DATE : 20 june 2024

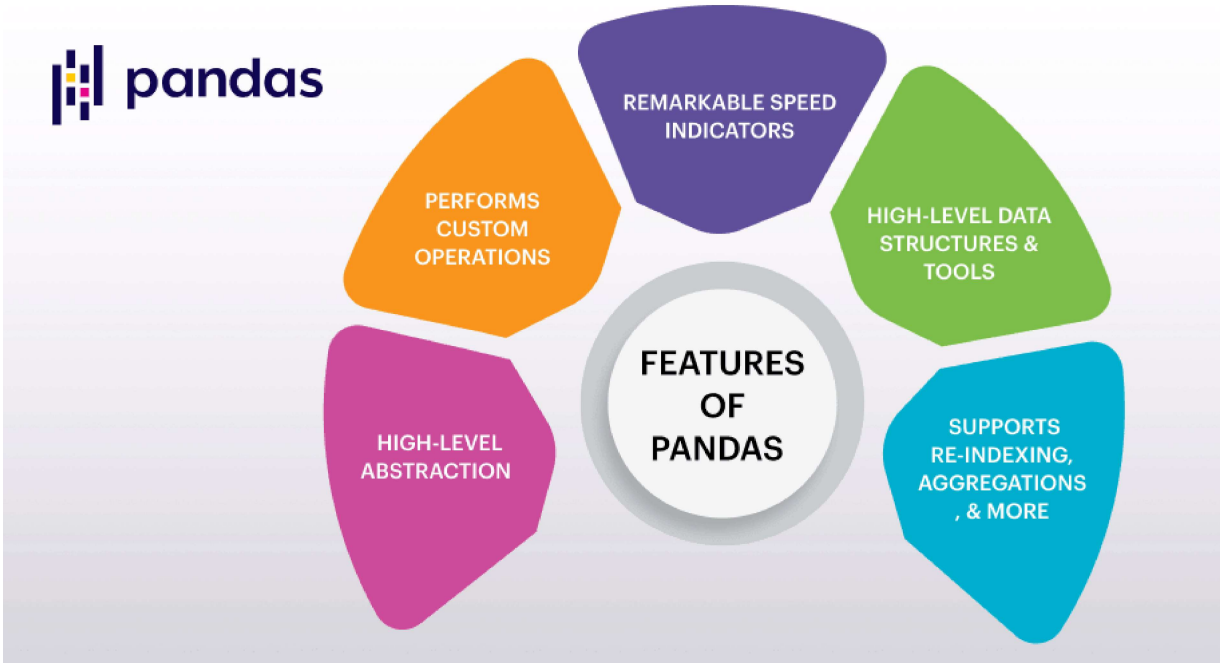
DAY : Thursday

TOPICS : Library: pandas

Pandas is a powerful and widely-used open-source library in Python for data manipulation and analysis. It provides data structures and functions needed to work with structured data seamlessly and efficiently. Here are some key features and components of the Pandas library:

Key Features of Pandas

1. **Data Structures**:
   - **Series**: A one-dimensional array-like object containing a sequence of values. It is similar to a list or a one-dimensional NumPy array but with labeled axes (index).
   - **DataFrame**: A two-dimensional table of data with labeled axes (rows and columns). It can be thought of as a collection of Series objects sharing the same index.
2. **Data Alignment**: Automatic alignment of data for arithmetic operations on Series and DataFrame objects, making it easy to manage missing data.
3. **Data Cleaning**: Functions for handling missing data, such as filling, replacing, and dropping missing values.
4. **Data Transformation**: Tools for reshaping, pivoting, and transforming data, including functions for merging, joining, and concatenating DataFrame objects.
5. **Data Aggregation and Grouping**: Powerful group-by functionality to split data into groups, apply functions to each group, and combine the results.
6. **Time Series Handling**: Specialized tools for working with time series data, including date range generation and frequency conversion.
7. **Input and Output**: Functions for reading and writing data to and from various file formats, including CSV, Excel, SQL databases, and more.
8. **Visualization**: Integration with plotting libraries like Matplotlib to provide basic data visualization capabilities directly from DataFrame and Series objects.




```
import numpy as np
import pandas as pd
```

