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

a = np.array([[1,2,3],[4,5,6]])
print("2D array: \n",a)
```

```
2D array:
 [[1 2 3]
 [4 5 6]]
```

In [8]:
```
a1 = np.array((1,2,3))
print("2D array tuple: \n",a1)
print("type of a1: ",type(a1))
```

```
2D array tuple:
 [1 2 3]
type of a1:  <class 'numpy.ndarray'>
```

In [12]:
```
a2 = np.arange(20).reshape(5,4)
print("Multiple array using arange module: \n",a2)
```

```
Multiple array using arange module:
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]]
```

In [13]:
```
a3 = np.random.randint(20,35,16).reshape(4,-1)
print("matrix: \n",a3)
```

```
matrix:
 [[23 26 21 21]
 [23 32 20 31]
 [26 25 20 30]
 [23 30 30 26]]
```

Pandas Module:

- it is one of the most important modules in data science
- Used to work with data in the form of frames/table
- Stands for Panel Data
- used for:
  - Data analysis, data manipulation and data cleaning
  - Data analysis: Analyzing the data
    - that means the stastics of the data,Volume of the data etc...
  - Data Manipulation: Transforming the data from one format to another format
    - like Merging,Concatenating etc....
  - Data Cleaning: Removing the Null Data.
    - Null Data means NaN(Not a Number)
- Data structures in pandas

1. Series:
   - sequentuial data
2. Data frame:
   - Data in tabular format i.e; rows and columns

- **Installation**
  - pip install pandas
  - import pandas

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

In [13]:
```
dic = {1:"mounav",2:"ajay",3:"charan",4:"rizwanullah"}
data = {'Name':['mounav', 'ajay', 'charan', 'rizwanullah'],'Roll':[291, '2A1', 26
d1 = pd.DataFrame(data)
print("Tabular data: \n",d1)
```

```
Tabular data:
           Name Roll
0       mounav  291
1         ajay  2A1
2       charan  268
3  rizwanullah  299
```

In [14]:
```
pd.__version__
```

Out[14]:
```
'1.2.4'
```

In [15]:
```
# we can create series using str,tuple,list, and dict

st = "I am a Student of Vresec"
r = pd.Series(st)
print("Series of string: ",r)
```

```
Series of string:  0    I am a Student of Vresec
dtype: object
```

In [16]:
```
# try with input().split()

li = input("enter a statement: ").split()
print("Result: \n",pd.Series(li))
```

```
enter a statement: hello my name is M.sai
Result:
 0    hello
1       my
2     name
3       is
4    M.sai
dtype: object
```

In [17]:
```
# using tuple
print("Result: \n",pd.Series(range(10,30,6)))
# print("Result_1: \n",pd.Series((1,2,3,4))
```

```
Result:
 0    10
1    16
2    22
3    28
dtype: int64
```

In [18]:
```
# Using Dictionary
k = {'34':"thirty Four", 12:90, 'tuple':(9,8.2,'sai'), 'list':[1,2,3,4]}
s = pd.Series(k)
print("Tabular form: \n\n",s)
print("index values: ",s.index)
```

```
Tabular form:

 34         thirty Four
12                   90
tuple    (9, 8.2, sai)
list      [1, 2, 3, 4]
dtype: object
index values:  Index(['34', 12, 'tuple', 'list'], dtype='object')
```

In [19]:
```
# creating series using numpy arrays
import numpy as np

ar = np.arange(10,40,5)
print("array: ",ar)

ar1 = pd.Series(ar)
print("index values also: \n",ar1)
```

```
array:  [10 15 20 25 30 35]
index values also:
 0    10
1    15
2    20
3    25
4    30
5    35
dtype: int32
```

In [20]:
```
# modifying index values
print("index data: ",ar1.index)
```

```
index data:  RangeIndex(start=0, stop=6, step=1)
```

In [21]:
```
ar1.index = [4,5,6,7,9,1]
print("user index: \n",ar1)
```

```
user index:
 4    10
5    15
6    20
7    25
9    30
1    35
dtype: int32
```

In [22]:
```
li2 = [9,3,4,5,6,'hi']
inx = [num for num in range(5,11)]
pd.Series(li2,index=inx)
```

Out[22]:
```
5      9
6      3
7      4
8      5
9      6
10    hi
dtype: object
```

In [23]:
```
# passing the user index to series created by dict
print("dictionary: ",dic,"\n length od dictionary: ",len(dic))
print()

ix = [num for num in range(10,14)]
e = pd.Series(dic,index = ix)
print("table: \n",e)
```

