PYTHON PANDAS

Pandas stands for
“Python Data Analysis Library”.

About Python Pandas

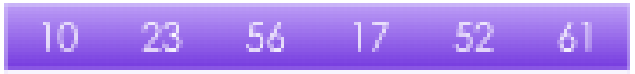
❖ Fast and Efficient Data Frame object with default and customized indexing.
❖ Support different File formats such as CSV and text files, Microsoft Excel, SQL databases etc. to be loaded into in-memory data objects.
❖ Intelligent Data alignment and integrated handling of missing data.
❖ Label-based slicing, indexing and sub setting of large data sets.
❖ Columns from a data structure can be deleted or inserted.
❖ Group by data for aggregation and transformations.

❖ High Performance merging and joining of data.
❖Python Pandas are widely used in academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

Data Structures in Python Pandas

Series

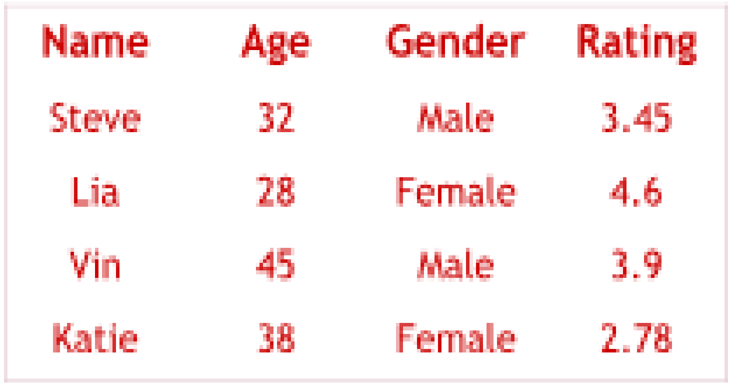
Series is a one-dimensional array like structure with homogeneous data. For example, the following series is a collection of integers 10, 23, 56, …



10	23	56	17	52	61

DataFrame

DataFrame is a two-dimensional array with heterogeneous data. For example,



Name	Age	Gender	Rating
Steve	32	Male	3.45
Lia	28	Female	4.6
Vin	45	Male	3.9
Katie	38	Female	2.78

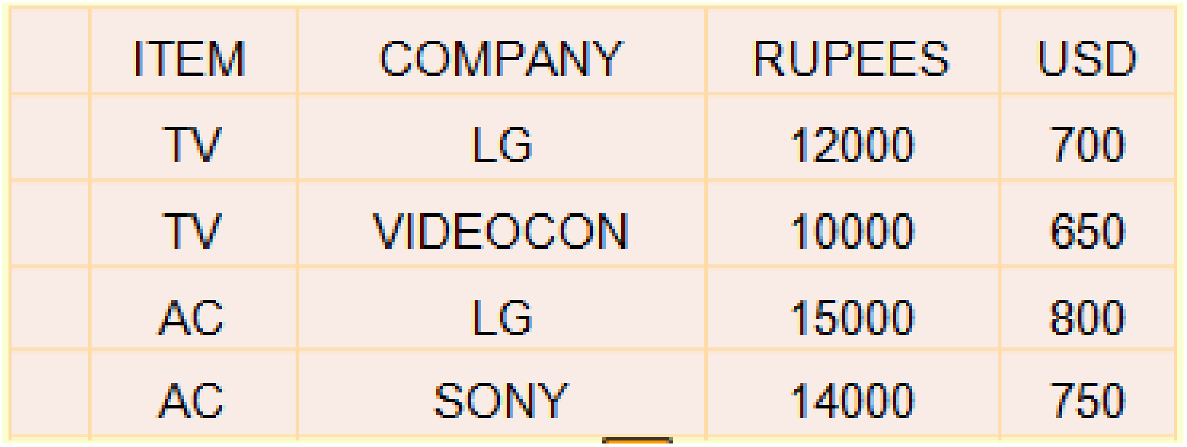
Pivoting - Dataframe

DataFrame -It is a 2-dimensional data structure with columns of different types. It is just similar to a spreadsheet or SQL table, or a dictionary of Series objects. It is generally the most commonly used pandas object.

Pivot –Pivot reshapes data and uses unique values from index/ columns to form axes of the resulting dataframe. Index is column name to use to make new frame's index.Columns is column name to use to make new frame's columns.Values is column name to use for populating new frame's values.

