In [23]:

```
d1 = {'key1':[1,2,3,4,5,6],
      'key2':[7,8,9,10,11,12],
      'key3':[13,14,15,16,17,18],
      'key4':[19,20,21,22,23,24]
}
```

1) The number of elements in each value should be equal...otherwise it shows an error

In [1]:

```
import pandas as pd
```

In [5]:

```
df1 = pd.DataFrame(d1)
```

In [6]:

```
df1
```

Out[6]:

```
   key1  key2  key3  key4
0     1     7    13    19
1     2     8    14    20
2     3     9    15    21
3     4    10    16    22
4     5    11    17    23
5     6    12    18    24
```

In [12]:

```
# 1) instead of number we can give any like as shown below

d2 = {'key1':[1,2,3,4,5,"Naveen"],
      'key2':[7,8,9,10,11,12],
      'key3':[13,14,15,16,17,18],
      'key4':[19,20,21,22,23,24]
}
```

In [14]:

```
df2 = pd.DataFrame(d2)
```

In [15]:

```
df2
```

Out[15]:

	key1	key2	key3	key4
0	1	7	13	19
1	2	8	14	20
2	3	9	15	21
3	4	10	16	22
4	5	11	17	23
5	Naveen	12	18	24

1) To change the index....follow as shown below

In [17]:

```
pd.DataFrame(d1,index = ['a','b','c','d','e','f'])
```

Out[17]:

	key1	key2	key3	key4
a	1	7	13	19
b	2	8	14	20
c	3	9	15	21
d	4	10	16	22
e	5	11	17	23
f	6	12	18	24

In [25]:

```
d2 = {'key1':[1,2,3,4,5,"Naveen"],
      'key2' : [7,8,9,"Thota",11,12],
      'key3':[13,14,'Data',16,17,18],
      'key4':[19,20,'Science',22,23,24]
}
```

In [22]:

```
# To represent in the form of DataFrame.. follow as shown below

pd.DataFrame(d2)
```

Out[22]:

	key1	key2	key3	key4
0	1	7	13	19
1	2	8	14	20
2	3	9	Data	Science
3	4	Thota	16	22
4	5	11	17	23
5	Naveen	12	18	24

In [24]:

```
df1 = pd.DataFrame(d1)
```

In [26]:

```
df2 = pd.DataFrame(d2)
```

In [27]:

```
df1
```

Out[27]:

	key1	key2	key3	key4
0	1	7	13	19
1	2	8	14	20
2	3	9	15	21
3	4	10	16	22
4	5	11	17	23
5	6	12	18	24

In [28]:

```
df2
```

Out[28]:

	key1	key2	key3	key4
0	1	7	13	19
1	2	8	14	20
2	3	9	Data	Science
3	4	Thota	16	22
4	5	11	17	23
5	Naveen	12	18	24

1) Now, we are going to see the merge operation of 'df1' and 'df2'. 2) By default merge perform the inner merge operation

In [29]:

```
pd.merge(df1,df2)
```

Out[29]:

	key1	key2	key3	key4
0	1	7	13	19
1	2	8	14	20
2	5	11	17	23

In [30]:

```
# Now, we will see some other parameters in merge function

pd.merge(df1,df2, how = 'inner', on = 'key1')
```

Out[30]:

	key1	key2_x	key3_x	key4_x	key2_y	key3_y	key4_y
0	1	7	13	19	7	13	19
1	2	8	14	20	8	14	20
2	3	9	15	21	9	Data	Science
3	4	10	16	22	Thota	16	22
4	5	11	17	23	11	17	23

In [34]:

```
pd.merge(df1,df2, how = 'inner', on = 'key4')
```

Out[34]:

	key1_x	key2_x	key3_x	key4	key1_y	key2_y	key3_y
0	1	7	13	19	1	7	13
1	2	8	14	20	2	8	14
2	4	10	16	22	4	Thota	16
3	5	11	17	23	5	11	17
4	6	12	18	24	Naveen	12	18

In [35]:

```
pd.merge(df1,df2, how = 'inner', on = 'key3')
```

Out[35]:

	key1_x	key2_x	key3	key4_x	key1_y	key2_y	key4_y
0	1	7	13	19	1	7	19
1	2	8	14	20	2	8	20
2	4	10	16	22	4	Thota	22
3	5	11	17	23	5	11	23
4	6	12	18	24	Naveen	12	24

In [36]:

```
# Left merge

pd.merge(df1,df2, how = 'left', on = 'key4')
```

Out[36]:

	key1_x	key2_x	key3_x	key4	key1_y	key2_y	key3_y
0	1	7	13	19	1	7	13
1	2	8	14	20	2	8	14
2	3	9	15	21	NaN	NaN	NaN
3	4	10	16	22	4	Thota	16
4	5	11	17	23	5	11	17
5	6	12	18	24	Naveen	12	18