```
dictionary:  {1: 'mounav', 2: 'ajay', 3: 'charan', 4: 'rizwanullah'}
 length od dictionary:  4

table:
 10    NaN
11    NaN
12    NaN
13    NaN
dtype: object
```

In [24]:
```
s.index
```

Out[24]:
```
Index(['34', 12, 'tuple', 'list'], dtype='object')
```

In [25]:
```
s['tuple']
```

Out[25]:
```
(9, 8.2, 'sai')
```

Data Frames:

- Data is in 2D Format

In [26]:
```
# dictionary into dataframe
print("data dictionary: ",data)
pd.DataFrame(data)
```

```
data dictionary:  {'Name': ['mounav', 'ajay', 'charan', 'rizwanullah'], 'Roll':
[291, '2A1', 268, 299]}
```

Out[26]:

	Name	Roll
0	mounav	291
1	ajay	2A1
2	charan	268
3	rizwanullah	299

In [27]:
```
s
```

Out[27]:
```
34          thirty Four
12                   90
tuple    (9, 8.2, sai)
list      [1, 2, 3, 4]
dtype: object
```

In [28]:
```
df = pd.DataFrame(s)
print("Tabular Form: \n",df)
```

```
Tabular Form:
                     0
34          thirty Four
12                   90
tuple    (9, 8.2, sai)
list      [1, 2, 3, 4]
```

In [38]:
```
# pd.DataFrame(k)
```

In [30]:
```
k
```

Out[30]:
```
{'34': 'thirty Four', 12: 90, 'tuple': (9, 8.2, 'sai'), 'list': [1, 2, 3, 4]}
```

Concatination Of 2 Data-Frames:

- the 2 data-frames must and should have same size or dimensions

In [31]:
```
des = {'int':89,'float':90.67,'str':'M.sai','prgm':'apssdc'}
pd.DataFrame(des,index = [1,2,3])
```

Out[31]:

	int	float	str	prgm
1	89	90.67	M.sai	apssdc
2	89	90.67	M.sai	apssdc
3	89	90.67	M.sai	apssdc

In [32]:
```
df2 = {'friend': ['prabhuram', 'saiesh', 'manohar', 'srinivas'], 'admin_no': [288
df4 = pd.DataFrame(df2)
df4
```

Out[32]:

	friend	admin_no
0	prabhuram	288
1	saiesh	2C7
2	manohar	295
3	srinivas	2A4

In [33]:
```
df3 = pd.DataFrame(data)
df3
```

Out[33]:

	Name	Roll
0	mounav	291
1	ajay	2A1
2	charan	268
3	rizwanullah	299

In [34]:
```
# Concatining 2 Data Frames
pd.concat([df3,df4]) # df3 and df4 should have same size
```

Out[34]:

	Name	Roll	friend	admin_no
0	mounav	291	NaN	NaN
1	ajay	2A1	NaN	NaN
2	charan	268	NaN	NaN
3	rizwanullah	299	NaN	NaN
0	NaN	NaN	prabhuram	288
1	NaN	NaN	saiesh	2C7
2	NaN	NaN	manohar	295
3	NaN	NaN	srinivas	2A4

In [35]:
```
# import os
# os.system("shutdown /s")
```

Merging 2 Data-Frames

In [36]:
```
# merging 2 data frames

c = pd.DataFrame({'Int_rate':[2,1,2,3], 'Int_gdp':[50,45,45,67]}, index = [2041,2
print("result_1: \n\n",c)
```

```
result_1:

       Int_rate  Int_gdp
2041         2       50
2002         1       45
2003         2       45
2204         3       67
```

In [37]:
```
u = pd.DataFrame({'low_tier_hpi':[80,90,70,60], 'Unemployement':[1,3,5,6]}, index
print("result_2: \n\n",u)
```

```
result_2:

       low_tier_hpi  Unemployement
2001            80              1
2003            90              3
2003            70              5
2005            60              6
```

In [44]:
```
# joining the data-frames
res = c.join(u)
print("Joining the data- frames: \n\n",res)
```

```
Joining the data- frames:

       Int_rate  Int_gdp  low_tier_hpi  Unemployement
2002         1       45           NaN            NaN
2003         2       45          90.0            3.0
2003         2       45          70.0            5.0
2041         2       50           NaN            NaN
2204         3       67           NaN            NaN
```

In [45]:
```
# merging the data-frames having similar column values

m = pd.DataFrame({'a':[4,5,6,7], 'b':[30,50,60,70], 'c':['a','b','c','d']}, inde
print("Data-Frame: \n\n",m)
m
```

