Introduction to Pandas

- **pandas** is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- It provides highly optimized performance with back-end source code is purely written in **C** or ***Python***.

Dimensions	Name	Description
1	Series	1D labeled homogeneously-typed array
2	DataFrame	General 2D labeled, size-mutable tabular structure with potentially heterogeneously-typed column

Key features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, indexing and subsetting 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.
- Time Series functionality.

Pandas - Series

- ***Series*** is one dimensional(1-D) array defined in pandas that can be used to store any data type.

```
#import the pandas library and aliasing as pd
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data)
print s
```

Index	Data
1	'A'
2	'B'
3	'C'
4	'D'
5	'E'

Pandas - DataFrame

- **DataFrames** is two-dimensional(2-D) data structure defined in pandas which consists of rows and columns.

```
# Program to Create Data Frame with two dict
dict1 ={'a':1, 'b':2, 'c':3, 'd':4}          #
dict2 ={'a':5, 'b':6, 'c':7, 'd':8, 'e':9} #
Data = {'first':dict1, 'second':dict2}  # De
df = pd.DataFrame(Data)  # Create DataFrame
```

```
import pandas as pd
data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,columns=['Name','Age'])
print df
```

Pandas - DataFrame

column index

row index	0	1	2	3
	country	**continent**	**GDP**	**population**
0	USA	North America	19,390,604	322,179,605
1	China	Asia	12,237,700	1,403,500,365
2	Japan	Asia	4,872,137	127,748,513
3	Germany	Europe	3,677,439	81,914,672
4	UK	Europe	2,622,434	65,788,574
5	India	Asia	2,597,491	1,324,171,354

Indexing Data in Pandas

- Indexing in pandas means simply selecting particular rows and columns of data from a DataFrame.
- Indexing can also be known as **Subset Selection**.

```
# importing pandas package
import pandas as pd

# making data frame from csv file
data = pd.read_csv("nba.csv", index_col ="Name")

# retrieving columns by indexing operator
first = data["Age"]
```

Indexing Data in Pandas

```
# importing pandas package
import pandas as pd

# making data frame from csv file
data = pd.read_csv("nba.csv", index_col ="Name")

# retrieving multiple columns by indexing operator
first = data[["Age", "College", "Salary"]]
```

Selecting Data in Pandas

```
# importing pandas package
import pandas as pd

# making data frame from csv file
data = pd.read_csv("nba.csv", index_col ="Name")

# retrieving row by loc method
first = data.loc["Avery Bradley"]
second = data.loc["R.J. Hunter"]


print(first, "\n\n\n", second)
```

Splitting Data in Pandas

Pandas provide a method to split string around a passed separator/delimiter. After that, the string can be stored as a list in a series or it can also be used to create multiple column data frames from a single separated string.

```
# importing pandas module
import pandas as pd

# reading csv file from url
data = pd.read_csv("https://media.geeksforgeeks.org/wp-cc

# dropping null value columns to avoid errors
data.dropna(inplace = True)
```

Splitting Data in Pandas

```
# new data frame with split value columns
new = data["Name"].str.split(" ", n = 1, expand = True)

# making separate first name column from new data frame
data["First Name"]= new[0]

# making separate last name column from new data frame
data["Last Name"]= new[1]

# Dropping old Name columns
data.drop(columns =["Name"], inplace = True)

# df display
data
```

Apply in Pandas

```
import pandas as pd
def add(a, b, c):
    return a + b + c
def main():
    data = {'A':[1, 2, 3], 'B':[4, 5, 6], 'C':[7, 8, 9] }
    df = pd.DataFrame(data)
    print("Original DataFrame:\n", df)
    df['add'] = df.apply(lambda row : add(row['A'],
                                          row['B'], row['C']), axis = 1)
    print('\nAfter Applying Function:')
    print(df)

if __name__ == '__main__':
    main()
```

Combining Data in Pandas

```
# importing pandas module
import pandas as pd

# creating first series
first =[1, 2, 5, 6, 3, 7, 11, 0, 4]

# creating second series
second =[5, 3, 2, 1, 3, 9, 21, 3, 1]

# making series
first = pd.Series(first)

# making seriesa
second = pd.Series(second)

# calling .combine() method
result = first.combine(second, (lambda x1, x2: x1 if x1 < x2 else x2))

# display
result
```