In [37]:

```
# Right merge

pd.merge(df1,df2, how = 'right', on = 'key4')
```

```
C:\Users\navee\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:91
6: FutureWarning: In a future version, the Index constructor will not infe
r numeric dtypes when passed object-dtype sequences (matching Series behav
ior)
  key_col = Index(lvals).where(~mask_left, rvals)
```

Out[37]:

	key1_x	key2_x	key3_x	key4	key1_y	key2_y	key3_y
0	1.0	7.0	13.0	19.0	1	7	13
1	2.0	8.0	14.0	20.0	2	8	14
2	NaN	NaN	NaN	Science	3	9	Data
3	4.0	10.0	16.0	22.0	4	Thota	16
4	5.0	11.0	17.0	23.0	5	11	17
5	6.0	12.0	18.0	24.0	Naveen	12	18

outer: use union of keys from both frames, similar to a SQL full outer join; sort keys lexicographically.

In [38]:

```
pd.merge(df1,df2, how = 'outer', on = 'key4')
```

```
C:\Users\navee\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:91
6: FutureWarning: In a future version, the Index constructor will not infe
r numeric dtypes when passed object-dtype sequences (matching Series behav
ior)
  key_col = Index(lvals).where(~mask_left, rvals)
```

Out[38]:

	key1_x	key2_x	key3_x	key4	key1_y	key2_y	key3_y
0	1.0	7.0	13.0	19.0	1	7	13
1	2.0	8.0	14.0	20.0	2	8	14
2	3.0	9.0	15.0	21.0	NaN	NaN	NaN
3	4.0	10.0	16.0	22.0	4	Thota	16
4	5.0	11.0	17.0	23.0	5	11	17
5	6.0	12.0	18.0	24.0	Naveen	12	18
6	NaN	NaN	NaN	Science	3	9	Data

In [39]:

```
# Here for 'on' we can give multiple column names as shown below

pd.merge(df1,df2, how = 'outer', on = ['key4','key2'])
```

```
C:\Users\navee\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:91
6: FutureWarning: In a future version, the Index constructor will not infe
r numeric dtypes when passed object-dtype sequences (matching Series behav
ior)
  key_col = Index(lvals).where(~mask_left, rvals)
```

Out[39]:

	key1_x	key2	key3_x	key4	key1_y	key3_y
0	1.0	7.0	13.0	19.0	1	13
1	2.0	8.0	14.0	20.0	2	14
2	3.0	9.0	15.0	21.0	NaN	NaN
3	4.0	10.0	16.0	22.0	NaN	NaN
4	5.0	11.0	17.0	23.0	5	17
5	6.0	12.0	18.0	24.0	Naveen	18
6	NaN	9	NaN	Science	3	Data
7	NaN	Thota	NaN	22	4	16

1) Till now we have seen that merge operation of two tables with respect to same column names 2) Incase of different column names we have to use the parameter called 'left_on' & 'right_on' As shown below

In [40]:

```
d11 = {'a':[1,2,3,4,5,6],
      'b':[7,8,9,10,11,12],
      'c':[13,14,15,16,17,18],
      'd':[19,20,21,22,23,24]
}
```

In [41]:

```
d12 = {'key1':[1,2,3,4,5,"Naveen"],
      'key2' : [7,8,9,"Thota",11,12],
      'key3':[13,14,'Data',16,17,18],
      'key4':[19,20,'Science',22,23,24]
}
```

In [42]:

```
df11 = pd.DataFrame(d11)
```

In [43]:

```
df11
```

Out[43]:

	a	b	c	d
0	1	7	13	19
1	2	8	14	20
2	3	9	15	21
3	4	10	16	22
4	5	11	17	23
5	6	12	18	24

In [44]:

```
df12 = pd.DataFrame(d12)
```

In [45]:

```
df12
```

Out[45]:

	key1	key2	key3	key4
0	1	7	13	19
1	2	8	14	20
2	3	9	Data	Science
3	4	Thota	16	22
4	5	11	17	23
5	Naveen	12	18	24

In [46]:

```
pd.merge(df11,df12, how = 'left', left_on = 'a', right_on = 'key1')
```

Out[46]:

	a	b	c	d	key1	key2	key3	key4
0	1	7	13	19	1	7	13	19
1	2	8	14	20	2	8	14	20
2	3	9	15	21	3	9	Data	Science
3	4	10	16	22	4	Thota	16	22
4	5	11	17	23	5	11	17	23
5	6	12	18	24	NaN	NaN	NaN	NaN

Now we will start the join operations

In [57]:

```
d10 = {'key11':[1,2,3,4,5,"Naveen"],
      'key21':[7,8,9,10,11,12],
      'key31':[13,14,15,16,17,18],
      'key41':[19,20,21,22,23,24]
}
```