```
Data-Frame:

         a   b  c
first    4  30  a
second   5  50  b
third    6  60  c
fourth   7  70  d
```

Out[45]:

	a	b	c
first	4	30	a
second	5	50	b
third	6	60	c
fourth	7	70	d

In [47]:
```
n = pd.DataFrame({'a':[4,9,6,7], 'b':[30,80,60,70], 'c':['a','b','c','d']}, inde
new = m.merge(n)
print("merged frame: \n\n",new)
m.merge(n)
```

```
merged frame:

    a   b  c
0   4  30  a
1   6  60  c
2   7  70  d
```

Out[47]:

	a	b	c
0	4	30	a
1	6	60	c
2	7	70	d

In [50]:
```
new.columns = ['f','s','th'] # renameing the column values
new.rows = [5,6,7] # updating the row values
print("modifyed frame: \n\n",new)
new
```

```
modifyed frame:

    f   s th
5   4  30  a
6   6  60  c
7   7  70  d

<ipython-input-50-84e120910af3>:2: UserWarning: Pandas doesn't allow columns to
be created via a new attribute name - see https://pandas.pydata.org/pandas-doc
s/stable/indexing.html#attribute-access (https://pandas.pydata.org/pandas-docs/
stable/indexing.html#attribute-access)
  new.rows = [5,6,7] # updating the row values
```

Out[50]:

	f	s	th
5	4	30	a
6	6	60	c
7	7	70	d

In [51]:
```
new.columns = ['f','s','th'] # renameing the column values
new.index = [5,6,7] # updating the row values
print("modifyed frame: \n\n",new)
new
```

```
modifyed frame:

    f   s th
5   4  30  a
6   6  60  c
7   7  70  d
```

Out[51]:

	f	s	th
5	4	30	a
6	6	60	c
7	7	70	d

In [54]:
```
# merging the dataframes based on particular column value

h = m.merge(n, on = 'c')
print("particular coulmn: \n\n",h)

m.merge(n, on = 'c')
```

```
particular coulmn:

    a_x  b_x  c  a_y  b_y
0    4   30  a    4   30
1    5   50  b    9   80
2    6   60  c    6   60
3    7   70  d    7   70
```

Out[54]:

	a_x	b_x	c	a_y	b_y
0	4	30	a	4	30
1	5	50	b	9	80
2	6	60	c	6	60
3	7	70	d	7	70

In [64]:
```
# data range

dates = pd.date_range('2022-01-01', '2022-01-10')
s3 = pd.Series(dates)
print("range of dates: \n",s3)
pd.Series(dates)
```

```
range of dates:
 0   2022-01-01
1   2022-01-02
2   2022-01-03
3   2022-01-04
4   2022-01-05
5   2022-01-06
6   2022-01-07
7   2022-01-08
8   2022-01-09
9   2022-01-10
dtype: datetime64[ns]
```

Out[64]:
```
0   2022-01-01
1   2022-01-02
2   2022-01-03
3   2022-01-04
4   2022-01-05
5   2022-01-06
6   2022-01-07
7   2022-01-08
8   2022-01-09
9   2022-01-10
dtype: datetime64[ns]
```

In [65]:
```
dt = pd.date_range('2022-01-10', periods = 5)
s1 = pd.Series(dt)
print("series of calender dates: \n",s1)

pd.Series(dt)
```

```
series of calender dates:
 0   2022-01-10
1   2022-01-11
2   2022-01-12
3   2022-01-13
4   2022-01-14
dtype: datetime64[ns]
```

Out[65]:
```
0   2022-01-10
1   2022-01-11
2   2022-01-12
3   2022-01-13
4   2022-01-14
dtype: datetime64[ns]
```

In [66]:
```
temp = [30,32,35,40,31]
s2 = pd.Series(temp, index = dt)
print("temeratures on days: \n\n",s2)

pd.Series(temp, index = dt)
```

```
temeratures on days:

 2022-01-10    30
2022-01-11    32
2022-01-12    35
2022-01-13    40
2022-01-14    31
Freq: D, dtype: int64
```

Out[66]:
```
2022-01-10    30
2022-01-11    32
2022-01-12    35
2022-01-13    40
2022-01-14    31
Freq: D, dtype: int64
```

Working with .csv file:

- reading .csv located at same location
- Syntax:
  - pd.read_csv('filename')

In [67]:
```
dir(pd) # pd ===> pandas
```