Combining Data in Pandas

```
# importing pandas module
import pandas as pd

# Define a dictionary containing employee data
data1 = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Define a dictionary containing employee data
data2 = {'Name':['Abhi', 'Ayushi', 'Dhiraj', 'Hitesh'],
        'Age':[17, 14, 12, 52],
        'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'],
        'Qualification':['Btech', 'B.A', 'Bcom', 'B.hons']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data1,index=[0, 1, 2, 3])

# Convert the dictionary into DataFrame
df1 = pd.DataFrame(data2, index=[4, 5, 6, 7])

print(df, "\n\n", df1)
```

Combining Data in Pandas

Now we apply `.concat` function in order to concat two dataframe

```
# using a .concat() method
frames = [df, df1]

res1 = pd.concat(frames)
res1
```

Combining Data in Pandas

	Name	Age	Address	Qualification
0	Jai	27	Nagpur	Msc
1	Princi	24	Kanpur	MA
2	Gaurav	22	Allahabad	MCA
3	Anuj	32	Kannuaj	Phd

	Name	Age	Address	Qualification
4	Abhi	17	Nagpur	Btech
5	Ayushi	14	Kanpur	B.A
6	Dhiraj	12	Allahabad	Bcom
7	Hitesh	52	Kannuaj	B.hons

	Name	Age	Address	Qualification
0	Jai	27	Nagpur	Msc
1	Princi	24	Kanpur	MA
2	Gaurav	22	Allahabad	MCA
3	Anuj	32	Kannuaj	Phd
4	Abhi	17	Nagpur	Btech
5	Ayushi	14	Kanpur	B.A
6	Dhiraj	12	Allahabad	Bcom
7	Hitesh	52	Kannuaj	B.hons

Combining Data in Pandas

Now we apply `.append()` function inorder to concat to dataframe

```
# using append function

res = df.append(df1)
res
```

Combining Data in Pandas

	Name	Age	Address	Qualification
0	Jai	27	Nagpur	Msc
1	Princi	24	Kanpur	MA
2	Gaurav	22	Allahabad	MCA
3	Anuj	32	Kannuaj	Phd

	Name	Age	Address	Qualification
4	Abhi	17	Nagpur	Btech
5	Ayushi	14	Kanpur	B.A
6	Dhiraj	12	Allahabad	Bcom
7	Hitesh	52	Kannuaj	B.hons

	Name	Age	Address	Qualification
0	Jai	27	Nagpur	Msc
1	Princi	24	Kanpur	MA
2	Gaurav	22	Allahabad	MCA
3	Anuj	32	Kannuaj	Phd
4	Abhi	17	Nagpur	Btech
5	Ayushi	14	Kanpur	B.A
6	Dhiraj	12	Allahabad	Bcom
7	Hitesh	52	Kannuaj	B.hons

Merging Data in Pandas

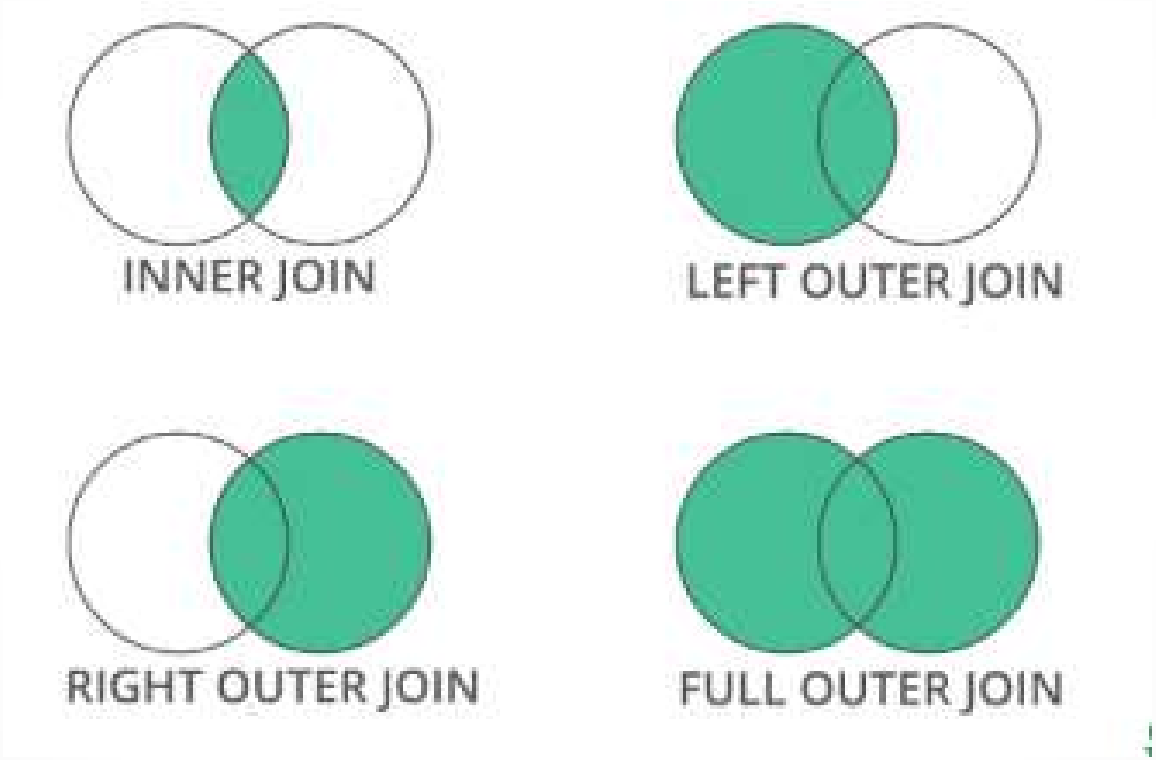
```
import pandas as pd

left = pd.DataFrame({ 'id':[1,2,3,4,5], 'Name': ['Alex',
'Amy', 'Allen', 'Alice', 'Ayoung'],
'subject_id':['sub1','sub2','sub4','sub6','sub5']})

right = pd.DataFrame( {'id':[1,2,3,4,5], 'Name': ['Billy',
'Brian', 'Bran', 'Bryce', 'Betty'],
'subject_id':['sub2','sub4','sub3','sub6','sub5']})

print left
print right
print pd.merge(left,right,on='id')
print pd.merge(left,right,on=['id','subject_id'])
print pd.merge(left, right, on='subject_id', how='left')
```