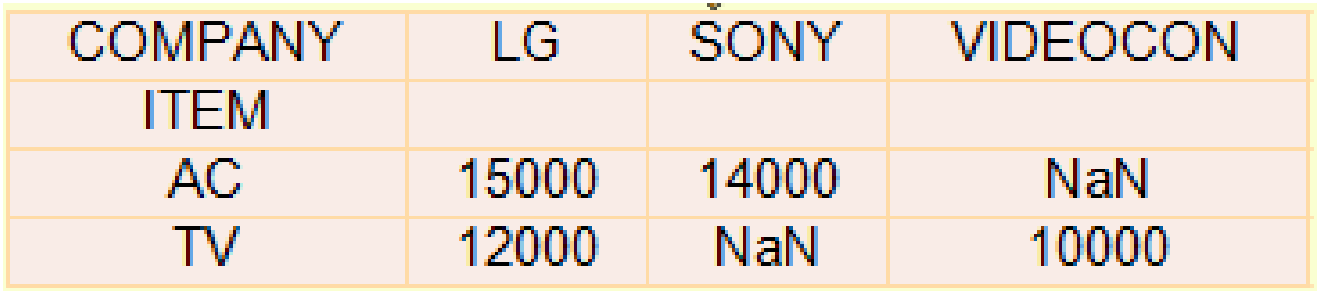
Pivot table - Pivot table is used to summarize and aggregate data inside dataframe.

Example of pivot:



	ITEM	COMPANY	RUPEES	USD
	TV	LG	12000	700
	TV	VIDEOCON	10000	650
	AC	LG	15000	800
	AC	SONY	14000	750

DATAFRAME



COMPANY	LG	SONY	VIDEOCON
ITEM			
AC	15000	14000	NaN
TV	12000	NaN	10000

PIVOT

Pivoting - Dataframe

There are two functions available in python for pivoting dataframe.

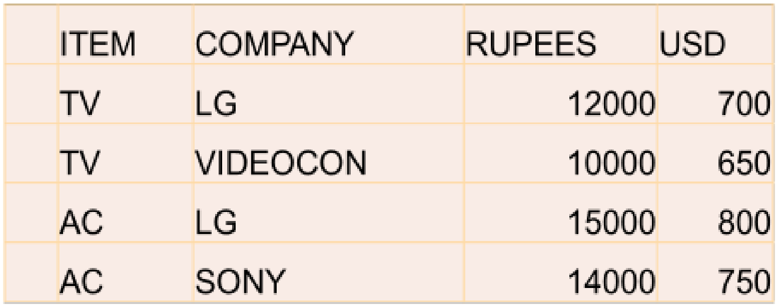
1. pivot()

2. pivot_table()

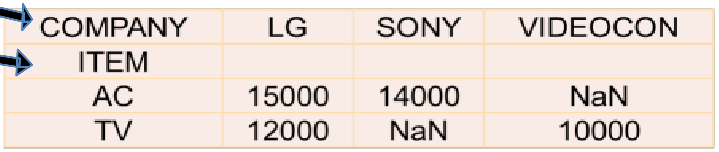
1. pivot() - This function is used to create a new derived table(pivot) from existing dataframe. It takes 3 arguments : index, columns, and values. As a value for each of these parameters we need to specify a column name in the original table(dataframe). Then the pivot function will create a new table(pivot), whose row and column indices are the unique values of the respective parameters. The cell values of the new table are taken from column given as the values parameter.

Pivoting - Dataframe

```
#pivot() e.g. program
from collections import OrderedDict
from pandas import DataFrame  import
pandas as pd
import numpy as np

table = OrderedDict((
    ("ITEM", ['TV', 'TV', 'AC', 'AC']),
    ('COMPANY',['LG', 'VIDEOCON', 'LG', 'SONY']),
    ('RUPEES', ['12000', '10000', '15000', '14000']),
    ('USD',   ['700', '650', '800', '750'])
))
d = DataFrame(table)  print("DATA
OF DATAFRAME")
print(d)
p = d.pivot(index='ITEM', columns='COMPANY', values='RUPEES')

print("\n\nDATA OF PIVOT")
print(p)
print (p[p.index=='TV'].LG.values)
```