Series

```
l1 = [1, 2, 3, 4, 5, 6]
labels = ['a', 'b', 'c', 'd', 'e', 'f']
d1 = {"A":10, "B":20, "C":30, "D":40, "E":50}
```

```
s1 = pd.Series(l1)
s1
```

```
0    1
1    2
2    3
3    4
4    5
5    6
dtype: int64
```

```
s1[4]
```

```
5
```

```
s2 = pd.Series(labels)
s2
```

```
0    a
1    b
2    c
3    d
4    e
5    f
dtype: object
```

```
s2[4]
```

```
'e'
```

```
s3 = pd.Series(data=l1, index=labels)
s3
```

```
a    1
b    2
c    3
d    4
e    5
f    6
dtype: int64
```

```
s3['a']
```

```
1
```

```
s3[0]
```

```
1
```

```
pd.Series(d1)
```

```
A    10
B    20
C    30
D    40
E    50
dtype: int64
```

DataFrame

```
arr = np.random.randint(low=1, high=100, size=(5, 6))
arr
```

```
array([[55, 53, 65,  2, 56, 33],
       [50, 26, 87,  5, 79, 44],
       [ 7, 73, 51, 32,  6, 19],
       [21, 28, 26,  5, 43, 64],
       [22, 58, 71, 18, 67, 47]])
```

```
type(arr)
```

```
numpy.ndarray
```

```
pd.DataFrame(arr)
```

	0	1	2	3	4	5
0	55	53	65	2	56	33
1	50	26	87	5	79	44
2	7	73	51	32	6	19
3	21	28	26	5	43	64
4	22	58	71	18	67	47

```
df = pd.DataFrame(arr, index=["A", "B", "C", "D", "E"], columns=["U", "V", "W", "X", "Y", "Z"])
df
```

	U	V	W	X	Y	Z
A	55	53	65	2	56	33
B	50	26	87	5	79	44
C	7	73	51	32	6	19
D	21	28	26	5	43	64
E	22	58	71	18	67	47

```
type(df)
```

```
pandas.core.frame.DataFrame
```

Grabbing Columns

```
df["X"]
```

```
A     2
B     5
C    32
D     5
E    18
Name: X, dtype: int64
```

```
df[["X", "Z", "V"]]
```

	X	Z	V
A	2	33	53
B	5	44	26
C	32	19	73
D	5	64	28
E	18	47	58

Grabbing Rows

```
df.loc["C"]
```

```
U     7
V    73
W    51
X    32
Y     6
Z    19
Name: C, dtype: int64
```

```
df.loc[["A", "B", "E"]]
```

	U	V	W	X	Y	Z
A	55	53	65	2	56	33
B	50	26	87	5	79	44
E	22	58	71	18	67	47

```
df.iloc[2]
```

```
U     7
V    73
W    51
X    32
Y     6
Z    19
Name: C, dtype: int64
```

Adding a New Column

```
df
```

	U	V	W	X	Y	Z
A	55	53	65	2	56	33
B	50	26	87	5	79	44
C	7	73	51	32	6	19
D	21	28	26	5	43	64
E	22	58	71	18	67	47

```
df['New'] = [10, 20, 30, 40, 50]
```

```
df
```

	U	V	W	X	Y	Z	New
A	55	53	65	2	56	33	10
B	50	26	87	5	79	44	20
C	7	73	51	32	6	19	30
D	21	28	26	5	43	64	40
E	22	58	71	18	67	47	50

```
df['New'] = [100, 200, 300, 400, 500]
```

```
df
```