Merging Data in Pandas

The **how** argument to merge specifies how to determine which keys are to be included in the resulting table. If a key combination does not appear in either the left or the right tables, the values in the joined table will be NA.

Merge Method	SQL Equivalent	Description
left	LEFT OUTER JOIN	Use keys from left object
right	RIGHT OUTER JOIN	Use keys from right object
outer	FULL OUTER JOIN	Use union of keys
inner	INNER JOIN	Use intersection of keys

Merging Data in Pandas



Descriptive Statistics in Pandas

Descriptive statistics is a study of data analysis to describe, show or summarize data in a meaningful way. It involves the calculation of various measures such as the **measure of center**, **the measure of variability**, **percentiles** and also the **construction of tables & graphs.**

Types of Data:

- **Categorical data:** non-numerical information such as gender, race, religion, marital status etc.
- **Numerical data:** measurement or count such as height, weight, age, salary, number of children, etc

Descriptive Statistics in Pandas

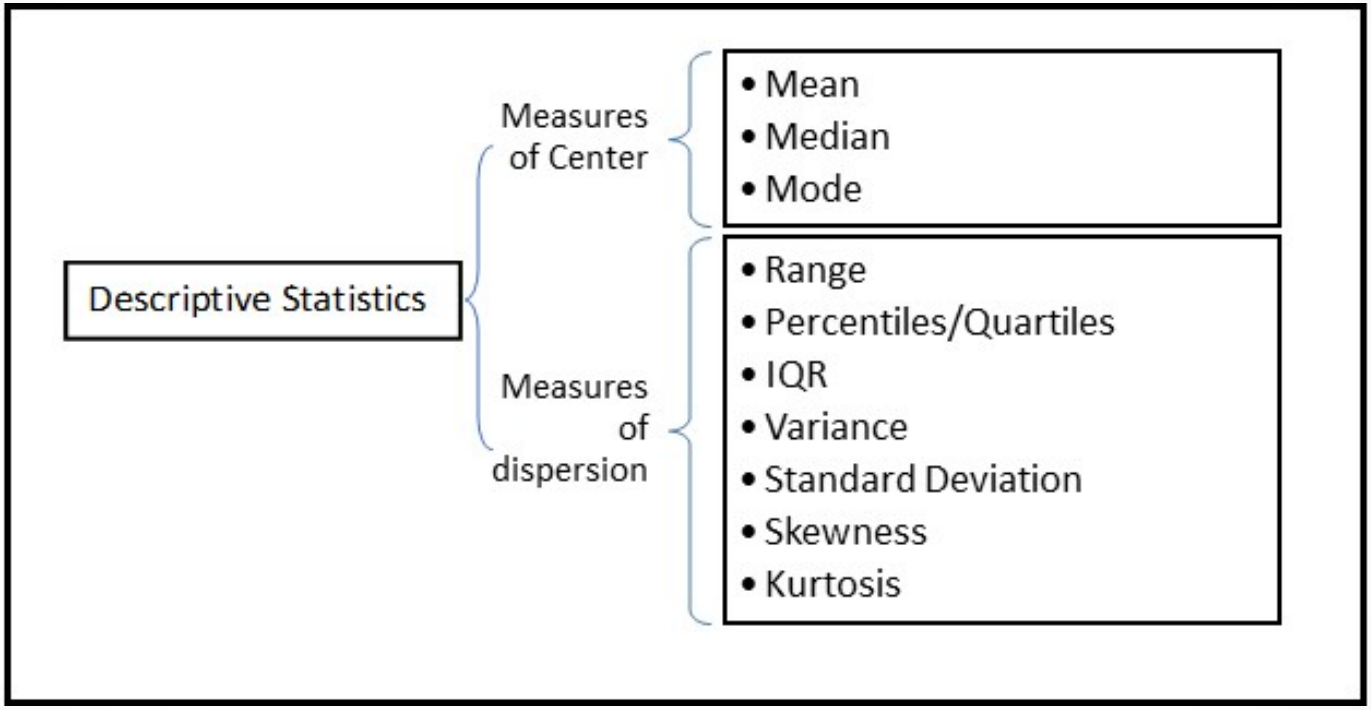
Finding sum of the data

```
import pandas as pd
import numpy as np

d={'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack','Lee','David','Gasper','Betina','Andres']),
'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65]) }

df = pd.DataFrame(d)
print df.sum() #df.mean(), df.std(),
```