In [58]:

```
d20 = {'key1':[1,2,3,4,5,"Naveen"],
      'key2' : [7,8,9,"Thota",11,12],
      'key3':[13,14,'Data',16,17,18],
      'key4':[19,20,'Science',22,23,24]
}
```

In [59]:

```
df10 = pd.DataFrame(d10 , index = [1,2,3,4,5,6])
```

In [68]:

```
df20 = pd.DataFrame(d20, index = [1,2,3,4,5,'Naveen'])
```

In [69]:

```
df10
```

Out[69]:

	key11	key21	key31	key41
1	1	7	13	19
2	2	8	14	20
3	3	9	15	21
4	4	10	16	22
5	5	11	17	23
6	Naveen	12	18	24

In [62]:

```
df20
```

Out[62]:

	key1	key2	key3	key4
1	1	7	13	19
2	2	8	14	20
3	3	9	Data	Science
4	4	Thota	16	22
5	5	11	17	23
6	Naveen	12	18	24

In [70]:

```
df10.join(df20,how = 'right')
```

Out[70]:

	key11	key21	key31	key41	key1	key2	key3	key4
1	1	7.0	13.0	19.0	1	7	13	19
2	2	8.0	14.0	20.0	2	8	14	20
3	3	9.0	15.0	21.0	3	9	Data	Science
4	4	10.0	16.0	22.0	4	Thota	16	22
5	5	11.0	17.0	23.0	5	11	17	23
Naveen	NaN	NaN	NaN	NaN	Naveen	12	18	24

In [71]:

```
df10.join(df20)
```

Out[71]:

	key11	key21	key31	key41	key1	key2	key3	key4
1	1	7	13	19	1	7	13	19
2	2	8	14	20	2	8	14	20
3	3	9	15	21	3	9	Data	Science
4	4	10	16	22	4	Thota	16	22
5	5	11	17	23	5	11	17	23
6	Naveen	12	18	24	NaN	NaN	NaN	NaN

In [73]:

```
df10.join(df20, how = 'inner')
```

Out[73]:

	key11	key21	key31	key41	key1	key2	key3	key4
1	1	7	13	19	1	7	13	19
2	2	8	14	20	2	8	14	20
3	3	9	15	21	3	9	Data	Science
4	4	10	16	22	4	Thota	16	22
5	5	11	17	23	5	11	17	23

In [74]:

```
df10.join(df20, how ='outer')
```

Out[74]:

	key11	key21	key31	key41	key1	key2	key3	key4
1	1	7.0	13.0	19.0	1	7	13	19
2	2	8.0	14.0	20.0	2	8	14	20
3	3	9.0	15.0	21.0	3	9	Data	Science
4	4	10.0	16.0	22.0	4	Thota	16	22
5	5	11.0	17.0	23.0	5	11	17	23
6	Naveen	12.0	18.0	24.0	NaN	NaN	NaN	NaN
Naveen	NaN	NaN	NaN	NaN	Naveen	12	18	24

Difference between merge and join operations :-

join :-Join always try to focus on indexes.

merge :- merge will always focus on the column values.

In [75]:

```
pd.concat([df10,df20], axis = 0)
```

Out[75]:

	key11	key21	key31	key41	key1	key2	key3	key4
1	1	7.0	13.0	19.0	NaN	NaN	NaN	NaN
2	2	8.0	14.0	20.0	NaN	NaN	NaN	NaN
3	3	9.0	15.0	21.0	NaN	NaN	NaN	NaN
4	4	10.0	16.0	22.0	NaN	NaN	NaN	NaN
5	5	11.0	17.0	23.0	NaN	NaN	NaN	NaN
6	Naveen	12.0	18.0	24.0	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	1	7	13	19
2	NaN	NaN	NaN	NaN	2	8	14	20
3	NaN	NaN	NaN	NaN	3	9	Data	Science
4	NaN	NaN	NaN	NaN	4	Thota	16	22
5	NaN	NaN	NaN	NaN	5	11	17	23
Naveen	NaN	NaN	NaN	NaN	Naveen	12	18	24

In [76]:

```
pd.concat([df1,df2], axis = 0)
```

Out[76]:

```
      key1   key2  key3     key4
0        1      7    13       19
1        2      8    14       20
2        3      9    15       21
3        4     10    16       22
4        5     11    17       23
5        6     12    18       24
0        1      7    13       19
1        2      8    14       20
2        3      9  Data  Science
3        4  Thota    16       22
4        5     11    17       23
5   Naveen     12    18       24
```

In [77]:

```
pd.concat([df1,df2], axis = 1)
```

Out[77]:

```
   key1  key2  key3  key4    key1   key2  key3     key4
0     1     7    13    19       1      7    13       19
1     2     8    14    20       2      8    14       20
2     3     9    15    21       3      9  Data  Science
3     4    10    16    22       4  Thota    16       22
4     5    11    17    23       5     11    17       23
5     6    12    18    24  Naveen     12    18       24
```

In [ ]:

```

```