or all Nan values

df.dropna(thresh=2): drop rows with more than 2 missing values

In [29]:

```
df1=df_emp.copy()
df2=df_emp.copy()
df3=df_emp.copy()
```

In [30]:

```
#remove missing values
df1=df1.dropna()
df1
```

Out[30]:

	ID	Name	Age	Salary (INR)	City	Joining Date	Department
0	1	Amit	25.0	500000.0	Mumbai	2022-01-15	HR
1	2	Priya	30.0	540000.0	Delhi	2021-08-20	Finance
3	3	Rahul	34.0	620000.0	Bangalore	2020-06-30	IT
4	4	Sneha	30.0	720000.0	Chennai	2019-11-25	HR
6	6	Raj	35.0	800000.0	Kolkata	2017-09-10	IT
7	7	Ananya	29.0	920000.0	Pune	2016-07-04	Finance
8	8	Kiran	30.0	1030000.0	Hyderabad	2015-05-21	IT
9	9	Neha	28.0	1140000.0	Ahmedabad	2014-12-11	HR

8.2 Fill values with 0 or mean or median or mode

if you have more number of missing values we have to fill them,there are 2 ways to fill the missing values

1. Fill with '0'

2. Fill with mean or median or mode

In [31]:

```python
#fill missing values with 0
df2['Salary (INR)']=df2['Salary (INR)'].fillna(0)
df2
```

Out[31]:

	ID	Name	Age	Salary (INR)	City	Joining Date	Department
0	1	Amit	25.0	500000.0	Mumbai	2022-01-15	HR
1	2	Priya	30.0	540000.0	Delhi	2021-08-20	Finance
3	3	Rahul	34.0	620000.0	Bangalore	2020-06-30	IT
4	4	Sneha	30.0	720000.0	Chennai	2019-11-25	HR
5	5	Vikram	40.0	0.0	Chennai	2018-03-14	HR
6	6	Raj	35.0	800000.0	Kolkata	2017-09-10	IT
7	7	Ananya	29.0	920000.0	Pune	2016-07-04	Finance
8	8	Kiran	30.0	1030000.0	Hyderabad	2015-05-21	IT
9	9	Neha	28.0	1140000.0	Ahmedabad	2014-12-11	HR

In [32]:

```python
#filling missing values with Median
df3['Salary (INR)']=df3['Salary (INR)'].fillna(df3['Salary (INR)'].median())
df3
```

Out[32]:

	ID	Name	Age	Salary (INR)	City	Joining Date	Department
0	1	Amit	25.0	500000.0	Mumbai	2022-01-15	HR
1	2	Priya	30.0	540000.0	Delhi	2021-08-20	Finance
3	3	Rahul	34.0	620000.0	Bangalore	2020-06-30	IT
4	4	Sneha	30.0	720000.0	Chennai	2019-11-25	HR
5	5	Vikram	40.0	760000.0	Chennai	2018-03-14	HR
6	6	Raj	35.0	800000.0	Kolkata	2017-09-10	IT
7	7	Ananya	29.0	920000.0	Pune	2016-07-04	Finance
8	8	Kiran	30.0	1030000.0	Hyderabad	2015-05-21	IT
9	9	Neha	28.0	1140000.0	Ahmedabad	2014-12-11	HR

Type *Markdown* and LaTeX: α^2

8. Sorting and Grouping of a dataframe

Take our employee dataset which is stored in df_emp and lets sort the employees and group the employess based on department and find the department wise average salary

In [33]:

```python
#sort the employees from high to low salary
df_sorted=df_emp.sort_values(by='Salary (INR)',ascending=False)
df_sorted
```

Out[33]:

	ID	Name	Age	Salary (INR)	City	Joining Date	Department
9	9	Neha	28.0	1140000.0	Ahmedabad	2014-12-11	HR
8	8	Kiran	30.0	1030000.0	Hyderabad	2015-05-21	IT
7	7	Ananya	29.0	920000.0	Pune	2016-07-04	Finance
6	6	Raj	35.0	800000.0	Kolkata	2017-09-10	IT
4	4	Sneha	30.0	720000.0	Chennai	2019-11-25	HR
3	3	Rahul	34.0	620000.0	Bangalore	2020-06-30	IT
1	2	Priya	30.0	540000.0	Delhi	2021-08-20	Finance
0	1	Amit	25.0	500000.0	Mumbai	2022-01-15	HR
5	5	Vikram	40.0	NaN	Chennai	2018-03-14	HR