	U	V	W	X	Y	Z	New
A	55	53	65	2	56	33	100
B	50	26	87	5	79	44	200
C	7	73	51	32	6	19	300
D	21	28	26	5	43	64	400
E	22	58	71	18	67	47	500

Deleting a Column

```
df
```

	U	V	W	X	Y	Z	New
A	55	53	65	2	56	33	100
B	50	26	87	5	79	44	200
C	7	73	51	32	6	19	300
D	21	28	26	5	43	64	400
E	22	58	71	18	67	47	500

```
df.drop('New', axis=1)
```

	U	V	W	X	Y	Z
A	55	53	65	2	56	33
B	50	26	87	5	79	44
C	7	73	51	32	6	19
D	21	28	26	5	43	64
E	22	58	71	18	67	47

```
df
```

```
df.drop('New', axis=1, inplace=True)
```

```
df
```

	U	V	W	X	Y	Z
A	55	53	65	2	56	33
B	50	26	87	5	79	44
C	7	73	51	32	6	19
D	21	28	26	5	43	64
E	22	58	71	18	67	47

Conditional Selection

```
df
```

	U	V	W	X	Y	Z
A	55	53	65	2	56	33
B	50	26	87	5	79	44
C	7	73	51	32	6	19
D	21	28	26	5	43	64
E	22	58	71	18	67	47

```
df['X']
```

```
A     2
B     5
C    32
D     5
E    18
Name: X, dtype: int64
```

```
df['X'] % 2 == 0
```

```
A     True
B    False
C     True
D    False
E     True
Name: X, dtype: bool
```

```
df[df['X'] % 2 == 0]
```

	U	V	W	X	Y	Z
A	55	53	65	2	56	33
C	7	73	51	32	6	19
E	22	58	71	18	67	47

```
df[df['X'] % 2 == 0]['Y']
```

```
A    56
C     6
E    67
Name: Y, dtype: int64
```

```
(df['X'] % 2 == 0) & (df['X'] > 50)
```

```
A    False
B    False
C    False
D    False
```

```
E    False
Name: X, dtype: bool
```

```
df[(df['X'] % 2 == 0) & (df['X'] > 50)]
```

	U	V	W	X	Y	Z

Setting an Index

```
df
```

	U	V	W	X	Y	Z
A	55	53	65	2	56	33
B	50	26	87	5	79	44
C	7	73	51	32	6	19
D	21	28	26	5	43	64
E	22	58	71	18	67	47

```
df.reset_index()
```

	index	U	V	W	X	Y	Z
0	A	55	53	65	2	56	33
1	B	50	26	87	5	79	44
2	C	7	73	51	32	6	19
3	D	21	28	26	5	43	64
4	E	22	58	71	18	67	47

```
df.reset_index(inplace=True)
```

```
df
```

	index	U	V	W	X	Y	Z
0	A	55	53	65	2	56	33
1	B	50	26	87	5	79	44
2	C	7	73	51	32	6	19
3	D	21	28	26	5	43	64
4	E	22	58	71	18	67	47

```
df['States'] = "PB RJ DL CHD J&K".split()
```

```
"PB RJ DL CHD J&K".split()
```

```
['PB', 'RJ', 'DL', 'CHD', 'J&K']
```

```
df
```

	index	U	V	W	X	Y	Z	States
0	A	55	53	65	2	56	33	PB
1	B	50	26	87	5	79	44	RJ
2	C	7	73	51	32	6	19	DL
3	D	21	28	26	5	43	64	CHD
4	E	22	58	71	18	67	47	J&K

```
df.set_index('States')
```

States	index	U	V	W	X	Y	Z
PB	A	55	53	65	2	56	33
RJ	B	50	26	87	5	79	44
DL	C	7	73	51	32	6	19
CHD	D	21	28	26	5	43	64
J&K	E	22	58	71	18	67	47