```
 'plotting',
 'qcut',
 'read_clipboard',
 'read_csv',
 'read_excel',
 'read_feather',
 'read_fwf',
 'read_gbq',
 'read_hdf',
 'read_html',
 'read_json',
 'read_orc',
 'read_parquet',
 'read_pickle',
 'read_sas',
 'read_spss',
 'read_sql',
 'read_sql_query',
 'read_sql_table',
 'read_stata'
```

In [71]:
```
marks = pd.read_csv('marks.csv')
marks
```

Out[71]:

	Roll	Name	Maths	Discreate	OS	CO	DS
0	299	Rizwan	97	100	92.00	95	100
1	2A0	Sai	89	91	93.00	89	99
2	295	manohar	88	71	93.00	49	69
3	291	mounav	85	94	93.00	85	98
4	2A1	ajay	83	98	99.00	99	100
5	2C7	saiesh	99	98	93.00	100	96
6	288	prabhuram	86	92	90.00	88	99
7	268	Charan	100	100	100.00	100	100
8	293	Uday	79	81	92.00	99	59
9	2C8	Yesu	99	99	99.99	100	99
10	2B2	Shanak	89	91	93.00	89	99

In [72]:
```
print("Student marks Table: \n\n",marks)
```

```
Student marks Table:

    Roll       Name  Maths  Discreate   OS      CO   DS
0   299     Rizwan     97         100   92.00   95  100
1   2A0        Sai     89          91   93.00   89   99
2   295    manohar     88          71   93.00   49   69
3   291     mounav     85          94   93.00   85   98
4   2A1       ajay     83          98   99.00   99  100
5   2C7     saiesh     99          98   93.00  100   96
6   288  prabhuram     86          92   90.00   88   99
7   268     Charan    100         100  100.00  100  100
8   293       Uday     79          81   92.00   99   59
9   2C8       Yesu     99          99   99.99  100   99
10  2B2     Shanak     89          91   93.00   89   99
```

In [76]:
```
# Data-Processing
# Select 5 Randon Sample
marks.sample() # Random sample in our data
```

Out[76]:

	Roll	Name	Maths	Discreate	OS	CO	DS
5	2C7	saiesh	99	98	93.0	100	96

In [78]:
```
marks.sample(5) # generates 5 rows with student marks
```

Out[78]:

	Roll	Name	Maths	Discreate	OS	CO	DS
7	268	Charan	100	100	100.0	100	100
6	288	prabhuram	86	92	90.0	88	99
2	295	manohar	88	71	93.0	49	69
1	2A0	Sai	89	91	93.0	89	99
4	2A1	ajay	83	98	99.0	99	100

In [86]:
```
s4 = marks.head()
print("top of the rows: \n",s4)
print("\ntop 3 rows or samples: \n",marks.head(3))

marks.head()
```

```
top of the rows:
   Roll     Name  Maths  Discreate  OS     CO   DS
0  299   Rizwan     97        100   92.0  95  100
1  2A0      Sai     89         91   93.0  89   99
2  295  manohar     88         71   93.0  49   69
3  291   mounav     85         94   93.0  85   98
4  2A1     ajay     83         98   99.0  99  100

top 3 rows or samples:
   Roll     Name  Maths  Discreate  OS     CO   DS
0  299   Rizwan     97        100   92.0  95  100
1  2A0      Sai     89         91   93.0  89   99
2  295  manohar     88         71   93.0  49   69
```

Out[86]:

	Roll	Name	Maths	Discreate	OS	CO	DS
0	299	Rizwan	97	100	92.0	95	100
1	2A0	Sai	89	91	93.0	89	99
2	295	manohar	88	71	93.0	49	69
3	291	mounav	85	94	93.0	85	98
4	2A1	ajay	83	98	99.0	99	100

In [87]:
```
marks.head(3)
```

Out[87]:

	Roll	Name	Maths	Discreate	OS	CO	DS
0	299	Rizwan	97	100	92.0	95	100
1	2A0	Sai	89	91	93.0	89	99
2	295	manohar	88	71	93.0	49	69

In [84]:
```
s5 = marks.tail()
print("last rows: \n",s5)
print("\nlast 3 rows or samples: \n",marks.tail(3))

marks.tail()
```

```
last rows:
    Roll       Name  Maths  Discreate   OS      CO   DS
6   288  prabhuram     86         92   90.00   88   99
7   268     Charan    100        100  100.00  100  100
8   293       Uday     79         81   92.00   99   59
9   2C8       Yesu     99         99   99.99  100   99
10  2B2     Shanak     89         91   93.00   89   99

