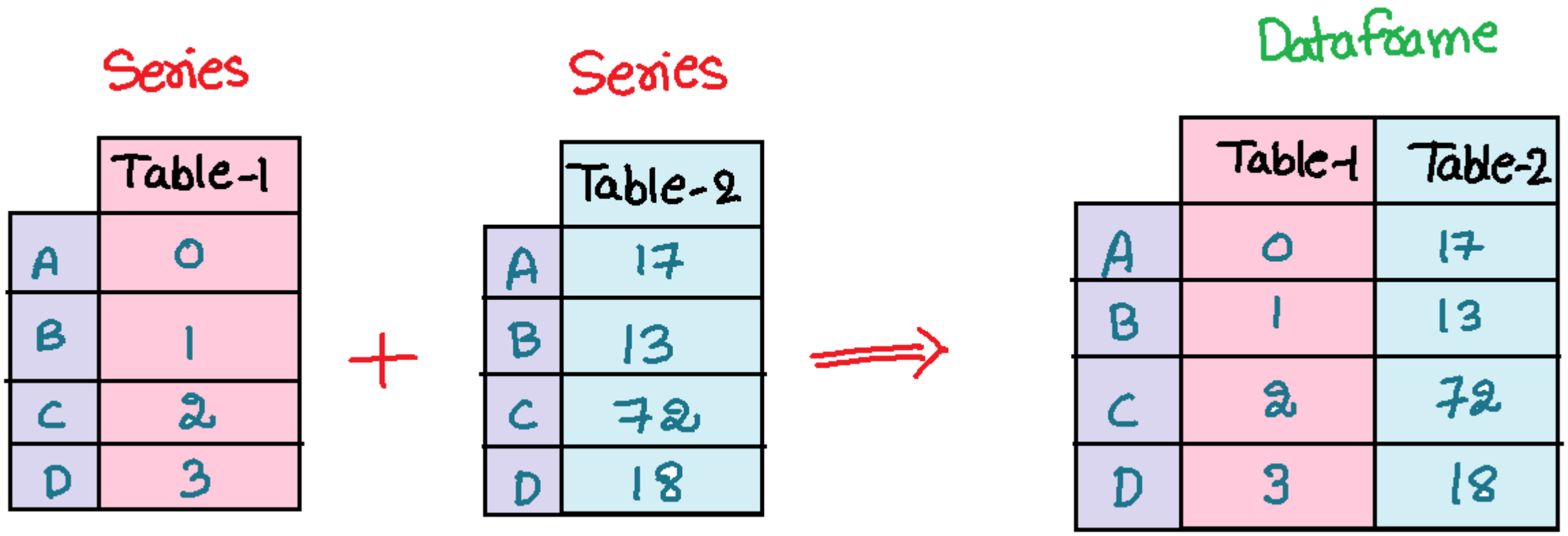
Series & Dataframe

- Series: 1D, DF : 2D
- Indices can be Labelled, not just integers
- Multiple Data types supported.



Create DF

	Tech1	Infra1	Energy1	Group	Defense1
Tata1	131	163	105	G5	193
Reliance1	192	117	174	G5	144
Wipro1	125	139	115	G10	154
Birla1	100	174	176	G5	137
Adani1	133	181	150	G20	136
Brittania1	166	105	195	G5	119
Unilever1	192	121	165	G5	115
Mahindra1	158	163	108	G10	165
Marico1	174	133	109	G20	130

Concatenation

Table-1

	A	B	C	D
0	10	20	30	40
1	11	21	31	41
2	12	22	32	42
3	13	23	33	43

+

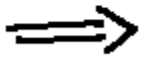
Table-2

	A	B	C	D
4	14	24	34	44
5	15	25	35	45
6	16	26	36	46
7	17	27	37	47

+

Table 3

	A	B	C	D
8	18	28	38	48
9	19	29	39	49

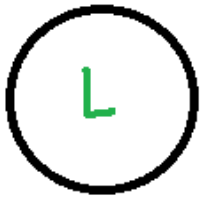


Pd.Concat(T_1, T_2, T_3)

	A	B	C	D
0	10	20	30	40
1	11	21	31	41
2	12	22	32	42
3	13	23	33	43
4	14	24	34	44
5	15	25	35	45
6	16	26	36	46
7	17	27	37	47
8	18	28	38	48
9	19	29	39	49

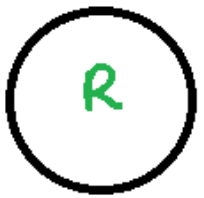
Merge Inner

Left table



Key	Left
0	a
1	b
2	c
3	d
4	e

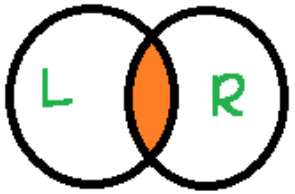
Right Table



Key	Right
2	f
3	g
4	h
5	i
6	j

maching the data

Merge. inner Join



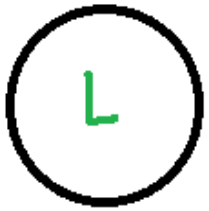
key	left	right
2	c	f
3	d	g
4	e	h

" inner join Return only the rows in which the left table have matching keys in the right table "

```
pd.merge(Left, right, on='key', how='inner')
```