Descriptive Statistics Functions

Sr.No.	Function	Description
1	count()	Number of non-null observations
2	sum()	Sum of values
3	mean()	Mean of Values
4	median()	Median of Values
5	mode()	Mode of values
6	std()	Standard Deviation of the Values
7	min()	Minimum Value
8	max()	Maximum Value
9	abs()	Absolute Value
10	prod()	Product of Values
11	cumsum()	Cumulative Sum
12	cumprod()	Cumulative Product

Descriptive Statistics Functions



Summarizing Statistics in Pandas

```
import pandas as pd
import numpy as np

d={'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack','Lee','Dav
id','Gasper','Betina','Andres']),
'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65]) }

df = pd.DataFrame(d)
print df.describe()
```

Summarizing Statistics in Pandas

This function gives the **mean, std** and **IQR** values.

It excludes the character columns and gives summary about numeric columns. **'include'** is the argument which is used to pass necessary information regarding what columns need to be considered for summarizing.

Takes the list of values; by default, 'number'.

- **object** – Summarizes String columns
- **number** – Summarizes Numeric columns
- **all** – Summarizes all columns together (Should not pass it as a list value)

Handling missing data in Pandas

Missing Data can occur when no information is provided for one or more items or for a whole unit. Missing Data is a very big problem in real life scenario.

Missing Data can also refer to as NA(Not Available) values in pandas.

In Pandas missing data is represented by two value:

- None: None is a Python singleton object that is often used for missing data in Python code.
- NaN : NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation.

Handling missing data in Pandas

Methods to handle missing data:

- isnull()
- notnull()
- dropna()
- fillna()
- replace()
- interpolate()

Handling missing data in Pandas

```
# importing pandas as pd
import pandas as pd

# importing numpy as np
import numpy as np

# dictionary of lists
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}

# creating a dataframe from list
df = pd.DataFrame(dict)

# using isnull() function
df.isnull()
```

Handling missing data in Pandas

```
# importing pandas package
import pandas as pd

# making data frame from csv file
data = pd.read_csv("employees.csv")

# creating bool series True for NaN values
bool_series = pd.isnull(data["Gender"])

# filtering data
# displaying data only with Gender = NaN
data[bool_series]
```

Handling missing data in Pandas

```
# importing pandas as pd
import pandas as pd

# importing numpy as np
import numpy as np

# dictionary of lists
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}

# creating a dataframe from dictionary
df = pd.DataFrame(dict)

# filling missing value using fillna()
df.fillna(0)
```

Handling missing data in Pandas

```
# importing pandas as pd
import pandas as pd

# importing numpy as np
import numpy as np

# dictionary of lists
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}

# creating a dataframe from dictionary
df = pd.DataFrame(dict)

# filling a missing value with
# previous ones
df.fillna(method ='pad')
```

Handling missing data in Pandas

```
# importing pandas package
import pandas as pd

# making data frame from csv file
data = pd.read_csv("employees.csv")

# filling a null values using fillna()
data["Gender"].fillna("No Gender", inplace = True)

data
```