last 3 rows or samples:
    Roll    Name  Maths  Discreate  OS      CO  DS
8   293    Uday     79         81  92.00   99  59
9   2C8    Yesu     99         99  99.99  100  99
10  2B2  Shanak     89         91  93.00   89  99
```

Out[84]:

	Roll	Name	Maths	Discreate	OS	CO	DS
6	288	prabhuram	86	92	90.00	88	99
7	268	Charan	100	100	100.00	100	100
8	293	Uday	79	81	92.00	99	59
9	2C8	Yesu	99	99	99.99	100	99
10	2B2	Shanak	89	91	93.00	89	99

In [85]:
```
marks.tail(3)
```

Out[85]:

	Roll	Name	Maths	Discreate	OS	CO	DS
8	293	Uday	79	81	92.00	99	59
9	2C8	Yesu	99	99	99.99	100	99
10	2B2	Shanak	89	91	93.00	89	99

In [89]:
```
marks['Roll'] # getting particular column
```

```
0     299
1     2A0
2     295
3     291
4     2A1
5     2C7
6     288
7     268
8     293
9     2C8
10    2B2
Name: Roll, dtype: object
```

In [90]:
```
marks[['Roll','Name','CO']]
```

Out[90]:

	Roll	Name	CO
0	299	Rizwan	95
1	2A0	Sai	89
2	295	manohar	49
3	291	mounav	85
4	2A1	ajay	99
5	2C7	saiesh	100
6	288	prabhuram	88
7	268	Charan	100
8	293	Uday	99
9	2C8	Yesu	100
10	2B2	Shanak	89

In [92]:
```
help(pd)
```

```
Help on package pandas:

NAME
    pandas

DESCRIPTION
    pandas - a powerful data analysis and manipulation library for Python
    =====================================================================

    **pandas** is a Python package providing fast, flexible, and expressive dat
a
    structures designed to make working with "relational" or "labeled" data bot
h
    easy and intuitive. It aims to be the fundamental high-level building block
for
    doing practical, **real world** data analysis in Python. Additionally, it h
as
    the broader goal of becoming **the most powerful and flexible open source d
ata
    analysis / manipulation tool available in any language**. It is already wel
l on
    its way toward this goal.

    Main Features
    -------------
    Here are just a few of the things that pandas does well:

      - Easy handling of missing data in floating point as well as non-floating
        point data.
      - Size mutability: columns can be inserted and deleted from DataFrame and
        higher dimensional objects
      - Automatic and explicit data alignment: objects can be explicitly aligne
d
        to a set of labels, or the user can simply ignore the labels and let
        `Series`, `DataFrame`, etc. automatically align the data for you in
        computations.
      - Powerful, flexible group by functionality to perform split-apply-combin
e
        operations on data sets, for both aggregating and transforming data.
      - Make it easy to convert ragged, differently-indexed data in other Pytho
n
        and NumPy data structures into DataFrame objects.
      - Intelligent label-based slicing, fancy indexing, and subsetting of larg
e
        data sets.
      - Intuitive merging and joining data sets.
      - Flexible reshaping and pivoting of data sets.
      - Hierarchical labeling of axes (possible to have multiple labels per tic
k).
      - Robust IO tools for loading data from flat files (CSV and delimited),
        Excel files, databases, and saving/loading data from the ultrafast HDF5
        format.
      - Time series-specific functionality: date range generation and frequency
        conversion, moving window statistics, date shifting and lagging.
```

```
PACKAGE CONTENTS
    _config (package)
    _libs (package)
    _testing
    _typing
    _version
    api (package)
    arrays (package)
    compat (package)
    conftest
    core (package)
    errors (package)
    io (package)
    plotting (package)
    testing
    tests (package)
    tseries (package)
    util (package)

SUBMODULES
    _hashtable
    _lib
    _tslib
    offsets

FUNCTIONS
    __getattr__(name)
        # GH 27101

DATA
    IndexSlice = <pandas.core.indexing._IndexSlice object>
    NA = <NA>
    NaT = NaT
    __docformat__ = 'restructuredtext'
    __git_version__ = '2cb96529396d93b46abab7bbc73a208e708c642e'
    describe_option = <pandas._config.config.CallableDynamicDoc object>
    get_option = <pandas._config.config.CallableDynamicDoc object>
    options = <pandas._config.config.DictWrapper object>
    reset_option = <pandas._config.config.CallableDynamicDoc object>
    set_option = <pandas._config.config.CallableDynamicDoc object>