	ITEM	COMPANY	RUPEES	USD
	TV	LG	12000	700
	TV	VIDEOCON	10000	650
	AC	LG	15000	800
	AC	SONY	14000	750



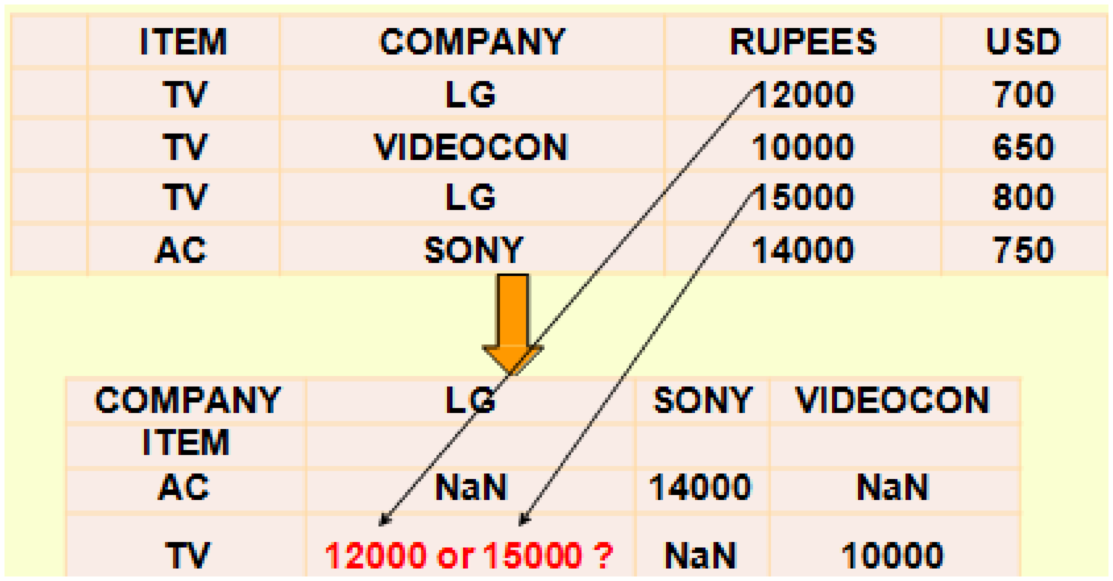
COMPANY	LG	SONY	VIDEOCON
ITEM			
AC	15000	14000	NaN
TV	12000	NaN	10000

#pivot() creates a new table/DataFrame whose columns are the unique values in COMPANY and whose rows are indexed with the unique values of ITEM.Last statement of above program return value of TV item LG company i.e. 12000

Pivoting - Dataframe

#Common Problem in Pivoting

pivot method takes at least 2 column names as parameters - the index and the columns name as parameters. Now the problem may arise- What happens if we have multiple rows with the same values for these columns? What will be the value of the corresponding cell in the pivoted table using pivot method? The following diagram depicts the problem:



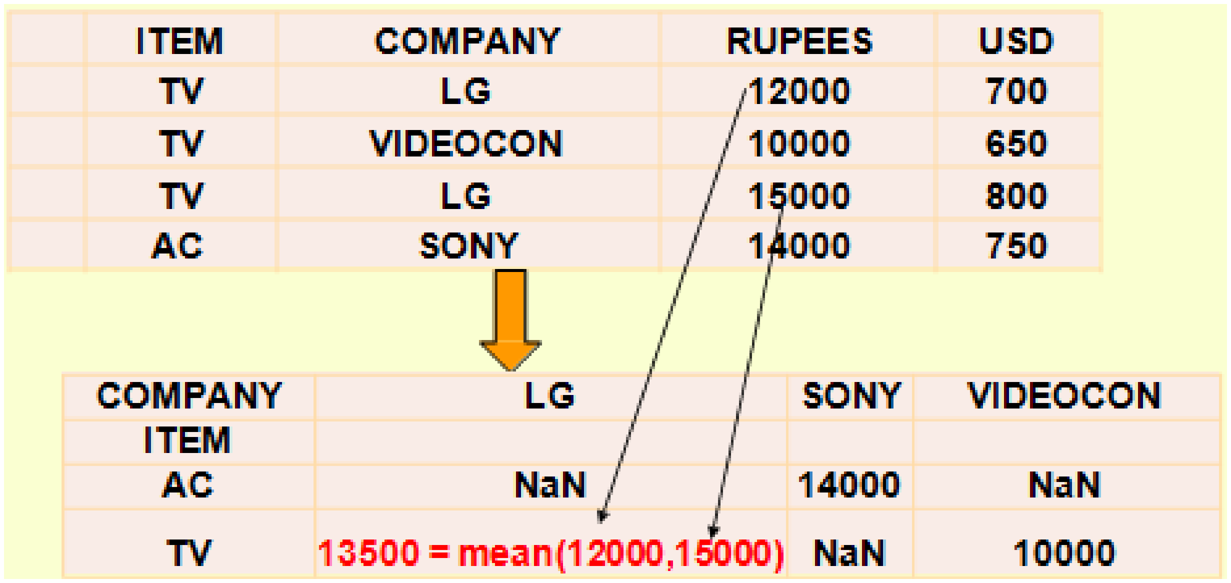
	ITEM	COMPANY	RUPEES	USD
	TV	LG	12000	700
	TV	VIDEOCON	10000	650
	TV	LG	15000	800
	AC	SONY	14000	750