Type *Markdown* and LaTeX: α^2

In [34]:

```python
# group the data based on department department wise average salary
df_grp=df_emp.groupby(by='Department')['Salary (INR)'].mean()
df_grp
```

Out[34]:

```
Department
Finance    730000.000000
HR         786666.666667
IT         816666.666667
Name: Salary (INR), dtype: float64
```

In [41]:

```python
#lets check our final cleaned employee data
df3
```

Out[41]:

	ID	Name	Age	Salary (INR)	City	Joining Date	Department
0	1	Amit	25.0	500000.0	Mumbai	2022-01-15	HR
1	2	Priya	30.0	540000.0	Delhi	2021-08-20	Finance
3	3	Rahul	34.0	620000.0	Bangalore	2020-06-30	IT
4	4	Sneha	30.0	720000.0	Chennai	2019-11-25	HR
5	5	Vikram	40.0	760000.0	Chennai	2018-03-14	HR
6	6	Raj	35.0	800000.0	Kolkata	2017-09-10	IT
7	7	Ananya	29.0	920000.0	Pune	2016-07-04	Finance
8	8	Kiran	30.0	1030000.0	Hyderabad	2015-05-21	IT
9	9	Neha	28.0	1140000.0	Ahmedabad	2014-12-11	HR

In [42]:

```
#Save our employee data which cleaned in .csv file
df_emp.to_csv("emp_cleaned.csv", index=False)
# file will save in the current working folder
```

11. Loading of datasets using pandas ,data can be available in diffrent file formates like .csv,.xls,.json...

In [43]:

```
#let us load iris( flower) data set .Make sure file avaible in
#current working folder or else give complete path
import pandas as pd
df_iris=pd.read_csv('iris.csv')
df_iris.head()
```

Out[43]:

	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

In [44]:

```
#Load employee data which is in excel format
emp=pd.read_excel('sample-staff.xlsx')
emp.head()

```

Out[44]:

	Emp ID	Name	Gender	Department	Salary	Start Date	FTE	Employee type	Work location
0	PR00147	Minerva Ricardot	Male	???	120000.00	12-Nov-18	1.0	Permanent	Remote
1	PR04686	Oona Donan	Female	Business Development	98000.00	2019-09-02 00:00:00	0.9	Permanent	Seattle, USA
2	SQ04612	Mick Spraberry	Female	Services	120000.00	2020-03-12 00:00:00	0.9	Permanent	Remote
3	VT01803	Freddy Linford	Female	Training	93128.34	Mar 5, 2018	1.0	Fixed Term	Seattle, USA
4	TN02749	Parasuramudu Jamakayala	Female	Training	57002.02	2-Apr-18	0.7	Permanent	Hyderabad, India

In [45]:

```python
#let us load the json data using pandas
df_js=pd.read_json('sample.json')
df_js.head()
```

Out[45]:

	OrderID	CustomerName	ProductName	Category	Quantity	UnitPrice	TotalPrice	OrderDate	
0	1001	Rajesh Sharma	Smartphone	Electronics	1	20000	20000	2023-10-01	
1	1002	Priya Mehra	Laptop	Electronics	1	45000	45000	2023-10-05	
2	1003	Amit Trivedi	Office Chair	Furniture	2	3000	6000	2023-10-08	Be
3	1004	Sneha Iyer	Running Shoes	Sports	1	3500	3500	2023-10-10	(
4	1005	Deepak Joshi	Refrigerator	Appliances	1	18000	18000	2023-10-12	

11.2 Loading of existing or inbuilt datasets using seaborn

Seaborn has inbuilt datasets for practice let us see how to load that kind of datasets like iris,tips,titanic,flights,mpg,penguins,taxis etc..

In [46]:

```python
import seaborn as sns
dt=sns.load_dataset('tips')# instead of tips you can use any other datasets
dt.head()
```

```
c:\users\vamsi2001\appdata\local\programs\python\python39\lib\site-packages\sci
py\__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is requir
ed for this version of SciPy (detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Out[46]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Pandas Excerice (DIY)

Type *Markdown* and LaTeX: α^2

In [ ]:

```
```