VERSION
    1.2.4

FILE
    c:\programdata\anaconda3\lib\site-packages\pandas\__init__.py
```

In [93]:
```
help(pd.api)
```

```
Help on package pandas.api in pandas:

NAME
    pandas.api - public toolkit API

PACKAGE CONTENTS
    extensions (package)
    indexers (package)
    types (package)

FILE
    c:\programdata\anaconda3\lib\site-packages\pandas\api\__init__.py
```

In [97]:
```
new1 = pd.read_csv('marks.csv', usecols = ['Name','OS']) # getting particular col
print(new1)
new1
```

```
         Name      OS
0      Rizwan   92.00
1         Sai   93.00
2     manohar   93.00
3      mounav   93.00
4        ajay   99.00
5      saiesh   93.00
6   prabhuram   90.00
7      Charan  100.00
8        Uday   92.00
9        Yesu   99.99
10     Shanak   93.00
```

Out[97]:

	Name	OS
0	Rizwan	92.00
1	Sai	93.00
2	manohar	93.00
3	mounav	93.00
4	ajay	99.00
5	saiesh	93.00
6	prabhuram	90.00
7	Charan	100.00
8	Uday	92.00
9	Yesu	99.99
10	Shanak	93.00

In [98]:
```
marks.columns
```

Out[98]:
```
Index(['Roll', 'Name', 'Maths', 'Discreate', 'OS   ', 'CO', 'DS'], dtype='objec
t')
```

In [100]:
```
marks.Roll
```

Out[100]:
```
0     299
1     2A0
2     295
3     291
4     2A1
5     2C7
6     288
7     268
8     293
9     2C8
10    2B2
Name: Roll, dtype: object
```

In [101]:
```
marks.index
```

Out[101]:
```
RangeIndex(start=0, stop=11, step=1)
```

In [103]:
```
marks.dtypes # data types of each column
```

Out[103]:
```
Roll          object
Name          object
Maths          int64
Discreate      int64
OS           float64
CO             int64
DS             int64
dtype: object
```

In [104]:
```
marks.info # textual information
```

Out[104]:
```
<bound method DataFrame.info of    Roll       Name  Maths  Discreate   OS
CO   DS
0   299     Rizwan     97        100   92.00   95  100
1   2A0        Sai     89         91   93.00   89   99
2   295    manohar     88         71   93.00   49   69
3   291     mounav     85         94   93.00   85   98
4   2A1       ajay     83         98   99.00   99  100
5   2C7     saiesh     99         98   93.00  100   96
6   288  prabhuram     86         92   90.00   88   99
7   268     Charan    100        100  100.00  100  100
8   293       Uday     79         81   92.00   99   59
9   2C8       Yesu     99         99   99.99  100   99
10  2B2     Shanak     89         91   93.00   89   99>
```

In [108]:
```
w = marks.describe()
print("statistics of the existing data or file: \n\n",w)

marks.describe() # statistics of the data
```

```
statistics of the existing data or file:

            Maths   Discreate       OS             CO          DS
count   11.000000   11.000000   11.000000   11.000000   11.000000
mean    90.363636   92.272727   94.362727   90.272727   92.545455
std      7.256345    9.034278    3.527504   14.826267   14.334320
min     79.000000   71.000000   90.000000   49.000000   59.000000
25%     85.500000   91.000000   92.500000   88.500000   97.000000
50%     89.000000   94.000000   93.000000   95.000000   99.000000
75%     98.000000   98.500000   96.000000   99.500000   99.500000
max    100.000000  100.000000  100.000000  100.000000  100.000000
```

Out[108]:

	Maths	Discreate	OS	CO	DS
count	11.000000	11.000000	11.000000	11.000000	11.000000
mean	90.363636	92.272727	94.362727	90.272727	92.545455
std	7.256345	9.034278	3.527504	14.826267	14.334320
min	79.000000	71.000000	90.000000	49.000000	59.000000
25%	85.500000	91.000000	92.500000	88.500000	97.000000
50%	89.000000	94.000000	93.000000	95.000000	99.000000
75%	98.000000	98.500000	96.000000	99.500000	99.500000
max	100.000000	100.000000	100.000000	100.000000	100.000000

In [113]:
```
help(marks.describe)
```

```
    same as the median.

    For object data (e.g. strings or timestamps), the result's index
    will include ``count``, ``unique``, ``top``, and ``freq``. The ``top``
    is the most common value. The ``freq`` is the most common value's
    frequency. Timestamps also include the ``first`` and ``last`` items.