COMPANY	LG	SONY	VIDEOCON
ITEM			
AC	NaN	14000	NaN
TV	12000 or 15000 ?	NaN	10000

d.pivot(index='ITEM', columns='COMPANY', values='RUPEES')
It throws an exception with the following message: ValueError: Index contains duplicate entries, cannot reshape

#Pivot Table

The pivot_table() method comes to solve this problem. It works like pivot, but it aggregates the values from rows with duplicate entries for the specified columns.



	ITEM	COMPANY	RUPEES	USD
	TV	LG	12000	700
	TV	VIDEOCON	10000	650
	TV	LG	15000	800
	AC	SONY	14000	750

COMPANY	LG	SONY	VIDEOCON
ITEM			
AC	NaN	14000	NaN
TV	13500 = mean(12000,15000)	NaN	10000

d.pivot_table(index='ITEM', columns='COMPANY', values='RUPEES‘,aggfunc=np.mean)

In essence pivot_table is a generalisation of *pivot*, which allows you to aggregate multiple values with the same destination in the pivoted table.

Sorting - Dataframe

Sorting means arranging the contents in ascending or descending order.There are two kinds of sorting available in pandas(Dataframe).

1. By value(column)

2. By index

1. By value - Sorting over dataframe column's elements is supported by sort_values() method. We will cover here three aspects of sorting values of dataframe.

- Sort a pandas dataframe in python by Ascending and Descending
- Sort a python pandas dataframe by single column
- Sort a pandas dataframe by multiple columns.

Sorting - Dataframe

Sort the python pandas Dataframe by single column – Ascending order

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Sachin','Dhoni','Virat','Rohit','Shikhar']),
   'Age':pd.Series([26,27,25,24,31]),
   'Score':pd.Series([87,89,67,55,47])}

#Create a DataFrame
df = pd.DataFrame(d)
print("Dataframe contents without sorting")
print (df)
df=df.sort_values(by='Score')
print("Dataframe contents after sorting")
print (df)
```

OUTPUT

```
Dataframe contents without sorting
     Name      Age  Score

1   Sachin     26     87
2    Dhoni     27     89
3    Virat     25     67
4    Rohit      24     55
5     Shikhar  31     47
```

```
Dataframe contents after sorting

   Name       Age  Score
 5 Shikhar    31     47
4 Rohit       24      55
3 Virat       25     67
2 Dhoni       27      87
1 Sachin       26    89
```

# In above example dictionary object is used to create the dataframe.Elements of dataframe object df is s orted by sort_value() method.As argument we are passing value score for by parameter only.by default it is sorting in ascending manner.

Sorting - Dataframe

Sort the python pandas Dataframe by single column – Descending order

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Sachin','Dhoni','Virat','Rohit','Shikhar']),
   'Age':pd.Series([26,27,25,24,31]),
   'Score':pd.Series([87,89,67,55,47])}

#Create a DataFrame
df = pd.DataFrame(d)
print("Dataframe contents without sorting")
print (df)
df=df.sort_values(by='Score',ascending=0)
print("Dataframe contents after sorting")
print (df)
```

```
OUTPUT
Dataframe contents without sorting
     Name       Age   Score
 1  Sachin      26     89
 2   Dhoni      27     87
 3   Virat      25     67
 4   Rohit      24     55
 5    Shikhar   31    47

 Dataframe contents after sorting
       Name      Age     Score
 1   Dhoni       27       89
 0  Sachin       26        87
 2   Virat       25       67
 3   Rohit      24        55
 4   Shikhar    31        47
```

# In above example dictionary object is used to create the dataframe. Elements of dataframe object df are sorted by sort_value() method. We are passing 0 for Ascending parameter ,which sort the data in descending order of score.