Missing Values

```
d = {"A":[1, 2, 3, np.nan],
    "B":[5, np.nan, np.nan, np.nan],
    "C":[10, 20, 30, 40],
    "D":[np.nan, np.nan, np.nan, np.nan]}

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

	A	B	C	D
0	1.0	5.0	10	NaN
1	2.0	NaN	20	NaN
2	3.0	NaN	30	NaN
3	NaN	NaN	40	NaN

```
df.isnull()
```

	A	B	C	D
0	False	False	False	True
1	False	True	False	True
2	False	True	False	True
3	True	True	False	True

```
df.isnull().sum()
```

```
A    1
B    3
C    0
D    4
dtype: int64
```

```
df.dropna(axis=1)
```

	C
0	10
1	20
2	30
3	40

```
df.dropna(axis=1, thresh=2)
```

	A	C
0	1.0	10
1	2.0	20
2	3.0	30
3	NaN	40

```
df
```

	A	B	C	D
0	1.0	5.0	10	NaN
1	2.0	NaN	20	NaN
2	3.0	NaN	30	NaN
3	NaN	NaN	40	NaN

```
df.fillna("FILL")
```

	A	B	C	D
0	1.0	5.0	10	FILL
1	2.0	FILL	20	FILL
2	3.0	FILL	30	FILL
3	FILL	FILL	40	FILL

```
df.fillna(df.mean())
```

	A	B	C	D
0	1.0	5.0	10	NaN
1	2.0	5.0	20	NaN
2	3.0	5.0	30	NaN
3	2.0	5.0	40	NaN

```
df.fillna(0)
```

	A	B	C	D
0	1.0	5.0	10	0.0
1	2.0	0.0	20	0.0
2	3.0	0.0	30	0.0
3	0.0	0.0	40	0.0

Grouping

```
d = {"Company":["FB", "GOOGLE", "MICROSOFT", "FB", "GOOGLE", "FB", "MICROSOFT", "FB"],
    "Employee":["Sam", "Rachel", "Maddy", "Joe", "Srishti", "Shivay", "Pushpa", "Kirti"],
    "Sales":[1000, 500, 550, 2000, 890, 500, 350, 350]}

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

	Company	Employee	Sales
0	FB	Sam	1000
1	GOOGLE	Rachel	500
2	MICROSOFT	Maddy	550
3	FB	Joe	2000
4	GOOGLE	Srishti	890
5	FB	Shivay	500
6	MICROSOFT	Pushpa	350
7	FB	Kirti	350

```
df.min()
```

```
Company      FB
Employee    Joe
Sales       350
dtype: object
```

```
df.max()
```

```
Company     MICROSOFT
Employee      Srishti
Sales            2000
dtype: object
```

```
grouped_df= df.groupby('Company')
grouped_df
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7fee3c692df0>
```

```
grouped_df.min()
```

Company	Employee	Sales
FB	Joe	350
GOOGLE	Rachel	500
MICROSOFT	Maddy	350

```
grouped_df.max()
```

Company	Employee	Sales
FB	Shivay	2000
GOOGLE	Srishti	890
MICROSOFT	Pushpa	550

```
grouped_df.describe()
```

	Sales							
	count	mean	std	min	25%	50%	75%	max
Company								
FB	4.0	962.5	745.402576	350.0	462.5	750.0	1250.0	2000.0
GOOGLE	2.0	695.0	275.771645	500.0	597.5	695.0	792.5	890.0
MICROSOFT	2.0	450.0	141.421356	350.0	400.0	450.0	500.0	550.0

```
df.describe()
```

	Sales
count	8.000000
mean	767.500000
std	551.251822
min	350.000000
25%	462.500000
50%	525.000000
75%	917.500000
max	2000.000000

pandas Exercise :

Import pandas

```
import pandas as pd
```

Read the data from Salaries.csv and store it in a dataframe

```
df = pd.read_csv("/content/Salaries.csv")
```

```
df
```

