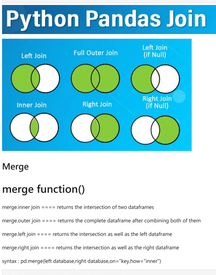
In [1]:
```
import pandas as pd
```



Merge

merge function()

merge.inner join ==== returns the intersection of two dataframes

merge.outer join ==== returns the complete dataframe after combining both of them

merge.left join ==== returns the intersection as well as the left dataframe

merge.right join ==== returns the intersection as well as the right dataframe

syntax : pd.merge(left dataframe,right dataframe,on="key,how="inner")

In [2]:
```
left_frame = pd.DataFrame({'key': range(5),
                           'left_value': ['a', 'b', 'c', 'd', 'e']})
right_frame = pd.DataFrame({'key': range(2, 7),
                            'right_value': ['f', 'g', 'h', 'i', 'j']})
print(left_frame)
print('\n')
print(right_frame)
```

```
   key left_value
0    0          a
1    1          b
2    2          c
3    3          d
4    4          e


   key right_value
0    2           f
1    3           g
2    4           h
3    5           i
4    6           j
```

Inner Join

In [3]:
```
pd.merge(left_frame, right_frame, on='key', how='inner')
```

Out[3]:

	key	left_value	right_value
0	2	c	f
1	3	d	g
2	4	e	h

Left Join

In [4]:
```
pd.merge(left_frame, right_frame, on='key', how='left')
```

Out[4]:

	key	left_value	right_value
0	0	a	NaN
1	1	b	NaN
2	2	c	f
3	3	d	g
4	4	e	h

Right Join

In [5]:
```
pd.merge(left_frame, right_frame, on='key', how='right')
```

Out[5]:

	key	left_value	right_value
0	2	c	f
1	3	d	g
2	4	e	h
3	5	NaN	i
4	6	NaN	j

Outer Join

In [6]:
```
pd.merge(left_frame, right_frame, on='key', how='outer')
```

Out[6]:

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

Merge on Index

Join can be used to combine columns of 2 dataframes that have different index values into a signle dataframe

The one difference between merge and join is that, `merge uses common columns to combine two dataframes, whereas join uses the row index to join two dataframes`

In [7]:
```
Table1 = pd.DataFrame({"Q1": ["101", "102", "103"],
                       "Q2": ["201", "202", "203"]},
                      index=["I0", "I1", "I2"])

Table2 = pd.DataFrame({"Q3": ["301", "302", "303"],
                       "Q4": ["401", "402", "403"]},
                      index=["I0", "I2", "I3"])
```

In [8]:
```
Table1
```

Out[8]:

	Q1	Q2
I0	101	201
I1	102	202
I2	103	203

In [9]:
```
Table2
```

Out[10]:

	Q3	Q4
I0	301	401
I2	302	402
I3	303	403

In [11]:
```
Table1.join(Table2)
```

Out[11]:

	Q1	Q2	Q3	Q4
I0	101	201	301	401
I1	102	202	NaN	NaN
I2	103	203	302	402

In [12]:
```
Table1.join(Table2, how='outer')
```

Out[12]:

	Q1	Q2	Q3	Q4
I0	101	201	301	401
I1	102	202	NaN	NaN
I2	103	203	302	402
I3	NaN	NaN	303	403

In [13]:
```
left_frame
```

Out[14]:

	key	left_value
0	0	a
1	1	b
2	2	c
3	3	d
4	4	e

In [15]:
```
right_frame
```

Out[15]:

	key	right_value
0	2	f
1	3	g
2	4	h
3	5	i
4	6	j

Concatenate

In [16]:
```
pd.concat([left_frame, right_frame])
```

Out[16]:

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

In [17]:
```
pd.concat([left_frame, right_frame], axis=1)
```

Out[18]:

	key	left_value	key	right_value
0	0	a	2	f
1	1	b	3	g
2	2	c	4	h
3	3	d	5	i
4	4	e	6	j

Combining

Lets combine the two dataframes

In [19]:
```
df=left_frame.append(right_frame, sort=True)
```

In [20]:
```
df
```

Out[20]:

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

Ignoring indexes on the concatenation axis

In [21]:
```
result = left_frame.append(right_frame, ignore_index=True)
```

In [22]:
```
result
```

Out[22]:

	key	left_value	right_value
0	0	a	NaN
1	1	b	NaN
2	2	c	NaN
3	3	d	NaN
4	4	e	NaN
5	2	NaN	f
6	3	NaN	g
7	4	NaN	h
8	5	NaN	i
9	6	NaN	j

In [23]:
```
df2 = {'key': 10, 'left_value': '89', 'right_value': 'test'}
df = result.append(df2, ignore_index = True)
df
```

Out[23]:

	key	left_value	right_value
0	0	a	NaN
1	1	b	NaN
2	2	c	NaN
3	3	d	NaN
4	4	e	NaN
5	2	NaN	f
6	3	NaN	g
7	4	NaN	h
8	5	NaN	i
9	6	NaN	j
10	10	89	test

Reshaping

Reshape DataFrame for better Analysis.

Lets `Reshape` the data frame using 'melt' function

In [24]:
```
df = pd.DataFrame({"Name": {0: 'John', 1: 'Bob', 2: 'Shiela'},
                   "Course": {0: 'Masters', 1: 'Graduate', 2: 'Graduate'},
                   "Age": {0: 27, 1: 23, 2: 21}})
df
```

Out[22]:

	Name	Course	Age
0	John	Masters	27
1	Bob	Graduate	23
2	Shiela	Graduate	21

In [25]:
```
# Name is id_vars and Course is value_vars
pd.melt(df, id_vars =['Name'], value_vars =['Course'])
```

Out[25]:

	Name	variable	value
0	John	Course	Masters
1	Bob	Course	Graduate
2	Shiela	Course	Graduate

In [26]:
```
# multiple unpivot columns
pd.melt(df, id_vars =['Name'], value_vars =['Course', 'Age'])
```

Out[26]:

	Name	variable	value
0	John	Course	Masters
1	Bob	Course	Graduate
2	Shiela	Course	Graduate
3	John	Age	27
4	Bob	Age	23
5	Shiela	Age	21

In [ ]:
```

```