Sorting - Dataframe

Sort the pandas Dataframe by Multiple Columns

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Sachin','Dhoni','Virat','Rohit','Shikhar']),
   'Age':pd.Series([26,25,25,24,31]),
 'Score':pd.Series([87,67,89,55,47])}

#Create a DataFrame  df =
pd.DataFrame(d)
print("Dataframe contents without sorting")
print (df)
df=df.sort_values(by=['Age', 'Score'],ascending=[True,False])
print("Dataframe contents after sorting")
print (df)
# In above example dictionary object is used to create  the
dataframe.Elements of dataframe object df  are sorted by
sort_value() method. We are passing two columns  as the
parameter value and in ascending parameter also  with
two parameters first true and second false,which  means
sort in ascending order of age and descending  order of
score
```

```
OUTPUT
Dataframe contents without sorting
     Name       Age      Score

 1   Sachin      26 87
 2    Dhoni     25 67
 3    Virat      25         89
 4    Rohit     24          55
 5     Shikhar  31     47

 Dataframe contents after sorting
      Name      Age        Score

  4    Rohit      24 55
  3    Virat     2589
  2    Dhoni      25  67
  1     Sachin    26 87
  5    Shikhar    31 47
```

Sorting - Dataframe

2. By index - Sorting on the basis of dataframe index is done using method sort_index(), in conjunction with sort_values() method. We will now see the two aspects of sorting on the basis of index of dataframe.

- **How to sort a pandas dataframe in python by index in Ascending order**
- **How to sort a pandas dataframe in python by index in Descending order**

Sorting - Dataframe

Sort the dataframe in python pandas by index in ascending order:

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Sachin','Dhoni','Virat','Rohit','Shikhar']),
   'Age':pd.Series([26,25,25,24,31]),
   'Score':pd.Series([87,67,89,55,47])}

# Create a DataFrame
df = pd.DataFrame(d)
df=df.reindex([1,4,3,2,0])
print("Dataframe contents without sorting")
print (df)
df1=df.sort_index()
print("Dataframe contents after sorting")
print (df1)
```

# In above example dictionary object is used to create the dataframe. Elements of dataframe object df is first reindexed by reindex() method,index 1 is positioned at 0,4 at 1 and so on.then sorting by sort_index() method. By default it is sorting in ascending order of index.

```
OUTPUT
Dataframe contents without sorting
     Name Age  Score
1   Dhoni   25    67
4 Shikhar   31     47
3   Rohit   24     55
2   Virat  25     89
0  Sachin   26     87
Dataframe contents after sorting
     Name Age  Score
0  Sachin   26     87
1   Dhoni   25     67
2   Virat  25     89
3   Rohit   24     55
4 Shikhar   31     47
```

index

Sorting - Dataframe

Sorting pandas dataframe by index in descending order:

```
import pandas as pd  import
numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Sachin','Dhoni','Virat','Rohit','Shikhar']),
   'Age':pd.Series([26,25,25,24,31]),
 'Score':pd.Series([87,67,89,55,47])}

 # Create a DataFrame  df =
 pd.DataFrame(d)
 df=df.reindex([1,4,3,2,0])
 print("Dataframe contents without sorting")
 print (df)  df1=df.sort_index(ascending=0)
 print("Dataframe contents after sorting")  print
 (df1)
```

#In above example dictionary object is used to create the dataframe.Elements of dataframe object df are first reindexed by reindex() method,index 1 is positioned at 0,4 at 1 and so on.then sorting by sort_index() method.

Passing ascending=0 as argument for descending order.

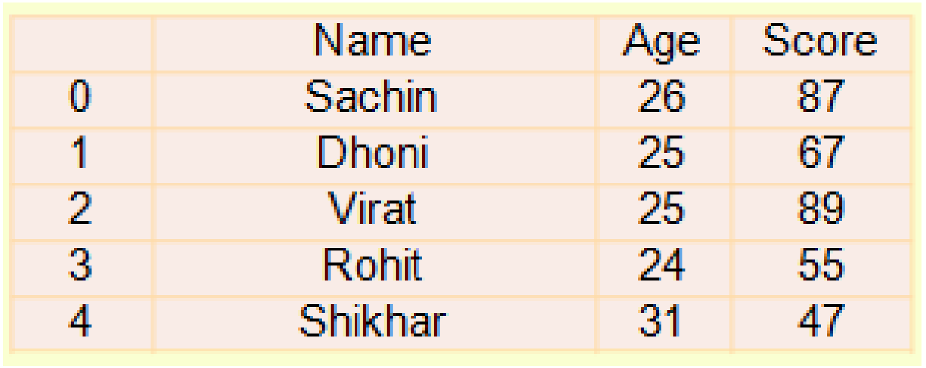
```
OUTPUT
Dataframe contents without sorting
         Name     Age     Score
     1    Dhoni    25        67
     4  Shikhar    31        47
     3   Rohit     24        55
     2   Virat     25        89
     0   Sachin    26        87

 Dataframe contents after sorting
         Name     Age    Score

      4  Shikhar  31    47
      3  Rohit     24   55
       2 Virat      25   89
      1  Dhoni      25   67
      0  Sachin      26   87
```