    If multiple object values have the highest count, then the
    ``count`` and ``top`` results will be arbitrarily chosen from
    among those with the highest count.

    For mixed data types provided via a ``DataFrame``, the default is to
    return only an analysis of numeric columns. If the dataframe consists
    only of object and categorical data without any numeric columns, the
    default is to return an analysis of both the object and categorical
    columns. If ``include='all'`` is provided as an option, the result
    will include a union of attributes of each type.

    The `include` and `exclude` parameters can be used to limit
    which columns in a ``DataFrame`` are analyzed for the output.
```

In [112]:
```
marks.Maths.value_counts()  # Maths is a subject name
```

Out[112]:
```
89     2
99     2
97     1
83     1
100    1
85     1
86     1
88     1
79     1
Name: Maths, dtype: int64
```

In [125]:
```
marks['Discreate'].value_counts()
```

Out[125]:
```
98     2
100    2
91     2
81     1
99     1
71     1
92     1
94     1
Name: Discreate, dtype: int64
```

In [126]:
```
marks['OS'].value_counts()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_l
oc(self, key, method, tolerance)
   3079             try:
-> 3080                 return self._engine.get_loc(casted_key)
   3081             except KeyError as err:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashT
able.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashT
able.get_item()

KeyError: 'OS'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
<ipython-input-126-f120debf5fa0> in <module>
----> 1 marks['OS'].value_counts()

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__
(self, key)
   3022             if self.columns.nlevels > 1:
   3023                 return self._getitem_multilevel(key)
-> 3024             indexer = self.columns.get_loc(key)
   3025             if is_integer(indexer):
   3026                 indexer = [indexer]

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_l
oc(self, key, method, tolerance)
   3080                 return self._engine.get_loc(casted_key)
   3081             except KeyError as err:
-> 3082                 raise KeyError(key) from err
   3083
   3084         if tolerance is not None:

KeyError: 'OS'
```

In [127]:
```
marks.sort_values('Roll',ascending = False)
```

Out[127]:

	Roll	Name	Maths	Discreate	OS	CO	DS
9	2C8	Yesu	99	99	99.99	100	99
5	2C7	saiesh	99	98	93.00	100	96
10	2B2	Shanak	89	91	93.00	89	99
4	2A1	ajay	83	98	99.00	99	100
1	2A0	Sai	89	91	93.00	89	99
0	299	Rizwan	97	100	92.00	95	100
2	295	manohar	88	71	93.00	49	69
8	293	Uday	79	81	92.00	99	59
3	291	mounav	85	94	93.00	85	98
6	288	prabhuram	86	92	90.00	88	99
7	268	Charan	100	100	100.00	100	100

In [128]:
```
marks.sort_values('Roll',ascending = True)
```

Out[128]:

	Roll	Name	Maths	Discreate	OS	CO	DS
7	268	Charan	100	100	100.00	100	100
6	288	prabhuram	86	92	90.00	88	99
3	291	mounav	85	94	93.00	85	98
8	293	Uday	79	81	92.00	99	59
2	295	manohar	88	71	93.00	49	69
0	299	Rizwan	97	100	92.00	95	100
1	2A0	Sai	89	91	93.00	89	99
4	2A1	ajay	83	98	99.00	99	100
10	2B2	Shanak	89	91	93.00	89	99
5	2C7	saiesh	99	98	93.00	100	96
9	2C8	Yesu	99	99	99.99	100	99

In [130]:
```
marks.sort_values('DS',ascending = False)
```

Out[130]:

	Roll	Name	Maths	Discreate	OS	CO	DS
0	299	Rizwan	97	100	92.00	95	100
4	2A1	ajay	83	98	99.00	99	100
7	268	Charan	100	100	100.00	100	100
1	2A0	Sai	89	91	93.00	89	99
6	288	prabhuram	86	92	90.00	88	99
9	2C8	Yesu	99	99	99.99	100	99
10	2B2	Shanak	89	91	93.00	89	99
3	291	mounav	85	94	93.00	85	98
5	2C7	saiesh	99	98	93.00	100	96
2	295	manohar	88	71	93.00	49	69
8	293	Uday	79	81	92.00	99	59

In [138]:
```
# adding new col into data-frame