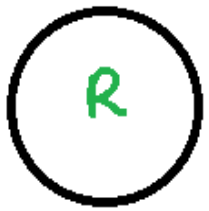
Merge Outer

Left table



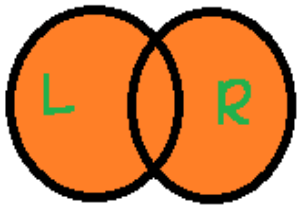
Key	Left
0	a
1	b
2	c
3	d
4	e

Right Table



Key	Right
2	f
3	g
4	h
5	i
6	j

Merge. Outer Join



key	left	right
0	a	NaN
1	b	NaN
2	c	f
3	d	g
4	e	h
5	NaN	i
6	NaN	j

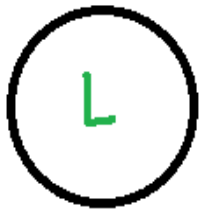
" Outer join returns all rows from both tables join records from the Left which have maching keys in the right table

```
pd.merge(Left, Right, on='key', how='outer')
```

↓ table name ↓ column name ↘ join type

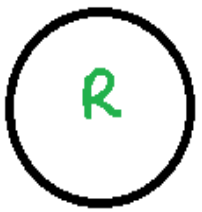
Merge on Right

Left table



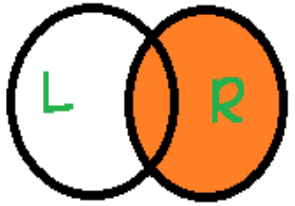
Key	Left
0	a
1	b
2	c
3	d
4	e

Right Table



Key	Right
2	f
3	g
4	h
5	i
6	j

Merge. rightJoin



key	left	right
2	a	f
3	b	g
4	e	h
5	NaN	i
6	NaN	j

" right join return all rows from the table, and any rows with matching keys from the Left table. "

```
pd.merge( Left, Right, on='key', how='right')
```

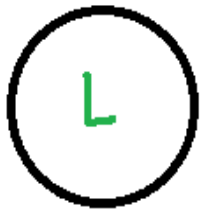
Tables

Column name

join types

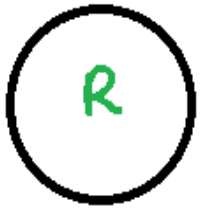
Merge on Left

Left table



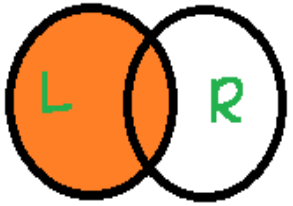
Key	Left
0	a
1	b
2	c
3	d
4	e

Right Table



Key	Right
2	f
3	g
4	h
5	i
6	j

Merge. Left Join



key	left	right
0	a	NaN
1	b	NaN
2	c	f
3	d	g
4	e	h

" Left join all rows from the Left table, and any rows with matching keys from right table "

```
pd.merge(Left, Right, on='key', how='Left')
```

Left, Right: table name

'key': column name

'Left': join type

Stack

"Reshape using stack() function in pandas converts the data into stacked format. ie. the column stacked row wise."

Reshape Using Stack()

Dataframe

	3_Months		6_Months	
	Python	ML	Python	ML
Jony	12	45	67	56
Anil	78	89	45	67
Sunil	45	67	89	90
Rocky	67	44	56	55

Dataframe.Stack()

		3_Months	6_Months
Jony	ML	45	56
	Python	12	67
Anil	ML	89	67
	Python	78	45
Sunil	ML	67	90
	Python	45	89
Rocky	ML	44	55
	Python	67	56

Dataframe.Stack(Level=0)

Level=0

		ML	Python
Jony	3_Months	45	12
	6_Months	56	67
Anil	3_Months	89	78
	6_Months	67	45
Sunil	3_Months	67	45
	6_Months	90	89
Rocky	3_Months	44	67
	6_Months	55	58

Dataframe.Stack(Level=1)

Level=1

		3_Months	6_Months
Jony	ML	45	56
	Python	12	67
Anil	ML	89	67
	Python	78	45
Sunil	ML	67	90
	Python	45	89
Rocky	ML	44	55
	Python	67	56

Level = 1 :- It will stack ML and Python column row wise.

Level = 0 :- It will stack 3_moths, 6-moths colums row wise

Unstack

Reshape using unstack

"Unstack() function in pandas coverts the data into unstacked format."

Stacked = df.Stack()

Stacked.unstack()

Data Fraame

	3_Months		6_Months	
	Python	ML	Python	ML
Jony	12	45	67	56
Anil	78	89	45	67
Sunil	45	67	89	90
Rocky	67	44	56	55

		3_Months	6_Months
Jony	ML	45	56
	Python	12	67
Anil	ML	89	67
	Python	78	45
Sunil	ML	67	90
	Python	45	89
Rocky	ML	44	55
	Python	67	56

	3_Months		6_Months	
	ML	Python	ML	Python
Jony	45	12	56	67
Anil	89	78	67	45
Sunil	67	45	90	89
Rocky	44	67	55	56

Dataframe.unstack:

Unstack prescribed level(s) from index axis onto column axis

Melt

Melt()

"Melt() function is useful to manage a Dataframe into a format where one or more columns are identifier variables."

Dataframe = df

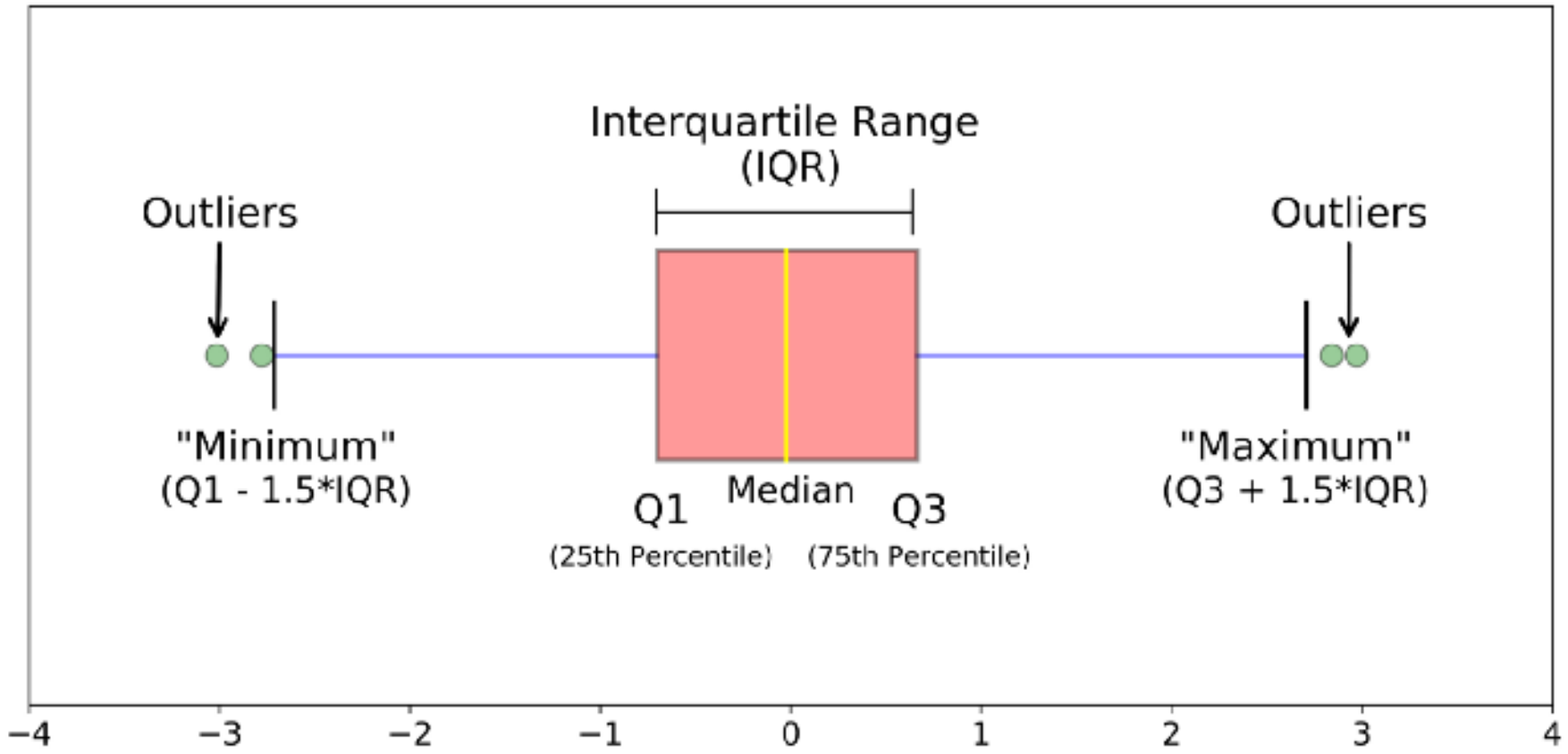
Location	Temperature	Jan-2019	Feb-2019	Mar-2019
Bangalore	Predict	30	45	24
Chennai	Actual	32	43	22

df = pd.melt(df....)

var-name = Data , value name = "Value"

Location	Temperature	Data	Value
Bangalore	Predict	Jan-2019	30
Chennai	Actual	Jan-2019	32
Bangalore	Predict	Feb-2019	45
Chennai	Actual	Feb-2019	43
Bangalore	Predict	Mar-2019	24
Chennai	Actual	Mar-2019	22

Box Plot Boundaries



Create Pandas DataFrame ?

Profit(Cr)	Food	Clothing	Telecom	IT	Infra	Energy	Defense
Tata	6300	5000	300	9900	200	4500	6500
Reliance	7500	5500	7000	300	350	6000	7500
ITC	9500	150	100	450	800	120	100
Birla	6700	8500	6500	750	560	5900	250
Adani	100	300	500	200	9900	8900	350