index

Aggregation/Descriptive statistics - Dataframe

Data aggregation –

Aggregation is the process of turning the values of a dataset (or a subset of it) into one single value or data aggregation is a multivalued function ,which require multiple values and return a single value as a result.There are number of aggregations possible like count,sum,min,max,median,quartile etc. These(count,sum etc.) are descriptive statistics and other related operations on DataFrame Let us make this clear! If we have a DataFrame like…



	Name	Age	Score
0	Sachin	26	87
1	Dhoni	25	67
2	Virat	25	89
3	Rohit	24	55
4	Shikhar	31	47

…then a simple aggregation method is to calculate the summary of the Score, which is 87+67+89+55+47= 345. Or a different aggregation method would be to count the number of Name, which is 5.

Aggregation/Descriptive statistics - dataframe

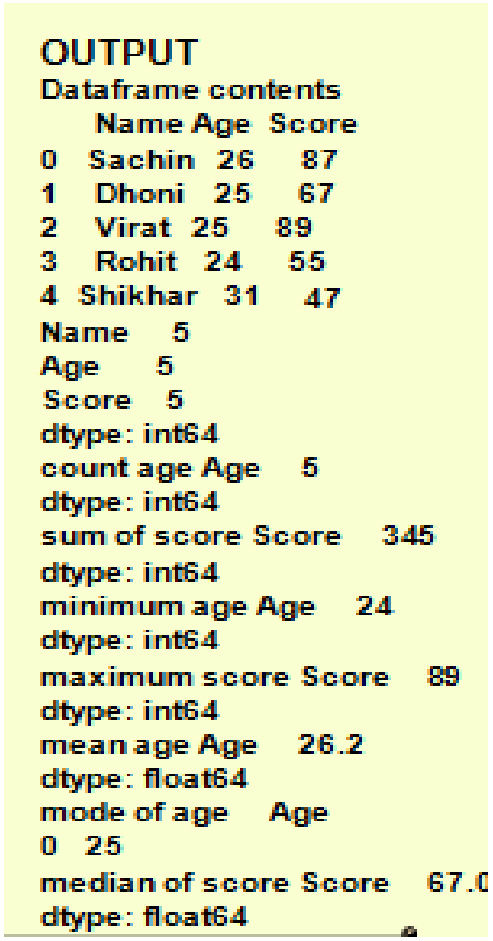
#e.g. program for data aggregation/descriptive statistics

import pandas as pd
import numpy as np

- #Create a Dictionary of series
- d = {'Name':pd.Series(['Sachin','Dhoni','Virat','Rohit','Shikhar']),
- 'Age':pd.Series([26,25,25,24,31]),
- 'Score':pd.Series([87,67,89,55,47])} #Create a DataFrame
- df = pd.DataFrame(d) print("Dataframe contents")

print (df)

- print(df.count())
- print("count age",df[['Age']].count())
- print("sum of score",df[['Score']].sum())
- print("minimum age",df[['Age']].min())
- print("maximum score",df[['Score']].max())
- print("mean age",df[['Age']].mean())
- print("mode of age",df[['Age']].mode())
- print("median of score",df[['Score']].median())



```
OUTPUT
Dataframe contents
    Name Age  Score
0  Sachin  26    87
1   Dhoni  25    67
2   Virat  25    89
3   Rohit  24    55
4 Shikhar  31    47
Name     5
Age      5
Score    5
dtype: int64
count age Age    5
dtype: int64
sum of score Score    345
dtype: int64
minimum age Age    24
dtype: int64
maximum score Score    89
dtype: int64
mean age Age    26.2
dtype: float64
mode of age    Age
0  25
median of score Score    67.0
dtype: float64
```

THANKS