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year	Notes	Ag
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.00	400184.25	NaN	567595.43	567595.43	2011	NaN	Fran
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	NaN	538909.28	538909.28	2011	NaN	Fran
2	3	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)	212739.13	106088.18	16452.60	NaN	335279.91	335279.91	2011	NaN	Fran
3	4	CHRISTOPHER CHONG	WIRE ROPE CABLE MAINTENANCE MECHANIC	77916.00	56120.71	198306.90	NaN	332343.61	332343.61	2011	NaN	Fran
4	5	PATRICK GARDNER	DEPUTY CHIEF OF DEPARTMENT, (FIRE DEPARTMENT)	134401.60	9737.00	182234.59	NaN	326373.19	326373.19	2011	NaN	Fran
...	...	...	...	...	...	...	...	...	...	...	...	...
148649	148650	Roy I Tillery	Custodian	0.00	0.00	0.00	0.0	0.00	0.00	2014	NaN	Fran
148650	148651	Not provided	Not provided	NaN	NaN	NaN	NaN	0.00	0.00	2014	NaN	Fran

Check if the dataframe is properly read or not using the head function

```
df.head()
```

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	Total
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.00	400184.25	NaN	56759
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	NaN	53890
2	3	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)	212739.13	106088.18	16452.60	NaN	33527

What columns exist in this dataframe?

```
df.columns
```

```
Index(['Id', 'EmployeeName', 'JobTitle', 'BasePay', 'OvertimePay', 'OtherPay',
       'Benefits', 'TotalPay', 'TotalPayBenefits', 'Year', 'Notes', 'Agency',
       'Status'],
      dtype='object')
```

How many rows does this dataframe have?

```
df.index
```

```
RangeIndex(start=0, stop=148654, step=1)
```

Display the information about the dataframe using the info function. Which of these columns have missing values in them?

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148654 entries, 0 to 148653
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Id                148654 non-null  int64  
 1   EmployeeName      148654 non-null  object 
 2   JobTitle          148654 non-null  object 
 3   BasePay           148045 non-null  float64
 4   OvertimePay       148650 non-null  float64
 5   OtherPay          148650 non-null  float64
 6   Benefits          112491 non-null  float64
 7   TotalPay          148654 non-null  float64
 8   TotalPayBenefits  148654 non-null  float64
 9   Year              148654 non-null  int64  
 10  Notes             0 non-null       float64
 11  Agency            148654 non-null  object 
 12  Status            0 non-null       float64
dtypes: float64(8), int64(2), object(3)
memory usage: 14.7+ MB
```

What is the total BasePay?

```
df["BasePay"].mean()
```

```
66325.4488404877
```

What is the highest amount of overtime pay?

```
df["OvertimePay"].max()
```

```
245131.88
```

What is the job title of JOSEPH DRISCOLL ? Note: Use all caps, otherwise you may get an answer that doesn't match up (there is also a lowercase Joseph Driscoll).

```
df[df["EmployeeName"] == "JOSEPH DRISCOLL"]["JobTitle"]
```

```
24    CAPTAIN, FIRE SUPPRESSION
Name: JobTitle, dtype: object
```

How much does JOSEPH DRISCOLL make (including benefits)?

```
df[df["EmployeeName"] == "JOSEPH DRISCOLL"]["TotalPayBenefits"]
```

```
24    270324.91
Name: TotalPayBenefits, dtype: float64
```

What is the name of highest paid person (including benefits)?

```
df[df['TotalPayBenefits']== df['TotalPayBenefits'].max()]
```

Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalP
		GENERAL MANAGER-					

What was the average (mean) BasePay of all employees per year? (2011-2014) ?

```
df.groupby('Year')['BasePay'].mean()
```

```
Year
2011    63595.956517
2012    65436.406857
2013    69630.030216
2014    66564.421924
Name: BasePay, dtype: float64
```

How many unique job titles exist in the dataframe?

```
df["JobTitle"].nunique()
```

```
2159
```

What is the name of lowest paid person (including benefits)? Do you notice something strange about how much he or she is paid?