marks['Total'] = 100 # modified only in  program
marks
```

Out[138]:

	Roll	Name	Maths	Discreate	OS	CO	DS	Total
0	299	Rizwan	97	100	92.00	95	100	100
1	2A0	Sai	89	91	93.00	89	99	100
2	295	manohar	88	71	93.00	49	69	100
3	291	mounav	85	94	93.00	85	98	100
4	2A1	ajay	83	98	99.00	99	100	100
5	2C7	saiesh	99	98	93.00	100	96	100
6	288	prabhuram	86	92	90.00	88	99	100
7	268	Charan	100	100	100.00	100	100	100
8	293	Uday	79	81	92.00	99	59	100
9	2C8	Yesu	99	99	99.99	100	99	100
10	2B2	Shanak	89	91	93.00	89	99	100

In [135]:
```
marks.Maths.sum() # summation of one column data
```

Out[135]: 994

In [136]:
```
marks.sum()
```

Out[136]:
```
Roll                                2992A02952912A12C72882682932C82B2
Name        RizwanSaimanoharmounavajaysaieshprabhuramChara...
Maths                                                         994
Discreate                                                    1015
OS                                                        1037.99
CO                                                            993
DS                                                           1018
Total                                                        1100
dtype: object
```

In [139]:
```
marks.Discreate.sum()
```

Out[139]:
```
1015
```

In [140]:
```
marks.isnull() # checking for the null values
```

Out[140]:

	Roll	Name	Maths	Discreate	OS	CO	DS	Total
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False
10	False	False	False	False	False	False	False	False

In [141]:
```
marks.isna() # same as isnull()
```

Out[141]:

	Roll	Name	Maths	Discreate	OS	CO	DS	Total
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False
10	False	False	False	False	False	False	False	False

In [142]:
```
# Finding the sum/count of Null values in our Data

marks.isnull().sum()
```

Out[142]:
```
Roll         0
Name         0
Maths        0
Discreate    0
OS           0
CO           0
DS           0
Total        0
dtype: int64
```

In [146]:
```
# finding the no.of unique values under each col

marks.nunique()
```

Out[146]:
```
Roll         11
Name         11
Maths         9
Discreate     8
OS            6
CO            7
DS            6
Total         1
dtype: int64
```

In [144]:
```
# Cleaning of the Data

clean = marks.dropna()
print("new data frame without nulls: \n\n",clean)

clean
```

```
new data frame without nulls:

    Roll       Name  Maths  Discreate   OS      CO   DS  Total
0   299     Rizwan     97        100   92.00   95  100    100
1   2A0        Sai     89         91   93.00   89   99    100
2   295    manohar     88         71   93.00   49   69    100
3   291     mounav     85         94   93.00   85   98    100
4   2A1       ajay     83         98   99.00   99  100    100
5   2C7     saiesh     99         98   93.00  100   96    100
6   288  prabhuram     86         92   90.00   88   99    100
7   268     Charan    100        100  100.00  100  100    100
8   293       Uday     79         81   92.00   99   59    100
9   2C8       Yesu     99         99   99.99  100   99    100
10  2B2     Shanak     89         91   93.00   89   99    100
```

Out[144]:

	Roll	Name	Maths	Discreate	OS	CO	DS	Total
0	299	Rizwan	97	100	92.00	95	100	100
1	2A0	Sai	89	91	93.00	89	99	100
2	295	manohar	88	71	93.00	49	69	100
3	291	mounav	85	94	93.00	85	98	100
4	2A1	ajay	83	98	99.00	99	100	100
5	2C7	saiesh	99	98	93.00	100	96	100
6	288	prabhuram	86	92	90.00	88	99	100
7	268	Charan	100	100	100.00	100	100	100
8	293	Uday	79	81	92.00	99	59	100
9	2C8	Yesu	99	99	99.99	100	99	100
10	2B2	Shanak	89	91	93.00	89	99	100

In [147]:
```
from sklearn.datasets import load_iris

d3 = load_iris()
d3
```

Out[147]:
```
{'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3. , 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5. , 3.6, 1.4, 0.2],
        [5.4, 3.9, 1.7, 0.4],
        [4.6, 3.4, 1.4, 0.3],
        [5. , 3.4, 1.5, 0.2],
        [4.4, 2.9, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.1],
        [5.4, 3.7, 1.5, 0.2],
        [4.8, 3.4, 1.6, 0.2],
        [4.8, 3. , 1.4, 0.1],
        [4.3, 3. , 1.1, 0.1],
        [5.8, 4. , 1.2, 0.2],
        [5.7, 4.4, 1.5, 0.4],
        [5.4, 3.9, 1.3, 0.4],
        [5.1, 3.5, 1.4, 0.3],
        [5.7, 3.8, 1.7, 0.3],
```

In [ ]: